aws

CodeArtifact Guia do usuário

CodeArtifact

CodeArtifact: CodeArtifact Guia do usuário

Table of Contents

O que é AWS CodeArtifact?

AWS CodeArtifact é um serviço de repositório de artefatos seguro, altamente escalável e gerenciado que ajuda as organizações a armazenar e compartilhar pacotes de software para desenvolvimento de aplicativos. Você pode usar CodeArtifact com ferramentas de compilação e gerenciadores de pacotes populares, como NuGet CLI, Maven, Gradle, npm, yarn, pip e twine. CodeArtifact ajuda a reduzir a necessidade de gerenciar seu próprio sistema de armazenamento de artefatos ou de se preocupar com a escalabilidade de sua infraestrutura. Não há limites no número ou no tamanho total dos pacotes que você pode armazenar em um CodeArtifact repositório.

Você pode criar uma conexão entre seu CodeArtifact repositório privado e um repositório público externo, como npmjs.com ou Maven Central. CodeArtifact em seguida, buscará e armazenará pacotes sob demanda do repositório público quando eles forem solicitados por um gerenciador de pacotes. Isso torna o consumo de dependências de código aberto usadas pelo aplicativo mais conveniente e ajuda a garantir que elas estejam sempre disponíveis para compilações e desenvolvimento. Você também pode publicar pacotes privados em um CodeArtifact repositório. Isso ajuda a compartilhar componentes de software proprietários entre vários aplicativos e equipes de desenvolvimento da organização.

Para obter mais informações, consulte AWS CodeArtifact.

Como CodeArtifact funciona?

CodeArtifact armazena pacotes de software em repositórios. Os repositórios são poliglotas, ou seja, um único repositório pode conter pacotes de qualquer tipo compatível. Cada CodeArtifact repositório é membro de um único CodeArtifact domínio. Recomendamos que você use um domínio de produção para a organização com um ou mais repositórios. Por exemplo, você pode usar cada repositório para uma equipe de desenvolvimento diferente. Dessa forma, os pacotes nos repositórios podem ser localizados e compartilhados entre as equipes de desenvolvimento.

Para adicionar pacotes a um repositório, configure um gerenciador de pacotes, como o npm ou Maven, para usar o endpoint (URL) do repositório. Em seguida, você pode usar o gerenciador de pacotes para publicar pacotes no repositório. Você também pode importar pacotes de código aberto para um repositório configurando-o com uma conexão externa a um repositório público, como npmjs, Gallery NuGet , Maven Central ou PyPI. Para obter mais informações, consulte Conectar um CodeArtifact repositório a um repositório público.

Você pode disponibilizar pacotes de um repositório para outro no mesmo domínio. Para fazer isso, configure um repositório como upstream do outro. Todas as versões do pacote disponíveis para o repositório upstream também estão disponíveis para o repositório downstream. Além disso, todos os pacotes que estão disponíveis para o repositório upstream por meio de uma conexão externa com um repositório público estão disponíveis para o repositório downstream. Para obter mais informações, consulte Trabalhando com repositórios upstream em CodeArtifact.

CodeArtifact exige que os usuários se autentiquem com o serviço para publicar ou consumir versões do pacote. Você deve se autenticar no CodeArtifact serviço criando um token de autorização usando suas AWS credenciais. Pacotes em CodeArtifact repositórios não podem ser disponibilizados publicamente. Para obter mais informações sobre autenticação e acesso em CodeArtifact, consulteAWS Autenticação e tokens do CodeArtifact.

AWS CodeArtifact conceitos

Aqui estão alguns conceitos e termos que você deve conhecer ao usar CodeArtifact.

Tópicos

- Ativo
- Domínio
- Repositório
- Pacote
- Grupo de pacotes
- Namespace de pacotes
- Versão do pacote
- Revisão da versão do pacote
- Repositório upstream

Ativo

Um ativo é um arquivo individual armazenado associado CodeArtifact a uma versão do pacote, como um `.tgz` arquivo npm ou arquivos Maven POM e JAR.

Domínio

Os repositórios são agrupados em uma entidade de nível mais alto conhecida como domínio. Todos os ativos e metadados do pacote são armazenados no domínio, mas são consumidos por meio de repositórios. Um determinado ativo de pacote, como um arquivo JAR do Maven, é armazenado uma vez por domínio, não importa em quantos repositórios ele esteja presente. Todos os ativos e metadados em um domínio são criptografados com a mesma AWS KMS key (chave KMS) armazenada em AWS Key Management Service (AWS KMS).

Cada repositório é membro de um único domínio e não pode ser movido para um domínio diferente.

Usando um domínio, você pode aplicar uma política organizacional em vários repositórios. Com essa abordagem, você determina quais contas podem acessar os repositórios no domínio e quais repositórios públicos podem ser usados como origens dos pacotes.

Embora uma organização possa ter vários domínios, a recomendação é ter um único domínio de produção que contenha todos os artefatos publicados. Dessa forma, as equipes podem localizar e compartilhar pacotes em toda a organização.

Repositório

Um CodeArtifact repositório contém um conjunto de versões de pacotes, cada uma delas mapeada para um conjunto de ativos. Os repositórios são poliglotas, ou seja, um único repositório pode conter pacotes de qualquer tipo compatível. Cada repositório expõe endpoints para buscar e publicar pacotes usando ferramentas como a CLI do NuGet, a CLI do npm, a CLI do Maven (mvn) e o pip. Você pode criar até 1.000 repositórios por domínio.

Pacote

Um pacote é formado pelo pacote de software e os metadados necessários para resolver dependências e instalar o software. Em CodeArtifact, um pacote consiste em um nome de pacote, um namespace opcional, como `@types` in`@types/node`, um conjunto de versões de pacote e metadados em nível de pacote, como tags npm.

AWS CodeArtifact suporta os formatos de pacote Cargo, genérico, Maven, npm, NuGetPyPI, Ruby, Swift.

Grupo de pacotes

Os grupos de pacotes podem ser usados para aplicar configurações a vários pacotes que correspondam a um padrão definido usando o formato, o namespace e o nome do pacote. Você pode usar grupos de pacotes para configurar de forma mais conveniente os controles de origem de pacotes para vários pacotes. Os controles de origem de pacotes são usados para bloquear ou permitir a ingestão ou publicação de novas versões de pacotes, protegendo os usuários de ações maliciosas conhecidas como ataques de substituição de dependências.

Namespace de pacotes

Alguns formatos de pacotes permitem a utilização de nomes de pacotes hierárquicos para organizar os pacotes em grupos lógicos e ajudar a evitar colisões de nomes. Por exemplo, o npm é compatível com escopos. Para obter mais informações, consulte a documentação de escopos do npm. O pacote `@types/node` do npm tem o escopo `@types` e o nome `node`. Há vários outros nomes de pacotes no escopo `@types`. Em CodeArtifact, o escopo (“tipos”) é chamado de namespace do pacote e o nome (“nó”) é chamado de nome do pacote. Para pacotes do Maven, o namespace do pacote corresponde ao groupID do Maven. O pacote `org.apache.logging.log4j:log4j` do Maven tem o groupID (namespace do pacote) `org.apache.logging.log4j` e o artifactID (nome do pacote) `log4j`. Para pacotes genéricos, o namespace é obrigatório. Alguns formatos de pacotes, como do PyPI, não permitem a utilização de nomes hierárquicos com um conceito semelhante ao escopo do npm ou ao groupID do Maven. Sem uma forma de agrupar nomes de pacotes, evitar colisões de nomes pode ser mais difícil.

Versão do pacote

A versão do pacote identifica a versão específica de um pacote, como `@types/node 12.6.9`. O formato e a semântica do número da versão variam conforme os diferentes formatos de pacote. Por exemplo, as versões dos pacotes do npm devem estar em conformidade com a especificação de Versionamento semântico. Em CodeArtifact, uma versão do pacote consiste no identificador da versão, nos metadados do nível da versão do pacote e em um conjunto de ativos.

Revisão da versão do pacote

A revisão da versão do pacote é uma string que identifica um conjunto específico de ativos e metadados da versão do pacote. Sempre que uma versão do pacote é atualizada, uma nova revisão da versão do pacote é criada. Por exemplo, você pode publicar um arquivo de distribuição de origem (sdist) de uma versão do pacote do Python e, posteriormente, adicionar uma wheel do Python que

contém código compilado à mesma versão. Quando você publica a wheel, uma nova revisão da versão do pacote é criada.

Repositório upstream

Um repositório é upstream quando as versões do pacote contidas nele podem ser acessadas a partir do endpoint do repositório downstream. Esta abordagem mescla com eficácia o conteúdo dos dois repositórios do ponto de vista de um cliente. Usando CodeArtifact, você pode criar um relacionamento upstream entre dois repositórios.

Como faço para começar com CodeArtifact?

É recomendável que você realize as etapas a seguir:

1. Saiba mais CodeArtifact lendoAWS CodeArtifact conceitos.
2. Configure seu Conta da AWS AWS CLI, o e um usuário do IAM seguindo as etapas emConfigurando com AWS CodeArtifact.
3. Use CodeArtifact seguindo as instruções emConceitos básicos do CodeArtifact.

Configurando com AWS CodeArtifact

Se já tiver se cadastrado na Amazon Web Services (AWS), você poderá começar a usar o AWS CodeArtifact imediatamente. Você pode abrir o CodeArtifact console, escolher Criar um domínio e repositório e seguir as etapas no assistente de inicialização para criar seu primeiro domínio e repositório.

Se você ainda não se inscreveu ou precisa de ajuda para AWS criar seu primeiro domínio e repositório, conclua as seguintes tarefas para se preparar para usar CodeArtifact:

Tópicos

Inscreva-se para um Conta da AWS

Para começar AWS, você precisa de um Conta da AWS. Para obter informações sobre como criar um Conta da AWS, consulte Introdução a um Conta da AWS no Guia de AWS Gerenciamento de contas referência.

Instale ou atualize e, em seguida, configure o AWS CLI

Para chamar CodeArtifact comandos do AWS Command Line Interface (AWS CLI) em uma máquina de desenvolvimento local, você deve instalar AWS CLI o.

Se você tiver uma versão mais antiga do AWS CLI instalado, deverá atualizá-la para que os CodeArtifact comandos estejam disponíveis. CodeArtifact os comandos estão disponíveis nas seguintes AWS CLI versões:

1. AWS CLI 1: 1.18.77 e versões mais recentes
2. AWS CLI 2: 2.0.21 e versões mais recentes

Para verificar a versão, use o comando `aws --version`.

Para instalar e configurar o AWS CLI

1. Instale ou atualize o AWS CLI com as instruções em Instalando AWS Command Line Interface o.
2. Configure o AWS CLI, com o comando configure, da seguinte forma.

   ```
   aws configure
   ```

   Quando solicitado, especifique a chave de AWS acesso e a chave de acesso AWS secreta do usuário do IAM com CodeArtifact o qual você usará. Quando o nome da região padrão Região da AWS for solicitado, especifique a região onde você criará o pipeline, por exemplo, `us-east-2`. Quando solicitado pelo formato de saída padrão, especifique `json`.

   > **Important**
   >
   > Ao configurar o AWS CLI, você será solicitado a especificar um Região da AWS. Escolha uma das regiões com suporte listada em Região e endpoints na Referência geral da AWS.

   Para obter mais informações, consulte Configurar a AWS Command Line Interface e Gerenciamento de chaves de acesso para usuários do IAM.
3. Para verificar a instalação ou atualização, chame o seguinte comando na AWS CLI.

   ```
   aws codeartifact help
   ```

   Se for bem-sucedido, esse comando exibirá uma lista dos CodeArtifact comandos disponíveis.

Em seguida, você pode criar um usuário do IAM e conceder acesso a esse usuário CodeArtifact a. Para obter mais informações, consulte Provisionar um usuário do IAM.

Provisionar um usuário do IAM

Siga estas instruções para preparar um usuário do IAM para usar CodeArtifact.

Para provisionar um usuário do IAM

1. Crie um usuário do IAM ou use um associado à sua conta da Conta da AWS. Para obter mais informações, consulte Criação de um usuário do IAM e Visão geral das políticas AWS do IAM no Guia do usuário do IAM.
2. Conceda ao usuário do IAM acesso CodeArtifact a.

   - Opção 1: criar uma política do IAM personalizada. Com uma política do IAM personalizada, você pode fornecer as permissões mínimas necessárias e alterar a duração dos tokens de autenticação. Para obter mais informações e políticas de exemplo, consulte Exemplos de políticas baseadas em identidade para AWS CodeArtifact.
   - Opção 2: Usar a política `AWSCodeArtifactAdminAccess` AWS gerenciada. O trecho a seguir mostra o conteúdo dessa política.

     > **Important**
     >
     > Essa política concede acesso a todas as CodeArtifact APIs. Recomendamos que você sempre use as permissões mínimas necessárias para realizar sua tarefa. Para obter mais informações, consulte Práticas recomendadas do IAM no Guia do usuário do IAM.

     JSON

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

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

> **Note**
>
> A permissão `sts:GetServiceBearerToken` deve ser adicionada à política de usuário ou perfil do IAM. Embora possa ser adicionada a uma política de recursos de CodeArtifact domínio ou repositório, a permissão não terá efeito nas políticas de recursos.

A `sts:GetServiceBearerToken` permissão é necessária para chamar a CodeArtifact `GetAuthorizationToken` API. Essa API retorna um token que deve ser usado ao usar um gerenciador de pacotes, como `npm` ou `pip` com CodeArtifact. Para usar um gerenciador de pacotes com um CodeArtifact repositório, seu usuário ou função do IAM deve permitir, `sts:GetServiceBearerToken` conforme mostrado no exemplo de política anterior.

Se você não instalou o gerenciador de pacotes ou a ferramenta de compilação com a qual planeja usar CodeArtifact, consulteInstalar o gerenciador de pacotes ou ferramenta de compilação.

Instalar o gerenciador de pacotes ou ferramenta de compilação

Para publicar ou consumir pacotes do CodeArtifact, você deve usar um gerenciador de pacotes. Há gerenciadores de pacotes diferentes para cada tipo de pacote. A lista a seguir contém alguns gerenciadores de pacotes com os quais você pode usar CodeArtifact. Caso ainda não tenha feito, instale os gerenciadores de pacotes para o tipo de pacote que você deseja usar.

- Para npm, use a CLI do npm ou o pnpm.
- Para o Maven, use o Apache Maven (`mvn`) ou o Gradle.
- Para Python, use pip para instalar pacotes e twine para publicar pacotes.
- Para NuGet, use o Toolkit for Visual Studio no Visual Studio ou as CLIs nuget ou dotnet.
- Para pacotes genéricos, use o AWS CLI ou SDK para publicar e baixar o conteúdo do pacote.

Próximas etapas

Suas próximas etapas dependerão do tipo ou tipos de pacote com CodeArtifact os quais você está usando e do estado dos seus CodeArtifact recursos.

Se você está começando CodeArtifact a usar pela primeira vez para si mesmo, sua equipe ou organização, consulte a documentação a seguir para obter informações gerais sobre como começar e ajudar a criar os recursos necessários.

- Conceitos básicos que usam o console
- Conceitos básicos do uso da AWS CLI

Se seus recursos já tiverem sido criados e você estiver pronto para configurar seu gerenciador de pacotes para enviar pacotes para ou instalar pacotes de um CodeArtifact repositório, consulte a documentação que corresponde ao seu tipo de pacote ou gerenciador de pacotes.

- Usando o CodeArtifact com npm
- Usando CodeArtifact com Python
- Usando o CodeArtifact com Maven
- Usando o CodeArtifact com NuGet
- Usando CodeArtifact com pacotes genéricos

Conceitos básicos do CodeArtifact

Neste tutorial de conceitos básicos, você usará o CodeArtifact para criar o seguinte:

- Um domínio chamado `my-domain`.
- Um repositório chamado `my-repo` contido em `my-domain`.
- Um repositório chamado `npm-store` contido em `my-domain`. O `npm-store` tem uma conexão externa com o repositório público npm. Essa conexão é usada para ingerir um pacote npm no repositório `my-repo`.

Antes de iniciar este tutorial, recomendamos que você revise o AWS CodeArtifact conceitos CodeArtifact.

> **Note**
>
> Este tutorial requer que você crie recursos que podem resultar em cobranças na sua conta da AWS. Para obter mais informações, consulte Preços do CodeArtifact.

Tópicos

Pré-requisitos

Você pode concluir este tutorial usando o Console de gerenciamento da AWS ou a AWS Command Line Interface (AWS CLI). Para seguir este tutorial, primeiro você precisará atender aos seguintes pré-requisitos:

- Siga as etapas em Configurando com AWS CodeArtifact.
- Instale a CLI do npm. Para obter mais informações, consulte Download e instalação do Node.js e npm, na documentação do npm.

Conceitos básicos que usam o console

Execute as etapas a seguir para começar com o CodeArtifact usando o Console de gerenciamento da AWS. Este guia usa o gerenciador de pacotes `npm`. Se você estiver usando um gerenciador de pacotes diferente, será necessário adaptar algumas das etapas a seguir.

1. Faça login no Console de gerenciamento da AWS e abra o console do AWS CodeArtifact em https://console.aws.amazon.com/codesuite/codeartifact/start. Para obter mais informações, consulte Configurando com AWS CodeArtifact.
2. Escolha Criar repositório.
3. Em Nome do repositório, digite **`my-repo`**.
4. (Opcional) Em Descrição do repositório, insira uma descrição opcional para esse repositório.
5. Em Publicar repositórios upstream, selecione npm-store para criar um repositório conectado ao npmjs que seja upstream do repositório `my-repo`.

   O CodeArtifact atribui o nome `npm-store` a esse repositório para você. Todos os pacotes disponíveis para o repositório upstream `npm-store` também estão disponíveis para o repositório downstream, `my-repo`.
6. Escolha Próximo.
7. Na conta da AWS, escolha Esta conta da AWS.
8. Em Nome do domínio, insira **`my-domain`**.
9. Expanda Configuração Adicional.
10. Você deve usar uma AWS KMS key (chave KMS) para criptografar todos os ativos em seu domínio. Você pode usar uma Chave gerenciada pela AWS ou uma chave KMS que você gerencia:

    - Escolha Chave gerenciada pela AWS se quiser usar a Chave gerenciada pela AWS padrão.
    - Escolha Chave gerenciada pelo cliente se quiser usar uma chave KMS que você gerencia. Para usar uma chave KMS que você gerencia, em ARN da chave gerenciada pelo cliente, pesquise e escolha a chave KMS.

    Para obter mais informações, consulte Chave gerenciada pela AWS e Chave gerenciada pelo cliente no Guia do desenvolvedor do AWS Key Management Service.
11. Escolha Próximo.
12. Em Analisar e criar, analise o que CodeArtifact está criando para você.

- Fluxo do pacote mostra como `my-domain`, `my-repo` e `npm-store` estão relacionados.
- A Etapa 1: Criar repositório mostra detalhes sobre `my-repo` e `npm-store`.
- Etapa 2: Selecionar domínio mostra detalhes sobre `my-domain`.

Quando estiver pronto, escolha Criar repositório.

13. Na página my-repo, escolha Visualizar instruções de conexão e, em seguida, escolha npm.
14. Use o AWS CLI para executar o comando `login` mostrado em Configurar seu cliente npm usando esse comando do AWS CLI CodeArtifact.

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

Você deve receber uma confirmação de que o login foi bem-sucedido.

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

Se você receber o erro `Could not connect to the endpoint URL`, verifique se AWS CLI está configurado e se o Nome da região padrão está definido como a mesma região em que o repositório foi criado. Consulte Configurar a interface de linha de comando da AWS.

Para obter mais informações, consulte . Configure e use o npm com CodeArtifact

15. Use a CLI do npm para instalar o pacote npm. Por exemplo, para instalar o popular pacote npm `lodash`, use o seguinte comando.

```
npm install lodash
```

16. Retorne ao console do CodeArtifact. Se seu repositório my-repo estiver aberto, atualize a página. Caso contrário, no painel de navegação, escolha Repositórios e, em seguida, escolha my-repo.

Em Pacotes, você deve ver a biblioteca ou o pacote npm instalado. Você pode escolher o nome do pacote para ver sua versão e status. Você pode escolher a versão mais recente para ver os detalhes do pacote, como dependências, ativos e muito mais.

> **Note**
>
> Pode haver um atraso entre a instalação do pacote e a ingestão dele no repositório.

17. Para evitar cobranças adicionais da AWS, exclua os recursos utilizados durante este tutorial:

> **Note**
>
> Você não pode excluir um domínio que contenha repositórios, então você deve excluir `my-repo` e `npm-store` antes de excluir `my-domain`.

    a. No painel de navegação, escolha Repositórios.
    b. Escolha npm-store, Excluir e siga as etapas para excluir o repositório.
    c. Escolha my-repo, Excluir e siga as etapas para excluir o repositório.
    d. No painel de navegação, escolha Domínios.
    e. Escolha my-domain, Excluir e siga as etapas para excluir o domínio.

Conceitos básicos do uso da AWS CLI

Execute as etapas a seguir para começar com o CodeArtifact usando a AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Instale ou atualize e, em seguida, configure o AWS CLI. Este guia usa o gerenciador de pacotes `npm`. Se você estiver usando um gerenciador de pacotes diferente, será necessário adaptar algumas das etapas a seguir.

1. Use a AWS CLI para executar o comando create-domain.

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

    Os dados formatados em JSON aparecem no resultado com detalhes sobre seu novo domínio.

```
{
    "domain": {
        "name": "my-domain",
        "owner": "111122223333",
        "arn": "arn:aws:codeartifact:us-west-2:111122223333:domain/my-domain",
```

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

Se você receber o erro `Could not connect to the endpoint URL`, verifique se AWS CLI está configurado e se o Nome da região padrão está definido como a mesma região em que o repositório foi criado. Consulte Configurar a interface de linha de comando da AWS.

2. Use o comando create-repository para criar um repositório no seu domínio.

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

Os dados formatados em JSON aparecem no resultado com detalhes sobre seu novo repositório.

```
{
    "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. Use o comando create-repository para criar um repositório upstream para seu repositório `my-repo`.

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

Os dados formatados em JSON aparecem no resultado com detalhes sobre seu novo repositório.

```
{
    "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 o comando associate-external-connection para adicionar uma conexão externa com o repositório público npm ao seu repositório `npm-store`.

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

Os dados formatados em JSON aparecem no resultado com detalhes sobre o repositório e a nova conexão externa dele.

```
{
    "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 obter mais informações, consulte Conectar um CodeArtifact repositório a um repositório público.

5. Use o comando update-repository para associar o repositório `npm-store` como um repositório upstream ao repositório `my-repo`.

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

Os dados formatados em JSON aparecem no resultado com detalhes sobre o repositório atualizado, incluindo o novo repositório upstream.

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

Para obter mais informações, consulte Adicionar ou remover repositórios upstream (AWS CLI).

6. Use o comando login para configurar o gerenciador de pacotes npm com o repositório `my-repo`.

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

Você deve receber uma confirmação de que o login foi bem-sucedido.

```
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 obter mais informações, consulte Configure e use o npm com CodeArtifact.

7. Use a CLI do npm para instalar o pacote npm. Por exemplo, para instalar o popular pacote npm `lodash`, use o seguinte comando.

```
npm install lodash
```

8. Use o comando list-packages para visualizar o pacote que você acabou de instalar no repositório `my-repo`.

> **Note**
>
> Pode haver um atraso entre a conclusão do comando `npm install` de instalação e o momento em que o pacote aparece no seu repositório. Para obter detalhes sobre essa latência típica ao buscar pacotes de repositórios públicos, consulte. Latência da conexão externa

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

Os dados formatados em JSON aparecem no resultado com o formato e o nome do pacote instalado.

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

Agora você tem três recursos do CodeArtifact:

- O domínio `my-domain`.
- Um repositório `my-repo` contido em `my-domain`. Esse repositório tem um pacote npm disponível para ele.

- Um repositório `npm-store` contido em `my-domain`. Esse repositório tem uma conexão externa com o repositório npm público e está associado como um repositório upstream ao repositório `my-repo`.

9. Para evitar cobranças adicionais da AWS, exclua os recursos utilizados durante este tutorial:

> **Note**
>
> Você não pode excluir um domínio que contenha repositórios, então você deve excluir `my-repo` e `npm-store` antes de excluir `my-domain`.

a. Use o comando delete-repository para excluir o repositório `npm-store`.

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

Os dados formatados em JSON aparecem no resultado com detalhes sobre o repositório excluído.

```
{
    "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. Use o comando delete-repository para excluir o repositório `npm-store`.

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

Os dados formatados em JSON aparecem no resultado com detalhes sobre o repositório excluído.

```
{
    "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. Use o comando delete-domain para excluir o repositório `my-domain`.

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

Os dados formatados em JSON aparecem no resultado com detalhes sobre o domínio excluído.

```
{
    "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
    }
}
```

Trabalhando com repositórios em CodeArtifact

Esses tópicos mostram como usar o CodeArtifact console e as CodeArtifact APIs para criar, listar, atualizar e excluir repositórios. AWS CLI

Tópicos

Criar um repositório

Como todos os pacotes CodeArtifact são armazenados em repositórios, para usar CodeArtifact, você deve criar um. Você pode criar um repositório usando o CodeArtifact console, o AWS Command Line Interface (AWS CLI) ou CloudFormation. Cada repositório está associado à AWS conta que você usa ao criá-lo. Você pode ter vários repositórios e eles são criados e agrupados em domínios. Quando você cria um repositório, ele não contém nenhum pacote. Os repositórios são poliglotas, o que significa que um único repositório pode conter pacotes de qualquer tipo compatível.

Para obter informações sobre limites de CodeArtifact serviço, como o número máximo de repositórios permitidos em um único domínio, consulteCotas em AWS CodeArtifact. Se você atingir o número máximo de repositórios permitidos, poderá excluir repositórios para abrir espaço para mais.

Um repositório pode ter um ou mais CodeArtifact repositórios associados a ele como repositórios upstream. Isso permite que um cliente gerenciador de pacotes acesse os pacotes contidos em mais de um repositório usando um único endpoint de URL. Para obter mais informações, consulte Trabalhando com repositórios upstream em CodeArtifact.

Para obter mais informações sobre como gerenciar CodeArtifact repositórios com CloudFormation, consulteCriação de recursos do CodeArtifact com AWS CloudFormation.

> **Note**
>
> Depois de criar um repositório, não é possível alterar o nome, a conta AWS associada ou o domínio dele.

Tópicos

Criar um repositório (console)

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, escolha Repositórios e, em seguida, escolha Criar repositório.
3. Para Nome do repositório, insira um nome para o repositório.
4. (Opcional) Em Descrição do repositório, insira uma descrição opcional para esse repositório.
5. (Opcional) Em Publicar repositórios upstream, adicione repositórios intermediários que conectem seus repositórios às autoridades de pacotes, como Maven Central ou npmjs.com.
6. Escolha Próximo.
7. Na conta da AWS, escolha Esta conta da AWS se você estiver conectado à conta que possui o domínio. Escolha Conta diferente da AWS se outra conta da AWS for proprietária do domínio.
8. Em Domínio, escolha o domínio no qual o repositório será criado.

   Se não houver domínios na conta, você deve criar um. Em Nome do domínio, insira o nome do novo domínio.

   Expanda Configuração Adicional.

   Você deve usar uma AWS KMS key (chave KMS) para criptografar todos os ativos em seu domínio. Você pode usar uma Chave gerenciada pela AWS ou uma chave KMS que você gerencia:

> **Important**
>
> CodeArtifact só oferece suporte a chaves KMS simétricas. Você não pode usar uma chave KMS assimétrica para criptografar seus domínios. CodeArtifact Para obter ajuda para determinar se uma chave do KMS é simétrica ou assimétrica, consulte Identifying symmetric and asymmetric KMS keys.

- Escolha Chave gerenciada pela AWS se quiser usar a Chave gerenciada pela AWS padrão.
- Escolha Chave gerenciada pelo cliente se quiser usar uma chave KMS que você gerencia. Para usar uma chave KMS que você gerencia, em ARN da chave gerenciada pelo cliente, pesquise e escolha a chave KMS.

Para obter mais informações, consulte Chaves gerenciadas pela AWS e Chave gerenciada pelo cliente no Guia do desenvolvedor do AWS Key Management Service .

9. Escolha Próximo.
10. Em Revisar e criar, revise o que CodeArtifact está criando para você.

- Fluxo do pacote mostra como domínio e repositórios estão conectados.
- Etapa 1: Criar repositório mostra detalhes sobre o repositório e os repositórios upstream opcionais que serão criados.
- Etapa 2: Selecionar domínio mostra detalhes sobre `my_domain`.

Quando estiver pronto, escolha Criar repositório.

Crie um repositório (AWS CLI)

Use o comando `create-repository` para criar um repositório no seu domínio.

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

Resultado do exemplo:

```
{
```

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

Um novo repositório não contém nenhum pacote. Cada repositório está associado à conta AWS na qual você está autenticado quando o repositório é criado.

Criar um repositório com tags

Para criar um repositório com tags, adicione o parâmetro `--tags` ao seu 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
```

Criar um repositório com um repositório upstream

Você pode especificar um ou mais repositórios upstream ao criar um repositório.

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

Resultado do exemplo:

```
{
    "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 criar um repositório com um upstream, você deve ter permissão para a ação `AssociateWithDownstreamRepository` no repositório upstream.

Para adicionar um upstream a um repositório após sua criação, consulte Adicionar ou remover repositórios upstream (console) e Adicionar ou remover repositórios upstream (AWS CLI).

Conexão a um repositório

Depois de configurar seu perfil e suas credenciais para se autenticar em sua AWS conta, decida em qual repositório usar. CodeArtifact Você tem as seguintes opções:

- Criar um repositório. Para obter mais informações, consulte Criar um repositório.
- Use um repositório que já exista em sua conta. Você pode usar o comando `list-repositories` para localizar os repositórios criados na conta AWS . Para obter mais informações, consulte Listar repositórios.
- Use um repositório em uma AWS conta diferente. Para obter mais informações, consulte Políticas de repositórios.

Use um cliente gerenciador de pacotes

Depois de saber qual repositório você deseja usar, consulte um dos tópicos a seguir.

- Usando CodeArtifact com o Maven

- Usando CodeArtifact com npm
- Usando CodeArtifact com NuGet
- Usando CodeArtifact com Python

Excluir um repositório

Você pode excluir um repositório usando o CodeArtifact console ou o. AWS CLI Depois que um repositório for excluído, você não poderá mais enviar pacotes para ele ou extrair pacotes dele. Todos os pacotes no repositório ficam permanentemente indisponíveis e não podem ser restaurados. Você pode criar um repositório com o mesmo nome, mas seu conteúdo estará vazio.

> **Important**
>
> A exclusão de um repositório não pode ser desfeita. Depois de excluir um repositório, não será mais possível recuperá-lo e ele não poderá ser restaurado.

Tópicos

- Excluir um repositório (console)
- Excluir um repositório (AWS CLI)
- Proteger repositórios contra exclusão

Excluir um repositório (console)

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, escolha Repositórios e escolha o repositório que você deseja excluir.
3. Escolha Excluir e siga as etapas para excluir o domínio.

Excluir um repositório (AWS CLI)

Use o comando `delete-repository` para excluir um repositório.

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

Resultado do exemplo:

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

Proteger repositórios contra exclusão

Você pode evitar que um repositório seja excluído acidentalmente incluindo uma política de domínio semelhante à seguinte:

JSON

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

Essa política impede que todas as entidades principais excluam o repositório, mas se em outro momento você decidir que precisa excluir o repositório, siga estas etapas:

1. Na política de domínio, atualize a política para o seguinte:

   JSON

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

   *repository-arn*Substitua pelo ARN do repositório que você gostaria de excluir.

2. No AWS CodeArtifact console, escolha Repositórios e exclua o repositório escolhido.

3. Depois de excluir o repositório, é possível alterar a política novamente para evitar exclusões acidentais.

   JSON

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

```
}
```

Como alternativa, você pode incluir a mesma declaração de negação em uma política de repositório. Isso permite que você tenha mais flexibilidade para proteger repositórios de alto valor contra exclusão.

Listar repositórios

Use os comandos neste tópico para listar repositórios em uma AWS conta ou domínio.

Listar repositórios em um AWS account

Use esse comando para listar todos os repositórios em sua AWS conta.

```
aws codeartifact list-repositories
```

Exemplo de saída:

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

Você pode paginar a resposta de `list-repositories` usando os parâmetros `--max-results` e `--next-token`. Para `--max-results`, especifique um número inteiro de 1 a 1000 para especificar o número de resultados retornados em uma única página. Ele assume 50 como padrão. Para retornar as páginas subsequentes, execute `list-repositories` outra vez e passe o valor `nextToken` recebido na saída do comando anterior para `--next-token`. Quando a opção `--next-token` não é usada, a primeira página de resultados sempre é retornada.

Listar repositórios no domínio

Use `list-repositories-in-domain` para obter uma lista de todos os repositórios em um domínio.

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

A saída mostra que alguns dos repositórios são administrados por contas 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"
        }
    ]
}
```

Você pode paginar a resposta de `list-repositories-in-domain` usando os parâmetros `--max-results` e `--next-token`. Para `--max-results`, especifique um número inteiro de 1 a 1000 para especificar o número de resultados retornados em uma única página. Ele assume 50 como padrão. Para retornar as páginas subsequentes, execute `list-repositories-in-domain` outra vez e passe o valor `nextToken` recebido na saída do comando anterior para `--next-token`. Quando a opção `--next-token` não é usada, a primeira página de resultados sempre é retornada.

Para gerar os nomes dos repositórios em uma lista mais compacta, tente o comando a seguir.

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

Exemplo de saída:

```
repo1
repo2
repo3
```

O exemplo a seguir mostra o ID da conta, além do nome do repositório.

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

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

Exemplo de saída:

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

Para obter mais informações sobre o `--query` parâmetro, consulte ListRepositoriesa Referência CodeArtifact da API.

Exibir ou modificar a configuração de um repositório

Você pode visualizar e atualizar detalhes sobre seu repositório usando o CodeArtifact console ou o AWS Command Line Interface (AWS CLI).

> **Note**
>
> Depois de criar um repositório, não é possível alterar o nome, a conta AWS associada ou o domínio dele.

Tópicos

- Exibir ou modificar a configuração de um repositório (console)
- Exibir ou modificar a configuração de um repositório (AWS CLI)

Exibir ou modificar a configuração de um repositório (console)

Você pode ver detalhes e atualizar seu repositório usando o CodeArtifact console.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, escolha Repositórios e escolha o nome do repositório que você deseja exibir ou modificar.
3. Expanda Detalhes para ver o seguinte:

   - O domínio do repositório. Escolha o nome do domínio para saber mais sobre ele.

- A política de recursos do repositório. Escolha Aplicar uma política de repositório para adicionar uma.
- O nome do recurso da Amazon (ARN) do repositório.
- Se o seu repositório tiver uma conexão externa, você poderá escolher a conexão para saber mais sobre ela. Um repositório pode ter apenas uma conexão externa. Para obter mais informações, consulte Conectar um CodeArtifact repositório a um repositório público.
- Se o repositório tiver repositórios upstream, você poderá escolher um para ver os detalhes. Um repositório pode ter até dez repositórios diretos upstream. Para obter mais informações, consulte Trabalhando com repositórios upstream em CodeArtifact.

> **Note**
>
> Um repositório pode ter uma conexão externa ou repositórios upstream, mas não ambos.

4. Em Pacotes, você pode ver todos os pacotes que estão disponíveis nesse repositório. Escolha um pacote para saber mais sobre ele.
5. Escolha Exibir instruções de conexão e, em seguida, escolha um gerenciador de pacotes para aprender como configurá-lo CodeArtifact.
6. Escolha Aplicar uma política de repositório para atualizar ou adicionar uma política de recursos ao repositório. Para obter mais informações, consulte Políticas de repositório.
7. Escolha Editar para adicionar ou atualizar o seguinte.

   - A descrição do repositório.
   - As tags associadas ao repositório.
   - Se o repositório tiver uma conexão externa, você poderá alterar a qual repositório público ele se conecta. Caso contrário, você poderá adicionar um ou mais repositórios existentes como repositórios upstream. Organize-os na ordem em que você deseja que sejam priorizados CodeArtifact quando um pacote for solicitado. Para obter mais informações, consulte Ordem de prioridade do repositório upstream.

Exibir ou modificar a configuração de um repositório (AWS CLI)

Para visualizar a configuração atual de um repositório em CodeArtifact, use o `describe-repository` comando.

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

Resultado do exemplo:

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

Modificar a configuração upstream de um repositório

Um repositório upstream permite que um cliente gerenciador de pacotes acesse os pacotes contidos em mais de um repositório usando um único endpoint de URL. Para adicionar ou alterar o relacionamento upstream de um repositório, use o comando `update-repository`.

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

Resultado do exemplo:

```
{
    "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 adicionar um repositório upstream, você deve ter permissão para a ação `AssociateWithDownstreamRepository` no repositório upstream.

