CodeArtifact Guía del usuario

CodeArtifact

CodeArtifact: CodeArtifact Guía del usuario

Table of Contents

¿Qué es AWS CodeArtifact?

AWS CodeArtifact es un servicio de repositorio de artefactos administrado, seguro y altamente escalable que ayuda a las organizaciones a almacenar y compartir paquetes de software para el desarrollo de aplicaciones. Puede usar CodeArtifact con herramientas de compilación y administradores de paquetes populares, como NuGet CLI, Maven, Gradle, npm, yarn, pip y twine. CodeArtifact ayuda a reducir la necesidad de administrar su propio sistema de almacenamiento de artefactos o de preocuparse por escalar su infraestructura. No hay límites en la cantidad o el tamaño total de los paquetes que puede almacenar en un repositorio de CodeArtifact.

Puede crear una conexión entre su repositorio privado de CodeArtifact y un repositorio público externo, como npmjs.com o el central de Maven. Luego, CodeArtifact buscará y almacenará los paquetes a pedido del repositorio público cuando los solicite un administrador de paquetes. Esto hace que sea más cómodo consumir las dependencias de código abierto que utiliza su aplicación y ayuda a garantizar que estén siempre disponibles para su compilación y desarrollo. También puede publicar paquetes privados en un repositorio de CodeArtifact. Esto le ayuda a compartir componentes de software propietario entre múltiples aplicaciones y equipos de desarrollo de su organización.

Para obtener más información, consulte AWS CodeArtifact.

¿Cómo funciona CodeArtifact?

CodeArtifact almacena paquetes de software en repositorios. Los repositorios son políglotas: un único repositorio puede contener paquetes de cualquier tipo compatible. Cada repositorio de CodeArtifact es miembro de un único dominio de CodeArtifact. Le recomendamos que utilice un dominio de producción para su organización con uno o más repositorios. Por ejemplo, puede utilizar cada repositorio para un equipo de desarrollo diferente. De este modo, los paquetes de sus repositorios se pueden descubrir y compartir entre sus equipos de desarrollo.

Para añadir paquetes a un repositorio, configure un administrador de paquetes como npm o Maven para usar el punto de conexión (URL) del repositorio. A continuación, puede usar el administrador de paquetes para publicar paquetes en el repositorio. También puede importar paquetes de código abierto a un repositorio configurándolo con una conexión externa a un repositorio público como npmjs, NuGet Gallery, el central de Maven o PyPI. Para obtener más información, consulte Conectar un repositorio CodeArtifact a un repositorio público.

Puede hacer que los paquetes de un repositorio estén disponibles para otro repositorio del mismo dominio. Para ello, configure un repositorio como un repositorio ascendente del otro. Todas las

versiones de paquetes disponibles en el repositorio ascendente también están disponibles en el repositorio descendente. Además, todos los paquetes que están disponibles en el repositorio ascendente a través de una conexión externa a un repositorio público están disponibles en el repositorio descendente. Para obtener más información, consulte Trabajar con repositorios ascendentes en CodeArtifact.

Conceptos de AWS CodeArtifact

Estos son algunos conceptos y términos que debe conocer cuando usa CodeArtifact.

Temas

Recurso

Un activo es un archivo individual almacenado en CodeArtifact que está asociado con una versión del paquete, como un archivo `.tgz` npm o archivos Maven POM y JAR.

Dominio

Los repositorios se agregan en una entidad de nivel superior conocida como dominio. Todos los activos y metadatos del paquete se almacenan en el dominio, pero se consumen en los repositorios. Un activo de paquete determinado, como un archivo JAR de Maven, se almacena una vez por dominio, independientemente del número de repositorios en los que esté presente. Todos los activos y metadatos de un dominio se cifran con la misma AWS KMS key (clave KMS) almacenada en AWS Key Management Service (AWS KMS).

Cada repositorio es miembro de un único dominio y no se puede mover a un dominio diferente.

Con un dominio, puede aplicar una política organizativa en varios repositorios. Con este enfoque, determina qué cuentas pueden acceder a los repositorios del dominio y qué repositorios públicos se pueden usar como fuentes de paquetes.

Aunque una organización puede tener varios dominios, recomendamos un único dominio de producción que contenga todos los artefactos publicados. De esta forma, los equipos pueden buscar y compartir paquetes en toda la organización.

Repositorio

Un repositorio de CodeArtifact contiene un conjunto de versiones de paquetes, cada una de las cuales se asigna a un conjunto de activos. Los repositorios son políglotas: un único repositorio puede contener paquetes de cualquier tipo compatible. Cada repositorio expone puntos de conexión para obtener y publicar paquetes mediante herramientas como la CLI nuget, la CLI npm, la CLI de Maven (mvn) y pip. Puede crear hasta 1000 repositorios por dominio.

Paquete

Un paquete es un paquete de software y los metadatos necesarios para resolver las dependencias e instalar el software. En CodeArtifact, un paquete consta de un nombre de paquete, un espacio de nombres opcional como `@types` en `@types/node`, un conjunto de versiones del paquete y metadatos a nivel de paquete, como las etiquetas npm.

AWS CodeArtifact admite los formatos de paquetes npm, PyPI, Maven, NuGet y generic.

Espacio de nombres de paquetes

Algunos formatos de paquetes admiten nombres de paquetes jerárquicos para organizar los paquetes en grupos lógicos y evitar colisiones de nombres. Por ejemplo, npm admite alcances. Para obtener más información, consulte la documentación de alcances de npm. El paquete npm `@types/node` tiene un alcance `@types` y un nombre de `node`. Hay muchos otros nombres de paquetes en el alcance `@types`. En CodeArtifact, el alcance («tipos») se denomina espacio de nombres del paquete y el nombre («nodo») se denomina nombre del paquete. En el caso de los paquetes de Maven, el espacio de nombres del paquete corresponde al ID de grupo de Maven. El paquete Maven `org.apache.logging.log4j:log4j` tiene un groupID (espacio de nombres del paquete) de `org.apache.logging.log4j` y un artifactID (nombre del paquete) `log4j`. Para los paquetes genéricos, se requiere un espacio de nombres. Algunos formatos de paquetes, como PyPI, no admiten nombres jerárquicos con un concepto similar al alcance de npm o al ID de grupo de Maven.

Sin una forma de agrupar los nombres de los paquetes, puede resultar más difícil evitar las colisiones de nombres.

Versión de paquete

La versión de un paquete identifica la versión específica de un paquete, por ejemplo `@types/node 12.6.9`. El formato y la semántica del número de versión varían según los distintos formatos de paquete. Por ejemplo, las versiones del paquete npm deben cumplir con la especificación de control de versiones semántico. En CodeArtifact, la versión de un paquete consta del identificador de la versión, los metadatos a nivel de versión del paquete y un conjunto de activos.

Revisión de la versión del paquete

La revisión de la versión de un paquete es una cadena que identifica un conjunto específico de activos y metadatos para una versión de paquete. Cada vez que se actualiza la versión de un paquete, se crea una nueva revisión de la versión del paquete. Por ejemplo, puede publicar un archivo de distribución de código fuente (sdist) para una versión de paquete de Python y, posteriormente, añadir una rueda de Python que contenga código compilado a la misma versión. Al publicar la rueda, se crea una nueva revisión de la versión del paquete.

Repositorio ascendente

Un repositorio es ascendente de otro cuando se puede acceder a las versiones de los paquetes que contiene desde el punto de conexión del repositorio descendente. Este enfoque fusiona de manera efectiva el contenido de los dos repositorios desde el punto de vista de un cliente. Con CodeArtifact, puede crear una relación ascendente entre dos repositorios.

¿Cómo empiezo a utilizar CodeArtifact?

Le recomendamos que siga los pasos que se describen a continuación:

1. Obtenga más información sobre CodeArtifact leyendo Conceptos de AWS CodeArtifact.
2. Configure su Cuenta de AWS, la AWS CLI y un usuario de IAM siguiendo los pasos que se indican en Configuración con AWS CodeArtifact.
3. Use CodeArtifact siguiendo las instrucciones de Introducción a CodeArtifact.

Configuración con AWS CodeArtifact

Si ya se ha inscrito en Amazon Web Services (AWS), puede comenzar a utilizar AWS CodeArtifact de inmediato. Puede abrir la consola CodeArtifact, elegir Crear un dominio y un repositorio y seguir los pasos del asistente de inicio para crear su primer dominio y repositorio.

Si aún no se ha registrado en AWS o necesita ayuda para crear su primer dominio y repositorio, complete las siguientes tareas para configurar el uso de CodeArtifact:

Temas

Registrarse en AWS

Cuando se registre en Amazon Web Services (AWS), solo se le cobrarán los servicios y recursos que utilice, incluido AWS CodeArtifact.

Si ya dispone de una Cuenta de AWS, pase a la siguiente tarea, Instale o actualice y, a continuación, configure la AWS CLI. Si no dispone de una Cuenta de AWS, utilice el siguiente procedimiento para crear una.

Para crear una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Al registrarse en una Cuenta de AWS, se crea un Usuario raíz de la cuenta de AWS. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario administrativo y utilice únicamente el usuario raíz para realizar tareas que requieran acceso de usuario raíz.

Instale o actualice y, a continuación, configure la AWS CLI

Debe instalar la AWS CLI para llamar a los comandos de CodeArtifact desde la AWS Command Line Interface (AWS CLI) en un equipo de desarrollo local.

Si tiene instalada una versión anterior de la AWS CLI, debe actualizarla para que estén disponibles los comandos de CodeArtifact. Los comandos CodeArtifact están disponibles en las siguientes versiones de AWS CLI:

1. AWS CLI1:1.18.77 y versiones posteriores
2. AWS CLI2:2.0.21 y versiones posteriores

Para comprobar la versión, utilice el comando `aws --version`.

Para instalar y configurar la AWS CLI

1. Instale o actualice la AWS CLI con las instrucciones en Instalación de la AWS Command Line Interface.
2. Configure la AWS CLI mediante el comando configure tal y como se indica a continuación.

   ```
   aws configure
   ```

   Cuando se le solicite, especifique las claves de acceso de AWS y la clave de acceso secreta de AWS del usuario de IAM que usará con CodeArtifact. Cuando se le solicite el nombre predeterminado de la Región de AWS, especifique la región en la que creará la canalización, como `us-east-2`. Cuando se le pregunte el formato de salida predeterminado, indique `json`.

   > **Important**
   >
   > Cuando configure la AWS CLI, se le pedirá que especifique una Región de AWS. Especifique una de las regiones admitidas que se encuentran en la lista de Regiones y puntos de enlace de la Referencia general de AWS.

   Para obtener más información, consulte Configuración de AWS Command Line Interface y Administración de claves de acceso para usuarios de IAM.
3. Para verificar la instalación o actualización, llame al siguiente comando desde la AWS CLI.

```
aws codeartifact help
```

Si se ejecuta correctamente, este comando muestra una lista de los comandos de CodeArtifact disponibles.

A continuación, puede crear un usuario de IAM y otorgarle acceso a CodeArtifact. Para obtener más información, consulte Aprovisionar un usuario de IAM.

Aprovisionar un usuario de IAM

Siga estas instrucciones para preparar a un usuario de IAM para utilizar CodeArtifact.

Para aprovisionar a un usuario de IAM

1. Cree un usuario de IAM o use uno asociado a su Cuenta de AWS. Para obtener más información, consulte Creación de un usuario de IAM y Descripción general de las políticas de IAM AWS en la Guía del usuario de IAM.
2. Conceda al usuario de IAM acceso a CodeArtifact.

   - Opción 1: crear una política de IAM personalizada. Con una política de IAM personalizada, puede proporcionar los permisos mínimos necesarios y cambiar la duración de los tokens de autenticación. Para obtener más información y políticas de ejemplo, consulte Ejemplos de políticas basadas en la identidad para AWS CodeArtifact.
   - Opción 2: utilice la política `AWSCodeArtifactAdminAccess` AWS gestionada. El siguiente fragmento muestra el contenido de esta política.

     Important

     Esta política otorga acceso a todas las API de CodeArtifact. Le recomendamos que utilice siempre los permisos mínimos necesarios para realizar la tarea. Para obtener más información, consulte Prácticas recomendadas de IAM en la Guía del usuario de IAM.

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

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

El permiso `sts:GetServiceBearerToken` es necesario para llamar a la API CodeArtifact `GetAuthorizationToken`. Esta API devuelve un token que debe usarse cuando se usa un administrador de paquetes como `npm` o `pip` con CodeArtifact. Para usar un administrador de paquetes con un repositorio de CodeArtifact, su usuario o rol de IAM debe permitirlo `sts:GetServiceBearerToken`, como se muestra en el ejemplo de política anterior.

Si no ha instalado el administrador de paquetes o la herramienta de compilación que planea usar con CodeArtifact, consulte Instale su gestor de paquetes o herramienta de compilación.

Instale su gestor de paquetes o herramienta de compilación

Para publicar o consumir paquetes de CodeArtifact, debe usar un administrador de paquetes. Hay diferentes administradores de paquetes para cada tipo de paquete. La siguiente lista contiene algunos administradores de paquetes que puede usar con CodeArtifact. Si aún no lo ha hecho, instale los administradores de paquetes para el tipo de paquete que desee usar.

- Para npm, utilice la CLI npm o pnpm.
- Para Maven, utilice Apache Maven (`mvn`) o Gradle.
- Para Python, utilice pip para instalar paquetes y twine para publicar paquetes.

- Para NuGet, utilice Toolkit for Visual Studio en Visual Studio o las CLI nuget o dotnet.
- En el caso de los paquetes genéricos, utilice la AWS CLI o el SDK para publicar y descargar el contenido del paquete.

Siguientes pasos

Los siguientes pasos dependerán del tipo o tipos de paquetes que utilice con CodeArtifact y del estado de sus recursos de CodeArtifact.

Si está comenzando a utilizar CodeArtifact por primera vez para usted, su equipo u organización, consulte la siguiente documentación para obtener información general sobre cómo comenzar y ayuda para crear los recursos que necesitará.

- Primeros pasos con la consola
- Primeros pasos con AWS CLI

Si sus recursos ya se han creado y está listo para configurar su administrador de paquetes para enviar paquetes o instalar paquetes desde un repositorio de CodeArtifact, consulte la documentación correspondiente a su tipo de paquete o administrador de paquetes.

- Uso de CodeArtifact con npm
- Uso de CodeArtifact con Python
- Uso de CodeArtifact con Maven
- Uso de CodeArtifact con NuGet
- Uso de CodeArtifact con paquetes genéricos

Introducción a CodeArtifact

En este tutorial de Introducción, use CodeArtifact para crear lo siguiente:

- Un dominio llamado `my-domain`.
- Un repositorio llamado `my-repo` así está contenido en `my-domain`.
- Un repositorio llamado `npm-store` así está contenido en `my-domain`. La `npm-store` tiene una conexión externa al repositorio público de npm. Esta conexión se utiliza para introducir un paquete npm en el repositorio `my-repo`.

Antes de comenzar este tutorial, se recomienda revisar CodeArtifact Conceptos de AWS CodeArtifact.

> **Note**
>
> Este tutorial requiere que cree recursos de que podrían dar lugar a cargos en su cuenta de AWS. Para obtener más información, consulte Precios de CodeArtifact.

Temas

Requisitos previos

Puede completar esta tarea mediante la AWS Management Console o la AWS Command Line Interface (AWS CLI). Para seguir el tutorial, primero debe completar los siguientes requisitos previos:

- Realice los pasos que se indican en Configuración con AWS CodeArtifact.
- Instale la CLI de npm. Para obtener más información, consulte Descarga e instalación de Node.js y npm en la documentación de npm.

Primeros pasos con la consola

Siga estos pasos para comenzar a utilizar CodeArtifact en AWS Management Console. Esta guía usa el administrador de paquetes `npm`. Si está utilizando un administrador de paquetes diferente, deberá modificar algunos de los siguientes pasos.

1. Inicie sesión en la AWS Management Console y abra la consola AWS CodeArtifact en https://console.aws.amazon.com/codesuite/codeartifact/start. Para obtener más información, consulte Configuración con AWS CodeArtifact.
2. Elija Create repository.
3. En Nombre del repositorio, introduzca **`my-repo`**.
4. (Opcional) En Descripción del repositorio, introduzca una descripción opcional para su repositorio.
5. En los repositorios públicos ascendentes, seleccione npm-store para crear un repositorio conectado a npmjs que esté situado aguas arriba de su repositorio `my-repo`.

   CodeArtifact le asigna el nombre `npm-store` a este repositorio. Todos los paquetes disponibles en el repositorio `npm-store` ascendente también están disponibles en su repositorio descendente, `my-repo`.
6. Elija Next (Siguiente).
7. En la cuenta de AWS, seleccione Esta cuenta de AWS.
8. En Nombre de dominio, introduzca **`my-domain`**.
9. Expanda Additional configuration (Configuración adicional).
10. Debe usar una AWS KMS key (clave KMS) para cifrar todos los activos de su dominio. Puede usar una clave Clave administrada de AWS o una clave KMS que administre:

    - Elija la clave administrada de AWS si quiere usar la Clave administrada de AWS predeterminada.
    - Elija la clave administrada por el cliente si quiere usar una clave de KMS que administre. Para usar una clave de KMS que administre, en ARN de clave administrada por el cliente, busque y elija la clave de KMS.

    Para obtener más información, consulte Clave administrada de AWS y las claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service.
11. Elija Next (Siguiente).

12. En Revisar y crear, revise lo que CodeArtifact está creando para usted.

    - El flujo de paquetes muestra cómo `my-domain`, `my-repo` y `npm-store` están relacionados.
    - Paso 1: Crear un repositorio muestra detalles sobre `my-repo` y `npm-store`.
    - Paso 2: El dominio seleccionado muestra detalles sobre `my-domain`.

    Cuando haya terminado, elija Crear repositorio.

13. En la página my-repo, seleccione Ver instrucciones de conexión y, a continuación, seleccione npm.

14. Use la AWS CLI para ejecutar el comando `login` que se muestra en Configure su cliente npm con este comando AWS CLI CodeArtifact.

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

    Debería recibir un resultado que confirme que su inicio de sesión se realizó correctamente.

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

    Si recibe el error `Could not connect to the endpoint URL`, asegúrese de que su AWS CLI está configurada y de que el nombre de región predeterminado es la misma región en la que creó el repositorio (consulte Configuración de la interfaz de línea de comandos de AWS).

    Para obtener más información, consultar Configurar y usar npm con CodeArtifact

15. Utilice la CLI de npm para instalar un paquete npm. Por ejemplo, para instalar el popular `lodash` del paquete npm, utilice el siguiente comando.

    ```
    npm install lodash
    ```

16. Vuelva a la consola de CodeArtifact. Si su repositorio my-repo está abierto, actualice la página. De lo contrario, en el panel de navegación, seleccione Repositorios y, a continuación, elija my-repo.

En Paquetes, debería ver la biblioteca o paquete npm que ha instalado. Puede elegir el nombre del paquete para ver su versión y estado. Puede elegir su última versión para ver los detalles del paquete, como las dependencias, los activos y más.

> **Note**
>
> Es posible que haya un retraso entre la instalación del paquete y el momento en que se introduce en el repositorio.

17. Para evitar cargos AWS adicionales, elimine los recursos utilizados en este tutorial:

> **Note**
>
> No puede eliminar un dominio que contenga repositorios, por lo que debe eliminar `my-repo` y `npm-store` antes de eliminar `my-domain`.

    a. En el panel de navegación, elija Repositorios.
    b. Elija npm-store, elija Eliminar y, a continuación, siga los pasos para eliminar el repositorio.
    c. Elija my-repo, elija Eliminar y, a continuación, siga los pasos para eliminar el repositorio.
    d. En el panel de navegación, seleccione Dominios.
    e. Elija mydomain, seleccione Eliminar y, a continuación, siga los pasos para eliminar el dominio.

Primeros pasos con AWS CLI

Siga estos pasos para comenzar a utilizar CodeArtifact usando la AWS Command Line Interface (AWS CLI). Para obtener más información, consulte Instale o actualice y, a continuación, configure la AWS CLI. Esta guía usa el administrador de paquetes `npm`. Si está utilizando un administrador de paquetes diferente, deberá modificar algunos de los siguientes pasos.

1. Utilice la AWS CLI para ejecutar el comando create-domain.

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

    Los datos con formato JSON aparecen en la salida con detalles sobre tu nuevo dominio.

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

Si recibe el error `Could not connect to the endpoint URL`, asegúrese de que su AWS CLI está configurada y de que el nombre de región predeterminado es la misma región en la que creó el repositorio (consulte Configuración de la interfaz de línea de comandos de AWS).

2. Utilice el comando create-repository para crear un repositorio en su dominio.

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

Los datos con formato JSON aparecen en la salida con detalles sobre el nuevo repositorio.

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

3. Usa el comando create-repository para crear un repositorio ascendente para su repositorio `my-repo`.

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

Los datos con formato JSON aparecen en la salida con detalles sobre el nuevo repositorio.

```
{
    "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. Use el comando associate-external-connection para añadir a su repositorio una conexión externa al repositorio público npm a su repositorio `npm-store`.

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

Los datos con formato JSON aparecen en la salida con detalles sobre el repositorio y su nueva conexión externa.

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

Para obtener más información, consulte Conectar un repositorio CodeArtifact a un repositorio público.

5. Utilice el comando update-repository para asociar el repositorio `npm-store` como repositorio ascendente al repositorio `my-repo`.

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

Los datos con formato JSON aparecen en la salida con detalles sobre el repositorio actualizado, incluido su nuevo repositorio ascendente.

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

Para obtener más información, consulte Añadir o eliminar repositorios ascendentes (AWS CLI).

6. Use el comando login para configurar su administrador de paquetes npm con su repositorio `my-repo`.

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

Debería recibir un resultado que confirme que su inicio de sesión se realizó correctamente.

```
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
```

Para obtener más información, consulte Configurar y usar npm con CodeArtifact.

7. Utilice la CLI de npm para instalar un paquete npm. Por ejemplo, para instalar el popular `lodash` del paquete npm, utilice el siguiente comando.

```
npm install lodash
```

8. Use el comando list-packages para ver el paquete que acaba de instalar en su repositorio `my-repo`.

> **Note**
>
> Puede haber un retraso entre el momento en que se completa el comando de instalación `npm install` y el momento en que el paquete está visible en el repositorio. Para obtener más información sobre la latencia típica al recuperar paquetes de repositorios públicos, consulte Latencia de conexión externa.

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

Los datos con formato JSON aparecen en la salida con el formato y el nombre del paquete que ha instalado.

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

```
    ]
}
```

Ahora tiene tres recursos de CodeArtifact:

- El dominio `my-domain`.
- El repositorio `my-repo` que se encuentra en `my-domain`. Este repositorio tiene un paquete npm disponible.
- El repositorio `npm-store` que se encuentra en `my-domain`. Este repositorio tiene una conexión externa al repositorio público de npm y está asociado como un repositorio ascendente con el repositorio `my-repo`.

9. Para evitar cargos AWS adicionales, elimine los recursos utilizados en este tutorial:

> **Note**
>
> No puede eliminar un dominio que contenga repositorios, por lo que debe eliminar `my-repo` y `npm-store` antes de eliminar `my-domain`.

a. Utilice el comando delete-repository para eliminar el repositorio `npm-store`.

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

Los datos con formato JSON aparecen en la salida con detalles sobre el repositorio eliminado.

```
{
    "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. Utilice el comando delete-repository para eliminar el repositorio `npm-store`.

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

Los datos con formato JSON aparecen en la salida con detalles sobre el repositorio eliminado.

```
{
    "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. Utilice el comando delete-domain para eliminar el repositorio `my-domain`.

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

Los datos con formato JSON aparecen en la salida con detalles sobre el dominio eliminado.

```
{
    "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
    }
}
```

Trabajar con repositorios en CodeArtifact

En estos temas se muestra cómo usar la CodeArtifact consola y CodeArtifact las API para crear, enumerar, actualizar y eliminar repositorios. AWS CLI

Temas

Crear un repositorio de

Como todos los paquetes CodeArtifact están almacenados en repositorios, para poder usarlos CodeArtifact, debe crear uno. Puede crear un repositorio mediante la CodeArtifact consola, el AWS Command Line Interface (AWS CLI) o AWS CloudFormation. Cada repositorio está asociado a la AWS cuenta que usaste al crearlo. Puede tener varios repositorios, que se crean y agrupan en dominios. Al crear un repositorio, no contiene ningún paquete. Los repositorios son políglotas, lo que significa que un único repositorio puede contener paquetes de cualquier tipo compatible.

Para obtener información sobre los límites de CodeArtifact servicio, como el número máximo de repositorios permitidos en un solo dominio, consulteCuotas en AWS CodeArtifact. Si alcanza el número máximo de repositorios permitidos, puede eliminar repositorios para dejar espacio para más.

Un repositorio puede tener uno o más CodeArtifact repositorios asociados como repositorios ascendentes. Esto permite a un cliente administrador de paquetes acceder a los paquetes contenidos en más de un repositorio mediante un único punto de conexión de URL. Para obtener más información, consulte Trabajar con repositorios ascendentes en CodeArtifact.

Para obtener más información sobre la administración de CodeArtifact repositorios con, consulte. CloudFormation Creación de recursos de CodeArtifact con AWS CloudFormation

> **Note**
>
> Después de haber creado un repositorio, no puede cambiar su nombre, cuenta de AWS asociada ni dominio.

Temas

Crear un repositorio (consola)

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Repositorios y luego elija Crear repositorio.
3. En Nombre del repositorio, introduzca un nombre para el repositorio.
4. (Opcional) En Descripción del repositorio, introduzca una descripción opcional para su repositorio.
5. (Opcional) En Publicar repositorios originales, añada repositorios intermedios que conecten sus repositorios con autoridades de paquetes, como Maven Central o npmjs.com.
6. Elija Siguiente.
7. En Cuenta de AWS, elija Esta cuenta de AWS si ha iniciado sesión en la cuenta propietaria del dominio. Elija Cuenta de AWS diferente si el dominio es propiedad de otra cuenta de AWS.
8. En Dominio, elija el dominio en el que se creará el repositorio.

   Si no hay ningún dominio en la cuenta, debe crear uno. Introduzca el nombre del nuevo dominio en Nombre de dominio.

   Expanda Configuración adicional.

   Debe usar una AWS KMS key (clave KMS) para cifrar todos los activos de su dominio. Puedes usar una clave Clave administrada de AWS o una clave KMS que administres:

> **Important**
>
> CodeArtifact solo admite claves KMS simétricas. No puede utilizar una clave KMS asimétrica para cifrar sus dominios. CodeArtifact Para obtener ayuda para determinar si una clave de KMS es simétrica o asimétrica, consulte Identificación de claves de KMS simétricas y asimétricas.

- Elija la clave administrada de AWS si quiere usar la Clave administrada de AWS predeterminada.
- Elija la clave administrada por el cliente si quiere usar una clave de KMS que administre. Para usar una clave de KMS que administre, en ARN de clave administrada por el cliente, busque y elija la clave de KMS.

Para obtener más información, consulte Claves administradas por AWS y las claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service .

9. Elija Siguiente.
10. En Revisar y crear, revisa lo que CodeArtifact está creando por ti.

    - Flujo de paquetes muestra cómo están conectados su dominio y sus repositorios.
    - Paso 1: Crear un repositorio muestra los detalles sobre el repositorio y los repositorios ascendentes opcionales que se van a crear.
    - Paso 2: Seleccionar el dominio, se muestran los detalles sobre `my_domain`.

    Cuando haya terminado, elija Crear repositorio.

Creación de un repositorio (AWS CLI)

Use el comando `create-repository` para crear un repositorio en su dominio.

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

Ejemplo de salida:

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

Un repositorio nuevo no contiene ningún paquete. Cada repositorio está asociado a la cuenta de AWS en la que se autenticó cuando se creó el repositorio.

Crear un repositorio con etiquetas

Para crear un repositorio con etiquetas, añada el parámetro `--tags` a su comando `create-domain`.

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

Crear un repositorio con un repositorio ascendente

Puede especificar uno o más repositorios ascendentes al crear un repositorio.

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

Ejemplo de salida:

```
{
    "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**
>
> Para crear un repositorio con un repositorio ascendente, debe tener permiso para realizar la acción `AssociateWithDownstreamRepository` en el repositorio ascendente.

Para añadir un archivo ascendente a un repositorio una vez creado, consulte Añadir o eliminar repositorios ascendentes (consola) y Añadir o eliminar repositorios ascendentes (AWS CLI).

Conectarse a un repositorio

Una vez que hayas configurado tu perfil y tus credenciales para autenticarte en tu AWS cuenta, decide en qué repositorio quieres usarlos. CodeArtifact Dispone de las opciones siguientes:

- Creación de un repositorio. Para obtener más información, consulte Creación de un repositorio.
- Use un repositorio que ya exista en su cuenta. Puede usar el comando `list-repositories` para buscar los repositorios creados en su cuenta de AWS . Para obtener más información, consulte Listar repositorios.
- Usa un repositorio en una AWS cuenta diferente. Para obtener más información, consulte Políticas de repositorios.

Uso de un cliente administrador de paquetes

Una vez que sepa qué repositorio desea utilizar, consulte uno de los siguientes temas.

- CodeArtifact Utilizándolo con Maven
- Utilizándolo CodeArtifact con npm
- Utilizándolo con CodeArtifact NuGet
- Uso CodeArtifact con Python

Eliminar un repositorio

Puede eliminar un repositorio mediante la CodeArtifact consola o el AWS CLI. Una vez que se ha eliminado un repositorio, ya no puede enviar paquetes a él ni extraer paquetes de él. Todos los paquetes del repositorio dejan de estar disponibles permanentemente y no se pueden restaurar. Puede crear un repositorio con el mismo nombre, pero su contenido estará vacío.

Temas

- Eliminar un repositorio (consola)
- Eliminar un repositorio (AWS CLI)

Eliminar un repositorio (consola)

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Repositorios y, a continuación, elija el repositorio que desea eliminar.
3. Elija Eliminar y luego siga los pasos para eliminar el dominio.

Eliminar un repositorio (AWS CLI)

Use el comando `delete-repository` para eliminar un repositorio.

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

Ejemplo de salida:

```
{
```

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

Listar repositorios

Use los comandos de este tema para enumerar los repositorios de una AWS cuenta o dominio.

Listado de los repositorios de una cuenta de AWS

Usa este comando para enumerar todos los repositorios de tu AWS cuenta.

```
aws codeartifact list-repositories
```

Resultado de ejemplo:

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

Puede paginar la respuesta `list-repositories` utilizando los parámetros `--max-results` y `--next-token`. Para `--max-results`, especifique un número entero comprendido entre 1 y 1000 para especificar el número de resultados devueltos en una sola página. El valor predeterminado es 50. Para volver a las páginas siguientes, ejecute `list-repositories` de nuevo y pase el valor `nextToken` recibido en el resultado del comando anterior a `--next-token`. Cuando no se utiliza la opción `--next-token`, siempre se devuelve la primera página de resultados.

Enumeración de los repositorios del dominio

Utilice `list-repositories-in-domain` para obtener una lista de todos los repositorios de un dominio.

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

El resultado muestra que algunos de los repositorios están administrados por cuentas de AWS diferentes.

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

Puede paginar la respuesta `list-repositories-in-domain` utilizando los parámetros `--max-results` y `--next-token`. Para `--max-results`, especifique un número entero comprendido entre 1 y 1000 para especificar el número de resultados devueltos en una sola página. El valor predeterminado es 50. Para volver a las páginas siguientes, ejecute `list-repositories-in-domain` de nuevo y pase el valor `nextToken` recibido en el resultado del comando anterior a `--next-token`. Cuando no se utiliza la opción `--next-token`, siempre se devuelve la primera página de resultados.

Para mostrar los nombres de los repositorios en una lista más compacta, pruebe el siguiente comando.

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

Resultado de ejemplo:

```
repo1
repo2
repo3
```

El siguiente ejemplo muestra el ID de la cuenta además del nombre del repositorio.

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

Resultado de ejemplo:

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

Para obtener más información sobre el `--query` parámetro, consulta la referencia ListRepositoriesde la CodeArtifact API.

Cómo ver o modificar la configuración de un repositorio

Puede ver y actualizar los detalles de su repositorio mediante la CodeArtifact consola o el AWS Command Line Interface (AWS CLI).

> **Note**
>
> Después de haber creado un repositorio, no puede cambiar su nombre, cuenta AWS asociada ni dominio.

Temas

- Ver o modificar la configuración de un repositorio (consola)
- Ver o modificar la configuración de un repositorio (AWS CLI)

Ver o modificar la configuración de un repositorio (consola)

Puedes ver los detalles del repositorio y actualizarlos mediante la CodeArtifact consola.

1. Abre la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Repositorios y luego elija el nombre del repositorio que desea ver o modificar.
3. Expanda Detalles para ver lo siguiente:

   - El dominio del repositorio. Elija el nombre de dominio para obtener más información sobre él.
   - La política de recursos del repositorio. Elija Aplicar una política de repositorio para añadir una.
   - El nombre de recurso de Amazon (ARN) del repositorio.
   - Si su repositorio tiene una conexión externa, puede elegir la conexión para obtener más información sobre ella. Un repositorio solo puede tener una conexión externa. Para obtener más información, consulte Conectar un repositorio CodeArtifact a un repositorio público.
   - Si su repositorio tiene repositorios anteriores, puede elegir uno para ver sus detalles. Un repositorio puede tener hasta 10 repositorios ascendentes directos. Para obtener más información, consulte Trabajar con repositorios ascendentes en CodeArtifact.

   > **Note**
   >
   > Un repositorio puede tener una conexión externa o repositorios ascendentes, pero no ambos.

4. En Paquetes, puede ver todos los paquetes que estén disponibles en este repositorio. Elija un paquete para obtener más información sobre él.
5. Seleccione Ver instrucciones de conexión y, a continuación, elija un administrador de paquetes para aprender a configurarlo CodeArtifact.
6. Seleccione Aplicar política de repositorio para actualizar o añadir una política de recursos a su repositorio. Para obtener más información, consulte Políticas de repositorios.
7. Seleccione Editar para añadir o actualizar lo siguiente.

   - La descripción del repositorio.
   - Las etiquetas asociadas al repositorio.
   - Si su repositorio tiene una conexión externa, puede cambiar el repositorio público al que se conecta. De lo contrario, puede añadir uno o más repositorios existentes como repositorios ascendentes. Organícelos en el orden en que desee que se prioricen CodeArtifact cuando

solicite un paquete. Para obtener más información, consulte Orden de prioridad del repositorio ascendente.

Ver o modificar la configuración de un repositorio (AWS CLI)

Para ver la configuración actual de un repositorio CodeArtifact, usa el `describe-repository` comando.

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

Ejemplo de salida:

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

Modificación de la configuración inicial de un repositorio

Un repositorio ascendente permite a un cliente administrador de paquetes acceder a los paquetes contenidos en más de un repositorio mediante un único punto de conexión de URL. Para añadir o cambiar la relación ascendente de un repositorio, use el comando `update-repository`.

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

Ejemplo de salida:

```
{
```

```
    "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**
>
> Para añadir un repositorio ascendente, debe tener permiso para realizar la acción `AssociateWithDownstreamRepository` en el repositorio ascendente.

Para eliminar la relación ascendente de un repositorio, utilice una lista vacía como argumento de la opción `--upstreams`.

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

Ejemplo de salida:

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

```
}
```

Políticas de repositorios

CodeArtifact utiliza permisos basados en recursos para controlar el acceso. Los permisos basados en recursos le permiten especificar quién tiene acceso a un repositorio y qué acciones puede realizar en él. De forma predeterminada, solo el propietario del repositorio tiene acceso a él. Puede aplicar un documento de política que permita a otras entidades principales de IAM el acceso a su repositorio.

Para obtener más información, consulte Políticas basadas en recursos y Políticas basadas en identidades y Políticas basadas en recursos.

Cree una política de recursos para conceder el acceso de lectura

Una política de recursos es un archivo de texto en formato JSON. El archivo debe especificar una entidad principal (actor), una o más acciones y un efecto (`Allow` o `Deny`). Por ejemplo, la siguiente política de recursos concede a la cuenta permiso `123456789012` para descargar paquetes del repositorio.

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

Como la política solo se evalúa para las operaciones realizadas en el repositorio al que está adjunta, no es necesario que especifique un recurso. Como el recurso está implícito, puede establecer el valor `Resource` en `*`. Para que un administrador de paquetes pueda descargar un paquete de este repositorio, también tendrá que crear una política de dominio para el acceso entre cuentas. La política de dominio debe conceder, como mínimo, permisos

`codeartifact:GetAuthorizationToken` y `sts:GetServiceBearerToken` a la entidad principal. Para ver un ejemplo de una política de dominio completa para otorgar acceso entre cuentas, consulte Ejemplo de políticas de dominio.

> **Note**
>
> La acción `codeartifact:ReadFromRepository` solo se puede usar en un recurso del repositorio. No puede colocar el nombre de recurso de Amazon (ARN) de un paquete como recurso con `codeartifact:ReadFromRepository` como la acción para permitir el acceso de lectura a un subconjunto de paquetes de un repositorio. Una entidad principal determinada puede leer todos los paquetes de un repositorio o ninguno de ellos.

Como la única acción especificada en el repositorio es `ReadFromRepository`, los usuarios y los roles de la cuenta `1234567890` puedan descargar paquetes del repositorio. Sin embargo, no pueden realizar otras acciones en ellos (por ejemplo, enumerar los nombres y las versiones de los paquetes). Por lo general, se conceden permisos en la siguiente política, además de `ReadFromRepository` porque un usuario que descarga paquetes de un repositorio también necesita interactuar con él de otras formas.

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

```
    ]
}
```

Configuración de una política

Después de haber creado un documento de políticas, utilice el comando `put-repository-permissions-policy` para adjuntarlo a un repositorio:

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

Al llamar a `put-repository-permissions-policy`, se ignora la política de recursos del repositorio al evaluar los permisos. Esto garantiza que el propietario de un dominio no pueda excluirse del repositorio, lo que le impediría actualizar la política de recursos.

> **Note**
>
> No puedes conceder permisos a otra AWS cuenta para actualizar la política de recursos de un repositorio mediante una política de recursos, ya que la política de recursos se ignora cuando se llama. put-repository-permissions-policy

Resultado de ejemplo:

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

El resultado del comando contiene el nombre de recurso de Amazon (ARN) del recurso de repositorio, todo el contenido del documento de política y un identificador de revisión. Puede pasar el identificador de revisión a `put-repository-permissions-policy` usando la opción `--policy-revision`. Esto garantiza que se sobrescriba una revisión conocida del documento y no una versión más reciente configurada por otro escritor.

Leer una política

Utilice el comando `get-repository-permissions-policy` para leer una versión existente de un documento de política. Para formatear la salida para que sea legible, utilice `--output` y `--query policy.document` junto con el módulo `json.tool` de Python.

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

Resultado de ejemplo:

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

Eliminar una política

Utilice el comando `delete-repository-permissions-policy` para eliminar una política de un repositorio.

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

El formato de la salida es el mismo que el del comando `get-repository-permissions-policy`.

Otorgar acceso de lectura a las entidades principales

Al especificar el usuario raíz de una cuenta como entidad principal en un documento de política, se concede acceso a todos los usuarios y roles de esa cuenta. Para limitar el acceso a los usuarios o roles seleccionados, utilice su ARN en la sección `Principal` de la política. Por ejemplo, utilice lo siguiente para conceder acceso de lectura al usuario de IAM `bob` de la cuenta `123456789012`.

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

Otorgar acceso de escritura a los paquetes

La acción `codeartifact:PublishPackageVersion` se utiliza para controlar el permiso para publicar nuevas versiones de un paquete. El recurso utilizado en esta acción debe ser un paquete. El formato de los CodeArtifact ARN de los paquetes es el siguiente.

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

El siguiente ejemplo muestra el ARN de un paquete npm con alcance `@parity` y nombre `ui` en el repositorio `example-repo` del dominio `my_domain`.

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

El ARN de un paquete npm sin alcance tiene la cadena vacía para el campo de espacio de nombres. Por ejemplo, el siguiente es el ARN de un paquete sin alcance y con un nombre `react` en el repositorio `example-repo` del dominio `my_domain`.

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

La siguiente política otorga permiso `123456789012` a la cuenta para publicar versiones de `@parity/ui` en el repositorio `example-repo`.

```
{
    "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/example-repo/npm/parity/ui"
        }
    ]
}
```

> **Important**
>
> Para conceder permiso para publicar versiones de NuGet paquetes y de Maven, añada además los siguientes permisos. `codeartifact:PublishPackageVersion`
>
> 1. NuGet: `codeartifact:ReadFromRepository` y especifique el recurso del repositorio
> 2. Maven: `codeartifact:PutPackageMetadata`

Como esta política especifica un dominio y un repositorio como parte del recurso, solo permite publicar cuando están adjuntos a ese repositorio.

Otorgar acceso de escritura a un repositorio

Puede usar caracteres comodín para conceder permisos de escritura a todos los paquetes de un repositorio. Por ejemplo, use la siguiente política para conceder permiso a una cuenta para escribir en todos los paquetes del repositorio `example-repo`.

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

Etiquete un repositorio en CodeArtifact

Las etiquetas son pares clave-valor asociados a los recursos de AWS. Puede aplicar etiquetas a sus repositorios en CodeArtifact. Para obtener información sobre el etiquetado de CodeArtifact recursos, los casos de uso, las restricciones de clave y valor de las etiquetas y los tipos de recursos compatibles, consulte. Etiquetado de recursos

Puede utilizar la CLI para especificar etiquetas al crear un repositorio. Puede utilizar la consola o la CLI para añadir o eliminar etiquetas, y para actualizar los valores de las etiquetas de un repositorio. Puede agregar hasta 50 etiquetas a cada repositorio.

Temas

- Etiquetar repositorios (CLI)
- Etiquetar repositorios (consola)

Etiquetar repositorios (CLI)

Puede utilizar la CLI para administrar las etiquetas del repositorio.

Temas

Agregar etiquetas a un repositorio (CLI)

Puede utilizar la consola o la AWS CLI para etiquetar repositorios.

Para agregar una etiqueta a un repositorio al crearlo, consulte Crear un repositorio de .

En estos pasos, se presupone que ya ha instalado una versión reciente de la AWS CLI o que la ha actualizado a la versión actual. Para obtener más información, consulte Instalación de la AWS Command Line Interface.

En el terminal o en la línea de comandos, ejecute el comando tag-resource, especificando el nombre de recurso de Amazon (ARN) del repositorio al que desea agregar etiquetas y la clave y el valor de la etiqueta que desea agregar.

> **Note**
>
> Para obtener el ARN del repositorio, ejecute el comando `describe-repository`:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Puede agregar más de una etiqueta a un repositorio. *Por ejemplo, para etiquetar un repositorio llamado my_repo en un dominio llamado my_domain con dos etiquetas, una clave de etiqueta llamada key1 con el valor de etiqueta value1 y una clave de etiqueta llamada key2 con el valor de etiqueta value2*:

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

Si tiene éxito, este comando no tiene salida.

Ver etiquetas para un repositorio (CLI)

Siga estos pasos para usar el AWS CLI para ver las AWS etiquetas de un repositorio. Si no se han añadido etiquetas, la lista obtenida está vacía.

En el terminal o la línea de comandos, ejecute el comando list-tags-for-resource.

> **Note**
>
> Para obtener el ARN del repositorio, ejecute el comando `describe-repository`:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Por ejemplo, para ver una lista de claves de etiquetas y valores de etiquetas para un repositorio llamado *my_repo* en un dominio llamado *my_domain* con el valor ARN `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
```

Si se ejecuta correctamente, este comando proporciona información similar a la siguiente:

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

Editar etiquetas para un repositorio (CLI)

Siga estos pasos para usar el AWS CLI para editar una etiqueta de un repositorio. Puede cambiar el valor de una clave existente o añadir otra clave.

En el terminal o la línea de comandos, ejecute el comando tag-resource, especificando el ARN del repositorio en el que desea actualizar una etiqueta y especifique la clave y el valor de la etiqueta.

> **Note**
>
> Para obtener el ARN del repositorio, ejecute el comando `describe-repository`:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

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

Si tiene éxito, este comando no tiene salida.

Eliminar etiquetas de un repositorio (CLI)

Siga estos pasos para usar el AWS CLI para eliminar una etiqueta de un repositorio.

> **Note**
>
> Si elimina un repositorio de , se quitan todas las asociaciones de etiquetas del repositorio eliminado. No es necesario quitar las etiquetas antes de eliminar un repositorio.

En el terminal o la línea de comandos, ejecute el comando untag-resource, especificando el ARN del repositorio cuya etiqueta desea eliminar y la clave de la etiqueta que desea eliminar.

> **Note**
>
> Para obtener el ARN del repositorio, ejecute el comando `describe-repository`:

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

Por ejemplo, para eliminar varias etiquetas de un repositorio llamado *my_repo* en un dominio llamado *my_domain* con las claves de etiqueta *key1* y *key2*:

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

Si tiene éxito, este comando no tiene salida. Tras eliminar las etiquetas, puede ver las etiquetas restantes del repositorio mediante el comando `list-tags-for-resource`.

Etiquetar repositorios (consola)

Puede utilizar la consola o la CLI para etiquetar recursos.

Temas

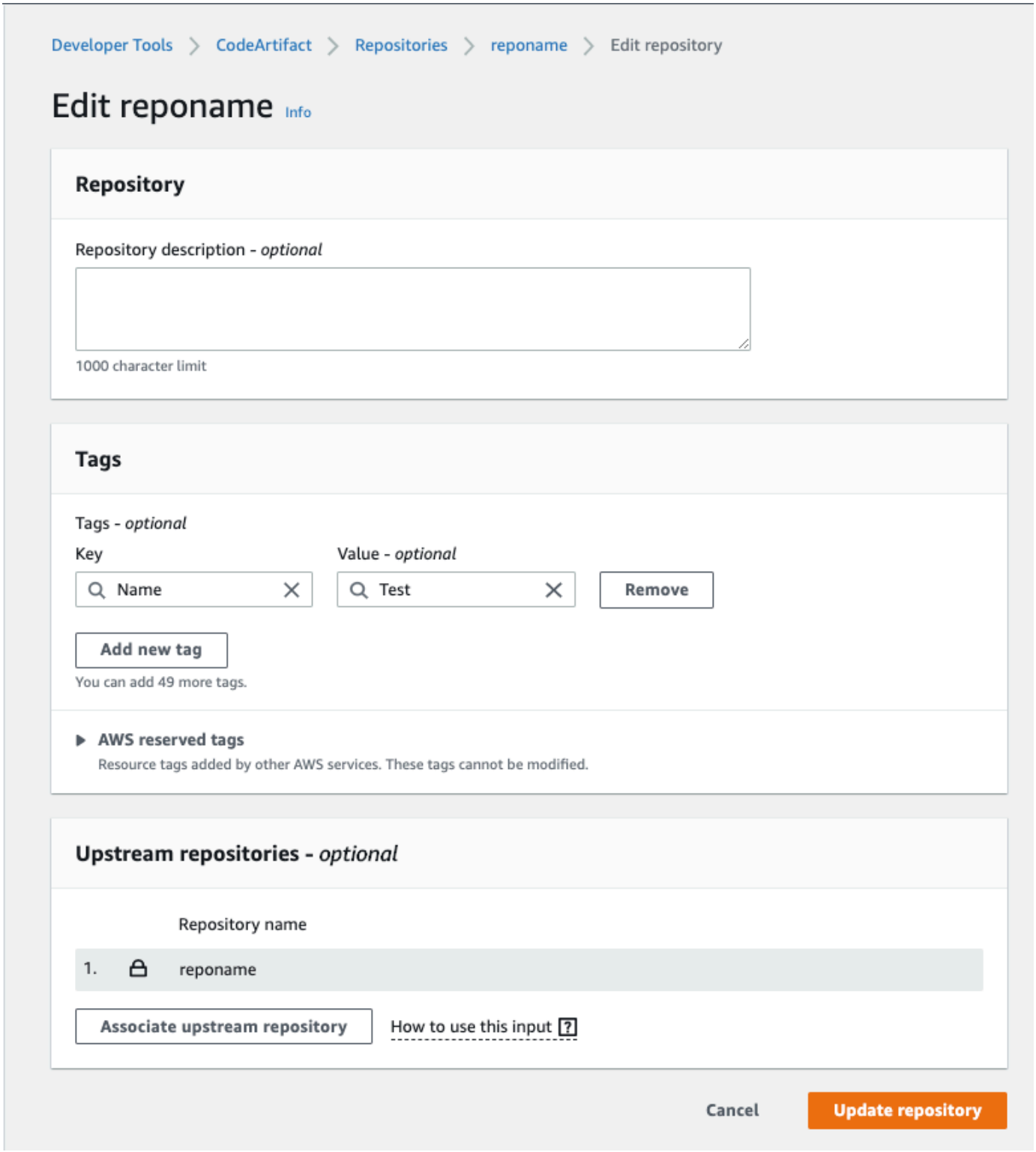
- Añadir etiquetas a un repositorio (consola)
- Ver etiquetas de un repositorio (consola)
- Editar etiquetas de un repositorio (consola)
- Eliminar etiquetas de un repositorio (consola)

Añadir etiquetas a un repositorio (consola)

Puede utilizar la consola para añadir etiquetas a un repositorio existente.

1. Abre la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En la página Repositorios, elija el repositorio al que desea agregar etiquetas.
3. Amplíe la sección Detalles.
4. En Etiquetas del repositorio, si no hay etiquetas en el repositorio, seleccione Añadir etiquetas de repositorio. Si hay etiquetas en el repositorio, seleccione Ver y editar etiquetas de repositorio.
5. Elija Añadir nueva etiqueta.

6. En los campos Clave y Valor, introduzca el texto para cada etiqueta que desee agregar. (El campo Value (Valor) es opcional). Por ejemplo, en Key (Clave), escriba **`Name`**. En Valor, escriba **`Test`**.



7. (Opcional) ElijaAdd tag (Añadir etiqueta) para añadir más filas y escribir más etiquetas.

8. Seleccione Actualizar repositorio.

Ver etiquetas de un repositorio (consola)

Puede utilizar la consola para obtener una lista de las etiquetas de los repositorios existentes.

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En la página Repositorios, elija el nombre del repositorio cuyas etiquetas desea ver.
3. Amplíe la sección Detalles.
4. En Etiquetas de repositorio, seleccione Ver y editar etiquetas de repositorio.

> **Note**
>
> Si no se ha agregado ninguna etiqueta a este repositorio, la consola mostrará Agregar etiquetas de repositorio.

Editar etiquetas de un repositorio (consola)

Puede utilizar la consola para editar las etiquetas que se han añadido a los repositorios.