Para remover a relação upstream de um repositório, use uma lista vazia como argumento para a opção `--upstreams`.

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

Resultado do exemplo:

```
{
    "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 repositório

CodeArtifact usa permissões baseadas em recursos para controlar o acesso. Resource-basedas permissões permitem especificar quem tem acesso a um repositório e quais ações eles podem realizar nele. Por padrão, somente o proprietário do repositório tem acesso a ele. Você pode aplicar um documento de política que permite que outras entidades principais do IAM acessem o repositório.

Para obter mais informações, consulte Resource-Based Políticas e Identity-Based Políticas e Resource-Based Políticas.

Crie uma política de recursos para conceder acesso de leitura

Uma política de recursos é um arquivo de texto no formato JSON. O arquivo deve especificar uma entidade principal (ator), uma ou mais ações e um efeito (`Allow` ou `Deny`). Por exemplo, a política de recursos a seguir concede à conta `123456789012` permissão para baixar pacotes do repositório.

JSON

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

Como a política é avaliada somente para operações no repositório ao qual está anexada, não é necessário especificar um recurso. Como o recurso está implícito, você pode definir `Resource` como `*`. Para que um gerenciador de pacotes baixe um pacote desse repositório, será necessário também criar uma política de domínio para acesso entre contas. A política de domínio deve conceder pelo menos a permissão `codeartifact:GetAuthorizationToken` à entidade principal. Para obter um exemplo de uma política completa de domínio para conceder acesso entre contas, consulte Exemplo de políticas de domínio.

> **Note**
>
> A ação `codeartifact:ReadFromRepository` só pode ser usada em um recurso do repositório. Você não pode colocar o nome do recurso da Amazon (ARN) de um pacote como

um recurso com `codeartifact:ReadFromRepository` como a ação para permitir acesso de leitura a um subconjunto de pacotes em um repositório. Uma determinada entidade principal pode ler todos os pacotes em um repositório ou nenhum deles.

Como a única ação especificada no repositório é `ReadFromRepository`, usuários e funções da conta `1234567890` podem baixar pacotes do repositório. No entanto, eles não podem realizar outras ações neles (por exemplo, listar nomes e versões de pacotes). Normalmente, você concede permissões na política a seguir além de `ReadFromRepository` porque um usuário que baixa pacotes de um repositório também precisa interagir com eles de outras maneiras.

JSON

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

Definir uma política

Depois de criar um documento de política, use o comando `put-repository-permissions-policy` para anexá-lo a um repositório:

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

Quando você chama `put-repository-permissions-policy`, a política de recursos no repositório é ignorada ao avaliar as permissões. Isso garante que o proprietário de um domínio não possa se bloquear do repositório, o que impediria que ele pudesse atualizar a política de recursos.

> **Note**
>
> Você não pode conceder permissões a outra AWS conta para atualizar a política de recursos em um repositório usando uma política de recursos, pois a política de recursos é ignorada ao chamar put-repository-permissions-policy.

Exemplo de saída:

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

A saída do comando contém o nome do recurso da Amazon (ARN) do recurso do repositório, o conteúdo completo do documento de política e um identificador de revisão. Você pode passar o identificador de revisão para `put-repository-permissions-policy` usando a opção `--policy-revision`. Isso garante que uma revisão conhecida do documento seja sobrescrita e não uma versão mais recente definida por outro redator.

Ler uma política

Use o comando `get-repository-permissions-policy` para ler uma versão existente de um documento de política. Para formatar a saída para facilitar a leitura, use `--output` e `--query policy.document` junto com o módulo `json.tool` Python.

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

Exemplo de saída:

JSON

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

Excluir uma política

Use o comando `delete-repository-permissions-policy` para excluir uma política de um repositório.

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

O formato da saída é o mesmo que o do comando `get-repository-permissions-policy`.

Conceda acesso de leitura às entidades principais

Ao especificar o usuário raiz de uma conta como entidade principal em um documento de política, você concede acesso a todos os usuários e funções dessa conta. Para limitar o acesso a usuários ou funções selecionados, use o ARN deles na seção `Principal` da política. Por exemplo, use o seguinte para conceder acesso de leitura ao usuário do IAM `bob` na conta `123456789012`.

JSON

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

Conceder acesso de gravação aos pacotes

A ação `codeartifact:PublishPackageVersion` é usada para controlar a permissão para publicar novas versões de um pacote. O recurso usado com essa ação deve ser um pacote. O formato dos ARNs CodeArtifact do pacote é o seguinte.

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

O exemplo a seguir mostra o ARN de um pacote npm com escopo `@parity` e nome `ui` no repositório `my_repo` no domínio `my_domain`.

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

O ARN de um pacote npm sem um escopo tem a string vazia para o campo do namespace. O exemplo a seguir mostra o ARN de um pacote sem escopo e com o nome `react` no repositório `my_repo` no domínio `my_domain`.

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

A política a seguir concede permissão `123456789012` à conta para publicar versões de `@parity/ui` no repositório `my_repo`.

JSON

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

```
    ]
}
```

> **Important**
>
> Para conceder permissão para publicar versões NuGet do Maven e do pacote, adicione as seguintes permissões além de `codeartifact:PublishPackageVersion`.
>
> 1. NuGet: `codeartifact:ReadFromRepository` e especifique o recurso do repositório
> 2. Maven: `codeartifact:PutPackageMetadata`

Como essa política especifica um domínio e um repositório como parte do recurso, ela permite publicar somente quando anexada a esse repositório.

Conceder acesso de gravação a um repositório

Você pode usar curingas para conceder permissão de gravação a todos os pacotes em um repositório. Por exemplo, use a política a seguir para conceder a uma conta permissão para gravar em todos os pacotes no repositório `my_repo`.

JSON

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

Interação entre políticas de repositório e domínio

CodeArtifact oferece suporte a políticas de recursos em domínios e repositórios. Políticas de recursos são opcionais. Cada domínio pode ter uma política e cada repositório no domínio pode ter a própria política de repositório. Se uma política de domínio e uma política de repositório estiverem presentes, ambas serão avaliadas ao determinar se uma solicitação para um CodeArtifact repositório é permitida ou negada. As políticas de domínio e repositório estão sendo avaliadas usando as seguintes regras:

- Nenhuma política de recursos é avaliada ao realizar operações em nível de conta, como ListDomainsou. ListRepositories
- Nenhuma política de repositório é avaliada ao realizar operações em nível de domínio, como ou. DescribeDomainListRepositoriesInDomain
- A política de domínio não é avaliada durante a execução PutDomainPermissionsPolicy. Essa regra evita bloqueios.
- A política de domínio é avaliada durante a execução PutRepositoryPermissionsPolicy, mas a política do repositório não é avaliada.
- Uma negação explícita em qualquer política anula uma permissão em outra política.
- Uma permissão explícita só é necessária em uma política de recursos. A omissão de uma ação de uma política de repositório não resultará em uma negação implícita se a política de domínio permitir a ação.
- Quando nenhuma política de recursos permite uma ação, o resultado é uma negação implícita, a menos que a conta da entidade principal da chamada seja a conta do proprietário do domínio ou do administrador do repositório e uma política baseada em identidade permita a ação.

As políticas de recursos são opcionais quando usadas para conceder acesso em um cenário de conta única, na qual a conta do chamador usada para acessar um repositório é a mesma que a conta do proprietário do domínio e do administrador do repositório. As políticas de recursos são necessárias para conceder acesso em um cenário entre contas em que a conta do chamador não é a mesma do proprietário do domínio ou da conta do administrador do repositório. Cross-account o access in CodeArtifact segue as regras gerais do IAM para acesso entre contas, conforme descrito em Determinar se uma solicitação entre contas é permitida no Guia do usuário do IAM.

- Uma entidade principal na conta do proprietário do domínio pode ter acesso a qualquer repositório no domínio por meio de uma política baseada em identidade. Nesse caso, nenhuma permissão explícita é necessária em uma política de domínio ou repositório.

- Uma entidade principal na conta do proprietário do domínio pode ter acesso a qualquer repositório por meio de uma política de domínio ou repositório. Observe que, nesse caso, nenhuma permissão explícita é necessária em uma política baseada em identidade.
- Uma entidade principal na conta do administrador do repositório pode ter acesso ao repositório por meio de uma política baseada em identidade. Nesse caso, nenhuma permissão explícita é necessária em uma política de domínio ou repositório.
- Uma entidade principal em outra conta só recebe acesso quando permitido por pelo menos uma política de recursos e pelo menos uma política baseada em identidade, sem nenhuma política que negue explicitamente a ação.

Marcar um repositório em CodeArtifact

As tags são pares de chave-valor associados a recursos da AWS. Você pode aplicar tags aos seus repositórios no CodeArtifact. Para obter informações sobre marcação de CodeArtifact recursos, casos de uso, restrições de valor e chave de tag e tipos de recursos compatíveis, consulte. Marcando atributos

Você pode usar a CLI para especificar tags ao criar um repositório. Você pode usar o console ou a CLI para adicionar ou remover tags e atualizar os valores de tags em um repositório. Você pode adicionar até 50 tags a cada repositório.

Tópicos

- Repositórios de tag (CLI)
- Repositórios de tags (console)

Repositórios de tag (CLI)

É possível usar a CLI para gerenciar tags de repositório.

Tópicos

- Adicionar tags a um repositório (CLI)
- Exibir tags para um repositório (CLI)
- Editar as tags para um repositório (CLI)
- Remover tags de um repositório (CLI)

Adicionar tags a um repositório (CLI)

Você pode usar o console ou o AWS CLI para marcar repositórios.

Para adicionar uma tag a um repositório ao criá-lo, consulte Criar um repositório.

Nestas etapas, partimos do princípio de que você já instalou uma versão recente da AWS CLI ou atualizou para a versão atual. Para obter mais informações, consulte Instalar a AWS Command Line Interface.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o nome de recurso da Amazon (ARN) do repositório no qual você deseja adicionar tags e a chave e o valor da tag que você deseja adicionar.

> **Note**
>
> Para obter o ARN do repositório, execute o comando `describe-repository`:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Você pode adicionar mais de uma tag a um repositório. Por exemplo, para marcar um repositório nomeado *my_repo* em um domínio chamado *my_domain* com duas tags, uma chave de tag nomeada *key1* com o valor de tag de *value1* e uma chave de tag nomeada *key2* com o valor de tag de*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
```

Se for bem-sucedido, este comando não terá saída.

Exibir tags para um repositório (CLI)

Siga estas etapas para usar o AWS CLI para visualizar as AWS tags de um repositório. Se não foram adicionadas tags, a lista retornará vazia.

No terminal ou na linha de comando, execute o comando list-tags-for-resource.

> **Note**
>
> Para obter o ARN do repositório, execute o comando `describe-repository`:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Por exemplo, para ver uma lista de chaves e valores de tag para um repositório nomeado *my_repo* em um domínio chamado *my_domain* com o valor `arn:aws:codeartifact:us-west-2:111122223333:repository/my_domain/my_repo` ARN:

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

Se houver êxito, o comando retornará informações semelhantes às seguintes:

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

Editar as tags para um repositório (CLI)

Siga estas etapas para usar o AWS CLI para editar uma tag para um repositório. Você pode alterar o valor para uma chave existente ou adicionar outra chave.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o ARN do repositório em que deseja atualizar uma tag e especifique a chave e o valor da tag.

> **Note**
>
> Para obter o ARN do repositório, execute o 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
```

Se for bem-sucedido, este comando não terá saída.

Remover tags de um repositório (CLI)

Siga estas etapas para usar o AWS CLI para remover uma tag de um repositório.

> **Note**
>
> Se você excluir um repositório, todas as associações de tags serão removidas do repositório excluído. Não é necessário remover tags antes de excluir um repositório.

No terminal ou na linha de comando, execute o comando untag-resource, especificando o ARN do repositório do qual deseja remover tags e a chave da tag que deseja remover.

> **Note**
>
> Para obter o ARN do repositório, execute o comando `describe-repository`:
>
> ```
> aws codeartifact describe-repository --domain my_domain --repository my_repo --
> query repository.arn
> ```

Por exemplo, para remover várias tags em um repositório chamado *my_repo* em um domínio nomeado *my_domain* com as chaves de tag *key1* e*key2*:

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

Se for bem-sucedido, este comando não terá saída. Depois de remover as tags, você pode exibir as tags restantes no repositório usando o comando `list-tags-for-resource`.

Repositórios de tags (console)

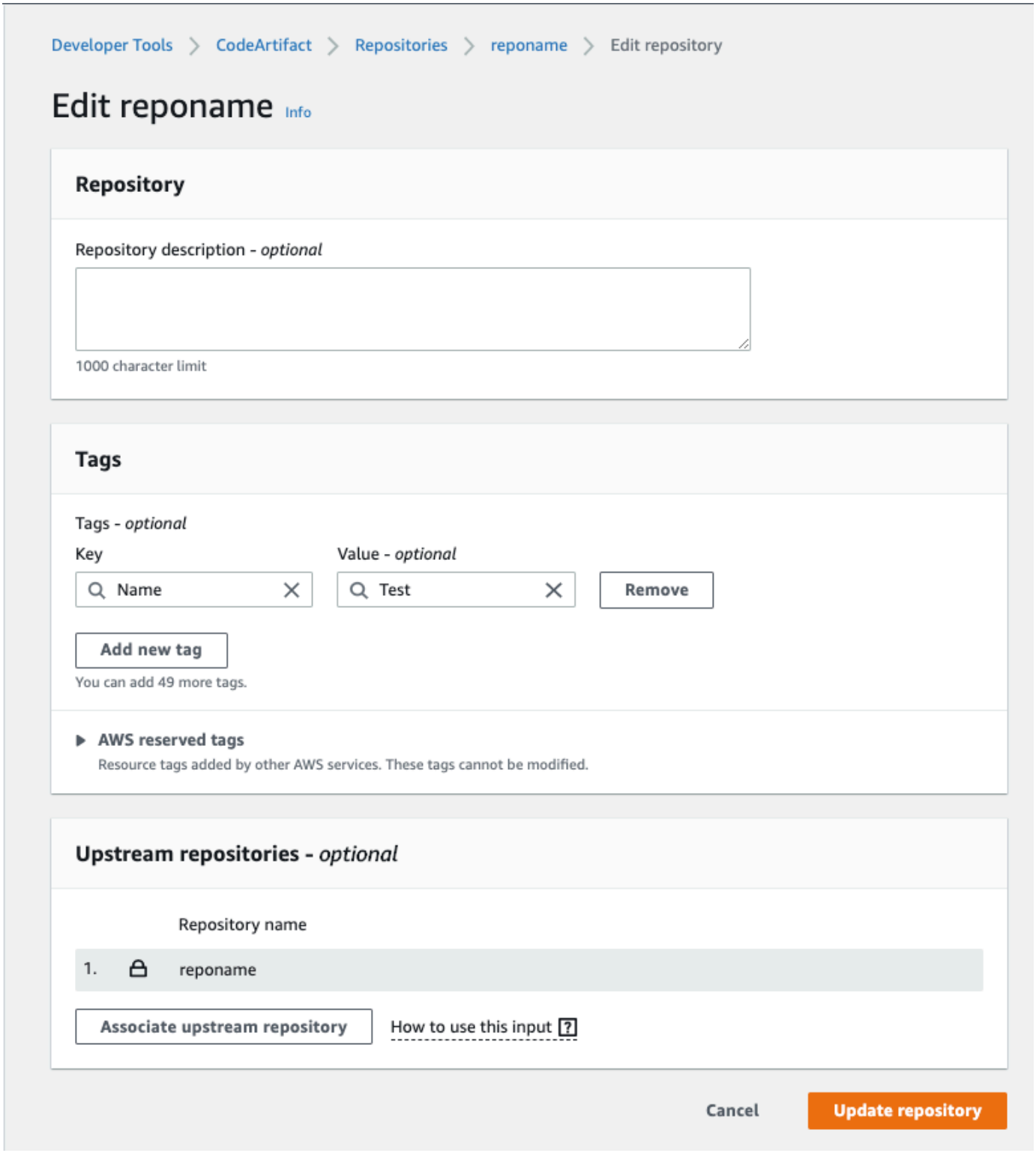
Você pode usar o console ou a CLI para marcar recursos.

Tópicos

Adicionar tags a um repositório (console)

Você pode usar o console para adicionar tags a um repositório existente.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Na página Repositórios, escolha o repositório ao qual deseja adicionar tags.
3. Expanda a seção Detalhes.
4. Em Tags do repositório, se não houver tags no repositório, escolha Adicionar tags do repositório. Se houver tags no repositório, escolha Exibir e editar tags do repositório.
5. Selecione Adicionar nova tag.
6. Nos campos Chave e Valor, insira o texto para cada tag que deseja adicionar. (O campo Value (Valor) é opcional.) Por exemplo, em Key (Chave), insira **`Name`**. Em Valor, informe **`Test`**.



7. (Opcional) Escolha Add tag (Adicionar tag) para adicionar mais linhas e inserir mais tags.
8. Escolha Atualizar repositório.

Exibir tags de um repositório (console)

Você pode usar o console para listar tags de repositórios existentes.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Na página Repositórios, escolha o repositório em que deseja exibir tags.
3. Expanda a seção Detalhes.
4. Em Tags do repositório, escolha Exibir e editar tags do repositório.

> **Note**
>
> Se não houver tags adicionadas a esse repositório, o console lerá Adicionar tags do repositório.

Editar tags de um repositório (console)

Você pode usar o console para editar as tags adicionadas ao repositório.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Na página Repositórios, escolha o repositório em que deseja atualizar tags.
3. Expanda a seção Detalhes.
4. Em Tags do repositório, escolha Exibir e editar tags do repositório.

> **Note**
>
> Se não houver tags adicionadas a esse repositório, o console lerá Adicionar tags do repositório.

5. Nos campos Key (Chave) e Value (Valor), atualize os valores em cada campo conforme necessário. Por exemplo, para a chave **`Name`**, em Value (Valor), altere **`Test`** para **`Prod`**.
6. Escolha Atualizar repositório.

Remover tags de um repositório (console)

Você pode usar o console para excluir tags de repositórios.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Na página Repositórios, escolha o repositório em que deseja remover tags.
3. Expanda a seção Detalhes.
4. Em Tags do repositório, escolha Exibir e editar tags do repositório.

> **Note**
>
> Se não houver tags adicionadas a esse repositório, o console lerá Adicionar tags do repositório.

5. Ao lado da chave e do valor para cada tag que você deseja excluir, escolha Remover.
6. Escolha Atualizar repositório.

Trabalhando com repositórios upstream em CodeArtifact

Um repositório pode ter outros AWS CodeArtifact repositórios como repositórios upstream. Isso permite que um cliente gerenciador de pacotes acesse os pacotes contidos em mais de um repositório usando um único endpoint de repositório.

Você pode adicionar um ou mais repositórios upstream a um AWS CodeArtifact repositório usando o Console de gerenciamento da AWS, AWS CLI, ou SDK. Para associar um repositório a um repositório upstream, você deve ter permissão para a ação `AssociateWithDownstreamRepository` no repositório upstream. Para obter mais informações, consulte Criar um repositório com um repositório upstream e Adicionar ou remover repositórios upstream.

Se um repositório upstream tiver uma conexão externa com um repositório público, os repositórios que estiverem na posição downstream poderão extrair pacotes desse repositório público. Por exemplo, suponha que o repositório `my_repo` tenha um repositório upstream chamado `upstream` e `upstream` tenha uma conexão externa com um repositório npm público. Nesse caso, um gerenciador de pacotes conectado ao `my_repo` pode extrair pacotes do repositório público npm. Para obter mais informações sobre como solicitar pacotes de repositórios upstream ou conexões externas, consulte Solicitar uma versão do pacote com repositórios upstream ou Solicitar pacotes de conexões externas.

Tópicos

Qual é a diferença entre repositórios upstream e conexões externas?

Em CodeArtifact, os repositórios upstream e as conexões externas se comportam basicamente da mesma forma, mas há algumas diferenças importantes.

1. Você pode adicionar até 10 repositórios upstream a um CodeArtifact repositório. Você só pode adicionar uma conexão externa.
2. Existem chamadas de API separadas para adicionar um repositório upstream ou uma conexão externa.
3. O comportamento de retenção de pacotes é um pouco diferente, pois os pacotes solicitados dos repositórios upstream são retidos nesses repositórios. Para obter mais informações, consulte Retenção de pacotes em repositórios intermediários.

Adicionar ou remover repositórios upstream

Siga as etapas nas seções a seguir para adicionar ou remover repositórios upstream de ou para um CodeArtifact repositório. Para obter mais informações sobre como repositórios upstream, consulte Trabalhando com repositórios upstream em CodeArtifact.

Este guia contém informações sobre como configurar outros CodeArtifact repositórios como repositórios upstream. Para obter informações sobre como configurar uma conexão externa com repositórios públicos como npmjs.com, Nuget Gallery, Maven Central ou PyPI, consulte Adicionar uma conexão externa.

Adicionar ou remover repositórios upstream (console)

Execute as etapas do procedimento a seguir para adicionar um repositório como repositório upstream usando o console. CodeArtifact Para obter informações sobre como adicionar um repositório upstream com o AWS CLI, consulte. Adicionar ou remover repositórios upstream (AWS CLI)

Para adicionar um repositório upstream usando o console CodeArtifact

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.

2. No painel de navegação, selecione Domínios e selecione o nome de domínio que contém o repositório.
3. Escolha o nome para o seu repositório.
4. Escolha Editar.
5. Em Repositórios upstream, escolha Associar repositório upstream e adicione o repositório que você deseja adicionar como repositório upstream. Você só pode adicionar repositórios no mesmo domínio como repositórios upstream.
6. Escolha Atualizar repositório.

Para remover um repositório upstream usando o console CodeArtifact

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, selecione Domínios e selecione o nome de domínio que contém o repositório.
3. Escolha o nome para o seu repositório.
4. Escolha Editar.
5. Em Repositórios upstream, localize a entrada da lista do repositório upstream que você deseja remover e escolha Dissociar.

> **Important**
>
> Depois de remover um repositório upstream de um CodeArtifact repositório, os gerenciadores de pacotes não terão acesso aos pacotes no repositório upstream ou em qualquer um de seus repositórios upstream.

6. Escolha Atualizar repositório.

Adicionar ou remover repositórios upstream (AWS CLI)

Você pode adicionar ou remover os CodeArtifact repositórios upstream de um repositório usando o AWS Command Line Interface ().AWS CLI Para tanto, use o comando `update-repository` e especifique os repositórios upstream usando o parâmetro `--upstreams`.

Você só pode adicionar repositórios no mesmo domínio como repositórios upstream.

Para adicionar repositórios upstream (AWS CLI)

1. Caso contrário, siga as etapas Configurando com AWS CodeArtifact para instalar e configurar o AWS CLI with CodeArtifact.
2. Use o comando `aws codeartifact update-repository` com o sinalizador `--upstreams` para adicionar repositórios upstream.

> **Note**
>
> Chamar o comando `update-repository` substitui os repositórios upstream configurados existentes pela lista de repositórios fornecida com o sinalizador `--upstreams`. Se quiser adicionar repositórios upstream e manter os existentes, você deve incluir os repositórios upstream existentes na chamada.

O exemplo de comando a seguir adiciona dois repositórios upstream a um repositório chamado `my_repo` que está em um domínio chamado `my_domain`. A ordem dos repositórios upstream no `--upstreams` parâmetro determina sua prioridade de pesquisa ao CodeArtifact solicitar um pacote do `my_repo` repositório. Para obter mais informações, consulte Ordem de prioridade do repositório upstream.

Para saber mais sobre como se conectar a repositórios públicos externos, como npmjs.com ou Maven Central, consulte Conectar um CodeArtifact repositório a um repositório público.

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

O resultado contém os repositórios upstream, da seguinte forma:

```
{
       "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 remover um repositório upstream (AWS CLI)

1. Caso contrário, siga as etapas Configurando com AWS CodeArtifact para instalar e configurar o AWS CLI with CodeArtifact.
2. Para remover repositórios upstream de um CodeArtifact repositório, use o `update-repository` comando com o sinalizador. `--upstreams` A lista de repositórios fornecida ao comando será o novo conjunto de repositórios upstream do repositório. CodeArtifact Inclua os repositórios upstream existentes que você deseja manter e omita os repositórios upstream que deseja remover.

   Para remover todos os repositórios upstream de um repositório, use o comando `update-repository` e inclua `--upstreams` sem um argumento. O comando a seguir remove repositórios upstream de um repositório chamado `my_repo` que está em um domínio chamado `my_domain`.

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

   O resultado mostra que a lista de `upstreams` está vazia.

   ```
   {
          "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 um CodeArtifact repositório a um repositório público

Você pode adicionar uma conexão externa entre um CodeArtifact repositório e um repositório público externo, como https://npmjs.como repositório Maven Central. Então, quando você solicita um pacote do CodeArtifact repositório que ainda não está presente no repositório, o pacote pode ser obtido na conexão externa. Isso possibilitará o consumo de dependências de código aberto utilizadas pelo seu aplicativo.

Em CodeArtifact, a forma pretendida de usar conexões externas é ter um repositório por domínio com uma conexão externa com um determinado repositório público. Por exemplo, se você quiser se conectar a npmjs.com, configure um repositório em seu domínio com uma conexão externa com npmjs.com e configure todos os outros repositórios com um upstream para ele. Dessa forma, todos os repositórios serão capazes de usar os pacotes que já foram obtidos de npmjs.com, em vez de buscá-los e armazená-los novamente.

Tópicos

- Conectar-se a um repositório externo (console)
- Conectar-se a um repositório externo (CLI)
- Repositórios de conexão externa compatíveis
- Remover uma conexão externa (CLI)

Conectar-se a um repositório externo (console)

Quando você usa o console para adicionar uma conexão a um repositório externo, o seguinte ocorrerá:

1. Um `-store` repositório para o repositório externo será criado em seu CodeArtifact domínio, caso ainda não exista um. Esses repositórios `-store` se comportam como repositórios intermediários entre o seu repositório e o externo e permitem que você se conecte a mais de um repositório externo.
2. O repositório `-store` adequado é adicionado como um upstream ao seu repositório.

A lista a seguir contém cada `-store` repositório CodeArtifact e o respectivo repositório externo ao qual eles se conectam.

1. `cargo-store` está conectado a crates.io.
2. `clojars-store` está conectado ao repositório Clojars.
3. `commonsware-store`está conectado ao Repositório CommonsWare Android.
4. `google-android-store` está conectado ao Google Android.
5. `gradle-plugins-store` está conectado aos plug-ins do Gradle.
6. `maven-central-store` está conectado ao repositório Maven Central.
7. `npm-store` está conectado a npmjs.com.
8. `nuget-store` está conectado a nuget.org.
9. `pypi-store` está conectado ao Python Packaging Authority.
10. `rubygems-store`está conectado a RubyGems .org.

Para se conectar a um repositório externo (console)

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, selecione Domínios e selecione o nome de domínio que contém o repositório.
3. Escolha o nome para o seu repositório.
4. Escolha Editar.
5. Em Repositórios upstream, escolha Associar repositório upstream e adicione o repositório `-store` correto conectado como upstream.
6. Escolha Atualizar repositório.

Depois que o `-store` repositório é adicionado como um repositório upstream, os gerenciadores de pacotes conectados ao seu CodeArtifact repositório podem buscar pacotes do respectivo repositório externo.

Conectar-se a um repositório externo (CLI)

Você pode usar o AWS CLI para conectar seu CodeArtifact repositório a um repositório externo adicionando uma conexão externa diretamente ao repositório. Isso permitirá que os usuários conectados ao CodeArtifact repositório, ou a qualquer um de seus repositórios downstream, busquem pacotes do repositório externo configurado. Cada CodeArtifact repositório só pode ter uma conexão externa.

É recomendável ter um repositório por domínio, com uma conexão externa com um determinado repositório público. Para conectar outros repositórios ao repositório público, basta adicionar o repositório com a conexão externa como um upstream a eles. Se você ou outra pessoa em seu domínio já tiver configurado conexões externas no console, seu domínio provavelmente já terá um repositório `-store` com uma conexão externa com o repositório público ao qual você deseja se conectar. Para obter mais informações sobre repositórios `-store` e conexão com o console, consulte Conectar-se a um repositório externo (console).

Para adicionar uma conexão externa a um CodeArtifact repositório (CLI)

- Use `associate-external-connection` para adicionar uma conexão externa. O exemplo a seguir mostra a conexão de um repositório ao registro npm público, npmjs.com. Para ver uma lista dos repositórios externos, consulte Repositórios de conexão externa compatíveis.

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

  Resultado do exemplo:

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

Depois de adicionar uma conexão externa, consulte Solicitar pacotes de conexões externas para obter informações sobre como solicitar pacotes de um repositório externo com uma conexão externa.

Repositórios de conexão externa compatíveis

CodeArtifact suporta uma conexão externa com os seguintes repositórios públicos. Para usar a CodeArtifact CLI para especificar uma conexão externa, use o valor na coluna Nome do `--external-connection` parâmetro ao executar o `associate-external-connection` comando.

Tipo de repositório	Description	Nome
Maven	Repositório do Clojars	`public:maven-clojars`
Maven	CommonsWare Repositório Android	`public:maven-commonsware`
Maven	Repositório do Google Android	`public:maven-googleandroid`
Maven	Repositório de plug-ins do Gradle	`public:maven-gradleplugins`
Maven	Maven Central	`public:maven-central`
npm	Registro npm público	`public:npmjs`

Tipo de repositório	Description	Nome
NuGet	NuGet Galeria	`public:nuget-org`
Python	Python Package Index	`public:pypi`
Ruby	RubyGems.org	`public:ruby-gems-org`
Rust	Crates.io	`public:crates-io`

Remover uma conexão externa (CLI)

Para remover uma conexão externa que foi adicionada usando o `associate-external-connection` comando no AWS CLI, use`disassociate-external-connection`.

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

Resultado do exemplo:

```
{
    "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 uma versão do pacote com repositórios upstream

Quando um cliente (por exemplo, npm) solicita uma versão de pacote de um CodeArtifact repositório chamado `my_repo` que tem vários repositórios upstream, o seguinte pode ocorrer:

- Se `my_repo` contiver a versão do pacote solicitada, ela será devolvida ao cliente.
- Se `my_repo` não contiver a versão do pacote solicitada, CodeArtifact procure-a nos `my_repo` repositórios upstream. Se a versão do pacote for encontrada, uma referência a ela será copiada para `my_repo` e a versão do pacote será devolvida ao cliente.
- Se `my_repo` nem seus repositórios upstream contiverem a versão do pacote, uma resposta HTTP 404 de `Not Found` será exibida ao cliente.

Quando você adiciona repositórios upstream usando o comando `create-repository` ou `update-repository`, a ordem em que eles são passados para o parâmetro `--upstreams` determina a prioridade quando uma versão do pacote é solicitada. Especifique os repositórios upstream `--upstreams` na ordem em que você deseja CodeArtifact usar quando uma versão do pacote for solicitada. Para obter mais informações, consulte Ordem de prioridade do repositório upstream.

A quantidade máxima de repositórios upstream diretos permitidos para um repositório é dez. Como os repositórios upstream diretos também podem ter seus próprios repositórios diretos, é CodeArtifact possível pesquisar versões de pacotes em mais de 10 repositórios. O número máximo de repositórios CodeArtifact examinados quando uma versão do pacote é solicitada é 25.

Retenção de pacotes de repositórios upstream

Se uma versão de pacote solicitada for encontrada em um repositório upstream, uma referência a ela será retida e estará sempre disponível no repositório downstream. A versão retida do pacote não é afetada por nenhum das seguintes ações:

- Excluir o repositório upstream.
- Desconectar o repositório upstream do repositório downstream.
- Excluir a versão do pacote do repositório upstream.
- Editar a versão do pacote no repositório upstream (por exemplo, adicionar um novo ativo a ele).

Buscar pacotes por meio de uma relação upstream

Se um CodeArtifact repositório tiver um relacionamento upstream com um repositório que tenha uma conexão externa, as solicitações de pacotes que não estão no repositório upstream serão copiadas do repositório externo. Por exemplo, considere a seguinte configuração: um repositório chamado `repo-A` tem um repositório upstream chamado. `repo-B` `repo-B`tem uma conexão externa com https://npmjs.com.

Se npm estiver configurado para usar o `repo-A` repositório, a execução `npm install` aciona a cópia dos pacotes de dentro. https://npmjs.com`repo-B` As versões instaladas também são incorporadas em `repo-A`. O exemplo a seguir instala o `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
```

Depois de executar `npm install`, `repo-A` contém apenas a versão mais recente (`lodash 4.17.20`) porque essa é a versão que foi obtida por npm de `repo-A`.

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

Resultado do exemplo:

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

Como `repo-B` tem uma conexão externa com https://npmjs.com, todas as versões do pacote importadas https://npmjs.comsão armazenadas em`repo-B`. Essas versões do pacote poderiam ter sido obtidas por qualquer repositório downstream com uma relação upstream com `repo-B`.

O conteúdo de `repo-B` fornece uma maneira de ver todos os pacotes e versões de pacotes importados https://npmjs.comao longo do tempo. Por exemplo, para ver todas as versões do pacote `lodash` importadas ao longo do tempo, você pode usar `list-package-versions`, como abaixo.

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

Resultado do exemplo:


```
{
    "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"
}
```

Retenção de pacotes em repositórios intermediários

CodeArtifact permite encadear repositórios upstream. Por exemplo, `repo-A` pode ter `repo-B` como upstream e `repo-B` pode ter `repo-C` como upstream. Essa configuração faz com que as versões do pacote sejam inseridas no `repo-B` e `repo-C` disponíveis do `repo-A`.

Quando um gerenciador de pacotes se conecta ao repositório `repo-A` e busca uma versão do pacote no repositório `repo-C`, a versão do pacote não será retida no repositório `repo-B`. A versão do pacote só será mantida no repositório mais downstream, neste exemplo `repo-A`. Ele não será retido em nenhum repositório intermediário. Isso também vale para cadeias mais longas. Por exemplo, se houvesse quatro repositórios: `repo-A`, `repo-B`, `repo-C` e `repo-D` e um gerenciador de pacotes conectado ao `repo-A` buscasse uma versão do pacote no `repo-D`, a versão do pacote seria retida em `repo-A`, mas não em `repo-B` ou `repo-C`.

O comportamento de retenção de pacotes é semelhante ao extrair uma versão de pacote de um repositório externo, exceto que a versão do pacote sempre é retida no repositório que tem a conexão externa anexada. Por exemplo, o `repo-A` tem o `repo-B` como um upstream. O `repo-B` tem o `repo-C` como upstream e o `repo-C` também tem o npmjs.com configurado como uma conexão externa; veja o diagrama a seguir.

Se um gerenciador de pacotes conectado ao **`repo-A`** solicitar uma versão de pacote, por exemplo, lodash 4.17.20, e a versão do pacote não estiver presente em nenhum dos três repositórios, ela será obtida em npmjs.com. Quando o lodash 4.17.20 for obtido, ele será retido no `repo-A`, pois é o repositório mais downstream e o `repo-C`, pois tem a conexão externa com npmjs.com anexada. O lodash 4.17.20 não será retido no `repo-B`, pois é um repositório intermediário.

Solicitar pacotes de conexões externas

As seções a seguir descrevem como solicitar um pacote de uma conexão externa e o CodeArtifact comportamento esperado ao solicitar um pacote.

Tópicos

- Importar versões de pacotes com mais de um ativo

Buscar pacotes de uma conexão externa

Para buscar pacotes de uma conexão externa depois de adicioná-los ao seu CodeArtifact repositório, conforme descrito emConectar um CodeArtifact repositório a um repositório público, configure seu gerenciador de pacotes para usar seu repositório e instalar os pacotes.

> **Note**
>
> As instruções a seguir usam o npm, para visualizar as instruções de configuração e uso de outros tipos de pacotes, consulte Usando o CodeArtifact com Maven, Usando o CodeArtifact com NuGet ouUsando CodeArtifact com Python.

Para buscar pacotes de uma conexão externa

1. Configure e autentique seu gerenciador de pacotes com seu CodeArtifact repositório. Para npm, use o comando `aws codeartifact login` a seguir.

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

2. Solicitar o pacote do repositório público. Paranpm, use o `npm install` comando a seguir, *`lodash`* substituindo pelo pacote que você deseja instalar.

   ```
   npm install lodash
   ```

3. Depois que o pacote for copiado para o seu CodeArtifact repositório, você poderá usar os `list-package-versions` comandos `list-packages` e para visualizá-lo.

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

   Resultado do exemplo:

   ```
   {
       "packages": [
           {
   ```

```
            "format": "npm",
            "package": "lodash"
        }
    ]
}
```

O `list-package-versions` comando lista todas as versões do pacote copiadas para o seu CodeArtifact repositório.

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

Resultado do exemplo:

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

Latência da conexão externa

Ao buscar um pacote de um repositório público usando uma conexão externa, há um atraso entre o momento em que o pacote é obtido do repositório público e o momento em que é armazenado em seu repositório. CodeArtifact Por exemplo, digamos que você tenha instalado a versão 1.2.5 do pacote npm “lodash”, conforme descrito em Buscar pacotes de uma conexão externa. Embora o comando `npm install lodash` lodash tenha sido concluído com êxito, a versão do pacote talvez ainda não apareça no seu CodeArtifact repositório. Normalmente, leva cerca de 3 minutos para que a versão do pacote apareça no repositório, embora, ocasionalmente, possa levar mais tempo.

Por causa dessa latência, você pode ter recuperado com êxito uma versão do pacote, mas talvez ainda não consiga ver a versão no seu repositório no CodeArtifact console ou ao chamar as operações ListPackages e ListPackageVersions da API. Depois CodeArtifact de persistir de forma assíncrona a versão do pacote, ela ficará visível no console e por meio de solicitações de API.

CodeArtifact comportamento quando um repositório externo não está disponível

Ocasionalmente, um repositório externo sofrerá uma interrupção, o que significa que CodeArtifact não é possível obter pacotes dele, ou a busca de pacotes é muito mais lenta do que o normal. Quando isso ocorrer, as versões do pacote já retiradas de um repositório externo (por exemplo, npmjs.com) e armazenadas em um CodeArtifact repositório continuarão disponíveis para download. CodeArtifact No entanto, pacotes que ainda não estão armazenados CodeArtifact podem não estar disponíveis, mesmo quando uma conexão externa com esse repositório foi configurada. Por exemplo, seu CodeArtifact repositório pode conter a versão do pacote npm `lodash 4.17.19` porque é isso que você está usando em seu aplicativo até agora. Quando você quiser atualizar para`4.17.20`, normalmente CodeArtifact buscará essa nova versão em npmjs.com e a armazenará em seu repositório. CodeArtifact No entanto, se npmjs.com estiver passando por uma interrupção, essa nova versão não ficará disponível. A única solução alternativa é tentar novamente mais tarde, após a recuperação de npmjs.com.

Interrupções no repositório externo também podem afetar a publicação de novas versões de pacotes no. CodeArtifact Em um repositório com uma conexão externa configurada, não CodeArtifact permitirá a publicação de uma versão do pacote que já esteja presente no repositório externo. Para obter mais informações, consulte Visão geral dos pacotes. No entanto, em casos raros, uma interrupção no repositório externo pode significar que CodeArtifact ele não tem up-to-date informações sobre quais pacotes e versões de pacotes estão presentes em um repositório externo. Nesse caso, CodeArtifact pode permitir a publicação de uma versão do pacote que normalmente seria negada.

Disponibilidade de novas versões de pacote

Para que uma versão do pacote em um repositório público, como npmjs.com, esteja disponível por meio de um CodeArtifact repositório, ela deve primeiro ser adicionada a um cache regional de metadados do pacote. Esse cache é mantido CodeArtifact em cada AWS região e contém metadados que descrevem o conteúdo dos repositórios públicos compatíveis. Por causa desse cache, há um atraso entre o momento em que uma nova versão do pacote é publicada em um

repositório público e o momento em que ela é disponibilizada. CodeArtifact Esse atraso varia conforme o tipo de pacote.

Para pacotes npm, Python e Nuget, pode haver um atraso de até 30 minutos a partir do momento em que uma nova versão do pacote é publicada em npmjs.com, pypi.org ou nuget.org e quando ela está disponível para instalação em um repositório. CodeArtifact CodeArtifact sincroniza automaticamente os metadados desses dois repositórios para garantir que o cache esteja atualizado.

Para pacotes Maven, pode haver um atraso de até 3 horas a partir do momento em que uma nova versão do pacote é publicada em um repositório público e quando está disponível para instalação em um CodeArtifact repositório. CodeArtifact verificará novas versões de um pacote no máximo uma vez a cada 3 horas. A primeira solicitação para um determinado nome de pacote após a expiração da vida útil do cache de 3 horas fará com que todas as novas versões desse pacote sejam importadas para o cache regional.

Para pacotes Maven de uso comum, as novas versões normalmente são importadas a cada 3 horas, porque a alta taxa de solicitações significa que o cache geralmente será atualizado assim que a vida útil do cache expirar. Para pacotes usados com pouca frequência, o cache não terá a versão mais recente até que uma versão do pacote seja solicitada em um CodeArtifact repositório. Na primeira solicitação, somente as versões importadas anteriormente estarão disponíveis CodeArtifact, mas essa solicitação fará com que o cache seja atualizado. Nas solicitações subsequentes, as novas versões do pacote serão adicionadas ao cache e estarão disponíveis para download.

Importar versões de pacotes com mais de um ativo

Tanto os pacotes Maven quanto Python podem ter vários ativos por versão de pacote. Isso torna a importação de pacotes desses formatos mais complexa do que npm e NuGet pacotes, que têm apenas um ativo por versão do pacote. Para obter descrições de quais ativos são importados para esses tipos de pacotes e como os ativos recém-adicionados são disponibilizados, consulte Solicitar pacotes Python de upstreams e conexões externas e Solicitação de pacotes Maven de upstreams e conexões externas.

Ordem de prioridade do repositório upstream

Ao solicitar uma versão do pacote de um repositório com um ou mais repositórios upstream, a prioridade deles corresponde à ordem em que foram listados ao chamar o comando `create-repository` ou `update-repository`. Quando a versão do pacote solicitada é encontrada, a pesquisa é interrompida, mesmo que ela não tenha pesquisado todos os repositórios upstream. Para obter mais informações, consulte Adicionar ou remover repositórios upstream (AWS CLI).

Use o comando `describe-repository` para ver a ordem de prioridade.

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

O resultado pode ser o seguinte: Isso mostra que a prioridade do repositório upstream é `upstream-1` em primeiro, `upstream-2` em segundo e `upstream-3` em terceiro.

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

Exemplo de ordem simples de prioridade

No diagrama a seguir, o repositório `my_repo` tem três repositórios upstream. A ordem de prioridade dos repositórios upstream é`upstream-1`, `upstream-2` e `upstream-3`.

Uma solicitação de uma versão de pacote em `my_repo` pesquisa os repositórios na seguinte ordem até que seja encontrada ou até que uma resposta HTTP 404 de `Not Found` seja exibida ao cliente:

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

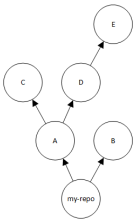
Se a versão do pacote for encontrada, a pesquisa é interrompida, mesmo que ela não tenha pesquisado todos os repositórios upstream. Por exemplo, se a versão do pacote for encontrada em`upstream-1`, a pesquisa será interrompida e CodeArtifact não examinará `upstream-2` ou`upstream-3`.

Quando você usa o AWS CLI comando `list-package-versions` para listar as versões do pacote`my_repo`, ele olha somente para dentro`my_repo`. Ele não lista as versões do pacote nos repositórios upstream.

Exemplo de ordem complexa de prioridade

Se um repositório upstream tiver seus próprios repositórios upstream, a mesma lógica será usada para localizar uma versão do pacote antes de passar para o próximo repositório upstream. Por exemplo, suponha que seu repositório `my_repo` tenha dois repositórios upstream: `A` e `B`. Se o repositório `A` tiver repositórios upstream, uma solicitação de uma versão do pacote no `my_repo` pesquisará primeiro no `my_repo`, em segundo no `A` e depois nos repositórios upstream do `A` em diante.

No diagrama a seguir, o repositório `my_repo` contém repositórios upstream. O repositório upstream `A` tem dois repositórios upstream e o `D` tem um repositório upstream. Os repositórios upstream no mesmo nível no diagrama aparecem na ordem de prioridade, da esquerda para a direita (o repositório `A` tem uma ordem de prioridade mais alta que o repositório e o repositório `B` e o `C` tem uma ordem de prioridade mais alta que o repositório `D`).



Nesse exemplo, uma solicitação de uma versão de pacote no `my_repo` pesquisa nos repositórios na seguinte ordem até que seja encontrada ou até que um gerenciador de pacotes exiba uma resposta HTTP 404 de `Not Found` ao cliente:

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

Comportamento da API com repositórios upstream

Quando você chama determinados CodeArtifact APIs repositórios conectados a repositórios upstream, o comportamento pode ser diferente dependendo se os pacotes ou as versões do pacote estão armazenados no repositório de destino ou no repositório upstream. O comportamento deles APIs está documentado aqui.

Para obter mais informações CodeArtifact APIs, consulte a Referência CodeArtifact da API.

A maioria das APIs referências a um pacote ou versão de pacote retornará um `ResourceNotFound` erro se a versão do pacote especificada não estiver presente no repositório de destino. Isso acontece mesmo se o pacote ou a versão do pacote estiver presente em um repositório upstream. Efetivamente, os repositórios upstream são ignorados ao chamá-los. APIs São APIs elas:

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

Para demonstrar esse comportamento, temos dois repositórios: o `target-repo` e `upstream-repo`. O `target-repo` está vazio e tem o `upstream-repo` configurado como um repositório upstream. O `upstream-repo` contém o pacote npm `lodash`.

Ao chamar a API `DescribePackageVersion` no `upstream-repo`, que contém o pacote `lodash`, obtemos o seguinte 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"
    }
```

Ao chamar a mesma API no `target-repo`, que está vazio, mas tem o `upstream-repo` configurado como upstream, obtemos o seguinte resultado:

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

A API `CopyPackageVersions` se comporta de forma diferente. Por padrão, a API `CopyPackageVersions` copia somente as versões do pacote que estão armazenadas no repositório de destino. Se uma versão do pacote for armazenada no repositório upstream, mas não no repositório de destino, ela não será copiada. Para incluir versões de pacotes que são armazenados somente no repositório upstream, defina o valor de `includeFromUpstream` para `true` na sua solicitação de API.

Para mais informações sobre a API `CopyPackageVersions`, consulte Copiar pacotes entre repositórios.

Trabalhando com pacotes em CodeArtifact

Os tópicos a seguir mostram como realizar ações em pacotes usando a CodeArtifact CLI e a API.

Tópicos

Visão geral dos pacotes

Um pacote é formado pelo pacote de software e os metadados necessários para resolver dependências e instalar o software. Em CodeArtifact, um pacote consiste em um nome de pacote, um namespace opcional, como `@types` in`@types/node`, um conjunto de versões de pacote e metadados em nível de pacote, como tags npm.

Sumário

  - Latência de publicação
- Satus da versão do pacote
- Normalização do nome e da versão do pacote e do nome do ativo

Formatos de pacote com suporte

AWS CodeArtifact suporta os formatos de pacote Cargo, genérico, Maven, npm, NuGetPyPI, Ruby, Swift.

Publicação de pacotes

Você pode publicar novas versões de qualquer formato de pacote compatível em um CodeArtifact repositório usando ferramentas como `npm`e`twine`,`Maven`,`Gradle`,`nuget`, e. `dotnet`

Permissões de publicação

Seu usuário ou função AWS Identity and Access Management (IAM) deve ter permissões para publicar no repositório de destino. As seguintes permissões são necessárias para publicar pacotes:

- Cargo: `codeartifact:PublishPackageVersion`
- generic: `codeartifact:PublishPackageVersion`
- Maven: `codeartifact:PublishPackageVersion` e `codeartifact:PutPackageMetadata`
- npm: `codeartifact:PublishPackageVersion`
- NuGet: `codeartifact:PublishPackageVersion` e `codeartifact:ReadFromRepository`
- Python: `codeartifact:PublishPackageVersion`
- Rubi: `codeartifact:PublishPackageVersion`
- Rápido: `codeartifact:PublishPackageVersion`

Na lista de permissões anterior, a política do IAM deve especificar o recurso `package` para as permissões `codeartifact:PublishPackageVersion` e `codeartifact:PutPackageMetadata`. Ela também deve especificar o recurso `repository` para a permissão `codeartifact:ReadFromRepository`.

Para obter mais informações sobre permissões em CodeArtifact, consulteAWS CodeArtifact referência de permissões.

Substituindo ativos do pacote

Você não pode republicar um ativo de pacote que já existe com conteúdo diferente. Por exemplo, suponha que você já tenha publicado um pacote Maven com um ativo `mypackage-1.0.jar` JAR. Você só poderá publicar esse ativo outra vez se a soma de verificação dos ativos antigos e novos for idêntica. Para republicar o mesmo ativo com novo conteúdo, primeiro exclua a versão do pacote usando o comando delete-package-versions. Tentar republicar o mesmo nome de ativo com conteúdo diferente resultará em um erro de conflito HTTP 409.

Para formatos de pacote compatíveis com vários ativos (genérico, PyPI e Maven), você pode adicionar novos ativos com nomes diferentes a uma versão de pacote existente, supondo que você tenha as permissões necessárias. Para pacotes genéricos, você pode adicionar novos ativos, desde que a versão do pacote esteja no estado `Unfinished`. Como o npm é compatível apenas com um único ativo por versão de pacote, para modificar uma versão de pacote publicada de qualquer forma, você deve primeiro excluí-la usando delete-package-versions.

Se você tentar republicar um ativo que já existe (por exemplo, `mypackage-1.0.jar`) e o conteúdo do ativo publicado e o do novo ativo forem idênticos, a operação será bem-sucedida porque é idempotente.

Pacotes privados e repositórios públicos

CodeArtifact não publica pacotes armazenados em CodeArtifact repositórios em repositórios públicos, como npmjs.com ou Maven Central. CodeArtifact importa pacotes de repositórios públicos para um CodeArtifact repositório, mas nunca move pacotes na outra direção. Os pacotes que você publica CodeArtifact nos repositórios permanecem privados e só estão disponíveis para as AWS contas, funções e usuários aos quais você concedeu acesso.

Publicação de versões de pacotes corrigidos

Às vezes, talvez você queira publicar uma versão modificada do pacote, potencialmente aquela disponível em um repositório público. Por exemplo, você pode ter encontrado um bug em uma dependência crítica do aplicativo chamada `mydep 1.1` e precisa corrigi-lo antes que o fornecedor do pacote possa revisar e aceitar a alteração. Conforme descrito anteriormente, CodeArtifact impede que você publique `mydep 1.1` no seu CodeArtifact repositório se o repositório público puder ser acessado a partir do seu repositório por meio de CodeArtifact repositórios upstream e uma conexão externa.

Para contornar isso, publique a versão do pacote em um CodeArtifact repositório diferente onde o repositório público não esteja acessível. Em seguida, use a `copy-package-versions` API para copiar a versão corrigida do `mydep 1.1` para o CodeArtifact repositório de onde você a consumirá.

Limites de tamanho de ativos para publicação

O tamanho máximo de um ativo de pacote que pode ser publicado é limitado pela cota Máxima do tamanho do arquivo do ativo mostrada em Cotas em AWS CodeArtifact. Por exemplo, você não pode publicar uma roda Maven JAR ou Python maior que a cota máxima atual do tamanho do arquivo do ativo. Se você precisar armazenar ativos maiores CodeArtifact, solicite um aumento de cota.

Além da cota máxima do tamanho do arquivo do ativo, o tamanho máximo de uma solicitação de publicação para pacotes npm é de 2 GB. Esse limite é independente da cota máxima do tamanho do arquivo do ativo e não pode ser aumentado com um aumento de cota. Em uma solicitação de publicação npm (HTTP PUT), os metadados do pacote e o conteúdo do arquivo tar do pacote npm são agrupados. Por esse motivo, o tamanho máximo real de um pacote npm que pode ser publicado varia e depende do tamanho dos metadados incluídos.

> **Note**
>
> Os pacotes npm publicados são limitados a um tamanho máximo inferior a 2 GB.

Latência de publicação

As versões de pacotes publicadas em um CodeArtifact repositório geralmente estão disponíveis para download em menos de um segundo. Por exemplo, se você publicar uma versão do pacote npm em CodeArtifact with`npm publish`, essa versão deverá estar disponível para um `npm install` comando em menos de um segundo. No entanto, a publicação pode ser inconsistente e, às vezes, demorar mais. Se você precisar usar uma versão do pacote imediatamente após a publicação, use novas tentativas para garantir que o download seja confiável. Por exemplo, depois de publicar a versão do pacote, repita o download até três vezes se a versão do pacote recém-publicada não estiver disponível inicialmente na primeira tentativa de download.

> **Note**
>
> A importação de uma versão do pacote de um repositório público normalmente leva mais tempo do que a publicação. Para obter mais informações, consulte Latência da conexão externa.

Satus da versão do pacote

Cada versão do pacote CodeArtifact tem um status que descreve o estado atual e a disponibilidade da versão do pacote. Você pode alterar o status da versão do pacote no AWS CLI e no SDK. Para obter mais informações, consulte Atualizar o status da versão do pacote.

Os seguintes valores de status da versão do pacote são possíveis:

- Publicado — A versão do pacote foi publicada com sucesso e pode ser solicitada usando um gerenciador de pacotes. A versão do pacote será incluída nas listas de versões de pacotes retornadas aos gerenciadores de pacotes, por exemplo, na saída de `npm view <package-name> versions`. Todos os ativos da versão do pacote estão disponíveis no repositório.
- Não concluído — O cliente carregou um ou mais ativos para uma versão do pacote, mas não a finalizou ao movê-la para o estado `Published`. No momento, apenas as versões genéricas e do pacote Maven podem ter o status de `Unfinished`. Para pacotes Maven, isso pode ocorrer quando o cliente carrega um ou mais ativos para uma versão do pacote, mas não publica um arquivo `maven-metadata.xml` para o pacote que inclui essa versão. Quando uma versão do pacote Maven estiver Não concluída, ela não será incluída nas listas de versões retornadas aos clientes, como `mvn` ou `gradle`. Portanto, ela não poderá ser usada como parte de uma compilação. Pacotes genéricos podem ser mantidos deliberadamente no `Unfinished` estado fornecendo o `unfinished` sinalizador ao chamar a PublishPackageVersionAPI. Um pacote genérico pode ser alterado para o `Published` estado omitindo a `unfinished` sinalização ou chamando a UpdatePackageVersionsStatusAPI.
- Não listado — Os ativos da versão do pacote estão disponíveis para download no repositório, mas a versão do pacote não está incluída na lista de versões retornadas aos gerenciadores de pacotes. Por exemplo, para um pacote npm, a saída de `npm view <package-name> versions` não incluirá a versão do pacote. Isso significa que a lógica de resolução de dependências do npm não selecionará a versão do pacote porque a versão não aparece na lista de versões disponíveis. No entanto, se a versão do pacote Não listado já estiver referenciada em um arquivo `npm package-`

`lock.json`, ela ainda poderá ser baixada e instalada, por exemplo, durante a execução de `npm ci`.

- Arquivado — Os ativos da versão do pacote não podem mais ser baixados. A versão do pacote não será incluída na lista de versões retornada aos gerenciadores de pacotes. Como os ativos não estão disponíveis, o consumo da versão do pacote pelos clientes é bloqueado. Se a compilação do seu aplicativo depender de uma versão atualizada para Arquivado, a compilação será interrompida, supondo que a versão do pacote não tenha sido armazenada em cache localmente. Você não pode usar um gerenciador de pacotes ou uma ferramenta de compilação para republicar uma versão de pacote arquivado porque ela ainda está presente no repositório, mas você pode alterar o status da versão do pacote de volta para Não listada ou Publicada com a API. UpdatePackageVersionsStatus
- Descartada — A versão do pacote não aparece nas listagens e os ativos não podem ser baixados do repositório. A principal diferença entre descartado e arquivado é que, com o status de descartado, os ativos da versão do pacote serão excluídos permanentemente por. CodeArtifact Por esse motivo, você não pode mover uma versão de pacote de Descartada para Arquivada, Não Listada ou Publicada. A versão do pacote não pode mais ser usada porque os ativos foram excluídos. Depois que uma versão do pacote for marcada como Descartada, você não será mais cobrado pelo armazenamento dos ativos do pacote.

As versões do pacote de todos os status serão retornadas por padrão ao chamar list-package-versions sem nenhum parâmetro `--status`.

Além dos estados listados anteriormente, uma versão do pacote também pode ser excluída com a DeletePackageVersionsAPI. Depois de ser excluída, uma versão do pacote não está mais no repositório e você pode republicá-la livremente usando um gerenciador de pacotes ou uma ferramenta de compilação. Depois que uma versão do pacote for excluída, você não será mais cobrado pelo armazenamento dos ativos dessa versão.

Normalização do nome e da versão do pacote e do nome do ativo

CodeArtifact normaliza nomes de pacotes, versões de pacotes e nomes de ativos antes de armazená-los, o que significa que os nomes ou versões CodeArtifact podem ser diferentes do nome ou versão fornecidos quando o pacote foi publicado. Para obter mais informações sobre como os nomes e as versões são normalizados CodeArtifact para cada tipo de pacote, consulte a documentação a seguir:

- Normalização do nome do pacote Python

- Normalização do nome, versão e nome do ativo do pacote NuGet

CodeArtifact não executa a normalização em outros formatos de pacote.

Listar nomes de pacotes

Use o `list-packages` comando in CodeArtifact para obter uma lista de todos os nomes de pacotes em um repositório. Esse comando retorna apenas os nomes dos pacotes, não as versões.

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

Exemplo de saída:

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

Listar nomes de pacotes npm

Para listar apenas os nomes dos pacotes npm, defina o valor da opção `--format` como `npm`.

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

Para listar pacotes npm em um namespace (escopo npm), use as opções `--namespace` e `--format`.

> **Important**
>
> O valor da opção `--namespace` não deve incluir o `@` inicial. Para pesquisar o namespace`@types`, defina o valor como. *`types`*

> **Note**
>
> A opção `--namespace` filtra por prefixo de namespace. Qualquer pacote npm com um escopo que comece com o valor passado para a opção `--namespace` será retornado na resposta `list-packages`.

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

Exemplo de saída:

```
{
    "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"

        }
    ]
}
```

Listar nomes de pacotes Maven

Para listar apenas os nomes dos pacotes Maven, defina o valor da opção `--format` como `maven`. Você também deve especificar o ID do grupo Maven na opção `--namespace`.

> **Note**
>
> A opção `--namespace` filtra por prefixo de namespace. Qualquer pacote npm com um escopo que comece com o valor passado para a opção `--namespace` será retornado na resposta `list-packages`.

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

Exemplo de saída:

```
{
    "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"

        }
    ]
}
```

Listar nomes de pacotes Python

Para listar apenas os nomes dos pacotes Python, defina o valor da opção `--format` como `pypi`.

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

Filtrar por prefixo do nome do pacote

Para retornar pacotes que começam com uma string especificada, você pode usar a opção `--package-prefix`.

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

Exemplo de saída:

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

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

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

```
            "format": "npm"

        }
    ]
}
```

Combinações de opções de pesquisa compatíveis

Você pode usar as opções `--format`, `--namespace` e `--package-prefix` em qualquer combinação, exceto que `--namespace` não pode ser usada sozinha. Pesquisar todos os pacotes npm com um escopo que começa com `@types` exige que a opção `--format` seja especificada. Usar `--namespace` por si só resulta em um erro.

Não usar nenhuma das três opções também é compatível com `list-packages` e retornará todos os pacotes de todos os formatos presentes no repositório.

Formatar resultado

Você pode usar parâmetros que estão disponíveis para todos os AWS CLI comandos para tornar a `list-packages` resposta compacta e mais legível. Use o parâmetro `--query` para especificar o formato de cada versão de pacote retornada. Use o parâmetro `--output` para formatar a resposta como texto sem formatação.

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

Exemplo de saída:

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

Para obter mais informações, consulte Controlar a saída do comando de AWS CLI no Guia do usuário do AWS Command Line Interface .

Padrões e outras opções

Por padrão, o número máximo de resultados retornados por `list-packages` é 100. Você pode alterar esse limite de resultados usando a opção `--max-results`.

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

O valor máximo permitido de `--max-results` é 1.000. Para permitir a listagem de pacotes em repositórios com mais de 1.000 pacotes, `list-packages` é compatível com a paginação usando o campo `nextToken` na resposta. Se o número de pacotes no repositório for maior que o valor de `--max-results`, você poderá passar o valor de `nextToken` para outra invocação de `list-packages` para obter a próxima página de resultados.

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

Listar versões de pacotes

Use o `list-package-versions` comando in AWS CodeArtifact para obter uma lista de todas as versões de um nome de pacote em um repositório.

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

Exemplo de saída:

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

Você pode adicionar o parâmetro `--status` à chamada `list-package-versions` para filtrar os resultados com base no status da versão do pacote. Para obter mais informações sobre o status da versão do pacote, consulte Satus da versão do pacote.

Você pode paginar a resposta de `list-package-versions` usando os parâmetros `--max-results` e `--next-token`. Para `--max-results`, especifique um número inteiro de 1 a 1000 para especificar o número de resultados retornados em uma única página. Ele assume 50 como padrão. Para retornar as páginas subsequentes, execute `list-package-versions` outra vez e passe o valor `nextToken` recebido na saída do comando anterior para `--next-token`. Quando a opção `--next-token` não é usada, a primeira página de resultados sempre é retornada.

O comando `list-package-versions` não lista versões do pacote em repositórios upstream. No entanto, as referências às versões do pacote em um repositório upstream que foram copiadas para o seu repositório durante uma solicitação de versão do pacote são listadas. Para obter mais informações, consulte Trabalhando com repositórios upstream em CodeArtifact.

Listar versões do pacote npm

Para listar todas as versões de um pacote npm, defina o valor da opção `--format` como `npm`.

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

Para listar versões do pacote npm em um namespace específico (escopo npm), use a opção `--namespace`. O valor da opção `--namespace` não deve incluir o `@` inicial. Para pesquisar o namespace`@types`, defina o valor como. *types*

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

Listar as versões do pacote Maven

Para listar todas as versões de um pacote Maven, defina o valor da opção `--format` como `maven`. Você também deve especificar o ID do grupo Maven na opção `--namespace`.

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

Classificar versões

O `list-package-versions` pode gerar versões classificadas em ordem decrescente com base no horário de publicação (as versões publicadas mais recentemente são listadas primeiro). Use o parâmetro `--sort-by` com um valor de `PUBLISHED_TIME`, como segue.

```
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
```

Exemplo de saída:

```
{

  "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...."
}
```

Versão de exibição padrão

O valor de retorno para `defaultDisplayVersion` depende do formato do pacote:

- Para pacotes genéricos, Maven e PyPI, é a versão mais recente do pacote publicada.
- Para pacotes npm, é a versão referenciada pela tag `latest`. Se a tag `latest` não estiver definida, é a versão mais recente do pacote publicada.

Formatar resultado

Você pode usar parâmetros que estão disponíveis para todos os AWS CLI comandos para tornar a `list-package-versions` resposta compacta e mais legível. Use o parâmetro `--query` para especificar o formato de cada versão de pacote retornada. Use o parâmetro `--output` para formatar a resposta como texto sem formatação.

```
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]'
```

Exemplo de saída:

```
0.1.1
0.1.2
0.1.0
```

```
3.0.0
```

Para obter mais informações, consulte Controlar a saída do comando de AWS CLI no Guia do usuário do AWS Command Line Interface .

Listar ativos da versão do pacote

Um ativo é um arquivo individual (por exemplo, um arquivo npm ou `.tgz` arquivo Maven POM ou JAR) armazenado e associado CodeArtifact a uma versão do pacote. Você pode usar o comando `list-package-version-assets` para listar os ativos em cada versão do pacote.

Execute o `list-package-version-assets` comando para retornar as seguintes informações sobre cada ativo em sua AWS conta e sua AWS região atual:

- Nome.
- Tamanho, em bytes.
- Um conjunto de valores de hash usados para validação da soma de verificação.

Por exemplo, use o comando a seguir para listar os ativos do pacote Python `flatten-json`, versão `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
```

A seguir, é mostrada a saída.

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

Listar ativos de um pacote npm

Um pacote npm sempre tem um único ativo com o nome de `package.tgz`. Para listar os ativos de um pacote npm com escopo definido, inclua o escopo na opção `--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
```

Listar ativos de um pacote Maven

Para listar os ativos de um pacote Maven, inclua o namespace do pacote na opção `--namespace`. Para listar os ativos do pacote `commons-cli:commons-cli` do Maven:

```
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
```

Baixe ativos da versão do pacote

Um ativo é um arquivo individual (por exemplo, um arquivo npm ou `.tgz` arquivo Maven POM ou JAR) armazenado e associado CodeArtifact a uma versão do pacote. Você pode baixar os ativos do pacote usando o `get-package-version-assets command`. Isso permite que você recupere ativos sem usar um cliente gerenciador de pacotes, como `npm` ou `pip`. Para baixar um ativo, você deve fornecer o nome do ativo, que pode ser obtido usando o comando `list-package-version-assets`. Para obter mais informações, consulte Listar ativos da versão do pacote. O ativo será baixado para o armazenamento local com um nome de arquivo especificado por você.

O exemplo a seguir baixa o *guava-27.1-jre.jar* ativo do pacote Maven *com.google.guava:guava* com a versão*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
```

Neste exemplo, o nome do arquivo foi especificado *guava-27.1-jre.jar* pelo último argumento no comando anterior, então o ativo baixado será nomeado*guava-27.1-jre.jar*.

A saída do comando será:

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

> **Note**
>
> Para baixar ativos de um pacote npm com escopo definido, inclua o escopo na opção `--namespace`. O símbolo `@` deve ser omitido ao usar `--namespace`. Por exemplo, se o escopo for `@types`, use `--namespace types`.

O download de ativos usando `get-package-version-asset` requer a permissão `codeartifact:GetPackageVersionAsset` no recurso do pacote. Para obter mais informações sobre políticas de permissão baseadas em recursos, consulte as Resource-based políticas no Guia do AWS Identity and Access Management usuário.

Copiar pacotes entre repositórios

Você pode copiar versões de pacotes de um repositório para outro no CodeArtifact. Isso pode ser útil para cenários como fluxos de trabalho de promoção de pacotes ou compartilhamento de versões de pacotes entre equipes ou projetos. Os repositórios de origem e de destino devem ter o mesmo domínio para copiar versões de pacotes.

Permissões obrigatórias do IAM para copiar pacotes

Para copiar as versões do pacote CodeArtifact, o usuário chamador deve ter as permissões necessárias do IAM e a política baseada em recursos anexada aos repositórios de origem e destino deve ter as permissões necessárias. Para obter mais informações sobre políticas e CodeArtifact repositórios de permissões com base em recursos, consulte. Políticas de repositório

O usuário que está chamando `copy-package-versions` deve ter a permissão `ReadFromRepository` no repositório de origem e a permissão `CopyPackageVersions` no repositório de destino.

O repositório de origem deve ter a permissão `ReadFromRepository` e o repositório de destino deve ter a permissão `CopyPackageVersions` atribuída à conta do IAM ou ao usuário que copia pacotes. As políticas a seguir são exemplos de políticas de repositório a serem adicionadas ao repositório de origem ou ao repositório de destino com o comando `put-repository-permissions-policy`. *111122223333*Substitua pelo ID da conta que está ligando`copy-package-versions`.

> **Note**
>
> Chamar `put-repository-permissions-policy` substituirá a política atual do repositório, se houver. Você pode usar o comando `get-repository-permissions-policy` para ver se existe uma política. Para obter mais informações, consulte Ler uma política. Se existir uma política, talvez você queira adicionar essas permissões a ela em vez de substituí-la.

Exemplo de política de permissões do repositório de origem

JSON

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

Exemplo de política de permissões do repositório de destino

JSON

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

Copiar versões do pacote

Use o `copy-package-versions` comando in CodeArtifact para copiar uma ou mais versões do pacote de um repositório de origem para um repositório de destino no mesmo domínio. O exemplo a seguir copiará as versões 6.0.2 e 4.0.0 de um pacote npm chamado `my-package` do repositório `my_repo` para o repositório `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
```

Copie várias versões do mesmo nome de pacote em uma única operação. Para copiar versões de nomes de pacotes diferentes, você deve chamar `copy-package-versions` para cada um deles.

O comando anterior produzirá a seguinte saída, supondo que ambas as versões possam ser copiadas com sucesso.

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

Copiar um pacote dos repositórios upstream

Normalmente, `copy-package-versions` só procura no repositório especificado pela opção `--source-repository` as versões a serem copiadas. No entanto, você pode copiar versões do repositório de origem e dos repositórios upstream usando a opção `--include-from-upstream`. Se você usa o CodeArtifact SDK, chame a `CopyPackageVersions` API com o `includeFromUpstream` parâmetro definido como true. Para obter mais informações, consulte Trabalhando com repositórios upstream em CodeArtifact.

Copiar um pacote npm com escopo definido

Para copiar uma versão do pacote npm em um escopo, use a opção `--namespace` para especificar o escopo. Por exemplo, para copiar o pacote `@types/react`, use `--namespace types`. O símbolo `@` deve ser omitido ao usar `--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 versões do pacote Maven

Para copiar versões do pacote Maven entre repositórios, especifique o pacote a ser copiado passando o ID do grupo Maven com a opção `--namespace` e o artifactID Maven com a opção `--name`. Por exemplo, para copiar uma única versão 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
```

Se a versão do pacote for copiada com sucesso, o resultado será semelhante ao seguinte.

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

Versões que não existem no repositório de origem

Se você especificar uma versão que não existe no repositório de origem, a cópia falhará. Se existirem algumas versões no repositório de origem e outras não existirem, nenhuma versão será

copiada. No exemplo a seguir, a versão 0.2.0 do pacote npm do `array-unique` está presente no repositório de origem, mas a versão 5.6.7 não está:

```
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
```

A saída neste cenário será semelhante ao seguinte:

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

O código de erro `SKIPPED` é usado para indicar que a versão não foi copiada para o repositório de destino porque outra versão não pôde ser copiada.

Versões que já existem no repositório de destino

Quando uma versão do pacote é copiada para um repositório onde ela já existe, CodeArtifact compara os ativos do pacote e os metadados do nível da versão do pacote nos dois repositórios.

Se os ativos e metadados da versão do pacote forem idênticos nos repositórios de origem e de destino, uma cópia não será executada, mas a operação será considerada bem-sucedida. Isso significa que `copy-package-versions` é idempotente. Quando isso ocorrer, a versão que já estava presente nos repositórios de origem e de destino não será listada na saída do`copy-package-versions`.

No exemplo a seguir, duas versões do pacote npm `array-unique` estão presentes no repositório de origem `repo-1`. A versão 0.2.1 também está presente no repositório de destino `dest-repo` e a versão 0.2.0 não está.

```
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
```

A saída neste cenário será semelhante ao seguinte:

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

A versão 0.2.0 está listada no `successfulVersions` porque foi copiada com sucesso do repositório de origem para o de destino. A versão 0.2.1 não é mostrada na saída, pois já estava presente no repositório de destino.

Se os ativos ou metadados da versão do pacote forem diferentes nos repositórios de origem e de destino, a operação de cópia falhará. Você pode usar o parâmetro `--allow-overwrite` para forçar uma substituição.

Se existirem algumas versões no repositório de destino e outras não existirem, nenhuma versão será copiada. No exemplo a seguir, a versão 0.3.2 do pacote npm `array-unique` está presente nos repositórios de origem e de destino, mas o conteúdo da versão do pacote é diferente. A versão 0.2.1 está presente no repositório de origem, mas não no 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
```

A saída neste cenário será semelhante ao seguinte:

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

A versão 0.2.1 está marcada como `SKIPPED` porque não foi copiada para o repositório de destino. Ela não foi copiada porque a cópia da versão 0.3.2 falhou porque ela já estava presente no repositório de destino, mas não era idêntica nos repositórios de origem e de destino.