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En la página Repositorios, elija el nombre del repositorio cuyas etiquetas desea actualizar.
3. Amplíe la sección Detalles.
4. En Etiquetas de repositorio, seleccione Ver y editar etiquetas de repositorio.

> **Note**
>
> Si no se ha agregado ninguna etiqueta a este repositorio, la consola mostrará Agregar etiquetas de repositorio.

5. En los campos Key (Clave) y Value (Valor), actualice los valores que sean necesarios. Por ejemplo, para la clave **`Name`**, en Value (Valor), cambie **`Test`** a **`Prod`**.
6. Seleccione Actualizar repositorio.

Eliminar etiquetas de un repositorio (consola)

Puede utilizar la consola para eliminar etiquetas de repositorios.

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En la página Repositorios, elija el nombre del repositorio cuyas etiquetas desea eliminar.
3. Amplíe la sección Detalles.
4. En Etiquetas de repositorio, seleccione Ver y editar etiquetas de repositorio.

> **Note**
>
> Si no se ha agregado ninguna etiqueta a este repositorio, la consola mostrará Agregar etiquetas de repositorio.

5. Junto a la clave y el valor de cada etiqueta que desea eliminar, elija Eliminar.
6. Seleccione Actualizar repositorio.

Trabajar con repositorios ascendentes en CodeArtifact

Un repositorio puede tener otros repositorios de AWS CodeArtifact como repositorios ascendentes. Esto permite a un cliente administrador de paquetes acceder a los paquetes que están contenidos en más de un repositorio mediante un único punto de conexión del repositorio.

Puede añadir uno o más repositorios ascendentes a un repositorio de AWS CodeArtifact mediante el SDK AWS Management Console o AWS CLI. Para asociar un repositorio a un repositorio ascendente, debe tener permiso para realizar la acción `AssociateWithDownstreamRepository` en el repositorio ascendente. Para obtener más información, consulte Crear un repositorio con un repositorio ascendente y Añadir o eliminar repositorios ascendentes.

Si un repositorio ascendente tiene una conexión externa a un repositorio público, los repositorios descendentes pueden extraer paquetes de ese repositorio público. Por ejemplo, supongamos que el repositorio `my_repo` tiene un repositorio ascendente denominado `upstream` y `upstream` tiene una conexión externa a un repositorio npm público. En este caso, un administrador de paquetes al que esté conectado a `my_repo` puede extraer paquetes del repositorio público de npm. Para obtener más información sobre cómo solicitar paquetes desde repositorios ascendentes o conexiones externas, consulte Solicitar una versión de paquete con repositorios ascendentes o Solicitud de paquetes desde conexiones externas.

Temas

¿Cuál es la diferencia entre los repositorios ascendentes y las conexiones externas?

En CodeArtifact, los repositorios ascendentes y las conexiones externas se comportan prácticamente de la misma manera, pero hay algunas diferencias importantes.

1. Puede añadir hasta 10 repositorios ascendentes a un repositorio de CodeArtifact. Sólo puede agregar una conexión externa.
2. Hay llamadas a la API independientes para añadir un repositorio ascendente o una conexión externa.
3. El comportamiento de retención de paquetes es ligeramente diferente, ya que los paquetes solicitados desde los repositorios ascendentes se conservan en esos repositorios. Para obtener más información, consulte Retención de paquetes en repositorios intermedios.

Añadir o eliminar repositorios ascendentes

Siga los pasos de las siguientes secciones para añadir o eliminar repositorios ascendentes de un repositorio de CodeArtifact o desde él. Para obtener más información sobre los repositorios ascendentes, consulte Trabajar con repositorios ascendentes en CodeArtifact.

Esta guía contiene información sobre cómo configurar otros repositorios de CodeArtifact como repositorios ascendentes. Para obtener información sobre cómo configurar una conexión externa a repositorios públicos como npmjs.com, Nuget Gallery, Central de Maven o PyPI, consulte Añadir una conexión externa.

Añadir o eliminar repositorios ascendentes (consola)

Realice los pasos del siguiente procedimiento para añadir un repositorio como repositorio ascendente mediante la consola CodeArtifact. Para obtener información sobre cómo añadir un repositorio ascendente con la AWS CLI, consulte Añadir o eliminar repositorios ascendentes (AWS CLI).

Para añadir un repositorio ascendente mediante la consola CodeArtifact

1. Abra la consola de AWS CodeArtifact en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Dominios y, a continuación, elija el nombre de dominio que contiene el repositorio.
3. Elija el nombre de su repositorio.
4. Elija Edit (Editar).
5. En repositorios ascendentes, elija Asociar un repositorio ascendente y añada el repositorio que desee añadir como repositorio ascendente. Solo puede añadir repositorios en el mismo dominio que los repositorios ascendentes.

6. Seleccione Actualizar repositorio.

Para eliminar un repositorio ascendente mediante la consola CodeArtifact

1. Abra la consola de AWS CodeArtifact en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Dominios y, a continuación, elija el nombre de dominio que contiene el repositorio.
3. Elija el nombre de su repositorio.
4. Elija Edit (Editar).
5. En repositorios ascendentes, busque la entrada de la lista del repositorio ascendente que desea eliminar y seleccione Desasociar.

> **Important**
>
> Una vez que elimine un repositorio ascendente de un repositorio de CodeArtifact, los administradores de paquetes no tendrán acceso a los paquetes del repositorio ascendente ni a ninguno de sus repositorios ascendentes.

6. Seleccione Actualizar repositorio.

Añadir o eliminar repositorios ascendentes (AWS CLI)

Puede añadir o eliminar los repositorios ascendentes de un repositorio de CodeArtifact mediante la AWS Command Line Interface (AWS CLI). Para ello, utilice el comando `update-repository` y especifique los repositorios ascendentes mediante el parámetro `--upstreams`.

Solo puede añadir repositorios en el mismo dominio que los repositorios ascendentes.

Para añadir repositorios ascendentes (AWS CLI)

1. Si no lo ha hecho, siga los pasos que se indican en Configuración con AWS CodeArtifact para establecer y configurar la AWS CLI con CodeArtifact.
2. Use el comando `aws codeartifact update-repository` con la marca `--upstreams` para añadir repositorios ascendentes.

> **Note**
>
> Al llamar al comando `update-repository`, se sustituyen los repositorios ascendentes configurados existentes por la lista de repositorios proporcionada con la marca `--upstreams`. Si desea añadir repositorios ascendentes y conservar los existentes, debe incluir los repositorios ascendentes existentes en la llamada.

El siguiente comando de ejemplo agrega dos repositorios ascendentes a un repositorio denominado `my_repo` que se encuentra en un dominio denominado `my_domain`. El orden que los repositorios ascendentes tengan en el parámetro `--upstreams` determinará la prioridad de búsqueda cuando CodeArtifact solicite un paquete del repositorio `my_repo`. Para obtener más información, consulte Orden de prioridad del repositorio ascendente.

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

El resultado contiene los repositorios ascendentes, tal y como se indica a continuación.

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

Para eliminar un repositorio ascendente (AWS CLI)

1. Si no lo ha hecho, siga los pasos que se indican en Configuración con AWS CodeArtifact para establecer y configurar la AWS CLI con CodeArtifact.
2. Para eliminar los repositorios ascendentes de un repositorio de CodeArtifact, use el comando `update-repository` con la marca `--upstreams`. La lista de repositorios proporcionada al comando será el nuevo conjunto de repositorios ascendentes para el repositorio CodeArtifact. Incluya los repositorios ascendentes existentes que desee conservar y omita los repositorios ascendentes que desee eliminar.

   Para eliminar todos los repositorios ascendentes de un repositorio, use el comando `update-repository` e incluya `--upstreams` sin un argumento. Lo siguiente elimina los repositorios ascendentes de un repositorio denominado `my_repo` que esté contenido en un dominio denominado `my_domain`.

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

   El resultado muestra que la lista de `upstreams` está vacía.

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

Conectar un repositorio CodeArtifact a un repositorio público

Puede añadir una conexión externa entre un repositorio de CodeArtifact y un repositorio público externo como https://npmjs.com o el repositorio central de Maven. Luego, cuando solicita un paquete del repositorio de CodeArtifact que aún no está presente en el repositorio, el paquete se puede

obtener de la conexión externa. Esto permite consumir las dependencias de código abierto que utiliza su aplicación.

En CodeArtifact, la forma prevista de utilizar conexiones externas es tener un repositorio por dominio con una conexión externa a un repositorio público determinado. Por ejemplo, si quiere conectarse a npmjs.com, configure un repositorio de su dominio con una conexión externa a npmjs.com y configure todos los demás repositorios con una conexión anterior. De esta forma, todos los repositorios pueden utilizar los paquetes que ya se han obtenido de npmjs.com, en lugar de volver a buscarlos y almacenarlos.

Temas

Conectarse a un repositorio externo (consola)

Cuando utilice la consola para añadir una conexión a un repositorio externo, ocurrirá lo siguiente:

1. Se creará un repositorio `-store` para el repositorio externo en su dominio CodeArtifact si aún no existe uno. Estos repositorios `-store` se comportan como repositorios intermedios entre su repositorio y el repositorio externo y le permiten conectarse a más de un repositorio externo.
2. El repositorio `-store` correspondiente se añade de forma previa a su repositorio.

La siguiente lista contiene cada repositorio `-store` de CodeArtifact y el repositorio externo respectivo al que se conectan.

1. `commonsware-store` está conectado al repositorio de Android de CommonsWare.
2. `google-android-store` está conectado a Google Android.
3. `gradle-plugins-store` está conectado a los complementos de Gradle.
4. `maven-central-store` está conectado al repositorio central de Maven.
5. `clojars-store` está conectado al repositorio de Clojars.
6. `npm-store` está conectado a npmjs.com.
7. `nuget-store` está conectado a nuget.org.

8. `pypi-store` está conectado a la Python Packaging Authority.

Para conectarse a un repositorio externo (consola)

1. Abra la consola de AWS CodeArtifact en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Dominios y, a continuación, elija el nombre de dominio que contiene el repositorio.
3. Elija el nombre de su repositorio.
4. Elija Edit (Editar).
5. En repositorios ascendentes, elija Asociar un repositorio ascendente y añada el repositorio `-store` correspondiente que esté conectado como un repositorio ascendente.
6. Seleccione Actualizar repositorio.

Una vez que el repositorio `-store` se haya agregado como repositorio principal, los administradores de paquetes conectados a su repositorio de CodeArtifact pueden recuperar los paquetes del repositorio externo respectivo.

Conectarse a un repositorio externo (CLI)

Puede usar la AWS CLI para conectar su repositorio de CodeArtifact a un repositorio externo añadiendo una conexión externa directamente al repositorio. Esto permitirá a los usuarios conectados al repositorio CodeArtifact, o a cualquiera de sus repositorios posteriores, obtener paquetes del repositorio externo configurado. Cada repositorio de CodeArtifact solo puede tener una conexión externa.

Se recomienda tener un repositorio por dominio con una conexión externa a un repositorio público determinado. Para conectar otros repositorios al repositorio público, agregue el repositorio con la conexión externa como ascendente para ellos. Si usted u otra persona en su dominio ya ha configurado conexiones externas en la consola, es probable que su dominio ya tenga un repositorio `-store` con una conexión externa al repositorio público al que desea conectarse. Para obtener más información sobre los repositorios `-store` y la conexión con la consola, consulte Conectarse a un repositorio externo (consola).

Para agregar una conexión externa a un repositorio de CodeArtifact (CLI)

- Utilice `associate-external-connection` para agregar una conexión externa. El siguiente ejemplo conecta un repositorio al registro público de npm, npmjs.com. Para ver una lista de los repositorios externos admitidos, consulte Repositorios de conexiones externas compatibles.

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

Ejemplo de resultados:

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

Tras añadir una conexión externa, consulte Solicitud de paquetes desde conexiones externas para obtener información sobre cómo solicitar paquetes desde un repositorio externo con una conexión externa.

Repositorios de conexiones externas compatibles

CodeArtifact admite una conexión externa a los siguientes repositorios públicos. Para usar la CLI de CodeArtifact para especificar una conexión externa, utilice el valor de la columna Nombre para

el parámetro `--external-connection` cuando ejecute el comando `associate-external-connection`.

Repository type	Descripción	Nombre
npm	registro público de npm	`public:npmjs`
Python	Índice de paquetes de Python	`public:pypi`
Maven	Central de Maven	`public:maven-central`
Maven	Repositorio de Google de Android	`public:maven-googleandroid`
Maven	Repositorio de complementos de Gradle	`public:maven-gradleplugins`
Maven	Repositorio de CommonsWare de Android	`public:maven-commonsware`
Maven	Repositorio de Clojars	`public:maven-clojars`
NuGet	Galería NuGet	`public:nuget-org`

Eliminar una conexión externa (CLI)

Para eliminar una conexión externa que se agregó mediante el comando `associate-external-connection` en AWS CLI, utilice `disassociate-external-connection`.

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

Ejemplo de resultados:

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

Solicitar una versión de paquete con repositorios ascendentes

Cuando un cliente (por ejemplo, npm) solicita una versión de paquete de un repositorio de CodeArtifact denominado `my_repo` que tiene varios repositorios ascendentes, puede ocurrir lo siguiente:

- Si `my_repo` contiene la versión del paquete solicitada, se devuelve al cliente.
- Si `my_repo` no contiene la versión del paquete solicitada, CodeArtifact la busca en los repositorios ascendentes de `my_repo`. Si se encuentra la versión del paquete, se copia una referencia a la misma en `my_repo` y la versión del paquete se devuelve al cliente.
- Si `my_repo` ni sus repositorios ascendentes contienen la versión del paquete, se devuelve una respuesta HTTP 404 `Not Found` al cliente.

Al añadir repositorios ascendentes mediante el comando `create-repository` o `update-repository`, el orden en que se pasan al parámetro `--upstreams` determina su prioridad cuando se solicita una versión de paquete. Especifique los repositorios ascendentes con `--upstreams` en el orden en que desea que CodeArtifact utilice cuando se solicite una versión de paquete. Para obtener más información, consulte Orden de prioridad del repositorio ascendente.

El número máximo de repositorios ascendentes directos permitidos para un repositorio es 10. Como los repositorios ascendentes directos también pueden tener sus propios repositorios ascendentes directos, CodeArtifact puede buscar versiones de paquetes en más de 10 repositorios. El número máximo de repositorios en los que CodeArtifact consulta cuando se solicita una versión de paquete es 25.

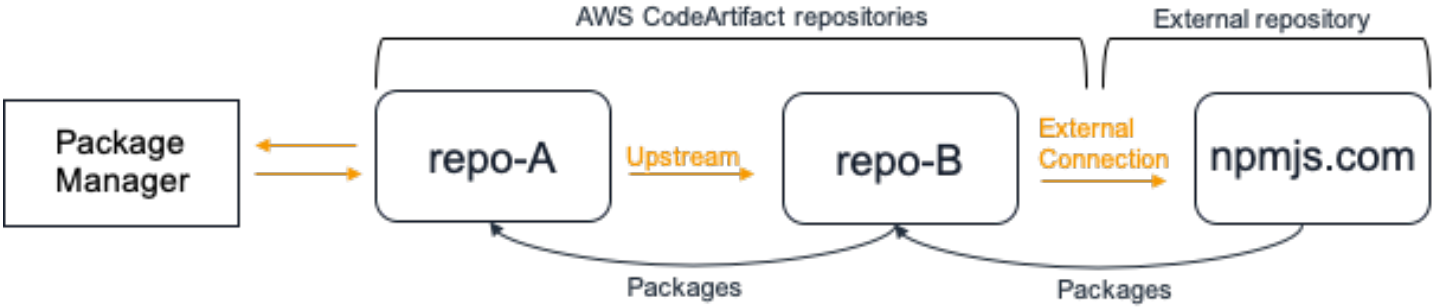
Retención de paquetes de repositorios ascendentes

Si la versión de un paquete solicitada se encuentra en un repositorio ascendente, se conserva una referencia a la misma y siempre está disponible en el repositorio descendente. La versión del paquete retenida no se ve afectada por ninguno de los siguientes factores:

- Eliminar el repositorio ascendente.
- Desconectar el repositorio ascendente del repositorio descendente.
- Eliminar la versión del paquete del repositorio ascendente.
- Editar la versión del paquete en el repositorio ascendente (por ejemplo, añadiéndole un nuevo activo).

Obtener paquetes a través de una relación ascendente

Si un repositorio de CodeArtifact tiene una relación ascendente con un repositorio que tiene una conexión externa, las solicitudes de paquetes que no están en el repositorio ascendente se copian del repositorio externo. Por ejemplo, considere la siguiente configuración: un repositorio denominado `repo-A` tiene un repositorio ascendente denominado `repo-B`. `repo-B` tiene una conexión externa a https://npmjs.com.



Si `npm` está configurado para usar el repositorio `repo-A`, la ejecución de `npm install` desencadena la copia de paquetes de https://npmjs.com a `repo-B`. También se incluyen las versiones instaladas`repo-A`. El siguiente ejemplo instala `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
```

Después de ejecutarse `npm install`, `repo-A` contiene solo la versión más reciente (`lodash 4.17.20`), ya que es la versión de la que `npm` obtuvo de `repo-A`.

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

Ejemplo de resultados:

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

Como `repo-B` tiene una conexión externa a https://npmjs.com, todas las versiones de los paquetes que se importan desde https://npmjs.com se almacenan en `repo-B`. Estas versiones de paquetes podrían haber sido recuperadas por cualquier repositorio descendente con una relación ascendente con `repo-B`.

El contenido de `repo-B` proporciona una forma de ver todos los paquetes y las versiones de los paquetes importados desde https://npmjs.com a lo largo del tiempo. Por ejemplo, para ver todas las versiones del paquete `lodash` importadas a lo largo del tiempo, puede utilizar `list-package-versions` de la siguiente manera.

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

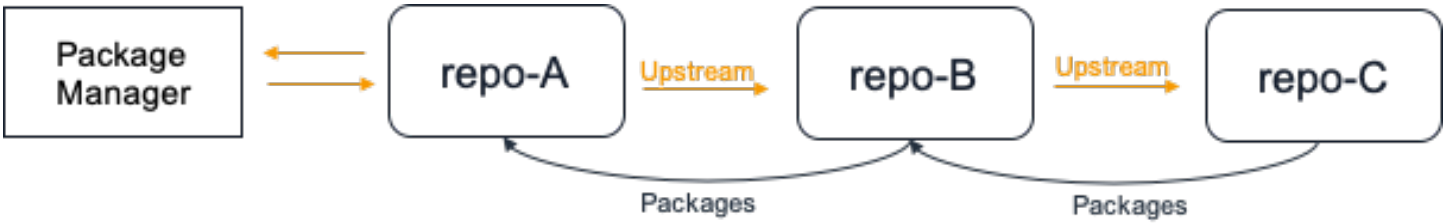
Ejemplo de resultados:

```
{
    "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"
}
```


Retención de paquetes en repositorios intermedios

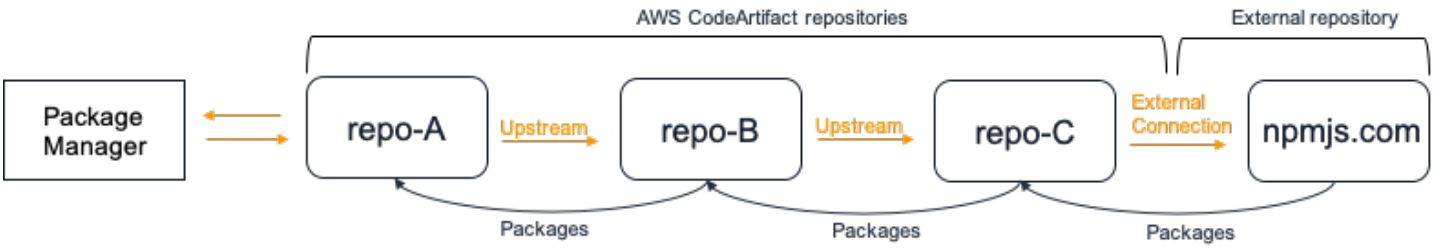
CodeArtifact permite encadenar repositorios ascendentes. Por ejemplo, `repo-A` puede tener `repo-B` como un flujo ascendente y `repo-B` puede tener `repo-C` como un flujo ascendente. Esta configuración hace que las versiones del paquete en `repo-B` y `repo-C` estén disponibles de `repo-A`.



Cuando un administrador de paquetes se conecta al repositorio `repo-A` y obtiene una versión del paquete del repositorio `repo-C`, la versión del paquete no se conservará en el repositorio `repo-`

B. La versión del paquete solo se conservará en el repositorio más descendente, en este ejemplo, `repo-A`. No se conservará en ningún repositorio intermedio. Esto también es válido para cadenas más largas; por ejemplo, si hubiera cuatro repositorios `repo-A`, `repo-B`, `repo-C` y `repo-D` y un administrador de paquetes conectado a `repo-A` extrajo una versión de paquete de `repo-D`, la versión del paquete se conservaría en `repo-A`, pero no en `repo-B` ni `repo-C`.

El comportamiento de retención de paquetes es similar cuando se extrae una versión de paquete de un repositorio externo, excepto que la versión del paquete siempre se conserva en el repositorio que tiene la conexión externa adjunta. Por ejemplo, `repo-A` tiene `repo-B` como flujo ascendente. `repo-B` tiene `repo-C` como conexión ascendente y `repo-C` también tiene npmjs.com configurado como una conexión externa; consulte el siguiente diagrama.



Si un administrador de paquetes conectado a `repo-A` solicita una versión de paquete, por ejemplo, lodash 4.17.20, y la versión del paquete no está presente en ninguno de los tres repositorios, se obtendrá de npmjs.com. Cuando se extraiga lodash 4.17.20, se conservará en `repo-A`, ya que es el repositorio más avanzado y `repo-C` ya que tiene conectada la conexión externa a npmjs.com. lodash 4.17.20 no se conservará en `repo-B` porque se trata de un repositorio intermedio.

Solicitud de paquetes desde conexiones externas

En las siguientes secciones se describe cómo solicitar un paquete desde una conexión externa y el comportamiento esperado de CodeArtifact al solicitar un paquete.

Temas

Extraer paquetes desde una conexión externa

Para recuperar paquetes de una conexión externa una vez que los haya agregado a su repositorio de CodeArtifact como se describe en Conectar un repositorio CodeArtifact a un repositorio público, configure su administrador de paquetes para usar su repositorio e instalar los paquetes.

> **Note**
>
> En las siguientes instrucciones se utiliza npm, para ver las instrucciones de configuración y uso de otros tipos de paquetes, consulte Uso de CodeArtifact con Maven, Uso de CodeArtifact con NuGet o Uso de CodeArtifact con Python.

Para extraer paquetes desde una conexión externa

1. Configure y autentique su administrador de paquetes con su repositorio CodeArtifact. Para npm, utilice el siguiente comando `aws codeartifact login`.

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

2. Solicite el paquete desde el repositorio público. Para npm, use el siguiente comando `npm install`, reemplazando *lodash* con el paquete que desea instalar.

   ```
   npm install lodash
   ```

3. Una vez que el paquete se haya copiado en su repositorio de CodeArtifact, puede usar los comandos `list-packages` y `list-package-versions` para verlo.

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

   Ejemplo de resultados:

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

```
    ]
}
```

El comando `list-package-versions` muestra todas las versiones del paquete copiadas en su repositorio de CodeArtifact.

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

Ejemplo de resultados:

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

Latencia de conexión externa

Al recuperar un paquete de un repositorio público mediante una conexión externa, hay un retraso desde que el paquete se recupera del repositorio público hasta que se almacena en su repositorio CodeArtifact. Por ejemplo, supongamos que ha instalado la versión 1.2.5 del paquete npm "lodash", tal y como se describe en Extraer paquetes desde una conexión externa. Aunque el comando lodash `npm install lodash` se completó correctamente, es posible que la versión del paquete aún no aparezca en su repositorio de CodeArtifact. Por lo general, la versión del paquete tarda unos 3 minutos en aparecer en su repositorio, aunque en ocasiones puede tardar más.

Debido a esta latencia, es posible que haya recuperado correctamente una versión del paquete, pero es posible que aún no pueda ver la versión en su repositorio en la consola CodeArtifact o al llamar a las operaciones API ListPackages y ListPackageVersions. Una vez que CodeArtifact haya

conservado asincrónicamente la versión del paquete, estará visible en la consola y a través de las solicitudes de la API.

Comportamiento de CodeArtifact cuando un repositorio externo no está disponible

Ocasionalmente, un repositorio externo sufre una interrupción, lo que significa que CodeArtifact no puede recuperar paquetes de él, o la recuperación de paquetes es mucho más lenta de lo normal. Cuando esto ocurra, las versiones de paquetes ya extraídas de un repositorio externo (por ejemplo, npmjs.com) y almacenadas en un repositorio de CodeArtifact seguirán estando disponibles para su descarga desde CodeArtifact. Sin embargo, es posible que los paquetes que aún no estén almacenados en CodeArtifact no estén disponibles, incluso cuando se haya configurado una conexión externa a ese repositorio. Por ejemplo, su repositorio CodeArtifact puede contener la versión `lodash 4.17.19` del paquete npm porque es lo que ha estado usando en su aplicación hasta ahora. Cuando desee actualizar a `4.17.20`, normalmente CodeArtifact obtendrá esa nueva versión de npmjs.com y la almacenará en su repositorio de CodeArtifact. Sin embargo, si npmjs.com sufre una interrupción, esta nueva versión no estará disponible. La única solución es volver a intentarlo más tarde, una vez que npmjs.com se haya recuperado.

Las interrupciones del repositorio externo también pueden afectar a la publicación de nuevas versiones de paquetes en CodeArtifact. En un repositorio con una conexión externa configurada, CodeArtifact no permitirá publicar una versión de paquete que ya esté presente en el repositorio externo. Para obtener más información, consulte Información general sobre paquetes. Sin embargo, en raras ocasiones, una interrupción del repositorio externo puede significar que CodeArtifact no tenga información actualizada sobre qué paquetes y versiones de paquetes están presentes en un repositorio externo. En este caso, CodeArtifact podría permitir que se publique una versión de paquete que normalmente rechazaría.

Disponibilidad de nuevas versiones de paquetes

Para que una versión de paquete en un repositorio público como npmjs.com esté disponible a través de un repositorio de CodeArtifact, primero debe añadirse a la caché de metadatos de un paquete regional. CodeArtifact mantiene esta caché en cada región AWS y contiene metadatos que describen el contenido de los repositorios públicos compatibles. Debido a esta caché, hay un retraso entre el momento en que se publica una nueva versión del paquete en un repositorio público y el momento en que está disponible en CodeArtifact. Este retraso varía según el tipo de paquete.

Para los paquetes npm, Python y Nuget, puede haber un retraso de hasta 30 minutos desde que se publique una nueva versión del paquete en npmjs.com, pypi.org o nuget.org y cuando esté disponible para su instalación desde un repositorio de CodeArtifact. CodeArtifact sincroniza automáticamente los metadatos de estos dos repositorios para garantizar que la caché esté actualizada.

Para los paquetes de Maven, puede haber un retraso de hasta 3 horas desde que se publique una nueva versión del paquete en un repositorio público hasta que esté disponible para su instalación desde un repositorio de CodeArtifact. CodeArtifact comprobará si hay nuevas versiones de un paquete como máximo una vez cada 3 horas. La primera solicitud de un nombre de paquete determinado después de que haya expirado la vida útil de la caché de 3 horas hará que todas las nuevas versiones de ese paquete se importen a la caché regional.

En el caso de los paquetes de Maven de uso común, las nuevas versiones suelen importarse cada 3 horas, ya que la alta tasa de solicitudes significa que la caché suele actualizarse tan pronto como haya expirado su vida útil. Para los paquetes que se utilizan con poca frecuencia, la caché no tendrá la última versión hasta que se solicite una versión del paquete desde un repositorio de CodeArtifact. En la primera solicitud, solo estarán disponibles en CodeArtifact las versiones previamente importadas, pero esta solicitud hará que se actualice la caché. En solicitudes posteriores, las nuevas versiones del paquete se añadirán a la memoria caché y estarán disponibles para su descarga.

Importación de versiones de paquetes con más de un activo

Tanto los paquetes de Maven como los de Python pueden tener varios activos por versión de paquete. Esto hace que la importación de paquetes de estos formatos sea más compleja que los paquetes npm y NuGet, que solo tienen un activo por versión de paquete. Para obtener descripciones de los activos que se importan para estos tipos de paquetes y de cómo están disponibles los activos recién agregados, consulte Solicitud de paquetes de Python desde conexiones ascendentes y externas y Solicitud de paquetes de Maven desde conexiones ascendentes y externas.

Orden de prioridad del repositorio ascendente

Cuando solicita una versión de paquete desde un repositorio con uno o más repositorios ascendentes, su prioridad corresponde al orden en que estaban listados al ejecutar el comando `create-repository` o `update-repository`. Cuando se encuentra la versión del paquete solicitada, la búsqueda se detiene, aunque no haya buscado en todos los repositorios ascendentes. Para obtener más información, consulte Añadir o eliminar repositorios ascendentes (AWS CLI).

Use el comando `describe-repository` para ver el orden de prioridad.

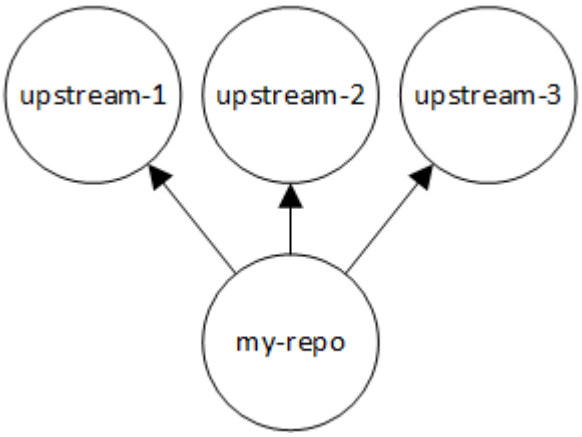
```
aws codeartifact describe-repository --repository my_repo --domain my_domain --domain-
owner 111122223333
```

El resultado puede ser el siguiente. Muestra que la prioridad del repositorio ascendente es `upstream-1` primero, `upstream-2` segundo y `upstream-3` tercero.

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

Ejemplo sencillo de orden de prioridad

En el siguiente diagrama, el repositorio `my_repo` tiene tres repositorios ascendentes. El orden de prioridad de los repositorios ascendentes es `upstream-1`, `upstream-2`, `upstream-3`.



Una solicitud de versión de paquete en `my_repo` busca en los repositorios en el siguiente orden hasta encontrarla o hasta que se devuelva al cliente una respuesta HTTP 404 `Not Found`:

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

Si se encuentra la versión del paquete, la búsqueda se detiene, aunque no haya buscado en todos los repositorios ascendentes. Por ejemplo, si la versión del paquete se encuentra en `upstream-1`, la búsqueda se detiene y CodeArtifact no busca en `upstream-2` o `upstream-3`.

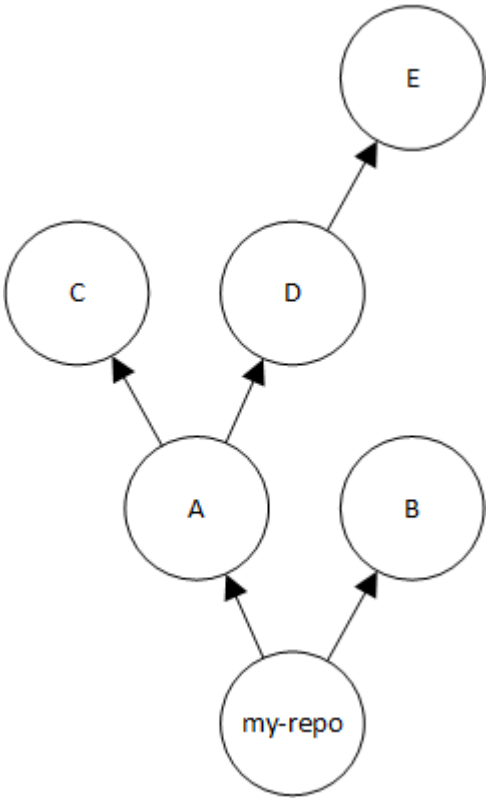
Cuando utiliza el comando AWS CLI `list-package-versions` para enumerar las versiones de los paquetes en `my_repo`, solo se busca en `my_repo`. No muestra las versiones de los paquetes en los repositorios ascendentes.

Ejemplo de orden de prioridad complejo

Si un repositorio ascendente tiene sus propios repositorios ascendentes, se utiliza la misma lógica para buscar la versión de un paquete antes de pasar al siguiente repositorio ascendente. Por ejemplo, supongamos que su repositorio `my_repo` tiene dos repositorios ascendentes, `A` y `B`. Si el repositorio `A` tiene repositorios ascendentes, una solicitud de una versión de paquete en `my_repo` busca primero en `my_repo`, en segundo lugar en `A` y luego en los repositorios ascendentes de `A`, y así sucesivamente.

En el siguiente diagrama, el repositorio `my_repo` contiene los repositorios ascendentes. El repositorio ascendente `A` tiene dos repositorios ascendentes y `D` un repositorio ascendente. Los

repositorios ascendentes del mismo nivel del diagrama aparecen en su orden de prioridad, de izquierda a derecha (el repositorio `A` tiene un orden de prioridad más alto que el repositorio `B` y el repositorio `C` tiene un orden de prioridad más alto que el repositorio `D`).



En este ejemplo, una solicitud de una versión de paquete en `my_repo` busca en los repositorios en el siguiente orden hasta que la encuentra o hasta que un administrador de paquetes devuelve una respuesta HTTP 404 `Not Found` al cliente:

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

Comportamiento de la API con los repositorios ascendentes

Cuando llama a determinadas API de CodeArtifact en repositorios que están conectados a repositorios ascendentes, el comportamiento puede variar según si los paquetes o las versiones

de los paquetes se almacenan en el repositorio de destino o en el repositorio ascendente. El comportamiento de estas API se documenta aquí.

Para obtener más información sobre las API de CodeArtifact, consulte la Referencia de las API de CodeArtifact.

La mayoría de las API que hacen referencia a un paquete o a una versión de paquete devolverán un error `ResourceNotFound` si la versión del paquete especificada no está presente en el repositorio de destino. Esto es cierto incluso si el paquete o la versión del paquete están presentes en un repositorio ascendente. De hecho, los repositorios ascendentes se ignoran al llamar a estas API. Estas API son:

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

Para demostrar este comportamiento, tenemos dos repositorios: `target-repo` y `upstream-repo`. `target-repo` está vacío y se ha configurado `upstream-repo` como un repositorio ascendente. `upstream-repo` contiene el paquete npm `lodash`.

Al llamar a la API `DescribePackageVersion` en `upstream-repo`, que contiene el paquete `lodash`, obtenemos el siguiente resultado:

```
{
    "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"
    }
```

Al llamar a la misma API en `target-repo`, que está vacía pero se ha configurado `upstream-repo` como una versión ascendente, obtenemos el siguiente resultado:

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

La API `CopyPackageVersions` se comporta de forma diferente. De forma predeterminada, la API `CopyPackageVersions` solo copia las versiones de los paquetes que están almacenadas en el repositorio de destino. Si la versión de un paquete está almacenada en el repositorio ascendente pero no en el repositorio de destino, no se copiará. Para incluir las versiones en paquete de los paquetes que se almacenan únicamente en el repositorio ascendente, establezca el valor de `includeFromUpstream` en `true` en su solicitud de API.

Para obtener información adicional sobre `CopyPackageVersions`, consulte Copiar paquetes entre repositorios.

Trabajando con paquetes en CodeArtifact

En estos temas se muestra cómo enumerar, copiar, eliminar y buscar paquetes mediante la CodeArtifact CLI y la API.

Temas

Información general sobre paquetes

Un paquete es un paquete de software y los metadatos necesarios para resolver las dependencias e instalar el software. En CodeArtifact, un paquete consta de un nombre de paquete, un espacio de nombres opcional, como `@types` in`@types/node`, un conjunto de versiones del paquete y metadatos a nivel de paquete, como las etiquetas npm.

Contenido

  - Latencia de publicación
- El estado de la versión del paquete
- Normalización del nombre del paquete, la versión del paquete y el nombre del activo

Formatos de paquetes admitidos

AWS CodeArtifact admite los formatos de paquetes npm, PyPI, Maven NuGety genéricos.

Publicación de paquetes

Puede publicar nuevas versiones de cualquier formato de paquete compatible en un CodeArtifact repositorio mediante herramientas comonpm,, `twine`Maven, `Gradle` y. `nuget` `dotnet`

Permisos de publicación

Su usuario o rol AWS Identity and Access Management (de IAM) debe tener permisos para publicar en el repositorio de destino. Se requieren los siguientes permisos para publicar paquetes:

- Maven: `codeartifact:PublishPackageVersion` y `codeartifact:PutPackageMetadata`
- npm: `codeartifact:PublishPackageVersion`
- NuGet: `codeartifact:PublishPackageVersion` y `codeartifact:ReadFromRepository`
- Python: `codeartifact:PublishPackageVersion`
- generic: `codeartifact:PublishPackageVersion`

En la lista de permisos anterior, su política de IAM debe especificar el recurso `package` para los permisos `codeartifact:PublishPackageVersion` y `codeartifact:PutPackageMetadata`. También debe especificar el recurso `repository` para el permiso `codeartifact:ReadFromRepository`.

Para obtener más información sobre los permisos en CodeArtifact, consulteReferencia de permisos de AWS CodeArtifact.

Sobrescribir los activos del paquete

No puede volver a publicar un activo de paquete que ya existe con un contenido diferente. Por ejemplo, suponga que ya publicó un paquete de Maven con un activo `mypackage-1.0.jar` JAR.

Solo puede volver a publicar ese activo si la suma de comprobación de los activos antiguos y nuevos es idéntica. Para volver a publicar el mismo recurso con contenido nuevo, elimine primero la versión del paquete con el comando delete-package-versions. Si intenta volver a publicar el mismo nombre de activo con un contenido diferente, se producirá un error de conflicto con el protocolo HTTP 409.

Para los formatos de paquetes que admiten varios activos (generic, PyPI y Maven), puede agregar nuevos activos con nombres diferentes a una versión de paquete existente, siempre que tenga los permisos necesarios. En el caso de los paquetes genéricos, puede añadir nuevos activos siempre que la versión del paquete esté en ese estado `Unfinished`. Como npm solo admite un activo por versión de paquete, para modificar una versión de paquete publicada de cualquier forma, primero debe eliminarla mediante delete-package-versions.

Si intenta volver a publicar un activo que ya existe (por ejemplo, `mypackage-1.0.jar`) y el contenido del activo publicado y el nuevo son idénticos, la operación se realizará correctamente porque la operación es idempotente.

Paquetes privados y repositorios públicos

CodeArtifact no publica los paquetes almacenados en CodeArtifact repositorios públicos como npmjs.com o Maven Central. CodeArtifact importa paquetes de repositorios públicos a un CodeArtifact repositorio, pero nunca los mueve en la otra dirección. Los paquetes que publiques en los CodeArtifact repositorios permanecen privados y solo están disponibles para las AWS cuentas, los roles y los usuarios a los que hayas concedido acceso.

Publicar versiones de paquetes parcheadas

A veces, es posible que desee publicar una versión de paquete modificada, posiblemente una que esté disponible en un repositorio público. Por ejemplo, es posible que haya encontrado un error en una de las dependencias críticas de una aplicación llamada `mydep 1.1`, y necesite solucionarlo antes de que el proveedor del paquete pueda revisar y aceptar el cambio. Como se ha descrito anteriormente, CodeArtifact impide publicar `mydep 1.1` en el CodeArtifact repositorio si se puede acceder al repositorio público desde el CodeArtifact repositorio a través de los repositorios anteriores y de una conexión externa.

Para solucionar este problema, publica la versión del paquete en un CodeArtifact repositorio diferente en el que no se pueda acceder al repositorio público. A continuación, utiliza la `copy-package-versions` API para copiar la versión parcheada de en el CodeArtifact repositorio desde el que la vayas `mydep 1.1` a consumir.

Límites de tamaño de los activos para su publicación

El tamaño máximo de un paquete que se puede publicar está limitado por la cuota máxima de tamaño del archivo de activos que se muestra en Cuotas en AWS CodeArtifact. Por ejemplo, no puede publicar una rueda JAR o Python de Maven que supere la cuota máxima de tamaño de su archivo de activos actual. Si necesitas almacenar activos más grandes CodeArtifact, solicita un aumento de cuota.

Además de la cuota máxima de tamaño de archivo de activos, el tamaño máximo de una solicitud de publicación para paquetes npm es de 2 GB. Este límite es independiente de la cuota máxima de tamaño del archivo de activos y no se puede aumentar con un aumento de cuota. En una solicitud de publicación de npm (HTTP PUT), los metadatos del paquete y el contenido del archivo tar del paquete npm se agrupan. Por este motivo, el tamaño máximo real de un paquete npm que se puede publicar varía y depende del tamaño de los metadatos incluidos.

> **Note**
>
> Los paquetes npm publicados están limitados a un tamaño máximo inferior a 2 GB.

Latencia de publicación

Las versiones de los paquetes publicadas en un CodeArtifact repositorio suelen estar disponibles para su descarga en menos de un segundo. Por ejemplo, si publicas una versión del paquete npm en CodeArtifact with`npm publish`, esa versión debería estar disponible para un `npm install` comando en menos de un segundo. Sin embargo, la publicación puede ser incoherente y, a veces, puede llevar más tiempo. Si debe utilizar una versión del paquete inmediatamente después de publicarla, vuelva a intentarlo para asegurarse de que la descarga es fiable. Por ejemplo, después de publicar la versión del paquete, repita la descarga hasta tres veces si la versión del paquete recién publicada no está disponible inicialmente en el primer intento de descarga.

> **Note**
>
> La importación de una versión de paquete desde un repositorio público suele tardar más que la publicación. Para obtener más información, consulte Latencia de conexión externa.

El estado de la versión del paquete

Cada versión del paquete CodeArtifact tiene un estado que describe el estado actual y la disponibilidad de la versión del paquete. Puede cambiar el estado de la versión del paquete en AWS CLI y el SDK. Para obtener más información, consulte Actualización del estado de la versión del paquete.

Los siguientes son valores posibles para el estado de la versión del paquete:

- Publicada: la versión del paquete se publicó correctamente y se puede solicitar mediante un administrador de paquetes. La versión del paquete se incluirá en las listas de versiones de los paquetes devueltas a los administradores de paquetes, por ejemplo, en la salida de `npm view <package-name> versions`. Todos los activos de la versión del paquete están disponibles en el repositorio.
- Inacabado: el cliente ha cargado uno o más activos para una versión de paquete, pero no los ha finalizado moviéndolos al estado `Published`. Actualmente, solo las versiones genéricas y de paquetes de Maven pueden tener un estado de `Unfinished`. En el caso de los paquetes de Maven, esto puede ocurrir cuando el cliente carga uno o más activos para una versión del paquete, pero no publica un archivo `maven-metadata.xml` para el paquete que incluye esa versión. Cuando una versión de un paquete de Maven esté inacabada, no se incluirá en las listas de versiones devueltas a los clientes como `mvn` o `gradle`, por lo que no podrá usarse como parte de una compilación. Los paquetes genéricos se pueden mantener en ese `Unfinished` estado de forma deliberada; para ello, se `unfinished` indica la marca al llamar a la PublishPackageVersionAPI. Un paquete genérico se puede cambiar al `Published` estado omitiendo la `unfinished` marca o llamando a la UpdatePackageVersionsStatusAPI.
- No listada: los activos de la versión del paquete están disponibles para su descarga desde el repositorio, pero la versión del paquete no se incluye en la lista de versiones devueltas a los administradores de paquetes. Por ejemplo, en el caso de un paquete npm, la salida de `npm view <package-name> versions` no incluirá la versión del paquete. Esto significa que la lógica de resolución de dependencias de npm no seleccionará la versión del paquete porque la versión no aparece en la lista de versiones disponibles. Sin embargo, si ya se hace referencia a la versión del paquete no listada en un archivo `npm package-lock.json`, aún se puede descargar e instalar, por ejemplo, cuando se ejecuta `npm ci`.
- Archivada: los activos de la versión del paquete ya no se pueden descargar. La versión del paquete no se incluirá en la lista de versiones devueltas a los administradores de paquetes. Como los activos no están disponibles, los clientes bloquean el consumo de la versión del paquete. Si la compilación de la aplicación depende de una versión actualizada a Archivada, la compilación

se interrumpirá si la versión del paquete no se ha almacenado en caché local. No puedes usar un administrador de paquetes o una herramienta de compilación para volver a publicar una versión de paquete archivada porque todavía está presente en el repositorio, pero puedes cambiar el estado de la versión del paquete a No listada o Publicada con la API. UpdatePackageVersionsStatus

- Eliminada: la versión del paquete no aparece en las listas y los activos no se pueden descargar del repositorio. La principal diferencia entre Descartado y Archivado es que, si el estado es Dispuesto, los activos de la versión del paquete se eliminarán permanentemente. CodeArtifact Por este motivo, no puede mover una versión de paquete de Eliminada a Archivada, No listada o Publicada. La versión del paquete ya no se puede utilizar porque se han eliminado los activos. Una vez que una versión del paquete se haya marcado como Eliminada, ya no se le facturará el almacenamiento de los activos del paquete.

Las versiones de paquete de todos los estados se devolverán de forma predeterminada cuando se llame list-package-versions sin `--status` parámetros.

Además de los estados enumerados anteriormente, la versión de un paquete también se puede eliminar con la DeletePackageVersionsAPI. Una vez eliminada, la versión de un paquete deja de estar en el repositorio y puede volver a publicarla libremente mediante un administrador de paquetes o una herramienta de compilación. Tras eliminar una versión de paquete, ya no se le facturará el almacenamiento de los activos de esa versión de paquete.

Normalización del nombre del paquete, la versión del paquete y el nombre del activo

CodeArtifact normaliza los nombres de los paquetes, las versiones de los paquetes y los nombres de los activos antes de almacenarlos, lo que significa que los nombres o las versiones CodeArtifact pueden ser diferentes del nombre o la versión proporcionados cuando se publicó el paquete. Para obtener más información sobre cómo se normalizan los nombres y las versiones CodeArtifact para cada tipo de paquete, consulte la siguiente documentación:

- Normalización de nombres de paquetes de Python
- Normalización del nombre, la versión y el nombre del activo del paquete NuGet

CodeArtifact no realiza la normalización en otros formatos de paquete.

Mostrar nombres de paquetes

Use el `list-packages` comando in CodeArtifact para obtener una lista de todos los nombres de paquetes de un repositorio. Este comando devuelve solo los nombres de los paquetes, no las versiones.

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

Resultado de ejemplo:

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

Enumerar los nombres de los paquetes de npm

Para enumerar solo los nombres de los paquetes npm, defina el valor de la opción `--format` en npm.

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

Para enumerar los paquetes npm en un espacio de nombres (alcance npm), utilice las opciones `--namespace` y `--format`.

Important

El valor de la opción `--namespace` no debe incluir el `@` inicial. Para buscar el espacio de nombres `@types`, defina el valor en *`tipos`*.

Note

La opción `--namespace` filtra por prefijo de espacio de nombres. Cualquier paquete npm con un alcance que comience con el valor pasado a la opción `--namespace` se devolverá en la respuesta `list-packages`.

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

Resultado de ejemplo:

```
{
    "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"

        }
```

```
    ]
}
```

Enumerar los nombres de los paquetes de Maven

Para enumerar solo los nombres de los paquetes Maven, defina el valor de la opción `--format` en `maven`. También debe especificar el ID del grupo Maven en la opción `--namespace`.

> **Note**
>
> La opción `--namespace` filtra por prefijo de espacio de nombres. Cualquier paquete npm con un alcance que comience con el valor pasado a la opción `--namespace` se devolverá en la respuesta `list-packages`.

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

Resultado de ejemplo:

```
{
    "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"
```

```
        }
    ]
}
```

Enumerar los nombres de los paquetes de Python

Para enumerar solo los nombres de los paquetes de Python, defina el valor de la opción `--format` en `pypi`.

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

Filtrar por prefijo de nombre de paquete

Para devolver paquetes que comiencen por una cadena específica, puede usar la opción `--package-prefix`.

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

Resultado de ejemplo:

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

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

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

```
        }
    ]
}
```

Combinaciones de opciones de búsqueda compatibles

Puede usar las opciones `--format`, `--namespace` y `--package-prefix` en cualquier combinación, excepto que `--namespace` no se puede usar por sí solo. La búsqueda de todos los paquetes npm con un alcance que comience por `@types` requiere que se especifique la opción `--format`. Usar `--namespace` por sí solo produce un error.

El uso de ninguna de las tres opciones también es compatible con `list-packages` y devolverá todos los paquetes de todos los formatos presentes en el repositorio.

Formatear salida

Puede usar los parámetros que están disponibles para todos los AWS CLI comandos para que la `list-packages` respuesta sea compacta y más legible. Utilice el parámetro `--query` para especificar el formato de cada versión del paquete devuelto. Utilice el parámetro `--output` para formatear la respuesta como texto sin formato.

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

Resultado de ejemplo:

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

Para obtener más información, consulte Control de la salida de comandos de la AWS CLI en la Guía del usuario de AWS Command Line Interface .

Valores predeterminados y otras opciones

De forma predeterminada, el número máximo de resultados devueltos por `list-packages` es 100. Puede cambiar este límite de resultados mediante la opción `--max-results`.

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

El valor máximo permitido de `--max-results` es de 1000. Para permitir la publicación de paquetes en repositorios con más de 1000 paquetes, `list-packages` admite la paginación mediante el campo `nextToken` de la respuesta. Si el número de paquetes del repositorio es superior al valor de `--max-results`, puede pasar el valor de `nextToken` a otra invocación de `list-packages` para obtener la siguiente página de resultados.

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

Listar las versiones de los paquetes

Use el `list-package-versions` comando in AWS CodeArtifact para obtener una lista de todas las versiones del nombre de un paquete en un repositorio.

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

Resultado de ejemplo:

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

Puede añadir el parámetro `--status` a la llamada `list-package-versions` para filtrar los resultados en función del estado de la versión del paquete. Para obtener más información sobre el estado de la versión del paquete, consulte El estado de la versión del paquete.

Puede paginar la respuesta `list-package-versions` utilizando los parámetros `--max-results` y `--next-token`. Para `--max-results`, especifique un número entero comprendido entre 1 y 1000 para especificar el número de resultados devueltos en una sola página. El valor predeterminado es 50. Para volver a las páginas siguientes, ejecute `list-package-versions` de nuevo y pase el valor `nextToken` recibido en el resultado del comando anterior a `--next-token`. Cuando no se utiliza la opción `--next-token`, siempre se devuelve la primera página de resultados.

El comando `list-package-versions` no muestra las versiones de los paquetes en los repositorios ascendentes. Sin embargo, se muestran las referencias a las versiones de paquetes de un repositorio principal que se copiaron en su repositorio durante una solicitud de versión de paquete. Para obtener más información, consulte Trabajar con repositorios ascendentes en CodeArtifact.

Enumerar las versiones del paquete npm

Para enumerar todas las versiones de un paquete npm, defina el valor de la opción `--format` en `npm`.

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

Para enumerar las versiones del paquete npm en un espacio de nombres específico (alcance npm), use la opción `--namespace`. El valor de la opción `--namespace` no debe incluir el `@` inicial. Para buscar el espacio de nombres `@types`, defina el valor en *`tipos`*.

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

```
 --namespace types
```

Enumerar las versiones de los paquetes de Maven

Para enumerar todas las versiones de un paquete de Maven, establezca el valor de la opción `--format` en `maven`. También debe especificar el ID del grupo Maven en la opción `--namespace`.

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

Ordenar versiones

`list-package-versions` puede generar versiones ordenadas en orden descendente según la hora de publicación (las versiones publicadas más recientemente aparecen primero). Utilice el parámetro `--sort-by` con un valor de `PUBLISHED_TIME`, de la siguiente manera.