Especificar uma revisão da versão do pacote

A revisão da versão do pacote é uma string que especifica um determinado conjunto de ativos e metadados da versão de um pacote. Você pode especificar uma revisão da versão do pacote para copiar as versões do pacote que estão em um estado definido. Para especificar uma revisão da versão do pacote, use o parâmetro `--version-revisions` para passar uma ou mais versões do pacote separadas por vírgula e os pares de revisão da versão do pacote para o comando `copy-package-versions`.

> **Note**
>
> É necessário especificar o parâmetro `--versions` ou `--version-revisions` com `copy-package-versions`. Não é possível especificar ambos.

O exemplo a seguir só copiará a versão 0.3.2 do pacote `my-package` se ela estiver presente no repositório de origem com a revisão da versão do pacote `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
```

O exemplo a seguir copia duas versões do pacote `my-package`, 0.3.2 e 0.3.13. A cópia só será bem-sucedida se, no repositório de origem, a versão 0.3.2 do `my-package` tiver revisão

REVISION-1-SAMPLE-6C81EFF7DA55CC e a versão 0.3.13 tiver revisão 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 localizar as revisões de uma versão do pacote, use o comando `describe-package-version` ou `list-package-versions`.

Para obter mais informações, consulte Revisão da versão do pacote e CopyPackageVersion na Referência da API do CodeArtifact .

Copiar pacotes npm

Para obter mais informações sobre o `copy-package-versions` comportamento com pacotes npm, consulte as tags npm e a CopyPackageVersions API.

Excluir um pacote ou uma versão do pacote

É possível excluir uma ou mais versões de pacotes de cada vez usando o comando `delete-package-versions`. Para remover completamente um pacote de um repositório, incluindo todas as versões e configurações associadas, use o comando `delete-package`. Pode existir um pacote em um repositório sem nenhuma versão do pacote. Isso pode acontecer quando todas as versões são excluídas usando o comando `delete-package-versions` ou se o pacote foi criado sem nenhuma versão usando a operação de API `put-package-origin-configuration` (consulte Editar controles de origem do pacote).

Tópicos

- Excluindo um pacote (AWS CLI)
- Excluir um pacote (console)
- Excluindo uma versão do pacote (AWS CLI)
- Excluir uma versão de pacote (console)
- Excluir um pacote npm ou uma versão do pacote
- Excluir um pacote Maven ou uma versão do pacote

- Práticas recomendadas para excluir pacotes ou versões de pacotes

Excluindo um pacote (AWS CLI)

Você pode excluir um pacote, incluindo todas as versões e configurações do pacote, usando o comando `delete-package`. O exemplo a seguir exclui o pacote PyPI chamado `my-package` no repositório `my_repo` no domínio `my_domain`:

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

Exemplo de saída:

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

Você pode confirmar que o pacote foi excluído executando `describe-package` para o mesmo nome de pacote:

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

Excluir um pacote (console)

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, escolha Repositories (Repositórios).

3. Escolha o Repositório do qual você deseja excluir um pacote.
4. Escolha o Pacote que você deseja excluir.
5. Escolha Excluir pacote.

Excluindo uma versão do pacote (AWS CLI)

É possível excluir uma ou mais versões de pacotes de cada vez usando o comando `delete-package-versions`. O exemplo a seguir exclui as versões `4.0.0`, `4.0.1` e `5.0.0` do pacote PyPI chamado `my-package` no `my_repo` no domínio `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
```

Exemplo de saída:

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

Você pode confirmar se as versões foram excluídas executando `list-package-versions` para o mesmo nome de pacote:

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

Excluir uma versão de pacote (console)

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, escolha Repositories (Repositórios).
3. Escolha o Repositório do qual você deseja excluir versões de pacotes.
4. Escolha o Pacote do qual você deseja excluir versões.
5. Selecione a Versão do pacote que você deseja excluir.
6. Escolha Excluir.

> **Note**
>
> No console, é possível excluir apenas uma versão de pacote de cada vez. Para excluir mais de um por vez, use a CLI.

Excluir um pacote npm ou uma versão do pacote

Para excluir um pacote npm ou versões de pacotes individuais, defina a opção `--format` como `npm`. Para excluir uma versão de pacote npm com um escopo, use a opção `--namespace` para especificar o escopo. Por exemplo, para excluir o pacote `@types/react`, use `--namespace types`. Omita o símbolo `@` ao usar `--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 excluir o pacote `@types/react`, incluindo todas as versões:

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

Excluir um pacote Maven ou uma versão do pacote

Para excluir um pacote Maven ou versões de pacotes individuais, defina a opção `--format` como `maven` e especifique o pacote a ser excluído passando o ID do grupo Maven com a opção `--namespace` e o artifactID Maven com a opção `--name`. O exemplo a seguir mostra como excluir uma única versão 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
```

O exemplo a seguir mostra como excluir o pacote `com.google.guava:guava`, incluindo todas as versões:

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

Práticas recomendadas para excluir pacotes ou versões de pacotes

Se você precisar excluir uma versão do pacote, como prática recomendada, convém criar um repositório para armazenar uma cópia de backup da versão do pacote que você deseja excluir. Você pode fazer isso primeiro chamando `copy-package-versions` para o repositório de backup:

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

Após copiar a versão do pacote, será possível chamar `delete-package-versions` no pacote ou na versão do pacote que deseja excluir.

```
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
```

Exiba e atualize os detalhes e dependências da versão do pacote

Você pode visualizar informações sobre a versão de um pacote, incluindo dependências, em CodeArtifact. Você também pode atualizar o status de uma versão de pacote. Para obter mais informações sobre o status da versão do pacote, consulte Satus da versão do pacote.

Exibir detalhes da versão de pacote

Use o comando `describe-package-version` para exibir os detalhes sobre as versões de pacote. Os detalhes da versão do pacote são extraídos de um pacote quando ele é publicado no. CodeArtifact Os detalhes em diferentes pacotes variam e dependem de seus formatos e da quantidade de informações que seus autores adicionaram a eles.

A maioria das informações na saída do comando `describe-package-version` depende do formato do pacote. Por exemplo, `describe-package-version` extrai as informações de um pacote npm do arquivo `package.json`. A revisão é criada por CodeArtifact. Para obter mais informações, consulte Especificar uma revisão da versão do pacote.

Duas versões de pacote com o mesmo nome podem estar no mesmo repositório se cada uma estiver em namespaces diferentes. Use o parâmetro `--namespace` opcional para especificar um namespace. Para obter mais informações, consulte Exibir detalhes da versão de pacote npm ou Exibir detalhes da versão de pacote Maven.

O exemplo a seguir retorna detalhes sobre a versão `1.9.0` de um pacote Python chamado `pyhamcrest` que está no repositório `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
```

A saída deverá ser semelhante a:

```
{
  "format": "pypi",
  "package": "PyHamcrest",
  "displayName": "PyHamcrest",
  "version": "1.9.0",
  "summary": "Hamcrest framework for matcher objects",
  "homePage": "https://github.com/hamcrest/PyHamcrest",
  "publishedTime": 1566002944.273,
```

```
  "licenses": [
    {
      "id": "license-id",
      "name": "license-name"
    }
  ],
  "revision": "REVISION-SAMPLE-55C752BEE9B772FC"
}
```

> **Note**
>
> CodeArtifact busca detalhes da versão do pacote, como a página inicial do pacote ou as informações da licença do pacote, a partir dos metadados fornecidos pelo autor do pacote. Se alguma dessas informações exceder 400 KB, que é o limite de tamanho do item do DynamoDB CodeArtifact , não será possível processar esses dados e talvez você não veja essas informações no console ou na resposta do. `describe-package-version` Por exemplo, um pacote python como o https://pypi.org/project/rapyd-sdk/tem um campo de licença muito grande, então essas informações não seriam processadas pelo CodeArtifact.

Exibir detalhes da versão de pacote npm

Para exibir detalhes sobre uma versão de pacote npm, defina o valor da opção `--format` como **`npm`**. Como opção, inclua o namespace da versão do pacote (escopo npm) na opção `--namespace`. O valor da opção `--namespace` não deve incluir o `@` inicial. Para pesquisar o namespace`@types`, defina o valor como. *`types`*

O exemplo a seguir retorna detalhes sobre a versão `4.41.5` de um pacote npm chamado `webpack` no escopo `@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
```

A saída deverá ser semelhante a:

```
{
  "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"
  }
}
```

Exibir detalhes da versão de pacote Maven

Para exibir detalhes sobre uma versão de pacote Maven, defina o valor da opção `--format` como `maven` e inclua o namespace da versão do pacote na opção `--namespace`.

O exemplo a seguir retorna detalhes sobre a versão `1.2` de um pacote Maven chamado `commons-rng-client-api` que está no namespace `org.apache.commons` e no repositório `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
```

A saída deverá ser semelhante a:

```
{
  "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 não extrai informações detalhadas da versão do pacote dos arquivos POM principais. Os metadados de uma determinada versão do pacote incluirão apenas informações no POM para essa versão exata do pacote, não para o POM principal ou qualquer outro POM referenciado transitivamente usando a tag `parent` do POM. Isso significa que a saída de `describe-package-version` omitirá metadados (como informações de licença) para versões do pacote Maven que dependem de uma referência `parent` para conter esses metadados.

Exibir dependências de versão de pacote

Use o comando `list-package-version-dependencies` para obter uma lista das dependências de uma versão do pacote. O comando a seguir lista as dependências de um pacote npm chamado `my-package`, versão `4.41.5`, no repositório `my_repo`, no domínio `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
```

A saída deverá ser semelhante a:

```
{
  "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 o intervalo de valores compatíveis com o campo dependencyType, consulte o tipo de PackageDependencydados na CodeArtifact API.

Exibir arquivo readme da versão do pacote

Alguns formatos de pacote, como npm, incluem um arquivo `README`. Use `get-package-version-readme` para obter o arquivo `README` de uma versão do pacote. O comando a seguir retorna o arquivo `README` de um pacote npm chamado `my-package`, versão `4.41.5`, no repositório `my_repo`, no domínio `my_domain`.

> **Note**
>
> CodeArtifact não suporta a exibição de arquivos readme de pacotes genéricos ou 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
```

A saída deverá ser semelhante a:

```
{
  "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"
}
```

Atualizar o status da versão do pacote

Cada versão do pacote CodeArtifact tem um status que descreve o estado atual e a disponibilidade da versão do pacote. Você pode alterar o status da versão do pacote usando o console AWS CLI e o console.

> **Note**
>
> Para obter mais informações sobre o status da versão do pacote, incluindo uma lista dos status disponíveis, consulte Satus da versão do pacote.

Atualizar o status da versão do pacote

Definir o status de uma versão de pacote permite controlar como uma versão de pacote pode ser usada sem excluí-la completamente do repositório. Por exemplo, quando uma versão de pacote tem o status de `Unlisted`, ela ainda pode ser baixada normalmente, mas não aparecerá nas listas de versões de pacotes retornadas para comandos como `npm view`. A UpdatePackageVersionsStatus API permite definir o status da versão do pacote de várias versões do mesmo pacote em uma única chamada de API. Para obter uma descrição dos diferentes status, consulte Visão geral dos pacotes.

Use o comando `update-package-versions-status` para alterar o status de uma versão do pacote para `Published`, `Unlisted` ou `Archived`. Para conferir as permissões necessárias do IAM para usar o comando, consulte Permissões obrigatórias do IAM para atualizar o status da versão do pacote. O exemplo a seguir define o status da versão 4.1.0 do pacote npm `chalk` como `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
```

Exemplo de saída:

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

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

Este exemplo usa um pacote npm, mas o comando funciona de forma idêntica para outros formatos. Várias versões podem ser movidas para o mesmo status de destino usando um único comando, confira o exemplo a seguir.

```
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
```

Exemplo de saída:

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

Observe que, uma vez publicada, uma versão do pacote não pode ser movida de volta ao estado `Unfinished`, portanto, esse status não é permitido como um valor para o parâmetro `--target-status`. Para mover a versão do pacote para o estado `Disposed`, use o comando `dispose-package-versions` no lugar, conforme descrito abaixo.

Permissões obrigatórias do IAM para atualizar o status da versão do pacote

Para chamar `update-package-versions-status` para um pacote, você deve ter a permissão `codeartifact:UpdatePackageVersionsStatus` no recurso do pacote. Isso significa que

você pode conceder permissão para chamar `update-package-versions-status` por pacote. Por exemplo, uma política do IAM que concedesse permissão para chamar `update-package-versions-status` o pacote npm *`chalk`* incluiria uma declaração como a seguinte.

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

Atualizar o status de um pacote npm com escopo definido

Para atualizar o status da versão do pacote de uma versão do pacote npm com um escopo, use o parâmetro `--namespace`. Por exemplo, para remover da lista a versão 8.0.0 do `@nestjs/core`, use o comando a seguir.

```
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
```

Atualizar o status de um pacote Maven

Os pacotes Maven sempre têm um ID de grupo, chamado de namespace em. CodeArtifact Use o parâmetro `--namespace` para especificar o ID do grupo Maven ao chamar `update-package-versions-status`. Por exemplo, para arquivar a versão 2.13.1 do pacote Maven `org.apache.logging.log4j:log4j`, use o comando a seguir.

```
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
```

Especificar uma revisão da versão do pacote

A revisão da versão do pacote é uma string que especifica um determinado conjunto de ativos e metadados da versão de um pacote. Você pode especificar uma revisão da versão do pacote para

atualizar o status das versões do pacote que estão em um estado definido. Para especificar uma revisão da versão do pacote, use o parâmetro `--version-revisions` para passar uma ou mais versões do pacote separadas por vírgula e os pares de revisão da versão do pacote. O status de uma versão do pacote só será atualizado se a revisão atual da versão do pacote corresponder ao valor especificado.

> **Note**
>
> O parâmetro –`--versions` também deve ser definido ao usar o parâmetro `--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 atualizar várias versões com um único comando, passe uma lista separada por vírgulas de pares de versão e de revisão de versão para as opções `--version-revisions`. O comando de exemplo a seguir define dois pares diferentes de versão do pacote e de revisão da versão do pacote.

```
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
```

Exemplo de saída:

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

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

Ao atualizar várias versões do pacote, as versões passadas para `--version-revisions` devem ser iguais às versões passadas para `--versions`. Se uma revisão for especificada de forma incorreta, o status dessa versão não será atualizado.

Usar o parâmetro de status esperado

O comando `update-package-versions-status` apresenta o parâmetro `--expected-status` compatível com a especificação do status atual esperado de uma versão do pacote. Se o status atual não corresponder ao valor passado para `--expected-status`, o status dessa versão do pacote não será atualizado.

Por exemplo, em*my_repo*, as versões 4.0.0 e 4.1.0 do pacote npm `chalk` atualmente têm o status de. `Published` Uma chamada para `update-package-versions-status` que especifica um status esperado de `Unlisted` falhará na atualização de ambas as versões do pacote devido à incompatibilidade de status.

```
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
```

Exemplo de saída:

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

Erros com versões de pacotes individuais

Há vários motivos pelos quais o status de uma versão do pacote não será atualizado durante a chamada de `update-package-versions-status`. Por exemplo, a revisão da versão do pacote pode ter sido especificada de maneira incorreta ou o status esperado não corresponde ao status atual. Nesses casos, a versão será incluída no mapa `failedVersions` na resposta da API. Se uma versão falhar, outras versões especificadas na mesma chamada para `update-package-versions-status` poderão ser ignoradas e não ter seu status atualizado. Essas versões também serão incluídas no mapa `failedVersions` com um `errorCode` de`SKIPPED`.

Na implementação atual do `update-package-versions-status`, se uma ou mais versões não puderem ter seu status alterado, todas as outras versões serão ignoradas. Ou seja, todas as versões são atualizadas com sucesso ou nenhuma versão é atualizada. Esse comportamento não é garantido no contrato da API; no futuro, algumas versões podem ser bem-sucedidas, enquanto outras falham em uma única chamada para `update-package-versions-status`.

O exemplo de comando a seguir inclui uma falha na atualização do status da versão causada por uma incompatibilidade na revisão da versão do pacote. Essa falha na atualização faz com que outra chamada de atualização de status da versão seja ignorada.

```
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
```

Exemplo de saída:

```
{
    "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="
```

```
        }
    }
}
```

Descartar as versões do pacote

O status do `Disposed` pacote tem um comportamento semelhante ao`Archived`, exceto que os ativos do pacote serão excluídos permanentemente CodeArtifact para que a conta do proprietário do domínio não seja mais cobrada pelo armazenamento de ativos. Consulte mais informações sobre o status da versão do pacote em Satus da versão do pacote. Para alterar o status de uma versão do pacote para `Disposed`, use o comando `dispose-package-versions`. Esse recurso é separado de `update-package-versions-status` porque o descarte de uma versão do pacote não é reversível. Como os ativos do pacote serão excluídos, o status da versão não pode ser alterado de volta para `Archived`, `Unlisted` ou `Published`. A única ação que pode ser executada em uma versão de pacote que foi descartada é excluí-la usando o comando `delete-package-versions`.

Para chamar `dispose-package-versions` com sucesso, a entidade principal do IAM que está chamando deve ter a permissão `codeartifact:DisposePackageVersions` no recurso do pacote.

O comportamento do comando `dispose-package-versions` é semelhante a `update-package-versions-status`, incluindo o comportamento das opções `--version-revisions` e `--expected-status` descritas nas seções de revisão da versão e status esperado. Por exemplo, o comando a seguir tenta descartar uma versão do pacote, mas falha devido a um status esperado incompatível.

```
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
```

Exemplo de saída:

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

```
    }
}
```

Se o mesmo comando for executado outra vez com um `--expected-status` de `Published`, o descarte será bem-sucedido.

```
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
```

Exemplo de saída:

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

Editar controles de origem do pacote

Em AWS CodeArtifact, as versões do pacote podem ser adicionadas a um repositório publicando-as diretamente, retirando-as de um repositório upstream ou ingerindo-as de um repositório público externo. Permitir que versões de um pacote sejam adicionadas por publicação direta e ingestão de repositórios públicos torna você vulnerável a um ataque de substituição de dependências. Para obter mais informações, consulte Ataques de substituição de dependências. Para se proteger contra um ataque de substituição de dependência, você pode configurar os controles de origem do pacote em um pacote em um repositório para limitar como as versões desse pacote podem ser adicionadas ao repositório.

A configuração dos controles de origem do pacote deve ser considerada por qualquer equipe que queira permitir que novas versões de pacotes diferentes venham tanto de fontes internas, como publicação direta, quanto de fontes externas, como repositórios públicos. Por padrão, os controles de origem do pacote serão configurados com base em como a primeira versão de um pacote é adicionada ao repositório. Para obter informações sobre as configurações de controle de origem do pacote e seus valores padrão, consulte Configurações de controle de origem do pacote.

Para remover o registro do pacote depois de usar a operação da API `put-package-origin-configuration`, use `delete-package` (consulte Excluir um pacote ou uma versão do pacote).

Cenários comuns de controle de acesso a pacotes

Esta seção inclui alguns cenários comuns quando uma versão de pacote é adicionada a um CodeArtifact repositório. As configurações de controle de origem do pacote serão definidas para novos pacotes, dependendo de como a primeira versão do pacote for adicionada.

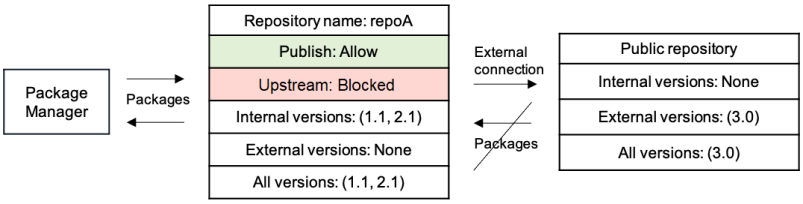
Nos cenários a seguir, um pacote interno é um pacote publicado diretamente de um gerenciador de pacotes no seu repositório, como um pacote que você ou sua equipe cria e mantém. Um pacote externo é um pacote que existe em um repositório público que pode ser ingerido em seu repositório com uma conexão externa.

Uma versão de pacote externo é publicada para um pacote interno existente

Nesse cenário, considere um pacote interno, packageA. Sua equipe publica a primeira versão do pacote do PackageA em um repositório. CodeArtifact Como essa é a primeira versão desse pacote, as configurações de controle de origem do pacote são automaticamente definidas como Publicar: Permitir e Upstream: Bloquear. Depois que o pacote existe no seu repositório, um pacote com o

mesmo nome é publicado em um repositório público conectado ao seu CodeArtifact repositório. Isso pode ser uma tentativa de ataque de substituição de dependência contra o pacote interno ou pode ser apenas uma coincidência. Independentemente disso, os controles de origem do pacote são configurados para bloquear a ingestão da nova versão externa para se protegerem contra um possível ataque.

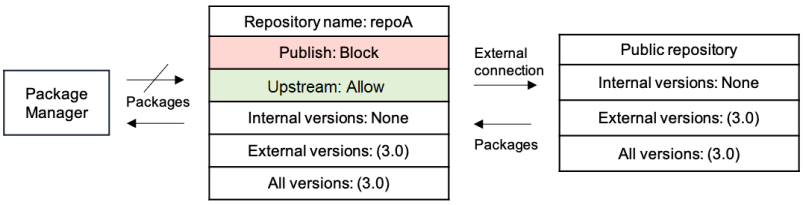
Na imagem a seguir, o RepoA é seu CodeArtifact repositório com uma conexão externa com um repositório público. Seu repositório contém as versões 1.1 e 2.1 de packageA, mas a versão 3.0 é publicada no repositório público. Normalmente, repoA poderia ingerir a versão 3.0 depois que o pacote fosse solicitado por um gerenciador de pacotes. Como a ingestão de pacotes está definida como Bloquear, a versão 3.0 não é ingerida em seu CodeArtifact repositório e não está disponível para gerenciadores de pacotes conectados a ele.



Uma versão de pacote interno é publicada para um pacote externo existente

Nesse cenário, um pacote, packageB, existe externamente em um repositório público que você conectou ao seu repositório. Quando um gerenciador de pacotes conectado ao seu repositório solicita packageB, a versão do pacote é ingerida no seu repositório a partir do repositório público. Como esta é a primeira versão do pacote de packageB adicionada ao seu repositório, as configurações de origem do pacote são definidas como Publicação: BLOQUEAR e Upstream: PERMITIR. Mais tarde, você tenta publicar uma versão com o mesmo nome de pacote no repositório. Ou você não conhece o pacote público e está tentando publicar um pacote não relacionado com o mesmo nome, ou está tentando publicar uma versão corrigida ou está tentando publicar diretamente a versão exata do pacote que já existe externamente. CodeArtifact rejeitará a versão que você está tentando publicar, mas permitirá que você anule explicitamente a rejeição e publique a versão, se necessário.

Na imagem a seguir, o RepoA é seu CodeArtifact repositório com uma conexão externa com um repositório público. Seu repositório contém a versão 3.0 que foi ingerida do repositório público. Você deseja publicar a versão 1.1 no seu repositório. Normalmente, você poderia publicar a versão 1.2 em repoA, mas como a publicação está definida como Bloquear, a versão 1.2 não pode ser publicada.



Publicar uma versão corrigida de um pacote externo existente

Nesse cenário, um pacote, packageB, existe externamente em um repositório público que você conectou ao seu repositório. Quando um gerenciador de pacotes conectado ao seu repositório solicita packageB, a versão do pacote é ingerida no seu repositório a partir do repositório público. Como esta é a primeira versão do pacote de packageB adicionada ao seu repositório, as configurações de origem do pacote são definidas como Publicação: BLOQUEAR e Upstream: PERMITIR. Sua equipe decide que precisa publicar versões corrigidas desse pacote no repositório. Para poder publicar versões de pacotes diretamente, sua equipe altera as configurações de controle de origem do pacote para Publicação: PERMITIR e Upstream: BLOQUEAR. Agora, as versões desse pacote podem ser publicadas diretamente no seu repositório e ingeridas de repositórios públicos. Depois que sua equipe publica as versões corrigidas do pacote, ela reverte as configurações de origem do pacote para Publicação: BLOQUEAR e Upstream: PERMITIR.

Configurações de controle de origem do pacote

Com os controles de origem do pacote, você pode configurar como as versões do pacote podem ser adicionadas a um repositório. As listas a seguir incluem as configurações e os valores disponíveis do controle de origem do pacote.

> **Note**
>
> As configurações e valores disponíveis são diferentes ao configurar os controles de origem em grupos de pacotes. Para obter mais informações, consulte Controles de origem do grupo de pacotes.

Publicar

Essa configuração define se as versões do pacote podem ser publicadas diretamente no repositório usando gerenciadores de pacotes ou ferramentas similares.

- PERMITIR: as versões do pacote podem ser publicadas diretamente.

- BLOCK: as versões do pacote não podem ser publicadas diretamente.

Upstream

Essa configuração define se as versões do pacote podem ser ingeridas de repositórios externos públicos ou retidas de repositórios upstream quando solicitadas por um gerenciador de pacotes.

- PERMITIR: Qualquer versão do pacote pode ser retida de outros CodeArtifact repositórios configurados como repositórios upstream ou ingerida de uma fonte pública com uma conexão externa.
- BLOCO: As versões do pacote não podem ser retidas de outros CodeArtifact repositórios configurados como repositórios upstream ou ingeridas de uma fonte pública com uma conexão externa.

Configurações de controle de origem do pacote padrão

As configurações de controle de origem do pacote padrão são definidas com base nas configurações de controle de origem do grupo de pacotes associado ao pacote. Para obter mais informações sobre grupos de pacotes e controles de origem de grupos de pacotes, consulte Trabalhar com grupos de pacotes no CodeArtifact e Controles de origem do grupo de pacotes.

Se um pacote estiver associado a um grupo de pacotes com configurações de restrição de `ALLOW` para cada tipo de restrição, os controles de origem de pacote padrão serão baseados em como a primeira versão desse pacote é adicionada ao repositório.

- Se a primeira versão do pacote for publicada diretamente por um gerenciador de pacotes, as configurações serão Publicação: PERMITIR e Upstream: BLOQUEAR.
- Se a primeira versão do pacote for ingerida de uma fonte pública, as configurações serão Publicação: BLOQUEAR e Upstream: PERMITIR.

> **Note**
>
> Pacotes que existiam em CodeArtifact repositórios antes de maio de 2022 terão os controles de origem de pacote padrão de Publish: ALLOW e Upstream: ALLOW. Os controles de origem do pacote devem ser definidos manualmente para esses pacotes. Os valores padrão atuais foram definidos em novos pacotes desde aquela época e começaram a ser aplicados quando o atributo foi lançado em 14 de julho de 2022. Para obter mais informações sobre

a configuração de controles de origem de pacotes, consulte Editar controles de origem do pacote.

Caso contrário, se um pacote estiver associado a um grupo de pacotes que tenha pelo menos uma configuração de restrição de `BLOCK` ou `ALLOW_SPECIFIC_REPOSITORIES`, as configurações de controle de origem padrão desse pacote serão definidas como Publish: ALLOW e Upstream: ALLOW.

Como os controles de origem do pacote interagem com os controles de origem do grupo de pacotes

Como tanto os pacotes quanto os grupos de pacotes associados têm configurações de controle de origem, é importante entender como essas duas configurações diferentes interagem entre si.

A interação entre as duas configurações é que uma configuração de `BLOCK` sempre prevalece sobre uma configuração de `ALLOW`. A tabela a seguir lista alguns exemplos de configurações e suas configurações de controle de origem eficazes.

Configuração de controle de origem do pacote	Configuração de controle de origem do grupo de pacotes	Configuração de controle de origem eficaz
PUBLISH: ALLOW UPSTREAM: ALLOW	PUBLISH: ALLOW UPSTREAM: ALLOW	PUBLISH: ALLOW UPSTREAM: ALLOW
PUBLISH: BLOCK UPSTREAM: ALLOW	PUBLISH: ALLOW UPSTREAM: ALLOW	PUBLISH: BLOCK UPSTREAM: ALLOW
PUBLISH: ALLOW UPSTREAM: ALLOW	PUBLISH: ALLOW UPSTREAM: BLOCK	PUBLISH: ALLOW UPSTREAM: BLOCK

Isso significa que um pacote com configurações de origem de Publish: ALLOW e Upstream: ALLOW está efetivamente adotando as configurações de controle de origem do grupo de pacotes associado.

Editar controles de origem do pacote

Os controles de origem do pacote são configurados automaticamente com base em como a primeira versão de um pacote é adicionada ao repositório. Para obter mais informações, consulte Configurações de controle de origem do pacote padrão. Para adicionar ou editar controles de origem de pacote para um pacote em um CodeArtifact repositório, execute as etapas no procedimento a seguir.

Para adicionar ou editar controles de origem do pacote (console)

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, escolha Repositórios e escolha o repositório que contém o pacote que você deseja editar.
3. Na tabela Pacotes, pesquise e selecione o pacote que você deseja editar.
4. Na página de resumo do pacote, em Controles de origem, escolha Editar.
5. Em Editar controles de origem, escolha os controles de origem do pacote que deseja definir para esse pacote. As duas configurações de controle de origem do pacote, Publicação e Upstream, devem ser definidas ao mesmo tempo.

   - Para permitir a publicação direta de versões do pacote, em Publicação, escolha Permitir. Para bloquear a publicação de versões do pacote, escolha Bloquear.
   - Para permitir a ingestão de pacotes de repositórios externos e a extração de pacotes de repositórios upstream, em Fontes upstream, escolha Permitir. Para bloquear toda a ingestão e extração de versões de pacotes de repositórios externos e upstream, escolha Bloquear.

Para adicionar ou editar controles de origem do pacote (AWS CLI)

1. Caso contrário, configure o AWS CLI seguindo as etapas emConfigurando com AWS CodeArtifact.
2. Use o comando `put-package-origin-configuration` para adicionar ou editar controles de origem do pacote. Substitua os campos a seguir:

   - *`my_domain`*Substitua pelo CodeArtifact domínio que contém o pacote que você deseja atualizar.
   - *`my_repo`*Substitua pelo CodeArtifact repositório que contém o pacote que você deseja atualizar.

- *npm*Substitua pelo formato do pacote que você deseja atualizar.
- *my_package*Substitua pelo nome do pacote que você deseja atualizar.
- Substitua *ALLOW* e *BLOCK* pelas configurações de controle de origem do pacote desejadas.

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

Repositórios de publicação e upstream

CodeArtifact não permite publicar versões de pacotes que estejam presentes em repositórios upstream acessíveis ou em repositórios públicos. Por exemplo, suponha que você queira publicar um pacote Maven `com.mycompany.mypackage:1.0` em um repositório `myrepo` e `myrepo` tenha um repositório upstream com uma conexão externa com Maven Central. Considere os seguintes cenários:

1. As configurações de controle de origem do pacote em `com.mycompany.mypackage` são Publicação: PERMITIR e Upstream: PERMITIR. Se `com.mycompany.mypackage:1.0` estiver presente no repositório upstream ou no Maven Central, CodeArtifact rejeita qualquer tentativa de publicar nele `myrepo` com um erro de conflito 409. Você ainda pode publicar uma versão diferente, como `com.mycompany.mypackage:1.1`.
2. As configurações de controle de origem do pacote em `com.mycompany.mypackage` são Publicação: PERMITIR e Upstream: BLOQUEAR. Você pode publicar qualquer versão do `com.mycompany.mypackage` no seu repositório que ainda não exista porque as versões do pacote não estão acessíveis.
3. As configurações de controle de origem do pacote em `com.mycompany.mypackage` são Publicação: BLOQUEAR e Upstream: PERMITIR. Você não pode publicar nenhuma versão do pacote diretamente no seu repositório.

Trabalhar com grupos de pacotes no CodeArtifact

Os grupos de pacotes podem ser usados para aplicar configurações a vários pacotes que correspondam a um padrão definido usando o formato, o namespace e o nome do pacote. Você pode usar grupos de pacotes para configurar de forma mais conveniente os controles de origem de pacotes para vários pacotes. Os controles de origem de pacotes são usados para bloquear ou permitir a ingestão ou publicação de novas versões de pacotes, protegendo os usuários de ações maliciosas conhecidas como ataques de substituição de dependências.

Cada domínio no CodeArtifact contém automaticamente um grupo de pacotes raiz. Esse grupo de pacotes raiz, `/*`, contém todos os pacotes e permite, por padrão, que versões de pacotes entrem em repositórios no domínio independentemente do tipo de origem. O grupo de pacotes raiz pode ser modificado, mas não pode ser excluído.

O recurso de configuração de grupo de pacotes opera de maneira eventualmente consistente ao criar um novo grupo de pacotes ou excluir um grupo de pacotes existente. Isso significa que, ao criar ou excluir um grupo de pacotes, os controles de origem serão aplicados aos pacotes associados esperados, mas com algum atraso devido ao eventual comportamento consistente. O tempo para alcançar a consistência eventual depende do número de grupos de pacotes no domínio, bem como do número de pacotes no domínio. Pode haver um breve período em que os controles de origem não sejam refletidos imediatamente nos pacotes associados após a criação ou exclusão de um grupo de pacotes.

Além disso, as atualizações nos controles de origem do grupo de pacotes entram em vigor quase imediatamente. Diferentemente da criação ou exclusão de grupos de pacotes, as alterações nos controles de origem de um grupo de pacotes existente são refletidas nos pacotes associados sem o mesmo atraso.

Os tópicos a seguir contêm informações sobre grupos de pacotes no AWS CodeArtifact.

Tópicos

- Marcar um grupo de pacotes no CodeArtifact

Criar um grupo de pacotes

É possível criar um grupo de pacotes usando o console do CodeArtifact, a AWS Command Line Interface (AWS CLI) ou o CloudFormation. Consulte mais informações sobre o gerenciamento de grupos de pacotes do CodeArtifact com o CloudFormation em Criação de recursos do CodeArtifact com AWS CloudFormation.

Criar um grupo de pacotes (console)

1. Abra o console do AWS CodeArtifact em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, selecione Domínios e escolha o domínio no qual você deseja criar um grupo de pacotes.
3. Selecione Grupos de pacotes e clique em Criar grupo de pacotes.
4. Em Definição de grupos de pacotes, insira a definição do grupo de pacotes para o seu grupo de pacotes. A definição do grupo de pacotes determina quais pacotes estão associados ao grupo. Você pode inserir a definição do grupo de pacotes manualmente com texto ou usar o modo visual para fazer seleções e a definição do grupo de pacotes será criada automaticamente.
5. Para usar o modo visual para criar a definição do grupo de pacotes:

    a. Selecione Visual para alternar para o modo visual.

    b. Em Formato do pacote, escolha o formato dos pacotes a serem associados a esse grupo.

    c. Em Namespace (escopo), escolha os critérios de namespace a serem correspondidos.

    - Igual a: corresponda exatamente o namespace especificado. Se escolhido, insira o namespace a ser correspondido.
    - Em branco: corresponda pacotes sem namespace.
    - Começa com a palavra: corresponda namespaces que começam com uma palavra especificada. Se escolhido, insira a palavra de prefixo a ser correspondida. Para obter mais informações sobre palavras e limites de palavras, consulte Palavras, limites de palavras e correspondência de prefixos.
    - Todos: corresponda pacotes em todos os namespaces.

d. Se a opção Igual a, Em branco ou Começa com a palavra estiver selecionada, em Nome do pacote, escolha os critérios do nome do pacote a serem correspondidos.

- Exatamente igual a: corresponda exatamente o nome do pacote especificado. Se escolhido, insira o nome do pacote a ser correspondido.
- Começa com o prefixo: corresponda pacotes que começam com o prefixo especificado.
- Começa com a palavra: corresponda pacotes que começam com uma palavra especificada. Se escolhido, insira a palavra de prefixo a ser correspondida. Para obter mais informações sobre palavras e limites de palavras, consulte Palavras, limites de palavras e correspondência de prefixos.
- Todos: corresponda todos os pacotes.

e. Clique em Próximo para revisar a definição.

6. Para inserir a definição do grupo de pacotes com texto:

a. Clique em Texto para alternar para o modo de texto.

b. Em Definição do grupo de pacotes, insira a definição do grupo de pacotes. Para obter mais informações sobre a sintaxe de definição de grupo de pacotes, consulte Sintaxe de definição de grupos de pacotes e comportamento de correspondência.

c. Clique em Próximo para revisar a definição.

7. Em Revisar definição, revise os pacotes que serão incluídos no novo grupo de pacotes com base na definição fornecida anteriormente. Após a revisão, clique em Próximo.

8. Em Informações do grupo de pacotes, opcionalmente, adicione uma descrição e e-mail de contato para o grupo de pacotes. Escolha Próximo.

9. Em Controles de origem de pacotes, configure os controles de origem a serem aplicados aos pacotes no grupo. Para obter mais informações sobre controles de origem do grupo de pacotes, consulte Controles de origem do grupo de pacotes.

10. Selecione Criar grupo de pacotes.

Criar um grupo de pacotes (AWS CLI)

Use o comando `create-package-group` para criar um grupo de pacotes no seu domínio. Para a opção `--package-group`, insira a definição do grupo de pacotes que determina quais pacotes estão associados ao grupo. Para obter mais informações sobre a sintaxe de definição de grupo de pacotes, consulte Sintaxe de definição de grupos de pacotes e comportamento de correspondência.

Caso não tenha feito isso, configure a AWS CLI seguindo as etapas emConfigurando com AWS CodeArtifact.

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

Visualizar ou editar um grupo de pacotes

Você pode ver uma lista de todos os grupos de pacotes, ver detalhes de um grupo de pacotes específico ou editar os detalhes ou a configuração de um grupo de pacotes usando o console CodeArtifact ou a AWS Command Line Interface (AWS CLI).

Visualizar ou editar um grupo de pacotes (console)

1. Abra o console do AWS CodeArtifact em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, selecione Domínios e escolha o domínio que contém o grupo de pacotes que você deseja visualizar ou editar.
3. Selecione Grupos de pacotes e escolha o grupo de pacotes que você deseja visualizar ou editar.
4. Em Detalhes, visualize informações sobre o grupo de pacotes, incluindo seu grupo principal, descrição, ARN, e-mail de contato e controles de origem do pacote.
5. Em Subgrupos, veja uma lista de grupos de pacotes que têm esse grupo como grupo principal. Os grupos de pacotes dessa lista podem herdar as configurações desse grupo de pacotes. Para obter mais informações, consulte Hierarquia de grupos de pacotes e especificidade de padrões.
6. Em Pacotes, visualize os pacotes que pertencem a esse grupo de pacotes com base na definição do grupo de pacotes. Na coluna Força, você pode ver a força da associação do pacote. Para obter mais informações, consulte Hierarquia de grupos de pacotes e especificidade de padrões.
7. Para editar as informações do grupo de pacotes, selecione Editar grupo de pacotes.

    a. Em Informações, atualize as informações de contato ou a descrição do grupo de pacotes. Você não pode editar a definição de um grupo de pacotes.

b. Em Controles de origem do grupo de pacotes, atualize as configurações de controle de origem do grupo de pacotes, que determinam como os pacotes associados podem entrar nos repositórios no domínio. Para obter mais informações, consulte Controles de origem do grupo de pacotes.

Visualizar ou editar um grupo de pacotes (AWS CLI)

Use os comandos a seguir para visualizar ou editar grupos de pacotes com a AWS CLI. Caso não tenha feito isso, configure a AWS CLI seguindo as etapas emConfigurando com AWS CodeArtifact.

Para visualizar todos os grupos de pacotes em um domínio, use o comando `list-package-groups`.

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

Para visualizar os detalhes sobre um grupo de pacotes, use o comando `describe-package-group`. Para obter mais informações sobre as definições de grupo de pacotes, consulte Sintaxe de definição de grupos de pacotes e exemplos.

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

Para visualizar os grupos de pacotes secundários de um grupo de pacotes, use o comando `list-sub-package-groups`.

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

Para visualizar o grupo de pacotes associado a um pacote, use o comando `get-associated-package-group`. Você deve usar o nome e o namespace normalizados do pacote para os formatos de pacote NuGet, Python e Swift. Para obter mais informações sobre como os nomes e namespaces dos pacotes são normalizados, consulte a documentação de normalização de nomes do NuGet, Python e Swift.

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

Para editar um grupo de pacotes, use o comando `update-package-group`. Esse comando é usado para atualizar as informações de contato ou a descrição de um grupo de pacotes. Para obter informações sobre as configurações de controle de origem do grupo de pacotes e sobre como adicioná-las ou editá-las, consulte Controles de origem do grupo de pacotes. Para obter mais informações sobre as definições de grupo de pacotes, consulte Sintaxe de definição de grupos de pacotes e exemplos

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

Excluir um grupo de pacotes

Você pode excluir um grupo de pacotes usando o console do CodeArtifact ou a AWS Command Line Interface (AWS CLI).

Observe o seguinte comportamento ao excluir grupos de pacote:

- O grupo de pacotes raiz, `/*`, não pode ser excluído.
- Os pacotes e as versões de pacotes associados a esse grupo de pacotes não são excluídos.
- Quando um grupo de pacotes é excluído, os grupos de pacotes secundários diretos se tornam secundários do grupo de pacotes principal direto do grupo de pacotes. Portanto, se algum dos grupos secundários estiver herdando alguma configuração do grupo principal, essas configurações poderão mudar.

Excluir um grupo de pacotes (console)

1. Abra o console do AWS CodeArtifact em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, selecione Domínios e escolha o domínio que contém o grupo de pacotes que você deseja visualizar ou editar.
3. Selecione Grupos de pacotes.
4. Escolha o grupo de pacotes que deseja excluir e clique em Excluir.
5. Insira excluir no campo e clique em Excluir.

Excluir um grupo de pacotes (AWS CLI)

Para excluir um grupo de pacotes, use o comando `delete-package-group`.

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

Controles de origem do grupo de pacotes

Os controles de origem do pacote são usados para configurar como as versões do pacote podem entrar em um domínio. Você pode configurar controles de origem em um grupo de pacotes para configurar como as versões de cada pacote associado ao grupo de pacotes podem entrar em repositórios específicos no domínio.

As configurações de controle de origem do grupo de pacotes consistem no seguinte:

- Configurações de restrição: essas configurações definem se os pacotes podem entrar em um repositório no CodeArtifact por meio de publicação, upstreams internos ou repositórios públicos externos.
- Listas de repositórios permitidos: cada configuração de restrição pode ser definida para permitir repositórios específicos. Se uma configuração de restrição for definida para permitir repositórios específicos, essa restrição terá uma lista correspondente de repositórios permitidos.

> **Note**
>
> As configurações de controle de origem para grupos de pacotes são ligeiramente diferentes das configurações de controle de origem para pacotes individuais. Para obter mais informações sobre as configurações de controle de origem para pacotes, consulte Configurações de controle de origem do pacote.

Configurações de restrição

As configurações de restrição das configurações de controle de origem de um grupo de pacotes determinam como os pacotes associados a esse grupo podem entrar nos repositórios no domínio.

PUBLISH

A configuração `PUBLISH` define se as versões do pacote podem ser publicadas diretamente em qualquer repositório no domínio usando gerenciadores de pacotes ou ferramentas similares.

- ALLOW: as versões do pacote podem ser publicadas diretamente em todos os repositórios.
- BLOCK: as versões do pacote não podem ser publicadas diretamente em nenhum repositório.
- ALLOW_SPECIFIC_REPOSITORIES: as versões do pacote só podem ser publicadas diretamente nos repositórios especificados na lista de repositórios permitidos para publicação.
- INHERIT: a configuração `PUBLISH` é herdada do primeiro grupo de pacotes principal com uma configuração que não é `INHERIT`.

EXTERNAL_UPSTREAM

A configuração `EXTERNAL_UPSTREAM` define se as versões do pacote podem ser ingeridas de repositórios externos públicos quando solicitadas por um gerenciador de pacotes. Para ver uma lista dos repositórios externos, consulte Repositórios de conexão externa compatíveis.

- ALLOW: qualquer versão do pacote pode ser ingerida em todos os repositórios de uma fonte pública com uma conexão externa.
- BLOCK: as versões do pacote não podem ser ingeridas em nenhum repositório de uma fonte pública com uma conexão externa.
- ALLOW_SPECIFIC_REPOSITORIES: as versões do pacote só podem ser ingeridas de uma fonte pública em repositórios especificados na lista de repositórios permitidos para upstreams externos.

- INHERIT: a configuração `EXTERNAL_UPSTREAM` é herdada do primeiro grupo de pacotes principal com uma configuração que não é `INHERIT`.

INTERNAL_UPSTREAM

A configuração `INTERNAL_UPSTREAM` define se as versões do pacote podem ser retidas de repositórios upstream internos no mesmo domínio do CodeArtifact quando solicitadas por um gerenciador de pacotes.

- ALLOW: qualquer versão do pacote pode ser retida de outros repositórios do CodeArtifact configurados como repositórios upstream.
- BLOCK: as versões do pacote não podem ser retidas de outros repositórios do CodeArtifact configurados como repositórios upstream.
- ALLOW_SPECIFIC_REPOSITORIES: as versões do pacote só podem ser retidas de outros repositórios do CodeArtifact configurados como repositórios upstream em repositórios especificados na lista de repositórios permitidos para upstreams internos.
- INHERIT: a configuração `INTERNAL_UPSTREAM` é herdada do primeiro grupo de pacotes principal com uma configuração que não é `INHERIT`.

Listas de repositórios permitidos

Quando uma configuração de restrição é definida como `ALLOW_SPECIFIC_REPOSITORIES`, o grupo de pacotes passa a incluir uma lista de repositórios permitidos, que contém os repositórios permitidos para essa configuração de restrição. Portanto, um grupo de pacotes contém de 0 a 3 listas de repositórios permitidos, uma para cada configuração definida como `ALLOW_SPECIFIC_REPOSITORIES`.

Ao adicionar um repositório à lista de repositórios permitidos de um grupo de pacotes, é necessário especificar em qual lista adicioná-lo.

As possíveis listas de repositórios permitidos são as seguintes:

- `EXTERNAL_UPSTREAM`: permite ou bloqueia a ingestão de versões de pacotes de repositórios externos no repositório adicionado.
- `INTERNAL_UPSTREAM`: permite ou bloqueia a extração de versões de pacotes de outro repositório do CodeArtifact no repositório adicionado.

- `PUBLISH`: permite ou bloqueia a publicação direta de versões de pacotes dos gerenciadores de pacotes no repositório adicionado.

Editar configurações de controle de origem do grupo de pacotes

Para adicionar ou editar controles de origem de um grupo de pacotes, execute as etapas no procedimento a seguir. Para obter informações sobre as configurações de controle de origem do grupo de pacotes, consulte Configurações de restrição e Listas de repositórios permitidos.

Como adicionar ou editar controles de origem do grupo de pacotes (CLI)

1. Caso não tenha feito isso, configure a AWS CLI seguindo as etapas emConfigurando com AWS CodeArtifact.
2. Use o comando `update-package-group-origin-configuration` para adicionar ou editar controles de origem do pacote.

   - Para `--domain`, insira o domínio do CodeArtifact que contém o grupo de pacotes que deseja atualizar.
   - Para `--domain-owner`, insira o número da conta do proprietário do domínio.
   - Para `--package-group`, insira o grupo de pacotes que deseja atualizar.
   - Para `--restrictions`, insira pares de chave-valor que representam as restrições de controle de origem.
   - Para `--add-allowed-repositories`, insira um objeto JSON contendo o tipo de restrição e o nome do repositório para adicionar à lista correspondente de repositórios permitidos para a restrição.
   - Para `--remove-allowed-repositories`, insira um objeto JSON contendo o tipo de restrição e o nome do repositório a serem removidos da lista correspondente de repositórios permitidos para a restrição.