```
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
```

Resultado de ejemplo:

```
{

  "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...."
}
```

Versión de visualización predeterminada

El valor devuelto para `defaultDisplayVersion` depende del formato del paquete:

- Para los paquetes genéricos, Maven y PyPI, es la versión del paquete publicada más recientemente.
- En el caso de los paquetes npm, es la versión a la que hace referencia la etiqueta `latest`. Si la etiqueta `latest` no está configurada, es la versión del paquete publicada más recientemente.

Formatear salida

Puede usar los parámetros que están disponibles para todos los AWS CLI comandos para que la `list-package-versions` respuesta sea compacta y más legible. Utilice el parámetro `--query` para especificar el formato de cada versión del paquete devuelto. Utilice el parámetro `--output` para dar formato a la respuesta como texto sin formato.

```
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]'
```

Resultado de ejemplo:

```
0.1.1
```

```
0.1.2
0.1.0
3.0.0
```

Para obtener más información, consulte Control de la salida de comandos de la AWS CLI en la Guía del usuario de AWS Command Line Interface .

Enumerar los activos de la versión del paquete

Un activo es un archivo individual (por ejemplo, un archivo npm o un `.tgz` archivo POM o JAR de Maven) almacenado y asociado a una versión de paquete. CodeArtifact Puede usar el comando `list-package-version-assets` para enumerar los activos de cada versión del paquete.

Ejecuta el `list-package-version-assets` comando para obtener la siguiente información sobre cada activo de tu AWS cuenta y de tu región actual AWS :

- Su nombre.
- Su tamaño, en bytes.
- Conjunto de valores hash que se utilizan para la validación de la suma de verificación.

Por ejemplo, utilice el siguiente comando para enumerar los activos del paquete `flatten-json` Python, versión `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
```

El ejemplo siguiente muestra la salida.

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

Enumerar los activos de un paquete npm

Un paquete npm siempre tiene un único activo con el nombre de `package.tgz`. Para enumerar los activos de un paquete npm con un alcance específico, incluya el alcance en la opción `--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
```

Enumerar los activos de un paquete Maven

Para enumerar los activos de un paquete Maven, incluya el espacio de nombres del paquete en la opción `--namespace`. Para enumerar los activos del paquete Maven: `commons-cli:commons-cli`

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

Descargar recursos de la versión del paquete

Un activo es un archivo individual (por ejemplo, un archivo npm o un `.tgz` archivo POM o JAR de Maven) almacenado y asociado a una versión de paquete. CodeArtifact Puede descargar los activos del paquete mediante `get-package-version-assets command`. Esto le permite recuperar activos sin utilizar un cliente administrador de paquetes como `npm` o `pip`. Para descargar un activo, debe proporcionar el nombre del activo, que se puede obtener mediante el comando `list-package-version-assets`; para obtener más información, consulte Enumerar los activos de la versión del paquete. El activo se descargará al almacenamiento local con el nombre de archivo que especifique.

En el siguiente ejemplo, se descarga el recurso *guava-27.1-jre.jar* del paquete de Maven *com.google.guava:guava* con la versión *27.1-jre*.

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

En este ejemplo, el nombre del archivo se especificó como *guava-27.1-jre.jar* en el último argumento del comando anterior, por lo que el activo descargado se denominará *guava-27.1-jre.jar*.

La salida del comando será:

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

> **Note**
>
> Para descargar activos de un paquete npm con alcance, incluya el alcance en la opción `--namespace`. Se debe omitir el símbolo `@` cuando se utilice `--namespace`. Por ejemplo, si el alcance es `@types`, utilice `--namespace types`.

La descarga de activos mediante el recurso del paquete `get-package-version-asset` requiere un permiso `codeartifact:GetPackageVersionAsset`. Para obtener más información sobre las políticas de permisos basadas en recursos, consulte Políticas basadas en recursos en la Guía del usuario de AWS Identity and Access Management .

Copiar paquetes entre repositorios

Puede copiar versiones de paquetes de un repositorio a otro en CodeArtifact. Esto puede resultar útil en situaciones como los flujos de trabajo de promoción de paquetes o el intercambio de versiones de paquetes entre equipos o proyectos. Los repositorios de origen y destino deben estar en el mismo dominio para copiar versiones de paquetes.

Permisos de IAM necesarios para copiar paquetes

Para copiar las versiones de los paquetes CodeArtifact, el usuario que realiza la llamada debe tener los permisos de IAM necesarios y la política basada en recursos asociada a los repositorios de origen y destino debe tener los permisos necesarios. Para obtener más información sobre las políticas de permisos y los repositorios basados en recursos, consulte. CodeArtifact Políticas de repositorios

El usuario que realiza la llamada a `copy-package-versions` debe tener el permiso `ReadFromRepository` en el repositorio de origen y el permiso `CopyPackageVersions` en el repositorio de destino.

El repositorio de origen debe tener permiso el `ReadFromRepository` y el repositorio de destino debe tener el permiso `CopyPackageVersions` asignado a la cuenta de IAM o al usuario que copia los paquetes. Las siguientes políticas son ejemplos de políticas de repositorio que se deben añadir al repositorio de origen o al repositorio de destino con el comando `put-repository-permissions-policy`. Sustituya *111122223333* por el identificador de la cuenta que realiza la llamada `copy-package-versions`.

> **Note**
>
> La llamada `put-repository-permissions-policy` sustituirá a la política de repositorio actual, si existe alguna. Puede usar el comando `get-repository-permissions-policy` para comprobar si existe una política; para obtener más información, consulte Leer una política. Si existe una política, es posible que desee agregarle estos permisos en lugar de reemplazarla.

Ejemplo de política de permisos del repositorio de origen

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

Ejemplo de política de permisos del repositorio de destino

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

```
        }
    ]
}
```

Copiar versiones de los paquetes

Utilice el `copy-package-versions` comando in CodeArtifact para copiar una o más versiones de paquetes de un repositorio de origen a un repositorio de destino del mismo dominio. El siguiente ejemplo copiará las versiones 6.0.2 y 4.0.0 de un paquete npm denominado `my-package` del repositorio `my_repo` al repositorio `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
```

Puede copiar varias versiones del mismo nombre de paquete en una sola operación. Para copiar versiones de distintos nombres de paquetes, debe llamar a `copy-package-versions` para cada una.

El comando anterior producirá el siguiente resultado, suponiendo que ambas versiones se hayan copiado correctamente.

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

Copiar un paquete de los repositorios originales

Normalmente, `copy-package-versions` solo busca en el repositorio especificado por la opción `--source-repository` las versiones que desee copiar. Sin embargo, puede copiar versiones

tanto del repositorio de origen como de sus repositorios anteriores mediante la opción `--include-from-upstream`. Si usas el CodeArtifact SDK, llama a la `CopyPackageVersions` API con el `includeFromUpstream` parámetro establecido en true. Para obtener más información, consulte Trabajar con repositorios ascendentes en CodeArtifact.

Copiar un paquete npm con alcance

Para copiar una versión del paquete npm en un alcance, utilice la opción `--namespace` para especificar el alcance. Por ejemplo, para copiar el paquete `@types/react`, utilice `--namespace types`. Se debe omitir el símbolo `@` cuando se utilice `--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
```

Copiar las versiones de los paquetes de Maven

Para copiar versiones de paquetes de Maven entre repositorios, especifique el paquete que desea copiar pasando el ID de grupo de Maven con la opción `--namespace` y el Maven ArtifactID con la opción `--name`. Por ejemplo, para copiar una sola versión de: `com.google.guava:guava`

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

Si se copia correctamente la versión del paquete, el resultado será similar al que se incluye a continuación.

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

```
}
```

Versiones que no existen en el repositorio de origen

Si especifica una versión que no existe en el repositorio de origen, se producirá un error en la copia. Si existen algunas versiones en el repositorio de origen y otras no, no se copiarán todas las versiones. En el siguiente ejemplo, la versión 0.2.0 del paquete npm `array-unique` está presente en el repositorio de origen, pero la versión 5.6.7 no:

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

El resultado en esta situación será similar al siguiente.

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

El código de error `SKIPPED` se usa para indicar que la versión no se copió en el repositorio de destino porque no se pudo copiar otra versión.

Versiones que ya existen en el repositorio de destino

Cuando la versión de un paquete se copia en un repositorio donde ya existe, CodeArtifact compara los activos del paquete y los metadatos a nivel de versión del paquete en los dos repositorios.

Si los activos y los metadatos de la versión del paquete son idénticos en los repositorios de origen y destino, no se realiza una copia, pero la operación se considera correcta. Esto significa que `copy-package-versions` es idempotente. Cuando esto ocurre, la versión que ya estaba presente en los repositorios de origen y destino no aparecerá en la salida de `copy-package-versions`.

En el siguiente ejemplo, hay dos versiones del paquete npm `array-unique` en el repositorio de origen `repo-1`. La versión 0.2.1 también está presente en el repositorio de destino `dest-repo` y la versión 0.2.0 no.

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

El resultado en esta situación será similar al siguiente.

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

La versión 0.2.0 aparece en la lista `successfulVersions` porque se copió correctamente del repositorio de origen al de destino. La versión 0.2.1 no se muestra en la salida porque ya estaba presente en el repositorio de destino.

Si los activos o los metadatos de la versión del paquete difieren en los repositorios de origen y destino, la operación de copia fallará. Puede usar el parámetro `--allow-overwrite` para forzar una sobrescritura.

Si existen algunas versiones en el repositorio de destino y otras no, no se copiarán todas las versiones. En el siguiente ejemplo, la versión 0.3.2 del paquete npm `array-unique` está presente en los repositorios de origen y de destino, pero el contenido de la versión del paquete es diferente. La versión 0.2.1 está presente en el repositorio de origen pero no en el de destino.

```
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
```

El resultado en esta situación será similar al siguiente.

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

La versión 0.2.1 está marcada como `SKIPPED` porque no se copió en el repositorio de destino. No se copió porque la copia de la versión 0.3.2 falló porque ya estaba presente en el repositorio de destino, pero no era idéntica en los repositorios de origen y destino.

Especificación de una versión de paquete

La revisión de la versión de un paquete es una cadena que especifica un conjunto específico de activos y metadatos para una versión de paquete. Puede especificar una revisión de la versión del paquete para copiar las versiones del paquete que se encuentran en un estado específico. Para especificar una revisión de la versión del paquete, utilice el parámetro `--version-revisions` para pasar una o más versiones del paquete separadas por comas y los pares de revisión de la versión del paquete al comando `copy-package-versions`.

> **Note**
>
> Debe especificar el parámetro `--versions` o `--version-revisions` con `copy-package-versions`. No puede especificar ambos.

El siguiente ejemplo solo copiará la versión 0.3.2 del paquete `my-package` si está presente en el repositorio de origen con la versión revisada del paquete `REVISION-1-SAMPLE-6C81EFF7DA55CC`.

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

En el siguiente ejemplo, se copian dos versiones del paquete `my-package`, la 0.3.2 y la 0.3.13. La copia solo se realizará correctamente si en el repositorio de origen la versión 0.3.2 de `my-package` tiene una revisión `REVISION-1-SAMPLE-6C81EFF7DA55CC` y la versión 0.3.13 tiene una revisión `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
```

Para buscar las revisiones de una versión de paquete, utilice el comando `describe-package-version` o `list-package-versions`.

Para obtener más información, consulta Revisión de la versión del paquete y consulta CopyPackageVersionla Referencia de la CodeArtifact API.

Copiar paquetes npm

Para obtener más información sobre `copy-package-versions` el comportamiento de los paquetes npm, consulta las etiquetas npm y la CopyPackageVersions API.

Eliminar un paquete

Puede eliminar una o más versiones de paquetes a la vez mediante el comando `delete-package-versions`. Para eliminar un paquete de un repositorio por completo, incluidas todas las versiones y la configuración asociadas, utilice el comando `delete-package`. Un paquete puede existir en un repositorio sin ninguna versión del paquete. Esto puede ocurrir cuando se eliminan todas las versiones mediante el comando `delete-package-versions` o si el paquete se creó sin ninguna versión mediante la operación de la API `put-package-origin-configuration` (consulte Edición de los controles de origen del paquete).

Temas

- Eliminación de un paquete (AWS CLI)
- Eliminar una versión del paquete (AWS CLI)
- Eliminación de un paquete (consola)
- Eliminación de una versión del paquete (consola)
- Eliminación de un paquete npm

- Eliminar un paquete de Maven

Eliminación de un paquete (AWS CLI)

Puede eliminar un paquete, incluidas todas sus versiones y configuraciones, mediante el comando `delete-package`. El siguiente ejemplo elimina el paquete PyPI nombrado `my-package` en el repositorio `my_repo` del dominio `my_domain`:

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

Resultado de ejemplo:

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

Para confirmar que el paquete se ha eliminado, ejecute `describe-package` para el mismo nombre de paquete:

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

Eliminar una versión del paquete (AWS CLI)

Puede eliminar una o más versiones de paquetes a la vez mediante el comando `delete-package-versions`. El siguiente ejemplo elimina las versiones `4.0.0`, `4.0.1` y `5.0.0` del paquete PyPI denominado `my-package` en `my_repo` en el dominio `my_domain`:

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

Resultado de ejemplo:

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

Puede confirmar que las versiones se eliminaron ejecutando `list-package-versions` para el mismo nombre de paquete:

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

Eliminación de un paquete (consola)

1. Abre la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Repositories (Repositorios).
3. Elija el repositorio del que desea eliminar un paquete.

4. Elija el paquete desea eliminar.
5. Seleccione Eliminar paquete.

Eliminación de una versión del paquete (consola)

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Repositories (Repositorios).
3. Elija el repositorio del que desea eliminar versiones de paquetes.
4. Elija el paquete del que desea eliminar versiones.
5. Seleccione la versión del paquete que desea eliminar.
6. Elija Eliminar.

> **Note**
>
> En la consola, solo puede eliminar una versión del paquete a la vez. Para eliminar más de una a la vez, utilice la CLI.

Eliminación de un paquete npm

Para eliminar un paquete npm o versiones de paquetes individuales, defina la opción `--format` en npm. Para eliminar una versión de paquete en un paquete npm con alcance, utilice la opción `--namespace` para especificar el alcance. Por ejemplo, para eliminar el paquete `@types/react`, utilice `--namespace types`. Omita el símbolo `@` cuando utilice `--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
```

Para eliminar el paquete `@types/react`, incluidas todas sus versiones:

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

Eliminar un paquete de Maven

Para eliminar un paquete de Maven o versiones de paquetes individuales, defina la opción `--format` en `maven` y especifique el paquete que desea eliminar pasando el ID de grupo de Maven con la opción `--namespace` y el Maven ArtifactID con la opción `--name`. Por ejemplo, a continuación se muestra cómo eliminar una única versión de `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
```

En el siguiente ejemplo se muestra cómo eliminar el paquete `com.google.guava:guava`, incluidas todas sus versiones:

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

Ver y actualizar los detalles y las dependencias de la versión del paquete

Puede ver información sobre la versión de un paquete, incluidas las dependencias, en CodeArtifact. También puede actualizar el estado de una versión del paquete. Para obtener más información sobre el estado de la versión del paquete, consulte El estado de la versión del paquete.

Ver detalles de la versión del paquete

Utilice el comando `describe-package-version` para ver detalles sobre versiones de paquetes. Los detalles de la versión del paquete se extraen de un paquete cuando se publica en CodeArtifact. Los detalles de los diferentes paquetes varían y dependen de sus formatos y de la cantidad de información que les hayan agregado sus autores.

La mayor parte de la información de la salida del comando `describe-package-version` depende del formato del paquete. Por ejemplo, `describe-package-version` extrae la información de un paquete npm de su archivo `package.json`. La revisión la crea CodeArtifact. Para obtener más información, consulte Especificación de una versión de paquete.

Dos versiones de paquetes con el mismo nombre pueden estar en el mismo repositorio si cada una está en espacios de nombres diferentes. Utilice el parámetro `--namespace` opcional para especificar un espacio de nombres. Para obtener más información, consulte Ver los detalles de la versión del paquete npm o Ver los detalles de la versión del paquete Maven.

El siguiente ejemplo devuelve detalles sobre la versión `1.9.0` de un paquete de Python denominado `pyhamcrest` que se encuentra en el repositorio `my_repo`.

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

El resultado puede tener el siguiente aspecto.

```
{
  "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"
}
```

Ver los detalles de la versión del paquete npm

Para ver los detalles sobre la versión de un paquete npm, defina el valor de la opción `--format` en **npm**. Si lo desea, incluya el espacio de nombres de la versión del paquete (alcance npm) en la opción `--namespace`. El valor de la opción `--namespace` no debe incluir el `@` inicial. Para buscar el espacio de nombres `@types`, defina el valor en *`tipos`*.

A continuación, se muestran detalles sobre la versión `4.41.5` de un paquete npm denominado `webpack` en el alcance `@types`.

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

El resultado puede tener el siguiente aspecto.

```
{
  "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"
  }
}
```

Ver los detalles de la versión del paquete Maven

Para ver los detalles sobre la versión de un paquete de Maven, establezca el valor de la opción `--format` en `maven` e incluya el espacio de nombres de la versión del paquete en la opción `--namespace`.

El siguiente ejemplo devuelve detalles sobre la versión `1.2` de un paquete de Maven denominado `commons-rng-client-api` que se encuentra en el espacio de nombres `org.apache.commons` y en el repositorio `my_repo`.

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

El resultado puede tener el siguiente aspecto.

```
{
  "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 no extrae la información detallada de la versión del paquete de los archivos POM principales. Los metadatos de una versión de paquete determinada solo incluirán información en el POM para esa versión exacta del paquete, no para el POM principal o cualquier otro POM al que se haga referencia transitivamente mediante la etiqueta `parent` POM. Esto significa que la salida de `describe-package-version` omitirá metadatos (como información de licencia) para las versiones del paquete Maven que dependen de una referencia `parent` para contener estos metadatos.

Ver las dependencias de la versión del paquete

Use el comando `list-package-version-dependencies` para obtener una lista de las dependencias de la versión de un paquete. El siguiente comando muestra las dependencias de un paquete npm denominado `my-package`, versión `4.41.5`, en el repositorio `my_repo`, en el dominio `my_domain`.

```
aws codeartifact list-package-version-dependencies --domain my_domain --domain-
owner 111122223333 --repository my_repo \
```

```
--format npm --package my-package --package-version 4.41.5
```

El resultado puede tener el siguiente aspecto.

```
{
  "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"
}
```

Para ver el rango de valores admitidos para el campo DependencyType, consulta el tipo de PackageDependencydatos en la API. CodeArtifact

Ver el archivo readme de la versión del paquete

Algunos formatos de paquete, como npm, incluyen un archivo `README`. Use `get-package-version-readme` para obtener el archivo `README` de una versión de paquete. El siguiente comando devuelve el archivo `README` de un paquete npm llamado `my-package`, versión `4.41.5`, en el repositorio `my_repo`, en el dominio `my_domain`.

> **Note**
>
> CodeArtifact no admite la visualización de archivos readme de paquetes genéricos o de Maven.

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

El resultado puede tener el siguiente aspecto.

```
{
  "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"
}
```

Actualización del estado de la versión del paquete

Cada versión del paquete CodeArtifact tiene un estado que describe el estado actual y la disponibilidad de la versión del paquete. Puede cambiar el estado de la versión del paquete mediante la consola AWS CLI y la consola.

> **Note**
>
> Para obtener más información sobre el estado de la versión del paquete, incluida una lista de los estados disponibles, consulte El estado de la versión del paquete.

Actualizar el estado de la versión del paquete

Establecer el estado de la versión de un paquete permite controlar cómo se puede usar una versión del paquete sin eliminarla por completo del repositorio. Por ejemplo, cuando la versión de un paquete tiene el estado de `Unlisted`, se puede seguir descargando de forma normal, pero no aparecerá en las listas de versiones de paquetes devueltas a comandos como `npm view`. La

UpdatePackageVersionsStatus API permite configurar el estado de la versión del paquete de varias versiones del mismo paquete en una sola llamada a la API. Para obtener una descripción de los diferentes estados, consulte Información general sobre paquetes.

Utilice el comando `update-package-versions-status` para cambiar el estado de una versión de paquete a `Published`, `Unlisted`, o `Archived`. Para ver los permisos de IAM necesarios para usar el comando, consulte Permisos de IAM necesarios para actualizar el estado de una versión de paquete. El siguiente ejemplo establece el estado de la versión 4.1.0 del paquete npm `chalk` en `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
```

Resultado de ejemplo:

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

En este ejemplo se utiliza un paquete npm, pero el comando funciona de forma idéntica en otros formatos. Se pueden mover varias versiones al mismo estado de destino con un solo comando; consulte el siguiente ejemplo.

```
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
```

Resultado de ejemplo:

```
{
    "successfulVersions": {
        "4.1.0": {
            "revision": "25/UjBleHs1DZewk+zozoeqH/R80Rc9gL1P8vbzVMJ4=",
```

```
            "status": "Archived"
        },
        "4.1.1": {
            "revision": "+Oz8skWbwY3k8M6SrNIqNj6bVH/ax+CxvkJx+No5j8I=",
            "status": "Archived"
        }
    },
    "failedVersions": {}
}
```

Tenga en cuenta que, una vez publicada, la versión de un paquete no se puede volver a mover al estado `Unfinished`, por lo que este estado no está permitido como valor para el parámetro `--target-status`. Para mover la versión del paquete al estado `Disposed`, utilice el comando `dispose-package-versions` en su lugar, tal como se describe a continuación.

Permisos de IAM necesarios para actualizar el estado de una versión de paquete

Para llamar a `update-package-versions-status` para un paquete, debe tener el permiso `codeartifact:UpdatePackageVersionsStatus` en el recurso del paquete. Esto significa que puede otorgar permiso para llamar a `update-package-versions-status` por paquete. Por ejemplo, una política de IAM que conceda permiso para llamar a `update-package-versions-status` en el paquete npm *`chalk`* incluiría una declaración como la siguiente.

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

Se está actualizando el estado de un paquete npm específico

Para actualizar el estado de la versión del paquete de una versión del paquete npm con un alcance, utilice el parámetro `--namespace`. Por ejemplo, para eliminar la versión 8.0.0 de `@nestjs/core`, utilice el siguiente comando.

```
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
```

Estado de actualización de un paquete de Maven

Los paquetes de Maven siempre tienen un ID de grupo, que se denomina espacio de nombres en. CodeArtifact Utilice el parámetro `--namespace` para especificar el ID de grupo de Maven al llamar a `update-package-versions-status`. Por ejemplo, para archivar la versión 2.13.1 del paquete Maven `org.apache.logging.log4j:log4j`, utilice el siguiente comando.

```
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
```

Especificación de una versión de paquete

La revisión de la versión de un paquete es una cadena que especifica un conjunto específico de activos y metadatos para una versión de paquete. Puede especificar una revisión de la versión del paquete para actualizar el estado de las versiones del paquete que se encuentran en un estado específico. Para especificar una revisión de la versión del paquete, utilice el parámetro `--version-revisions` para pasar una o más versiones del paquete separadas por comas y los pares de revisiones de versiones del paquete. El estado de la versión de un paquete solo se actualizará si la revisión actual de la versión del paquete coincide con el valor especificado.

> **Note**
>
> El parámetro `–-versions` también debe definirse cuando se utilice el parámetro `--version-revisions`.

```
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
```

Para actualizar varias versiones con un solo comando, pase a las opciones una lista separada por comas de pares de versiones y revisiones a las opciones `--version-revisions`. El siguiente

comando de ejemplo define dos pares diferentes de versiones de paquetes y revisiones de versiones de paquetes.

```
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
```

Resultado de ejemplo:

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

Al actualizar varias versiones de paquetes, las versiones que se han pasado a `--version-revisions` deben ser las mismas que las versiones pasadas a `--versions`. Si se especifica una revisión de forma incorrecta, el estado de esa versión no se actualizará.

Uso del parámetro de estado esperado

El comando `update-package-versions-status` proporciona el parámetro `--expected-status` que permite especificar el estado actual esperado de una versión del paquete. Si el estado actual no coincide con el valor al que se ha pasado a `--expected-status`, el estado de esa versión del paquete no se actualizará.

Por ejemplo, en *my_repo*, las versiones 4.0.0 y 4.1.0 del paquete npm `chalk` tienen actualmente un estado de `Published`. Una llamada a `update-package-versions-status` que especifique el estado esperado de no `Unlisted` podrá actualizar ambas versiones del paquete debido a la discordancia de estados.

```
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
```

Resultado de ejemplo:

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

Errores con las versiones individuales de los paquetes

Existen varios motivos por los que el estado de la versión de un paquete no se actualiza cuando se llama a `update-package-versions-status`. Por ejemplo, es posible que la revisión de la versión del paquete se haya especificado incorrectamente o que el estado esperado no coincida con el estado actual. En estos casos, la versión se incluirá en el mapa `failedVersions` de la respuesta de la API. Si una versión falla, es posible que se omitan las demás versiones especificadas en la misma llamada a `update-package-versions-status` y que su estado no se actualice. Estas versiones también se incluirán en el mapa `failedVersions` con un `errorCode` de `SKIPPED`.

En la implementación actual de `update-package-versions-status`, si no se puede cambiar el estado de una o más versiones, se omitirán todas las demás versiones. Es decir, todas las versiones se actualizan correctamente o no se actualiza ninguna versión. Este comportamiento no está garantizado en el contrato de la API; en el futuro, algunas versiones podrían funcionar correctamente y otras no en una sola llamada a `update-package-versions-status`.

El siguiente comando de ejemplo incluye un error en la actualización del estado de la versión provocado por una discordancia entre las versiones de un paquete. Ese error de actualización provoca que se omita otra llamada de actualización del estado de la versión.

```
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
```

Resultado de ejemplo:

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

Eliminación de las versiones de los paquetes

El estado `Disposed` del paquete tiene un comportamiento similar al del paquete`Archived`, excepto que los activos del paquete se eliminarán permanentemente CodeArtifact para que el almacenamiento de activos ya no se facture a la cuenta del propietario del dominio. Para obtener más información sobre el estado de cada versión del paquete, consulte El estado de la versión del paquete. Para cambiar el estado de una versión de paquete a `Disposed`, use el comando `dispose-package-versions`. Esta capacidad es independiente de la de `update-package-versions-status` debido a que la eliminación de una versión de paquete no es reversible. Como los activos del paquete se eliminarán, el estado de la versión no se puede volver a cambiar a `Archived`, `Unlisted` o `Published`. La única acción que se puede realizar en una versión de paquete que se ha eliminado es eliminarla mediante el comando `delete-package-versions`.

Para llamar a `dispose-package-versions` correctamente, la entidad principal de IAM que realiza la llamada debe tener el permiso `codeartifact:DisposePackageVersions` sobre el recurso del paquete.

El comportamiento del comando `dispose-package-versions` es similar a `update-package-versions-status`, incluido el comportamiento de las opciones `--version-revisions` y `--expected-status` que se describen en las secciones de revisión de versión y estado esperado. Por ejemplo, el siguiente comando intenta eliminar la versión de un paquete, pero falla debido a un estado esperado que no coincide.

```
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
```

Resultado de ejemplo:

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

Si se vuelve a ejecutar el mismo comando con un `--expected-status` de `Published`, la eliminación se realizará correctamente.

```
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
```

Resultado de ejemplo:

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

```
}
```

Edición de los controles de origen del paquete

En AWS CodeArtifact, las versiones de los paquetes se pueden añadir a un repositorio publicándolas directamente, extrayéndolas de un repositorio anterior o incorporándolas desde un repositorio público externo. Permitir que las versiones en paquetes de un paquete se agreguen mediante publicación directa o ingestión desde repositorios públicos hace que sea vulnerable a un ataque de sustitución de dependencias. Para obtener más información, consulte Ataques de sustitución de dependencias. Para protegerle de un ataque de sustitución de dependencias, puede configurar los controles de origen de los paquetes de un repositorio para limitar la forma en que se pueden añadir las versiones de ese paquete al repositorio.

Cualquier equipo que desee permitir que las nuevas versiones de diferentes paquetes provengan tanto de fuentes internas, como la publicación directa, como de fuentes externas, como los repositorios públicos, debería considerar la posibilidad de configurar los controles de origen de los paquetes. De forma predeterminada, los controles de origen de los paquetes se configurarán en función de cómo se añada la primera versión del paquete al repositorio. Para obtener información sobre la configuración del control de origen del paquete y sus valores predeterminados, consulte Configuración de control de origen del paquete.

Para eliminar el registro del paquete después de utilizar la operación de la API `put-package-origin-configuration`, utilice `delete-package` (consulte Eliminar un paquete).

Escenarios comunes de control de acceso a paquetes

En esta sección se incluyen algunos escenarios comunes cuando se agrega una versión de paquete a un CodeArtifact repositorio. Los ajustes de control de origen del paquete se establecerán para los paquetes nuevos en función de cómo se añada la primera versión del paquete.

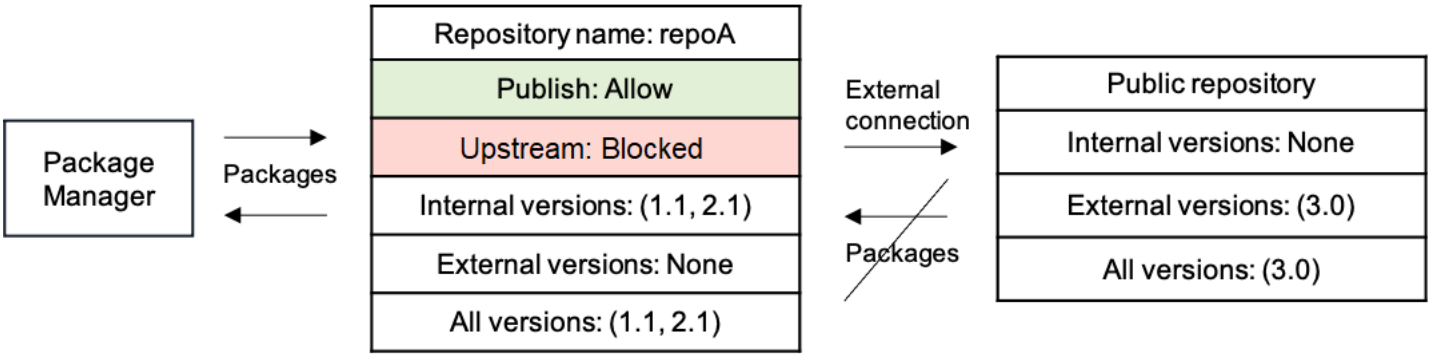
En los siguientes escenarios, un paquete interno es un paquete que se publica directamente desde un administrador de paquetes en su repositorio, como un paquete que usted o su equipo crean y mantienen. Un paquete externo es un paquete que existe en un repositorio público y que puede incorporarse a su repositorio mediante una conexión externa.

Se publica una versión de paquete externo para un paquete interno existente

En este escenario, consideremos un paquete interno, packageA. Tu equipo publica la primera versión del paquete del paquete A en un CodeArtifact repositorio. Como esta es la primera versión del paquete, la configuración del control de origen del paquete se establece automáticamente en

Publish: Allow and Upstream: Block. Una vez que el paquete existe en tu repositorio, se publica un paquete con el mismo nombre en un repositorio público que está conectado a tu CodeArtifact repositorio. Podría tratarse de un intento de ataque de sustitución de dependencias contra el paquete interno, o podría ser simplemente una coincidencia. En cualquier caso, los controles de origen de los paquetes están configurados para bloquear la ingesta de la nueva versión externa y protegerse así de un posible ataque.

En la siguiente imagen, RePoA es tu CodeArtifact repositorio con una conexión externa a un repositorio público. Su repositorio contiene las versiones 1.1 y 2.1 de packageA, pero la versión 3.0 está publicada en el repositorio público. Normalmente, repoA ingiere la versión 3.0 después de que un administrador de paquetes solicite el paquete. Como la ingesta de paquetes está configurada como Bloquear, la versión 3.0 no se ingiere en su CodeArtifact repositorio y no está disponible para los administradores de paquetes conectados a él.

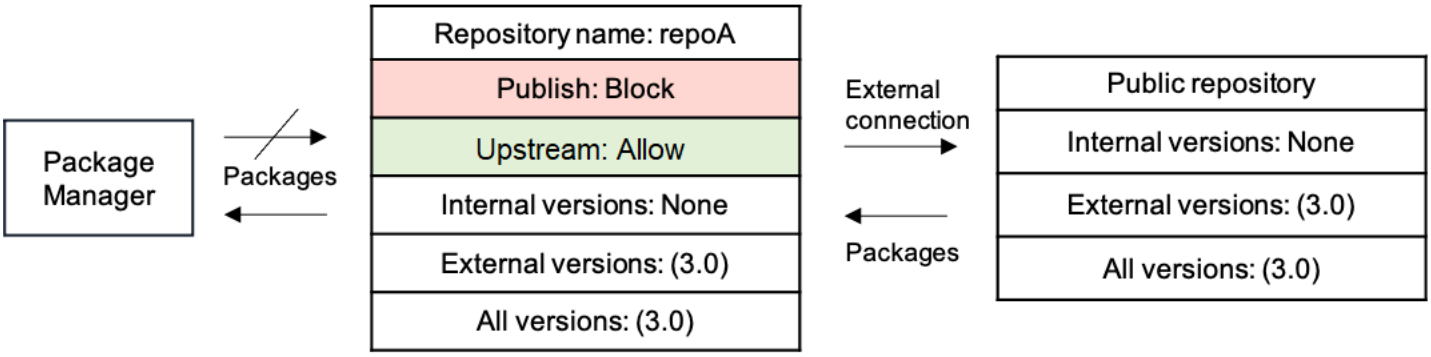


Se publica una versión de paquete interno para un paquete externo existente

En este escenario, un paquete, packageB, existe externamente en un repositorio público que usted ha conectado a su repositorio. Cuando un administrador de paquetes conectado a su repositorio solicita el packageB, la versión del paquete se introduce en su repositorio desde el repositorio público. Como esta es la primera versión de paquete de packageB que se agrega a su repositorio, los ajustes de origen del paquete están configurados como Publish: BLOCK y Upstream: ALLOW. Más adelante, intenta publicar una versión con el mismo nombre de paquete en el repositorio. O bien no conoce el paquete público y está intentando publicar un paquete no relacionado con el mismo nombre, o bien está intentando publicar una versión parcheada, o bien está intentando publicar directamente la versión exacta del paquete que ya existe externamente. CodeArtifact rechazará la versión que está intentando publicar, pero le permitirá anular el rechazo de forma explícita y publicar la versión si es necesario.

En la siguiente imagen, RePoA es tu CodeArtifact repositorio con una conexión externa a un repositorio público. Su repositorio contiene la versión 3.0 que ingirió del repositorio público. Quiere

publicar la versión 1.1 en su repositorio. Normalmente, podría publicar la versión 1.2 en repoA, pero como la publicación está configurada en Block, la versión 1.2 no se puede publicar.



Publicar una versión parcheada de un paquete externo existente

En este escenario, un paquete, packageB, existe externamente en un repositorio público que usted ha conectado a su repositorio. Cuando un administrador de paquetes conectado a su repositorio solicita el packageB, la versión del paquete se introduce en su repositorio desde el repositorio público. Como esta es la primera versión de paquete de packageB que se agrega a su repositorio, los ajustes de origen del paquete están configurados como Publish: BLOCK y Upstream: ALLOW. Su equipo decide que necesita publicar las versiones de los paquetes parcheados de este paquete en el repositorio. Para poder publicar las versiones de los paquetes directamente, su equipo cambia la configuración del control de origen de los paquetes a Publish: ALLOW y Upstream: BLOCK. Las versiones de este paquete ahora se pueden publicar directamente en su repositorio e ingerir desde los repositorios públicos. Una vez que su equipo publique las versiones de los paquetes parcheados, cambiará la configuración de origen del paquete a Publish: BLOCK and Upstream: ALLOW.

Configuración de control de origen del paquete

Con los controles de origen de los paquetes, puede configurar cómo se pueden añadir las versiones de los paquetes a un repositorio. Las siguientes listas incluyen la configuración y los valores de control de origen de los paquetes disponibles.

Publicación

Esta configuración configura si las versiones de los paquetes se pueden publicar directamente en el repositorio mediante administradores de paquetes o herramientas similares.

- ALLOW: las versiones de los paquetes se pueden publicar directamente.
- BLOCK: las versiones de los paquetes no se pueden publicar directamente.

Repositorios ascendentes

Esta configuración configura si las versiones de los paquetes pueden ingerirse desde repositorios públicos externos o conservarse desde repositorios originales cuando lo solicite un administrador de paquetes.

- PERMITIR: Se puede conservar cualquier versión de paquete de otros CodeArtifact repositorios configurados como repositorios ascendentes o se puede ingerir desde una fuente pública con una conexión externa.
- BLOCK: Las versiones de los paquetes no se pueden conservar de otros CodeArtifact repositorios configurados como repositorios ascendentes ni se pueden ingerir desde una fuente pública con una conexión externa.

Configuración de control de origen de paquetes predeterminada

Los controles de origen de los paquetes predeterminados se basarán en la forma en que se añada la primera versión de ese paquete al repositorio.

> **Note**
>
> Los paquetes que existían en CodeArtifact los repositorios aproximadamente antes de mayo de 2022 tendrán los controles de origen predeterminados Publish: ALLOW y Upstream: ALLOW. Los controles de origen de los paquetes deben configurarse manualmente para dichos paquetes. Desde entonces, los valores predeterminados actuales se han establecido en los paquetes nuevos y comenzaron a aplicarse cuando se lanzó la característica el 14 de julio de 2022. Para obtener más información sobre la configuración de controles de origen de paquetes, consulte Edición de los controles de origen del paquete.

- Si un administrador de paquetes publica directamente la primera versión del paquete, la configuración será Publish: ALLOW y Upstream: BLOCK.
- Si la primera versión del paquete proviene de una fuente pública, los ajustes serán Publish: ALLOW y Upstream: BLOCK.

Edición de los controles de origen del paquete

Los controles de origen de los paquetes se configuran automáticamente en función de cómo se agrega la primera versión del paquete al repositorio; para obtener más información, consulte Configuración de control de origen de paquetes predeterminada. Para añadir o editar los controles de origen de los paquetes de un CodeArtifact repositorio, lleve a cabo los pasos del siguiente procedimiento.

Para añadir o editar los controles de origen del paquete (consola)

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Repositorios y elija el repositorio que contiene el paquete que desea editar.
3. En la tabla Paquetes, busque y seleccione el paquete que desee editar.
4. En la página de resumen del paquete, en controles de Origin, seleccione Editar.
5. En Editar controles de origen, seleccione los controles de origen del paquete que desea configurar para este paquete. Los dos ajustes de control de origen del paquete, Publish y Upstream, deben configurarse al mismo tiempo.

   - Para permitir la publicación directa de las versiones de los paquetes, en Publicar, seleccione Permitir. Para bloquear la publicación de versiones de paquetes, seleccione Bloquear.
   - Para permitir la ingesta de paquetes de repositorios externos y la extracción de paquetes de repositorios anteriores, en Recursos ascendentes, seleccione Permitir. Para bloquear la ingestión y extracción de versiones de paquetes desde repositorios externos y ascendentes, seleccione Bloquear.

Para añadir o editar los controles de origen del paquete (AWS CLI)

1. Si no lo ha hecho, configúrela AWS CLI siguiendo los pasos que se indican enConfiguración con AWS CodeArtifact.
2. Utilice el comando `put-package-origin-configuration` para añadir o editar los controles de origen de los paquetes. Sustituya los campos siguientes:

   - Sustituya *`my_domain`* por el CodeArtifact dominio que contiene el paquete que desea actualizar.

- Sustituya *my_repo* por el CodeArtifact repositorio que contiene el paquete que desea actualizar.
- Reemplace *npm* con el formato de paquete que desea actualizar.
- Reemplace *my_package* con el nombre del paquete que desea actualizar.
- Sustituya *ALLOW* y *BLOCK* por la configuración de control de origen del paquete que desee.

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

Repositorios editoriales y originales

CodeArtifact no permite publicar versiones de paquetes que estén presentes en repositorios ascendentes o repositorios públicos accesibles. Por ejemplo, supongamos que desea publicar un paquete de Maven `com.mycompany.mypackage:1.0` en un repositorio `myrepo` y `myrepo` tiene un repositorio ascendente con una conexión externa al central de Maven. Considere los siguientes escenarios.

1. Los ajustes de control de origen de los paquetes de `com.mycompany.mypackage` son Publish: ALLOW y Upstream: ALLOW. Si `com.mycompany.mypackage:1.0` está presente en el repositorio principal o en Maven Central, CodeArtifact rechaza cualquier intento de publicar en `myrepo` él con un error de conflicto 409. Aún puede publicar una versión diferente, por ejemplo `com.mycompany.mypackage:1.1`.
2. Los ajustes de control de origen de los paquetes de `com.mycompany.mypackage` son Publish: ALLOW y Upstream: BLOCK. Puede publicar en su repositorio cualquier versión de `com.mycompany.mypackage` que aún no exista porque no se puede acceder a las versiones de los paquetes.
3. Los ajustes de control de origen de los paquetes de `com.mycompany.mypackage` son Publish: BLOCK y Upstream: ALLOW. No puede publicar ninguna versión del paquete directamente en su repositorio.

Trabajando con dominios en CodeArtifact

CodeArtifact los dominios facilitan la administración de varios repositorios en una organización. Puede utilizar un dominio para aplicar permisos en numerosos repositorios que sean propiedad de diferentes cuentas de AWS. Un activo se almacena solo una vez en un dominio, incluso si está disponible en varios repositorios.

Aunque puede tener varios dominios, le recomendamos un único dominio de producción que contenga todos los artefactos publicados para que sus equipos de desarrollo puedan encontrar y compartir paquetes. Puede usar un segundo dominio de preproducción para probar los cambios en la configuración del dominio de producción.

En estos temas se describe cómo utilizar la CodeArtifact consola AWS CLI, la consola y cómo AWS CloudFormation crear o configurar CodeArtifact dominios.

Temas

Información general del dominio

Cuando trabajas con ellos CodeArtifact, los dominios son útiles para lo siguiente:

- Almacenamiento deduplicado: un activo solo debe almacenarse una vez en un dominio, incluso si está disponible en 1 o 1000 repositorios. Esto significa que solo paga por el almacenamiento una vez.
- Copia rápida: cuando extraes paquetes de un CodeArtifact repositorio principal a uno descendente o utilizas la CopyPackageVersions API, solo se deben actualizar los registros de metadatos. No se copia ningún activo. Esto agiliza la configuración de un nuevo repositorio para su puesta en escena o pruebas. Para obtener más información, consulte Trabajar con repositorios ascendentes en CodeArtifact.

- Fácil de compartir entre repositorios y equipos: todos los activos y metadatos de un dominio se cifran con una sola clave AWS KMS key (clave KMS). No es necesario administrar una clave para cada repositorio ni permitir que varias cuentas accedan a una sola clave.
- Aplicar la política en varios repositorios: el administrador del dominio puede aplicar la política en todo el dominio. Esto incluye restringir qué cuentas tienen acceso a los repositorios del dominio y quién puede configurar las conexiones a los repositorios públicos para utilizarlos como fuentes de paquetes. Para obtener más información, consulte Políticas de dominio.
- Nombres de repositorio únicos: el dominio proporciona un espacio de nombres para los repositorios. Los nombres de los repositorios solo tienen que ser exclusivos dentro del dominio. Debe utilizar nombres significativos que sean fáciles de entender.

Los nombres de dominios deben ser únicos dentro de una cuenta.

No puede crear un repositorio sin un dominio. Cuando usas la CreateRepositoryAPI para crear un repositorio, debes especificar un nombre de dominio. No puede mover un repositorio de un dominio a otro.

Un repositorio puede ser propiedad de la misma AWS cuenta propietaria del dominio o de una cuenta diferente. Si las cuentas propietarias son diferentes, se debe conceder el permiso `CreateRepository` sobre el recurso del dominio a la cuenta propietaria del repositorio. Para ello, añada una política de recursos al dominio mediante el PutDomainPermissionsPolicycomando.

Si bien una organización puede tener varios dominios, se recomienda tener un único dominio de producción que contenga todos los artefactos publicados para que los equipos de desarrollo puedan encontrar y compartir paquetes en toda su organización. Un segundo dominio de preproducción puede resultar útil para probar los cambios en la configuración del dominio de producción.

Dominios entre cuentas

Los nombres de dominio solo deben ser únicos dentro de una cuenta, lo que significa que puede haber varios dominios dentro de una región que tengan el mismo nombre. Por ello, si desea acceder a un dominio que es propiedad de una cuenta en la que no está autenticado, debe proporcionar el ID del propietario del dominio junto con el nombre del dominio tanto en la CLI como en la consola. Vea los siguientes ejemplos de CLI.

Acceder a un dominio propiedad de una cuenta en la que esté autenticado:

Al acceder a un dominio de la cuenta en la que se ha autenticado, solo necesita especificar el nombre del dominio. En el siguiente ejemplo, se enumeran los paquetes del repositorio *my_repo* del dominio *my_domain* que pertenece a su cuenta.

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

Acceder a un dominio propiedad de una cuenta en la que no esté autenticado:

Al acceder a un dominio que sea propiedad de una cuenta en la que no esté autenticado, debe especificar el propietario del dominio y el nombre del dominio. En el siguiente ejemplo, se enumeran los paquetes del repositorio *other-repo* del dominio *other-domain* que son propiedad de una cuenta en la que no está autenticado. Observe la adición del parámetro `--domain-owner`.

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

Tipos de AWS KMS claves compatibles en CodeArtifact

CodeArtifact solo admite claves KMS simétricas. No puede usar una clave KMS asimétrica para cifrar sus dominios. CodeArtifact Para obtener más información, consulte Identificación de claves de KMS simétricas y asimétricas. Para aprender a crear una nueva clave administrada por el cliente, consulte Creación de claves KMS de cifrado simétricas en la Guía para desarrolladores de AWS Key Management Service .

CodeArtifact admite almacenes de claves AWS KMS externos (XKS). Usted es responsable de la disponibilidad, la durabilidad y la latencia de las operaciones clave con las claves XKS, lo que puede afectar a la disponibilidad, la durabilidad y la latencia. CodeArtifact Algunos ejemplos de los efectos del uso de claves XKS con: CodeArtifact

- Como todos los activos de un paquete solicitado y todas sus dependencias están sujetos a una latencia de descifrado, la latencia de compilación se puede aumentar considerablemente con un aumento de la latencia de operación de XKS.
- Como todos los activos están cifrados CodeArtifact, la pérdida del material clave XKS implicará la pérdida de todos los activos asociados al dominio que utilice la clave XKS.

Para obtener más información sobre las claves XKS, consulte Almacenes de claves externos en la Guía para desarrolladores de AWS Key Management Service .

Crear un dominio

Puede crear un dominio mediante la CodeArtifact consola, el AWS Command Line Interface (AWS CLI) o AWS CloudFormation. Al crear un dominio, no contiene ningún repositorio. Para obtener más información, consulte Crear un repositorio de . Para obtener más información sobre la administración de CodeArtifact dominios con CloudFormation, consulteCreación de recursos de CodeArtifact con AWS CloudFormation.

Temas

- Crear un dominio (consola)
- Crear un dominio (AWS CLI)

Crear un dominio (consola)

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Dominios y luego Crear dominio.
3. En Nombre, escriba el nombre para su dominio.
4. Expanda Configuración adicional.
5. Utilice una AWS KMS key (clave KMS) para cifrar todos los activos de su dominio. Puede usar una clave de KMS administrada AWS o una clave de KMS que administre. Para obtener más información sobre los tipos de claves KMS compatibles CodeArtifact, consulteTipos de AWS KMS claves compatibles en CodeArtifact.

   - Elija la clave administrada de AWS si quiere usar la Clave administrada de AWS predeterminada.
   - Elija la clave administrada por el cliente si quiere usar una clave de KMS que administre. Para usar una clave de KMS que administre, en ARN de clave administrada por el cliente, busque y elija la clave de KMS.

   Para obtener más información, consulte Clave administrada de AWS y las claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service .
6. Seleccione Create domain (Crear un dominio).

Crear un dominio (AWS CLI)

Para crear un dominio con AWS CLI, utilice el `create-domain` comando. Debe usar una AWS KMS key (clave KMS) para cifrar todos los activos de su dominio. Puede usar una clave de KMS AWS administrada o una clave de KMS que usted administre. Si usa una clave de KMS AWS administrada, no use el `--encryption-key` parámetro.

Para obtener más información sobre los tipos de claves KMS compatibles CodeArtifact, consulteTipos de AWS KMS claves compatibles en CodeArtifact. Para obtener más información sobre claves KMS, consulte Clave administrada de AWS y Claves administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service .

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

Los datos con formato JSON aparecen en la salida con detalles sobre tu nuevo dominio.

```
{
    "domain": {
        "name": "my_domain",
        "owner": "111122223333",
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/my_domain",
        "status": "Active",
        "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/your-kms-key",
        "repositoryCount": 0,
        "assetSizeBytes": 0,
        "createdTime": "2020-10-12T16:51:18.039000-04:00"
    }
}
```

Si utiliza una clave de KMS que administra, incluya su nombre de recurso de Amazon (ARN) con el parámetro `--encryption-key`.