```
aws codeartifact update-package-group-origin-configuration \
   --domain my_domain \
   --domain-owner 111122223333 \
   --package-group '/nuget/*' \
   --restrictions INTERNAL_UPSTREAM=ALLOW_SPECIFIC_REPOSITORIES \
   --add-allowed-repositories
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo \
```

```
   --remove-allowed-repositories
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo2
```

O exemplo a seguir adiciona várias restrições e vários repositórios em um comando.

```
aws codeartifact update-package-group-origin-configuration \
   --domain my_domain \
   --domain-owner 111122223333 \
   --package-group '/nuget/*' \
   --
restrictions PUBLISH=BLOCK,EXTERNAL_UPSTREAM=ALLOW_SPECIFIC_REPOSITORIES,INTERNAL_UPSTREAM=
 \
   --add-allowed-repositories
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo2 \
   --remove-allowed-repositories
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=my_repo2
```

Exemplos de configuração do controle de origem do grupo de pacotes

Os exemplos a seguir mostram configurações de controle de origem de pacotes para cenários comuns de gerenciamento de pacotes.

Permitir que pacotes com nomes privados sejam publicados, mas não ingeridos

Este cenário provavelmente é comum no gerenciamento de pacotes:

- Permita que pacotes com nomes privados sejam publicados em repositórios em seu domínio por meio de gerenciadores de pacotes e impeça que sejam ingeridos em repositórios em seu domínio por meio de repositórios públicos externos.
- Permita que todos os outros pacotes sejam ingeridos em repositórios em seu domínio por meio de repositórios públicos externos e impeça que sejam publicados em repositórios em seu domínio por meio de gerenciadores de pacotes.

Para fazer isso, você deve configurar um grupo de pacotes com um padrão que inclua os nomes privados e as configurações de origem de PUBLISH: ALLOW, EXTERNAL_UPSTREAM: BLOCK e INTERNAL_UPSTREAM: ALLOW. Isso garantirá que pacotes com nomes privados possam ser publicados diretamente, mas não possam ser ingeridos de repositórios externos.

Os seguintes comandos da AWS CLI criam e configuram um grupo de pacotes com configurações de restrição de origem que correspondem ao comportamento desejado:

Para criar o grupo de pacotes:

```
aws codeartifact create-package-group \
   --domain my_domain \
   --package-group /npm/space/anycompany~ \
   --domain-owner 111122223333 \
   --contact-info contact@email.com | URL \
   --description "my package group"
```

Para atualizar a configuração de origem do grupo de pacotes:

```
aws codeartifact update-package-group-origin-configuration \
   --domain my_domain \
   --domain-owner 111122223333 \
   --package-group '/npm/space/anycompany~' \
   --restrictions PUBLISH=ALLOW,EXTERNAL_UPSTREAM=BLOCK,INTERNAL_UPSTREAM=ALLOW
```

Permitir a ingestão de repositórios externos por meio de um repositório

Neste cenário, seu domínio tem vários repositórios. Desses repositórios, `repoA` tem uma conexão upstream com `repoB`, que tem uma conexão externa com o repositório público, `npmjs.com`, conforme mostrado:

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

Você deseja permitir a ingestão de pacotes de um grupo de pacotes específico, `/npm/space/anycompany~` de `npmjs.com` para `repoA`, mas somente por meio de `repoB`. Você também deseja bloquear a ingestão de pacotes associados ao grupo de pacotes em qualquer outro repositório em seu domínio e bloquear a publicação direta de pacotes com gerenciadores de pacotes. Para fazer isso, você cria e configura o grupo de pacotes da seguinte forma:

Configurações de restrição de origem de PUBLISH: BLOCK, EXTERNAL_UPSTREAM: ALLOW_SPECIFIC_REPOSITORIES e INTERNAL_UPSTREAM: ALLOW_SPECIFIC_REPOSITORIES.

`repoA` e `repoB` adicionados à lista apropriada de repositórios permitidos:

- `repoA` deve ser adicionado à lista `INTERNAL_UPSTREAM`, pois obterá pacotes de seu upstream interno, `repoB`.
- `repoB` deve ser adicionado à lista `EXTERNAL_UPSTREAM`, pois obterá pacotes do repositório externo, `npmjs.com`.

Os seguintes comandos da AWS CLI criam e configuram um grupo de pacotes com configurações de restrição de origem que correspondem ao comportamento desejado:

Para criar o grupo de pacotes:

```
aws codeartifact create-package-group \
   --domain my_domain \
   --package-group /npm/space/anycompany~ \
   --domain-owner 111122223333 \
   --contact-info contact@email.com | URL \
   --description "my package group"
```

Para atualizar a configuração de origem do grupo de pacotes:

```
aws codeartifact update-package-group-origin-configuration \
   --domain my_domain \
   --domain-owner 111122223333 \
   --package-group /npm/space/anycompany~ \
   --
restrictions PUBLISH=BLOCK,EXTERNAL_UPSTREAM=ALLOW_SPECIFIC_REPOSITORIES,INTERNAL_UPSTREAM=ALLO
 \
   --add-allowed-repositories
 originRestrictionType=INTERNAL_UPSTREAM,repositoryName=repoA
 originRestrictionType=EXTERNAL_UPSTREAM,repositoryName=repoB
```

Como as configurações de controle de origem do grupo de pacotes interagem com as configurações de controle de origem do pacote

Como tanto os pacotes quanto os grupos de pacotes associados têm configurações de controle de origem, é importante entender como essas duas configurações diferentes interagem entre si. Para obter mais informações sobre a interação entre as configurações, consulte Como os controles de origem do pacote interagem com os controles de origem do grupo de pacotes.

Sintaxe de definição de grupos de pacotes e comportamento de correspondência

Este tópico contém informações sobre como definir grupos de pacotes, comportamento de correspondência de padrões, força de associação de pacotes e hierarquia de grupos de pacotes.

Sumário

Sintaxe de definição de grupos de pacotes e exemplos

A sintaxe padrão para definir grupos de pacotes segue de perto a formatação dos caminhos de pacotes. Um caminho de pacote é criado com base nos componentes de coordenadas de um pacote (formato, namespace e nome), adicionando uma barra no início e separando cada um dos componentes com uma barra. Por exemplo, o caminho do pacote npm chamado anycompany-ui-components no namespace space é /npm/space/anycompany-ui-components.

Um padrão de grupo de pacotes segue a mesma estrutura de um caminho de pacote, exceto que os componentes que não são especificados como parte da definição do grupo são omitidos e o padrão é finalizado com um sufixo. O sufixo incluído determina o comportamento correspondente do padrão, da seguinte forma:

- Um sufixo $ corresponderá à coordenada completa do pacote.
- Um sufixo ~ corresponderá a um prefixo.
- Um sufixo * corresponderá a todos os valores do componente definido anteriormente.

Veja abaixo exemplos de padrões para cada uma das combinações permitidas:

1. Todos os formatos de pacote: `/*`
2. Formato de pacote específico: `/npm/*`
3. Formato e prefixo do namespace do pacote: `/maven/com.anycompany~`
4. Formato e namespace do pacote: `/npm/space/*`
5. Formato, namespace e prefixo do nome do pacote: `/npm/space/anycompany-ui~`
6. Formato, namespace e nome do pacote: `/maven/org.apache.logging.log4j/log4j-core$`

Conforme mostrado nos exemplos acima, o sufixo ~ é adicionado ao final de um namespace ou nome para representar uma correspondência de prefixo, e * é adicionado após uma barra quando usado para corresponder a todos os valores do próximo componente no caminho (todos os formatos, todos os namespaces ou todos os nomes).

Definição e normalização de grupos de pacotes

O CodeArtifact normaliza os nomes de pacotes NuGet, Python e Swift, e normaliza os namespaces dos pacotes Swift antes de armazená-los. O CodeArtifact usa esses nomes normalizados ao corresponder pacotes com definições de grupos de pacotes. Portanto, grupos de pacotes que contêm um namespace ou nome nesses formatos devem usar o namespace e o nome normalizados. Para obter mais informações sobre como os nomes e namespaces dos pacotes são normalizados, consulte a documentação de normalização de nomes do NuGet, Python e Swift.

Namespaces nas definições de grupos de pacotes

Para pacotes ou formatos de pacotes sem um namespace (Python e NuGet), os grupos de pacotes não devem conter um namespace. A definição do grupo de pacotes para esses grupos de pacotes contém uma seção de namespace em branco. Por exemplo, o caminho para o pacote Python chamado requests é /python//requests.

Para pacotes ou formatos de pacotes com um namespace (Maven, genérico e Swift), o namespace deverá ser incluído se o nome do pacote estiver incluído. Para o formato de pacote Swift, o namespace normalizado do pacote será usado. Para obter mais informações sobre como os namespaces de pacote Swift são normalizados, consulte Normalização do nome e do namespace do pacote Swift.

Hierarquia de grupos de pacotes e especificidade de padrões

Os pacotes que estão “em” ou “associados a” um grupo de pacotes são pacotes com um caminho de pacote que corresponde ao padrão do grupo, mas não corresponde ao padrão de um grupo mais específico. Por exemplo, dados os grupos de pacotes `/npm/*` e `/npm/space/*`, o caminho do pacote /npm//react está associado ao primeiro grupo (`/npm/*`), enquanto /npm/space/aui.components e /npm/space/amplify-ui-core estão associados ao segundo grupo (`/npm/space/*`). Embora um pacote possa corresponder a vários grupos, cada pacote está associado somente a um único grupo, a correspondência mais específica, e somente a configuração desse grupo se aplica ao pacote.

Quando um caminho de pacote corresponde a vários padrões, o padrão “mais específico” pode ser considerado o padrão de correspondência mais longo. Alternativamente, o padrão mais específico é aquele que corresponde a um subconjunto próprio dos pacotes que correspondem ao padrão menos específico. Em exemplo anterior, todo pacote que corresponde a `/npm/space/*` também corresponde a `/npm/*`, mas o inverso não é verdadeiro, o que torna `/npm/space/*` o padrão mais específico porque é um subconjunto próprio de `/npm/*`. Como um grupo é um subconjunto de outro grupo, cria-se uma hierarquia, na qual `/npm/space/*` é um subgrupo do grupo principal, `/npm/*`.

Embora somente a configuração mais específica do grupo de pacotes se aplique a um pacote, esse grupo pode ser configurado para herdar da configuração do grupo principal.

Palavras, limites de palavras e correspondência de prefixos

Antes de discutir a correspondência de prefixos, vamos definir alguns termos-chave:

- Uma palavra é uma letra ou número seguido por zero ou mais letras, números ou caracteres diacríticos (como acentos, tremas etc.).
- O limite de uma palavra ocorre no final de uma palavra, quando um caractere que não é uma palavra é atingido. Caracteres que não são palavras são caracteres de pontuação, como `.`, `-` e `_`.

Especificamente, o padrão regex para uma palavra é `[\p{L}\p{N}][\p{L}\p{N}\p{M}]*`, que pode ser detalhado da seguinte forma:

- `\p{L}` representa qualquer letra.
- `\p{N}` representa qualquer número.
- `\p{M}` representa qualquer caractere diacrítico, como acentos, tremas etc.

Portanto, `[\p{L}\p{N}]` representa um número ou letra, e `[\p{L}\p{N}\p{M}]*` representa zero ou mais letras, números ou caracteres diacríticos, e um limite de palavra está no final de cada correspondência desse padrão regex.

> **Note**
>
> A correspondência de limites de palavras é baseada nessa definição de "palavra". Não é baseada em palavras definidas em um dicionário ou em camelCase. Por exemplo, não há limite de palavras em `oneword` ou `OneWord`.

Agora que a palavra e o limite da palavra estão definidos, podemos usá-los para descrever a correspondência de prefixos no CodeArtifact. Para indicar uma correspondência de prefixo no limite de uma palavra, um caractere de correspondência (~) é usado após um caractere de palavra. Por exemplo, o padrão `/npm/space/foo~` corresponde aos caminhos do pacote `/npm/space/foo` e `/npm/space/foo-bar`, mas não `/npm/space/food` ou `/npm/space/foot`.

É necessário usar um curinga (`*`) em vez de ~ após um caractere que não seja uma palavra, como no padrão `/npm/*`.

Diferenciação de letras maiúsculas e minúsculas

As definições de grupos de pacotes diferenciam maiúsculas de minúsculas, o que significa que padrões que diferem somente por maiúsculas e minúsculas podem existir como grupos de pacotes separados. Por exemplo, um usuário pode criar grupos de pacotes separados com os padrões `/npm//AsyncStorage$`, `/npm//asyncStorage$` e `/npm//asyncstorage$` para os três pacotes separados que existem no Registro Público do npm: AsyncStorage, asyncStorage, asyncstorage, que diferem somente por maiúsculas e minúsculas.

Embora a distinção entre maiúsculas e minúsculas seja importante, o CodeArtifact ainda associa pacotes a um grupo de pacotes se o pacote tiver uma variação do padrão que difere por maiúsculas e minúsculas. Se um usuário criar o grupo de pacotes `/npm//AsyncStorage$` sem criar os outros dois grupos mostrados acima, todas as variações de maiúsculas e minúsculas do nome AsyncStorage, incluindo asyncStorage e asyncstorage, serão associadas ao grupo de pacotes. Mas, conforme descrito na próxima seção, Correspondência forte e fraca, essas variações serão tratadas de forma diferente do AsyncStorage, que corresponde exatamente ao padrão.

Correspondência forte e fraca

As informações na seção anterior, Diferenciação de letras maiúsculas e minúsculas, afirmam que os grupos de pacotes diferenciam maiúsculas de minúsculas e, em seguida, explicam que não diferenciam. Isso ocorre porque as definições de grupos de pacotes no CodeArtifact têm um conceito de correspondência forte (ou correspondência exata) e correspondência fraca (ou correspondência de variação). Uma correspondência forte é quando o pacote corresponde exatamente ao padrão, sem qualquer variação. Uma correspondência fraca ocorre quando o pacote corresponde a uma variação do padrão, como letras maiúsculas e minúsculas diferentes. O comportamento de correspondência fraca impede que pacotes que são variações do padrão de um grupo de pacotes sejam acumulados em um grupo de pacotes mais geral. Quando um pacote é uma variação (correspondência fraca) do padrão do grupo de correspondência mais específico, o pacote é associado ao grupo, mas é bloqueado em vez de aplicar a configuração de controle de origem do grupo, impedindo que novas versões do pacote sejam retiradas de upstreams ou publicadas. Esse comportamento reduz o risco de ataques à cadeia de suprimentos resultantes da confusão de dependências de pacotes com nomes quase idênticos.

Para ilustrar um comportamento de correspondência fraca, suponha que o grupo de pacotes `/npm/*` permita a ingestão e bloqueie a publicação. Um grupo de pacotes mais específico, `/npm//anycompany-spicy-client$`, está configurado para bloquear a ingestão e permitir a publicação. O pacote chamado anycompany-spicy-client é uma correspondência forte do grupo de pacotes, que permite que versões de pacotes sejam publicadas e bloqueia a ingestão de versões de pacotes. A única forma permitida de publicar o nome do pacote com maiúsculas e minúsculas é anycompany-spicy-client, pois é uma correspondência forte com o padrão de definição do pacote. Uma variação diferente, como AnyCompany-spicy-client, é bloqueada para publicação porque é uma correspondência fraca. Mais importante ainda, o grupo de pacotes bloqueia a ingestão de todas as variações de maiúsculas e minúsculas, não somente do nome em minúsculas usado no padrão, reduzindo o risco de um ataque de confusão de dependências.

Variações adicionais

Além das diferenças entre maiúsculas e minúsculas, a correspondência fraca também ignora as diferenças em sequências de traços `-`, pontos `.`, sublinhados `_` e caracteres que podem ser confundidos (como caracteres de aparência semelhante de alfabetos diferentes). Durante a normalização usada para correspondência fraca, o CodeArtifact realiza a conversão de maiúsculas e minúsculas (semelhante à conversão para minúsculas), substitui sequências de caracteres de traço, ponto e sublinhado por um único ponto e normaliza caracteres que podem ser confundidos.

A correspondência fraca trata traços, pontos e sublinhados como equivalentes, mas não os ignora completamente. Isso significa que foo-bar, foo.bar, foo..bar e foo_bar são todos equivalentes de correspondência fraca, mas foobar não é. Embora vários repositórios públicos implementem etapas para evitar esses tipos de variações, a proteção fornecida pelos repositórios públicos não torna esse recurso de grupos de pacotes desnecessário. Por exemplo, repositórios públicos como o Registro Público do npm só evitarão novas variações do pacote chamado my-package se my-package já estiver publicado nele. Se my-package for um pacote interno e o grupo de pacotes `/npm//my-package$` for criado permitindo a publicação e bloqueando a ingestão, você provavelmente não desejará publicar my-package no Registro Público do npm para evitar que uma variante como my.package seja permitida.

Embora alguns formatos de pacote, como o Maven, tratem esses caracteres de forma diferente (o Maven trata `.` como um separador de hierarquia de namespace, mas não `-` ou `_`), algo como com.act-on ainda pode ser confundido com com.act.on.

> **Note**
>
> Observe que sempre que várias variações são associadas a um grupo de pacotes, um administrador pode criar um novo grupo de pacotes para uma variação específica para configurar um comportamento diferente para essa variação.

Marcar um grupo de pacotes no CodeArtifact

As tags são pares de chave-valor associados a recursos da AWS. Você pode aplicar tags aos grupos de pacotes no CodeArtifact. Para obter informações sobre a atribuição de tags do recurso do CodeArtifact, casos de uso, restrições de chave e valor de tag, além de tipos de recursos compatíveis, consulte Marcando atributos.

Você pode usar a CLI para especificar tags ao criar um grupo de pacotes ou adicionar, remover ou atualizar o valor das tags de um grupo de pacotes existente.

Grupos de pacotes de tag (CLI)

É possível usar a CLI para gerenciar tags de grupos de pacotes.

Caso não tenha feito isso, configure a AWS CLI seguindo as etapas emConfigurando com AWS CodeArtifact.

> **Tip**
>
> Para adicionar tags, você deve fornecer o nome do recurso da Amazon (ARN) do grupo de pacotes. Para obter o ARN do grupo de pacotes, execute o comando `describe-package-group`:
>
> ```
> aws codeartifact describe-package-group \
>    --domain my_domain \
>    --package-group /npm/scope/anycompany~ \
>    --query packageGroup.arn
> ```

Tópicos

Adicionar tags a um grupo de pacotes (CLI)

É possível adicionar tags aos grupos de pacotes quando eles são criados ou a um grupo de pacotes existente. Para obter informações sobre como adicionar tags a um grupo de pacotes ao criá-lo, consulte Criar um grupo de pacotes.

Para adicionar uma tag a um grupo de pacotes existente com a AWS CLI, no terminal ou na linha de comando, execute o comando tag-resource, especificando o nome do recurso da Amazon (ARN) do grupo de pacotes ao qual você deseja adicionar tags, bem como a chave e o valor da tag que deseja adicionar. Para obter informações sobre ARNs de grupos de pacotes, consulte Package group ARNs.

Você pode adicionar mais de uma tag a um grupo de pacotes. Por exemplo, para marcar um grupo de pacotes, */npm/scope/anycompany~*, com duas tags, uma chave de tag chamada *key1* com o valor de tag de *value1* e uma chave de tag chamada *key2* com o valor de tag de *value2*:

```
aws codeartifact tag-resource \
   --resource-arn arn:aws:codeartifact:us-west-2:123456789012:package-
group/my_domain/npm/scope/anycompany~ \
   --tags key=key1,value=value1 key=key2,value=value2
```

Se for bem-sucedido, este comando não terá saída.

Exibir tags para um grupo de pacotes (CLI)

Siga estas etapas para usar a AWS CLI para visualizar as tags da AWS para um grupo de pacotes. Se não foram adicionadas tags, a lista retornará vazia.

No terminal ou na linha de comando, execute o comando list-tags-for-resource com o nome do recurso da Amazon (ARN) do grupo de recursos. Para obter informações sobre ARNs de grupos de pacotes, consulte Package group ARNs.

Por exemplo, para visualizar uma lista de chaves e valores de tag para um grupo de pacotes, */npm/scope/anycompany~*, nomeado com um valor de ARN de `arn:aws:codeartifact:`*`us-west-2`*`:`*`123456789012`*`:package-group/`*`my_domain`*`/`*`npm/scope/anycompany~`*

```
aws codeartifact list-tags-for-resource \
   --resource-arn arn:aws:codeartifact:us-west-2:123456789012:package-
group/my_domain/npm/scope/anycompany~
```

Se houver êxito, o comando retornará informações semelhantes às seguintes:

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

Editar tags para um grupo de pacotes (CLI)

Siga estas etapas para usar a AWS CLI para editar uma tag para um grupo de pacotes. Você pode alterar o valor para uma chave existente ou adicionar outra chave. Você também pode remover tags de um grupo de pacotes, como mostrado na próxima seção.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o ARN do grupo de pacotes em que deseja atualizar uma tag e especifique a chave e o valor da tag. Para obter informações sobre ARNs de grupos de pacotes, consulte Package group ARNs.

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

```
   --tags key=key1,value=newvalue1
```

Se for bem-sucedido, este comando não terá saída.

Remover tags de um grupo de pacotes (CLI)

Siga estas etapas para usar a AWS CLI para remover uma tag de um grupo de pacotes.

> **Note**
>
> Se você excluir um grupo de pacotes, todas as associações de tag serão removidas do grupo de pacotes excluído. Não é necessário remover as tags antes de excluir um grupo de pacotes.

No terminal ou na linha de comando, execute o comando untag-resource, especificando o ARN do grupo de pacotes no qual você deseja remover tags e a chave da tag que deseja remover. Para obter informações sobre ARNs de grupos de pacotes, consulte Package group ARNs.

Por exemplo, para remover várias tags em um grupo de pacotes, */npm/scope/anycompany ~*, com as chaves de tag *key1* e *key2*:

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

Se for bem-sucedido, este comando não terá saída. Após remover as tags, você pode exibir as tags restantes no grupo de pacotes usando o comando `list-tags-for-resource`.

Trabalhando com domínios em CodeArtifact

CodeArtifact os domínios facilitam o gerenciamento de vários repositórios em uma organização. É possível usar um domínio para aplicar permissões em vários repositórios de propriedade de diferentes contas da AWS. Um ativo é armazenado somente uma vez em um domínio, mesmo que esteja disponível em vários repositórios.

Embora você possa ter vários domínios, a recomendação é ter um único domínio de produção que contenha todos os artefatos publicados para que as equipes de desenvolvimento possam encontrar e compartilhar pacotes. Você pode usar um segundo domínio de pré-produção para testar as alterações na configuração do domínio de produção.

Esses tópicos descrevem como usar o CodeArtifact console AWS CLI, o e CloudFormation para criar ou configurar CodeArtifact domínios.

Tópicos

Visão geral do domínio

Quando você trabalha com CodeArtifact, os domínios são úteis para o seguinte:

- Armazenamento desduplicado: um ativo só precisa ser armazenado uma vez em um domínio, mesmo que esteja disponível em 1 ou 1.000 repositórios. Isso significa que você paga pelo armazenamento apenas uma vez.
- Cópia rápida: quando você puxa pacotes de um CodeArtifact repositório upstream para um downstream ou usa a CopyPackageVersions API, somente os registros de metadados devem ser atualizados. Nenhum ativo é copiado. Isso agiliza a configuração de um novo repositório para preparação ou teste. Para obter mais informações, consulte Trabalhando com repositórios upstream em CodeArtifact.

- Fácil compartilhamento entre repositórios e equipes: todos os ativos e metadados em um domínio são criptografados com uma única AWS KMS key (chave KMS). Você não precisa gerenciar uma chave para cada repositório nem conceder a várias contas acesso a uma única chave.
- Aplique a política em vários repositórios: o administrador do domínio pode aplicar a política em todo o domínio. Isso inclui restringir quais contas têm acesso aos repositórios no domínio e quem pode configurar conexões com repositórios públicos para serem usadas como fontes de pacotes. Para obter mais informações, consulte Políticas de domínio.
- Nomes de repositórios exclusivos: o domínio disponibiliza um namespace para repositórios. Os nomes dos repositórios só precisam ser exclusivos dentro do domínio. Você deve usar nomes significativos que sejam fáceis de entender.

Os nomes de domínio devem ser exclusivos dentro de uma conta.

Você não pode criar um repositório sem um domínio. Ao usar a CreateRepositoryAPI para criar um repositório, você deve especificar um nome de domínio. Você não pode mover um repositório de um domínio para outro.

Um repositório pode pertencer à mesma AWS conta que possui o domínio ou a uma conta diferente. Se as contas proprietárias forem diferentes, a conta proprietária do repositório deverá receber a permissão `CreateRepository` no recurso do domínio. Você pode fazer isso adicionando uma política de recursos ao domínio usando o PutDomainPermissionsPolicycomando.

Embora uma organização possa ter vários domínios, a recomendação é ter um único domínio de produção que contenha todos os artefatos publicados para que as equipes de desenvolvimento possam encontrar e compartilhar pacotes na organização. Um segundo domínio de pré-produção pode ser útil para testar as alterações na configuração do domínio de produção.

Cross-account domínios

Os nomes de domínio só precisam ser exclusivos em uma conta, o que significa que pode haver vários domínios em uma região com o mesmo nome. Por esse motivo, se você quiser acessar um domínio pertencente a uma conta na qual você não está autenticado, será necessário apresentar a ID do proprietário do domínio junto com o nome do domínio na CLI e no console. Confira os exemplos de CLI a seguir.

Acesse um domínio pertencente a uma conta na qual você está autenticado:

Ao acessar um domínio na conta na qual você está autenticado, só é preciso especificar o nome do domínio. O exemplo a seguir lista os pacotes no *my_repo* repositório no *my_domain* domínio que pertence à sua conta.

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

Acesse um domínio pertencente a uma conta na qual você não está autenticado:

Ao acessar um domínio que pertence a uma conta na qual você não está autenticado, é necessário especificar o proprietário do domínio e também o nome do domínio. O exemplo a seguir lista pacotes no *other-repo* repositório no *other-domain* domínio pertencentes a uma conta na qual você não está autenticado. Observe a adição do parâmetro `--domain-owner`.

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

Tipos de AWS KMS teclas suportadas em CodeArtifact

CodeArtifact suporta somente chaves KMS simétricas. Você não pode usar uma chave KMS assimétrica para criptografar seus domínios. CodeArtifact Para obter mais informações, consulte Identificar chaves KMS simétricas e assimétricas. Para aprender a criar uma nova chave gerenciada pelo cliente, consulte Criar chaves KMS de criptografia simétricas no AWS Key Management Service Guia do desenvolvedor do .

CodeArtifact suporta Armazenamentos de chaves AWS KMS externos (XKS). Você é responsável pela disponibilidade, durabilidade e latência das principais operações com chaves XKS, que podem afetar a disponibilidade, a durabilidade e a latência com. CodeArtifact Alguns exemplos de efeitos do uso de teclas XKS com CodeArtifact:

- Como cada ativo de um pacote solicitado e todas as suas dependências estão sujeitos à latência de descriptografia, a latência de compilação pode ser aumentada substancialmente com um aumento na latência da operação do XKS.
- Como todos os ativos são criptografados CodeArtifact, a perda dos materiais da chave XKS resultará na perda de todos os ativos associados ao domínio usando a chave XKS.

Para obter mais informações sobre as chaves XKS, consulte Repositórios de chaves externos no Guia do desenvolvedor do AWS Key Management Service .

Criar um domínio

Você pode criar um domínio usando o CodeArtifact console, o AWS Command Line Interface (AWS CLI) ou CloudFormation. Quando você cria um domínio, ele não contém nenhum repositório. Para obter mais informações, consulte Criar um repositório. Para obter mais informações sobre como gerenciar CodeArtifact domínios com CloudFormation, consulteCriação de recursos do CodeArtifact com AWS CloudFormation.

Tópicos

Criar um domínio (console)

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, escolha Domínios e, em seguida, escolha Criar domínio.
3. Em Nome, insira um nome para o domínio.
4. Expanda Configuração Adicional.
5. Use uma AWS KMS key (chave KMS) para criptografar todos os ativos em seu domínio. Você pode usar uma chave KMS gerenciada pela AWS ou uma chave KMS que você gerencia. Para obter mais informações sobre os tipos de chaves KMS suportados em CodeArtifact, consulteTipos de AWS KMS teclas suportadas em CodeArtifact.

   - Escolha Chave gerenciada pela AWS se quiser usar a Chave gerenciada pela AWS padrão.
   - Escolha Chave gerenciada pelo cliente se quiser usar uma chave KMS que você gerencia. Para usar uma chave KMS que você gerencia, em ARN da chave gerenciada pelo cliente, pesquise e escolha a chave KMS.

   Para obter mais informações, consulte Chave gerenciada pela AWS e Chave gerenciada pelo cliente no Guia do desenvolvedor do AWS Key Management Service .
6. Escolha Criar domínio.

Crie um domínio (AWS CLI)

Para criar um domínio com o AWS CLI, use o `create-domain` comando. Você deve usar uma AWS KMS key (chave KMS) para criptografar todos os ativos em seu domínio. Você pode usar uma chave KMS AWS gerenciada ou uma chave KMS que você gerencia. Se você usar uma chave KMS AWS gerenciada, não use o `--encryption-key` parâmetro.

Para obter mais informações sobre os tipos de chaves KMS suportados em CodeArtifact, consulteTipos de AWS KMS teclas suportadas em CodeArtifact. Para obter mais informações sobre chaves KMS, consulte Chave gerenciada pela AWS e Chave gerenciada pelo cliente no Guia do desenvolvedor do AWS Key Management Service .

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

JSON-formatted os dados aparecem na saída com detalhes sobre seu novo domínio.

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

Se você usar uma chave KMS que você mesmo gerencia, inclua o Nome do recurso da Amazon (ARN) com o parâmetro `--encryption-key`.

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

JSON-formatted os dados aparecem na saída com detalhes sobre seu novo domínio.