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

Los datos con formato JSON aparecen en la salida con detalles sobre tu nuevo dominio.

```
{
    "domain": {
```

```
        "name": "my_domain",
        "owner": "111122223333",
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/my_domain",
        "status": "Active",
        "encryptionKey": "arn:aws:kms:us-west-2:111122223333:key/your-kms-key",
        "repositoryCount": 0,
        "assetSizeBytes": 0,
        "createdTime": "2020-10-12T16:51:18.039000-04:00"
    }
}
```

Crear un dominio con etiquetas

Para crear un dominio con etiquetas, añada el parámetro `--tags` al comando `create-domain`.

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

Eliminar un dominio

Puedes eliminar un dominio con la CodeArtifact consola o con AWS Command Line Interface (AWS CLI).

Temas

- Restricciones a la eliminación de dominios
- Eliminar un dominio (consola)
- Eliminar un dominio (AWS CLI)

Restricciones a la eliminación de dominios

Normalmente, no se puede eliminar un dominio que contiene repositorios. Antes de eliminar el dominio, debe eliminar sus repositorios. Para obtener más información, consulte Eliminar un repositorio.

Sin embargo, si ya CodeArtifact no tienes acceso a la clave KMS del dominio, puedes eliminarlo aunque todavía contenga repositorios. Esta situación se producirá si eliminas la clave de KMS del dominio o revocas la autorización de KMS que se CodeArtifact utiliza para acceder a la clave. En

este estado, no puede acceder a los repositorios del dominio ni a los paquetes almacenados en ellos. Tampoco es posible incluir y eliminar repositorios si no se CodeArtifact puede acceder a la clave KMS del dominio. Por este motivo, la eliminación de un dominio no comprueba si el dominio contiene repositorios cuando no se puede acceder a la clave KMS del dominio.

> **Note**
>
> Cuando se elimina un dominio que aún contiene repositorios, CodeArtifact los eliminará de forma asíncrona en 15 minutos. Una vez eliminado el dominio, los repositorios seguirán visibles en la CodeArtifact consola y en el resultado del `list-repositories` comando hasta que se realice la limpieza automática del repositorio.

Eliminar un dominio (consola)

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En el panel de navegación, elija Dominios y luego elija el dominio que desea eliminar.
3. Elija Eliminar.

Eliminar un dominio (AWS CLI)

Ejecute el comando delete-domain para eliminar un dominio.

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

Los datos con formato JSON aparecen en la salida con detalles sobre el dominio eliminado.

```
{
    "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"
```

```
    }
}
```

Políticas de dominio

CodeArtifact admite el uso de permisos basados en recursos para controlar el acceso. Los permisos basados en recursos le permiten especificar quién tiene acceso a un recurso y qué acciones puede realizar en él. De forma predeterminada, solo la cuenta de AWS propietaria del dominio puede crear repositorios en el dominio y acceder a ellos. Puede aplicar un documento de política a un dominio para permitir que otros responsables de IAM accedan a él.

Para obtener más información, consulte Políticas y permisos y Políticas basadas en identidad y Políticas basadas en recursos.

Temas

- Habilitar el acceso entre cuentas a un dominio
- Ejemplo de políticas de dominio
- Ejemplo de política de dominio con AWS Organizations
- Establecer una política de dominio
- Leer una política de dominio
- Eliminar una política de dominio

Habilitar el acceso entre cuentas a un dominio

Una política de recursos es un archivo de texto en formato JSON. El archivo debe especificar una entidad principal (actor), una o más acciones y un efecto (`Allow` o `Deny`). Para crear un repositorio en un dominio propiedad de otra cuenta, se debe conceder a la entidad principal el permiso `CreateRepository` sobre el recurso del dominio.

Por ejemplo, la siguiente política de recursos concede a la cuenta el permiso `123456789012` para crear un repositorio en el dominio.

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

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

Para permitir la creación de repositorios con etiquetas, debe incluir el permiso `codeartifact:TagResource`. Esto también permitirá a la cuenta añadir etiquetas al dominio y a todos los repositorios que contiene.

Como la política solo se evalúa para las operaciones realizadas en el dominio al que está asociada, no es necesario que especifique un recurso. Como el recurso está implícito, el `Resource` se puede establecer en `*`.

Para acceder a los paquetes de un dominio propiedad de otra cuenta, se debe conceder el permiso `GetAuthorizationToken` a una entidad principal sobre el recurso del dominio. Esto permite al propietario del dominio controlar qué cuentas pueden leer el contenido de los repositorios del dominio.

Por ejemplo, la siguiente política de recursos otorga a la cuenta el permiso `123456789012` para recuperar un token de autenticación para cualquier repositorio del dominio.

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

```
    ]
}
```

> **Note**
>
> A la entidad principal que desee obtener paquetes de un punto de conexión del repositorio se le debe conceder el permiso `ReadFromRepository` sobre el recurso del repositorio además del permiso `GetAuthorizationToken` sobre el dominio. Del mismo modo, a la entidad principal que desee publicar paquetes en un punto de conexión del repositorio también se le debe otorgar el permiso `PublishPackageVersion` además de `GetAuthorizationToken`.
>
> Para obtener más información acerca de los permisos de `ReadFromRepository` y `PublishPackageVersion`, consulte Políticas de repositorios.

Ejemplo de políticas de dominio

Cuando varias cuentas utilizan un dominio, se les debe conceder un conjunto básico de permisos para permitir el uso completo del dominio. La siguiente política de recursos enumera un conjunto de permisos que permiten el uso total del dominio.

```
{
    "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**
>
> No es necesario crear una política de dominio si un dominio y todos sus repositorios pertenecen a una sola cuenta y solo es necesario utilizarlos desde esa cuenta.

Ejemplo de política de dominio con AWS Organizations

Puedes usar la clave `aws:PrincipalOrgID` condicionada para conceder acceso a un CodeArtifact dominio desde todas las cuentas de tu organización, de la siguiente manera.

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

Para obtener más información sobre el uso de la clave de condición `aws:PrincipalOrgID`, consulte Claves de contexto de condición global de AWS en la Guía del usuario de IAM.

Establecer una política de dominio

Puede usar el comando `put-domain-permissions-policy` para adjuntar una política a un dominio.

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

Al llamar a `put-domains-permissions-policy`, se ignora la política de recursos del dominio al evaluar los permisos. Esto garantiza que el propietario de un dominio no pueda excluirse del dominio, lo que le impediría actualizar la política de recursos.

> **Note**
>
> No puedes conceder permisos a otra AWS cuenta para actualizar la política de recursos de un dominio mediante una política de recursos, ya que la política de recursos se ignora al realizar una llamada put-domain-permissions-policy.

Resultado de ejemplo:

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

El resultado del comando contiene el nombre de recurso de Amazon (ARN) del recurso de dominio, todo el contenido del documento de política y un identificador de revisión. El identificador de revisión se puede pasar a `put-domain-permissions-policy` mediante la opción `--policy-revision`. Esto garantiza que se sobrescriba una revisión conocida del documento y no una versión más reciente configurada por otro escritor.

Leer una política de dominio

Para leer una versión existente de un documento de política, utilice el comando `get-domain-permissions-policy`. Para formatear la salida para que sea legible, utilice `--output` y `--query policy.document` junto con el módulo `json.tool` de Python, de la siguiente manera.

```
aws codeartifact get-domain-permissions-policy --domain my_domain --domain-
owner 111122223333 \
```

```
    --output text --query policy.document | python -m json.tool
```

Resultado de ejemplo:

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

Eliminar una política de dominio

Utilice el comando `delete-domain-permissions-policy` para eliminar una política de un dominio.

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

El formato de la salida es el mismo que el de los comandos `get-domain-permissions-policy` y `delete-domain-permissions-policy`.

Etiquete un dominio en CodeArtifact

Las etiquetas son pares clave-valor asociados a los recursos de AWS. Puede aplicar etiquetas a sus dominios en CodeArtifact. Para obtener información sobre el etiquetado de CodeArtifact recursos, los

casos de uso, las restricciones de clave y valor de las etiquetas y los tipos de recursos compatibles, consulteEtiquetado de recursos.

Puede utilizar la CLI para especificar etiquetas al crear un dominio. Puede utilizar la consola o la CLI para añadir o eliminar etiquetas, y para actualizar los valores de las etiquetas de un dominio. Puede agregar hasta 50 etiquetas a cada dominio.

Temas

Etiquetado de dominios (CLI)

Puede utilizar la CLI para administrar etiquetas de dominio.

Temas

Agregado de etiquetas a un dominio (CLI)

Puede utilizar la consola o la AWS CLI para etiquetar dominios.

Para añadir una etiqueta a un dominio al crearlo, consulte Crear un repositorio de .

En estos pasos, se presupone que ya ha instalado una versión reciente de la AWS CLI o que la ha actualizado a la versión actual. Para obtener más información, consulte Instalación de la AWS Command Line Interface.

En el terminal o la línea de comandos, ejecute el comando tag-resource, especificando el nombre de recurso de Amazon (ARN) del dominio a la que desea añadir etiquetas, y la clave y el valor de la etiqueta que desea añadir.

> **Note**
>
> Para obtener el ARN del dominio, ejecute el comando `describe-domain`:

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

Puede añadir más de una etiqueta a un dominio. Por ejemplo, para etiquetar un dominio llamado *my_domain* con dos etiquetas, una clave de etiqueta llamada *key1* con el valor de etiqueta *value1* y una clave de etiqueta llamada *key2* con el valor de etiqueta *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
```

Si tiene éxito, este comando no tiene salida.

Visualización de etiquetas de un dominio (CLI)

Sigue estos pasos para usar el AWS CLI para ver las AWS etiquetas de un dominio. Si no se han añadido etiquetas, la lista obtenida está vacía.

En el terminal o en la línea de comandos, ejecute el comando list-tags-for-resource con el nombre de recurso de Amazon (ARN) del dominio.

> **Note**
>
> Para obtener el ARN del dominio, ejecute el comando `describe-domain`:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Por ejemplo, para ver una lista de claves de etiquetas y valores de etiquetas para un dominio llamado *my_domain* con el valor ARN `arn:aws:codeartifact:us-west-2:123456789012:domain/my_domain`:

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

Si se ejecuta correctamente, este comando proporciona información similar a la siguiente:

```
{
```

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

Edición de etiquetas de un dominio (CLI)

Sigue estos pasos para usar el AWS CLI para editar una etiqueta de un dominio. Puede cambiar el valor de una clave existente o añadir otra clave. También puede eliminar etiquetas de un dominio, tal y como se muestra en la sección siguiente.

En el terminal o la línea de comandos, ejecute el comando tag-resource, especificando el ARN del dominio en el que desea actualizar una etiqueta y especifique la clave y el valor de la etiqueta:

> **Note**
>
> Para obtener el ARN del dominio, ejecute el comando `describe-domain`:
>
> ```
> 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
```

Si tiene éxito, este comando no tiene salida.

Eliminación de las etiquetas de un dominio (CLI)

Sigue estos pasos para usar el AWS CLI para eliminar una etiqueta de un dominio.

> **Note**
>
> Si elimina un dominio, todas las asociaciones de etiquetas se quitan del dominio eliminado. No es necesario eliminar las etiquetas antes de eliminar un dominio.

En el terminal o la línea de comandos, ejecute el comando untag-resource, especificando el ARN del dominio cuya etiqueta desea quitar y la clave de la etiqueta que desea quitar.

> **Note**
>
> Para obtener el ARN del dominio, ejecute el comando `describe-domain`:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Por ejemplo, para eliminar varias etiquetas de un dominio denominado *mydomain* con las claves *key1* y *key2*:

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

Si tiene éxito, este comando no tiene salida. Tras eliminar las etiquetas, puede ver las etiquetas restantes del repositorio mediante el comando `list-tags-for-resource`.

Etiquetado de dominios (consola)

Puede utilizar la consola o la CLI para etiquetar recursos.

Temas

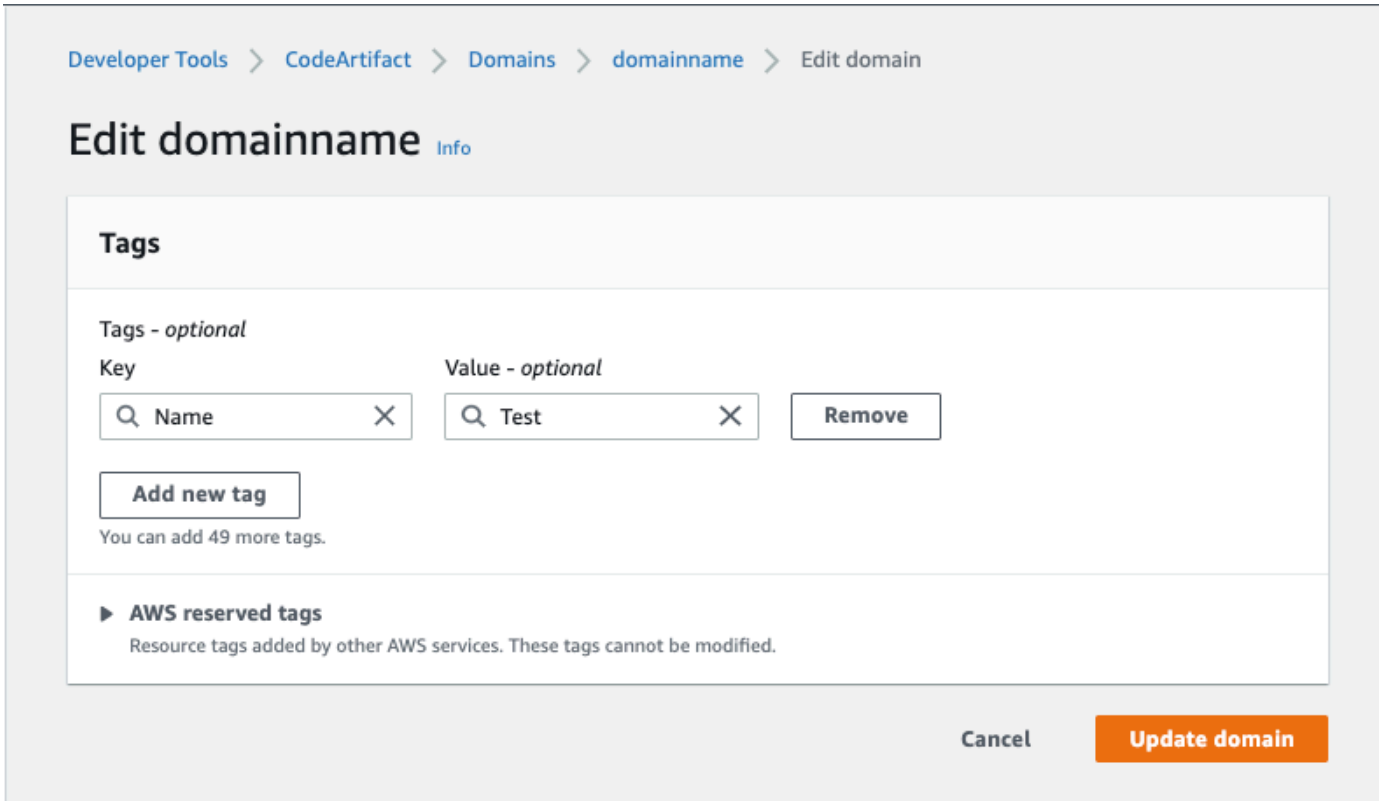
- Agregado de etiquetas a un dominio (consola)
- Visualización de etiquetas de un dominio (consola)
- Edición de etiquetas de un dominio (consola)
- Eliminación de las etiquetas de un dominio (consola)

Agregado de etiquetas a un dominio (consola)

Puede utilizar la consola para añadir etiquetas a un dominio existente.

1. Abre la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En la página Dominios, elija el dominio al que desea agregar etiquetas.
3. Amplíe la sección Detalles.
4. En Etiquetas de dominio, seleccione Agregar etiquetas de dominio si no hay etiquetas en el dominio o seleccione Ver y editar etiquetas de dominio si las hay.

5. Elija Añadir nueva etiqueta.
6. En los campos Clave y Valor, introduzca el texto para cada etiqueta que desee agregar. (El campo Value (Valor) es opcional). Por ejemplo, en Key (Clave), escriba **`Name`**. En Valor, escriba **`Test`**.



7. (Opcional) ElijaAdd tag (Añadir etiqueta) para añadir más filas y escribir más etiquetas.
8. Elija Actualizar dominio.

Visualización de etiquetas de un dominio (consola)

Puede utilizar la consola para obtener una lista de las etiquetas de los dominios existentes.

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En la página Dominios, elija el dominio en el que quiere ver las etiquetas.
3. Amplíe la sección Detalles.
4. En Etiquetas de dominio, seleccione Ver y editar etiquetas de dominio.

> **Note**
>
> Si no se ha agregado ninguna etiqueta a este dominio, la consola mostrará Agregar etiquetas de dominio.

Edición de etiquetas de un dominio (consola)

Puede utilizar la consola para editar las etiquetas que se han añadido al dominio.

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En la página Dominios, elija el dominio en el que quiere actualizar las etiquetas.
3. Amplíe la sección Detalles.
4. En Etiquetas de dominio, seleccione Ver y editar etiquetas de dominio.

> **Note**
>
> Si no se ha agregado ninguna etiqueta a este dominio, la consola mostrará Agregar etiquetas de dominio.

5. En los campos Key (Clave) y Value (Valor), actualice los valores que sean necesarios. Por ejemplo, para la clave **`Name`**, en Value (Valor), cambie **`Test`** a **`Prod`**.
6. Elija Actualizar dominio.

Eliminación de las etiquetas de un dominio (consola)

Puede utilizar la consola para eliminar etiquetas de dominios.

1. Abra la AWS CodeArtifact consola en https://console.aws.amazon.com/codesuite/codeartifact/home.
2. En la página Dominios, elija el dominio del que quiere eliminar las etiquetas.
3. Amplíe la sección Detalles.
4. En Etiquetas de dominio, seleccione Ver y editar etiquetas de dominio.

> **Note**
>
> Si no se ha agregado ninguna etiqueta a este dominio, la consola mostrará Agregar etiquetas de dominio.

5. Junto a la clave y el valor de cada etiqueta que desea eliminar, elija Eliminar.
6. Elija Actualizar dominio.

Uso de CodeArtifact con npm

Estos temas describen cómo usar npm, el administrador de paquetes Node.js, con CodeArtifact.

> **Note**
>
> CodeArtifact es compatible con `node v4.9.1` posteriores y `npm v5.0.0` y posteriores.

Temas

Configurar y usar npm con CodeArtifact

Después de crear un repositorio en CodeArtifact, puede usar el cliente npm para instalar y publicar paquetes. El método recomendado para configurar npm con el punto de conexión del repositorio y el token de autorización es mediante el comando `aws codeartifact login`. También puede configurar npm de forma manual.

Contenido

Configurar npm con el comando login

Use el comando `aws codeartifact login` para obtener las credenciales para usarlas con npm.

> **Note**
>
> Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

> **Important**
>
> Si utiliza npm 10.x o posterior, debe usar la versión 2.9.5 de AWS CLI o posterior para ejecutar correctamente el comando `aws codeartifact login`.

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

Este comando realiza los siguientes cambios en el archivo ~/.npmrc:

- Añada un token de autorización después de obtenerlo de CodeArtifact con sus credenciales AWS.
- Establece el registro npm en el repositorio especificado por la opción `--repository`.
- Para npm 6 y versiones anteriores: se suma `"always-auth=true"` para que el token de autorización se envíe para cada comando de npm.

El período de autorización predeterminado después de una llamada `login` es de 12 horas y `login` debe invocarse para actualizar periódicamente el token. Para obtener más información sobre el token de autorización creado con el comando `login`, consulte Tokens creados con el comando `login`.

Configurar npm sin usar el comando login

Puede configurar npm con su repositorio CodeArtifact sin el comando `aws codeartifact login` actualizando manualmente la configuración de npm.

Para configurar npm sin usar el comando login

1. En una línea de comandos, busque un token de autorización de CodeArtifact y guárdelo en una variable de entorno. npm usará este token para autenticarse en su repositorio de CodeArtifact.

> **Note**
>
> El siguiente comando es para máquinas macOS o Linux. Para obtener información sobre la configuración de variables de entorno en un equipo Windows, consulte Pasar un token de autenticación mediante una variable de entorno.

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

2. Obtenga el siguiente comando para obtener el punto de conexión del repositorio de CodeArtifact. El punto de conexión de su repositorio se usa para dirigir npm a su repositorio para instalar o publicar paquetes.

   - Sustituya *my_domain* por su nombre de dominio de CodeArtifact.
   - Sustituya *111122223333* por el ID de cuenta AWS del propietario del dominio. Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.
   - Sustituya *my_repo* por el nombre de su repositorio de CodeArtifact.

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

   La siguiente URL es un punto de conexión de repositorio de ejemplo.

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

   > **Important**
   >
   > La URL de registro debe terminar con una barra diagonal (/). De lo contrario, no puede conectarse al repositorio.

3. Use el comando `npm config set` para establecer el registro en su repositorio de CodeArtifact. Sustituya la URL con la URL del punto de conexión del repositorio del paso anterior.

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

4. Use el comando `npm config set` para agregar su token de autorización a su configuración de npm.

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

   Para npm 6 o versiones anteriores: para que npm siempre pase el token de autenticación a CodeArtifact, incluso para solicitudes `GET`, defina la variable de configuración `always-auth` con `npm config set`.

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

Ejemplo de archivo de configuración npm (**`.npmrc`**)

El siguiente es un archivo `.npmrc` de ejemplo después de seguir las instrucciones anteriores para configurar el punto de conexión del registro CodeArtifact, agregar un token de autenticación y configurar `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
```

Ejecutar comandos npm

Después de configurar el cliente npm, puede ejecutar los comandos npm. Suponiendo que un paquete esté presente en su repositorio o en uno de sus repositorios anteriores, puede instalarlo con `npm install`. Por ejemplo, utilice lo siguiente para instalar el paquete `lodash`.

```
npm install lodash
```

Use el siguiente comando para publicar un nuevo paquete npm en un repositorio de CodeArtifact.

```
npm publish
```

Para obtener información sobre cómo crear paquetes npm, consulte Creación de módulos de Node.js en el sitio web de documentación de npm. Para obtener una lista de los comandos npm compatibles con CodeArtifact, consulte Soporte de comandos npm.

Verificar la autenticación y autorización de npm

La invocación del comando `npm ping` es una forma de verificar lo siguiente:

- Ha configurado correctamente sus credenciales para poder autenticarse en un repositorio de CodeArtifact.
- La configuración de autorización le otorga el permiso `ReadFromRepository`.

El resultado de una invocación exitosa de `npm ping` tendría el siguiente aspecto.

```
$ 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
```

La opción `-d` hace que npm imprima información de depuración adicional, incluida la URL del repositorio. Esta información facilita la confirmación de que npm está configurado para usar el repositorio esperado.

Volver al registro npm predeterminado

La configuración de npm con CodeArtifact establece el registro npm en el repositorio CodeArtifact especificado. Puede ejecutar el siguiente comando para volver a establecer el registro npm en su registro predeterminado cuando termine de conectarse a CodeArtifact.

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

Solución de problemas de instalaciones lentas con npm 8.x o superior

Existe un problema conocido en las versiones de npm 8.x y posteriores por el que si se realiza una solicitud a un repositorio de paquetes y el repositorio redirige el cliente a Amazon S3 en lugar de transmitir los activos directamente, el cliente npm puede bloquearse durante varios minutos por dependencia.

Puesto que los repositorios de CodeArtifact están diseñados para redirigir siempre la solicitud a Amazon S3, a veces se produce este problema, lo que provoca tiempos de compilación prolongados debido a los largos tiempos de instalación de npm. Las instancias de este comportamiento se mostrarán como una barra de progreso durante varios minutos.

Para evitar este problema, utilice los marcadores `--no-progress` o `progress=false` con los comandos de la CLI `npm`, como se muestra en el siguiente ejemplo.

```
npm install lodash --no-progress
```

Configurar y usar Yarn con CodeArtifact

Después de crear un repositorio, puede usar el cliente Yarn para administrar los paquetes npm.

> **Note**
>
> `Yarn 1.X` lee y usa la información de su archivo de configuración de npm (.npmrc), pero `Yarn 2.X` no. La configuración de `Yarn 2.X` debe definirse en el archivo .yarnrc.yml.

Contenido

Configure Yarn 1.X con el comando **`aws codeartifact login`**

Para `Yarn 1.X`, puede configurar Yarn con CodeArtifact mediante el comando `aws codeartifact login`. El comando `login` configurará su archivo ~/.npmrc con la información y las

credenciales del punto de conexión del repositorio CodeArtifact. Con `Yarn 1.X`, los comandos `yarn` utilizan la información de configuración del archivo ~/.npmrc.

Para configurar **`Yarn 1.X`** con el comando login

1. Si aún no lo ha hecho, debe configurar sus credenciales AWS para utilizarlas con la AWS CLI, tal y como se describe en Introducción a CodeArtifact.
2. Para ejecutar el comando `aws codeartifact login` correctamente, npm debe estar instalado. Consulte Descarga e instalación de Node.js y npm en la documentación de npm para obtener instrucciones de instalación.
3. Use el comando `aws codeartifact login` para obtener las credenciales de CodeArtifact y configurar su archivo ~/.npmrc.

   - Sustituya *`my_domain`* por su nombre de dominio de CodeArtifact.
   - Sustituya *`111122223333`* por el ID de cuenta AWS del propietario del dominio. Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.
   - Sustituya *`my_repo`* por el nombre de su repositorio de CodeArtifact.

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

   El comando `login` realiza los siguientes cambios en el archivo ~/.npmrc:

   - Añada un token de autorización después de obtenerlo de CodeArtifact con sus credenciales AWS.
   - Establece el registro npm en el repositorio especificado por la opción `--repository`.
   - Para npm 6 y versiones anteriores: se suma `"always-auth=true"` para que el token de autorización se envíe para cada comando de npm.

   El período de autorización predeterminado después de una llamada `login` es de 12 horas y `login` debe invocarse para actualizar periódicamente el token. Para obtener más información sobre el token de autorización creado con el comando `login`, consulte Tokens creados con el comando `login`.
4. Para npm 7.X y 8.X, debe añadir `always-auth=true` a su archivo ~/.npmrc para usar Yarn.

- Abra su archivo ~/.npmrc en un editor de texto y añada `always-auth=true` en una nueva línea.

Puede usar el comando `yarn config list` para comprobar que Yarn está usando la configuración correcta. Tras ejecutar el comando, compruebe los valores de la sección `info npm config`. El contenido debe ser similar al siguiente fragmento.

```
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
}
```

Configure Yarn 2.X con el comando **`yarn config set`**

El siguiente procedimiento detalla cómo realizar la configuración de `Yarn 2.X` actualizando su configuración `.yarnrc.yml` desde la línea de comandos con el comando `yarn config set`.

Para actualizar la configuración **`yarnrc.yml`** desde la línea de comandos

1. Si aún no lo ha hecho, debe configurar sus credenciales AWS para utilizarlas con la AWS CLI, tal y como se describe en Introducción a CodeArtifact.
2. Use el comando `aws codeartifact get-repository-endpoint` para obtener el punto de conexión de tu repositorio de CodeArtifact.

   - Sustituya *`my_domain`* por su nombre de dominio de CodeArtifact.
   - Sustituya *`111122223333`* por el ID de cuenta AWS del propietario del dominio. Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.
   - Sustituya *`my_repo`* por el nombre de su repositorio de CodeArtifact.

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

3. Actualice el valor `npmRegistryServer` de su archivo .yarnrc.yml con el punto de conexión de su repositorio.

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

4. Obtenga un token de autorización CodeArtifact y guárdelo en una variable de entorno.

> **Note**
>
> El siguiente comando es para máquinas macOS o Linux. Para obtener información sobre la configuración de variables de entorno en un equipo Windows, consulte Pasar un token de autenticación mediante una variable de entorno.

- Sustituya *my_domain* por su nombre de dominio de CodeArtifact.
- Sustituya *111122223333* por el ID de cuenta AWS del propietario del dominio. Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.
- Sustituya *my_repo* por el nombre de su repositorio de CodeArtifact.

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

5. Use el comando `yarn config set` para añadir su token de autenticación CodeArtifact a su archivo .yarnrc.yml. Sustituya la URL del siguiente comando por la URL del punto de conexión del repositorio del paso 2.

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAuthToken'
 "${CODEARTIFACT_AUTH_TOKEN}"
```

6. Utilice el comando `yarn config set` para establecer el valor de `npmAlwaysAuth` a `true`. Sustituya la URL del siguiente comando por la URL del punto de conexión del repositorio del paso 2.

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

Tras la configuración, el archivo de configuración .yarnrc.yml debería tener un contenido similar al del siguiente fragmento.

```
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/"
```

También puede usar el comando `yarn config` para comprobar los valores de `npmRegistries` y `npmRegistryServer`.

soporte de comandos npm

En las siguientes secciones se resumen los comandos npm compatibles con los repositorios de CodeArtifact, además de los comandos específicos que no son compatibles.

Contenido

Comandos compatibles que interactúan con un repositorio

En esta sección se enumeran los comandos npm en los que el cliente npm realiza una o más solicitudes al registro con el que se ha configurado (por ejemplo, con `npm config set registry`). Se ha comprobado que estos comandos funcionan correctamente cuando se invocan en un repositorio de CodeArtifact.

Comando	Descripción
bugs	Intenta adivinar la ubicación de la URL del rastreador de errores de un paquete y, a continuación, intenta abrirla.
ci	Instala un proyecto desde cero.
deprecate	Deja en desuso una versión de un paquete.
dist-tag	Modifica las etiquetas de distribución de paquetes.
docs	Intenta adivinar la ubicación de la URL de la documentación de un paquete y, a continuac ión, intenta abrirla mediante el parámetro de configuración `--browser` .
doctor	Ejecuta una serie de comprobaciones para garantizar que su instalación de npm tenga lo que necesita para administrar sus paquetes de JavaScript.
install	Instala un paquete.
install-ci-test	Instala un proyecto desde cero y ejecuta pruebas. Alias: `npm cit`. Este comando se ejecuta un `npm ci` seguido inmediatamente por un `npm test`.
install-test	Instala el paquete y ejecuta las pruebas. Ejecuta un `npm install` seguido inmediata mente por un `npm test`.
outdated	Comprueba el registro configurado para ver si alguno de los paquetes instalados está desactualizado actualmente.

Comando	Descripción
ping	Hace ping al registro npm configurado o dado y verifica la autenticación.
publish	Publica una versión del paquete en el registro.
update	Averigua la ubicación de la URL del repositorio de un paquete y, a continuación, intenta abrirla mediante el parámetro de configuración `--browser` .
view	Muestra metadatos del paquete. Se puede usar para imprimir las propiedades de los metadatos .

Comandos del lado del cliente admitidos

Estos comandos no requieren ninguna interacción directa con un repositorio, por lo que CodeArtifact no necesita hacer nada para respaldarlos.

Comando	Descripción
build	Crea un paquete.
cache	Manipula la caché de paquetes.
completion	Permite completar tabulaciones en todos los comandos de npm.
config	Actualiza el contenido de los archivos `npmrc` globales y de usuario.
dedupe	Busca en el árbol de paquetes local e intenta simplificar la estructura moviendo las dependencias más arriba en el árbol, donde pueden compartirse de manera más eficaz entre varios paquetes dependientes.

Comando	Descripción
edit	Edita un paquete instalado. Selecciona una dependencia en el directorio de trabajo actual y abre la carpeta del paquete en el editor predeterminado.
explore	Busca un paquete instalado. Genera una subshell en el directorio del paquete instalado especificado. Si se especifica un comando, se ejecuta en la subshell, que finaliza inmediata mente.
help	Obtiene ayuda sobre npm.
help-search	Busca en la documentación de ayuda de npm.
init	Crea un archivo `package.json` .
link	Enlaza simbólicamente una carpeta de paquetes.
ls	Muestra los paquetes instalados.
pack	Crea un tarball a partir de un paquete.
prefix	Muestra el prefijo. Este es el directorio principal más cercano que contiene un archivo `package.json` , a menos que también se especifique `-g`.
prune	Elimina los paquetes que no figuran en la lista de dependencias del paquete principal.
rebuild	Ejecuta el comando `npm build` en las carpetas coincidentes.

Comando	Descripción
restart	Ejecuta los scripts de parada, reinicio e inicio de un paquete, así como los scripts previos y posteriores asociados.
root	Imprime la carpeta `node_modules` efectiva en formato estándar.
run-script	Ejecuta scripts de paquetes arbitrarios.
shrinkwrap	Bloquea las versiones dependientes para su publicación.
uninstall	Desinsta un paquete.

Comandos admitidos

Estos comandos npm no son compatibles con los repositorios de CodeArtifact.

Comando	Descripción	Notas
access	Establece el nivel de acceso de los paquetes publicados.	CodeArtifact usa un modelo de permisos que es diferente del repositorio público npmjs.
adduser	Añade una cuenta de usuario de registro	CodeArtifact usa un modelo de usuario diferente del repositorio público npmjs.
audit	Realiza una auditoría de seguridad.	CodeArtifact no vende actualmente datos sobre vulnerabilidades de seguridad.
hook	Administra los enlaces npm, lo que incluye agregar, eliminar, enumerar y actualizar.	CodeArtifact no admite actualmente ningún tipo de mecanismo de notificación de cambios.

Comando	Descripción	Notas
login	Autentica a un usuario. Este es un alias para `npm adduser`.	CodeArtifact utiliza un modelo de autenticación diferente del repositorio público npmjs. Para obtener información, consulte Autenticación con npm.
logout	Cierra la sesión del registro.	CodeArtifact utiliza un modelo de autenticación diferente del repositorio público npmjs. No hay forma de cerrar sesión en un repositorio de CodeArtifact, pero los tokens de autentica ción caducan después del tiempo de caducidad configura ble. La duración predeterm inada del token es de 12 horas.
owner	Administra a los propietarios de los paquetes.	CodeArtifact usa un modelo de permisos que es diferente del repositorio público npmjs.
profile	Cambia la configuración de su perfil de registro.	CodeArtifact usa un modelo de usuario diferente del repositorio público npmjs.
search	Busca en el registro paquetes que coincidan con los términos de búsqueda.	CodeArtifact admite una funcionalidad de búsqueda limitada con el comando list-packages.
star	Marca sus paquetes favoritos.	CodeArtifact actualmente no admite ningún tipo de mecanismo de favoritos.

Comando	Descripción	Notas
stars	Visualiza los paquetes marcados como favoritos.	CodeArtifact actualmente no admite ningún tipo de mecanismo de favoritos.
team	Administra los equipos de la organización y las membresía s de los equipos.	CodeArtifact utiliza un modelo de membresía de usuarios y grupos que es diferente del repositorio público npmjs. Para obtener información, consulte Identidades (usuarios , grupos y roles) en la Guía del usuario de IAM.
token	Administra sus tokens de autenticación.	CodeArtifact usa un modelo diferente para obtener los tokens de autenticación. Para obtener información, consulte Autenticación con npm.
unpublish	Elimina un paquete del registro.	CodeArtifact no admite la eliminación de una versión de paquete de un repositor io mediante el cliente npm. Puede usar el comando delete-package-version.
whoami	Muestra el nombre de usuario de npm.	CodeArtifact usa un modelo de usuario diferente del repositorio público npmjs.

control de etiquetas npm

Los registros npm admiten etiquetas, que son alias de cadena para las versiones de los paquetes. Puede usar etiquetas para proporcionar un alias en lugar de números de versión. Por ejemplo, puede tener un proyecto con varios flujos de desarrollo y usar una etiqueta diferente (por ejemplo `stable`,

`beta`, `dev`, `canary`) para cada flujo. Para obtener más información, consulte dist-tag en el sitio web de npm.

De forma predeterminada, npm usa la etiqueta `latest` para identificar la versión actual de un paquete. `npm install` *`pkg`*(sin especificador `@`*`version`* `@`*`tag`*) instala la última etiqueta. Por lo general, los proyectos utilizan la etiqueta más reciente solo para las versiones estables. Se utilizan otras etiquetas para las versiones inestables o preliminares.

Editar etiquetas con el cliente npm

Los tres `npm dist-tag` comandos (`add`, `rm`, y `ls`) funcionan de forma idéntica en los repositorios de CodeArtifact que en el registro npm predeterminado.

Etiquetas npm y la API CopyPackageVersions

Cuando usa la API `CopyPackageVersions` para copiar una versión del paquete npm, todas las etiquetas que dan un alias a esa versión se copian en el repositorio de destino. Cuando una versión que se está copiando tiene una etiqueta que también está presente en el destino, la operación de copia establece el valor de la etiqueta en el repositorio de destino para que coincida con el valor del repositorio de origen.

Por ejemplo, supongamos que tanto el repositorio S como el repositorio D contienen una única versión del paquete `web-helper` con el conjunto de etiquetas latest, como se muestra en esta tabla.

Repositorio	Package name	Etiquetas de paquetes
S	`web-helper`	latest (alias de la versión 1.0.1)
D	`web-helper`	latest (alias de la versión 1.0.0)

`CopyPackageVersions` se invoca para copiar `web-helper` 1.0.1 de S a D. Una vez finalizada la operación, la etiqueta `latest` en `web-helper` del repositorio D pasa a ser el alias 1.0.1, no el 1.0.0.

Si necesita cambiar las etiquetas después de copiarlas, utilice el comando `npm dist-tag` para modificarlas directamente en el repositorio de destino. Para obtener más información sobre la API `CopyPackageVersions`, consulte Copiar paquetes entre repositorios.

Etiquetas npm y repositorios ascendentes

Cuando npm solicita las etiquetas de un paquete y las versiones de ese paquete también están presentes en un repositorio ascendente, CodeArtifact fusiona las etiquetas antes de devolverlas al cliente. Por ejemplo, un repositorio denominado R tiene un repositorio ascendente denominado U. En la siguiente tabla se muestran las etiquetas de un paquete denominado `web-helper` que está presente en ambos repositorios.

Repositorio	Package name	Etiquetas de paquetes
R	`web-helper`	latest (alias de la versión 1.0.0)
U	`web-helper`	alpha (alias de la versión 1.0.1)

En este caso, cuando el cliente npm busca las etiquetas del paquete `web-helper` del repositorio R, recibe las etiquetas latest y alpha. Las versiones a las que apuntan las etiquetas no cambiarán.

Cuando la misma etiqueta está presente en el mismo paquete tanto en el repositorio ascendente como en el descendente, CodeArtifact usa la etiqueta que está presente en el repositorio ascendente. Por ejemplo, supongamos que las etiquetas de webhelper se han modificado para que tengan el siguiente aspecto.

Repositorio	Package name	Etiquetas de paquetes
R	`web-helper`	latest (alias de la versión 1.0.0)
U	`web-helper`	latest (alias de la versión 1.0.1)

En este caso, cuando el cliente npm busque las etiquetas del paquete web-helper del repositorio R, la etiqueta latest usará el alias de la versión 1.0.1 porque es lo que está en el repositorio ascendente. Esto facilita el consumo de nuevas versiones de paquetes en un repositorio ascendente que aún no están presentes en un repositorio descendente mediante la ejecución de `npm update`.

El uso de la etiqueta en el repositorio ascendente puede resultar problemático a la hora de publicar nuevas versiones de un paquete en un repositorio descendente. Por ejemplo, supongamos que la última etiqueta del paquete web-helper es la misma tanto en R como en U.

Repositorio	Package name	Etiquetas de paquetes
R	`web-helper`	latest (alias de la versión 1.0.1)
U	`web-helper`	latest (alias de la versión 1.0.1)

Cuando la versión 1.0.2 se publica en R, npm actualiza la etiqueta latest a 1.0.2.

Repositorio	Package name	Etiquetas de paquetes
R	`web-helper`	latest (alias de la versión 1.0.2)
U	`web-helper`	latest (alias de la versión 1.0.1)

Sin embargo, el cliente npm nunca ve este valor de etiqueta porque el valor de latest en U es 1.0.1. Si se ejecuta `npm install` en el repositorio R inmediatamente después de publicar la 1.0.2, se instala la 1.0.1 en lugar de la versión que se acaba de publicar. Para instalar la versión publicada más recientemente, debe especificar la versión exacta del paquete, de la siguiente manera.

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

Soporte para gestores de paquetes compatibles con npm

Estos otros administradores de paquetes son compatibles con CodeArtifact y funcionan con el formato de paquete npm y el protocolo npm wire:

- Administrador de paquetes pnpm. La última versión confirmada que funciona con CodeArtifact es la 3.3.4, que se lanzó el 18 de mayo de 2019.

- Administrador de paquetes Yarn. La última versión confirmada que funciona con CodeArtifact es la 1.21.1, que se lanzó el 11 de diciembre de 2019.

> **Note**
>
> Recomendamos usar Yarn 2.x con CodeArtifact. Yarn 1.x no tiene reintentos HTTP, lo que significa que es más susceptible a fallos de servicio intermitentes que se traducen en códigos de estado de nivel 500 o errores. No hay forma de configurar una estrategia de reintentos diferente para Yarn 1.x, pero se ha añadido en Yarn 2.x. Puede usar Yarn 1.x, pero es posible que tenga que añadir reintentos de nivel superior en los scripts de compilación. Por ejemplo, ejecute el comando yarn en bucle para que vuelva a intentarlo si se produce un error al descargar los paquetes.

Uso de CodeArtifact con Python

En estos temas se describe cómo usar `pip`, el administrador de paquetes Python y `twine`, la utilidad de publicación de paquetes Python con CodeArtifact.

Temas

Configurar y usar pip con CodeArtifact

pip es el instalador de paquetes para los paquetes de Python. Para usar pip para instalar paquetes de Python desde su repositorio de CodeArtifact, primero debe configurar el cliente pip con la información y las credenciales de su repositorio de CodeArtifact.

pip solo se puede usar para instalar paquetes de Python. Para publicar paquetes de Python, puede usar twine. Para obtener más información, consulte Configurar y usar twine con CodeArtifact.

Configure pip con el comando **`login`**

En primer lugar, configure sus credenciales AWS para usarlas con AWS CLI, tal y como se describe en Introducción a CodeArtifact. A continuación, utilice el comando `login` de CodeArtifact para obtener las credenciales y configurar `pip` con ellas.

> **Note**
>
> Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

Ejecute el siguiente comando para configurar pip.

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

`login` obtiene un token de autorización de CodeArtifact con sus credenciales AWS. El `login` comando configurará `pip` para su uso con CodeArtifact editando `~/.config/pip/pip.conf` para establecer el `index-url` en el repositorio especificado por la opción `--repository`.

El período de autorización predeterminado después de una llamada `login` es de 12 horas y `login` debe invocarse para actualizar periódicamente el token. Para obtener más información sobre el token de autorización creado con el comando `login`, consulte Tokens creados con el comando `login`.

Configurar pip sin el comando login

Si no puede usar el comando `login` para configurar `pip`, puede usar `pip config`.

1. Use la AWS CLI para obtener un nuevo token de autorización.

   > **Note**
   >
   > Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir el `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

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

2. Use `pip config` para configurar la URL y las credenciales del registro de CodeArtifact. El siguiente comando actualizará únicamente el archivo de configuración del entorno actual. Para actualizar el archivo de configuración de todo el sistema, sustituya `site` por `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**
>
> La dirección URL del registro debe terminar con una barra diagonal (/). De lo contrario, no puede conectarse al repositorio.

Ejemplo de archivo de configuración de pip

El siguiente es un ejemplo de un archivo `pip.conf` después de configurar la URL y las credenciales del registro de CodeArtifact.

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

Ejecutar pip

Para ejecutar comandos `pip`, debe configurar `pip` con CodeArtifact. Para obtener más información, consulte la documentación siguiente.

1. Siga los pasos de la sección Configuración con AWS CodeArtifact para configurar su cuenta AWS, sus herramientas y sus permisos.
2. Configure `twine` siguiendo los pasos que se describen en Configurar y usar twine con CodeArtifact.

Suponiendo que un paquete esté presente en su repositorio o en uno de sus repositorios ascendentes, puede instalarlo con `pip install`. Por ejemplo, utilice el siguiente comando para instalar el paquete `requests`.

```
pip install requests
```

Use la opción `-i` para volver temporalmente a instalar paquetes desde https://pypi.org en lugar de desde su repositorio CodeArtifact.

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

Configurar y usar twine con CodeArtifact

twine es una utilidad de publicación de paquetes para paquetes de Python. Para usar twine para publicar paquetes de Python en su repositorio de CodeArtifact, primero debe configurar twine con la información y las credenciales de su repositorio de CodeArtifact.

twine solo se puede usar para publicar paquetes de Python. Para instalar paquetes de Python, puede usar pip. Para obtener más información, consulte Configurar y usar pip con CodeArtifact.

Configurar twine con el comando **`login`**

En primer lugar, configure sus credenciales AWS para usarlas con AWS CLI, tal y como se describe en Introducción a CodeArtifact. A continuación, utilice el comando `login` de CodeArtifact para obtener las credenciales y configurar twine con ellas.

> **Note**
>
> Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

Ejecute el siguiente comando para configurar el hilo.

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

`login` obtiene un token de autorización de CodeArtifact con sus credenciales AWS. El comando `login` configura twine para usarlo con CodeArtifact editando `~/.pypirc` para agregar el repositorio especificado por la opción `--repository` con credenciales.

El período de autorización predeterminado después de una llamada `login` es de 12 horas y `login` debe invocarse para actualizar periódicamente el token. Para obtener más información sobre el token de autorización creado con el comando `login`, consulte Tokens creados con el comando `login`.

Configurar twine sin el comando **`login`**

Si no puede usar el comando `login` para configurar twine, puede usar el archivo `~/.pypirc` o las variables de entorno. Para usar el archivo `~/.pypirc`, agréguele las siguientes entradas. La contraseña debe ser un token de autenticación adquirido por la API `get-authorization-token`.

```
[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
```

Para utilizar variables de entorno, haga lo siguiente.

> **Note**
>
> Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir el `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

```
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`
```

Ejecutar twine

Antes de usar twine para publicar los activos del paquete de Python, primero debe configurar los permisos y recursos de CodeArtifact.

1. Siga los pasos de la sección Configuración con AWS CodeArtifact para configurar su cuenta AWS, sus herramientas y sus permisos.
2. Configure Twine siguiendo los pasos que se indican en Configurar twine con el comando `login` o Configurar twine sin el comando `login`.

Tras configurar twine, puede ejecutar comandos `twine`. Utilice el siguiente comando para publicar los activos del paquete Python.

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

Para obtener información sobre cómo crear y empaquetar una aplicación de Python, consulte Generación de archivos de distribución en el sitio web de Python Packaging Authority.

Normalización de nombres de paquetes de Python

CodeArtifact normaliza los nombres de los paquetes antes de almacenarlos, lo que significa que los nombres de los paquetes en CodeArtifact pueden ser diferentes del nombre proporcionado cuando se publicó el paquete.

Para los paquetes de Python, al realizar la normalización, el nombre del paquete aparece en minúsculas y todas las instancias de los caracteres `.`, `-` y `_` se sustituyen por un solo carácter `-`. Por lo tanto, los nombres de los paquetes `pigeon_cli` y `pigeon.cli` se normalizan y se almacenan como `pigeon-cli`. pip y twine pueden usar el nombre no normalizado, pero el nombre normalizado debe usarse en las solicitudes de CLI o API de CodeArtifact (como `list-package-versions`) y en los ARN. Para obtener más información sobre la normalización de los nombres de paquete de Python, consulte PEP 503 en la documentación de Python.

Compatibilidad con Python

CodeArtifact no es compatible con PyPI ni con las API `XML-RPC` o `JSON`.

CodeArtifact es compatible con las API `Legacy` de PyPI, excepto la API `simple`. Si bien CodeArtifact no es compatible con el punto de conexión de la API `/simple/`, `/simple/<project>/` sí lo admite.

Para obtener más información, consulte el siguiente enlace en el repositorio GitHub de Python Packaging Authority.

- API XML-RPC
- API JSON
- API heredada

soporte de comandos pip

Las siguientes secciones resumen los comandos pip que admiten los repositorios de CodeArtifact, además de los comandos específicos que no son compatibles.

Temas

Comandos compatibles que interactúan con un repositorio

En esta sección se enumeran los comandos `pip` en los que el cliente `pip` realiza una o más solicitudes al registro con el que se ha configurado. Se ha comprobado que estos comandos funcionan correctamente cuando se invocan en un repositorio de CodeArtifact.

Comando	Descripción
install	Instalar paquetes.
download	Paquetes de descarga.

CodeArtifact no implementa `pip search`. Si ha configurado `pip` con un repositorio CodeArtifact, al ejecutar `pip search` se buscarán y mostrarán los paquetes de PyPI.

Comandos del lado del cliente compatibles

Estos comandos no requieren ninguna interacción directa con un repositorio, por lo que CodeArtifact no necesita hacer nada para respaldarlos.

Comando	Descripción
uninstall	Desinstalar paquetes.
freeze	Salida de paquetes instalados en formato de requisitos.
list	Ver una lista de los paquetes instalados.
show	Mostrar información acerca de los paquetes instalados.
check	Compruebe que los paquetes instalados tengan dependencias compatibles.

Comando	Descripción
config	Administre la configuración local y global.
wheel	Construya ruedas a partir de sus necesidades.
hash	Calcule los hashes de los archivos de paquetes.
completion	Ayuda a completar los comandos.
debug	Mostrar información útil para depuración.
help	Mostrar la ayuda para los comandos.

Solicitud de paquetes de Python desde conexiones ascendentes y externas

Al importar una versión de paquete de Python desde pypi.org, CodeArtifact importará todos los activos de esa versión de paquete. Si bien la mayoría de los paquetes de Python contienen una pequeña cantidad de activos, algunos contienen más de 100 activos, normalmente para admitir múltiples arquitecturas de hardware e intérpretes de Python.

Es habitual que se publiquen nuevos activos en pypi.org para una versión de paquete existente. Por ejemplo, algunos proyectos publican nuevos activos cuando se lanzan nuevas versiones de Python. Cuando se instala un paquete de Python desde CodeArtifact con `pip install`, las versiones del paquete que se conservan en el repositorio de CodeArtifact se actualizan para reflejar el último conjunto de activos de pypi.org.

Del mismo modo, si hay nuevos activos disponibles para una versión de paquete en un repositorio de CodeArtifact anterior que no están presentes en el repositorio de CodeArtifact actual, se conservarán en el repositorio actual cuando se ejecute `pip install`.

Versiones de paquetes retiradas

Algunas versiones de paquetes en pypi.org están marcadas como retiradas, lo que indica al instalador del paquete (por ejemplo, pip) que la versión no debe instalarse a menos que sea la única

que coincida con un especificador de versión (utilizando == o ===). Consulte PEP_592 para obtener más información.

Si la versión de un paquete en CodeArtifact se obtuvo originalmente desde una conexión externa a pypi.org, al instalar la versión del paquete desde un repositorio de CodeArtifact, CodeArtifact se asegura de que los metadatos extraídos actualizados de la versión del paquete se obtengan de pypi.org.

Cómo saber si se ha retirado una versión de un paquete

Para comprobar si una versión de un paquete está eliminada en CodeArtifact, puede intentar instalarla con `pip install` *`packageName`*`===`*`packageVersion`*. Si la versión del paquete está retirada, recibirá un mensaje de advertencia similar al siguiente:

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

Para comprobar si una versión de paquete ha sido eliminada de pypi.org, puede visitar la lista de pypi.org de la versión del paquete en `https://pypi.org/project/`*`packageName`*`/`*`packageVersion`*`/`.

Establecer el estado de retirada en los paquetes privados

CodeArtifact no admite la configuración de metadatos retirados para los paquetes publicados directamente en los repositorios de CodeArtifact.

¿Por qué CodeArtifact no obtiene los últimos metadatos o activos retirados para una versión de paquete?

Normalmente, CodeArtifact garantiza que cuando se obtiene una versión de un paquete de Python de un repositorio de CodeArtifact, los metadatos extraídos estén actualizados con el último valor de pypi.org. Además, la lista de activos de la versión de paquete también se mantiene actualizada con el conjunto más reciente en pypi.org y en cualquier repositorio principal de CodeArtifact. Esto es cierto tanto si instala la versión del paquete por primera vez y CodeArtifact la importa desde pypi.org a su repositorio de CodeArtifact, como si ha instalado el paquete anteriormente. Sin embargo, hay casos en los que el cliente del administrador de paquetes, como pip, no extrae los últimos metadatos extraídos de pypi.org o repositorios principales. En su lugar, CodeArtifact devolverá los datos que ya están almacenados en su repositorio. En esta sección se describen las tres formas en que esto puede ocurrir:

Configuración inicial: si elimina la conexión externa a pypi.org desde el repositorio o sus fuentes de distribución mediante disassociate-external-connection, los metadatos extraídos ya no se actualizarán de pypi.org. Del mismo modo, si elimina un repositorio principal, los activos del repositorio eliminado y de las fuentes de distribución del repositorio eliminado dejarán de estar disponibles en el repositorio actual. Lo mismo ocurre si utiliza los controles de origen del paquete CodeArtifact para evitar que se extraigan nuevas versiones de un paquete específico; la configuración `upstream=BLOCK` bloqueará la actualización de los metadatos retirados.

Estado de la versión del paquete: si establece el estado de la versión de un paquete en cualquier otra opción excepto `Published` o `Unlisted`, los metadatos y activos retirados de la versión del paquete no se actualizarán. Del mismo modo, si está buscando una versión de paquete específica (por ejemplo, `torch 2.0.1`) y la misma versión de paquete está presente en un repositorio principal con un estado que no es `Published` o `Unlisted`, esto también impedirá la propagación de los metadatos y activos retirados del repositorio principal al repositorio actual. Esto se debe a que los estados de las versiones de otros paquetes indican que las versiones ya no están destinadas a consumirse en ningún repositorio.

Publicación directa: si publica una versión de paquete específica directamente en un repositorio de CodeArtifact, esto impedirá la actualización de los metadatos y activos retirados para la versión del paquete de sus repositorios principales y pypi.org. Por ejemplo, supongamos que descarga un activo de la versión `torch 2.0.1` del paquete, por ejemplo `torch-2.0.1-cp311-none-macosx_11_0_arm64.whl`, usando un navegador web y luego lo publica en su repositorio de CodeArtifact usando twine como `torch 2.0.1`. CodeArtifact rastrea que la versión del paquete se introdujo en el dominio mediante la publicación directa en su repositorio, no desde una conexión externa a pypi.org o un repositorio principal. En este caso, CodeArtifact no mantiene los metadatos retirados sincronizados con los repositorios principales o con pypi.org. Lo mismo ocurre si publica `torch 2.0.1` en un repositorio principal: la presencia de la versión del paquete impedirá la propagación de los metadatos y activos retirados a los repositorios que se encuentran más abajo en el gráfico inicial.

Uso de CodeArtifact con Maven

El formato de repositorio Maven lo utilizan muchos lenguajes diferentes, incluidos Java, Kotlin, Scala y Clojure. Es compatible con muchas herramientas de compilación diferentes, como Maven, Gradle, Scala SBT, Apache Ivy y Leiningen.

Hemos probado y confirmado la compatibilidad con CodeArtifact para las siguientes versiones:

- Última versión de Maven: 3.6.3.
- También se probó la última versión de Gradle: 6.4.1. 5.5.1.
- También se ha probado la última versión de Clojure: 1.11.1.

Temas

Uso de CodeArtifact con Gradle

Después de tener el token de autenticación CodeArtifact en una variable de entorno, como se describe en Pasar un token de autenticación mediante una variable de entorno, siga estas instrucciones para consumir paquetes de Maven desde un repositorio de CodeArtifact y publicar nuevos paquetes en él.

Temas

- Ejecutar una compilación de Gradle en IntelliJ IDEA

Extraer dependencias

Para obtener las dependencias de CodeArtifact en una compilación de Gradle, use el siguiente procedimiento.

Para obtener dependencias de CodeArtifact en una compilación de Gradle

1. Si no lo ha hecho, cree y almacene un token de autenticación CodeArtifact en una variable de entorno siguiendo el procedimiento en Pasar un token de autenticación mediante una variable de entorno.
2. Añada una sección `maven` a la sección `repositories` del archivo del proyecto `build.gradle`.

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

La `url` del ejemplo anterior es el punto de conexión de su repositorio de CodeArtifact. Gradle usa el punto de conexión para conectarse al repositorio. En el ejemplo, `my_domain` es el nombre de tu dominio, `111122223333` es el ID del propietario del dominio y `my_repo` es el nombre de su repositorio. Puede recuperar un punto de conexión de su repositorio mediante el comando `get-repository-endpoint` AWS CLI.

Por ejemplo, con un repositorio llamado *my_repo* dentro de un dominio llamado *my_domain*, el comando es el siguiente:

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

El comando `get-repository-endpoint` devolverá el punto de conexión del repositorio:

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

El objeto `credentials` del ejemplo anterior incluye el token de autenticación CodeArtifact que creó en el paso 1 y que Gradle usa para autenticarse en CodeArtifact.

3. (Opcional): para usar el repositorio CodeArtifact como la única fuente de las dependencias de su proyecto, elimine cualquier otra sección en `repositories` de `build.gradle`. Si tiene más de un repositorio, Gradle busca las dependencias en cada repositorio en el orden en que aparecen en la lista.
4. Después de configurar el repositorio, puede agregar las dependencias del proyecto a la sección `dependencies` con la sintaxis estándar de Gradle.

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

Complementos de búsqueda

De forma predeterminada, Gradle resolverá los complementos desde el portal de complementos de Gradle público. Para extraer complementos de un repositorio de CodeArtifact, utilice el siguiente procedimiento.

Para extraer complementos de un repositorio de CodeArtifact

1. Si no lo ha hecho, cree y almacene un token de autenticación CodeArtifact en una variable de entorno siguiendo el procedimiento en Pasar un token de autenticación mediante una variable de entorno.
2. Agregue un bloque `pluginManagement` a su archivo `settings.gradle`. El bloque `pluginManagement` debe aparecer antes de cualquier otra declaración en `settings.gradle`; consulte el siguiente fragmento:

```
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
            }
        }
    }
}
```

Esto garantizará que Gradle resuelva los complementos del repositorio especificado. El repositorio debe tener un repositorio principal con una conexión externa al portal de complementos de Gradle (por ejemplo `gradle-plugins-store`) para que los complementos de Gradle que se requieren con más frecuencia estén disponibles en la compilación. Para obtener más información, consulte la documentación de Gradle.

Publicar artefactos

En esta sección, se describe cómo publicar una biblioteca Java creada con Gradle en un repositorio de CodeArtifact.

Primero, agregue el complemento `maven-publish` a la sección `plugins` del archivo del proyecto `build.gradle`.

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

A continuación, añada una sección `publishing` al archivo `build.gradle` del proyecto.

```
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
            }
        }
    }
}
```

El `maven-publish` genera un archivo POM basado en el `groupId`, `artifactId` y `version` especificado en la sección `publishing`.

Una vez completados estos cambios en `build.gradle`, ejecute el siguiente comando para crear el proyecto y subirlo al repositorio.

```
./gradlew publish
```

Utilice `list-package-versions` para comprobar que el paquete se ha publicado correctamente.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven\
  --namespace com.company.framework --package my-package-name
```

Resultados de ejemplo:

```
{
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "example",
    "versions": [
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

Para obtener más información, consulte estos temas en el sitio web de Gradle:

- Creación de bibliotecas Java
- Publicar un proyecto como módulo

Ejecutar una compilación de Gradle en IntelliJ IDEA

Puede ejecutar una compilación de Gradle en IntelliJ IDEA que extraiga dependencias de CodeArtifact. Para autenticarse con CodeArtifact, debe proporcionar a Gradle un token de autorización de CodeArtifact. Existen tres métodos para proporcionar un token de autenticación.

- Método 1: almacenar el token de autenticación en `gradle.properties`. Utilice este método si puede sobrescribir o añadir contenido al archivo `gradle.properties`.
- Método 2: almacenar el token de autenticación en un archivo independiente. Utilice este método si no desea modificar el archivo `gradle.properties`.
- Método 3: generar un nuevo token de autenticación para cada ejecución ejecutando `aws` como un script en línea en `build.gradle`. Utilice este método si quiere que el script de Gradle busque un nuevo token en cada ejecución. El token no se almacenará en el sistema de archivos.

Token stored in gradle.properties

Método 1: almacenar el token de autenticación en **`gradle.properties`**

> **Note**
>
> El ejemplo muestra el archivo `gradle.properties` ubicado en `GRADLE_USER_HOME`.

1. Actualice el archivo `build.gradle` con el siguiente fragmento:

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

```
            }
    }
}
```

2. Para obtener complementos de CodeArtifact, agregue un bloque `pluginManagement` a su archivo `settings.gradle`. El bloque `pluginManagement` debe aparecer antes de cualquier otra declaración en `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. Obtenga un token de autorización de CodeArtifact:

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

4. Escriba el token de autenticación en el archivo `gradle.properties`:

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

Token stored in separate file

Método 2: almacenar el token de autenticación en un archivo independiente

1. Actualice el archivo `build.gradle` con el siguiente fragmento:

```
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. Para obtener complementos de CodeArtifact, agregue un bloque `pluginManagement` a su archivo `settings.gradle`. El bloque `pluginManagement` debe aparecer antes de cualquier otra declaración en `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. Obtenga un token de autorización de CodeArtifact:

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

4. Escriba el token de autenticación en el archivo que se especificó en su archivo `build.gradle`:

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

Token generated for each run in build.gradle

Método 3: generar un nuevo token de autenticación para cada ejecución ejecutando **`aws`** como un script en línea en **`build.gradle`**

1. Actualice el archivo `build.gradle` con el siguiente fragmento:

```
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. Para obtener complementos de CodeArtifact, agregue un bloque `pluginManagement` a su archivo `settings.gradle`. El bloque `pluginManagement` debe aparecer antes de cualquier otra declaración en `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
            }
        }
    }
}
```

Usar CodeArtifact con mvn

El comando `mvn` se usa para ejecutar compilaciones de Maven. En esta sección se muestra cómo configurar `mvn` para usar un repositorio de CodeArtifact.

Temas

- Extraer dependencias
- Publicar artefactos
- Publicación de artefactos de terceros
- Restrinja las descargas de dependencias de Maven a un repositorio de CodeArtifact
- Información del proyecto Apache Maven

Extraer dependencias

Para configurar `mvn` para recuperar dependencias de un repositorio de CodeArtifact, debe editar el archivo de configuración de Maven, `settings.xml` y, opcionalmente, el POM de su proyecto.

1. Si no lo ha hecho, cree y almacene un token de autenticación de CodeArtifact en una variable de entorno como se describe en Pasar un token de autenticación mediante una variable de entorno para configurar la autenticación en su repositorio de CodeArtifact.
2. En `settings.xml` (normalmente se encuentra en `~/.m2/settings.xml`), añada una sección `<servers>` con una referencia a la variable de entorno `CODEARTIFACT_AUTH_TOKEN` para que Maven pase el token en las solicitudes HTTP.

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

```
        </server>
    </servers>
...
</settings>
```

3. Añada el punto de conexión de la URL de su repositorio de CodeArtifact en un elemento `<repository>`. Puede hacerlo en `settings.xml` o en el archivo POM de su proyecto.

   Puede recuperar el punto de conexión de su repositorio mediante el comando `get-repository-endpoint` AWS CLI.

   Por ejemplo, con un repositorio llamado *my_repo* dentro de un dominio llamado *my_domain*, el comando es el siguiente:

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

   El comando `get-repository-endpoint` devolverá el punto de conexión del repositorio:

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

   Añada el punto de conexión del repositorio a `settings.xml` de la siguiente manera.

   ```
   <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>
```

O bien, puede añadir la sección `<repositories>` a un archivo POM del proyecto para usar CodeArtifact solo para ese proyecto.

```
<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**
>
> Puede usar cualquier valor en el elemento `<id>`, pero debe ser el mismo en los elementos `<server>` y `<repository>`. Esto permite incluir las credenciales especificadas en las solicitudes a CodeArtifact.

Después de realizar estos cambios de configuración, puede crear el proyecto.

```
mvn compile
```

Maven registra la URL completa de todas las dependencias que descarga en la consola.

```
[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)
```

Publicar artefactos

Para publicar un artefacto de Maven con `mvn` en un repositorio de CodeArtifact, también debe editar `~/.m2/settings.xml` y proyectar el POM.

1. Si no lo ha hecho, cree y almacene un token de autenticación de CodeArtifact en una variable de entorno como se describe en Pasar un token de autenticación mediante una variable de entorno para configurar la autenticación en su repositorio de CodeArtifact.
2. Agregue una sección `<servers>` a `settings.xml` con una referencia a la variable de entorno `CODEARTIFACT_AUTH_TOKEN` para que Maven pase el token en las solicitudes HTTP.

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

3. Agregue una sección `<distributionManagement>` al `pom.xml` de su proyecto.

   ```
   <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>
```

Tras realizar estos cambios de configuración, puede crear el proyecto y publicarlo en el repositorio especificado.

```
mvn deploy
```

Utilice `list-package-versions` para comprobar que el paquete se ha publicado correctamente.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven \
  --namespace com.company.framework --package my-package-name
```

Salida de ejemplo:

```
{
    "defaultDisplayVersion": null,
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "my-package-name",
    "versions": [
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

Publicación de artefactos de terceros

Puede publicar artefactos Maven de terceros en un repositorio de CodeArtifact con `mvn deploy:deploy-file`. Esto puede resultar útil para los usuarios que desean publicar artefactos y solo tienen archivos JAR y no tienen acceso al código fuente del paquete o a los archivos POM.

El comando `mvn deploy:deploy-file` generará un archivo POM en función de la información pasada en la línea de comandos.

Publicación de artefactos Maven de terceros

1. Si no lo ha hecho, cree y almacene un token de autenticación de CodeArtifact en una variable de entorno como se describe en Pasar un token de autenticación mediante una variable de entorno para configurar la autenticación en su repositorio de CodeArtifact.
2. Cree un archivo `~/.m2/settings.xml` con los siguientes contenidos:

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

3. Ejecute el comando `mvn deploy:deploy-file`:

```
mvn deploy:deploy-file -DgroupId=commons-cli          \
-DartifactId=commons-cli       \
-Dversion=1.4                  \
-Dfile=./commons-cli-1.4.jar   \
-Dpackaging=jar                \
-DrepositoryId=codeartifact    \
-Durl=https://my_domain-111122223333.d.codeartifact.region.amazonaws.com/
maven/repo-name/
```

> **Note**
>
> El ejemplo anterior publica `commons-cli 1.4`. Modifique los argumentos GroupID, ArtifactID, version y file para publicar un JAR diferente.

Estas instrucciones se basan en los ejemplos de la Guía para implementar archivos JAR de terceros en un repositorio remoto de la documentación de Apache Maven.

Restrinja las descargas de dependencias de Maven a un repositorio de CodeArtifact

Si un paquete no se puede recuperar de un repositorio configurado, de forma predeterminada, el comando `mvn` lo obtiene del central de Maven. Agregue el elemento `mirrors` a `settings.xml` para que `mvn` siempre use su repositorio CodeArtifact.

```
<settings>
  ...
    <mirrors>
      <mirror>
        <id>central-mirror</id>
        <name>CodeArtifact Maven Central mirror</name>
        <url>https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
maven/my_repo/</url>
        <mirrorOf>central</mirrorOf>
      </mirror>
    </mirrors>
  ...
</settings>
```

Si agrega un elemento `mirrors`, también debe tener un elemento `pluginRepository` en su `settings.xml` o `pom.xml`. El siguiente ejemplo busca las dependencias de la aplicación y los complementos de Maven de un repositorio de CodeArtifact.

```
<settings>
...
  <profiles>
    <profile>
      <pluginRepositories>
        <pluginRepository>
          <id>codeartifact</id>
          <name>CodeArtifact Plugins</name>
          <url>https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
maven/my_repo/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
```

```
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
...
</settings>
```

En el siguiente ejemplo, se obtienen las dependencias de la aplicación de un repositorio de CodeArtifact y se obtienen los complementos de Maven del central de Maven.

```
<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>
```

Información del proyecto Apache Maven

Para obtener más información sobre Maven, consulte estos temas en el sitio web del Proyecto Apache Maven:

- Configuración de varios repositorios
- Referencia de configuración

- Gestión de la distribución
- Perfiles

Usar CodeArtifact con deps.edn

Se usa `deps.edn` con `clj` para administrar las dependencias de los proyectos de Clojure. En esta sección se muestra cómo configurar `deps.edn` para usar un repositorio de CodeArtifact.

Temas

- Extraer dependencias
- Publicar artefactos

Extraer dependencias

Para configurar `Clojure` para recuperar dependencias de un repositorio de CodeArtifact, debe editar el archivo de configuración de Maven, `settings.xml`.

1. En `settings.xml`, añada una sección `<servers>` con una referencia a la variable de entorno `CODEARTIFACT_AUTH_TOKEN` para que Clojure pase el token en las solicitudes HTTP.

   > **Note**
   >
   > Clojure espera que el archivo settings.xml esté ubicado en `~/.m2/settings.xml`. Si está en otro lugar, cree el archivo en esta ubicación.

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

2. Si aún no tiene uno, genere un XML POM para su proyecto utilizando `clj -Spom`.
3. En su archivo de configuración `deps.edn`, añada un repositorio que coincida con el identificador del servidor de Maven `settings.xml`.

```
:mvn/repos {
  "clojars" nil
  "central" nil
  "codeartifact" {:url "https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/maven/my_repo/"}
}
```

> **Note**
>
> - `tools.deps` garantiza que los repositorios `central` y `clojars` se comprueben primero para ver si hay bibliotecas de Maven. Después, se revisarán los demás repositorios listados en `deps.edn`.
> - Para evitar que se descarguen directamente desde Clojar y Maven Central, hay que configurar `central` y `clojars` en `nil`.

Asegúrese de tener el token de autenticación CodeArtifact en una variable de entorno (consulte Pasar un token de autenticación mediante una variable de entorno). Al compilar el paquete después de estos cambios, las dependencias en `deps.edn` se obtendrán de CodeArtifact.

Publicar artefactos

1. Actualice su configuración de Maven y `deps.edn` para incluir CodeArtifact como un servidor reconocido por Maven (consulte Extraer dependencias). Puede usar una herramienta como deps-deploy para cargar artefactos en CodeArtifact.
2. En su `build.clj`, añada una tarea `deploy` para cargar los artefactos necesarios en el repositorio `codeartifact` previamente configurado.

```
(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. Publique el artefacto ejecutando el comando `clj -T:build deploy`:

Para obtener más información sobre la modificación de los repositorios predeterminados, consulte Modificación de los repositorios predeterminados en Clojure Deps y CLI Reference Rationale.

Publicación con curl

En esta sección se muestra cómo utilizar el cliente HTTP `curl` para publicar artefactos de Maven en un repositorio de CodeArtifact. Publicar artefactos con `curl` puede resultar útil si no tiene o no desea instalar el cliente Maven en sus entornos.

Publicación de un artefacto de Maven con **`curl`**

1. Obtenga un token de autorización de CodeArtifact siguiendo los pasos descritos en Pasar un token de autenticación mediante una variable de entorno y vuelva a estos pasos.
2. Utilice el siguiente comando `curl` para publicar el JAR en un repositorio de CodeArtifact:

   En cada uno de los comandos `curl` de este procedimiento, sustituya los siguientes marcadores de posición:

   - Sustituya *`my_domain`* por su nombre de dominio de CodeArtifact.
   - Sustituya *`111122223333`* por el ID del propietario de su dominio CodeArtifact.
   - Sustituya *`us-west-2`* por la región en la que reside su dominio de CodeArtifact.
   - Sustituya *`my_repo`* por el nombre de su repositorio de CodeArtifact.

```
curl --request PUT https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/maven/my_repo/com/mycompany/app/my-app/1.0/my-app-1.0.jar \
     --user "aws:$CODEARTIFACT_AUTH_TOKEN" --header "Content-Type: application/
octet-stream" \
     --data-binary @my-app-1.0.jar
```

> **Important**
>
> Debe anteponer el valor del parámetro `--data-binary` con un carácter `@`. Al escribir el valor entre comillas, `@` debe incluirse dentro de las comillas.

3. Utilice el siguiente comando `curl` para publicar el POM en un repositorio de CodeArtifact:

```
curl --request PUT https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/maven/my_repo/com/mycompany/app/my-app/1.0/my-app-1.0.pom \
     --user "aws:$CODEARTIFACT_AUTH_TOKEN" --header "Content-Type: application/
octet-stream" \
     --data-binary @my-app-1.0.pom
```

4. En este punto, el artefacto Maven estará en su repositorio de CodeArtifact con un estado de `Unfinished`. Para poder consumir el paquete, debe estar en el estado `Published`. Puede mover el paquete de `Unfinished` a `Published` cargando un archivo `maven-metadata.xml` en el paquete o llamando a la API UpdatePackageVersionsStatus para cambiar el estado.

   a. Opción 1: use el siguiente comando `curl` para añadir un archivo `maven-metadata.xml` al paquete:

```
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
```

   El siguiente ejemplo muestra el contenido de un archivo `maven-metadata.xml`:

```
<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. Opción 2: actualice el estado del paquete a `Published` con la API `UpdatePackageVersionsStatus`.

```
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
```

Si solo tiene el archivo JAR de un artefacto, puede publicar una versión de paquete consumible en un repositorio de CodeArtifact utilizando `mvn`. Esto puede resultar útil si no tiene acceso al código fuente o al POM del artefacto. Para obtener más información, consulte Publicación de artefactos de terceros.

Uso de sumas de comprobación de Maven

Cuando se publica un artefacto de Maven en un repositorio de AWS CodeArtifact, la suma de comprobación asociada a cada activo o archivo del paquete se utiliza para validar la carga. Algunos ejemplos de activos son los archivos jar, pom y war. Para cada activo, el artefacto de Maven contiene varios archivos de suma de comprobación que utilizan el nombre del activo con una extensión adicional, como `md5` o `sha1`. Por ejemplo, los archivos de suma de comprobación de un archivo denominado `my-maven-package.jar` podrían ser `my-maven-package.jar.md5` y `my-maven-package.jar.sha1`.

> **Note**
>
> Maven usa el término `artifact`. En esta guía, un paquete de Maven es lo mismo que un artefacto de Maven. Para obtener más información, consulte paquete AWS CodeArtifact.

Almacenamiento de sumas de comprobación

CodeArtifact no almacena las sumas de comprobación de Maven como activos. Esto significa que las sumas de comprobación no aparecen como activos individuales en la salida de la API ListPackageVersionAssets. En cambio, las sumas de comprobación calculadas por CodeArtifact están disponibles para cada activo en todos los tipos de suma de comprobación compatibles. Por ejemplo, parte de la respuesta al llamar a ListPackageVersionAssets en la versión del paquete Maven `commons-lang:commons-lang 2.1` es:


```
{
    "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
    }
}
```

Aunque las sumas de comprobación no se almacenan como activos, los clientes de Maven pueden publicar y descargar las sumas de comprobación en las ubicaciones esperadas. Por ejemplo, si `commons-lang:commons-lang 2.1` estuviera en un repositorio llamado `maven-repo`, la ruta URL de la suma de comprobación SHA-256 del archivo JAR sería:

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

Si está cargando paquetes de Maven existentes (por ejemplo, paquetes previamente almacenados en Amazon S3) a CodeArtifact mediante un cliente HTTP genérico, por ejemplo `curl`, no es necesario cargar las sumas de comprobación. CodeArtifact los generará automáticamente. Si desea comprobar que los activos se han cargado correctamente, puede utilizar la operación de la API ListPackageVersionAssets para comparar las sumas de comprobación de la respuesta con los valores de las sumas de comprobación originales de cada activo.

La suma de comprobación no coincide durante la publicación

Además de los activos y las sumas de comprobación, los artefactos de Maven también contienen un archivo `maven-metadata.xml`. La secuencia de publicación normal de un paquete de Maven es que todos los activos y las sumas de comprobación se carguen primero y, a continuación, se cargue `maven-metadata.xml`. Por ejemplo, la secuencia de publicación de la versión `commons-lang 2.1` del paquete Maven descrita anteriormente, suponiendo que el cliente estuviera configurado para publicar archivos de suma de comprobación SHA-256, sería:

```
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
```

Al cargar el archivo de suma de control de un activo, como un archivo JAR, la solicitud de carga de la suma de control fallará con una respuesta de 400 (solicitud errónea) si no coincide el valor de la suma de control cargado y el valor de la suma de comprobación calculado por CodeArtifact. Si el activo correspondiente no existe, la solicitud fallará y generará una respuesta 404 (no encontrado). Para evitar este error, primero debe cargar el activo y, a continuación, cargar la suma de comprobación.

Cuando `maven-metadata.xml` se carga, CodeArtifact normalmente cambia el estado de la versión del paquete Maven de `Unfinished` a `Published`. Si se detecta una discrepancia en la suma de

comprobación para algún activo, CodeArtifact devolverá un 400 (solicitud errónea) en respuesta a la solicitud de publicación `maven-metadata.xml`. Este error puede provocar que el cliente deje de cargar los archivos para esa versión del paquete. Si esto ocurre y el archivo `maven-metadata.xml` no se carga, no se podrá descargar ningún activo de la versión del paquete que ya se haya cargado. Esto se debe a que el estado de la versión del paquete no está establecido en `Published` y permanece `Unfinished`.

CodeArtifact permite agregar más activos a una versión del paquete Maven incluso después de que se haya cargado `maven-metadata.xml` y el estado de la versión del paquete se haya establecido en `Published`. En este estado, una solicitud para cargar un archivo de suma de comprobación que no coincida también fallará y recibirá una respuesta de 400 (solicitud errónea). Sin embargo, dado que el estado de la versión del paquete ya está establecido en `Published`, puede descargar cualquier recurso del paquete, incluidos aquellos en los que no se pudo cargar el archivo de suma de comprobación. Al descargar una suma de comprobación para un activo en el que no se pudo cargar el archivo de suma de comprobación, el valor de la suma de comprobación que reciba el cliente será el valor de la suma de comprobación calculado por CodeArtifact en función de los datos del activo cargado.

Las comparaciones de sumas de comprobación de CodeArtifact distinguen entre mayúsculas y minúsculas, y las sumas de comprobación calculadas por CodeArtifact están formateadas en minúsculas. Por lo tanto, si se carga la suma de comprobación `909FA780F76DA393E992A3D2D495F468`, fallará y la suma de comprobación no coincidirá porque CodeArtifact no la trata como igual a `909fa780f76da393e992a3d2d495f468`.

Recuperarse de discrepancias en las sumas de comprobación

Si se produce un error al cargar una suma de comprobación debido a una discrepancia en la suma de comprobación, intente realizar una de las siguientes acciones para recuperarla:

- Vuelva a ejecutar el comando que publica el artefacto de Maven. Esto podría funcionar si un problema de red dañara el archivo de suma de comprobación. Si esto resuelve el problema de la red, la suma de comprobación coincide y la descarga se ha realizado correctamente.
- Elimine la versión del paquete y, a continuación, vuelva a publicarla. Para obtener más información, consulte DeletePackageVersions en la referencia de la API de AWS CodeArtifact.

Uso de instantáneas de Maven

Una instantánea de Maven es una versión especial de un paquete de Maven que hace referencia al código de rama de producción más reciente. Es una versión de desarrollo que precede a la versión de lanzamiento final. Puede identificar una versión instantánea de un paquete de Maven por el sufijo `SNAPSHOT` que se adjunta a la versión del paquete. Por ejemplo, la instantánea de la versión `1.1` es `1.1-SNAPSHOT`. Para obtener más información, consulte ¿Qué es una versión de SNAPSHOT? en el sitio web del Proyecto Apache Maven.

AWS CodeArtifact admite la publicación y el consumo de instantáneas de Maven. Las instantáneas únicas que utilizan un número de versión basado en el tiempo son las únicas instantáneas compatibles. CodeArtifact no admite instantáneas no únicas generadas por los clientes de Maven 2. Puede publicar una instantánea de Maven compatible en cualquier repositorio de CodeArtifact.

Temas

Publicación de instantáneas en CodeArtifact

AWSCodeArtifact admite los patrones de solicitud que los clientes, por ejemplo `mvn`, utilizan al publicar instantáneas. Por eso, puede seguir la documentación de su herramienta de compilación o administrador de paquetes sin tener una comprensión detallada de cómo se publican las instantáneas de Maven. Si está haciendo algo más complejo, en esta sección se describe en detalle cómo CodeArtifact gestiona las instantáneas.

Cuando se publica una instantánea de Maven en un repositorio de CodeArtifact, su versión anterior se conserva en una nueva versión llamada compilación. Cada vez que se publica una instantánea de Maven, se crea una nueva versión de compilación. Todas las versiones anteriores de una instantánea se mantienen en sus versiones de compilación. Cuando se publica una instantánea de Maven, el estado de la versión del paquete se establece en `Published` y el estado de la

compilación que contiene la versión anterior se establece en `Unlisted`. Este comportamiento solo se aplica a las versiones del paquete de Maven en las que la versión del paquete tiene un sufijo `-SNAPSHOT`.

Por ejemplo, las versiones instantáneas de un paquete Maven denominado `com.mycompany.myapp:pkg-1` se cargan en un repositorio de CodeArtifact denominado `my-maven-repo`. La versión de instantánea es `1.0-SNAPSHOT`. Hasta el momento, no se ha publicado ninguna versión de `com.mycompany.myapp:pkg-1`. En primer lugar, los activos de la versión inicial se publican en las siguientes rutas:

```
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
```

Tenga en cuenta que la marca de tiempo `20210728.194552-1` la genera el cliente que publica las compilaciones instantáneas.

Una vez cargados los archivos .pom y .jar, la única versión de `com.mycompany.myapp:pkg-1` que existe en el repositorio es `1.0-20210728.194552-1`. Esto ocurre aunque lo sea la versión especificada en la ruta anterior sea `1.0-SNAPSHOT`. El estado de la versión del paquete en este momento es `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"
}
```

A continuación, el cliente carga el archivo `maven-metadata.xml` de la versión del paquete:

```
PUT my-maven-repo/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/maven-metadata.xml
```

Cuando el archivo maven-metadata.xml se carga correctamente, CodeArtifact crea la versión del paquete `1.0-SNAPSHOT` y establece la versión `1.0-20210728.194552-1` en `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"
}
```

En este punto, la versión de instantánea `1.0-SNAPSHOT` se puede consumir en una compilación. Si bien hay dos versiones de `com.mycompany.myapp:pkg-1` en el repositorio `my-maven-repo`, ambas contienen los mismos activos.

```
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"
]
```

Ejecutar el mismo comando `list-package-version-assets` como se mostró anteriormente con el parámetro `--package-version` cambiado a `1.0-20210728.194552-1` da como resultado una salida idéntica.

A medida que se añaden compilaciones adicionales de `1.0-SNAPSHOT` al repositorio, se crea una nueva versión del paquete `Unlisted` para cada nueva compilación. Los activos de la versión `1.0-SNAPSHOT` se actualizan cada vez para que la versión siempre haga referencia a la última compilación de esa versión. La actualización de `1.0-SNAPSHOT` con los activos más recientes se inicia cargando el archivo `maven-metadata.xml` de la nueva compilación.

Consumo de versiones de instantánea

Si solicita una instantánea, se devuelve la versión con el estado `Published`. Esta es siempre la versión más reciente de la instantánea de Maven. También puede solicitar una compilación concreta de una instantánea utilizando el número de versión de la compilación (por ejemplo, `1.0-20210728.194552-1`) en lugar de la versión de la instantánea (por ejemplo, `1.0-SNAPSHOT`) en la ruta URL. Para ver las versiones de compilación de una instantánea de Maven, use la API ListPackageVersions en la Guía de la API CodeArtifact y establezca el parámetro de estado en `Unlisted`.

Eliminación de versiones de instantánea

Para eliminar todas las versiones compiladas de una instantánea de Maven, utilice la API DeletePackageVersions y especifique las versiones que desea eliminar.

Publicación de instantáneas con curl

Si tiene versiones de instantánea existentes almacenadas en Amazon Simple Storage Service (Amazon S3) u otro producto de repositorio de artefactos, puede que desee volver a publicarlas en AWS CodeArtifact. Debido a la forma en que CodeArtifact admite instantáneas de Maven (consulte Publicación de instantáneas en CodeArtifact), publicar instantáneas con un cliente HTTP genérico como `curl` es más complejo que publicar versiones de lanzamiento de Maven como se describe en Publicación con curl. Tenga en cuenta que esta sección no es relevante si está creando e implementando versiones instantáneas con un cliente Maven como `mvn` o `gradle`. Debe seguir la documentación de ese cliente.

La publicación de una versión instantánea implica publicar una o más compilaciones de una versión instantánea. En CodeArtifact, si hay n compilaciones de una versión de instantánea, habrá n + 1 versiones de CodeArtifact: n versiones de compilación, todas con un estado de `Unlisted`, y una versión instantánea (la última compilación publicada) con un estado de `Published`. La versión instantánea (es decir, la versión con una cadena de versión que contiene «-SNAPSHOT») contiene un conjunto de activos idéntico al de la última compilación publicada. La forma más sencilla de crear esta estructura usando `curl` es la siguiente:

1. Publique todos los activos de todas las compilaciones utilizando `curl`.
2. Publique el archivo `maven-metadata.xml` de la última compilación (es decir, la compilación con la marca de fecha y hora más reciente) con `curl`. Esto creará una versión con «`-SNAPSHOT`» en la cadena de la versión y con el conjunto correcto de activos.
3. Use la API UpdatePackageVersionsStatus para establecer el estado de todas las versiones de compilación que no sean las más recientes en `Unlisted`.

Use los siguientes comandos `curl` para publicar activos de instantáneas (como archivos.jar y .pom) para la versión `1.0-SNAPSHOT` de instantánea de un paquete `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
```

Al usar estos ejemplos:

- Sustituya *`my_domain`* por su nombre de dominio de CodeArtifact.
- Sustituya *`111122223333`* por el ID Cuenta de AWS del propietario de su dominio CodeArtifact.
- Sustituya *`us-west-2`* por la Región de AWS en la que reside su dominio de CodeArtifact.
- Sustituya *`my_maven_repo`* por el nombre de su repositorio de CodeArtifact.

> **Important**
>
> Debe anteponer el valor del parámetro `--data-binary` con el carácter `@`. Al escribir el valor entre comillas, `@` debe incluirse dentro de las comillas.

Es posible que tenga que cargar más de dos recursos para cada compilación. Por ejemplo, puede haber archivos Javadoc y JAR de origen además del JAR principal y `pom.xml`. No es necesario publicar archivos de suma de verificación para los activos de la versión del paquete porque CodeArtifact genera automáticamente sumas de verificación para cada activo cargado. Para verificar que los activos se cargaron correctamente, busque las sumas de verificación generadas mediante el comando `list-package-version-assets` y compárelas con las sumas de verificación originales. Para obtener más información acerca de cómo administra CodeArtifact las sumas de verificación de Maven, consulte Uso de sumas de comprobación de Maven.

Utilice el siguiente comando curl para publicar el archivo `maven-metadata.xml` de la última versión de compilación:

```
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
```

El archivo `maven-metadata.xml` debe hacer referencia al menos a uno de los activos de la última versión de compilación del elemento `<snapshotVersions>`. Además, el valor `<timestamp>` debe estar presente y debe coincidir con la marca de tiempo de los nombres de los archivos de los activos. Por ejemplo, para la compilación `20210729.171330-2` publicada anteriormente, el contenido de `maven-metadata.xml` sería:

```
<?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>
```

Una vez publicada `maven-metadata.xml`, el último paso consiste en configurar todas las demás versiones de compilación (es decir, todas las versiones de compilación excepto la última) para que tengan un estado de versión de paquete de `Unlisted`. Por ejemplo, si la versión `1.0-SNAPSHOT` tiene dos compilaciones, siendo la primera compilación `20210728.194552-1`, el comando para configurar esa compilación en `Unlisted` es:

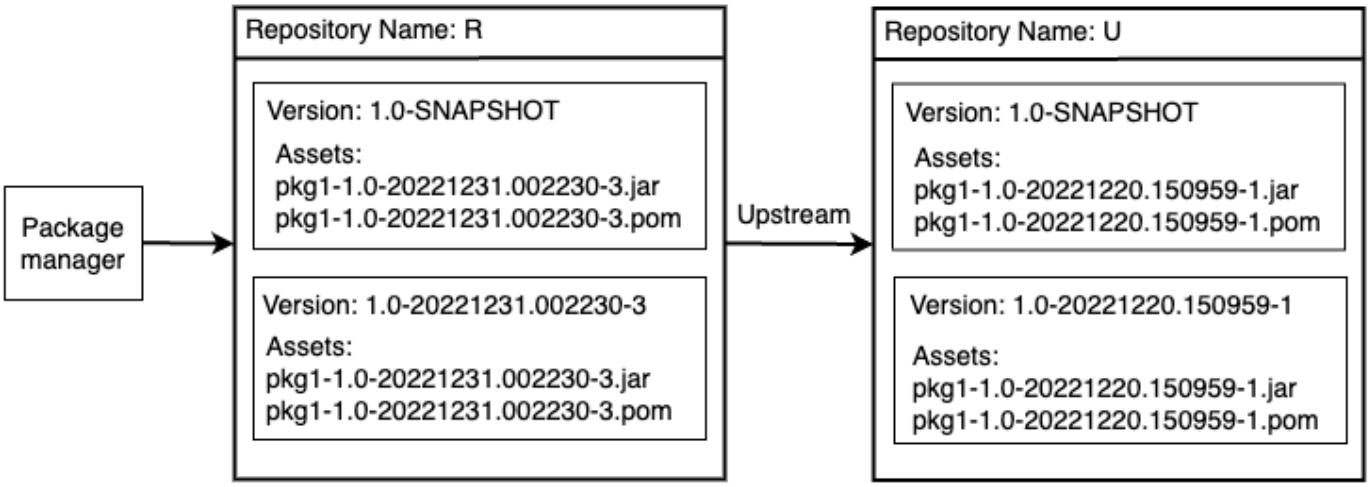
```
aws codeartifact update-package-versions-status --domain my-domain --domain-owner
 111122223333 \
   --repository my-maven-repo --format maven --namespace com.mycompany.app --package
 pkg-1 \
   --versions 1.0-20210728.194552-1 --target-status Unlisted
```

Instantáneas y conexiones externas

Las instantáneas de Maven no se pueden obtener de un repositorio público de Maven a través de una conexión externa AWS. CodeArtifact solo admite la importación de versiones de Maven.

Instantáneas y repositorios originales

En general, las instantáneas de Maven funcionan de la misma manera que las versiones de lanzamiento de Maven cuando se utilizan con repositorios originales. Por ejemplo, supongamos que hay dos repositorios en un dominio de AWS CodeArtifact `R` y `U`, donde `U` es un ascendente de `R`. En este caso, puede publicar libremente compilaciones instantáneas de un paquete determinado (por ejemplo, `1.0-SNAPSHOT` de `com.mycompany.app:pkg-1`) tanto en `R` como en `U`. Sin embargo, hay algunos comportamientos importantes que hay que tener en cuenta a la hora de consumir compilaciones instantáneas desde `R` (el repositorio descendente).



1. Si `1.0-SNAPSHOT` está presente en `R`, solo se pueden recuperar los activos de `1.0-SNAPSHOT` en `R` con un administrador de paquetes configurado para extraer paquetes de `R`. No se puede recuperar un activo de `1.0-SNAPSHOT` en `U` a través de `R`. Esto se debe a que la versión instantánea de `U` aparece sombreada por la versión de `R`. Este comportamiento es idéntico al de las versiones de lanzamiento de Maven y al comportamiento de otros formatos de paquetes. En el diagrama, un `GET` de `/maven/R/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/pkg-1-1.0-20221231.002230-3.jar` devolverá un código de respuesta HTTP 200 (OK), pero un `GET` de `/maven/R/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/pkg-1-1.0-20221220.150959-1.jar` devolverá un código de respuesta HTTP 404 (no encontrado).
2. Si `1.0-SNAPSHOT` está presente en `U` pero no en `R`, puede extraer activos `1.0-SNAPSHOT` de `R`. Esto hará que `1.0-SNAPSHOT` se conserve en `R`, al igual que en una versión de lanzamiento.
3. Una vez retenido `1.0-SNAPSHOT` en `R`, puede publicar compilaciones adicionales de `1.0-SNAPSHOT` en `U`. Sin embargo, no se podrá acceder a ellas desde `R` debido al comportamiento descrito en el punto (1). Esto significa que la razón estándar para usar versiones instantáneas, es decir, consumir la última versión de una dependencia a través de una versión instantánea específica, no funciona como se esperaba en una relación anterior. Aunque se publiquen nuevas versiones de `1.0-SNAPSHOT` en `U`, los consumidores no pueden acceder a la versión más reciente de `1.0-SNAPSHOT` de `R`. Para solucionar este problema, elimine periódicamente la versión `1.0-SNAPSHOT` en `R` o configure el cliente para que extraiga versiones de `1.0-SNAPSHOT` de `U`.
4. Se puede acceder a las versiones de compilación de la instantánea `Unlisted` desde el repositorio posterior. En el diagrama, un `GET` de `/maven/R/com/mycompany/myapp/pkg-1/1.0-20221220.150959-1/pkg-1-1.0-20221220.150959-1.jar` devolverá un

código de respuesta de 200 (OK). Aunque esto solicite un activo presente en el repositorio original, dado que la versión se direcciona mediante la cadena de versión compilada (`1.0-20221220.150959-1`), el activo se puede recuperar a través del repositorio descendente. Este `GET` también hará que la versión `1.0-20221220.150959-1` se conserve en `R`, con un estado de versión del paquete de `Unlisted`.

Solicitud de paquetes de Maven desde conexiones ascendentes y externas

Importación de nombres de activos estándar

Al importar una versión de paquete de Maven desde un repositorio público, como el central de Maven, AWS CodeArtifact intenta importar todos los activos de esa versión de paquete. Como se describe en Solicitar una versión de paquete con repositorios ascendentes, la importación se produce cuando:

- Un cliente solicita un activo de Maven desde un repositorio de CodeArtifact.
- La versión del paquete aún no está presente en el repositorio ni en sus fuentes de distribución.
- Hay una conexión externa accesible a un repositorio público de Maven.

Aunque es posible que el cliente solo haya solicitado un activo, CodeArtifact intenta importar todos los activos que puede encontrar para esa versión del paquete. La forma en que CodeArtifact descubre qué activos están disponibles para una versión de paquete Maven depende del repositorio público en particular. Algunos repositorios públicos de Maven admiten la solicitud de una lista de activos, pero otros no. Para los repositorios que no proporcionan una forma de enumerar los activos, CodeArtifact genera un conjunto de nombres de activos que probablemente existan. Por ejemplo, cuando se solicita cualquier activo de la versión `junit 4.13.2` del paquete Maven, CodeArtifact intentará importar los siguientes activos:

- `junit-4.13.2.pom`
- `junit-4.13.2.jar`
- `junit-4.13.2-javadoc.jar`
- `junit-4.13.2-sources.jar`

Importación de nombres de activos no estándar

Cuando un cliente de Maven solicita un activo que no coincide con uno de los patrones descritos anteriormente, CodeArtifact comprueba si ese activo está presente en el repositorio público. Si el activo está presente, se importará y se añadirá al registro de versiones del paquete existente, si existe alguno. Por ejemplo, la versión `com.android.tools.build:aapt2 7.3.1-8691043` del paquete Maven contiene los siguientes activos:

- `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`

Cuando un cliente solicita el archivo POM, si CodeArtifact no puede enumerar los activos de la versión del paquete, el POM será el único activo importado. Esto se debe a que ninguno de los demás activos coincide con los patrones de nombres de activos estándar. Sin embargo, cuando el cliente solicita uno de los activos JAR, ese activo se importará y agregará a la versión del paquete existente almacenada en CodeArtifact. Las versiones de los paquetes tanto en el repositorio más avanzado (el repositorio en el que el cliente realizó la solicitud) como en el repositorio con la conexión externa adjunta se actualizarán para incluir el nuevo activo, tal y como se describe en Retención de paquetes de repositorios ascendentes.

Normalmente, una vez que la versión de un paquete se conserva en un repositorio de CodeArtifact, no se ve afectada por los cambios en los repositorios anteriores. Para obtener más información, consulte Retención de paquetes de repositorios ascendentes. Sin embargo, el comportamiento de los activos de Maven con nombres no estándar descritos anteriormente es una excepción a esta regla. Si bien la versión posterior del paquete no cambiará sin que un cliente solicite un activo adicional, en esta situación, la versión del paquete retenido se modifica después de ser retenida inicialmente y, por lo tanto, no es inmutable. Este comportamiento es necesario porque, de lo contrario, no se podría acceder a los activos de Maven con nombres no estándar a través de CodeArtifact. El comportamiento también se habilita si se agregan a una versión del paquete Maven en un repositorio público después de que la versión del paquete se haya conservado en un repositorio de CodeArtifact.

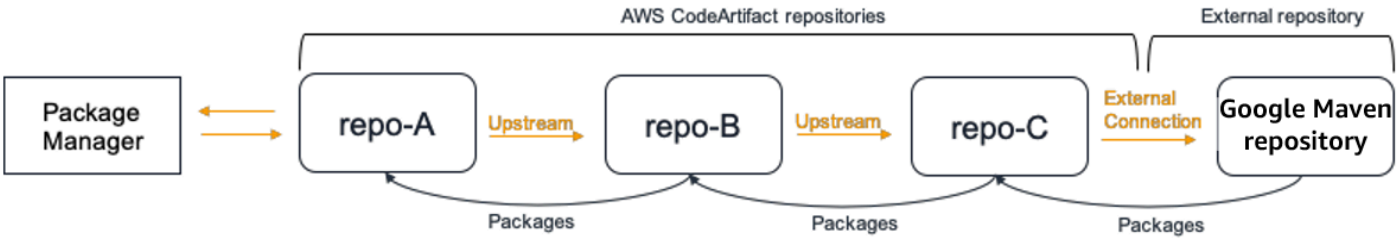
Comprobar el origen de los activos

Al agregar un nuevo activo a una versión del paquete Maven previamente retenida, CodeArtifact confirma que el origen de la versión del paquete retenido es el mismo que el origen del nuevo activo. Esto evita crear una versión de paquete «mixta» en la que los diferentes activos se originen en diferentes repositorios públicos. Sin esta comprobación, la mezcla de activos podría producirse si la versión de un paquete de Maven se publica en más de un repositorio público y esos repositorios forman parte del gráfico ascendente de un repositorio de CodeArtifact.

Importación de nuevos activos y el estado de las versiones de los paquetes en los repositorios originales

El estado de la versión del paquete de las versiones del paquete en los repositorios ascendentes puede impedir que CodeArtifact conserve esas versiones en los repositorios descendentes.

Por ejemplo, supongamos que un dominio tiene tres repositorios: `repo-A`, `repo-B` y `repo-C`, donde `repo-B` es una fuente ascendente de `repo-A` y `repo-C` ascendente de `repo-B`.



La versión de paquete `7.3.1` del paquete Maven `com.android.tools.build:aapt2` está presente en `repo-B` y tiene un estado de `Published`. No está presente en `repo-A`. Si un cliente solicita un activo de esta versión de paquete de `repo-A`, la respuesta será 200 (OK) y se conservará la versión `7.3.1` del paquete Maven en `repo-A`. Sin embargo, si el estado de la versión del paquete `7.3.1` en `repo-B` es `Archived` o `Disposed`, la respuesta será 404 (No encontrado) porque los activos de las versiones del paquete en esos dos estados no se pueden descargar.

Tenga en cuenta que si configura el control de origen del paquete como `upstream=BLOCK` en `com.android.tools.build:aapt2`, `repo-A`, `repo-B` y `repo-C` evitarán que se obtengan nuevos recursos para todas las versiones de ese paquete de `repo-A`, independientemente del estado de la versión del paquete.

Solución de problemas de Maven

La siguiente información puede ayudarle a solucionar problemas comunes con Maven y CodeArtifact.

Deshabilite las transferencias paralelas para corregir el error 429: demasiadas solicitudes

A partir de la versión 3.9.0, Maven carga los artefactos del paquete en paralelo (hasta 5 archivos a la vez). Esto puede provocar que CodeArtifact responda ocasionalmente con un código de respuesta de error 429 (Demasiadas solicitudes). Si encuentra este error, puede desactivar las transferencias paralelas para solucionarlo.

Para deshabilitar las transferencias paralelas, establezca la propiedad `aether.connector.basic.parallelPut` en `false` en su perfil en su archivo `settings.xml` como se muestra en el siguiente ejemplo:

```
<settings>
    <profiles>
        <profile>
            <id>default</id>
            <properties>
                <aether.connector.basic.parallelPut>false</
aether.connector.basic.parallelPut>
            </properties>
        </profile>
    </profiles>
<settings>
```

Para obtener más información, consulte las opciones de configuración de Artifact Resolver en la documentación de Maven.

Uso de CodeArtifact con NuGet

En estos temas se describe cómo consumir y publicar paquetes `NuGet` mediante CodeArtifact.

> **Note**
>
> AWS CodeArtifact solo es compatible con la versión 4.8 y posteriores de NuGet.exe.

Temas

Uso de CodeArtifact con Visual Studio

Puede consumir paquetes de CodeArtifact directamente en Visual Studio con el proveedor de credenciales de CodeArtifact. El proveedor de credenciales simplifica la configuración y la autenticación de los repositorios de CodeArtifact en Visual Studio y está disponible en AWS Toolkit for Visual Studio.