```
{
    "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"
    }
}
```

Criar um domínio com tags

Para criar um domínio com tags, adicione o parâmetro `--tags` ao comando `create-domain`.

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

Exemplo AWS KMS política chave

Ao criar um domínio em CodeArtifact, você usa uma chave KMS para criptografar todos os ativos no domínio. Você pode escolher uma chave do KMS gerenciada pela AWS ou uma chave gerenciada pelo cliente que você gerencia. Para obter mais informações sobre as chaves do KMS, consulte o Guia do desenvolvedor do AWS Key Management Service.

Para usar uma chave gerenciada pelo cliente, sua chave KMS deve ter uma política de chaves que conceda acesso a. CodeArtifact Uma política de chaves é uma política de recursos para uma AWS KMS chave e é a principal forma de controlar o acesso às chaves do KMS. Cada chave do KMS deve ter exatamente uma política de chaves. As instruções na política de chaves determinam quem tem permissão para usar a chave do KMS e como eles podem usá-la.

O exemplo de declaração de política fundamental AWS CodeArtifact a seguir permite criar concessões e visualizar detalhes importantes em nome de usuários autorizados. Esta declaração de política limita a permissão para CodeArtifact agir em nome do ID da conta especificado usando as chaves de `kms:CallerAccount` condição `kms:ViaService` e. Ele também concede todas AWS KMS as permissões ao usuário raiz do IAM, para que a chave possa ser gerenciada depois de criada.

JSON

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

```
    "Id": "key-consolepolicy-3",
    "Statement": [
        {
            "Sid": "Allow access through AWS CodeArtifact for all principals in
 the account that are authorized to use CodeArtifact",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:CallerAccount": "111122223333",
                    "kms:ViaService": "codeartifact.us-west-2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        }
    ]
}
```

Excluir um domínio

Você pode excluir um domínio usando o CodeArtifact console ou o AWS Command Line Interface (AWS CLI).

Tópicos

- Restrições à exclusão de domínio
- Excluir um domínio (console)

- Excluir um domínio (AWS CLI)

Restrições à exclusão de domínio

Normalmente, você não pode excluir um domínio que contenha repositórios. Antes de excluir o domínio, primeiro exclua os repositórios dele. Para obter mais informações, consulte Excluir um repositório.

No entanto, se CodeArtifact não tiver mais acesso à chave KMS do domínio, você poderá excluir o domínio mesmo que ele ainda contenha repositórios. Essa situação ocorrerá se você excluir a chave KMS do domínio ou revogar a concessão do KMS CodeArtifact usada para acessar a chave. Nesse estado, você não pode acessar os repositórios no domínio ou os pacotes armazenados neles. A listagem e a exclusão de repositórios também não são possíveis quando não é CodeArtifact possível acessar a chave KMS do domínio. Por esse motivo, a exclusão do domínio não verifica se ele contém repositórios quando a chave KMS dele está inacessível.

> **Note**
>
> Quando um domínio que ainda contém repositórios for excluído, os repositórios CodeArtifact serão excluídos de forma assíncrona em 15 minutos. Depois que o domínio for excluído, os repositórios ainda estarão visíveis no CodeArtifact console e na saída do `list-repositories` comando até que a limpeza automática do repositório ocorra.

Excluir um domínio (console)

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, escolha Domínios e, em seguida, escolha o domínio que deseja excluir.
3. Escolha Excluir.

Excluir um domínio (AWS CLI)

Use o comando delete-domain para excluir um domínio.

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

JSON-formatted os dados aparecem na saída com detalhes sobre o domínio excluído.

```
{
    "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 domínio

CodeArtifact suporta o uso de permissões baseadas em recursos para controlar o acesso. Resource-basedas permissões permitem que você especifique quem tem acesso a um recurso e quais ações eles podem realizar nele. Por padrão, somente a conta da AWS proprietária do domínio pode criar e acessar repositórios no domínio. Você pode aplicar um documento de política a um domínio para permitir que outras entidades principais do IAM o acessem.

Para obter mais informações, consulte Políticas e permissões e Identity-Based Políticas e Resource-Based políticas.

Tópicos

- Permitir acesso entre contas a um domínio
- Exemplo de políticas de domínio
- Exemplo de política de domínio com AWS Organizations
- Definir uma política de domínio
- Ler uma política de domínio
- Excluir uma política de domínio

Permitir acesso entre contas a um domínio

Uma política de recursos é um arquivo de texto no formato JSON. O arquivo deve especificar uma entidade principal (ator), uma ou mais ações e um efeito (`Allow` ou `Deny`). Para criar um

repositório em um domínio pertencente a outra conta, a entidade principal deve receber a permissão `CreateRepository` no recurso do domínio.

Por exemplo, a política de recursos a seguir concede à conta `123456789012` permissão para criar um repositório no domínio.

JSON

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

Para permitir a criação de repositórios com tags, você deve incluir a permissão `codeartifact:TagResource`. Isso também dará à conta acesso para adicionar tags ao domínio e a todos os repositórios nele contidos.

A política de domínio é avaliada para todas as operações contra o domínio e todos os recursos no domínio. Isso significa que a política de domínio pode ser usada para aplicar permissões a repositórios e pacotes no domínio. Quando o elemento `Resource` é definido como `*`, a declaração se aplica a todos os recursos no domínio. Por exemplo, se a política acima também incluísse `codeartifact:DescribeRepository` na lista de ações permitidas do IAM, a política permitiria chamar `DescribeRepository` em todos os repositórios no domínio. Uma política de domínio pode ser usada para aplicar permissões a recursos específicos no domínio usando ARNs de recursos específicos no elemento `Resource`.

> **Note**
>
> As políticas de domínio e repositório podem ser usadas para configurar permissões. Quando ambas as políticas estiverem presentes, as duas políticas serão avaliadas e uma ação será permitida se for permitida por qualquer uma das políticas. Para obter mais informações, consulte Interação entre políticas de repositório e domínio.

Para acessar pacotes em um domínio pertencente a outra conta, uma entidade principal deve receber a permissão `GetAuthorizationToken` no recurso do domínio. Isso permite que o proprietário do domínio exerça controle sobre quais contas podem ler o conteúdo dos repositórios no domínio.

Por exemplo, a política de recursos a seguir concede à conta `123456789012` permissão para recuperar um token auth para qualquer repositório no domínio.

JSON

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

> **Note**
>
> Uma entidade principal que deseja buscar pacotes de um endpoint do repositório deve receber a permissão `ReadFromRepository` no recurso do repositório além da permissão `GetAuthorizationToken` no domínio. Da mesma forma, uma entidade principal

que deseja publicar pacotes em um endpoint de repositório deve receber a permissão `PublishPackageVersion` além de `GetAuthorizationToken`.
Para obter mais informações sobre as permissões `ReadFromRepository` e `PublishPackageVersion`, consulte Políticas de repositório.

Exemplo de políticas de domínio

Quando várias contas estão usando um domínio, elas devem receber um conjunto básico de permissões para permitir o uso total do domínio. A política de recursos a seguir lista um conjunto de permissões que autorizam o uso total do domínio.

JSON

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

> **Note**
>
> Você não precisa criar uma política de domínio se um domínio e todos os respectivos repositórios forem de propriedade de uma única conta e precisarem ser usados somente a partir dessa conta.

Exemplo de política de domínio com AWS Organizations

Você pode usar a chave de `aws:PrincipalOrgID` condição para conceder acesso a um CodeArtifact domínio de todas as contas da sua organização, da seguinte maneira.

JSON

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

Para obter mais informações sobre como usar a chave de condição `aws:PrincipalOrgID`, consulte Chaves de contexto de condição global da AWS no Guia do usuário do IAM.

Definir uma política de domínio

Você pode usar o comando `put-domain-permissions-policy` para anexar uma política a um domínio.

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

Quando você chama `put-domains-permissions-policy`, a política de recursos no domínio é ignorada ao avaliar as permissões. Isso garante que o proprietário de um domínio não possa se bloquear do domínio, o que impediria que ele pudesse atualizar a política de recursos.

> **Note**
>
> Você não pode conceder permissões a outra AWS conta para atualizar a política de recursos em um domínio usando uma política de recursos, pois a política de recursos é ignorada ao chamar put-domain-permissions-policy.

Exemplo de saída:

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

A saída do comando contém o nome do recurso da Amazon (ARN) do recurso do domínio, o conteúdo completo do documento de política e um identificador de revisão. O identificador de revisão pode ser passado para `put-domain-permissions-policy` usando a opção `--policy-revision`. Isso garante que uma revisão conhecida do documento seja sobrescrita e não uma versão mais recente definida por outro redator.

Ler uma política de domínio

Use o comando `get-domain-permissions-policy` para ler uma versão existente de um documento de política. Para formatar a saída para facilitar a leitura, use `--output` e `--query policy.document` junto com o módulo `json.tool` Python da seguinte maneira.

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

Exemplo de saída:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "BasicDomainPolicy",
            "Action": [
                "codeartifact:GetDomainPermissionsPolicy",
                "codeartifact:ListRepositoriesInDomain",
                "codeartifact:GetAuthorizationToken",
                "codeartifact:CreateRepository"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            }
        }
    ]
}
```

Excluir uma política de domínio

Use o comando `delete-domain-permissions-policy` para excluir uma política de um domínio.

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

O formato da saída é o mesmo que o dos comandos `get-domain-permissions-policy` e `delete-domain-permissions-policy`.

Marcar um domínio em CodeArtifact

As tags são pares de chave-valor associados a recursos da AWS. Você pode aplicar tags aos seus domínios em CodeArtifact. Para obter informações sobre marcação de CodeArtifact recursos, casos de uso, restrições de valor e chave de tag e tipos de recursos compatíveis, consulte. Marcando atributos

Você pode usar a CLI para especificar tags ao criar um domínio. Você pode usar o console ou a CLI para adicionar ou remover tags e atualizar os valores das tags em um domínio. Você pode adicionar até 50 tags a cada domínio.

Tópicos

Domínios de tag (CLI)

É possível usar a CLI para gerenciar tags de domínio.

Tópicos

Adicionar tags a um domínio (CLI)

Você pode usar o console ou o AWS CLI para marcar domínios.

Para adicionar uma tag a um domínio ao criá-lo, consulte Criar um repositório.

Nestas etapas, partimos do princípio de que você já instalou uma versão recente da AWS CLI ou atualizou para a versão atual. Para obter mais informações, consulte Instalar a AWS Command Line Interface.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o nome do recurso da Amazon (ARN) do domínio em que você deseja adicionar tags e a chave e o valor da tag que você deseja adicionar.

> **Note**
>
> Para obter o ARN do domínio, execute o comando `describe-domain`:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Você pode adicionar mais de uma tag a um domínio. Por exemplo, para marcar um domínio chamado *my_domain* com duas tags, uma chave de tag *key1* com o valor de tag de *value1* e uma chave de tag *key2* com o valor de tag de*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
```

Se for bem-sucedido, este comando não terá saída.

Exibir tags para um domínio (CLI)

Siga estas etapas para usar o AWS CLI para visualizar as AWS tags de um domínio. Se não foram adicionadas tags, a lista retornará vazia.

No terminal ou na linha de comando, execute o comando list-tags-for-resource com o nome do recurso da Amazon (ARN) do domínio.

> **Note**
>
> Para obter o ARN do domínio, execute o comando `describe-domain`:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Por exemplo, para ver uma lista de chaves e valores de tag para um domínio chamado *my_domain* com o valor `arn:aws:codeartifact:`*us-west-2*`:`*123456789012*`:domain/`*my_domain* ARN:

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

Se houver êxito, o comando retornará informações semelhantes às seguintes:

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

Editar tags para um domínio (CLI)

Siga estas etapas para usar o AWS CLI para editar uma tag para um domínio. Você pode alterar o valor para uma chave existente ou adicionar outra chave. Você também pode remover tags de um domínio, como mostrado na próxima seção.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o ARN do domínio no qual deseja atualizar uma tag e especifique a chave e o valor da tag:

> **Note**
>
> Para obter o ARN do domínio, execute o 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
```

Se for bem-sucedido, este comando não terá saída.

Remover tags de um domínio (CLI)

Siga estas etapas para usar o AWS CLI para remover uma tag de um domínio.

> **Note**
>
> Se você excluir um domínio, todas as associações de tag serão removidas do domínio excluído. Você não precisa remover as tags antes de excluir um domínio.

No terminal ou na linha de comando, execute o comando untag-resource, especificando o ARN do domínio de onde você deseja remover tags e a chave da tag que deseja remover.

> **Note**
>
> Para obter o ARN do domínio, execute o comando `describe-domain`:
>
> ```
> aws codeartifact describe-domain --domain my_domain --query domain.arn
> ```

Por exemplo, para remover várias tags em um domínio chamado *mydomain* com as chaves de tag *key1* e*key2*:

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

Se for bem-sucedido, este comando não terá saída. Após remover as tags, você pode exibir as tags restantes no domínio usando o comando `list-tags-for-resource`.

Marcar domínios (console)

Você pode usar o console ou a CLI para marcar recursos.

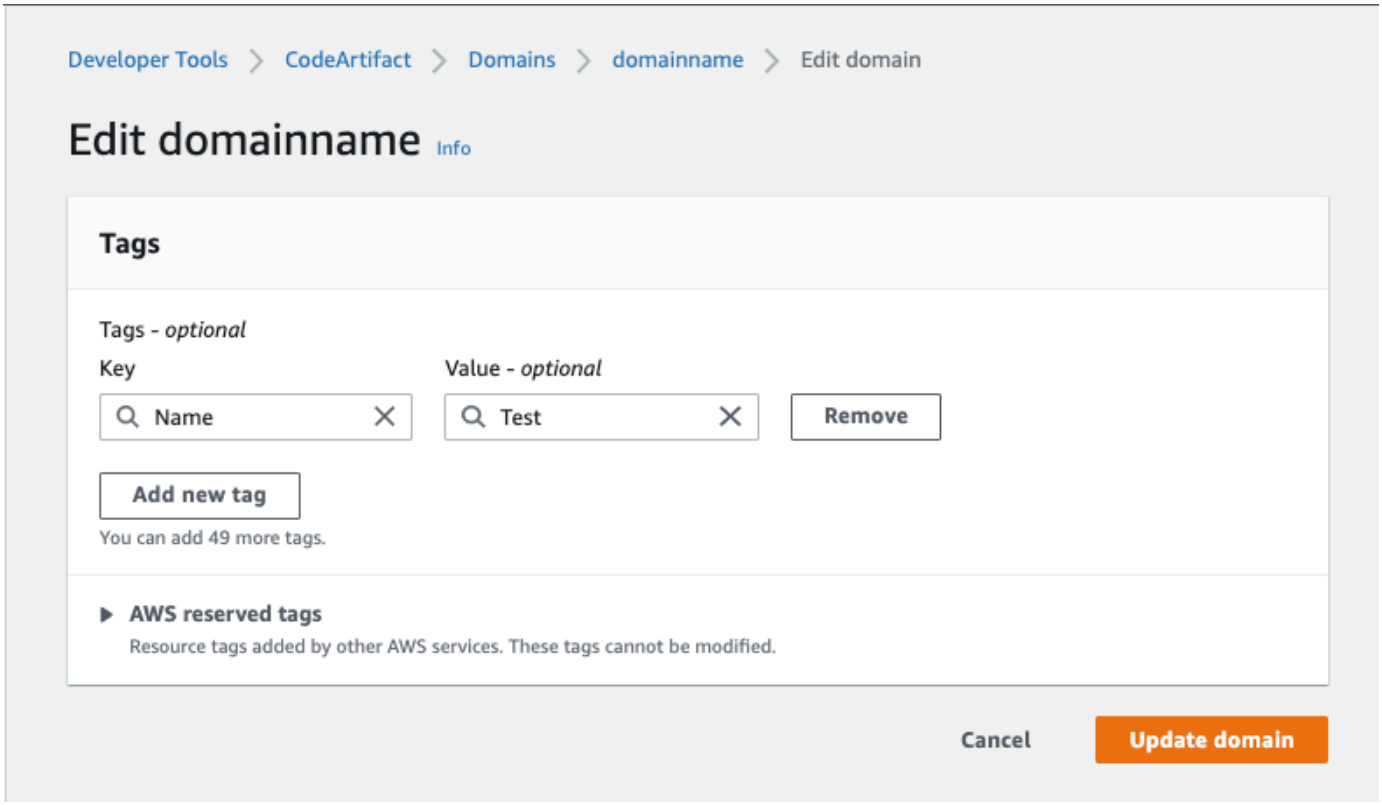
Tópicos

- Adicionar tags a um domínio (console)
- Exibir tags para um domínio (console)
- Editar tags para um domínio (console)
- Remover tags de um domínio (console)

Adicionar tags a um domínio (console)

Você pode usar o console para adicionar tags a um domínio existente.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Na página Domínios, escolha o domínio ao qual deseja adicionar tags.
3. Expanda a seção Detalhes.
4. Em Tags de domínio, escolha Adicionar tags de domínio se não houver tags no domínio ou escolha Exibir e editar tags de domínio, se houver.
5. Selecione Adicionar nova tag.
6. Nos campos Chave e Valor, insira o texto para cada tag que deseja adicionar. (O campo Value (Valor) é opcional.) Por exemplo, em Key (Chave), insira **`Name`**. Em Valor, informe **`Test`**.



7. (Opcional) Escolha Add tag (Adicionar tag) para adicionar mais linhas e inserir mais tags.
8. Escolha Atualizar domínio.

Exibir tags para um domínio (console)

Você pode usar o console para listar tags de domínios existentes.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Na página Domínios, escolha o domínio em que deseja exibir tags.
3. Expanda a seção Detalhes.
4. Em Tags de domínio, escolha Exibir e editar tags de domínio.

> **Note**
>
> Se não houver tags adicionadas a esse domínio, o console lerá Adicionar tags do domínio.

Editar tags para um domínio (console)

Você pode usar o console para editar as tags adicionadas ao domínio.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. Na página Domínios, escolha o domínio em que deseja atualizar tags.
3. Expanda a seção Detalhes.
4. Em Tags de domínio, escolha Exibir e editar tags de domínio.

> **Note**
>
> Se não houver tags adicionadas a esse domínio, o console lerá Adicionar tags do domínio.

5. Nos campos Key (Chave) e Value (Valor), atualize os valores em cada campo conforme necessário. Por exemplo, para a chave **`Name`**, em Value (Valor), altere **`Test`** para **`Prod`**.
6. Escolha Atualizar domínio.

Remover tags de um domínio (console)

Você pode usar o console para excluir tags de domínios.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.

2. Na página Domínios, escolha o domínio em que deseja remover tags.
3. Expanda a seção Detalhes.
4. Em Tags de domínio, escolha Exibir e editar tags de domínio.

> **Note**
>
> Se não houver tags adicionadas a esse domínio, o console lerá Adicionar tags do domínio.

5. Ao lado da chave e do valor para cada tag que você deseja excluir, escolha Remover.
6. Escolha Atualizar domínio.

Usar o CodeArtifact com Cargo

Estes tópicos descrevem como usar o Cargo, o gerenciador de pacotes Rust, com o CodeArtifact.

> **Note**
>
> O CodeArtifact é compatível somente com Cargo 1.74.0 e superior. O Cargo 1.74.0 é a versão mais antiga compatível com a autenticação em um repositório do CodeArtifact.

Tópicos

Configurar e usar o Cargo com o CodeArtifact

Você pode usar o Cargo para publicar e baixar crates dos repositórios do CodeArtifact ou para buscar crates em crates.io, o registro de crates da comunidade Rust. Este tópico descreve como configurar o Cargo para se autenticar e usar um repositório do CodeArtifact.

Configurar o Cargo com o CodeArtifact

Para usar o Cargo para instalar e publicar crates por meio do AWS CodeArtifact, primeiro você precisa configurá-los com as informações do seu repositório do CodeArtifact. Siga as etapas em um dos procedimentos a seguir para configurar o Cargo com as informações e credenciais do endpoint do repositório do CodeArtifact.

Configurar o Cargo usando as instruções do console

É possível usar as instruções de configuração no console para conectar o Cargo ao repositório do CodeArtifact. As instruções do console fornecem uma configuração do Cargo personalizada para o repositório do CodeArtifact. Você pode usar essa configuração personalizada para configurar o Cargo sem precisar encontrar e preencher suas informações do CodeArtifact.

1. Abra o console do AWS CodeArtifact em https://console.aws.amazon.com/codesuite/codeartifact/home.

2. No painel de navegação, selecione Repositórios e, em seguida, escolha um repositório para conectar ao Cargo.
3. Clique em Visualizar instruções de conexão.
4. Selecione o seu sistema operacional.
5. Selecione Cargo.
6. Siga as instruções geradas para conectar o Cargo ao seu repositório do CodeArtifact.

Configurar o Cargo manualmente

Se você não puder ou não quiser usar as instruções de configuração do console, poderá usar as instruções a seguir para conectar manualmente o Cargo ao seu repositório do CodeArtifact.

macOS and Linux

Para configurar o Cargo com o CodeArtifact, você precisa definir seu repositório do CodeArtifact como um registro na configuração do Cargo e fornecer as credenciais.

- Substitua *my_registry* pelo nome do seu registro.
- Substitua *my_domain* pelo seu nome de domínio do CodeArtifact.
- Substitua *111122223333* pelo ID da conta da AWS do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
- Substitua *my_repo* pelo nome do seu repositório do CodeArtifact.

Copie a configuração para publicar e baixar pacotes Cargo em seu repositório e salve-a no arquivo `~/.cargo/config.toml` para uma configuração em nível de sistema ou `.cargo/config.toml` para uma configuração em nível de projeto:

```
[registries.my_registry]
index = "sparse+https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/cargo/my_repo/"
credential-provider = "cargo:token-from-stdout aws codeartifact get-authorization-
token --domain my_domain --domain-owner 111122223333 --region us-west-2 --query
 authorizationToken --output text"

[registry]
default = "my_registry"
```

```
[source.crates-io]
replace-with = "my_registry"
```

Windows: Download packages only

Para configurar o Cargo com o CodeArtifact, você precisa definir seu repositório do CodeArtifact como um registro na configuração do Cargo e fornecer as credenciais.

- Substitua *my_registry* pelo nome do seu registro.
- Substitua *my_domain* pelo seu nome de domínio do CodeArtifact.
- Substitua *111122223333* pelo ID da conta da AWS do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
- Substitua *my_repo* pelo nome do seu repositório do CodeArtifact.

Copie a configuração para baixar somente pacotes Cargo do seu repositório e salve-a no arquivo `%USERPROFILE%\.cargo\config.toml` para uma configuração em nível de sistema ou `.cargo\config.toml` para uma configuração em nível de projeto:

```
[registries.my_registry]
index = "sparse+https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/cargo/my_repo/"
credential-provider = "cargo:token-from-stdout aws codeartifact get-authorization-
token --domain my_domain --domain-owner 111122223333 --region us-west-2 --query
 authorizationToken --output text"

[registry]
default = "my_registry"

[source.crates-io]
replace-with = "my_registry"
```

Windows: Publish and download packages

1. Para configurar o Cargo com o CodeArtifact, você precisa definir seu repositório do CodeArtifact como um registro na configuração do Cargo e fornecer as credenciais.

   - Substitua *my_registry* pelo nome do seu registro.
   - Substitua *my_domain* pelo seu nome de domínio do CodeArtifact.

- Substitua *111122223333* pelo ID da conta da AWS do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
- Substitua *my_repo* pelo nome do seu repositório do CodeArtifact.

Copie a configuração para publicar e baixar pacotes Cargo em seu repositório e salve-a no arquivo `%USERPROFILE%\.cargo\config.toml` para uma configuração em nível de sistema ou `.cargo\config.toml` em nível de projeto.

É recomendável usar o provedor de credenciais `cargo:token`, que usa as credenciais armazenadas em seu arquivo `~/.cargo/credentials.toml`. Você pode encontrar um erro durante `cargo publish` se usar `cargo:token-from-stdout`, porque o cliente Cargo não corta o token de autorização adequadamente durante `cargo publish`.

```
[registries.my_registry]
index = "sparse+https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/cargo/my_repo/"
credential-provider = "cargo:token"

[registry]
default = "my_registry"

[source.crates-io]
replace-with = "my_registry"
```

2. Para publicar pacotes Cargo em seu repositório com o Windows, você deve usar o comando `get-authorization-token` do CodeArtifact e o comando `login` do Cargo para obter um token de autorização e suas credenciais.

   - Substitua *my_registry* pelo nome do seu registro, conforme definido em `[registries.my_registry]`.
   - Substitua *my_domain* pelo seu nome de domínio do CodeArtifact.
   - Substitua *111122223333* pelo ID da conta da AWS do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

```
aws codeartifact get-authorization-token --domain my_domain --domain-
owner 111122223333 --region us-west-2 --query authorizationToken --output text |
 cargo login --registry my_registry
```

> **Note**
>
> O token de autorização gerado é válido por 12 horas. Você precisará criar um novo se tiverem passado 12 horas desde a criação do token.

A seção `[registries.my_registry]` no exemplo anterior define um registro com *my_registry* e fornece as informações de `index` e `credential-provider`.

- `index` especifica o URL do índice do seu registro, que é o endpoint do repositório do CodeArtifact que termina com `/`. O prefixo `sparse+` é obrigatório para registros que não são repositórios Git.

  > **Note**
  >
  > Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

- `credential-provider` especifica o provedor de credenciais para o registro fornecido. Se `credential-provider` não estiver definido, os provedores em `registry.global-credential-providers` serão usados. Ao configurar `credential-provider` como `cargo:token-from-stdout`, o cliente Cargo buscará o novo token de autorização automaticamente ao publicar ou baixar do seu repositório do CodeArtifact, portanto, não é necessário atualizar manualmente o token de autorização a cada 12 horas.

A seção `[registry]` define o registro padrão usado.

- `default` especifica o nome do registro definido em `[registries.my_registry]`, para usar por padrão ao publicar ou baixar do seu repositório do CodeArtifact.

A seção `[source.crates-io]` define o registro padrão usado quando um não é especificado.

- `replace-with = "my_registry"` substitui o registro público, crates.io, pelo seu repositório do CodeArtifact definido em `[registries.my_registry]`. Essa configuração é recomendada se você precisar solicitar pacotes da conexão externa, como crates.io.

  Para obter todos os benefícios do CodeArtifact, como o controle de origem do pacote que evita ataques de confusão de dependências, é recomendável usar a substituição de origem. Com a substituição de origem, o CodeArtifact faz o proxy de todas as solicitações para a conexão externa e copia o pacote da conexão externa para o seu repositório. Sem a substituição de origem, o cliente Cargo recuperará diretamente o pacote com base na configuração no arquivo `Cargo.toml` em seu projeto. Se uma dependência não estiver marcada com `registry=my_registry`, o cliente Cargo a recuperará diretamente do crates.io sem se comunicar com seu repositório do CodeArtifact.

  > **Note**
  >
  > Se você começar a usar a substituição de origem e depois atualizar seu arquivo de configuração para não usar a substituição de origem, poderá encontrar erros. O cenário oposto também poderá levar a erros. Portanto, é recomendável evitar alterar a configuração do seu projeto.

Instalar crates Cargo

Use os procedimentos a seguir para instalar crates Cargo por meio de um repositório do CodeArtifact ou de crates.io.

Instalar crates Cargo por meio do CodeArtifact

É possível usar a CLI do Cargo (`cargo`) para instalar rapidamente uma versão específica de um crate Ruby por meio do repositório do CodeArtifact.

Como instalar crates Cargo por meio de um repositório do CodeArtifact com **`cargo`**

1. Caso ainda não tenha feito isso, siga as etapas em Configurar e usar o Cargo com o CodeArtifact para configurar a CLI `cargo` para usar o repositório do CodeArtifact com as credenciais adequadas.
2. Use o seguinte comando para instalar crates Cargo por meio do CodeArtifact:

```
cargo add my_cargo_package@1.0.0
```

Para obter mais informações, consulte cargo add em The Cargo Book.

Publicar crates Cargo no CodeArtifact

Use o procedimento a seguir para publicar crates Cargo em um repositório do CodeArtifact usando a CLI `cargo`.

1. Caso ainda não tenha feito isso, siga as etapas em Configurar e usar o Cargo com o CodeArtifact para configurar a CLI `cargo` para usar o repositório do CodeArtifact com as credenciais adequadas.
2. Use o seguinte comando para publicar crates Cargo em um repositório do CodeArtifact:

   ```
   cargo publish
   ```

   Para obter mais informações, consulte cargo publish em The Cargo Book.

Suporte para comandos Cargo

As seções a seguir resumem os comandos do Cargo que são suportados pelos CodeArtifact repositórios, além dos comandos específicos que não são suportados.

Sumário

Comandos compatíveis que exigem acesso ao registro

Esta seção lista os comandos Cargo nos quais o cliente Cargo requer acesso ao registro com o qual foi configurado. Verificou-se que esses comandos funcionam corretamente quando invocados em um CodeArtifact repositório.

Command	Description
build	Cria pacotes locais e suas dependências.
check	Verifica se há erros nos pacotes locais e suas dependências.
buscar	Busca as dependências de um pacote.
publish	Publica um pacote no registro.

Comandos incompatíveis

Esses comandos do Cargo não são suportados pelos CodeArtifact repositórios.

Command	Description
owner	Gerencia os proprietários do crate no registro.
pesquisa	Pesquisa pacotes no registro.

Usando o CodeArtifact com Maven

O formato do repositório Maven é usado por várias linguagens diferentes, incluindo Java, Kotlin, Scala e Clojure. Ele é compatível com várias ferramentas de construção diferentes, incluindo Maven, Gradle, Scala SBT, Apache Ivy e Leiningen.

Testamos e confirmamos a compatibilidade com o CodeArtifact para as seguintes versões:

- Versão mais recente do Maven: 3.6.3.
- A versão mais recente do Gradle: 6.4.1. 5.5.1 também foi testada.
- A versão mais recente do Clojure: 1.11.1 também foi testada.

Tópicos

Usar o CodeArtifact com o Gradle

Depois que o token de autenticação do CodeArtifact estiver em uma variável de ambiente, conforme descrito em Passar um token de autenticação usando uma variável de ambiente, siga estas instruções para consumir pacotes Maven e publicar novos pacotes em um repositório do CodeArtifact.

Tópicos

- Executar uma compilação do Gradle no IntelliJ IDEA

Buscar dependências

Para buscar dependências do CodeArtifact em uma compilação do Gradle, use o procedimento a seguir.

Para buscar dependências do CodeArtifact em uma compilação do Gradle

1. Caso contrário, crie e armazene um token de autorização do CodeArtifact em uma variável de ambiente seguindo o procedimento em Passar um token de autenticação usando uma variável de ambiente.
2. Adicione uma seção `maven` à seção `repositories` no arquivo `build.gradle` do projeto.

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

O `url` exemplo anterior é o endpoint do seu repositório do CodeArtifact. O Gradle usa o endpoint para se conectar ao repositório. No exemplo, `my_domain` é o nome do seu domínio; `111122223333`, o ID do proprietário do domínio e `my_repo`, o nome do seu repositório. Você pode recuperar o endpoint de um repositório usando o comando `get-repository-endpoint` AWS CLI.

Por exemplo, com um repositório chamado *my_repo* dentro de um domínio chamado *my_domain*, o comando é o seguinte:

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

O comando `get-repository-endpoint` retornará o endpoint do repositório:

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

O objeto `credentials` no exemplo anterior inclui o token de autorização do CodeArtifact criado na Etapa 1, que o Gradle usa para se autenticar no CodeArtifact.

> **Note**
>
> Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

3. (Opcional) - Para usar o repositório do CodeArtifact como a única fonte para as dependências do seu projeto, remova todas as outras seções de `repositories` `build.gradle`. Se você tiver mais de um repositório, o Gradle pesquisará dependências em cada repositório, na ordem em que estão listadas.
4. Depois de configurar o repositório, você pode adicionar dependências do projeto à seção `dependencies` com a sintaxe padrão do Gradle.

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

Buscar plug-ins

Por padrão, o Gradle resolverá plug-ins do Portal de plug-ins do Gradle público. Para extrair plug-ins de um repositório do CodeArtifact, use o procedimento a seguir.

Para extrair plug-ins de um repositório do CodeArtifact

1. Caso contrário, crie e armazene um token de autorização do CodeArtifact em uma variável de ambiente seguindo o procedimento em Passar um token de autenticação usando uma variável de ambiente.

2. Adicione um bloco `pluginManagement` ao seu arquivo `settings.gradle`. O bloco `pluginManagement` deve aparecer antes de qualquer outra declaração em `settings.gradle`; consulte o seguinte trecho:

```
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
            }
        }
    }
}
```

Isso garantirá que o Gradle resolva plug-ins do repositório especificado. O repositório deve ter um repositório upstream com uma conexão externa com o Portal de plug-ins do Gradle (por exemplo, `gradle-plugins-store`) para que os plug-ins do Gradle normalmente exigidos estejam disponíveis para a compilação. Para obter mais informações, consulte a documentação do Gradle.

Publicar artefatos

Esta seção descreve como publicar uma biblioteca Java criada com o Gradle em um repositório do CodeArtifact.

Primeiro, adicione o plug-in `maven-publish` à seção `plugins` do arquivo `build.gradle` do projeto.

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

Em seguida, adicione uma seção `publishing` ao arquivo `build.gradle` do projeto.