> **Note**
>
> AWS Toolkit for Visual Studio no está disponible para Visual Studio para Mac.

Para configurar y usar NuGet con herramientas CLI, consulte Usar CodeArtifact con la CLI de nuget o dotnet.

Temas

Configurar Visual Studio con el proveedor de credenciales CodeArtifact

El proveedor de credenciales de CodeArtifact simplifica la configuración y la autenticación continua entre CodeArtifact y Visual Studio. Los tokens de autenticación de CodeArtifact son válidos durante un máximo de 12 horas. Para evitar tener que actualizar el token manualmente mientras se trabaja en Visual Studio, el proveedor de credenciales busca periódicamente un nuevo token antes de que caduque el token actual.

> **Important**
>
> Para usar el proveedor de credenciales, asegúrese de borrar del archivo todas las credenciales de AWSCodeArtifact existentes estén borradas de su archivo `nuget.config` que puedan haberse agregado manualmente o ejecutando `aws codeartifact login` para configurar NuGet previamente.

Utilice CodeArtifact en Visual Studio con AWS Toolkit for Visual Studio

1. Instale AWS Toolkit for Visual Studio siguiendo los siguientes pasos. El kit de herramientas es compatible con Visual Studio 2017 y 2019 siguiendo estos pasos. AWS CodeArtifact no es compatible con Visual Studio 2015 y versiones anteriores.
    1. El kit de herramientas de Visual Studio para Visual Studio 2017 y Visual Studio 2019 se distribuye en Visual Studio Marketplace. También puede instalar y actualizar el kit de herramientas dentro de Visual Studio usando Herramientas ≫ Extensiones y actualizaciones (Visual Studio 2017) o Extensiones ≫ Administrar extensiones (Visual Studio 2019).
    2. Una vez instalado el kit de herramientas, ábralo eligiendo AWSExplorer en el menú Ver.
2. Configure el Toolkit for Visual Studio con sus credenciales AWS siguiendo los pasos que se indican en Proporcionar credenciales de AWS en la Guía del usuario de AWS Toolkit for Visual Studio.
3. (Opcional) Establezca el perfil AWS que quiere usar con CodeArtifact. Si no se establece, CodeArtifact utilizará el perfil predeterminado. Para configurar el perfil, vaya a Herramientas > NuGet Package Manager > Seleccionar perfil AWS de CodeArtifact.
4. Agregue su repositorio de CodeArtifact como fuente de paquete en Visual Studio.
    1. Navegue hasta su repositorio en la ventana AWS Explorer, haga clic con el botón derecho y seleccione `Copy NuGet Source Endpoint`.

2. Use el comando Herramientas > Opciones y desplácese hasta NuGet Package Manager.
3. Seleccione el nodo Fuentes de paquetes.
4. Seleccione +, edite el nombre y pegue el punto de conexión de la URL del repositorio copiado en el paso 3a en el cuadro Fuente y seleccione Actualizar.
5. Seleccione la casilla de verificación de la fuente de paquetes recién agregada para habilitarla.

> **Note**
>
> Recomendamos añadir una conexión externa a NuGet.org en su repositorio de CodeArtifact y deshabilitar el origen de paquetes nuget.org en Visual Studio. Cuando utilice una conexión externa, todos los paquetes extraídos de Nuget.org se almacenarán en su repositorio de CodeArtifact. Si NuGet.org deja de estar disponible, las dependencias de su aplicación seguirán estando disponibles para las compilaciones de CI y el desarrollo local. Para obtener más información sobre las conexiones externas, use Conectar un repositorio CodeArtifact a un repositorio público.

5. Reinicie Visual Studio para que los cambios surtan efecto.

Tras la configuración, Visual Studio puede consumir paquetes de su repositorio de CodeArtifact, de cualquiera de sus repositorios anteriores o de NuGet.org si ha agregado una conexión externa. Para obtener más información sobre cómo buscar e instalar paquetes NuGet en Visual Studio, consulte Instalar y administrar paquetes en Visual Studio mediante el administrador de paquetes NuGet en la documentación de NuGet.

Utilizar la consola del administrador de paquetes de Visual Studio

La consola del administrador de paquetes de Visual Studio no utilizará la versión de Visual Studio del proveedor de credenciales CodeArtifact. Para usarla, tendrá que configurar el proveedor de credenciales de línea de comandos. Para obtener más información, consulte Usar CodeArtifact con la CLI de nuget o dotnet.

Usar CodeArtifact con la CLI de nuget o dotnet

Puede usar herramientas CLI como `nuget` y `dotnet` para publicar y consumir paquetes de CodeArtifact. Este documento proporciona información sobre la configuración de las herramientas CLI y su uso para publicar o consumir paquetes.

Temas

Configurar la CLI de nuget o dotnet

Puede configurar la CLI de nuget o dotnet con el proveedor de credenciales NuGet de CodeArtifact, con la AWS CLI o manualmente. Se recomienda encarecidamente configurar NuGet con el proveedor de credenciales para simplificar la configuración y continuar con la autenticación.

Método 1: configurar con el proveedor de credenciales NuGet de CodeArtifact

El proveedor de credenciales NuGet de CodeArtifact simplifica la autenticación y la configuración de CodeArtifact con las herramientas CLI de NuGet. Los tokens de autenticación de CodeArtifact son válidos durante un máximo de 12 horas. Para evitar tener que actualizar manualmente el token mientras se usa la CLI nuget o dotnet, el proveedor de credenciales busca periódicamente un nuevo token antes de que caduque el token actual.

> **Important**
>
> Para usar el proveedor de credenciales, asegúrese de borrar del archivo todas las credenciales de AWSCodeArtifact existentes estén borradas de su archivo `nuget.config` que puedan haberse agregado manualmente o ejecutando `aws codeartifact login` para configurar NuGet previamente.

Instalar y configurar el proveedor de credenciales NuGet de CodeArtifact

dotnet

1. Descargue la última versión de la herramienta AWS.CodeArtifact.Nuget.CredentialProvider de NuGet.org con el siguiente comando `dotnet`.

```
dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
```

2. Use el comando `codeartifact-creds install` para copiar el proveedor de credenciales a la carpeta de complementos de NuGet.

```
dotnet codeartifact-creds install
```

3. (Opcional): Defina el perfil AWS que quiere usar con el proveedor de credenciales. Si no se establece, el proveedor de credenciales utilizará el perfil predeterminado. Para obtener más información sobre los perfiles AWS CLI, consulte Perfiles con nombre.

```
dotnet codeartifact-creds configure set profile profile_name
```

nuget

Realice los siguientes pasos para usar la CLI de NuGet para instalar el proveedor de credenciales NuGet de CodeArtifact desde un bucket de Amazon S3 y configurarlo. El proveedor de credenciales utilizará el perfil AWS CLI predeterminado. Para obtener más información sobre los perfiles, consulte Perfiles con nombre.

1. Descargue la última versión del proveedor de credenciales NuGet de CodeArtifact (codeartifact-nuget-credentialprovider.zip) desde un bucket de Amazon S3.

   Para ver y descargar versiones anteriores, consulte Versiones del proveedor de credenciales NuGet de CodeArtifact.

2. Descomprima el archivo.
3. Copie la carpeta AWS.CodeArtifact.NuGetCredentialProvider de la carpeta netfx a `%user_profile%/.nuget/plugins/netfx/` en Windows o `~/.nuget/plugins/netfx` en Linux o macOS.
4. Copie la carpeta AWS.CodeArtifact.NuGetCredentialProvider de la carpeta netcore a `%user_profile%/.nuget/plugins/netcore/` en Windows o `~/.nuget/plugins/netcore` en Linux o macOS.

Después de crear un repositorio y configurar el proveedor de credenciales, puede usar las herramientas de la CLI `nuget` o `dotnet` para instalar y publicar paquetes. Para obtener más información, consulte Consumir paquetes NuGet de CodeArtifact y Publicar paquetes NuGet en CodeArtifact.

Método 2: configurar nuget o dotnet con el comando login

El comando `codeartifact login` de AWS CLI añade un punto de conexión de repositorio y un token de autorización a su archivo de configuración de NuGet, lo que permite que nuget o dotnet se conecten a su repositorio de CodeArtifact. Esto modificará la configuración de NuGet a nivel de usuario, que se encuentra en `%appdata%\NuGet\NuGet.Config` para Windows y `~/.config/NuGet/NuGet.Config` o `~/.nuget/NuGet/NuGet.Config` para Mac/Linux. Para obtener más información sobre las configuraciones de NuGet, consulte Configuraciones comunes de NuGet.

Configurar nuget o dotnet con el comando **`login`**

1. Configure sus credenciales AWS para usarlas con AWS CLI, tal y como se describe en Introducción a CodeArtifact.
2. Asegúrese de que la herramienta (`nuget` o `dotnet`) NuGet CLI se haya instalado y configurado correctamente. Para obtener instrucciones, consulte la documentación de nuget o dotnet.
3. Usa el comando `login` CodeArtifact para obtener las credenciales y usarlas con NuGet.

> **Note**
>
> Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

dotnet

> **Important**
>
> Usuarios de Linux y macOS: dado que el cifrado no es compatible con plataformas que no son de Windows, las credenciales recuperadas se almacenarán como texto sin formato en el archivo de configuración.

```
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
```

El comando de inicio de sesión hará lo siguiente:

- Obtener un token de autorización de CodeArtifact con sus credenciales AWS.
- Actualizar la configuración de NuGet a nivel de usuario con una nueva entrada para la fuente del paquete NuGet. La fuente que apunta al punto de conexión de su repositorio CodeArtifact se llamará *domain_name/repo_name*.

El período de autorización predeterminado después de una llamada `login` es de 12 horas y `login` debe invocarse para actualizar periódicamente el token. Para obtener más información sobre el token de autorización creado con el comando `login`, consulte Tokens creados con el comando `login`.

Después de crear un repositorio y configurar la autenticación, puede utilizar los clientes CLI `nuget`, `dotnet` o `msbuild` para instalar y publicar paquetes. Para obtener más información, consulte Consumir paquetes NuGet de CodeArtifact y Publicar paquetes NuGet en CodeArtifact.

Método 3: configurar nuget o dotnet sin el comando login

Para la configuración manual, debe añadir un punto de conexión de repositorio y un token de autorización a su archivo de configuración de NuGet para permitir que nuget o dotnet se conecten a su repositorio de CodeArtifact.

Configure manualmente nuget o dotnet para que se conecten a su repositorio de CodeArtifact.

1. Determine el punto de conexión del repositorio CodeArtifact mediante el comando `get-repository-endpoint` AWS CLI.

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

   Ejemplo de resultados:

   ```
   {
   ```

```
   "repositoryEndpoint": "https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/nuget/my_repo/"
}
```

2. Obtenga un token de autorización para conectarse a su repositorio desde su administrador de paquetes mediante el comando `get-authorization-token` AWS CLI.

```
aws codeartifact get-authorization-token --domain my_domain
```

   Ejemplo de resultados:

```
{
   "authorizationToken": "eyJ2I...viOw",
   "expiration": 1601616533.0
}
```

3. Crea la URL completa del punto de conexión del repositorio agregando `/v3/index.json` a la URL devuelta por `get-repository-endpoint` en el paso 3.
4. Configure nuget o dotnet para usar el punto de conexión del repositorio del paso 1 y el token de autorización del paso 2.

> **Note**
>
> La URL de origen debe terminar en `/v3/index.json` para que nuget o dotnet se conecten correctamente a un repositorio de CodeArtifact.

   dotnet

   Usuarios de Linux y macOS: dado que el cifrado no se admite en plataformas que no sean Windows, debe añadir la marca `--store-password-in-clear-text` al siguiente comando. Tenga en cuenta que esto almacenará la contraseña como texto sin formato en el archivo de configuración.

```
dotnet nuget add source https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/nuget/my_repo/v3/index.json --name packageSourceName --
password eyJ2I...viOw --username aws
```

> **Note**
>
> Para actualizar una fuente existente, utilice el comando `dotnet nuget update source`.

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
```

Ejemplo de resultados:

```
Package source with Name: domain_name/repo_name added successfully.
```

Consumir paquetes NuGet de CodeArtifact

Una vez que haya configurado NuGet con CodeArtifact, puede consumir los paquetes NuGet que estén almacenados en su repositorio de CodeArtifact o en uno de sus repositorios anteriores.

Para consumir una versión de paquete de un repositorio de CodeArtifact o de uno de sus repositorios anteriores con `nuget` o `dotnet`, ejecute el siguiente comando sustituyendo *`packageName`* por el nombre del paquete que desea consumir y *`packageSourceName`* por el nombre de la fuente de su repositorio de CodeArtifact en su archivo de configuración de NuGet. Si utilizó el comando `login` para configurar la configuración de NuGet, el nombre de la fuente es *`domain_name/repo_name.`*

> **Note**
>
> Cuando se solicita un paquete, el cliente NuGet almacena en caché qué versiones de ese paquete existen. Debido a este comportamiento, es posible que se produzca un error en la instalación de un paquete que se solicitó anteriormente antes de que la versión deseada estuviera disponible. Para evitar este error e instalar correctamente un paquete existente, puede borrar la caché de NuGet antes de la instalación con `nuget locals all --clear` o `dotnet nuget locals all --clear` bien evitar el uso de la caché durante

los comandos `install` y `restore` proporcionando la opción `-NoCache` para `nuget` o la opción `--no-cache` para `dotnet`.

dotnet

```
dotnet add package packageName --source packageSourceName
```

nuget

```
nuget install packageName -Source packageSourceName
```

Para instalar una versión específica de un paquete

dotnet

```
dotnet add package packageName --version 1.0.0 --source packageSourceName
```

nuget

```
nuget install packageName -Version 1.0.0 -Source packageSourceName
```

Consulte Administrar paquetes mediante la CLI nuget.exe o Instalar y administrar paquetes mediante la CLI dotnet en la documentación de Microsoft para obtener más información.

Consumir paquetes NuGet de NuGet.org

Puede consumir paquetes NuGet de NuGet.org a través de un repositorio de CodeArtifact configurando el repositorio con una conexión externa a NuGet.org. Los paquetes consumidos en Nuget.org se ingieren y almacenan en su repositorio de CodeArtifact. Para obtener más información acerca de la adición de conexiones externas, consulte Conectar un repositorio CodeArtifact a un repositorio público.

Publicar paquetes NuGet en CodeArtifact

Una vez que haya configurado NuGet con CodeArtifact, puede usar `nuget` o `dotnet` para publicar versiones de paquetes en los repositorios de CodeArtifact.

Para enviar una versión de paquete a un repositorio de CodeArtifact, ejecute el siguiente comando con la ruta completa a su archivo `.nupkg` y el nombre de la fuente de su repositorio de CodeArtifact en su archivo de configuración de NuGet. Si utilizó el comando `login` para configurar la configuración de NuGet, el nombre de la fuente es `domain_name/repo_name`.

> **Note**
>
> Puede crear un paquete NuGet si no tiene uno para publicar. Para obtener más información, consulte el flujo de trabajo de creación de paquetes en la documentación de Microsoft.

dotnet

```
dotnet nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg --source packageSourceName
```

nuget

```
nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg -Source packageSourceName
```

Referencia del proveedor de credenciales NuGet de CodeArtifact

El proveedor de credenciales NuGet de CodeArtifact facilita la configuración y la autenticación de NuGet con sus repositorios de CodeArtifact.

Comandos del proveedor de credenciales NuGet de CodeArtifact

Esta sección incluye la lista de comandos del proveedor de credenciales NuGet de CodeArtifact. Estos comandos deben tener el prefijo `dotnet codeartifact-creds` como en el siguiente ejemplo.

```
dotnet codeartifact-creds command
```

- `configure set profile profile`: configura el proveedor de credenciales para que utilice el perfil AWS proporcionado.
- `configure unset profile`: elimina el perfil configurado si está establecido.
- `install`: copia el proveedor de credenciales en la carpeta `plugins`.

- `install --profile profile`: copia el proveedor de credenciales en la carpeta `plugins` y lo configura para que utilice el perfil AWSproporcionado.
- `uninstall`: desinstala el proveedor de credenciales. Esto no elimina los cambios en el archivo de configuración.
- `uninstall --delete-configuration`: desinstala el proveedor de credenciales y elimina todos los cambios en el archivo de configuración.

Registros del proveedor de credenciales NuGet de CodeArtifact

Para habilitar el registro para el proveedor de credenciales NuGet de CodeArtifact, debe configurar el archivo de registro en su entorno. Los registros del proveedor de credenciales contienen información de depuración útil, como:

- El perfil AWS utilizado para realizar las conexiones
- Cualquier error de autenticación
- Si el punto de conexión proporcionado no es una URL de CodeArtifact

Configure el archivo de registro del proveedor de credenciales NuGet de CodeArtifact

```
export AWS_CODEARTIFACT_NUGET_LOGFILE=/path/to/file
```

Una vez configurado el archivo de registro, cualquier comando `codeartifact-creds` añadirá su salida de registro al contenido de ese archivo.

Versiones del proveedor de credenciales NuGet de CodeArtifact

La siguiente tabla contiene información del historial de versiones y enlaces de descarga del proveedor de credenciales NuGet de CodeArtifact.

Versión	Cambios	Fecha de publicación	Enlace de descarga (S3)
1.0.1 (última versión)	Se agregó soporte para los perfiles net5, net6 y SSO	05/03/2022	Descargar v1.0.1

Versión	Cambios	Fecha de publicación	Enlace de descarga (S3)
1.0.0	Versión inicial del proveedor de credenciales NuGet de CodeArtifact	20/11/2020	Descargar v1.0.0

Normalización del nombre, la versión y el nombre del activo del paquete NuGet

CodeArtifact normaliza los nombres de paquetes y activos y las versiones de los paquetes antes de almacenarlos, lo que significa que los nombres o versiones en CodeArtifact pueden ser diferentes a los proporcionados cuando se publicó el paquete o activo.

Normalización de nombres de paquetes: CodeArtifact normaliza los nombres de los paquetes NuGet al convertir todas las letras a minúsculas.

Normalización de la versión del paquete: CodeArtifact normaliza las versiones del paquete NuGet utilizando el mismo patrón que NuGet. La siguiente información proviene de los números de versión normalizados de la documentación de NuGet.

- Los ceros iniciales se eliminan de los números de versión:
  - `1.00` se trata como `1.0`
  - `1.01.1` se trata como `1.1.1`
  - `1.00.0.1` se trata como `1.0.0.1`
- Se omitirá un cero en la cuarta parte del número de versión:
  - `1.0.0.0` se trata como `1.0.0`
  - `1.0.01.0` se trata como `1.0.1`
- Se eliminan los metadatos de la compilación de SemVer 2.0.0:
  - `1.0.7+r3456` se trata como `1.0.7`

Normalización del nombre del activo del paquete: CodeArtifact construye el nombre del activo del paquete NuGet a partir del nombre y la versión del paquete normalizados.

El nombre del paquete y el nombre de la versión no normalizados se pueden usar con las solicitudes de API y CLI porque CodeArtifact normaliza las entradas de nombre y versión del paquete para esas solicitudes. Por ejemplo, las entradas de `--package Newtonsoft.JSON` y `--version 12.0.03.0` se normalizarían y devolverían un paquete con un nombre de `newtonsoft.json` y una versión normalizados de `12.0.3`.

Debe usar el nombre del activo del paquete normalizado en las solicitudes de API y CLI, ya que CodeArtifact no realiza la normalización de la entrada `--asset`.

Debe usar nombres y versiones normalizados en los ARN.

Para buscar el nombre normalizado de un paquete, utilice el comando `aws codeartifact list-packages`. Para obtener más información, consulte Mostrar nombres de paquetes.

Para buscar el nombre no normalizado de un paquete, utilice el comando `aws codeartifact describe-package-version`. El nombre no normalizado del paquete se devuelve en el campo `displayName`. Para obtener más información, consulte Ver y actualizar los detalles y las dependencias de la versión del paquete.

Compatibilidad con NuGet

Esta guía contiene información sobre la compatibilidad de CodeArtifact con diferentes herramientas y versiones de NuGet.

Temas

- Compatibilidad general con NuGet
- Compatibilidad con línea de comando de NuGet

Compatibilidad general con NuGet

AWS CodeArtifact es compatible con NuGet 4.8 y versiones posteriores.

AWS CodeArtifact solo es compatible con la versión 3 del protocolo HTTP NuGet. Esto significa que algunos comandos CLI que se basan en la V2 del protocolo no son compatibles. Consulte la sección Soporte de comandos nuget.exe para obtener más información.

AWS CodeArtifact no es compatible con PowerShellGet 2.x.

Compatibilidad con línea de comando de NuGet

AWS CodeArtifact es compatible con las herramientas CLI NuGet (`nuget.exe`) y .NET Core (`dotnet`).

Soporte de comandos nuget.exe

Como CodeArtifact solo es compatible con la versión 3 del protocolo HTTP de NuGet, los siguientes comandos no funcionarán cuando se usen contra los recursos de CodeArtifact:

- `list`: El comando `nuget list` muestra una lista de paquetes de una fuente determinada. Para obtener una lista de paquetes en un repositorio de CodeArtifact, puede usar el comando Mostrar nombres de paquetes de la CLI AWS.

Uso CodeArtifact con Swift

En estos temas se describe cómo utilizar el Swift Package Manager CodeArtifact para instalar y publicar paquetes Swift.

> **Note**
>
> CodeArtifact es compatible con Swift 5.8 y versiones posteriores y con Xcode 14.3 y versiones posteriores.
> CodeArtifact recomienda Swift 5.9 y versiones posteriores y Xcode 15 y versiones posteriores.

Temas

Configure el Swift Package Manager con CodeArtifact

Para usar el Swift Package Manager para publicar o consumir paquetes AWS CodeArtifact, primero tendrás que configurar las credenciales para acceder a tu CodeArtifact repositorio. El método recomendado para configurar la CLI de Swift Package Manager con sus CodeArtifact credenciales y el punto final del repositorio es mediante el `aws codeartifact login` comando. También puede configurar el Swift Package Manager manualmente.

Configurar Swift con el comando login

Utilice el `aws codeartifact login` comando para configurar el Swift Package Manager con CodeArtifact.

> **Note**
>
> Para usar el comando de inicio de sesión, se requiere Swift 5.8 o posterior y se recomienda Swift 5.9 o posterior.

El comando `aws codeartifact login` hará lo siguiente:

1. Obtenga un token de autenticación CodeArtifact y guárdelo en su entorno. La forma en que se almacenan las credenciales depende del sistema operativo del entorno:

   a. macOS: se crea una entrada en la aplicación MacOS Keychain.

   b. Linux y Windows: se crea una entrada en el archivo `~/.netrc`.

   En todos los sistemas operativos, si existe una entrada de credenciales, este comando la reemplaza por un nuevo token.

2. Busca la URL del punto final de tu CodeArtifact repositorio y agrégala a tu archivo de configuración de Swift. El comando añade la URL del punto de conexión del repositorio al archivo de configuración a nivel de proyecto ubicado en `/path/to/project/.swiftpm/configuration/registries.json`.

> **Note**
>
> El comando `aws codeartifact login` llama a comandos `swift package-registry` que deben ejecutarse desde el directorio que contiene el archivo `Package.swift`. Por ello, debe ejecutarse el comando `aws codeartifact login` desde el proyecto Swift.

Para configurar Swift con el comando login

1. Navegue hasta el directorio del proyecto Swift que contiene el archivo `Package.swift` de su proyecto.
2. Ejecute el siguiente comando de la `aws codeartifact login`.

   Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

   ```
   aws codeartifact login --tool swift --domain my_domain \
   --domain-owner 111122223333 --repository my_repo \
   [--namespace my_namespace]
   ```

La `--namespace` opción configura la aplicación para que solo consuma paquetes de tu CodeArtifact repositorio si están en el espacio de nombres designado. CodeArtifact Los espacios de nombres son

sinónimos de ámbitos y se utilizan para organizar el código en grupos lógicos y evitar las colisiones de nombres que pueden producirse cuando la base de código incluye varias bibliotecas.

El período de autorización predeterminado después de una llamada `login` es de 12 horas y `login` debe invocarse para actualizar periódicamente el token. Para obtener más información sobre el token de autorización creado con el comando `login`, consulte Tokens creados con el comando `login`.

Configurar Swift sin el comando login

Si bien se recomienda configurar Swift con el comando `aws codeartifact login`, también puede configurar Swift Package Manager sin el comando login actualizando manualmente la configuración de Swift Package Manager.

En el siguiente procedimiento, utilizará el AWS CLI para hacer lo siguiente:

1. Obtenga un token de autenticación CodeArtifact y guárdelo en su entorno. La forma en que se almacenan las credenciales depende del sistema operativo del entorno:
   a. macOS: se crea una entrada en la aplicación MacOS Keychain.
   b. Linux y Windows: se crea una entrada en el archivo `~/.netrc`.
2. Obtenga la URL del punto final de su CodeArtifact repositorio.
3. En el archivo de configuración `~/.swiftpm/configuration/registries.json`, añada una entrada con la URL del punto de conexión del repositorio y el tipo de autenticación.

Para configurar Swift sin el comando login

1. En una línea de comandos, usa el siguiente comando para obtener un token de CodeArtifact autorización y almacenarlo en una variable de entorno.
   - Sustituye *`my_domain`* por tu CodeArtifact nombre de dominio.
   - Sustituya *`111122223333`* por el ID de AWS cuenta del propietario del dominio. Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

macOS and Linux

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

Windows

- Windows (mediante el intérprete de comandos predeterminado):

```
for /f %i in ('aws codeartifact get-authorization-token --domain my_domain --
domain-owner 111122223333 --query authorizationToken --output text') do set
 CODEARTIFACT_AUTH_TOKEN=%i
```

- Windows: PowerShell

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

2. Obtenga el punto final de su CodeArtifact repositorio ejecutando el siguiente comando. El punto de conexión de su repositorio se utiliza para dirigir el Swift Package Manager a su repositorio para consumir o publicar paquetes.

   - Sustituye *my_domain* por tu nombre de CodeArtifact dominio.
   - Sustituya *111122223333* por el ID de AWS cuenta del propietario del dominio. Si va a acceder a un repositorio de un dominio de su propiedad, no tiene que incluir `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.
   - Sustituye *my_repo por el nombre de tu repositorio*. CodeArtifact

   macOS and Linux

```
export CODEARTIFACT_REPO=`aws codeartifact get-repository-endpoint --
domain my_domain --domain-owner 111122223333 --repository my_repo --format swift
 --query repositoryEndpoint --output text`
```

Windows

- Windows (mediante el intérprete de comandos predeterminado):

```
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
```

- PowerShellWindows:

```
$env:CODEARTIFACT_REPO = aws codeartifact get-repository-endpoint --
domain my_domain --domain-owner 111122223333 --repository my_repo --format
 swift --query repositoryEndpoint --output text
```

La siguiente URL es un punto de conexión de repositorio de ejemplo.

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

> **Important**
>
> Debe añadir `login` al final del punto de conexión de la URL del repositorio cuando se utilice para configurar el Swift Package Manager. Esto se hace automáticamente en los comandos de este procedimiento.

3. Con estos dos valores almacenados en variables de entorno, páselos a Swift usando el comando `swift package-registry login` de la siguiente manera:

   macOS and Linux

   ```
   swift package-registry login ${CODEARTIFACT_REPO}login --token
    ${CODEARTIFACT_AUTH_TOKEN}
   ```

   Windows

   - Windows (mediante el intérprete de comandos predeterminado):

```
swift package-registry login %CODEARTIFACT_REPO%login --token
 %CODEARTIFACT_AUTH_TOKEN%
```

- Ventanas PowerShell:

```
swift package-registry login $Env:CODEARTIFACT_REPO+"login" --token
 $Env:CODEARTIFACT_AUTH_TOKEN
```

4. A continuación, actualice el registro de paquetes utilizado por la aplicación para que cualquier dependencia se extraiga del CodeArtifact repositorio. Este comando debe ejecutarse en el directorio del proyecto en el que intenta resolver la dependencia del paquete:

   macOS and Linux

   ```
   $ swift package-registry set ${CODEARTIFACT_REPO} [--scope my_scope]
   ```

   Windows

   - Windows (mediante el intérprete de comandos predeterminado):

     ```
     $ swift package-registry set %CODEARTIFACT_REPO% [--scope my_scope]
     ```

   - Windows PowerShell:

     ```
     $ swift package-registry set $Env:CODEARTIFACT_REPO [--scope my_scope]
     ```

   La `--scope` opción configura la aplicación para que solo consuma paquetes de tu CodeArtifact repositorio si están dentro del ámbito designado. Los ámbitos son sinónimos de CodeArtifact espacios de nombres y se utilizan para organizar el código en grupos lógicos y evitar las colisiones de nombres que pueden producirse cuando la base de código incluye varias bibliotecas.

5. Para confirmar que la configuración se ha configurado correctamente, consulte el contenido del archivo `.swiftpm/configuration/registries.json` a nivel de proyecto ejecutando el siguiente comando en el directorio del proyecto:

   ```
   $ 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
}
```

Ahora que has configurado el Swift Package Manager con tu CodeArtifact repositorio, puedes usarlo para publicar y consumir paquetes Swift desde y hacia él. Para obtener más información, consulte Consumir y publicar paquetes de Swift.

Consumir y publicar paquetes de Swift

Consumir paquetes Swift desde CodeArtifact

Utilice el siguiente procedimiento para consumir paquetes Swift de un AWS CodeArtifact repositorio.

Para consumir paquetes Swift de un CodeArtifact repositorio

1. Si no lo has hecho, sigue los pasos que se indican Configure el Swift Package Manager con CodeArtifact a continuación para configurar el Swift Package Manager para que utilice tu CodeArtifact repositorio con las credenciales adecuadas.

   > **Note**
   >
   > El token de autorización generado es válido durante 12 horas. Deberá crear uno nuevo si han pasado 12 horas desde que se creó el token.

2. Edite el archivo `Package.swift` en la carpeta del proyecto de su aplicación para actualizar las dependencias del paquete que utilizará su proyecto.

   a. Si el archivo `Package.swift` no contiene una sección `dependencies`, añada una.

   b. En la sección `dependencies` del archivo `Package.swift`, añada el paquete que desee utilizar añadiendo su identificador de paquete. El identificador del paquete consiste en el

ámbito y el nombre del paquete separados por un punto. Consulte el fragmento de código que sigue a un paso posterior para ver un ejemplo.

> **Tip**
>
> Para encontrar el identificador del paquete, puedes usar la CodeArtifact consola. Busque la versión específica del paquete que desea utilizar y consulte las instrucciones de acceso directo de instalación en la página de la versión del paquete.

c. Si el archivo `Package.swift` no contiene una sección `targets`, añada una.

d. En la sección `targets`, añada los destinos que necesitarán usar la dependencia.

El siguiente fragmento es un ejemplo de fragmento que muestra las secciones `dependencies` y `targets` configuradas en un archivo `Package.swift`:

```
...
    ],
    dependencies: [
        .package(id: "my_scope.package_name", from: "1.0.0")
    ],
    targets: [
      .target(
         name: "MyApp",
         dependencies: ["package_name"]
      ),...
    ],
...
```

3. Ahora que todo está configurado, usa el siguiente comando para descargar las dependencias del paquete. CodeArtifact

```
swift package resolve
```

Consumir paquetes Swift desde CodeArtifact Xcode

Utilice el siguiente procedimiento para consumir paquetes Swift de un CodeArtifact repositorio en Xcode.

Para consumir paquetes Swift de un CodeArtifact repositorio en Xcode

1. Si no lo has hecho, sigue los pasos que se indican Configure el Swift Package Manager con CodeArtifact a continuación para configurar el Swift Package Manager para que utilice tu CodeArtifact repositorio con las credenciales adecuadas.

   > **Note**
   >
   > El token de autorización generado es válido durante 12 horas. Deberá crear uno nuevo si han pasado 12 horas desde que se creó el token.

2. Agregue los paquetes como una dependencia en su proyecto en Xcode.

   a. Seleccione Archivo > Añadir paquetes.

   b. Busque su paquete usando la barra de búsqueda. Su búsqueda debe estar en la forma `package_scope.package_name`.

   c. Una vez encontrado, elija el paquete y elija Añadir paquete.

   d. Una vez que se verifique el paquete, elija los productos del paquete que desea agregar como dependencia y elija Agregar paquete.

Si tienes problemas al usar tu CodeArtifact repositorio con Xcode, consulta Solución de problemas de Swift los problemas más comunes y las posibles soluciones.

Publicar paquetes de Swift en CodeArtifact

CodeArtifact recomienda Swift 5.9 o posterior y usar el `swift package-registry publish` comando para publicar paquetes de Swift. Si utiliza una versión anterior, debe utilizar un comando Curl para publicar los paquetes de Swift en ella. CodeArtifact

Publicar CodeArtifact paquetes con el comando **`swift package-registry publish`**

Utilice el siguiente procedimiento con Swift 5.9 o posterior para publicar paquetes de Swift en un CodeArtifact repositorio con Swift Package Manager.

1. Si no lo has hecho, sigue los pasos que se indican Configure el Swift Package Manager con CodeArtifact a continuación para configurar el Swift Package Manager para que utilice tu CodeArtifact repositorio con las credenciales adecuadas.

> **Note**
>
> El token de autorización generado es válido durante 12 horas. Deberá crear uno nuevo si han pasado 12 horas desde su creación.

2. Navegue hasta el directorio del proyecto Swift que contiene el archivo `Package.swift` de su paquete.
3. Ejecute el comando siguiente, para publicar el paquete `swift package-registry publish`. El comando crea un archivo fuente del paquete y lo publica en su CodeArtifact repositorio.

```
swift package-registry publish packageScope.packageName packageVersion
```

Por ejemplo:

```
swift package-registry publish myScope.myPackage 1.0.0
```

4. Puede confirmar que el paquete se publicó y existe en el repositorio comprobando en la consola o usando el comando `aws codeartifact list-packages` de la siguiente manera:

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

Puede enumerar la versión única del paquete usando el comando `aws codeartifact list-package-versions` de la siguiente manera:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Publicar CodeArtifact paquetes con Curl

Aunque se recomienda usar el `swift package-registry publish` comando que viene con Swift 5.9 o una versión posterior, también puedes usar Curl para publicar paquetes de Swift en él. CodeArtifact

Utilice el siguiente procedimiento para publicar paquetes de Swift en un AWS CodeArtifact repositorio con Curl.

1. Si no lo ha hecho, cree y actualice las variables de entorno `CODEARTIFACT_AUTH_TOKEN` y `CODEARTIFACT_REPO` siguiendo los pasos que se indican en Configure el Swift Package Manager con CodeArtifact.

   > **Note**
   >
   > El token de autorización es válido durante 12 horas. Deberá actualizar la variable de entorno `CODEARTIFACT_AUTH_TOKEN` con nuevas credenciales si han pasado 12 horas desde su creación.

2. En primer lugar, si no tiene un paquete Swift creado, puede hacerlo ejecutando los siguientes comandos:

   ```
   mkdir testDir && cd testDir
   swift package init
   git init .
   swift package archive-source
   ```

3. Usa el siguiente comando Curl para publicar tu paquete Swift en: 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. Puede confirmar que el paquete se publicó y existe en el repositorio comprobando en la consola o usando el comando `aws codeartifact list-packages` de la siguiente manera:

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

Puede enumerar la versión única del paquete usando el comando `aws codeartifact list-package-versions` de la siguiente manera:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Extraer paquetes de Swift GitHub y volver a publicarlos en CodeArtifact

Utilice el siguiente procedimiento para obtener un paquete Swift de un repositorio GitHub y volver a publicarlo en él. CodeArtifact

Para obtener un paquete de Swift GitHub y volver a publicarlo en CodeArtifact

1. Si no lo has hecho, sigue los pasos que se indican Configure el Swift Package Manager con CodeArtifact a continuación para configurar el Swift Package Manager para que utilice tu CodeArtifact repositorio con las credenciales adecuadas.

   > **Note**
   >
   > El token de autorización generado es válido durante 12 horas. Deberá crear uno nuevo si han pasado 12 horas desde que se creó el token.

2. Clone el repositorio git del paquete Swift que desea recuperar y volver a publicar con el siguiente comando `git clone`. Para obtener información sobre la clonación de GitHub repositorios, consulta Cómo clonar un repositorio en los documentos. GitHub

   ```
   git clone repoURL
   ```

3. Navegue hasta el repositorio que acaba de clonar:

   ```
   cd repoName
   ```

4. Crea el paquete y publícalo en. CodeArtifact

   a. Recomendado: Si utilizas Swift 5.9 o una versión posterior, puedes usar el siguiente `swift package-registry publish` comando para crear el paquete y publicarlo en el CodeArtifact repositorio que hayas configurado.

```
swift package-registry publish packageScope.packageName versionNumber
```

Por ejemplo:

```
swift package-registry publish myScope.myPackage 1.0.0
```

b. Si utilizas una versión de Swift anterior a la 5.9, debes usar el comando `swift archive-source` para crear el paquete y luego usar un comando Curl para publicarlo.

i. Si no has configurado las variables de entorno `CODEARTIFACT_AUTH_TOKEN` y `CODEARTIFACT_REPO` o han pasado más de 12 horas desde que lo hizo, siga los pasos que se indican en Configurar Swift sin el comando login.

ii. Cree el paquete Swift mediante el siguiente comando `swift package archive-source`:

```
swift package archive-source
```

Si tiene éxito, verá `Created package_name.zip` en la línea de comandos.

iii. Usa el siguiente comando Curl para publicar el paquete Swift en: 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. Puede confirmar que el paquete se publicó y existe en el repositorio comprobando en la consola o usando el comando `aws codeartifact list-packages` de la siguiente manera:

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

Puede enumerar la versión única del paquete usando el comando `aws codeartifact list-package-versions` de la siguiente manera:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Normalización del nombre del paquete y del espacio de nombres de Swift

CodeArtifact normaliza los nombres y los espacios de nombres de los paquetes antes de almacenarlos, lo que significa que los nombres CodeArtifact pueden ser diferentes a los proporcionados cuando se publicó el paquete.

Normalización de nombres de paquetes y espacios de nombres: CodeArtifact normaliza los nombres de paquetes y espacios de nombres de Swift convirtiendo todas las letras a minúsculas.

Normalización de versiones de paquetes: CodeArtifact no normaliza las versiones de paquetes de Swift. Tenga en cuenta que CodeArtifact solo es compatible con los patrones de versión 2.0 de Semantic Versioning. Para obtener más información sobre el control de versiones semántico, consulte Semantic Versioning 2.0.0.

El nombre y el espacio de nombres del paquete no normalizados se pueden usar con las solicitudes de API y CLI porque CodeArtifact normalizan las entradas de esas solicitudes. Por ejemplo, las entradas de `--package myPackage` y `--namespace myScope` se normalizarían y devolverían un paquete con un nombre de paquete `mypackage` y un espacio de nombres de `myscope` normalizados.

Debe usar nombres normalizados en los ARN, como en las políticas de IAM.

Para buscar el nombre normalizado de un paquete, utilice el comando `aws codeartifact list-packages`. Para obtener más información, consulte Mostrar nombres de paquetes.

Solución de problemas de Swift

La siguiente información puede ayudarle a solucionar problemas comunes con Swift y. CodeArtifact

Recibo un error 401 en Xcode incluso después de configurar el Swift Package Manager

Problema: Cuando intentas añadir un paquete de tu CodeArtifact repositorio como una dependencia a tu proyecto de Swift en Xcode, recibes un error 401 no autorizado incluso después de haber seguido las instrucciones para conectar Swift a. CodeArtifact

Posibles soluciones: esto puede deberse a un problema con la aplicación MacOS Keychain, donde se almacenan CodeArtifact las credenciales. Para solucionar este problema, te recomendamos abrir la aplicación Keychain y eliminar todas las CodeArtifact entradas y volver a configurar Swift Package Manager con tu CodeArtifact repositorio siguiendo las instrucciones que se indican enConfigure el Swift Package Manager con CodeArtifact.

Xcode se bloquea en la máquina CI debido a que el llavero solicita la contraseña

Problema: cuando intentas extraer paquetes de Swift CodeArtifact como parte de una versión de Xcode en un servidor de integración continua (CI), como Actions, la autenticación mediante GitHub Actions CodeArtifact puede bloquearse y, finalmente, fallar con un mensaje de error similar al siguiente:

```
Failed to save credentials for
\'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com\'
to keychain: status -60008
```

Posibles soluciones: Esto se debe a que las credenciales no se guardan en el llavero de las máquinas de CI y Xcode solo admite las credenciales guardadas en Keychain. Para solucionar este problema, te recomendamos crear la entrada del llavero manualmente siguiendo estos pasos:

1. Prepara el llavero.

```
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. Obtenga un token de CodeArtifact autenticación y el punto final de su repositorio.

```
export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token \
                                --region us-west-2 \
                                --domain my_domain   \
                                --domain-owner 111122223333   \
                                --query authorizationToken  \
                                --output text`

export CODEARTIFACT_REPO=`aws codeartifact get-repository-endpoint  \
                            --region us-west-2   \
                            --domain my_domain   \
                            --domain-owner 111122223333   \
                            --format swift     \
                            --repository my_repo    \
                            --query repositoryEndpoint   \
                            --output text`
```

3. Crea manualmente la entrada del llavero.

```
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}"
```

Para obtener más información sobre este error y la solución, consulta https://github.com/apple/ swift-package-manager /issues/7236.

Uso de CodeArtifact con paquetes genéricos

En estos temas se muestra cómo consumir y publicar paquetes genéricos mediante AWS CodeArtifact.

Temas

Información general de los paquetes genéricos

Con el formato de paquete `generic`, puede cargar cualquier tipo de archivo para crear un paquete en un repositorio de CodeArtifact. Los paquetes genéricos no están asociados a ningún lenguaje de programación, tipo de archivo o ecosistema de administración de paquetes específico. Esto puede resultar útil para almacenar y controlar versiones de artefactos de compilación arbitrarios, como instaladores de aplicaciones, modelos de machine learning, archivos de configuración y otros.

Un paquete genérico consta de un nombre de paquete, un espacio de nombres, una versión y uno o más activos (o archivos). Los paquetes genéricos pueden existir junto con paquetes de otros formatos en un único repositorio de CodeArtifact.

Puedes usar la AWS CLI o el SDK para trabajar con paquetes genéricos. Para obtener una lista completa de comandos AWS CLI que funcionan con paquetes genéricos, consulte Comandos compatibles con paquetes genéricos.

Restricciones de paquetes genéricos

- Nunca se obtienen de los repositorios originales. Solo se pueden obtener del repositorio en el que se publicaron.
- No pueden declarar las dependencias que se devolverán desde ListPackageVersionDependencies ni se mostrarán en AWS Management Console.
- Pueden almacenar archivos README y LICENSE, pero CodeArtifact no los interpreta. La información de estos archivos no la devuelven getPackageVersionREADME ni DescribePackageVersion y no aparece en la AWS Management Console.

- Como todos los paquetes de CodeArtifact, existen límites para el tamaño de los activos y la cantidad de activos por paquete. Para obtener más información acerca de los límites y cuotas de CodeArtifact, consulte Cuotas en AWS CodeArtifact.
- Los nombres de los activos que contienen deben seguir estas reglas:
  - Los nombres de los activos pueden usar letras y números Unicode. En concreto, se permiten estas categorías de caracteres Unicode: letra minúscula (`Ll`), letra modificadora (`Lm`), otra letra (`Lo`), letra de caso del título (`Lt`), letra mayúscula (`Lu`), letra número (`Nl`) y número decimal (`Nd`).
  - Se permiten los siguientes caracteres especiales: `~!@^&()-_+[]{};,.`
  - Los activos no pueden tener un nombre `.` o `..`
  - Los espacios son el único carácter de espacio en blanco permitido. Los nombres de los activos no pueden comenzar ni terminar con un carácter de espacio ni incluir espacios consecutivos.

Comandos compatibles con paquetes genéricos

Puede usar la AWS CLI o el SDK para trabajar con paquetes genéricos. Los siguientes comandos de CodeArtifact funcionan con paquetes genéricos:

- copy-package-versions (consulte Copiar paquetes entre repositorios)
- delete-package (consulte Eliminación de un paquete (AWS CLI))
- delete-package-versions (consulte Eliminar una versión del paquete (AWS CLI))
- describe-package
- describe-package-version (consulte Ver y actualizar los detalles y las dependencias de la versión del paquete)
- dispose-package-versions (consulte Eliminación de las versiones de los paquetes)
- get-package-version-asset (consulte Descargar recursos de la versión del paquete)
- list-package-version-assets (consulte Enumerar los activos de la versión del paquete)
- list-package-versions (consulte Listar las versiones de los paquetes)
- list-packages (consulte Mostrar nombres de paquetes)
- publish-package-version (consulte Publicar un paquete genérico)
- put-package-origin-configuration (consulte) Edición de los controles de origen del paquete

> **Note**
>
> Puede usar la configuración de control de origen `publish` para permitir o bloquear la publicación de un nombre de paquete genérico en un repositorio. Sin embargo, la configuración `upstream` no se aplica a los paquetes genéricos porque no se pueden recuperar de un repositorio principal.

- update-package-versions-status (consulte Actualizar el estado de la versión del paquete)

Publicar y consumir paquetes genéricos

Para publicar una versión genérica de un paquete y sus activos relacionados, utilice el comando `publish-package-version`. Puede enumerar los activos de un paquete genérico mediante el comando `list-package-version-asset` y descargarlos mediante`get-package-version-asset`. El siguiente tema contiene instrucciones paso a paso para publicar paquetes genéricos o descargar activos de paquetes genéricos mediante estos comandos.

Publicar un paquete genérico

Un paquete genérico consta de un nombre de paquete, un espacio de nombres, una versión y uno o más activos (o archivos). En este tema se muestra cómo publicar un paquete denominado `my-package`, con el espacio de nombres `my-ns`, la versión `1.0.0` y que contenga un activo denominado `asset.tar.gz`.

Requisitos previos:

- Configurar y configurar AWS Command Line Interface con CodeArtifact (consulte Configuración con AWS CodeArtifact)
- Disponer de un dominio y un repositorio de CodeArtifact (consulte Primeros pasos con AWS CLI)

Para publicar un paquete genérico

1. Utilice el siguiente comando para generar el hash SHA256 para cada archivo que desee cargar en una versión de paquete y coloque el valor en una variable de entorno. Este valor se utiliza como comprobación de integridad para comprobar que el contenido del archivo no ha cambiado después de su envío original.

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. Llame a `publish-package-version` para cargar el activo y crear una nueva versión del paquete.

> **Note**
>
> Si su paquete contiene más de un activo, puede llamar a `publish-package-version` una vez para cargar cada uno de ellos. Incluya el argumento `--unfinished` de cada llamada a `publish-package-version`, excepto cuando suba el último activo. Si se omite `--unfinished`, se establecerá el estado de la versión del paquete en `Published` y evitará que se carguen recursos adicionales en él.
> Como alternativa, incluya `--unfinished` para cada llamada a `publish-package-version` y, a continuación, establezca el estado de la versión del paquete en `Published` con el comando `update-package-versions-status`.

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%
```

El ejemplo siguiente muestra la salida.

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

Listar los activos de los paquetes genéricos

Para enumerar los activos contenidos en un paquete genérico, utilice el comando `list-package-version-assets`. Para obtener más información, consulte Enumerar los activos de la versión del paquete.

El siguiente ejemplo muestra los activos de la versión `1.0.0` del paquete `my-package`.

Para enumerar los activos de la versión del paquete

- Llame a `list-package-version-assets` para enumerar los activos contenidos en un paquete genérico.

  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
  ```

  El ejemplo siguiente muestra la salida.

  ```
  {
      "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"
}
```

Descargar los activos de los paquetes genéricos

Para descargar los activos de un paquete genérico, utilice el comando `get-package-version-asset`. Para obtener más información, consulte Descargar recursos de la versión del paquete.

El siguiente ejemplo descarga el activo `asset.tar.gz` de la versión `1.0.0` del paquete `my-package` al directorio de trabajo actual, en un archivo también denominado `asset.tar.gz`.

Para descargar los activos de la versión del paquete

- Llame a `get-package-version-asset` para descargar los activos de un paquete genérico.

  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
  ```

  El ejemplo siguiente muestra la salida.

  ```
  {
      "assetName": "asset.tar.gz",
      "packageVersion": "1.0.0",
      "packageVersionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"
  }
  ```

Uso de CodeArtifact con CodeBuild

En estos temas se describe cómo usar los paquetes de un repositorio de CodeArtifact en un proyecto de compilación AWS CodeBuild.

Temas

Uso de paquetes npm en CodeBuild

Los siguientes pasos se han probado con los sistemas operativos que figuran en las imágenes de Docker proporcionadas por CodeBuild.

Configure los permisos necesarios para los roles de IAM

Estos pasos son necesarios cuando se utilizan paquetes npm de CodeArtifact en CodeBuild.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Seleccione Roles (Roles) en el panel de navegación. En la página Roles, edite el rol que usa su proyecto de compilación de CodeBuild. Este rol debe tener los siguientes permisos.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [ "codeartifact:GetAuthorizationToken",
                      "codeartifact:GetRepositoryEndpoint",
                      "codeartifact:ReadFromRepository"
                      ],
          "Resource": "*"
      },
      {
```

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

> **Important**
>
> Si también quiere usar CodeBuild para publicar paquetes, añada el permiso **`codeartifact:PublishPackageVersion`**.

Para obtener información, consulte Modificación de un rol en la Guía del usuario de IAM.

Regístrese y utilice npm

Para usar los paquetes npm de CodeBuild, ejecute el comando `login` de la sección `pre-build` de su proyecto `npm` para configurar la búsqueda de paquetes `buildspec.yaml` de CodeArtifact. Para obtener más información, consulte Autenticación con npm.

Una vez que se haya ejecutado `login` correctamente, puede ejecutar los comandos `npm` de la sección `build` para instalar los paquetes npm.

Linux

> **Note**
>
> Solo es necesario actualizar la AWS CLI con `pip3 install awscli --upgrade --user` si está utilizando una imagen de CodeBuild anterior. Si está utilizando las versiones de imagen más recientes, puede eliminar esa línea.

```
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
```

Uso de paquetes de Python en CodeBuild

Los siguientes pasos se han probado con los sistemas operativos que figuran en las imágenes de Docker proporcionadas por CodeBuild.

Configure los permisos necesarios para los roles de IAM

Estos pasos son necesarios cuando se utilizan paquetes Python de CodeArtifact en CodeBuild.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Seleccione Roles (Roles) en el panel de navegación. En la página Roles, edite el rol que usa su proyecto de compilación de CodeBuild. Este rol debe tener los siguientes permisos.

   ```
   {
   ```

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

> **Important**
>
> Si también quiere usar CodeBuild para publicar paquetes, añada el permiso **`codeartifact:PublishPackageVersion`**.