```
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
            }
        }
    }
}
```

O plug-in `maven-publish` gera um arquivo POM com base no `groupId`, `artifactId` e `version` especificados na seção `publishing`.

Depois que essas alterações em `build.gradle` forem concluídas, execute o comando a seguir para criar o projeto e carregá-lo no repositório.

```
./gradlew publish
```

Use `list-package-versions` para verificar se o pacote foi publicado com sucesso.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven\
  --namespace com.company.framework --package my-package-name
```

Exemplo de resultado:

```
{
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "example",
    "versions": [
        {
```

```
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

Para obter mais informações, consulte os tópicos a seguir no site do Gradle:

- Criar bibliotecas Java
- Publicar um projeto como um módulo

Executar uma compilação do Gradle no IntelliJ IDEA

Você pode executar uma compilação do Gradle no IntelliJ IDEA que extraia dependências do CodeArtifact. Para se autenticar com o CodeArtifact, você precisa fornecer ao Gradle um token de autorização do CodeArtifact. Há três métodos para fornecer um token de autorização.

- Método 1: armazenar o token de autorização em `gradle.properties`. Use esse método se você conseguir fazer a substituição ou adição ao conteúdo do arquivo `gradle.properties`.
- Método 2: armazenar o token de autorização em um arquivo separado. Use esse método se você não quiser modificar o arquivo `gradle.properties`.
- Método 3: gerar um novo token de autorização para cada execução usando `aws` como um script em linha no `build.gradle`. Use esse método se quiser que o script do Gradle busque um novo token a cada execução. O token não será armazenado no sistema de arquivos.

Token stored in gradle.properties

Método 1: armazenar o token de autorização em **`gradle.properties`**

> **Note**
>
> O exemplo mostra o arquivo `gradle.properties` localizado em `GRADLE_USER_HOME`.

1. Atualize o arquivo `build.gradle` com o seguinte trecho:

```
repositories {
```

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

2. Para obter plug-ins do CodeArtifact, adicione um bloco `pluginManagement` ao arquivo `settings.gradle`. O bloco `pluginManagement` deve aparecer antes de qualquer outra declaração em `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. Obtenha um token de autorização do CodeArtifact:

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

4. Grave o token de autorização no arquivo `gradle.properties`:

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

Token stored in separate file

Método 2: armazenar o token de autorização em um arquivo separado

1. Atualize o arquivo `build.gradle` com o seguinte trecho:

```
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 obter plug-ins do CodeArtifact, adicione um bloco `pluginManagement` ao arquivo `settings.gradle`. O bloco `pluginManagement` deve aparecer antes de qualquer outra declaração em `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. Obtenha um token de autorização do CodeArtifact:

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

4. Grave o token de autorização no arquivo que foi especificado em seu arquivo `build.gradle`:

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

Token generated for each run in build.gradle

Método 3: gerar um novo token de autenticação para cada execução usando **`aws`** como um script em linha no **`build.gradle`**

1. Atualize o arquivo `build.gradle` com o seguinte trecho:

```
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 obter plug-ins do CodeArtifact, adicione um bloco `pluginManagement` ao arquivo `settings.gradle`. O bloco `pluginManagement` deve aparecer antes de qualquer outra declaração em `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
            }
        }
    }
}
```

Use CodeArtifact com mvn

É possível usar o comando `mvn` para executar compilações do Maven. Esta seção mostra como configurar `mvn` para usar um CodeArtifact repositório.

Tópicos

- Buscar dependências
- Publicar artefatos
- Publicar artefatos de terceiros
- Restringir downloads de dependências do Maven em um repositório CodeArtifact
- Informações do Apache Maven Project

Buscar dependências

Para configurar `mvn` para buscar dependências de um CodeArtifact repositório, você deve editar o arquivo de configuração do Maven e, opcionalmente`settings.xml`, o POM do seu projeto.

1. Caso contrário, crie e armazene um token de CodeArtifact autenticação em uma variável de ambiente conforme descrito em Passar um token de autenticação usando uma variável de ambiente Para configurar a autenticação no seu CodeArtifact repositório.
2. Em `settings.xml` (geralmente encontrado em `~/.m2/settings.xml`), adicione uma seção `<servers>` com uma referência à variável de ambiente `CODEARTIFACT_AUTH_TOKEN` para que o Maven passe o token nas solicitações HTTP.

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

3. Adicione o endpoint de URL do seu CodeArtifact repositório em um `<repository>` elemento. Você pode fazer isso no `settings.xml` ou arquivo POM do seu projeto.

   Você pode recuperar o endpoint do seu repositório usando o comando. `get-repository-endpoint` AWS CLI

   Por exemplo, com um repositório nomeado *`my_repo`* dentro de um domínio chamado*`my_domain`*, o comando é o seguinte:

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

   O comando `get-repository-endpoint` retornará o endpoint do repositório:

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

   > **Note**
   >
   > Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

   Adicione o endpoint do repositório ao `settings.xml` da seguinte forma.

   ```
   <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>
```

Ou você pode adicionar a `<repositories>` seção a um arquivo POM do projeto CodeArtifact para usar somente nesse projeto.

```
<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

Você pode usar qualquer valor no elemento `<id>`, mas ele deve ser o mesmo nos elementos `<server>` e `<repository>`. Isso permite que as credenciais especificadas sejam incluídas nas solicitações para CodeArtifact.

Depois de fazer essas alterações na configuração, você pode criar o projeto.

```
mvn compile
```

O Maven registra o URL completo de todas as dependências que ele baixa para o console.

```
[INFO] ------------------< com.example.example:myapp >------------------
[INFO] Building myapp 1.0
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.pom
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.pom (11 kB at 3.9 kB/s)
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/org/apache/commons/commons-parent/42/commons-parent-42.pom (68 kB at 123
 kB/s)
Downloading from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.jar
Downloaded from codeartifact: https://<domain>.d.codeartifact.us-west-2.amazonaws.com/
maven/myrepo/commons-cli/commons-cli/1.4/commons-cli-1.4.jar (54 kB at 134 kB/s)
```

Publicar artefatos

Para publicar um artefato do Maven em um CodeArtifact repositório, você também deve editar o POM do `~/.m2/settings.xml` projeto. `mvn`

1. Caso contrário, crie e armazene um token de CodeArtifact autenticação em uma variável de ambiente conforme descrito em Passar um token de autenticação usando uma variável de ambiente Para configurar a autenticação no seu CodeArtifact repositório.
2. Adicione uma seção `<servers>` a `settings.xml` com uma referência à variável de ambiente `CODEARTIFACT_AUTH_TOKEN` para que o Maven passe o token nas solicitações HTTP.

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

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

3. Adicione uma seção `<distributionManagement>` ao `pom.xml` do seu projeto.

```
<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>
```

Depois de fazer essas alterações na configuração, você pode criar o projeto e publicá-lo no repositório específico.

```
mvn deploy
```

Use `list-package-versions` para verificar se o pacote foi publicado com sucesso.

```
aws codeartifact list-package-versions --domain my_domain --domain-owner 111122223333
 --repository my_repo --format maven \
  --namespace com.company.framework --package my-package-name
```

Exemplo de saída:

```
{
    "defaultDisplayVersion": null,
    "format": "maven",
    "namespace": "com.company.framework",
    "package": "my-package-name",
    "versions": [
```

```
        {
            "version": "1.0",
            "revision": "REVISION-SAMPLE-1-C7F4S5E9B772FC",
            "status": "Published"
        }
    ]
}
```

Publicar artefatos de terceiros

Você pode publicar artefatos Maven de terceiros em um CodeArtifact repositório com. `mvn deploy:deploy-file` Isso pode ser útil para usuários que desejam publicar artefatos e têm somente arquivos JAR e não têm acesso ao código-fonte do pacote ou aos arquivos POM.

O comando `mvn deploy:deploy-file` gerará um arquivo POM com base nas informações passadas na linha de comando.

Publicar artefatos Maven de terceiros

1. Caso contrário, crie e armazene um token de CodeArtifact autenticação em uma variável de ambiente conforme descrito em Passar um token de autenticação usando uma variável de ambiente Para configurar a autenticação no seu CodeArtifact repositório.
2. Crie um arquivo `~/.m2/settings.xml` com o seguinte conteúdo:

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

3. Execute o 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**
>
> O exemplo acima publica `commons-cli 1.4`. Modifique os argumentos groupID, artifactID, version e file para publicar um JAR diferente.

Essas instruções são baseadas em exemplos no Guia para implantação de terceiros JARs em repositórios remotos da documentação do Apache Maven.

Restringir downloads de dependências do Maven em um repositório CodeArtifact

Se um pacote não puder ser obtido de um repositório configurado, por padrão, o `mvn` comando o buscará no Maven Central. Adicione o `mirrors` elemento para `settings.xml` fazer com que `mvn` sempre use seu CodeArtifact repositório.

```
<settings>
  ...
    <mirrors>
      <mirror>
        <id>central-mirror</id>
        <name>CodeArtifact Maven Central mirror</name>
        <url>https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/
maven/my_repo/</url>
        <mirrorOf>central</mirrorOf>
      </mirror>
    </mirrors>
  ...
</settings>
```

Se você adicionar um elemento `mirrors`, você também deve ter um elemento `pluginRepository` em seu `settings.xml` ou `pom.xml`. O exemplo a seguir busca dependências de aplicativos e plug-ins do Maven de um repositório. 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>
```

O exemplo a seguir busca dependências de aplicativos de um CodeArtifact repositório e busca plug-ins Maven do Maven Central.

```
<profiles>
   <profile>
     <id>default</id>
     ...
     <pluginRepositories>
       <pluginRepository>
         <id>central-plugins</id>
         <name>Central Plugins</name>
         <url>https://repo.maven.apache.org/maven2/</url>
         <releases>
             <enabled>true</enabled>
         </releases>
         <snapshots>
             <enabled>true</enabled>
         </snapshots>
       </pluginRepository>
     </pluginRepositories>
```

```
    ....
    </profile>
  </profiles>
```

Informações do Apache Maven Project

Para obter mais informações sobre o Maven, consulte esses tópicos no site do Apache Maven Project:

- Configurar vários repositórios
- Referência de configurações
- Gerenciamento de distribuição
- Perfis

Usar o CodeArtifact com deps.edn

É possível usar `deps.edn` com `clj` para gerenciar dependências para projetos do Clojure. Esta seção mostra como configurar o `deps.edn` para usar um repositório do CodeArtifact.

Tópicos

- Buscar dependências
- Publicar artefatos

Buscar dependências

Para configurar o `Clojure` para buscar dependências de um repositório do CodeArtifact, você deve editar o arquivo de configuração do Maven, `settings.xml`.

1. Em `settings.xml`, adicione uma seção `<servers>` com uma referência à variável de ambiente `CODEARTIFACT_AUTH_TOKEN` para que o Clojure passe o token nas solicitações HTTP.

   > **Note**
   >
   > O Clojure supõe que o arquivo settings.xml esteja localizado em `~/.m2/settings.xml`. Se estiver em outro lugar, crie o arquivo nesse local.

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

2. Se você ainda não tiver um, gere um POM xml para seu projeto usando `clj -Spom`.
3. No seu arquivo de configuração `deps.edn`, adicione um repositório que corresponda ao ID do servidor do 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` garante que os repositórios `central` e `clojars` sejam verificados primeiro para as bibliotecas do Maven. Depois disso, os outros repositórios listados em `deps.edn` serão verificados.
> - Para evitar o download direto do Clojars e do Maven Central, `central` e `clojars` precisam ser configurados como `nil`.

Verifique se você tem o token de autenticação do CodeArtifact em uma variável de ambiente (consulte Passar um token de autenticação usando uma variável de ambiente). Ao criar o pacote após essas alterações, as dependências em `deps.edn` serão obtidas do CodeArtifact.

> **Note**
>
> Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

Publicar artefatos

1. Atualize suas configurações do Maven e `deps.edn` para incluir o CodeArtifact como um servidor reconhecido pelo Maven (consulte Buscar dependências). Você pode usar uma ferramenta como deps-deploy para fazer upload de artefatos no CodeArtifact.
2. No seu `build.clj`, adicione uma tarefa `deploy` para carregar os artefatos necessários no repositório `codeartifact` configurado anteriormente.

```
(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 o artefato executando o comando: `clj -T:build deploy`

Para obter mais informações sobre a modificação de repositórios padrão, consulte Modificar repositórios padrão no Clojure Deps e Lógica da referência da CLI.

Publicar com curl

Esta seção mostra como usar o cliente HTTP `curl` para publicar artefatos do Maven em um repositório do CodeArtifact. A publicação de artefatos com `curl` pode ser útil se você não tiver ou quiser instalar o cliente Maven em seus ambientes.

Publicar um artefato do Maven com **`curl`**

1. É possível buscar um token de autorização do CodeArtifact seguindo as etapas em Passar um token de autenticação usando uma variável de ambiente e voltar para essas etapas.

2. Use o seguinte comando `curl` para publicar o JAR em um repositório do CodeArtifact:

   Em cada um dos comandos `curl` desse procedimento, substitua os seguintes espaços reservados:

   - Substitua *`my_domain`* pelo seu nome de domínio do CodeArtifact.
   - Substitua *`111122223333`* pelo ID do proprietário do seu domínio do CodeArtifact.
   - Substitua *`us-west-2`* pela região na qual seu domínio do CodeArtifact está localizado.
   - Substitua *`my_repo`* pelo nome do seu repositório do 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**
   >
   > Você deve prefixar o valor do parâmetro `--data-binary` com um caractere `@`. Ao colocar o valor entre aspas, `@` deve ser incluído entre aspas.

3. Use o seguinte comando `curl` para publicar o POM em um repositório do 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. Nesse ponto, o artefato do Maven estará no seu repositório do CodeArtifact com o status de `Unfinished`. Para poder consumir o pacote, ele deve estar no estado `Published`. Você pode mover o pacote de `Unfinished` para `Published` fazendo o upload de um arquivo `maven-metadata.xml` em seu pacote ou chamando a API UpdatePackageVersionsStatus para alterar o status.

   a. Opção 1: use o comando `curl` a seguir para adicionar um arquivo `maven-metadata.xml` ao seu pacote:

```
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
```

Um exemplo dos conteúdos em um arquivo `maven-metadata.xml` é mostrado a seguir:

```
<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. Opção 2: atualize o status do pacote para `Published` com a 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
```

Se você tiver apenas o arquivo JAR de um artefato, poderá publicar uma versão de pacote consumível em um repositório do CodeArtifact usando `mvn`. Isso pode ser útil se você não tiver acesso ao código-fonte do artefato ou ao POM. Para mais detalhes, consulte Publicar artefatos de terceiros.

Usar somas de verificação do Maven

Quando um artefato Maven é publicado em um AWS CodeArtifact repositório, a soma de verificação associada a cada ativo ou arquivo no pacote é usada para validar o upload. Os arquivos jar, pom e war são exemplos de ativos. Para cada ativo, o artefato do Maven contém vários arquivos de soma de verificação que usam o nome do ativo com uma extensão adicional, como `md5` ou `sha1`. Por exemplo, os arquivos de soma de verificação de um arquivo chamado `my-maven-package.jar` podem ser `my-maven-package.jar.md5` e `my-maven-package.jar.sha1`.

> **Note**
>
> O Maven usa o termo`artifact`. Neste guia, um pacote Maven é o mesmo que um artefato do Maven. Para obter mais informações, consulte o AWS CodeArtifactpacote.

Armazenamento da soma de verificação

CodeArtifact não armazena somas de verificação do Maven como ativos. Isso significa que as somas de verificação não aparecem como ativos individuais na saída da ListPackageVersionAssets API. Em vez disso, as somas de verificação calculadas por CodeArtifact estão disponíveis para cada ativo em todos os tipos de soma de verificação compatíveis. Por exemplo, parte da resposta da chamada ListPackageVersionAssets na versão do pacote Maven `commons-lang:commons-lang 2.1` é:

```
{
    "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
    }
}
```

Mesmo que as somas de verificação não sejam armazenadas como ativos, os clientes do Maven ainda podem publicar e baixar as somas de verificação nos locais esperados. Por exemplo, se `commons-lang:commons-lang 2.1` estivesse em um repositório chamado `maven-repo`, o caminho do URL para a soma de verificação SHA-256 do arquivo JAR seria:

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

Se você estiver fazendo o upload de pacotes Maven existentes (por exemplo, pacotes previamente armazenados no Amazon S3) CodeArtifact para usar um cliente HTTP genérico, `curl` como, por exemplo, não é necessário carregar as somas de verificação. CodeArtifact os gerará automaticamente. Se quiser verificar se os ativos foram carregados corretamente, você pode usar a operação da ListPackageVersionAssets API para comparar as somas de verificação na resposta com os valores originais de cada ativo.

Incompatibilidades da soma de verificação durante a publicação

Além dos ativos e das somas de verificação, os artefatos do Maven também contêm um arquivo `maven-metadata.xml`. A sequência normal de publicação de um pacote Maven é que todos os ativos e somas de verificação sejam carregados primeiro, seguidos por `maven-metadata.xml`. Por exemplo, a sequência de publicação da versão `commons-lang 2.1` do pacote Maven descrita anteriormente, supondo que o cliente tenha sido configurado para publicar arquivos da soma de verificação SHA-256, seria:

```
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
```

Ao carregar o arquivo de soma de verificação de um ativo, como um arquivo JAR, a solicitação de upload da soma de verificação falhará com uma resposta 400 (Solicitação inválida) se houver uma incompatibilidade entre o valor da soma de verificação carregada e o valor da soma de verificação calculado por. CodeArtifact Se o ativo correspondente não existir, a solicitação falhará com a resposta 404 (Não encontrado). Para evitar esse erro, você deve primeiro carregar o ativo e, em seguida, fazer o upload da soma de verificação.

Quando `maven-metadata.xml` é carregado, CodeArtifact normalmente muda o status da versão do pacote Maven de `Unfinished` para`Published`. Se uma incompatibilidade de soma de verificação for detectada para qualquer ativo, CodeArtifact retornará 400 (solicitação inválida) em resposta à solicitação de `maven-metadata.xml` publicação. Esse erro pode impedir que o cliente faça upload de arquivos para essa versão do pacote. Se isso ocorrer e o arquivo `maven-metadata.xml` não for carregado, nenhum ativo da versão do pacote já carregado poderá ser baixado. Isso ocorre porque o status da versão do pacote não está definido como `Published` e permanece como `Unfinished`.

CodeArtifact permite adicionar mais ativos a uma versão do pacote Maven mesmo após `maven-metadata.xml` o upload e o status da versão do pacote ter sido definido como. `Published` Nesse status, uma solicitação para carregar um arquivo de soma de verificação incompatível também falhará com uma resposta 400 (Solicitação inválida). No entanto, como o status da versão do pacote já foi definido como `Published`, você pode baixar qualquer ativo do pacote, incluindo aqueles para os quais o upload do arquivo de soma de verificação falhou. Ao baixar uma soma de verificação para um ativo em que o upload do arquivo de soma de verificação falhou, o valor da soma de verificação que o cliente receberá será o valor da soma de verificação calculado CodeArtifact com base nos dados do ativo carregado.

CodeArtifact as comparações de somas de verificação diferenciam maiúsculas de minúsculas e as somas de verificação calculadas por CodeArtifact são formatadas em minúsculas. Portanto, se a soma de verificação `909FA780F76DA393E992A3D2D495F468` for carregada, ela falhará com uma incompatibilidade de soma de verificação porque CodeArtifact não a trata como igual a. `909fa780f76da393e992a3d2d495f468`

Recuperação no caso de incompatibilidades da soma de verificação

Se o upload de uma soma de verificação falhar como resultado de uma incompatibilidade, tente uma das seguintes opções de recuperação:

- Execute o comando que publica o artefato do Maven novamente. Isso pode funcionar se um problema de rede tiver corrompido o arquivo de soma de verificação. Se isso resolver o problema de rede, a soma de verificação será correspondente e o download será concluído.
- Exclua a versão do pacote e, em seguida, faça uma nova publicação. Para obter mais informações, consulte DeletePackageVersionsa AWS CodeArtifact API Reference.

Usar snapshot do Maven

Um snapshot do Maven é uma versão especial de um pacote do Maven que se refere ao código de ramificação de produção mais recente. É uma versão de desenvolvimento que precede a versão final de lançamento. Você pode identificar a versão de snapshot de um pacote do Maven pelo sufixo `SNAPSHOT` anexado à versão do pacote. Por exemplo, o snapshot da versão `1.1` é `1.1-SNAPSHOT`. Para obter mais informações, consulte O que é uma versão de SNAPSHOT? no site do Apache Maven Project.

AWS CodeArtifact suporta a publicação e o consumo de instantâneos do Maven. Instantâneos exclusivos que usam um número de versão baseado em tempo são os únicos instantâneos compatíveis. CodeArtifact não suporta instantâneos não exclusivos que são gerados por clientes Maven 2. Você pode publicar um snapshot compatível do Maven em qualquer CodeArtifact repositório.

Tópicos

Publicação de instantâneos em CodeArtifact

AWS CodeArtifact suporta os padrões de solicitação que os clientes, por exemplomvn, usam ao publicar instantâneos. Por isso, você pode seguir a documentação da sua ferramenta de compilação ou gerenciador de pacotes sem ter uma compreensão detalhada de como os snapshots do Maven são publicados. Se você estiver fazendo algo mais complexo, esta seção descreve em detalhes como CodeArtifact lidar com instantâneos.

Quando um snapshot do Maven é publicado em um CodeArtifact repositório, sua versão anterior é preservada em uma nova versão chamada compilação. Cada vez que um snapshot do Maven é publicado, uma nova versão de compilação é criada. Todas as versões anteriores de um snapshot são mantidas em suas versões de compilação. Quando um snapshot do Maven é publicado, o status da versão do pacote é definido como `Published` e o status da compilação que contém a versão anterior é definido como `Unlisted`. Esse comportamento se aplica somente às versões do pacote Maven onde o sufixo é `-SNAPSHOT`.

Por exemplo, versões instantâneas de um pacote maven chamado `com.mycompany.myapp:pkg-1` são carregadas em um CodeArtifact repositório chamado. `my-maven-repo` A versão de snapshot é `1.0-SNAPSHOT`. Até o momento, nenhuma versão do `com.mycompany.myapp:pkg-1` foi publicada. Primeiro, os ativos da compilação inicial são publicados nos seguintes caminhos:

```
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
```

Observe que o timestamp `20210728.194552-1` é gerado pelo cliente que publica as compilações do snapshot.

Depois que os arquivos .pom e .jar forem carregados, a única versão do `com.mycompany.myapp:pkg-1` presente no repositório será `1.0-20210728.194552-1`. Isso acontece mesmo que a versão especificada no caminho anterior seja `1.0-SNAPSHOT`. O status da versão do pacote nesse momento é `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"
}
```

Em seguida, o cliente faz o upload do arquivo `maven-metadata.xml` para a versão do pacote:

```
PUT my-maven-repo/com/mycompany/myapp/pkg-1/1.0-SNAPSHOT/maven-metadata.xml
```

Quando o arquivo maven-metadata.xml é carregado com sucesso, CodeArtifact cria a versão do `1.0-SNAPSHOT` pacote e define a `1.0-20210728.194552-1` versão como`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"
}
```

Nesse ponto, a versão de snapshot `1.0-SNAPSHOT` pode ser consumida em uma compilação. Embora existam duas versões do `com.mycompany.myapp:pkg-1` no repositório `my-maven-repo`, ambas contêm os mesmos ativos.

```
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"
]
```

Executar o mesmo comando `list-package-version-assets` mostrado anteriormente com o parâmetro `--package-version` alterado para `1.0-20210728.194552-1` resulta em uma saída idêntica.

À medida que outras compilações do `1.0-SNAPSHOT` são adicionadas ao repositório, uma nova versão `Unlisted` do pacote é criada para cada nova compilação. Os ativos da versão `1.0-SNAPSHOT` são atualizados todas as vezes, para que a versão sempre se refira à compilação mais recente. A atualização do `1.0-SNAPSHOT` com os ativos mais recentes é iniciada com o upload do arquivo `maven-metadata.xml` para a nova compilação.

Consumir versões de snapshot

Se você solicitar um snapshot, a versão com o status `Published` será retornada. É sempre a versão mais recente de snapshot do Maven. Você também pode solicitar uma compilação específica de um snapshot usando o número da versão da compilação (por exemplo, `1.0-20210728.194552-1`) em vez da versão de snapshot (por exemplo, `1.0-SNAPSHOT`) no caminho do URL. Para ver as versões de compilação de um snapshot do Maven, use a ListPackageVersions API no Guia da CodeArtifact API e defina o parâmetro de status como. `Unlisted`

Excluir versões de snapshot

Para excluir todas as versões de compilação de um snapshot do Maven, use a DeletePackageVersionsAPI, especificando as versões que você deseja excluir.

Publicação de snapshot com curl

Se você tiver versões de snapshots existentes armazenadas no Amazon Simple Storage Service (Amazon S3) ou em outro produto de repositório de artefatos, talvez queira republicá-las no. AWS CodeArtifact Devido à forma como CodeArtifact suporta instantâneos do Maven (consulte Publicação de instantâneos em CodeArtifact), publicar instantâneos com um cliente HTTP genérico, como o,

`curl` é mais complexo do que publicar versões de lançamento do Maven, conforme descrito em. Publicar com curl Observe que esta seção não será relevante se você estiver criando e implantando versões de snapshot com um cliente Maven como `mvn` ou `gradle`. É preciso seguir a documentação desse cliente.

Publicar uma versão de snapshot envolve a publicação de uma ou mais compilações. Em CodeArtifact, se houver n compilações de uma versão de instantâneo, haverá n + 1 CodeArtifact versões: n versões de compilação, todas com um status de`Unlisted`, e uma versão de instantâneo (a última compilação publicada) com um status de. `Published` A versão de snapshot (ou seja, a versão com uma string de versão que contém “-SNAPSHOT”) contém um conjunto de ativos idêntico ao da compilação mais recente publicada. A forma mais fácil de criar essa estrutura usando `curl` é a seguinte:

1. Publique todos os ativos de todas as compilações usando `curl`.
2. Publique o arquivo `maven-metadata.xml` da compilação mais recente (ou seja, a compilação com a marca de data e hora mais recente) com `curl`. Isso criará uma versão com “`-SNAPSHOT`” na string da versão e com o conjunto correto de ativos.
3. Use a UpdatePackageVersionsStatusAPI para definir o status de todas as versões de compilação não mais recentes como`Unlisted`.

Use os comandos `curl` a seguir para publicar ativos de snapshots (como arquivos. jar e .pom) para a versão `1.0-SNAPSHOT` de snapshot de um pacote `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
```

Ao usar esses exemplos:

- *my_domain*Substitua pelo seu nome de CodeArtifact domínio.
- *111122223333*Substitua pelo Conta da AWS ID do proprietário do seu CodeArtifact domínio.
- *us-west-2*Substitua pelo Região da AWS em que seu CodeArtifact domínio está localizado.
- *my_maven_repo*Substitua pelo nome CodeArtifact do seu repositório.

> **Important**
>
> Será preciso prefixar o valor do parâmetro `--data-binary` com o caractere `@`. Ao colocar o valor entre aspas, `@` deve ser incluído entre aspas.

Você pode ter mais de dois ativos para carregar em cada compilação. Por exemplo, pode haver arquivos Javadoc e JAR de origem, além do JAR principal e `pom.xml`. Não é necessário publicar arquivos de soma de verificação para os ativos da versão do pacote porque gera CodeArtifact automaticamente somas de verificação para cada ativo carregado. Para verificar se os ativos foram carregados corretamente, busque as somas de verificação geradas usando o comando `list-package-version-assets` e compare-as com as originais. Para obter mais informações sobre como CodeArtifact manipula as somas de verificação do Maven, consulte. Usar somas de verificação do Maven

Use o comando curl a seguir para publicar o arquivo `maven-metadata.xml` para a versão de compilação mais recente:

```
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
```

O arquivo `maven-metadata.xml` deve fazer referência a pelo menos um dos ativos na versão de compilação mais recente do elemento `<snapshotVersions>`. Além disso, o valor `<timestamp>` deve estar presente e corresponder ao timestamp nos nomes dos arquivos do ativo. Por exemplo, para a compilação de `20210729.171330-2` publicada anteriormente, o conteúdo de `maven-metadata.xml` deve ser:

```
<?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>
```

Depois da publicação de `maven-metadata.xml`, a última etapa é definir que todas as outras versões de compilação (ou seja, todas as versões de compilação, exceto a compilação mais recente) tenham o status de versão do pacote de `Unlisted`. Por exemplo, se a versão `1.0-SNAPSHOT` tiver duas compilações, sendo a primeira compilação `20210728.194552-1`, o comando para definir essa compilação como `Unlisted` será:

```
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
```

Snapshots e conexões externas

Os instantâneos do Maven não podem ser obtidos de um repositório público do Maven por meio de uma conexão externa. AWS CodeArtifact só suporta a importação de versões de lançamento do Maven.

Snapshots e repositórios upstream

Em geral, os snapshots do Maven funcionam da mesma forma que as versões de lançamento do Maven, quando usadas com repositórios upstream. Porém, há uma limitação se você planeja publicar snapshots da mesma versão do pacote em dois repositórios que tenham uma relação upstream. Por exemplo, digamos que há dois repositórios em um AWS CodeArtifact domínio `R` e`U`, onde `U` está um upstream de. `R` Se você publicar uma nova compilação`R`, quando um cliente Maven solicitar a compilação mais recente dessa versão de snapshot, CodeArtifact retornará a versão mais recente de. `U` Isso pode ser inesperado, já que a versão mais recente agora é `R`, não `U`. Há duas maneiras de evitar isso:

1. Não publique compilações de uma versão de snapshot, como `1.0-SNAPSHOT` em `R`, se `1.0-SNAPSHOT` existir em `U`.
2. Use os controles de origem CodeArtifact do pacote para desativar os upstreams desse pacote em. `R` Isso permitirá que você publique compilações do `1.0-SNAPSHOT` em `R`, mas também impedirá que `R` obtenha outras versões desse pacote de `U` que ainda não tenham sido retidas.

Solicitação de pacotes Maven de upstreams e conexões externas

Importação de nomes de ativos padrão

Ao importar uma versão do pacote Maven de um repositório público, como o Maven Central, a AWS CodeArtifact tenta importar todos os ativos dessa versão do pacote. Conforme descrito em Solicitar uma versão do pacote com repositórios upstream, a importação ocorre quando:

- Um cliente solicita um ativo Maven de um CodeArtifact repositório.
- A versão do pacote ainda não está presente em seu repositório ou em upstreams.
- Há uma conexão externa acessível com um repositório público do Maven.

Mesmo que o cliente tenha solicitado apenas um ativo, CodeArtifact tenta importar todos os ativos encontrados para essa versão do pacote. A forma como CodeArtifact descobrir quais ativos estão disponíveis para uma versão do pacote Maven depende do repositório público específico. Alguns repositórios públicos do Maven oferecem suporte à solicitação de uma lista de ativos, mas outros não. Para repositórios que não fornecem uma forma de listar ativos, CodeArtifact gera um conjunto de nomes de ativos que provavelmente existem. Por exemplo, quando qualquer ativo da versão do pacote Maven `junit 4.13.2` for solicitado, CodeArtifact tentará importar os seguintes ativos:

- `junit-4.13.2.pom`
- `junit-4.13.2.jar`
- `junit-4.13.2-javadoc.jar`
- `junit-4.13.2-sources.jar`

Importação de nomes de ativos não padrão

Quando um cliente Maven solicita um ativo que não corresponde a um dos padrões descritos acima, CodeArtifact verifica se esse ativo está presente no repositório público. Se o ativo estiver presente, ele será importado e adicionado ao registro da versão do pacote existente, se houver. Por exemplo, a versão `com.android.tools.build:aapt2 7.3.1-8691043` do pacote Maver contém os seguintes ativos:

- `aapt2-7.3.1-8691043.pom`
- `aapt2-7.3.1-8691043-windows.jar`
- `aapt2-7.3.1-8691043-osx.jar`
- `aapt2-7.3.1-8691043-linux.jar`

Quando um cliente solicita o arquivo POM, CodeArtifact se não conseguir listar os ativos da versão do pacote, o POM será o único ativo importado. Isso ocorre porque nenhum dos outros ativos corresponde aos padrões de nomes de ativos padrão. No entanto, quando o cliente solicita um dos ativos JAR, esse ativo será importado e adicionado à versão do pacote existente armazenada em CodeArtifact. As versões do pacote no repositório mais downstream (o repositório no qual o cliente fez a solicitação) e no repositório com a conexão externa anexada serão atualizadas para conter o novo ativo, conforme descrito em Retenção de pacotes de repositórios upstream.

Normalmente, quando uma versão do pacote é retida em um CodeArtifact repositório, ela não é afetada pelas alterações nos repositórios upstream. Para obter mais informações, consulte Retenção de pacotes de repositórios upstream. No entanto, o comportamento dos ativos do Maven com nomes não padrão descritos anteriormente é uma exceção a essa regra. Embora a versão downstream do pacote não mude sem que um ativo adicional seja solicitado por um cliente, nessa situação, a versão retida do pacote é modificada após ser inicialmente retida e, portanto, não é imutável. Esse comportamento é necessário porque ativos do Maven com nomes não padrão, de outra forma, não estariam acessíveis por meio do. CodeArtifact O comportamento também é ativado se eles forem

adicionados a uma versão do pacote Maven em um repositório público após a versão do pacote ter sido retida em um repositório. CodeArtifact

Verificação das origens dos ativos

Ao adicionar um novo ativo a uma versão de pacote Maven retida anteriormente, CodeArtifact confirma que a origem da versão retida do pacote é a mesma origem do novo ativo. Isso impede a criação de uma versão de pacote “mista” em que diferentes ativos são originários de diferentes repositórios públicos. Sem essa verificação, a mistura de ativos pode ocorrer se uma versão do pacote Maven for publicada em mais de um repositório público e esses repositórios fizerem parte do gráfico upstream de um CodeArtifact repositório.

Importação de novos ativos e status da versão do pacote em repositórios upstream

O status da versão do pacote das versões do pacote nos repositórios upstream pode CodeArtifact impedir a retenção dessas versões nos repositórios downstream.