Para obtener información, consulte Modificación de un rol en la Guía del usuario de IAM.

Regístrese y utilice pip o twine

Para usar los paquetes Python de CodeBuild, ejecute el comando `login` de la sección `pre-build` de su archivo del proyecto `pip` para configurar la búsqueda de paquetes `buildspec.yaml` de CodeArtifact. Para obtener más información, consulte Uso de CodeArtifact con Python.

Una vez que se haya ejecutado `login` correctamente, puede ejecutar los comandos `pip` de la sección `build` para instalar o publicar paquetes Python.

Linux

> **Note**
>
> Solo es necesario actualizar la AWS CLI con `pip3 install awscli --upgrade --user` si está utilizando una imagen de CodeBuild anterior. Si está utilizando las versiones de imagen más recientes, puede eliminar esa línea.

Para instalar paquetes de Python mediante `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
```

Para publicar paquetes de Python mediante `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

Para instalar paquetes de Python mediante `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
```

Para publicar paquetes de Python mediante `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
```

Uso de paquetes Maven en CodeBuild

Configure los permisos necesarios para los roles de IAM

Estos pasos son necesarios cuando se utilizan paquetes Maven de CodeArtifact en CodeBuild.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Seleccione Roles (Roles) en el panel de navegación. En la página Roles, edite el rol que usa su proyecto de compilación de CodeBuild. Este rol debe tener los siguientes permisos.

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

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

> **Important**
>
> Si también quiere usar CodeBuild para publicar paquetes, añada los permisos **`codeartifact:PublishPackageVersion`** y **`codeartifact:PutPackageMetadata`**.

Para obtener información, consulte Modificación de un rol en la Guía del usuario de IAM.

Usar gradle o mvn

Para usar paquetes de Maven con `gradle` o `mvn`, almacene el token de autenticación CodeArtifact en una variable de entorno, como se describe en Pasar un token de autenticación a una variable de entorno. A continuación, se muestra un ejemplo.

> **Note**
>
> Solo es necesario actualizar la AWS CLI con `pip3 install awscli --upgrade --user` si está utilizando una imagen de CodeBuild anterior. Si está utilizando las versiones de imagen más recientes, puede eliminar esa línea.

```
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`
```

Para usar Gradle:

Si hizo referencia a la variable `CODEARTIFACT_AUTH_TOKEN` en su archivo `build.gradle` de Gradle como se describe en Cómo usar CodeArtifact con Gradle, puede invocar su compilación de Gradle desde la sección `buildspec.yaml` `build`.

```
build:
  commands:
    - gradle build
```

Para usar mvn:

Debe configurar sus archivos de configuración de Maven (`settings.xml` y `pom.xml`) seguir las instrucciones de Uso de CodeArtifact con mvn.

Uso de paquetes NuGet en CodeBuild

Los siguientes pasos se han probado con los sistemas operativos que figuran en las imágenes de Docker proporcionadas por CodeBuild.

Temas

- Configure los permisos necesarios para los roles de IAM
- Consumir paquetes NuGet
- Compilar paquetes NuGet
- Publicar paquetes NuGet

Configure los permisos necesarios para los roles de IAM

Estos pasos son necesarios cuando se utilizan paquetes NuGet de CodeArtifact en CodeBuild.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Seleccione Roles (Roles) en el panel de navegación. En la página Roles, edite el rol que usa su proyecto de compilación de CodeBuild. Este rol debe tener los siguientes permisos.

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

> **Important**
>
> Si también quiere usar CodeBuild para publicar paquetes, añada el permiso **`codeartifact:PublishPackageVersion`**.

Para obtener información, consulte Modificación de un rol en la Guía del usuario de IAM.

Consumir paquetes NuGet

Para consumir paquetes NuGet de CodeBuild, incluya lo siguiente en el archivo `buildspec.yaml` de su proyecto.

1. En la sección `install`, instale el proveedor de credenciales de CodeArtifact para configurar las herramientas de línea de comandos, por ejemplo, `msbuild` y `dotnet` para crear y publicar paquetes en CodeArtifact.
2. En la sección `pre-build`, agregue su repositorio CodeArtifact a su configuración de NuGet.

Vea los siguientes ejemplos de `buildspec.yaml`. Para obtener más información, consulte Uso de CodeArtifact con NuGet.

Una vez instalado el proveedor de credenciales y agregado la fuente del repositorio, puede ejecutar los comandos de la herramienta NuGet CLI desde la sección `build` para consumir los paquetes NuGet.

Linux

Para consumir paquetes NuGet mediante `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

Para consumir paquetes NuGet mediante `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
```

Compilar paquetes NuGet

Para compilar con paquetes NuGet de CodeBuild, incluya lo siguiente en el archivo `buildspec.yaml` de su proyecto.

1. En la sección `install`, instale el proveedor de credenciales de CodeArtifact para configurar las herramientas de línea de comandos, por ejemplo, `msbuild` y `dotnet` para crear y publicar paquetes en CodeArtifact.
2. En la sección `pre-build`, agregue su repositorio CodeArtifact a su configuración de NuGet.

Vea los siguientes ejemplos de `buildspec.yaml`. Para obtener más información, consulte Uso de CodeArtifact con NuGet.

Una vez instalado el proveedor de credenciales y agregado la fuente del repositorio, puede ejecutar los comandos de la herramienta NuGet CLI como `dotnet build` de la sección `build`.

Linux

Para crear paquetes NuGet usando `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
```

Para crear paquetes NuGet usando `msbuild`:

```
version: 0.2

phases:
  install:
    runtime-versions:
      dotnet: latest
    commands:
      - export PATH="$PATH:/root/.dotnet/tools"
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact $(aws codeartifact get-repository-
endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format
 nuget --query repositoryEndpoint --output text)"v3/index.json"
  build:
    commands:
      - msbuild -t:Rebuild -p:Configuration=Release
```

Windows

Para crear paquetes NuGet usando `dotnet`:

```
version: 0.2

phases:
  install:
    commands:
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact "$(aws codeartifact get-repository-
endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format
 nuget --query repositoryEndpoint --output text)v3/index.json"
  build:
    commands:
      - dotnet build
```

Para crear paquetes NuGet usando `msbuild`:

```
version: 0.2

phases:
  install:
    commands:
      - dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
      - dotnet codeartifact-creds install
  pre_build:
    commands:
      - dotnet nuget add source -n codeartifact "$(aws codeartifact get-repository-
endpoint --domain my_domain --domain-owner 111122223333 --repository my_repo --format
 nuget --query repositoryEndpoint --output text)v3/index.json"
  build:
    commands:
      - msbuild -t:Rebuild -p:Configuration=Release
```

Publicar paquetes NuGet

Para publicar paquetes NuGet de CodeBuild, incluya lo siguiente en el archivo `buildspec.yaml` de su proyecto.

1. En la sección `install`, instale el proveedor de credenciales de CodeArtifact para configurar las herramientas de línea de comandos, por ejemplo, `msbuild` y `dotnet` para crear y publicar paquetes en CodeArtifact.

2. En la sección `pre-build`, agregue su repositorio CodeArtifact a su configuración de NuGet.

Vea los siguientes ejemplos de `buildspec.yaml`. Para obtener más información, consulte Uso de CodeArtifact con NuGet.

Una vez instalado el proveedor de credenciales y agregado la fuente del repositorio, puede ejecutar los comandos de la herramienta NuGet CLI desde la sección `build` y publicar sus paquetes NuGet.

Linux

Para publicar paquetes NuGet mediante `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

Para publicar paquetes NuGet mediante `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
```

Almacenamiento en caché de dependencias

Puede habilitar el almacenamiento en caché local en CodeBuild para reducir la cantidad de dependencias que deben obtenerse de CodeArtifact para cada compilación. Para obtener más información, consulte Cómo compilar el almacenamiento en caché en AWS CodeBuild en la Guía del usuario de AWS CodeBuild. Después de habilitar una caché local personalizada, añada el directorio de caché al archivo `buildspec.yaml` del proyecto.

Por ejemplo, si está utilizando `mvn`, utilice lo siguiente.

```
cache:
  paths:
    - '/root/.m2/**/*'
```

Para otras herramientas, use las carpetas de caché que se muestran en esta tabla.

Herramienta	Directorio de caché
mvn	`/root/.m2/**/*`
gradle	`/root/.gradle/caches/**/*`
pip	`/root/.cache/pip/**/*`
npm	`/root/.npm/**/*`
nuget	`/root/.nuget/**/*`
yarn (classic)	`/root/.cache/yarn/**/*`

Supervisión de CodeArtifact

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de CodeArtifact y de sus otras soluciones de AWS. AWS ofrece las siguientes herramientas de supervisión para vigilar CodeArtifact, informar cuando algo no va bien y tomar medidas automáticamente cuando proceda:

- Puede utilizar Amazon EventBridge para automatizar los servicios de AWS y responder automáticamente a eventos del sistema, como problemas de disponibilidad de las aplicaciones o cambios en los recursos. Los eventos de los servicios de AWS se envían a EventBridge casi en tiempo real. Puede crear reglas sencillas para indicar qué eventos le resultan de interés, así como qué acciones automatizadas se van a realizar cuando un evento cumple una de las reglas. Para obtener más información, consulte la Guía del usuario de Amazon EventBridge y Formato y ejemplo de evento CodeArtifact.
- Puede usar las métricas de Amazon CloudWatch para ver el uso de CodeArtifact por operación. Las métricas de CloudWatch incluyen todas las solicitudes realizadas a CodeArtifact y las solicitudes se muestran por cuenta. Para ver estas métricas en las métricas de CloudWatch, vaya al espacio de nombres Uso/Por recurso de AWS de AWS. Para obtener más información, consulte Uso de las métricas de Amazon CloudWatch en la Guía del usuario de Amazon CloudWatch.

Temas

Supervisión de eventos de CodeArtifact

CodeArtifact está integrado con Amazon EventBridge, un servicio que automatiza y responde a los eventos, incluidos los cambios en un repositorio de CodeArtifact. Puede crear reglas para eventos y configurar qué sucede cuando un evento cumpla una de las reglas. Antes, EventBridge se llamaba CloudWatch Events.

Un evento puede desencadenar las siguientes acciones:

- Invocar una función de AWS Lambda,

- Activar una máquina de estado de AWS Step Functions,
- Notificar un tema de Amazon SNS o una cola de Amazon SQS.
- Iniciar una canalización en AWS CodePipeline.

CodeArtifact crea un evento cuando se crea, modifica o elimina una versión del paquete. Los siguientes son ejemplos de eventos CodeArtifact:

- Publicar una nueva versión del paquete (por ejemplo, ejecutando `npm publish`).
- Añadir un nuevo activo a una versión de paquete existente (por ejemplo, insertando un nuevo archivo JAR en un paquete Maven existente).
- Copiar una versión de paquete de un repositorio en otro mediante `copy-package-versions`. Para obtener más información, consulte Copiar paquetes entre repositorios.
- Eliminar versiones de paquetes mediante `delete-package-versions`. Para obtener más información, consulte Eliminar un paquete.
- Eliminar versiones de paquetes mediante `delete-package`. Se publicará un evento para cada versión del paquete eliminado. Para obtener más información, consulte Eliminar un paquete.
- Conservar la versión de un paquete en un repositorio descendente cuando se ha obtenido de un repositorio ascendente. Para obtener más información, consulte Trabajar con repositorios ascendentes en CodeArtifact.
- Ingerir una versión de paquete de un repositorio externo a un repositorio de CodeArtifact. Para obtener más información, consulte Conectar un repositorio CodeArtifact a un repositorio público.

Los eventos se envían tanto a la cuenta propietaria del dominio como a la cuenta que administra el repositorio. Por ejemplo, supongamos que esa cuenta `111111111111` es la propietaria del dominio `my_domain`. La cuenta `222222222222` crea un repositorio en `my_domain` llamado `repo2`. Cuando se publica una nueva versión del paquete en `repo2`, ambas cuentas reciben los eventos de EventBridge. La cuenta propietaria del dominio (`111111111111`) recibe los eventos de todos los repositorios del dominio. Si una sola cuenta es propietaria tanto del dominio como del repositorio que contiene, solo se entrega un evento.

En los temas siguientes se describe el formato de eventos CodeArtifact. Le muestran cómo configurar los eventos de CodeArtifact y cómo utilizar los eventos con otros servicios AWS. Para obtener más información, consulte Introducción a Amazon EventBridge en la Guía del usuario de Amazon EventBridge.

Formato y ejemplo de evento CodeArtifact

Los siguientes son campos y descripciones de eventos junto con un ejemplo de un evento CodeArtifact.

Formato de eventos CodeArtifact

Todos los eventos de CodeArtifact incluyen los siguientes campos.

Campo del evento	Descripción
versión	La versión del formato del evento de . Actualmente sólo existe una única versión, `0`.
id	Un identificador único para el evento.
tipo-detalle	El tipo de evento. Esto determina los campos del objeto `detail`. El `detail-type` que se admite actualmente es `CodeArtifact Package Version State Change`.
source	Origen del evento. Para CodeArtifact, será `aws.codeartifact` .
cuenta	El ID de la cuenta de AWS que recibe el evento.
hora	La hora exacta en la que se activó el evento.
región	La región en la que se desencadenó el evento.
resources	Una lista que contiene el ARN del paquete que se modificó. La lista contiene una entrada. Para obtener información sobre el formato ARN del paquete, consulte Otorgar acceso de escritura a los paquetes.
domainName	El dominio que contiene el repositorio que contiene el paquete.

Campo del evento	Descripción
domainOwner	El ID de la cuenta de AWS del propietario del dominio.
repositoryName	El repositorio que contiene el paquete.
repositoryAdministrator	El ID de cuenta de AWS del administrador del repositorio.
packageFormat	El formato del paquete que desencadenó el evento.
packageNamespace	El espacio de nombres del paquete que desencadenó el evento.
packageName	El nombre del paquete que desencadenó el evento.
packageVersion	La versión del paquete que desencadenó el evento.
packageVersionState	El estado de la versión del paquete cuando se desencadenó el evento. Los valores posibles son `Unfinished` , `Published` , `Unlisted`, `Archived` y `Disposed`.
packageVersionRevision	Un valor que identifica de forma exclusiva el estado de los activos y los metadatos de la versión del paquete cuando se activó el evento. Si se modifica la versión del paquete (por ejemplo, añadiendo otro archivo JAR a un paquete de Maven), `packageVersionRevision` cambia.
changes.assetsAdded	La cantidad de activos agregados a un paquete que desencadenaron un evento. Algunos ejemplos de activos son un archivo JAR de Maven o una rueda de Python.

Campo del evento	Descripción
changes.assetsRemoved	El número de activos retirados de un paquete que desencadenaron un evento.
changes.assetsUpdated	El número de activos modificados en el paquete que desencadenó el evento.
changes.metadataUpdated	Un valor booleano que se establece en `true` si el evento incluye metadatos modificados a nivel de paquete. Por ejemplo, un evento puede modificar un archivo `pom.xml` Maven.
changes.statusChanged	Un valor booleano que se establece en `true` si se modifica el evento `packageVersionStatus` (por ejemplo, si `packageVersionStatus` cambia de `Unfinished` a `Published`).
operationType	Describe el tipo de cambio de versión del paquete de alto nivel. Los valores posibles son `Created`, `Updated` y `Deleted`.
sequenceNumber	Un entero que especifica el número de evento de un paquete. Cada evento de un paquete incrementa el `sequenceNumber` para que los eventos se puedan organizar secuencialmente. Un evento puede incrementar `sequenceNumber` en cualquier número entero. **Note** Es posible que los eventos de EventBridge se reciban fuera de orden. Se puede utilizar `sequenceNumber` para determinar su orden real.

Campo del evento	Descripción
eventDeduplicationId	Un identificador que se utiliza para diferenci ar los eventos de EventBridge duplicados. En raras ocasiones, EventBridge puede activar la misma regla más de una vez para un solo evento o una hora programada. O bien, puede invocar el mismo objetivo más de una vez para una regla activada determinada.

Ejemplo de evento CodeArtifact

El siguiente es el ejemplo de un evento CodeArtifact que puede desencadenarse cuando se publique un paquete npm.

```
{
      "version":"0",
      "id":"73f03fec-a137-971e-6ac6-07c8ffffffff",
      "detail-type":"CodeArtifact Package Version State Change",
      "source":"aws.codeartifact",
      "account":"123456789012",
      "time":"2019-11-21T23:19:54Z",
      "region":"us-west-2",
      "resources":["arn:aws:codeartifact:us-west-2:111122223333:package/my_domain/
myrepo/npm//mypackage"],
      "detail":{
        "domainName":"my_domain",
        "domainOwner":"111122223333",
        "repositoryName":"myrepo",
        "repositoryAdministrator":"123456789012",
        "packageFormat":"npm",
        "packageNamespace":null,
        "packageName":"mypackage",
        "packageVersion":"1.0.0",
        "packageVersionState":"Published",
        "packageVersionRevision":"0E5DE26A4CD79FDF3EBC4924FFFFFFFF",
        "changes":{
          "assetsAdded":1,
          "assetsRemoved":0,
          "metadataUpdated":true,
```

```
          "assetsUpdated":0,
          "statusChanged":true
        },
        "operationType":"Created",
        "sequenceNumber":1,
        "eventDeduplicationId":"2mEO0A2Ke07rWUTBXk3CAiQhdTXF4N94LNaT/ffffff="
      }
    }
```

Use un evento para iniciar una ejecución de CodePipeline

En este ejemplo se muestra cómo configurar una regla de Amazon EventBridge para que una ejecución AWS CodePipeline comience cuando se publique, modifique o elimine una versión de paquete de un repositorio de CodeArtifact.

Temas

- Configuración de permisos de EventBridge
- Crear la regla de EventBridge
- Crear el objetivo de la regla de EventBridge

Configuración de permisos de EventBridge

Debe añadir permisos para que EventBridge utilice CodePipeline para invocar la regla que cree. Para añadir estos permisos mediante AWS Command Line Interface (AWS CLI), siga el paso 1 de la sección Crear una regla de eventos de CloudWatch para una fuente de CodeCommit (CLI) de la Guía del usuario de AWS CodePipeline.

Crear la regla de EventBridge

Para crear la regla, utilice el comando `put-rule` con los parámetros `--name` y `--event-pattern`. El patrón de eventos especifica los valores que coinciden con el contenido de cada evento. El objetivo se activa si el patrón coincide con el evento. Por ejemplo, el siguiente patrón coincide con los eventos CodeArtifact del repositorio `myrepo` del dominio `my_domain`.

```
aws events put-rule --name MyCodeArtifactRepoRule --event-pattern \
    '{"source":["aws.codeartifact"],"detail-type":["CodeArtifact Package Version State
 Change"],
```

```
    "detail":{"domainName":["my_domain"],"domainOwner":
["111122223333"],"repositoryName":["myrepo"]}}'
```

Crear el objetivo de la regla de EventBridge

El siguiente comando agrega un destino a la regla para que, cuando un evento coincida con la regla, se desencadene una ejecución de CodePipeline. Para el parámetro `RoleArn`, especifique el nombre de recurso de Amazon (ARN) del rol creado anteriormente en este tema.

```
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'
```

Utilizar un evento para ejecutar una función de Lambda

Este ejemplo muestra cómo configurar una regla de EventBridge que inicia una función AWS Lambda cuando se publica, modifica o elimina una versión de paquete en un repositorio de CodeArtifact.

Para obtener más información, consulte el Tutorial: Programar funciones AWS Lambda con EventBridge en la Guía del usuario de Amazon EventBridge.

Temas

- Crear la regla de EventBridge
- Crear el objetivo de la regla de EventBridge
- Configuración de permisos de EventBridge

Crear la regla de EventBridge

Para crear una regla que inicie una función de Lambda, utilice el comando `put-rule` con las opciones `--name` y `--event-pattern`. El siguiente patrón especifica los paquetes npm en el ámbito `@types` de cualquier repositorio del dominio `my_domain`.

```
aws events put-rule --name "MyCodeArtifactRepoRule" --event-pattern \
  '{"source":["aws.codeartifact"],"detail-type":["CodeArtifact Package Version State
 Change"],
```

```
  "detail":{"domainName":["my_domain"],"domainOwner":
["111122223333"],"packageNamespace":["types"],"packageFormat":["npm"]}}'
```

Crear el objetivo de la regla de EventBridge

El siguiente comando agrega un destino a la regla que ejecuta la función de Lambda cuando un evento coincide con la regla. Para el parámetro `arn`, especifique el nombre de recurso de Amazon (ARN) de la función de Lambda.

```
aws events put-targets --rule MyCodeArtifactRepoRule --targets \
  Id=1,Arn=arn:aws:lambda:us-west-2:111122223333:function:MyLambdaFunction
```

Configuración de permisos de EventBridge

Utilice el comando `add-permission` para conceder permisos para que la regla invoque una función de Lambda. Para el parámetro `--source-arn`, especifique el ARN de la regla que creó anteriormente en este ejemplo.

```
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
```

Seguridad en CodeArtifact

La seguridad en la nube de AWS es la mayor prioridad. Como cliente de AWS, se beneficiará de una arquitectura de red y de centros de datos diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta los servicios de AWS en la nube de AWS. AWS también proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los AWSProgramas de conformidad de . Para obtener información sobre los programas de conformidad que se aplican a CodeArtifact, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad se determina según el servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza CodeArtifact. En los siguientes temas, se le mostrará cómo configurar CodeArtifact para satisfacer sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros servicios de AWS que le ayudarán a monitorear y a proteger los recursos de CodeArtifact.

Temas

Protección de los datos en AWS CodeArtifact

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en AWS CodeArtifact. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta la totalidad de Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog Modelo de responsabilidad compartida y GDPR de AWS en el Blog de seguridad de AWS.

Con fines de protección de datos, recomendamos proteger las credenciales de la cuenta de Cuenta de AWS y configurar cuentas de usuario individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se conceden a cada usuario los permisos necesarios para cumplir con sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos de AWS. reSe recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los Servicios de AWS.
- Utilice servicios de seguridad gestionados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados FIPS 140-2 al acceder a AWS a través de una interfaz de la línea de comandos o una API, utilice un punto de conexión de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye los casos en los que debe trabajar con CodeArtifact u otros Servicios de AWS a través de la consola, la API, AWS CLI o los SDK de AWS. Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos

encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Cifrado de datos

El cifrado es una parte importante de la seguridad de CodeArtifact. Algunos cifrados, por ejemplo, para el cifrado de datos en tránsito se proporcionan de forma predeterminada y no es necesario que haga nada. Otros, por ejemplo, para el cifrado de datos en reposo, se pueden configurar cuando cree su proyecto o compilación.

- Cifrado de datos en reposo: todos los activos almacenados en CodeArtifact se cifran mediante AWS KMS keys (claves KMS). Esto incluye todos los activos de todos los paquetes de todos los repositorios. Se utiliza una clave KMS para cada dominio para cifrar todos sus activos. De forma predeterminada, se usa una clave de KMS AWS administrada, por lo que no es necesario crear una clave de KMS. Si lo desea, puede usar una clave KMS administrada por el cliente que cree y configure. Para obtener más información, consulte Creación de claves y conceptos del Servicio de administración de claves AWS en la Guía del usuario de AWS Key Management Service. Puede especificar una clave KMS administrada por el cliente al crear un dominio. Para obtener más información, consulte Trabajando con dominios en CodeArtifact.
- Cifrado de datos en tránsito: todas las comunicaciones entre los clientes y CodeArtifact y entre CodeArtifact y sus dependencias posteriores están protegidas mediante el cifrado TLS.

Privacidad de tráfico

Puede mejorar la seguridad de sus dominios de CodeArtifact y los activos que contienen configurando CodeArtifact para que use un punto de conexión de la nube privada virtual (VPC) de interfaz. Para ello, no necesita una puerta de enlace de Internet, ni un dispositivo NAT, ni una puerta de enlace privada virtual. Para obtener más información, consulte Trabajar con puntos de conexión de VPC de Amazon. Para obtener más información sobre AWS PrivateLink y los puntos de conexión de VPC, consulte AWS PrivateLink y Acceso a servicios de AWS a través de PrivateLink.

Supervisión de CodeArtifact

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de AWS CodeArtifact y sus soluciones de AWS. Debe recopilar datos de monitorización de todas las partes de su solución de AWS para que pueda más fácilmente depurar un error

multipunto si se produce. AWS proporciona las siguientes herramientas para monitorizar sus recursos de CodeArtifact y responder a posibles incidentes:

Temas

Registro de llamadas a la API de CodeArtifact con AWS CloudTrail

CodeArtifact se integra a AWS CloudTrail, un servicio que brinda un registro de las acciones que realiza un usuario, un rol o un servicio de AWS en CodeArtifact. CloudTrail captura todas las llamadas a la API de CodeArtifact como eventos, incluidas las llamadas procedentes de los clientes de los gestores de paquetes.

Si crea un registro de seguimiento, puede habilitar la entrega continua de eventos de CloudTrail a un bucket de Amazon Simple Storage Service (Amazon S3), incluidos los eventos para CodeArtifact. Si no configura un registro de seguimiento, puede ver los eventos más recientes de la consola de CloudTrail en el Historial de eventos. Mediante la información que recopila CloudTrail, puede determinar la solicitud que se hizo a CodeArtifact, la dirección IP desde la que se hizo la solicitud, quién la hizo y cuándo e información adicional.

Para obtener más información acerca de CloudTrail, consulte la Guía del usuario de AWS CloudTrail.

Información de CodeArtifact en CloudTrail

CloudTrail se habilita en su cuenta de AWS cuando la crea. Cuando se produce una actividad en CodeArtifact, esa actividad se registra en un evento de CloudTrail junto con otros eventos de servicio de AWS en Historial de eventos. Puede ver, buscar y descargar los últimos eventos de la cuenta de AWS. Para obtener más información, consulte Ver eventos con el historial de eventos de CloudTrail.

Para un registro continuo de los eventos de la cuenta de AWS, incluidos los eventos de CodeArtifact, cree un registro de seguimiento. Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones de AWS. El registro de seguimiento registra los eventos de todas las regiones de la partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. Puede configurar otros servicios de AWS para analizar y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para obtener más información, consulte los siguientes temas:

- Creación de un registro de seguimiento de su cuenta de AWS

- Servicios e integraciones compatibles con CloudTrail
- Configuración de notificaciones de Amazon SNS para CloudTrail

Cuando el registro de CloudTrail está habilitado en su cuenta de AWS, las llamadas a la API realizadas a las acciones de CodeArtifact se escriben en los archivos de registro de CloudTrail junto con otros registros de servicio de AWS. CloudTrail determina cuándo debe crearse un nuevo archivo y escribir en él en función del periodo de tiempo y del tamaño del archivo.

CloudTrail registra todas las acciones de CodeArtifact. Por ejemplo, las llamadas a las acciones `ListRepositories` (en AWS CLI, `aws codeartifact list-repositories`), `CreateRepository` (`aws codeartifact create-repository`) y `ListPackages` (`aws codeartifact list-packages`) generan entradas en los archivos de registro de CloudTrail, además de los comandos del cliente del administrador de paquetes. Los comandos del cliente del administrador de paquetes suelen realizar más de una solicitud HTTP al servidor. Cada solicitud genera un evento de registro de CloudTrail independiente.

Entrega de registros de CloudTrail entre cuentas

Hasta tres cuentas independientes reciben los registros de CloudTrail para una sola llamada a la API:

- La cuenta que realizó la solicitud, por ejemplo, la cuenta que llamó a `GetAuthorizationToken`.
- La cuenta del administrador del repositorio: por ejemplo, la cuenta que administra el repositorio al que llamó `ListPackages`.
- La cuenta del propietario del dominio, por ejemplo, la cuenta propietaria del dominio que contiene el repositorio en el que se llamó a la API.

En el caso de API como `ListRepositoriesInDomain`, que son acciones contra un dominio y no contra un repositorio específico, solo la cuenta que realiza la llamada y la cuenta del propietario del dominio reciben el registro de CloudTrail. En el caso de API como `ListRepositories` que no están autorizadas para ningún recurso, solo la cuenta de la persona que llama recibe el registro de CloudTrail.

Descripción de las entradas de los archivos de registro de CodeArtifact

Los archivos de registro de CloudTrail pueden contener una o más entradas de registro. Cada entrada muestra varios eventos con formato JSON. Un evento de registro representa una única solicitud de cualquier origen e incluye información sobre la acción solicitada, la fecha y la hora de la

acción, los parámetros de la solicitud, etcétera. Las entradas de registro no son un rastro de la pila ordenada de las llamadas API públicas, por lo que no aparecen en ningún orden específico.

Temas

Ejemplo: una entrada de registro para llamar a la API GetAuthorizationToken

Una entrada de registro creada por `GetAuthorizationToken` incluye el nombre de dominio en el campo `requestParameters`.


```
{
  "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"
}
```

Ejemplo: una entrada de registro para obtener una versión del paquete npm

En las solicitudes realizadas por todos los clientes del administrador de paquetes, incluido el cliente **npm**, se registran en el campo datos adicionales, como el nombre del dominio, el nombre del repositorio y el nombre del paquete `requestParameters`. La ruta URL y el método HTTP se registran en el campo `additionalEventData`.

```
{
   "eventVersion": "1.05",
   "userIdentity": {
       "type": "AssumedRole",
       "principalId": "AIDACKCEVSQ6C2EXAMPLE",
       "arn": "arn:aws:sts::123456789012:assumed-role/Console/example",
       "accountId": "123456789012",
       "accessKeyId": "ASIAIJIOBJIBSREXAMPLE",
       "sessionContext": {
           "attributes": {
               "mfaAuthenticated": "false",
               "creationDate": "2018-12-17T02:05:16Z"
           },
           "sessionIssuer": {
               "type": "Role",
               "principalId": "AIDACKCEVSQ6C2EXAMPLE",
               "arn": "arn:aws:iam::123456789012:role/Console",
               "accountId": "123456789012",
               "userName": "Console"
           }
       }
   },
   "eventTime": "2018-12-17T02:05:46Z",
   "eventSource": "codeartifact.amazonaws.com",
```

```
    "eventName": "ReadFromRepository",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.50",
    "userAgent": "npm/6.14.15 node/v12.22.9 linux x64 ci/custom",
    "requestParameters": {
        "domainName": "example-domain",
        "domainOwner": "123456789012",
        "repositoryName": "example-repo",
        "packageName": "lodash",
        "packageFormat": "npm",
        "packageVersion": "4.17.20"
    },
    "responseElements": null,
    "additionalEventData": {
        "httpMethod": "GET",
        "requestUri": "/npm/lodash/-/lodash-4.17.20.tgz"
    },
    "requestID": "9f74b4f5-3607-4bb4-9229-ffffffffffff",
    "eventID": "c74e40dd-8847-4058-a14d-ffffffffffff",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Validación de la conformidad en AWS CodeArtifact

Para saber si un servicio de Servicio de AWS está incluido en el ámbito de programas de conformidad específicos, consulte Servicios de Servicios de AWS en el ámbito del programa de conformidad y escoja el programa de conformidad que le interese. Para obtener información general, consulte Programas de conformidad de AWS.

Puede descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de conformidad al utilizar Servicios de AWS se determina en función de la sensibilidad de los datos, los objetivos de cumplimiento de su empresa y la legislación y los reglamentos correspondientes. AWS proporciona los siguientes recursos para ayudar con la conformidad:

- Guías de inicio rápido de seguridad y conformidad: estas guías de implementación tratan consideraciones sobre arquitectura y ofrecen pasos para implementar los entornos de referencia centrados en la seguridad y la conformidad en AWS.
- Arquitectura para la seguridad y el cumplimiento de la HIPAA en Amazon Web Services: en este documento técnico, se describe cómo las empresas pueden utilizar AWS para crear aplicaciones aptas para HIPAA.

> **Note**
>
> No todos los Servicios de AWS son aptos para HIPAA. Para obtener más información, consulte la Referencia de servicios aptos para HIPAA.

- Recursos de conformidad de AWS: este conjunto de manuales y guías podría aplicarse a su sector y ubicación.
- Guías de cumplimiento para clientes de AWS: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad de los Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés), el Consejo de Estándares de Seguridad de la Industria de Tarjetas de Pago (PCI, por sus siglas en inglés) y la Organización Internacional de Normalización (ISO, por sus siglas en inglés)).
- Evaluación de recursos con reglas en la Guía para desarrolladores de AWS Config: el servicio AWS Config evalúa en qué medida las configuraciones de sus recursos cumplen las prácticas internas, las directrices del sector y las normativas.
- AWS Security Hub: este servicio de Servicio de AWS proporciona una visión completa de su estado de seguridad en AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.
- AWS Audit Manager: este Servicio de AWS le ayuda a auditar continuamente el uso de AWS con el fin de simplificar la forma en que administra el riesgo y la conformidad con las normativas y los estándares del sector.

AWS CodeArtifact autenticación y tokens

CodeArtifact requiere que los usuarios se autentiquen con el servicio para poder publicar o consumir versiones de paquetes. Debe autenticarse en el CodeArtifact servicio creando un token

de autorización con sus AWS credenciales. Para crear un token de autorización, debe contar con los permisos correctos. Para conocer los permisos necesarios para crear un token de autorización, consulte la entrada `GetAuthorizationToken` enReferencia de permisos de AWS CodeArtifact. Para obtener más información general sobre CodeArtifact los permisos, consulte¿Cómo AWS CodeArtifact funciona con IAM.

Para obtener un token de autorización CodeArtifact, debes llamar a la GetAuthorizationToken API. Con el AWS CLI, puede llamar `GetAuthorizationToken` con el `get-authorization-token` comando `login` o.

> **Note**
>
> Los usuarios raíz no pueden llamar a `GetAuthorizationToken`.

- `aws codeartifact login`: Este comando facilita la configuración de los administradores de paquetes más comunes para usarlos CodeArtifact en un solo paso. Al llamar, se `login` obtiene un token `GetAuthorizationToken` y se configura el administrador de paquetes con el token y el punto final del CodeArtifact repositorio correcto. Los gestores de paquetes de soporte son los siguientes:
  - dotnet
  - npm
  - pepita
  - pip
  - rápido
  - cordel
- `aws codeartifact get-authorization-token`: En el caso de los gestores de paquetes que no sean compatibles con `login`, puede llamar a `get-authorization-token` directamente y, a continuación, configurar su gestor de paquetes con el token según sea necesario, por ejemplo, añadiéndolo a un archivo de configuración o almacenándolo en una variable de entorno.

CodeArtifact los tokens de autorización son válidos durante un período predeterminado de 12 horas. Los tokens se pueden configurar con una vida útil de entre 15 minutos y 12 horas. Cuando caduque la vida útil, debe buscar otro token. La vida útil del token comienza después de que se cancele `login` o `get-authorization-token`.

Si se llama a `login` o `get-authorization-token` mientras se asume un rol, puede configurar la duración del token para que sea igual al tiempo restante de la duración de la sesión del rol estableciendo el valor de `--duration-seconds` en `0`. De lo contrario, la duración del token es independiente de la duración máxima de la sesión del rol. Por ejemplo, supongamos que llama `sts assume-role` y especifica una duración de sesión de 15 minutos y, a continuación, llama `login` para obtener un token de CodeArtifact autorización. En este caso, el token es válido durante todo el período de 12 horas, aunque sea superior a los 15 minutos de duración de la sesión. Para obtener información sobre cómo controlar la duración de la sesión, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Tokens creados con el comando **`login`**

El `aws codeartifact login` comando buscará un token `GetAuthorizationToken` y configurará el administrador de paquetes con el token y el punto final del CodeArtifact repositorio correcto.

En la siguiente tabla se describen los parámetros del comando `login`.

Parámetro	Obligatoria	Descripción
`--tool`	Sí	El administrador de paquetes en el que autenticarse. Los valores posibles son `dotnet`npm, `nuget`pip, `swift` ytwine.
`--domain`	Sí	El nombre de dominio al que pertenece el repositorio.
`--domain-owner`	No	El ID del propietario del dominio. Este parámetro es obligatorio si se accede a un dominio que es propiedad de una AWS cuenta en la que no estás autenticado. Para obtener más información, consulte Dominios entre cuentas.
`--repository`	Sí	El nombre del repositorio en el que se va a autenticar.

Parámetro	Obligatoria	Descripción
`--duration-seconds`	No	Período en segundos, durante el que es válida la información de inicio de sesión. El valor mínimo es 900* y el valor máximo es 43200.
`--namespace`	No	Asocia un espacio de nombres a la herramienta de repositorio.
`--dry-run`	No	Imprima únicamente los comandos que se ejecutarían para conectar la herramienta con el repositorio sin realizar ningún cambio en la configuración.

*El valor 0 también es válido cuando se llama a `login` mientras se asume un rol. Al llamar a `login` con `--duration-seconds 0` , se crea un token con una duración igual al tiempo restante de la sesión de un rol asumido.

En el siguiente ejemplo se muestra cómo obtener un token de autorización con el comando `login`.

```
aws codeartifact login \
    --tool dotnet | npm | nuget | pip | swift | twine \
    --domain my_domain \
    --domain-owner 111122223333 \
    --repository my_repo
```

Para obtener instrucciones específicas sobre cómo usar el comando `login` con npm, consulte Configurar y usar npm con CodeArtifact. Para Python, consulte Uso de CodeArtifact con Python.

Tokens creados con la API **`GetAuthorizationToken`**

Puede llamar `get-authorization-token` para obtener un token de autorización. CodeArtifact

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
```

```
    --output text
```

Puede cambiar el tiempo de validez de un token usando el argumento `--duration-seconds`. El valor mínimo es 900 y el máximo, 43200. En el siguiente ejemplo se crea un token que durará 1 hora (3600 segundos).

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 3600
```

Si se llama a `get-authorization-token` mientras se asume un rol, la duración del token es independiente de la duración máxima de la sesión del rol. Puede configurar el token para que caduque cuando venza la duración de la sesión del rol asumido configurando `--duration-seconds` en 0.

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 0
```

Para obtener más información, consulte la documentación siguiente:

- Para obtener información sobre los tokens y las variables de entorno, consulte Pasar un token de autenticación mediante una variable de entorno.
- Para los usuarios de Python, consulte Configurar pip sin el comando login o Configurar y usar twine con CodeArtifact.
- Para los usuarios de Maven, consulte Uso de CodeArtifact con Gradle o Usar CodeArtifact con mvn.
- Para los usuarios de npm, consulte Configurar npm sin usar el comando login.

Pasar un token de autenticación mediante una variable de entorno

AWS CodeArtifact usa los tokens de autorización que vende la `GetAuthorizationToken` API para autenticar y autorizar las solicitudes de herramientas de compilación, como Maven y Gradle. Para obtener más información sobre estos tokens de autenticación, consulte Tokens creados con la API `GetAuthorizationToken`.

Puedes almacenar estos tokens de autenticación en una variable de entorno que pueda leer una herramienta de compilación para obtener el token que necesita para recuperar los paquetes de un CodeArtifact repositorio o publicarlos en él.

Por motivos de seguridad, este enfoque es preferible a almacenar el token en un archivo donde puedan leerlo otros usuarios o procesos, o bien introducirlo accidentalmente en el control de código fuente.

1. Configure sus AWS credenciales como se describe en. Instale o actualice y, a continuación, configure la AWS CLI
2. Establezca la variable de entorno `CODEARTIFACT_AUTH_TOKEN`:

   > **Note**
   >
   > En algunos casos, no es necesario incluir el argumento `--domain-owner`. Para obtener más información, consulte Dominios entre cuentas.

   - macOS o Linux:

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

   - Windows (mediante el intérprete de comandos predeterminado):

     ```
     for /f %i in ('aws codeartifact get-authorization-token --domain my_domain --
     domain-owner 111122223333 --query authorizationToken --output text') do set
      CODEARTIFACT_AUTH_TOKEN=%i
     ```

   - Windows PowerShell:

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

Revocación de los tokens CodeArtifact de autorización

Cuando un usuario autenticado crea un token para acceder a CodeArtifact los recursos, ese token permanece activo hasta que finalice el período de acceso personalizable. El período de acceso predeterminado es de 12 horas. En algunos casos, es posible que le sea útil revocar el acceso a un token antes de que el período de acceso haya expirado. Puedes revocar el acceso a CodeArtifact los recursos siguiendo estas instrucciones.

Si creó el token de acceso con credenciales de seguridad temporales, como roles asumidos o acceso de usuarios federados, puede revocar el acceso actualizando una política de IAM para denegar el acceso. Para obtener información, consulte Deshabilitar permisos para credenciales de seguridad temporales en la Guía del usuario de IAM.

Si ha utilizado credenciales de usuario de IAM de larga duración para crear el token de acceso, debe modificar la política del usuario para denegar el acceso o eliminar al usuario de IAM. Para obtener más información, consulte Cambiar permisos para un usuario de IAM o Eliminar un usuario de IAM.

Resiliencia en AWS CodeArtifact

La infraestructura global de AWS se compone de regiones y zonas de disponibilidad de AWS. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. AWS CodeArtifact opera en varias zonas de disponibilidad y almacena datos y metadatos de artefactos en Amazon S3 y Amazon DynamoDB. Sus datos cifrados se almacenan de forma redundante en múltiples instalaciones y múltiples dispositivos en cada instalación, lo que los hace altamente disponibles y duraderos.

Para obtener más información sobre las regiones y zonas de disponibilidad de AWS, consulte Infraestructura global de AWS.

Seguridad de la infraestructura AWS CodeArtifact

Como se trata de un servicio administrado, AWS CodeArtifact está protegido por la seguridad de red global de AWS. Para obtener información sobre los servicios de seguridad de AWS y cómo AWS protege la infraestructura, consulte Seguridad en la nube de AWS. Para diseñar su entorno de AWS con las prácticas recomendadas de seguridad de infraestructura, consulte Protección de la infraestructura en Portal de seguridad de AWS Well-Architected Framework.

Puede utilizar llamadas a la API publicadas en AWS para obtener acceso a CodeArtifact a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Nosotros exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) tales como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad de seguridad de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Ataques de sustitución de dependencias

Los gestores de paquetes simplifican el proceso de empaquetar y compartir código reutilizable. Estos paquetes pueden ser paquetes privados desarrollados por una organización para usarlos en sus aplicaciones, o pueden ser públicos, generalmente paquetes de código abierto que se desarrollan fuera de una organización y se distribuyen en repositorios de paquetes públicos. Al solicitar paquetes, los desarrolladores confían en su administrador de paquetes para obtener nuevas versiones de sus dependencias. Los ataques de sustitución de dependencias, también conocidos como ataques de confusión de dependencias, aprovechan el hecho de que un administrador de paquetes normalmente no tiene forma de distinguir las versiones legítimas de un paquete de las versiones maliciosas.

Los ataques de sustitución de dependencias pertenecen a un subconjunto de hackeos conocidos como ataques a la cadena de suministro de software. Un ataque a la cadena de suministro de software es un ataque que aprovecha las vulnerabilidades en cualquier punto de la cadena de suministro de software.

Un ataque de sustitución de dependencias puede dirigirse a cualquier persona que utilice tanto paquetes desarrollados internamente como paquetes extraídos de repositorios públicos. Los atacantes identifican los nombres de los paquetes internos y, a continuación, colocan estratégicamente el código malicioso con el mismo nombre en los repositorios de paquetes públicos. Normalmente, el código malicioso se publica en un paquete con un número de versión alto. Los administradores de paquetes obtienen el código malicioso de estas fuentes públicas porque creen que los paquetes maliciosos son las últimas versiones del paquete. Esto provoca una «confusión» o una «sustitución» entre el paquete deseado y el paquete malicioso, lo que hace que el código quede comprometido.

Para evitar los ataques de sustitución de dependencias, AWS CodeArtifact proporciona controles de origen de los paquetes. Los controles de origen de los paquetes son ajustes que controlan cómo se pueden añadir los paquetes a tus repositorios. Los controles se configuran automáticamente cuando se agrega la primera versión de paquete de un paquete nuevo a un repositorio de CodeArtifact. Los controles pueden garantizar que las versiones de los paquetes no puedan publicarse directamente en su repositorio ni ingerirse de fuentes públicas, lo que lo protege de los ataques de sustitución de dependencias. Para obtener más información sobre los controles de origen de los paquetes y cómo cambiarlos, consulte Edición de los controles de origen del paquete.

Identity and Access Management para AWS CodeArtifact

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos. CodeArtifact La IAM es una Servicio de AWS opción que puede utilizar sin coste adicional.

Temas

- Solución de problemas de AWS CodeArtifact identidad y acceso

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo en el que se realice. CodeArtifact

Usuario del servicio: si utiliza el CodeArtifact servicio para realizar su trabajo, el administrador le proporcionará las credenciales y los permisos que necesita. A medida que vaya utilizando más CodeArtifact funciones para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una función en CodeArtifact, consulteSolución de problemas de AWS CodeArtifact identidad y acceso.

Administrador de servicios: si está a cargo de CodeArtifact los recursos de su empresa, probablemente tenga acceso total a ellos CodeArtifact. Su trabajo consiste en determinar a qué CodeArtifact funciones y recursos deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar la IAM CodeArtifact, consulte¿Cómo AWS CodeArtifact funciona con IAM.

Administrador de IAM: si es administrador de IAM, puede que le interese obtener más información sobre cómo redactar políticas para administrar el acceso. CodeArtifact Para ver ejemplos de políticas CodeArtifact basadas en la identidad que puede usar en IAM, consulte. Ejemplos de políticas basadas en la identidad para AWS CodeArtifact

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar las solicitudes de la AWS API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS Single Sign-On y Uso de la autenticación multifactor (MFA) en AWS en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS Single Sign-On. Puede crear usuarios y grupos en el Centro de identidades de IAM, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente una función de IAM en el AWS Management Console cambiando de función. Puede asumir un rol llamando a una operación de AWS API AWS CLI o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del usuario de IAM. Si utiliza IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS Single Sign-On.
- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.
- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en ellas AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.
  - Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio

desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

- Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puede usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y realizan AWS CLI solicitudes a la API. AWS Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia EC2. Para asignar una AWS función a una instancia EC2 y ponerla a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia EC2 obtener credenciales temporales. Para más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Para conceder permiso a los usuarios para realizar acciones en los recursos que necesiten, un administrador de IAM puede crear políticas de IAM. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede adjuntar a una identidad, como un usuario, un grupo de usuarios o un rol de IAM. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios que admiten las ACL. AWS WAF Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifique el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): las SCP son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o a todas sus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una. Usuario raíz de la cuenta de AWS Para obtener más información acerca de Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo AWS determinar si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

¿Cómo AWS CodeArtifact funciona con IAM

Antes de utilizar IAM para gestionar el acceso CodeArtifact, infórmese sobre las funciones de IAM disponibles para su uso. CodeArtifact

Funciones de IAM que puede utilizar con AWS CodeArtifact

Característica de IAM	CodeArtifact soporte
Políticas basadas en identidades	Sí
Políticas basadas en recursos	Sí
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	No
ACL	No
ABAC (etiquetas en políticas)	Parcial
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	No
Roles vinculados al servicio	No

Para obtener una visión general de cómo CodeArtifact funcionan otros AWS servicios con la mayoría de las funciones de IAM, consulte AWS los servicios que funcionan con IAM en la Guía del usuario de IAM.

Políticas basadas en la identidad para CodeArtifact

Compatibilidad con las políticas basadas en identidades	Sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está adjunto. Para más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en la identidad para CodeArtifact

Para ver ejemplos de políticas CodeArtifact basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS CodeArtifact

Políticas basadas en recursos incluidas CodeArtifact

Compatibilidad con las políticas basadas en recursos	Sí

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico.

Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política en función de recursos concede el acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para más información, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

Acciones políticas para CodeArtifact

Admite acciones de política	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de CodeArtifact acciones, consulta las acciones definidas AWS CodeArtifact en la Referencia de autorización del servicio.

Las acciones políticas CodeArtifact utilizan el siguiente prefijo antes de la acción:

```
codeartifact
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "codeartifact:action1",
      "codeartifact:action2"
         ]
```

Puede utilizar caracteres comodín (*) para especificar varias acciones . Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

```
"Action": "codeartifact:Describe*"
```

Para ver ejemplos de políticas CodeArtifact basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS CodeArtifact

Recursos de políticas para CodeArtifact

Admite recursos de políticas	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de los tipos de CodeArtifact recursos y sus ARN, consulte los recursos definidos AWS CodeArtifact en la Referencia de autorización de servicios. Para saber con qué acciones puede especificar el ARN de cada recurso, consulte Acciones definidas por. AWS CodeArtifact Para

ver ejemplos de cómo especificar los ARN de los CodeArtifact recursos en las políticas, consulte. Recursos y operaciones de AWS CodeArtifact

Claves de condición de la política para CodeArtifact

Admite claves de condición de políticas específicas del servicio	No

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación lógica `AND`. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulte Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

> **Note**
>
> AWS CodeArtifact no admite las siguientes claves de contexto de condiciones AWS globales:
>
> - Referer
> - UserAgent

Para ver una lista de claves de CodeArtifact condición, consulte las claves de condición AWS CodeArtifact en la Referencia de autorización de servicio. Para saber con qué acciones y recursos puede utilizar una clave de condición, consulte Acciones definidas por AWS CodeArtifact.

Para ver ejemplos de políticas CodeArtifact basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS CodeArtifact

ACL en CodeArtifact

Admite las ACL	No

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con CodeArtifact

Admite ABAC (etiquetas en las políticas)	Parcial

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte ¿Qué es ABAC? en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Para obtener más información sobre el etiquetado de CodeArtifact los recursos, incluidos ejemplos de políticas basadas en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte. Uso de etiquetas para controlar el acceso a los recursos de CodeArtifact

Usar credenciales temporales con CodeArtifact

Compatible con el uso de credenciales temporales	Sí

Algunas Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluida la información sobre cuáles Servicios de AWS funcionan con credenciales temporales, consulta Cómo Servicios de AWS funcionan con IAM en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para más información sobre el cambio de roles, consulte Cambio a un rol (consola) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para más información, consulte Credenciales de seguridad temporales en IAM.

Permisos principales entre servicios para CodeArtifact

Admite Forward access sessions (FAS)	Sí

Cuando utilizas un usuario o un rol de IAM para realizar acciones en él AWS, se te considera principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra

acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

Hay dos acciones de la CodeArtifact API que requieren que el responsable de la llamada tenga permisos en otros servicios:

1. `GetAuthorizationToken` requiere `sts:GetServiceBearerToken` junto con `codeartifact:GetAuthorizationToken`.
2. `CreateDomain`, al proporcionar una clave de cifrado no predeterminada, requiere tanto `kms:DescribeKey` como `kms:CreateGrant` en la clave KMS junto con `codeartifact:CreateDomain`.

Para obtener más información sobre los permisos y recursos necesarios para las acciones en CodeArtifact, consulteReferencia de permisos de AWS CodeArtifact.

Roles de servicio para CodeArtifact

Compatible con funciones de servicio	No

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir CodeArtifact la funcionalidad. Edite las funciones de servicio solo cuando se CodeArtifact proporcionen instrucciones para hacerlo.

Funciones vinculadas al servicio para CodeArtifact

Compatible con roles vinculados al servicio	No

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Ejemplos de políticas basadas en la identidad para AWS CodeArtifact

De forma predeterminada, los usuarios y los roles no tienen permiso para crear o modificar CodeArtifact recursos. Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o la AWS API. Para conceder permiso a los usuarios para realizar acciones en los recursos que necesiten, un administrador de IAM puede crear políticas de IAM. A continuación, el administrador puede añadir las políticas de IAM a roles, y los usuarios pueden asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por cada uno de los tipos de recursos CodeArtifact, incluido el formato de los ARN para cada uno de los tipos de recursos, consulte las claves de condición, recursos y acciones de la Referencia de autorización de servicios. AWS CodeArtifact

Temas

- Prácticas recomendadas sobre las políticas
- Mediante la consola de CodeArtifact
- Políticas administradas por AWS (predefinidas) para AWS CodeArtifact
- Permitir a un usuario consultar sus propios permisos

- Permitir que un usuario obtenga información sobre repositorios y dominios
- Permitir que un usuario obtenga información sobre dominios específicos
- Permitir que un usuario obtenga información sobre repositorios específicos
- Limitar la duración del token de autorización

Prácticas recomendadas sobre las políticas

Las políticas basadas en la identidad determinan si alguien puede crear CodeArtifact recursos de tu cuenta, acceder a ellos o eliminarlos. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de trabajo en la Guía del usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utilice el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para

más información, consulte Política de validación de Analizador de acceso de IAM en la Guía de usuario de IAM.

- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para solicitar la MFA cuando se invocan las operaciones de la API, agregue las condiciones de la MFA a sus políticas. Para más información, consulte Configuración del acceso a una API protegido por MFA en la Guía de usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte las Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Mediante la consola de CodeArtifact

Para acceder a la AWS CodeArtifact consola, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los CodeArtifact recursos de su cuenta Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realicen llamadas a la API AWS CLI o a la AWS API. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para garantizar que los usuarios y los roles puedan seguir utilizando la CodeArtifact consola, adjunte también la política `AWSCodeArtifactReadOnlyAccess` AWS gestionada `AWSCodeArtifactAdminAccess` o la política gestionada a las entidades. Para más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM:

Políticas administradas por AWS (predefinidas) para AWS CodeArtifact

AWS aborda muchos casos de uso comunes al proporcionar políticas de IAM independientes que son creadas y administradas por. AWS Estas políticas AWS gestionadas conceden los permisos necesarios para los casos de uso más habituales, de forma que no tengas que investigar qué permisos son necesarios. Para más información, consulte Políticas administradas por AWS en la Guía del usuario de IAM.

Las siguientes políticas AWS administradas, que puedes adjuntar a los usuarios de tu cuenta, son específicas de AWS CodeArtifact.

- `AWSCodeArtifactAdminAccess`— Proporciona acceso completo e CodeArtifact incluye permisos para administrar CodeArtifact dominios.

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

- `AWSCodeArtifactReadOnlyAccess`— Proporciona acceso de solo lectura a. 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"
            }
         }
      }
   ]
}
```

Para crear y gestionar funciones CodeArtifact de servicio, también debe adjuntar la política AWS gestionada denominada. `IAMFullAccess`

También puedes crear tus propias políticas de IAM personalizadas para permitir permisos para CodeArtifact acciones y recursos. Puede asociar estas políticas personalizadas a los grupos o usuarios de IAM que requieran esos permisos.

Permitir a un usuario consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la AWS CLI API o. AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
```

```
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Permitir que un usuario obtenga información sobre repositorios y dominios

La siguiente política permite a un usuario o rol de IAM enumerar y describir cualquier tipo de CodeArtifact recurso, incluidos dominios, repositorios, paquetes y activos. La política también incluye el `codeArtifact:ReadFromRepository` permiso, que permite al director obtener paquetes de un repositorio. CodeArtifact No permite crear nuevos dominios o repositorios y no permite publicar nuevos paquetes.

Se requieren permisos `codeartifact:GetAuthorizationToken` y `sts:GetServiceBearerToken` para llamar a la API `GetAuthorizationToken`.

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

Permitir que un usuario obtenga información sobre dominios específicos

A continuación se muestra un ejemplo de una política de permisos que permite a un usuario enumerar dominios sólo en la región `us-east-2` para la cuenta `123456789012` para cualquier dominio que comience con el nombre `my`.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "codeartifact:ListDomains",
         "Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/my*"
      }
   ]
}
```

Permitir que un usuario obtenga información sobre repositorios específicos

A continuación se muestra un ejemplo de una política de permisos que permite al usuario obtener información sobre los repositorios que terminan en `test`, incluida la información sobre los paquetes que contienen. El usuario no podrá publicar, crear ni eliminar recursos.

Se requieren permisos `codeartifact:GetAuthorizationToken` y `sts:GetServiceBearerToken` para llamar a la API `GetAuthorizationToken`.

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

Limitar la duración del token de autorización

Los usuarios deben autenticarse CodeArtifact con tokens de autorización para publicar o consumir versiones de paquetes. Los tokens de autorización son válidos solo durante su vida útil configurada. Los tokens tienen una vida útil predeterminada de 12 horas. Para obtener más información sobre los tokens de autorización, consulte AWS CodeArtifact autenticación y tokens.

Al buscar un token, los usuarios pueden configurar la vida útil del token. Los valores válidos durante la vida útil de un token de autorización son `0`, y cualquier número comprendido entre `900` (15 minutos) y `43200` (12 horas). Un valor de `0` creará un token con una duración igual a las credenciales temporales del rol del usuario.

Los administradores pueden limitar los valores válidos durante la vigencia de un token de autorización mediante la clave de condición `sts:DurationSeconds` de la política de permisos adjunta al usuario o grupo. Si el usuario intenta crear un token de autorización con una duración superior a los valores válidos, no se podrá crear el token.

Los siguientes ejemplos de políticas limitan las posibles duraciones de un token de autorización creado por CodeArtifact los usuarios.

Ejemplo de política: limitar la vida útil del token a exactamente 12 horas (43200 segundos)

Con esta política, los usuarios solo podrán crear tokens de autorización con una duración de 12 horas.

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

Política de ejemplo: limitar la vida útil del token entre 15 minutos y 1 hora, o igual al período de credenciales temporales del usuario

Con esta política, los usuarios podrán crear tokens con una validez de entre 15 minutos y 1 hora. Los usuarios también podrán crear un token que dure el tiempo que duren las credenciales temporales de su rol si especifican `0` para `--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"
                }
            }
        }
    ]
}
```

Uso de etiquetas para controlar el acceso a los recursos de CodeArtifact

Las condiciones de las declaraciones de política de usuario de IAM forman parte de la sintaxis que se utiliza para especificar permisos para los recursos que necesitan las acciones de CodeArtifact.

El uso de etiquetas en las condiciones es una manera de controlar el acceso a los recursos y las solicitudes. Para obtener más información acerca del etiquetado de recursos de CodeArtifact, consulte Etiquetado de recursos. En este tema, se explica el control de acceso basado en etiquetas.

Al diseñar políticas de IAM, es posible que se establezcan permisos detallados mediante la concesión de acceso a recursos específicos. A medida que crezca la cantidad de recursos que administra, esta tarea será más complicada. El etiquetado de recursos y uso de etiquetas en las condiciones de declaración de política pueden facilitar esta tarea. Puede conceder acceso de forma masiva a cualquier recurso con una determinada etiqueta. A continuación, aplique repetidamente esta etiqueta a los recursos pertinentes durante la creación o después.

Las etiquetas se pueden asociar al recurso o pasarse dentro de la solicitud a los servicios que admiten etiquetado. En CodeArtifact, los recursos pueden tener etiquetas y algunas acciones puede incluir etiquetas. Al crear una política de IAM, puede utilizar las claves de condición de etiqueta para controlar:

- Qué usuarios pueden realizar acciones en un dominio o recurso de repositorio, según las etiquetas que ya tiene.
- Qué etiquetas se pueden pasar en la solicitud de una acción.
- Si se pueden utilizar claves de etiqueta específicas en una solicitud.

Para la sintaxis y semántica completas de las claves de condición de etiquetas, consulte Control del acceso mediante etiquetas en la Guía del usuario de IAM.

> **Important**
>
> Cuando se utilizan etiquetas en los recursos para limitar las acciones, las etiquetas deben estar en el recurso en el que se ejecuta la acción. Por ejemplo, para denegar permisos `DescribeRepository` con etiquetas, estas deben estar en cada repositorio y no en el dominio. Consulte Referencia de permisos de AWS CodeArtifact para obtener una lista de acciones en CodeArtifact y los recursos con los que operan.

Ejemplos de control de acceso basado en etiquetas

Los siguientes ejemplos muestran cómo especificar las condiciones de etiqueta en las políticas para los usuarios de CodeArtifact.

Example 1: Limitar acciones en función de etiquetas en la solicitud

La política de usuario administrada `AWSCodeArtifactAdminAccess` proporciona a los usuarios permisos ilimitados para realizar cualquier acción de CodeArtifact en cualquier recurso.

La siguiente política limita esta posibilidad y deniega el permiso a usuarios no autorizados para crear repositorios a menos que la solicitud contenga determinadas etiquetas. Para ello, deniega la acción `CreateRepository` si la solicitud no especifica una etiqueta denominada `costcenter` con uno de los valores 1 o 2. El administrador de un cliente debe asociar esta política de IAM a los usuarios de IAM no autorizados, además de la política de usuario administrada.

```
{
  "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: Limitar acciones en función de etiquetas de recursos

La política de usuario administrada `AWSCodeArtifactAdminAccess` proporciona a los usuarios permisos ilimitados para realizar cualquier acción de CodeArtifact en cualquier recurso.

La siguiente política limita esta capacidad y deniega a los usuarios no autorizados el permiso para realizar acciones en repositorios en dominios específicos. Para ello, deniega algunas acciones si el recurso tiene una etiqueta denominada `Key1` con el valor `Value1` o `Value2`. (La clave de condición `aws:ResourceTag` se utiliza para controlar el acceso a los recursos en función de las etiquetas que tengan los recursos). El administrador de un cliente debe asociar esta política de IAM a los usuarios de IAM no autorizados, además de la política de usuario administrada.

```
{
  "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: Permitir acciones en función de las etiquetas de recursos

La siguiente política otorga a los usuarios permiso para realizar acciones y obtener información sobre repositorios y paquetes en CodeArtifact.

Para ello, permite realizar determinadas acciones si el repositorio tiene una etiqueta denominada `Key1` con el valor `Value1`. (La clave de condición `aws:RequestTag` se utiliza para controlar qué etiquetas se pueden pasar en una solicitud de IAM). La condición `aws:TagKeys` garantiza la distinción entre mayúsculas y minúsculas de las claves de etiqueta. Esta política es útil para los usuarios de IAM que no tienen asociada la política de usuario administrada `AWSCodeArtifactAdminAccess`. La política administrada proporciona a los usuarios permisos ilimitados para realizar cualquier acción de CodeArtifact en cualquier recurso.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:UpdateRepository",
        "codeartifact:DeleteRepository",
        "codeartifact:ListPackages"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

Example 4: Permitir acciones en función de las etiquetas en la solicitud

La siguiente política concede a los usuarios permiso para crear repositorios en determinados dominios en CodeArtifact.

Para ello, permite las acciones `CreateRepository` y `TagResource` si la API de creación de recursos en la solicitud especifica una etiqueta denominada `Key1` con el valor `Value1`. (La clave de condición `aws:RequestTag` se utiliza para controlar qué etiquetas se pueden pasar en una solicitud

de IAM). La condición `aws:TagKeys` garantiza la distinción entre mayúsculas y minúsculas de las claves de etiqueta. Esta política es útil para los usuarios de IAM que no tienen asociada la política de usuario administrada `AWSCodeArtifactAdminAccess`. La política administrada proporciona a los usuarios permisos ilimitados para realizar cualquier acción de CodeArtifact en cualquier recurso.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:CreateRepository",
        "codeartifact:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

Referencia de permisos de AWS CodeArtifact

Recursos y operaciones de AWS CodeArtifact

En AWS CodeArtifact, el recurso principal es un dominio. En una política, se usa un nombre de recurso de Amazon (ARN) para identificar el recurso al que se aplica la política. Los repositorios son también recursos y tienen ARN asociados. Para obtener más información, consulte Nombres de recurso de Amazon (ARN) en la Referencia general de Amazon Web Services.

Tipo de recurso	Formato de ARN
Dominio	`arn:aws:codeartifact:` *region-ID* `:`*account-ID*`:domain/`*my_domain*
Repositorio	`arn:aws:codeartifact:` *region-ID* `:`*account-ID*`:repository/` *my_domain* `/`*my_repo*

Tipo de recurso	Formato de ARN
Paquete con un espacio de nombres	`arn:aws:codeartifact:` *region-ID* `:`*account-ID* `:package/` *my_domain* `/`*my_repo*`/`*package-format*`/`*namespace*`/`*package-name*
Paquete sin un espacio de nombres	`arn:aws:codeartifact:` *region-ID* `:`*account-ID* `:package/` *my_domain* `/`*my_repo*`/`*package-format*`//`*package-name*
Todos los recursos de CodeArtifact	`arn:aws:codeartifact:*`
Todos los recursos de CodeArtifact que pertenecen a la cuenta especificada en la región de AWS indicada	`arn:aws:codeartifact:` *region-ID* `:`*account-ID* `:*`

El ARN del recurso que especifique depende de la acción o acciones cuyo acceso desee controlar.

Puede indicar un dominio específico (*myDomain*) en su declaración utilizando su ARN de la siguiente manera.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain"
```

Puede indicar un repositorio específico (*myRepo*) en la instrucción usando su ARN de este modo.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain/myRepo"
```

Para especificar varios recursos en una única instrucción, separe sus ARN con comas. La siguiente declaración se aplica a todos los paquetes y repositorios de un dominio específico.

```
"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**
>
> Muchos servicios de AWS tratan el carácter de dos puntos (:) o la barra inclinada (/) como el mismo carácter en los ARN. Sin embargo, CodeArtifact utiliza una coincidencia exacta en los patrones de recursos y reglas. Asegúrese de utilizar los caracteres correctos cuando cree patrones de eventos para que coincidan con la sintaxis de ARN en el recurso.

Permisos y operaciones de la API de AWS CodeArtifact

Puede utilizar la siguiente tabla como referencia cuando configure políticas de control de acceso y de escritura pueda asociar a una identidad de IAM (políticas basadas en identidad).

Puede utilizar claves de condiciones generales de AWS en sus políticas de AWS CodeArtifact para expresar condiciones. Para más información, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del Usuario de IAM.

Las acciones se especifican en el campo `Action` de la política. Para especificar una acción, use el prefijo `codeartifact:` seguido del nombre de operación de API (por ejemplo, `codeartifact:CreateDomain` y `codeartifact:AssociateExternalConnection`). Para especificar varias acciones en una única instrucción, sepárelas con comas (por ejemplo, `"Action": [ "codeartifact:CreateDomain", "codeartifact:AssociateExternalConnection" ]`).

Uso de caracteres comodín

Debe especificar un ARN, con o sin un carácter comodín (*), como el valor del recurso en el campo `Resource` de la política. Puede utilizar un carácter comodín para especificar varias acciones o recursos. Por ejemplo, `codeartifact:*` especifica todas las acciones de CodeArtifact y `codeartifact:Describe*` especifica todas las acciones de CodeArtifact que comienzan por la palabra `Describe`.

Solución de problemas de AWS CodeArtifact identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas habituales que pueden surgir al trabajar con un CodeArtifact IAM.

Temas

- No estoy autorizado a realizar ninguna acción en CodeArtifact

- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis CodeArtifact recursos

No estoy autorizado a realizar ninguna acción en CodeArtifact

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `codeartifact:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codeartifact:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `codeartifact:`*`GetWidget`*.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis CodeArtifact recursos

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para saber si CodeArtifact es compatible con estas funciones, consulte¿Cómo AWS CodeArtifact funciona con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro usuario de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.

- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM.

Trabajar con puntos de conexión de VPC de Amazon

Puede configurarlo CodeArtifact para usar un punto final de interfaz de nube privada virtual (VPC) para mejorar la seguridad de su VPC.

Los puntos finales de VPC utilizan AWS PrivateLink un servicio que le permite acceder a CodeArtifact las API a través de direcciones IP privadas. AWS PrivateLink restringe todo el tráfico de red entre la VPC CodeArtifact y la red de AWS. Cuando utiliza un punto de conexión de VPC de interfaz, no necesita una puerta de enlace de Internet, un dispositivo NAT o una puerta de enlace privada virtual. Para obtener más información, consulte Puntos de enlace de la VPC en la Guía del usuario de Amazon Virtual Private Cloud.

Important

- Los puntos finales de VPC no admiten solicitudes entre regiones.AWS Asegúrese de crear su punto de conexión en la misma AWS región a la que planea realizar las llamadas a la API. CodeArtifact
- Los puntos de conexión de VPC solo admiten DNS proporcionadas por Amazon a través de Amazon Route 53. Si desea utilizar su propio DNS, puede utilizar el enrutamiento de DNS condicional. Para obtener más información, consulte Conjuntos de opciones de DHCP en la Guía del usuario de Amazon Virtual Private Cloud.
- El grupo de seguridad asociado al punto de conexión de la VPC debe permitir las conexiones entrantes en el puerto 443 desde la subred privada de la VPC.

Temas

Cree puntos finales de VPC para CodeArtifact

Para crear puntos de enlace de nube privada virtual (VPC) CodeArtifact, utilice el comando Amazon EC2. `create-vpc-endpoint` AWS CLI Para obtener más información, consulte Puntos de

conexión de VPC de tipo interfaz (AWS PrivateLink) en la Guía del usuario de Amazon Virtual Private Cloud.

Se requieren dos puntos de enlace de VPC para que todas las solicitudes CodeArtifact estén en la red. AWS El primer punto final se utiliza para llamar a CodeArtifact las API (por ejemplo, `GetAuthorizationToken` y`CreateRepository`).

```
com.amazonaws.region.codeartifact.api
```

El segundo punto final se usa para acceder a CodeArtifact los repositorios mediante administradores de paquetes y herramientas de compilación (por ejemplo, npm y Gradle).

```
com.amazonaws.region.codeartifact.repositories
```

El siguiente comando crea un punto final para acceder a los repositorios. CodeArtifact

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --vpc-endpoint-type Interface \
  --service-name com.amazonaws.region.codeartifact.api --subnet-ids subnetid \
  --security-group-ids groupid --no-private-dns-enabled
```

El siguiente comando crea un punto de conexión para acceder a los administradores de paquetes y a las herramientas de compilación.

```
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**
>
> Cuando crea un punto de conexión `codeartifact.repositories`, debe crear un nombre de host DNS privado usando la opción `--private-dns-enabled`. Sin embargo, dado que actualmente no se admiten varios nombres de host DNS privados para los puntos de conexión `codeartifact.api` y `codeartifact.repositories`, utilice la opción `--no-private-dns-enabled` para `codeartifact.api`. Si no puedes o no quieres crear un nombre de host DNS privado al crear el `codeartifact.repositories` punto final, debes seguir un paso de configuración adicional para usar tu administrador de paquetes CodeArtifact desde una VPC. Para obtener más información, consulte Utilice el punto de conexión `codeartifact.repositories` sin DNS privado.

Después de crear los puntos de enlace de la VPC, es posible que tenga que realizar más configuraciones con las reglas de los grupos de seguridad para usar los puntos de enlace. CodeArtifact Para obtener más información sobre los grupos de seguridad en Amazon VPC, consulte Grupos de seguridad.

Si tiene problemas para conectarse CodeArtifact, puede utilizar la herramienta Reachability Analyzer de VPC para depurar el problema. Para obtener más información, consulte ¿Qué es VPC Reachability Analyzer?

Creación del punto de enlace de gateway de Amazon S3

CodeArtifact utiliza Amazon Simple Storage Service (Amazon S3) para almacenar los activos del paquete. Para extraer paquetes CodeArtifact, debe crear un punto de enlace de enlace para Amazon S3. Cuando su proceso de compilación o implementación descarga paquetes CodeArtifact, debe acceder CodeArtifact para obtener los metadatos del paquete y Amazon S3 para descargar los activos del paquete (por ejemplo, `.jar` archivos Maven).

> **Note**
>
> No se necesita un punto de conexión Amazon S3 cuando se utilizan los formatos de paquete Python o Swift.

Para crear el punto de enlace de la puerta de enlace Amazon S3 CodeArtifact, utilice el comando Amazon EC2 `create-vpc-endpoint` AWS CLI . Cuando cree el punto de conexión, debe seleccionar las tablas de enrutamiento para su VPC. Para obtener más información, consulte Puntos de conexión de VPC de la puerta de enlace en la Guía del usuario de Amazon Virtual Private Cloud.

El comando siguiente crea un punto de conexión de Amazon S3.

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --service-name com.amazonaws.region.s3 \
  --route-table-ids routetableid
```

Permisos mínimos de bucket de Amazon S3 para AWS CodeArtifact

El punto de enlace de gateway de Amazon S3 utiliza un documento de políticas de IAM para limitar el acceso al servicio. Para permitir solo los permisos mínimos del bucket de Amazon S3 CodeArtifact,

restrinja el acceso al bucket de Amazon S3 que se CodeArtifact utiliza al crear el documento de política de IAM para el punto final.

En la siguiente tabla se describen los buckets de Amazon S3 a los que debe hacer referencia en sus políticas para permitir el acceso CodeArtifact en cada región.

Región	ARN del bucket de Amazon S3
us-east-1	arn:aws:s3:::assets-193858265520-us-east-1
us-east-2	arn:aws:s3:::assets-250872398865-us-east-2
us-west-2	arn:aws:s3:::assets-787052242323-us-west-2
eu-west-1	arn:aws:s3:::assets-438097961670-eu-west-1
eu-west-2	arn:aws:s3:::assets-247805302724-eu-west-2
eu-west-3	arn:aws:s3:::assets-762466490029-eu-west-3
eu-north-1	arn:aws:s3:::assets-611884512288-eu-north-1
eu-south-1	arn:aws:s3:::assets-484130244270-eu-south-1
eu-central-1	arn:aws:s3:::assets-769407342218-eu-central-1
ap-northeast-1	arn:aws:s3:::assets-660291247815-ap-northeast-1
ap-southeast-1	arn:aws:s3:::assets-421485864821-ap-southeast-1
ap-southeast-2	arn:aws:s3:::assets-860415559748-ap-southeast-2
ap-south-1	arn:aws:s3:::assets-681137435769-ap-south-1

Puede usar el `aws codeartifact describe-domain` comando para recuperar el bucket de Amazon S3 utilizado por un CodeArtifact dominio.

```
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"
  }
}
```

Ejemplo

El siguiente ejemplo ilustra cómo proporcionar acceso a los buckets de Amazon S3 necesarios para CodeArtifact las operaciones en la `us-east-1` región. Para otras regiones, actualice la entrada `Resource` con el ARN de permiso correcto para su región según la tabla anterior.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::assets-193858265520-us-east-1/*"]
    }
  ]
}
```

Uso CodeArtifact desde una VPC

Para llamar a CodeArtifact las API mediante el SDK AWS CLI o mediante un punto de enlace de VPC, debe anular el punto de enlace predeterminado utilizado por. CodeArtifact Siga las instrucciones de Configure el AWS CLI para usar el `codeartifact.api` punto final para obtener el nombre de host del punto de conexión de VPC y configurar la CLI con él.

Si no puede o no quiere habilitar el DNS privado en el punto de enlace de `com.amazonaws.region.codeartifact.repositories` VPC en el que creóCree puntos finales de VPC para CodeArtifact, debe usar una configuración diferente para el punto de enlace de los repositorios para usarlo desde CodeArtifact una VPC. Siga las instrucciones Utilice el punto de conexión `codeartifact.repositories` sin DNS privado para configurar CodeArtifact si el `com.amazonaws.region.codeartifact.repositories` punto final no tiene habilitado el DNS privado.

Configure el AWS CLI para usar el **`codeartifact.api`** punto final

Siga las instrucciones siguientes para anular el nombre de CodeArtifact host predeterminado por el nombre de host utilizado por el punto final de la `com.amazonaws.region.codeartifact.api` VPC.

1. Ejecute el siguiente comando para buscar un punto de conexión de VPC y usarlo para anular el nombre de host.

```
$ aws ec2 describe-vpc-endpoints --filters Name=service-
name,Values=com.amazonaws.region.codeartifact.api \
  --query 'VpcEndpoints[*].DnsEntries[*].DnsName'
```

   El resultado es similar al siguiente.

```
[
  [
    "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"
  ]
]
```

En este ejemplo, puede usar cualquier nombre de host para anular el punto de conexión `com.amazonaws.region.codeartifact.api`.

2. Si utiliza el CodeArtifact AWS CLI, utilice el `codeartifact login` comando para anular el CodeArtifact nombre de host predeterminado con el punto de enlace de Amazon VPC pasando el punto de enlace al parámetro. `--endpoint-url` Consulte el siguiente ejemplo.

> **Warning**
>
> El comando `login` no es compatible con Maven ni Gradle. Para configurar esos administradores de paquetes, consulte Uso de CodeArtifact con Maven.

```
aws codeartifact login --tool npm --domain mydomain --domain-owner 111122223333 --
repository myrepo --endpoint-url VPC_endpoint
```

Sustituya *VPC_endpoint* por su punto de conexión de VPC de Amazon, con el prefijo `https://`. Vea el siguiente punto de conexión de ejemplo.

```
https://vpce-0743fe535b883ffff-76ddffff.api.codeartifact.us-
west-2.vpce.amazonaws.com
```

Si utiliza el SDK, consulte la documentación del SDK para obtener información sobre cómo anular un nombre de host. La forma de hacerlo varía según el idioma que utilice.

Utilice el punto de conexión **codeartifact.repositories** sin DNS privado

Si no puede o no quiere habilitar el DNS privado en el punto de enlace de `com.amazonaws.region.codeartifact.repositories` VPC en el que creóCree puntos finales de VPC para CodeArtifact, debe seguir estas instrucciones para configurar el administrador de paquetes con la URL correcta CodeArtifact .

1. Ejecute el siguiente comando para buscar un punto de conexión de VPC y usarlo para anular el nombre de host.

```
$ aws ec2 describe-vpc-endpoints --filters Name=service-
name,Values=com.amazonaws.region.codeartifact.repositories \
  --query 'VpcEndpoints[*].DnsEntries[*].DnsName'
```

El resultado es similar al siguiente.

```
[
  [
    "vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-west-2.vpce.amazonaws.com"
  ]
]
```

2. Actualice la ruta del punto final de la VPC para incluir el formato del paquete, el nombre de CodeArtifact dominio y el nombre del CodeArtifact repositorio. Consulte el siguiente ejemplo.

```
https://vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-
west-2.vpce.amazonaws.com/format/d/domain_name-domain_owner/repo_name
```

Sustituya los siguientes campos del punto de conexión de ejemplo.

- *formato*: sustitúyalo por un formato de CodeArtifact paquete válido, por ejemplo, `npm` o`pypi`.
- *domain_name*: Sustitúyalo por el CodeArtifact dominio que contiene el CodeArtifact repositorio que aloja tus paquetes.
- *domain_owner*: Sustitúyalo por el ID del propietario del CodeArtifact dominio, por ejemplo,. `111122223333`
- *repo_name*: Sustitúyalo por el CodeArtifact repositorio que aloja tus paquetes.

La siguiente URL es un punto de conexión de repositorio npm de ejemplo.

```
https://vpce-0dc4daf7fca331ed6-et36qa1d.d.codeartifact.us-
west-2.vpce.amazonaws.com/npm/d/domainName-111122223333/repoName
```

3. Configure el administrador de paquetes para que utilice el punto de conexión de VPC actualizado del paso anterior. Debe configurar el administrador de paquetes sin usar el comando. CodeArtifact `login` Para ver las instrucciones de configuración de cada formato de paquete, consulte la siguiente documentación.

- npm: Configurar npm sin usar el comando login

- nuget: configurar nuget o dotnet sin el comando login
- pip: Configurar pip sin el comando login
- twine: Configurar y usar twine con CodeArtifact
- Gradle: Uso de CodeArtifact con Gradle
- mvn: Usar CodeArtifact con mvn

Creación de una política de puntos de conexión de VPC para CodeArtifact

Para crear una política de puntos de conexión de VPC CodeArtifact, especifique lo siguiente:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden realizar acciones.

El siguiente ejemplo de política especifica que los directores de la cuenta 123456789012 pueden llamar a la `GetAuthorizationToken` API y obtener paquetes de un repositorio. CodeArtifact

```
{
  "Statement": [
    {
      "Action": [
        "codeartifact:GetAuthorizationToken",
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ReadFromRepository",
        "sts:GetServiceBearerToken"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Principal": {
         "AWS": "arn:aws:iam::123456789012:root"
       }
    }
  ]
}
```

Creación de recursos de CodeArtifact con AWS CloudFormation

CodeArtifact está integrado con AWS CloudFormation, un servicio que lo ayuda a modelar y configurar los recursos de AWS para que pueda dedicar menos tiempo a crear y administrar sus recursos e infraestructura. Puede crear una plantilla que describa todos los recursos de AWS que desea (como dominios o repositorios) y AWS CloudFormation se encargará del aprovisionamiento y la configuración de dichos recursos.

Cuando utiliza AWS CloudFormation, puede volver a utilizar la plantilla para configurar sus recursos de CodeArtifact de forma coherente y repetida. Solo tiene que describir los recursos una vez y, luego, aprovisionar los mismos recursos una y otra vez en varias cuentas y regiones de AWS.

CodeArtifact y plantillas AWS CloudFormation

Para aprovisionar y configurar los recursos de CodeArtifact y sus servicios relacionados, debe entender las plantillas de AWS CloudFormation. Las plantillas son archivos de texto con formato JSON o YAML. Estas plantillas describen los recursos que desea aprovisionar en sus pilas de AWS CloudFormation. Si no conoce bien JSON o YAML, puede utilizar Designer de AWS CloudFormation para comenzar a utilizar las plantillas de AWS CloudFormation. Para obtener más información, consulte ¿Qué es AWS CloudFormation Designer? en la Guía de usuario de AWS CloudFormation.

CodeArtifact admite la creación de dominios y repositorios en AWS CloudFormation. Para obtener más información, incluidos ejemplos de plantillas JSON y YAML para dominios y repositorios, consulte AWS::CodeArtifact::Domain and AWS::CodeArtifact::Repository.

Evitar la eliminación de recursos de CodeArtifact

Los repositorios de CodeArtifact contienen dependencias de aplicaciones críticas que, si se pierden, pueden que se no recreen con facilidad. Para proteger los recursos de CodeArtifact frente a la eliminación accidental al administrar los recursos de CodeArtifact con CloudFormation, incluya los atributos `DeletionPolicy` y `UpdateRetainPolicy` con un valor de `Retain` en todos los dominios y repositorios. Esto evitará que se borre si el recurso se elimina de la plantilla de pila o si se elimina accidentalmente toda la pila. El siguiente fragmento de código YAML muestra un dominio y un repositorio básicos con los siguientes atributos:

```
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
```

Para obtener más información sobre estos atributos, consulte DeletionPolicy yd UpdateReplacePolicy en la Guía del usuario de AWS CloudFormation.

Obtener más información sobre AWS CloudFormation

Para conocer más información acerca de AWS CloudFormation, consulte los siguientes recursos:

- AWS CloudFormation
- Guía del usuario de AWS CloudFormation
- Guía del usuario de la interfaz de la línea de comandos de AWS CloudFormation

Solución de problemas de AWS CodeArtifact

La siguiente información puede ayudarle a solucionar problemas comunes con CodeArtifact.

Para obtener información sobre la solución de problemas específicos del formato, consulte los siguientes temas:

- Solución de problemas de Maven
- Solución de problemas de Swift

No se pueden visualizar las notificaciones

Problema: cuando se encuentra en la consola de herramientas para desarrolladores y elige Notificaciones (Notificaciones) en la pestaña Settings (Configuración), aparecerá un error de permisos.

Posibles soluciones: las notificaciones, aunque son una característica de la consola de herramientas para desarrolladores, CodeArtifact actualmente no admite notificaciones. Ninguna de las políticas administradas de CodeArtifact incluye permisos que permitan a los usuarios ver o administrar las notificaciones. Si utiliza otros servicios de la consola de herramientas para desarrolladores y esos servicios admiten notificaciones, las políticas gestionadas de esos servicios incluyen los permisos necesarios para ver y gestionar las notificaciones de esos servicios.

Etiquetado de recursos

Una etiqueta es un atributo personalizado que usted o AWS asignan a un recurso de AWS. Cada etiqueta de AWS tiene dos partes:

- Una clave de etiqueta (por ejemplo, `CostCenter`, `Environment`, `Project` o `Secret`). Las claves de etiqueta distinguen entre mayúsculas y minúsculas.
- Un campo opcional que se denomina valor de etiqueta (por ejemplo, `111122223333` o `Production` o el nombre de un equipo). Omitir el valor de etiqueta es lo mismo que utilizar una cadena vacía. Al igual que las claves de etiqueta, los valores de etiqueta distinguen entre mayúsculas y minúsculas.

En conjunto, se conocen como pares clave-valor.

Las etiquetas le ayudan a identificar y organizar los recursos de AWS. Muchos servicios de AWS admiten el etiquetado, por lo que puede asignar la misma etiqueta a los recursos de diferentes servicios para indicar que los recursos están relacionados. Por ejemplo, puede asignar la misma etiqueta a un repositorio que se asigna a un proyecto de AWS CodeBuild.

Para obtener consejos y prácticas recomendadas sobre el uso de etiquetas, consulte el documento técnico Prácticas recomendadas para etiquetar recursos de AWS.

Puede etiquetar los siguientes tipos de recursos de CodeArtifact:

- Etiquete un repositorio en CodeArtifact
- Etiquete un dominio en CodeArtifact

Puede usar la consola AWS CLI, las API de CodeArtifact o los SDK AWS para:

- Agregue etiquetas a un dominio o repositorio cuando lo cree*.
- Agregue, administre y elimine etiquetas para un dominio o repositorio.

* No puede agregar etiquetas a un dominio o repositorio cuando lo crea en la consola.

Además de utilizar etiquetas para identificar, organizar y realizar el seguimiento de un recurso, puede utilizarlas en políticas de IAM para ayudar a controlar quién puede ver el recurso e interactuar con

él. Para ver ejemplos de políticas de acceso basadas en etiquetas, consulte Uso de etiquetas para controlar el acceso a los recursos de CodeArtifact.

Asignación de costes de CodeArtifact con etiquetas

Puede usar etiquetas para asignar los costes de almacenamiento y solicitud en CodeArtifact.

Asignación de costes de almacenamiento de datos en CodeArtifact

Los costes de almacenamiento de datos están vinculados a los dominios, por lo tanto, para asignar sus costes de almacenamiento de CodeArtifact, puede usar cualquier etiqueta que se aplique a sus dominios. Para obtener más información acerca de cómo añadir etiquetas a dominios, consulte Etiquete un dominio en CodeArtifact.

Asignación de costes de solicitud en CodeArtifact

La mayoría del uso de las solicitudes está vinculado a los repositorios, por lo tanto, para asignar los costes de las solicitudes de CodeArtifact, puede usar cualquier etiqueta que se aplique a sus repositorios. Para obtener más información acerca de cómo añadir etiquetas a repositorios, consulte Etiquete un repositorio en CodeArtifact.

Algunos tipos de solicitudes están asociados a dominios y no a repositorios, por lo que el uso de las solicitudes y los costes relacionados con las solicitudes se asignarán a las etiquetas del dominio. La mejor manera de determinar si un tipo de solicitud está asociado a un dominio o un repositorio es usar la tabla Acciones definidas por AWS CodeArtifact en la Referencia de autorización del servicio. Busque el tipo de solicitud en la columna Acciones y observe el valor en la columna de tipos de recursos correspondiente. Si el tipo de recurso es dominio, las solicitudes de ese tipo se facturarán al dominio. Si el tipo de recurso es repositorio o paquete, las solicitudes de ese tipo se facturarán al repositorio. Algunas acciones muestran ambos tipos de recursos; en el caso de esas acciones, el recurso facturado depende del valor que se transfiera a la solicitud.

Cuotas en AWS CodeArtifact

La siguiente tabla describe las cuotas de recursos en CodeArtifact. Para ver las cuotas de recursos junto con la lista de puntos de conexión de servicio de CodeArtifact, consulte las cuotas de servicio de AWS en Referencia general de Amazon Web Services.

Puede solicitar un aumento de la cuota de servicio para las siguientes cuotas de recursos de CodeArtifact. Para obtener más información acerca de cómo solicitar un aumento de cuotas de servicio, consulte AWS Service Quotas.

Nombre	Valor predeterminado	Ajusta	Descripción
Tamaño del archivo de activos	Cada región admitida: 5 gigabytes	Sí	El tamaño de archivo máximo por recurso.
Activos por versión de paquete	Cada región admitida: 150	No	El número máximo de activos por versión del paquete.
Solicitudes de CopyPackageVersions por segundo	Cada región admitida: 5	Sí	El número máximo de llamadas que pueden realizarse a CopyPackageVersions por segundo.
Flujos de acceso directos por repositorio	Cada región admitida: 10	No	El número máximo de repositorios ascendentes directos por repositorio.
Dominios por cuenta AWS	Cada región admitida: 10	Sí	El número máximo de dominios que se pueden crear por cuenta AWS.
Solicitudes de GetAuthorizationToken por segundo	Cada región admitida: 40	Sí	El número máximo de tokens de autorización recuperados por segundo.

Nombre	Valor predeterminado	Ajusta	Descripción
Solicitudes de GetPackageVersionAsset por segundo	Cada región admitida: 50	Sí	El número máximo de llamadas que pueden realizarse a GetPackageVersionAsset por segundo.
Solicitudes de ListPackageVersionAssets por segundo	Cada región admitida: 200	Sí	El número máximo de llamadas que pueden realizarse a ListPackageVersionAssets por segundo.
Solicitudes de ListPackageVersions por segundo	Cada región admitida: 200	Sí	El número máximo de llamadas que pueden realizarse a ListPackageVersions por segundo.
Solicitudes de ListPackages por segundo	Cada región admitida: 200	Sí	El número máximo de llamadas que pueden realizarse a ListPackages por segundo.
Solicitudes de PublishPackageVersion por segundo	Cada región admitida: 10	Sí	El número máximo de llamadas que pueden realizarse a PublishPackageVersion por segundo.
Leer las solicitudes por segundo desde una sola cuenta AWS	Cada región admitida: 800	Sí	El número máximo de solicitudes de lectura de una cuenta AWS por segundo.

Nombre	Valor predeterminado	Ajusta	Descripción
Repositorios por dominio	Cada región admitida: 1000	Sí	El número máximo de repositorios que se pueden crear por dominio.
Solicitudes por segundo mediante un único token de autenticación	Cada región admitida: 1200	No	El número máximo de solicitudes por segundo que utilizan un único token de autenticación.
Solicitudes sin token de autenticación por dirección IP	Cada región admitida: 600	No	El número máximo de solicitudes por segundo sin un token de autenticación desde una única dirección IP.
Se buscaron repositorios ascendentes	Cada región admitida: 25	No	El número máximo de repositorios ascendentes buscados al resolver un paquete.
Escribir solicitudes por segundo desde una sola cuenta AWS	Cada región admitida: 100	Sí	El número máximo de solicitudes de escritura de una cuenta AWS por segundo.

AWS CodeArtifact historial de documentos de la guía del usuario

En la siguiente tabla se describen los cambios importantes en la documentación de CodeArtifact.

Cambio	Descripción	Fecha
Se agregaron administradores de paquetes válidos adicionales a la documentación sobre el `aws codeartifact login` comando.	Se agregaron `dotnetnuget`, y `swift` a la lista de administradores de paquetes válidos para usar con el `aws codeartifact login` comando. Para obtener más información, consulte AWS CodeArtifact autenticación y tokens.	18 de febrero de 2024
Se agregó una entrada a la documentación de solución de problemas de Swift sobre Xcode colgado en máquinas de CI	Se ha añadido información, incluida una solución, sobre un problema que puede provocar que Xcode se bloquee en las máquinas de CI debido a que el llavero solicita la contraseña. Para obtener más información, consulte Xcode se bloquea en la máquina CI debido a que el llavero solicita la contraseña.	6 de febrero de 2024
Se ha agregado información sobre la solución de problemas de tiempos lentos de instalación de paquetes npm con npm 8.x o superior	Se agregó información sobre cómo evitar los tiempos lentos de instalación de los paquetes npm CodeArtifact, lo que podría provocar tiempos de compilación lentos. Para obtener más informaci	29 de diciembre de 2023

	ón, consulte Solución de problemas de instalaciones lentas con npm 8.x o superior.	
Información actualizada sobre el comportamiento de los metadatos y los activos del paquete de Python en CodeArtifact	Información actualizada sobre cómo CodeArtifact los repositorios conservan y actualizan los activos y metadatos de la versión del paquete Python. Para obtener más información, consulte Solicitud de paquetes de Python desde conexiones ascendentes y externas.	14 de diciembre de 2023
Documentación reorganizada sobre la supervisión CodeArtifact	Se reorganizó la información sobre CodeArtifact los eventos de monitoreo y se agregó información sobre la visualización de CodeArtifact las solicitudes con las CloudWatch métricas de Amazon. Para obtener más información, consulte Supervisión de CodeArtifact.	14 de diciembre de 2023

Se agregó más información sobre la administración de CodeArtifact recursos con AWS CloudFormation	Se agregaron referencias y enlaces a la documentación sobre cómo administrar CodeArtifact los recursos con CloudFormation, incluida una sección sobre cómo evitar la eliminación de CodeArtifact los recursos administrados con CloudFormation. Para obtener más información, consulte Evitar la eliminación de recursos de CodeArtifact.	7 de diciembre de 2023
Se agregó documentación que detalla CodeArtifact la compatibilidad con los almacenes de claves AWS KMS externos (XKS)	Se agregó una sección con información sobre CodeArtifact la compatibilidad con las claves KMS, incluido el uso de claves XKS con. CodeArtifact Para obtener más información, consulte Tipos de AWS KMS claves compatibles en CodeArtifact.	31 de octubre de 2023
Se actualizó la documentación de solución de problemas existente y se agregó una nueva	Se agregó un tema de solución de problemas de Maven e incluyó enlaces a la documentación de solución de problemas de Swift y Maven en el tema de solución de problemas general. Para obtener más información, consulte Solución de problemas de AWS CodeArtifact.	28 de septiembre de 2023

Documentación actualizada para incluir el comando de publicación de Swift Package Manager	Swift 5.9 introdujo un comando `swift package-registry publish` para crear y publicar un paquete Swift en un repositorio de paquetes. Se actualizó la documentación de Swift para incluir instrucciones para usar ese comando. Para obtener más información, consulte Uso CodeArtifact con Swift.	25 de septiembre de 2023
Se agregó documentación para configurar CodeArtifact con Swift	CodeArtifact ahora es compatible con los paquetes Swift. Se agregó documentación con instrucciones sobre cómo configurar Swift para usar CodeArtifact repositorios. Para obtener más información, consulte Uso CodeArtifact con Swift.	20 de septiembre de 2023
Se agregó una guía sobre cómo CodeArtifact maneja las versiones de paquetes de Python retiradas	Se agregó documentación con información sobre cómo saber si una versión de un paquete de Python está retirada, cómo CodeArtifact maneja las versiones de paquetes retiradas y respuestas a preguntas comunes. Para obtener más información, consulte Versiones de paquetes retiradas.	2 de agosto de 2023

Se corrigió un comando de línea de comandos incorrecto en la documentación de Yarn	Se ha corregido un comando de línea de comandos incorrecto que buscaba un token de CodeArtifact autorización y lo almacenaba en una variable de entorno en la documentación de Yarn.	20 de julio de 2023
Adiciones menores y correcciones de pequeños errores en la documentación de Python	Se agregó información sobre pip y twine en sus respectivas documentaciones y se corrigió lo que sucedía al usar el comando `codeartifact login` con twine. Para más información, consulte Configurar y usar pip con CodeArtifact y Configurar y usar twine con CodeArtifact.	14 de julio de 2023
Se corrigieron los comandos dotnet incorrectos en la documentación CodeBuild	Se corrigieron los comandos `dotnet add package` de la documentación Uso de paquetes NuGet en CodeBuild .	13 de julio de 2023
Actualización AWS CodeArtifact y documentación AWS Identity and Access Management	Se revisó el IAM en la CodeArtifact documentación para añadir claridad y coherencia a la documentación de otros servicios. AWS Consulte Identity and Access Management para AWS CodeArtifact.	24 de mayo de 2023

Se agregó información sobre las versiones de paquetes de Python retiradas	Se agregó información sobre cómo se CodeArtifact conservan los metadatos de las versiones de paquetes de Python extraídas. Para obtener más informaci ón, consulteVersiones de paquetes retiradas.	11 de abril de 2023
Se agregó información sobre la compatibilidad con Clojure	Se agregó información sobre el soporte de Clojure, incluida la administración de las dependencias para los proyectos de Clojure. Para obtener más información, consulte Usar CodeArtifact con deps.edn.	21 de marzo de 2023
Se agregó información sobre la publicación de paquetes genéricos	Se agregó información sobre los paquetes genéricos y sobre cómo publicar y descargar el contenido de los paquetes con AWS CLI. Para obtener más informaci ón, consulte Uso de CodeArtif act con paquetes genéricos, Publicar y consumir paquetes genéricos y Comandos compatibles con paquetes genéricos.	10 de marzo de 2023
Se agregó información sobre los límites de tamaño de los activos para su publicación	Se agregó una sección a la publicación de paquetes para explicar los límites de tamaño de los activos para la publicaci ón.	21 de junio de 2022

Se refactorizó la documentación de conexión externa	Se trasladó la documentación de conexión externa y se reorganizó para centrarse en el objetivo final del usuario, que es conectar su CodeArtifact repositorio a los repositorios de paquetes públicos. También se agregó más orientación e información sobre los diferentes métodos para lograr ese objetivo. Para obtener más información, consulte Conectar un repositorio CodeArtifact a un repositorio público.	9 de mayo de 2022
Se actualizó la información del CodeArtifact evento para Amazon CloudWatch Events	Se agregó más información al campo `account` y se agregó el campo `repositoryAdministrator` . Para obtener más información, consulte Formato y ejemplo de evento CodeArtifact.	7 de marzo de 2022

Se agregaron instrucciones de configuración para su uso CodeArtifact desde una VPC sin DNS privado	Si no puede o no quiere habilitar el DNS privado en su punto de enlace de `codeartifact.repositories` VPC, debe usar una configuración diferente para el punto de enlace de los repositorios para usarlo desde CodeArtifact una VPC. Para obtener más información, consulte Utilice el punto de conexión `codeartifact.repositories` sin DNS privado.	8 de febrero de 2022
Se agregó documentación detallada para actualizar el estado de las versiones de los paquetes	Se amplió la documentación sobre el estado de la versión del paquete de actualización para incluirla en su propio tema. Se agregó documentación para actualizar el estado de la versión de un paquete, incluidos los permisos de IAM necesarios, ejemplos de AWS CLI comandos para varios escenarios y posibles errores. Para obtener más información, consulte Actualización del estado de la versión del paquete.	1 de septiembre de 2021

Se actualizó la documentación de las versiones de los paquetes copiados con información más detallada sobre los permisos	Se agregó más información sobre los permisos de IAM y de políticas basadas en recursos necesarios para `aws codeartifact copy-package-versions` ejecutar el comando y copiar versiones de paquetes de un repositorio a otro dentro del mismo dominio en. CodeArtifact Junto con más información, ahora hay ejemplos de las políticas basadas en recursos requeridas para el repositorio de origen y destino. Para obtener más información, consulte Permisos de IAM necesarios para copiar paquetes.	25 de agosto de 2021

Documentación actualizada para ejecutar una compilación de Gradle en IntelliJ IDEA	Se actualizó la documentación para ejecutar una compilación de Gradle en IntelliJ IDEA con los pasos para configurar Gradle para obtener complementos. CodeArtifact También se agregó una opción para crear un nuevo token de CodeArtifact autorización para cada nueva ejecución con una llamada en línea a. `aws codeartifact get-authorization-token` Para obtener más información, consulte Ejecutar una compilación de Gradle en IntelliJ IDEA.	23 de agosto de 2021
Se agregó documentación para configurar y usar Yarn con AWS CodeArtifact	Se agregó documentación para configurar y usar Yarn 1.X y Yarn 2.X para administrar paquetes npm. CodeArtifact Para obtener más información, consulte Configurar y usar Yarn con CodeArtifact.	30 de julio de 2021

AWS CodeArtifact ahora admite paquetes NuGet	CodeArtifact los usuarios ahora pueden publicar y consumir NuGet paquetes. Se agregó documentación para configurar y usar tanto Visual Studio como herramientas de línea de NuGet comandos, como `nuget` y `dotnet` con CodeArtifact los repositorios. Para obtener más información, consulte Uso de CodeArtifact con NuGet.	19 de noviembre de 2020
Etiquetado de recursos en AWS CodeArtifact	Se agregó documentación sobre el etiquetado de repositorios y dominios en. AWS CodeArtifact Consulte Etiquetado de recursos.	30 de octubre de 2020
CodeArtifact ahora es compatible AWS CloudFormation	CodeArtifact los usuarios ahora pueden usar AWS CloudFormation plantillas para crear CodeArtifact repositorios y dominios. Consulte Creación de recursos de CodeArtifact con AWS CloudFormation para obtener más información y comenzar.	8 de octubre de 2020

Añada información sobre la creación de puntos de enlace de puerta de enlace de Amazon S3 para utilizarlos CodeArtifact con Amazon VPC	Se agregó información sobre la creación de puntos de enlace de puerta de enlace de Amazon S3 con el comando Amazon AWS CLI EC2. Esta documentación también contiene información sobre los permisos específicos que se CodeArtifact deben utilizar con los entornos de Amazon VPC. Consulte Creación del punto de enlace de gateway de Amazon S3.	12 de agosto de 2020
Publicar artefactos de Maven con curl y publicar artefactos de Maven de terceros	Se agregó una guía para Publicación con curl y Publicación de artefactos de terceros.	10 de agosto de 2020
Versión de disponibilidad general (GA)	Versión inicial de la Guía del CodeArtifact usuario.	10 de junio de 2020

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.