Por exemplo, digamos que um domínio tenha três repositórios:`repo-A`, `repo-B` e `repo-C`, onde `repo-B` é um upstream de `repo-A` e `repo-C` é upstream de `repo-B`.

A versão `7.3.1` do pacote Maven `com.android.tools.build:aapt2` está presente em `repo-B` e tem o status de `Published`. Não está presente em `repo-A`. Se um cliente solicitar um ativo dessa versão do pacote do `repo-A`, a resposta será 200 (OK) e a versão `7.3.1` do pacote Maven será retida no `repo-A`. No entanto, se o status da versão `7.3.1` do pacote no `repo-B` for `Archived` ou`Disposed`, a resposta será 404 (Não encontrado), pois os ativos das versões do pacote nesses dois status não podem ser baixados.

Observe que definir o controle de origem do pacote como `upstream=BLOCK` para `com.android.tools.build:aapt2` no `repo-A`, `repo-B` e `repo-C` evitará que novos ativos sejam buscados para todas as versões desse pacote do `repo-A`, independentemente do status da versão do pacote.

Solução de problemas do Maven

As informações a seguir podem ajudar a solucionar problemas comuns com o Maven e o CodeArtifact.

Desative as opções paralelas para corrigir o erro 429: Excesso de solicitações

A partir da versão 3.9.0, o Maven carrega artefatos de pacotes em paralelo (até 5 arquivos por vez). Isso pode fazer com que o CodeArtifact responda ocasionalmente com um código de resposta de erro 429 (Excesso de solicitações). Se você se deparar com esse erro, pode desabilitar as opções paralelas para corrigi-lo.

Para desativar as opções paralelas, defina a `aether.connector.basic.parallelPut` propriedade como `false` em seu perfil em seu arquivo `settings.xml`, conforme mostrado no exemplo a seguir:

```
<settings>
    <profiles>
        <profile>
            <id>default</id>
            <properties>
                <aether.connector.basic.parallelPut>false</
aether.connector.basic.parallelPut>
            </properties>
        </profile>
    </profiles>
<settings>
```

Para obter mais informações, consulte Opções de configuração do resolvedor de artefatos na documentação do Maven.

Usando o CodeArtifact com npm

Esses tópicos descrevem como usar npm, o gerenciador de pacotes Node.js, com o CodeArtifact.

> **Note**
>
> O CodeArtifact é compatível com o `node v4.9.1` e posterior e com o `npm v5.0.0` e posterior.

Tópicos

Configure e use o npm com CodeArtifact

Depois de criar um repositório no CodeArtifact, você pode usar o cliente npm para instalar e publicar pacotes. O método recomendado para configurar o npm com o endpoint e o token de autorização do repositório é usando o comando `aws codeartifact login`. Também é possível configurar o npm manualmente.

Sumário

Configuração do npm com o comando login

Use o comando `aws codeartifact login` para buscar credenciais para uso com o npm.

> **Note**
>
> Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

> **Important**
>
> Se você estiver usando o npm 10.x ou mais recente, deverá usar a AWS CLI versão 2.9.5 ou mais recente para executar o comando com êxito. `aws codeartifact login`

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

Esse comando faz as seguintes alterações em seu arquivo ~/.npmrc:

- Adiciona um token de autorização depois de buscá-lo CodeArtifact usando suas AWS credenciais.
- Define o registro npm para o repositório especificado pela opção `--repository`.
- Para npm 6 e inferior: adiciona `"always-auth=true"` para que o token de autorização seja enviado para cada comando npm.

O período de autorização padrão após chamar o `login` é de 12 horas e o `login` deve ser chamado para atualizar o token periodicamente. Para obter mais informações sobre o token de autorização criado com o comando `login`, consulte Tokens criados com o comando `login`.

Configuração do npm sem usar o comando login

Você pode configurar o npm com seu CodeArtifact repositório sem o `aws codeartifact login` comando atualizando manualmente a configuração do npm.

Para configurar o npm sem usar o comando login

1. Em uma linha de comando, busque um token de CodeArtifact autorização e armazene-o em uma variável de ambiente. O npm usará esse token para se autenticar no seu repositório. CodeArtifact

   > **Note**
   >
   > O comando a seguir é para máquinas macOS ou Linux. Para ver informações sobre como configurar variáveis de ambiente em uma máquina Windows, consulte Passar um token de autenticação usando uma variável de ambiente.

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

2. Obtenha o endpoint do seu CodeArtifact repositório executando o comando a seguir. O endpoint do repositório é usado para direcionar o npm ao seu repositório para instalar ou publicar pacotes.

   - *my_domain*Substitua pelo seu nome de CodeArtifact domínio.
   - *111122223333*Substitua pelo ID da AWS conta do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
   - *my_repo*Substitua pelo nome CodeArtifact do seu repositório.

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

   O URL a seguir é um exemplo de endpoint de repositório.

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

> **Important**
>
> O URL de registro deve terminar com uma barra inclinada (/). Do contrário, você não poderá se conectar ao repositório.

3. Use o `npm config set` comando para definir o registro no seu CodeArtifact repositório. Substitua o URL pelo URL do endpoint do repositório da etapa anterior.

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

> **Note**
>
> Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

4. Use o comando `npm config set` para adicionar seu token de autorização à configuração do npm.

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

Para npm 6 ou inferior: para fazer com que o npm sempre passe o token de autenticação para CodeArtifact, mesmo para `GET` solicitações, definir a variável de `always-auth` configuração com. `npm config set`

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

Exemplo de arquivo de configuração do npm (**`.npmrc`**)

Veja a seguir um exemplo de `.npmrc` arquivo após seguir as instruções anteriores para definir o endpoint CodeArtifact do registro, adicionar um token de autenticação e 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
```

Execução de comandos npm

Depois de configurar o cliente npm, você pode executar comandos npm. Supondo que um pacote esteja presente em seu repositório ou em um de seus repositórios upstream, você pode instalá-lo com `npm install`. Por exemplo, use o seguinte para instalar o pacote `lodash`.

```
npm install lodash
```

Use o comando a seguir para publicar um novo pacote npm em um CodeArtifact repositório.

```
npm publish
```

Para ver informações sobre como criar pacotes npm, consulte Criação de módulos Node.js no site de documentação do npm. Para obter uma lista dos comandos npm compatíveis com CodeArtifact, consulte npm Command Support.

Verificar autorização e autenticação de npm

Invocar o comando `npm ping` é uma forma de verificar o seguinte:

- Você configurou corretamente suas credenciais para poder se autenticar em um CodeArtifact repositório.
- A configuração de autorização concede a você a permissão `ReadFromRepository`.

O resultado de uma invocação bem-sucedida de `npm ping` se parece com o seguinte.

```
$ 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
```

A opção `-d` faz com que o npm imprima informações adicionais de depuração, incluindo o URL do repositório. Essas informações facilitam a confirmação de que o npm está configurado para usar o repositório esperado.

Mudança de volta para o registro npm padrão

Configurar o npm com CodeArtifact define o registro npm para o repositório especificado. CodeArtifact Você pode executar o comando a seguir para definir o registro npm de volta ao registro padrão quando terminar de se conectar a. CodeArtifact

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

Solução de problemas de instalações lentas com npm 8.x ou posterior

Há um problema conhecido nas versões 8.x e posteriores do npm em que, se uma solicitação for feita para um repositório de pacotes e o repositório redirecionar o cliente para o Amazon S3 em vez de transmitir os ativos diretamente, o cliente npm poderá travar por vários minutos por dependência.

Como CodeArtifact os repositórios são projetados para sempre redirecionar a solicitação para o Amazon S3, às vezes esse problema ocorre, o que causa longos tempos de compilação devido aos longos tempos de instalação do npm. Casos desse comportamento serão apresentados como uma barra de progresso exibida por vários minutos.

Para evitar esse problema, use os sinalizadores `--no-progress` ou `progress=false` com os comandos `npm` da CLI, conforme mostrado no exemplo a seguir.

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

Configurar e usar o Yarn com o CodeArtifact

Depois de criar um repositório, você pode usar o cliente Yarn para gerenciar pacotes npm.

> **Note**
>
> O `Yarn 1.X` lê e usa informações do seu arquivo de configuração npm (.npmrc), o que não acontece com o `Yarn 2.X`. A configuração do `Yarn 2.X` deve ser definida no arquivo .yarnrc.yml.

Sumário

Configure o Yarn 1.X com o comando **`aws codeartifact login`**

Para o `Yarn 1.X`, você pode configurar o Yarn com o CodeArtifact usando o comando `aws codeartifact login`. O comando `login` configurará o arquivo ~/.npmrc com as informações e credenciais do endpoint do repositório do CodeArtifact. Com o `Yarn 1.X`, os comandos `yarn` usam as informações de configuração do arquivo ~/.npmrc.

Para configurar o **`Yarn 1.X`** com o comando login

1. Caso ainda não tenha feito isso, configure suas credenciais AWS para uso com o AWS CLI, conforme descrito em Conceitos básicos do CodeArtifact.
2. Para executar o comando `aws codeartifact login`, o npm deve estar instalado. Consulte Baixar e instalar o Node.js e o npm na documentação do npm para obter instruções de instalação.
3. Use o comando `aws codeartifact login` para buscar as credenciais do CodeArtifact e configurar o arquivo ~/.npmrc.
   - Substitua *`my_domain`* pelo seu nome de domínio do CodeArtifact.
   - Substitua *`111122223333`* pelo ID da conta da AWS do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
   - Substitua *`my_repo`* pelo nome do seu repositório do CodeArtifact.

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

O comando `login` faz as seguintes alterações no arquivo ~/.npmrc:

- Adiciona um token de autorização depois de buscá-lo no CodeArtifact usando suas credenciais da AWS.
- Define o registro npm para o repositório especificado pela opção `--repository`.
- Para npm 6 e inferior: adiciona "`always-auth=true`" para que o token de autorização seja enviado para cada comando npm.

O período de autorização padrão após a chamada do `login` é de 12 horas e `login` deve ser chamado para atualizar periodicamente o token. Para obter mais informações sobre o token de autorização criado com o comando `login`, consulte Tokens criados com o comando `login`.

4. Para o npm 7.X e 8.X, você deve adicionar `always-auth=true` ao arquivo ~/.npmrc para usar o Yarn.

   - Abra o arquivo ~/.npmrc em um editor de texto e adicione `always-auth=true` em uma nova linha.

Você pode usar o comando `yarn config list` para verificar se o Yarn está usando a configuração correta. Depois de executar o comando, verifique os valores na seção `info npm config`. O conteúdo deve ser semelhante ao trecho a seguir.

```
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 o Yarn 2.X com o comando **`yarn config set`**

O procedimento a seguir detalha como configurar o `Yarn 2.X` atualizando a configuração do `.yarnrc.yml` na linha de comando com o comando `yarn config set`.

Para atualizar a configuração do **`yarnrc.yml`** a partir da linha de comando

1. Caso ainda não tenha feito isso, configure suas credenciais AWS para uso com o AWS CLI, conforme descrito em Conceitos básicos do CodeArtifact.
2. Execute o comando `aws codeartifact get-repository-endpoint` para obter o endpoint do seu repositório do CodeArtifact.

   - Substitua *`my_domain`* pelo seu nome de domínio do CodeArtifact.
   - Substitua *`111122223333`* pelo ID da conta da AWS do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
   - Substitua *`my_repo`* pelo nome do seu repositório do CodeArtifact.

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

3. Atualize o valor `npmRegistryServer` no arquivo .yarnrc.yml com o endpoint do repositório.

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

4. Obtenha um token de autorização do CodeArtifact e armazene-o em uma variável de ambiente.

   > **Note**
   >
   > O comando a seguir é para máquinas macOS ou Linux. Para ver informações sobre como configurar variáveis de ambiente em uma máquina Windows, consulte Passar um token de autenticação usando uma variável de ambiente.

   - Substitua *`my_domain`* pelo seu nome de domínio do CodeArtifact.

- Substitua *111122223333* pelo ID da conta da AWS do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
- Substitua *my_repo* pelo nome do seu repositório do CodeArtifact.

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

5. Use o comando `yarn config set` para adicionar seu token de autenticação do CodeArtifact ao arquivo .yarnrc.yml. Substitua o URL no comando a seguir pelo URL do endpoint do repositório da Etapa 2.

```
yarn config set
 'npmRegistries["https://my_domain-
111122223333.d.codeartifact.region.amazonaws.com/npm/my_repo/"].npmAuthToken'
 "${CODEARTIFACT_AUTH_TOKEN}"
```

6. Use o comando `yarn config set` para definir o valor de `npmAlwaysAuth` como `true`. Substitua o URL no comando a seguir pelo URL do endpoint do repositório da Etapa 2.

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

Depois da configuração, o arquivo .yarnrc.yml deve ter um conteúdo semelhante ao seguinte trecho.

```
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/"
```

Você também pode usar o comando `yarn config` para verificar os valores de `npmRegistries` e `npmRegistryServer`.

Suporte para comandos npm

As seções a seguir resumem os comandos npm que são suportados pelos CodeArtifact repositórios, além dos comandos específicos que não são suportados.

Sumário

Comandos compatíveis que interagem com um repositório

Esta seção lista os comandos npm em que o cliente npm faz uma ou mais solicitações ao registro com o qual foi configurado (por exemplo, com `npm config set registry`). Foi verificado que esses comandos funcionam corretamente quando invocados em um CodeArtifact repositório.

Command	Description
bugs	Tenta adivinhar a localização do URL do rastreador de bugs de um pacote e, em seguida, tenta abri-lo.
ci	Instala um projeto do zero.
deprecate	Deprecia uma versão de um pacote.
dist-tag	Modifica as tags de distribuição do pacote.
docs	Tenta adivinhar a localização do URL de documentação de um pacote e, em seguida, tenta abri-lo usando o parâmetro de configura ção `--browser` .
doctor	Executa um conjunto de verificações para garantir que sua instalação do npm tenha o que precisa para gerenciar seus JavaScript pacotes.

Command	Description
install	Instala um pacote.
install-ci-test	Instala um projeto do zero e executa testes. Alias: `npm cit`. Esse comando executa um `npm ci`, seguido imediatamente por um `npm test`.
install-test	Instala o pacote e executa testes. Executa um `npm install`, seguido imediatamente por um `npm test`.
outdated	Verifica o registro configurado para ver se algum pacote instalado está desatualizado no momento.
ping	Faz ping no registro npm configurado ou fornecido e verifica a autenticação.
publish	Publica uma versão do pacote no registro.
update	Adivinha a localização do URL do repositór io de um pacote e, em seguida, tenta abri- lo usando o parâmetro de configuração `--browser` .
view	Exibe os metadados do pacote. Pode ser usado para imprimir propriedades de metadados.

Comandos do lado do cliente compatíveis

Esses comandos não exigem nenhuma interação direta com um repositório, portanto, CodeArtifact não é necessário fazer nada para suportá-los.

Command	Description
build	Cria um pacote.
cache	Manipula o cache de pacotes.
completion	Ativa o preenchimento de guias em todos os comandos npm.
config	Atualiza o conteúdo do usuário e dos arquivos `npmrc` globais.
dedupe	Pesquisa a árvore de pacotes local e tenta simplificar a estrutura movendo as dependênc ias mais para cima na árvore, onde elas podem ser compartilhadas com mais eficiência por vários pacotes dependentes.
edit	Edita um pacote instalado. Seleciona uma dependência no diretório de trabalho atual e abre a pasta do pacote no editor padrão.
explore	Navega por um pacote instalado. Gera um subshell no diretório do pacote instalado especificado. Se um comando for especific ado, ele será executado no subshell, que será encerrado imediatamente.
help	Recebe ajuda no npm.
help-search	Pesquisa a documentação de ajuda do npm.
init	Cria um arquivo `package.json` .
link	Cria Symlinks de uma pasta de pacotes.
ls	Lista pacotes instalados.
pack	Cria um tarball a partir de um pacote.

Command	Description
prefix	Exibe o prefixo. Esse é o diretório pai mais próximo que contém um arquivo `package.json` , a menos que `-g` também seja especificado.
prune	Remove pacotes que não estão listados na lista de dependências do pacote pai.
rebuild	Executa o comando `npm build` nas pastas correspondentes.
restart	Executa os scripts de parada, reinicialização e inicialização de um pacote e os scripts anteriores e posteriores associados.
raiz	Imprime a pasta `node_modules` efetiva na saída padrão.
run-script	Executa scripts de pacotes arbitrários.
shrinkwrap	Bloqueia as versões de dependência para publicação.
uninstall	Desinstala um pacote.

Comandos incompatíveis

Esses comandos npm não são compatíveis com CodeArtifact repositórios.

Command	Description	Observações
access	Define o nível de acesso nos pacotes publicados.	CodeArtifact usa um modelo de permissão diferente do repositório público npmjs.

Command	Description	Observações
adduser	Adiciona uma conta de usuário de registro	CodeArtifact usa um modelo de usuário diferente do repositório público npmjs.
audit	Executa uma auditoria de segurança.	CodeArtifact atualmente não vende dados de vulnerabi lidade de segurança.
hook	Gerencia hooks do npm, incluindo adição, remoção, listagem e atualização.	CodeArtifact atualmente não oferece suporte a nenhum tipo de mecanismo de notificação de alterações.
login	Autentica um usuário. Este é um alias para `npm adduser`.	CodeArtifact usa um modelo de autenticação diferente do repositório público npmjs. Para obter mais informações, consulte Autenticação com npm.
logout	Sai do registro.	CodeArtifact usa um modelo de autenticação diferente do repositório público npmjs. Não há como sair de um CodeArtifact repositório, mas os tokens de autenticação expiram após o tempo de expiração configurável. A duração padrão do token é de 12 horas.
owner	Gerencia proprietários de pacotes.	CodeArtifact usa um modelo de permissões diferente do repositório público npmjs.

Command	Description	Observações
profile	Altera as configurações no seu perfil de registro.	CodeArtifact usa um modelo de usuário diferente do repositório público npmjs.
pesquisa	Pesquisa no registro por pacotes que correspondam aos termos de pesquisa.	CodeArtifact suporta a funcionalidade de pesquisa limitada com o comando list-packages.
star	Marca seus pacotes favoritos.	CodeArtifact atualmente não suporta nenhum tipo de mecanismo de favoritos.
stars	Exibe pacotes marcados como favoritos.	CodeArtifact atualmente não suporta nenhum tipo de mecanismo de favoritos.
team	Gerencia equipes organizac ionais e associações de equipes.	CodeArtifact usa um modelo de associação de usuário e grupo diferente do repositór io público npmjs. Para obter mais informações, consulte Identidades (usuários, grupos e perfis) no Guia do usuário do IAM.
token	Gerencia seus tokens de autenticação.	CodeArtifact usa um modelo diferente para obter tokens de autenticação. Para obter mais informações, consulte Autenticação com npm.

Command	Description	Observações
unpublish	Remove um pacote do registro.	CodeArtifact não suporta a remoção de uma versão de pacote de um repositório usando o cliente npm. Você pode usar o comando delete-package-version.
whoami	Exibe o nome do usuário npm.	CodeArtifact usa um modelo de usuário diferente do repositório público npmjs.

Tratamento de tags npm

Os registros npm são compatíveis com tags, que são aliases de string para versões de pacotes. Você pode usar tags para proporcionar um alias em vez de números de versão. Por exemplo, você pode ter um projeto com vários fluxos de desenvolvimento e usar uma tag diferente (por exemplo, `stable`, `beta`, `dev`, `canary`) para cada fluxo. Para ver mais informações, consulte dist-tag no site do npm.

Por padrão, o npm usa a tag `latest` para identificar a versão atual de um pacote. `npm install` *`pkg`* (sem especificador de `@`*`version`* ou `@`*`tag`*) instala a tag mais recente. Normalmente, os projetos usam a tag mais recente somente para versões de lançamento estáveis. Outras tags são usadas para versões instáveis ou de pré-lançamento.

Edite tags com o cliente npm

Os três comandos `npm dist-tag` (`add`, `rm` e `ls`) funcionam de forma idêntica nos repositórios do CodeArtifact, assim como no registro npm padrão.

Tags npm e a API CopyPackageVersions

Quando você usa a API `CopyPackageVersions` para copiar uma versão do pacote npm, todas as tags que dão alias a essa versão são copiadas para o repositório de destino. Quando uma versão que está sendo copiada tem uma tag que também está presente no destino, a operação de cópia define o valor da tag no repositório de destino para corresponder ao valor no repositório de origem.

Por exemplo, digamos que o repositório S e o repositório D contenham uma única versão do pacote `web-helper` com o conjunto de tags mais recente, conforme mostrado nesta tabela.

Repositório	Nome do pacote	Tags do pacote
S	`web-helper`	latest (alias para a versão 1.0.1)
D	`web-helper`	latest (alias para a versão 1.0.0)

`CopyPackageVersions` é invocada para copiar o `web-helper` 1.0.1 de S para D. Depois que a operação for concluída, a tag `latest` no repositório `web-helper` no repositório D tem o alias 1.0.1, não 1.0.0.

Se você precisar alterar as tags após a cópia, use o comando `npm dist-tag` para modificar as tags diretamente no repositório de destino. Para obter mais informações sobre a API `CopyPackageVersions`, consulte Copiar pacotes entre repositórios.

Tags npm e repositórios upstream

Quando o npm solicita as tags de um pacote e as versões desse pacote também estão presentes em um repositório upstream, o CodeArtifact mescla as tags antes de devolvê-las ao cliente. Por exemplo, um repositório chamado R tem um repositório upstream chamado U. A tabela a seguir mostra as tags de um pacote chamado `web-helper` presente nos dois repositórios.

Repositório	Nome do pacote	Tags do pacote
R	`web-helper`	latest (alias para a versão 1.0.0)
U	`web-helper`	alpha (alias para a versão 1.0.1)

Nesse caso, quando o cliente npm busca as tags do pacote `web-helper` no repositório R, ele recebe as tags latest e alpha. As versões para as quais as tags apontam não mudarão.

Quando a mesma tag está presente no mesmo pacote no repositório upstream e downstream, o CodeArtifact usa a tag que está presente no repositório upstream. Por exemplo, suponha que as tags no webhelper tenham sido modificadas para se parecerem com as seguintes.

Repositório	Nome do pacote	Tags do pacote
R	`web-helper`	latest (alias para a versão 1.0.0)
U	`web-helper`	latest (alias para a versão 1.0.1)

Nesse caso, quando o cliente npm buscar as tags para o pacote web-helper do repositório R, a tag latest terá um alias para a versão 1.0.1, pois é isso que está no repositório upstream. Isso facilita o consumo de novas versões de pacotes em um repositório upstream que ainda não estão presentes em um repositório downstream por meio da execução de `npm update`.

Usar a tag no repositório upstream pode ser problemático ao publicar novas versões de um pacote em um repositório downstream. Por exemplo, digamos que a tag latest no pacote web-helper seja a mesma em R e U.

Repositório	Nome do pacote	Tags do pacote
R	`web-helper`	latest (alias para a versão 1.0.1)
U	`web-helper`	latest (alias para a versão 1.0.1)

Quando a versão 1.0.2 é publicada no R, o npm atualiza a tag latest para 1.0.2.

Repositório	Nome do pacote	Tags do pacote
R	`web-helper`	latest (alias para a versão 1.0.2)

Repositório	Nome do pacote	Tags do pacote
U	`web-helper`	latest (alias para a versão 1.0.1)

No entanto, o cliente npm nunca vê esse valor de tag porque o valor de latest em U é 1.0.1. A execução de `npm install` no repositório R imediatamente após a publicação da versão 1.0.2 instala a 1.0.1 em vez da versão que acabou de ser publicada. Para instalar a versão publicada mais recentemente, é necessário especificar a versão exata do pacote, da seguinte forma.

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

Suporte para gerenciadores de pacotes compatíveis com o npm

Esses outros gerenciadores de pacotes são compatíveis com o CodeArtifact e funcionam com o formato de pacote npm e o protocolo npm wire:

- gerenciador de pacotes pnpm. A versão mais recente confirmada como compatível com o CodeArtifact é a 3.3.4, que foi lançada em 18 de maio de 2019.
- Gerenciador de pacotes Yam. A versão mais recente confirmada como compatível com o CodeArtifact é a 1.21.1, que foi lançada em 11 de dezembro de 2019.

> **Note**
>
> Recomendamos usar o Yarn 2.x com o CodeArtifact. O Yarn 1.x não tem repetições de HTTP, o que significa que é mais suscetível a falhas de serviço intermitentes que resultam em códigos de status ou erros de nível 500. Não há como configurar uma estratégia de repetição diferente para o Yarn 1.x, mas isso foi adicionado no Yarn 2.x. Você pode usar o Yarn 1.x, mas talvez seja necessário adicionar repetições de nível superior nos scripts de compilação. Por exemplo, executar o comando yarn em um loop para que ele repita um dowload não concluído de pacote.

Usando o CodeArtifact com NuGet

Esses tópicos descrevem como consumir e publicar `NuGet` pacotes usando o CodeArtifact.

> **Note**
>
> AWS O CodeArtifact é compatível apenas com o NuGet 4.8 e mais recente.

Tópicos

Use o CodeArtifact com o Visual Studio

Você pode consumir pacotes do CodeArtifact diretamente no Visual Studio com o provedor de credenciais do CodeArtifact. O provedor de credenciais simplifica a configuração e a autenticação dos repositórios do CodeArtifact no Visual Studio e está disponível no AWS Toolkit for Visual Studio.

> **Note**
>
> O AWS Toolkit for Visual Studio não está disponível para o Visual Studio para Mac.

Para configurar e usar o NuGet com ferramentas de CLI, consulte Use CodeArtifact com a CLI nuget ou dotnet.

Tópicos

Configure o Visual Studio com o provedor de credenciais do CodeArtifact

O provedor de credenciais do CodeArtifact simplifica a configuração e a autenticação contínua entre o CodeArtifact e o Visual Studio. Os tokens de autenticação do CodeArtifact são válidos por no máximo 12 horas. Para evitar a necessidade de atualizar manualmente o token enquanto estiver trabalhando no Visual Studio, o provedor de credenciais busca periodicamente um novo token antes que o atual expire.

> **Important**
>
> Para usar o provedor de credenciais, verifique se todas as credenciais existentes do AWS CodeArtifact foram apagadas do seu arquivo `nuget.config`, que podem ter sido adicionadas manualmente ou ao executar o `aws codeartifact login` para configurar o NuGet anteriormente.

Use o CodeArtifact no Visual Studio com o AWS Toolkit for Visual Studio

1. Instale o AWS Toolkit for Visual Studio usando as etapas a seguir. O kit de ferramentas é compatível com o Visual Studio 2017 e 2019 usando essas etapas. AWS O CodeArtifact não é compatível com o Visual Studio 2015 e versões anteriores.

    1. O kit de ferramentas para Visual Studio para o Visual Studio 2017 e Visual Studio 2019 é distribuído no Visual Studio Marketplace. Você também pode instalar e atualizar o kit de ferramentas no Visual Studio Tools usando Ferramentas ≫ Extensões e atualizações (Visual Studio 2017) ou Extensões ≫ Gerenciar extensões (Visual Studio 2019).
    2. Após a instalação do kit de ferramentas, abra-o e selecione AWSExplorer no menu Visualizar.
2. Configure o kit de ferramentas para o Visual Studio com as suas credenciais da AWS seguindo as etapas em Fornecer credenciais da AWS no Guia do usuário do AWS Toolkit for Visual Studio.
3. (Opcional) Defina o AWS perfil que você deseja usar com o CodeArtifact. Se não for definido, o CodeArtifact usará o perfil padrão. Para definir o perfil, vá para Ferramentas > Gerenciador de pacotes do NuGet > Selecione AWS Perfil do CodeArtifact.
4. Adicione o seu repositório do CodeArtifact como uma fonte de pacote no Visual Studio.

    1. Navegue até o repositório na janela do AWS Explorer, clique com o botão direito do mouse e selecione `Copy NuGet Source Endpoint`.

2. Use o comando Ferramentas > Opções e vá até Gerenciador de pacotes do NuGet.
3. Selecione o nó Fontes de pacotes.
4. Selecione +, edite o nome do endpoint de URL do repositório copiado na etapa 3 na caixa Fonte e selecione Atualizar.
5. Marque a caixa de seleção da fonte de pacote recém-adicionada para ativá-la.

> **Note**
>
> Recomendamos adicionar uma conexão externa ao NuGet.org ao seu repositório do CodeArtifact e desabilitar a fonte do pacote nuget.org no Visual Studio. Ao usar uma conexão externa, todos os pacotes obtidos do NuGet.org serão armazenados no seu repositório do CodeArtifact. Se o NuGet.org ficar indisponível, suas dependências de aplicativos ainda estarão disponíveis para compilações de CI e desenvolvimento local. Para obter mais informações sobre conexões externas, consulte Conectar um CodeArtifact repositório a um repositório público.

5. Reinicie o Visual Studio para que as alterações entrem em vigor.

Após a configuração, o Visual Studio pode consumir pacotes do seu repositório do CodeArtifact, de qualquer um de seus repositórios upstream ou do NuGet.org se você tiver adicionado uma conexão externa. Para obter mais informações sobre como navegar e instalar pacotes NuGet no Visual Studio, consulte Instalar e gerenciar pacotes no Visual Studio usando o gerenciador de pacotes do NuGet na documentação do NuGet.

Use o console do gerenciador de pacotes do Visual Studio

O console do gerenciador de pacotes do Visual Studio não usará a versão do Visual Studio do provedor de credenciais do CodeArtifact. Para usá-lo, você precisa configurar o provedor de credenciais da linha de comando. Consulte Use CodeArtifact com a CLI nuget ou dotnet para obter mais informações.

Use CodeArtifact com a CLI nuget ou dotnet

Você pode usar ferramentas de CLI como `nuget` e `dotnet` para publicar e consumir pacotes do. CodeArtifact Este documento fornece informações sobre como configurar as ferramentas de CLI e usá-las para publicar ou consumir pacotes.

Tópicos

Configurar a CLI do nuget ou dotnet

Você pode configurar a CLI do nuget ou dotnet com o provedor de credenciais, com CodeArtifact NuGet o ou manualmente. AWS CLI A configuração NuGet com o provedor de credenciais é altamente recomendada para simplificar a configuração e a autenticação contínua.

Método 1: Configurar com o provedor de CodeArtifact NuGet credenciais

O provedor de CodeArtifact NuGet credenciais simplifica a autenticação e a configuração com ferramentas CodeArtifact CLI NuGet . CodeArtifact os tokens de autenticação são válidos por no máximo 12 horas. Para evitar a necessidade de atualizar manualmente o token ao usar a CLI do nuget ou dotnet, o provedor de credenciais busca periodicamente um novo token antes que o atual expire.

> **Important**
>
> Para usar o provedor de credenciais, certifique-se de que todas AWS CodeArtifact as credenciais existentes sejam apagadas do seu `nuget.config` arquivo que possam ter sido adicionadas manualmente ou executando `aws codeartifact login` a configuração anterior. NuGet

Instalar e configurar o provedor de CodeArtifact NuGet credenciais

dotnet

1. Baixe a versão mais recente da AWS. CodeArtifact. NuGet. CredentialProvider ferramenta do NuGet domínio.org com o seguinte `dotnet` comando.

```
dotnet tool install -g AWS.CodeArtifact.NuGet.CredentialProvider
```

2. Use o `codeartifact-creds install` comando para copiar o provedor de credenciais para a pasta de NuGet plug-ins.

```
dotnet codeartifact-creds install
```

3. (Opcional): defina o AWS perfil que você deseja usar com o provedor de credenciais. Se não for definido, o provedor de credenciais usará o perfil padrão. Para obter mais informações sobre AWS CLI perfis, consulte Perfis nomeados.

```
dotnet codeartifact-creds configure set profile profile_name
```

nuget

Execute as etapas a seguir para usar a NuGet CLI para instalar o provedor de CodeArtifact NuGet credenciais a partir de um bucket do Amazon S3 e configurá-lo. O provedor de credenciais usará o AWS CLI perfil padrão. Para obter mais informações sobre perfis, consulte Perfis nomeados.

1. Baixe a versão mais recente do provedor de CodeArtifact NuGet credenciais (codeartifact-nuget-credentialprovider.zip) de um bucket do Amazon S3.

   Para visualizar e baixar versões anteriores, consulte CodeArtifact NuGet Versões do Credential Provider.

2. Descompacte o arquivo.
3. Copie a AWS. CodeArtifact. NuGetCredentialProviderpasta da pasta netfx para `%user_profile%/.nuget/plugins/netfx/` no Windows, no Linux ou `~/.nuget/plugins/netfx` no macOS.
4. Copie a AWS. CodeArtifact. NuGetCredentialProviderpasta da pasta netcore para `%user_profile%/.nuget/plugins/netcore/` no Windows, no Linux ou `~/.nuget/plugins/netcore` no macOS.

Depois de criar um repositório e configurar o provedor de credenciais, você pode usar as ferramentas de CLI `nuget` ou `dotnet` para instalar e publicar pacotes. Para obter mais informações, consulte Consuma NuGet pacotes de CodeArtifact e Publique NuGet pacotes em CodeArtifact.

Método 2: configurar o nuget ou dotnet com o comando login

O `codeartifact login` comando no AWS CLI adiciona um endpoint do repositório e um token de autorização ao seu arquivo de NuGet configuração, permitindo que o nuget ou o dotnet se conectem ao seu repositório. CodeArtifact Isso modificará a NuGet configuração em nível de usuário que está localizada em `%appdata%\NuGet\NuGet.Config` para Windows `~/.config/NuGet/NuGet.Config` e/ou `~/.nuget/NuGet/NuGet.Config` para Mac/Linux. Para obter mais informações sobre NuGet configurações, consulte NuGet Configurações comuns.

Configurar o nuget ou dotnet com o comando **`login`**

1. Configure suas AWS credenciais para uso com o AWS CLI, conforme descrito emConceitos básicos do CodeArtifact.
2. Certifique-se de que a ferramenta NuGet CLI (`nugetoudotnet`) tenha sido instalada e configurada corretamente. Para ver instruções, consulte a documentação do nuget ou do dotnet.
3. Use o CodeArtifact `login` comando para buscar credenciais para uso com. NuGet

> **Note**
>
> Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

dotnet

> **Important**
>
> Usuários de Linux e macOS: como a criptografia não são compatíveis com plataformas que não sejam Windows, suas credenciais buscadas serão armazenadas como texto sem formatação no seu arquivo de configuração.

```
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
```

O comando de login fará o seguinte:

- Obtenha um token de autorização CodeArtifact usando suas AWS credenciais.
- Atualize sua NuGet configuração em nível de usuário com uma nova entrada para a fonte NuGet do pacote. A fonte que aponta para o endpoint do seu CodeArtifact repositório será chamada. *domain_name/repo_name*

O período de autorização padrão após chamar o `login` é de 12 horas e o `login` deve ser chamado para atualizar o token periodicamente. Para obter mais informações sobre o token de autorização criado com o comando `login`, consulte Tokens criados com o comando `login`.

Depois de criar um repositório e configurar a autenticação, você pode usar os clientes CLI `nuget`, `dotnet` ou `msbuild` para instalar e publicar pacotes. Para obter mais informações, consulte Consuma NuGet pacotes de CodeArtifact e Publique NuGet pacotes em CodeArtifact.

Método 3: configurar o nuget ou dotnet sem o comando login

Para configuração manual, você deve adicionar um endpoint do repositório e um token de autorização ao seu arquivo de NuGet configuração para permitir que o nuget ou o dotnet se conectem ao seu repositório. CodeArtifact

Configure manualmente o nuget ou o dotnet para se conectar ao seu repositório. CodeArtifact

1. Determine o endpoint do seu CodeArtifact repositório usando o `get-repository-endpoint` AWS CLI comando.

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

   Resultado do exemplo:

   ```
   {
   ```

```
   "repositoryEndpoint": "https://my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/nuget/my_repo/"
}
```

2. Obtenha um token de autorização para se conectar ao seu repositório a partir do seu gerenciador de pacotes usando o `get-authorization-token` AWS CLI comando.

```
aws codeartifact get-authorization-token --domain my_domain
```

   Resultado do exemplo:

```
{
   "authorizationToken": "eyJ2I...viOw",
   "expiration": 1601616533.0
}
```

3. Crie o URL completo do endpoint do repositório anexando `/v3/index.json` ao URL retornado por `get-repository-endpoint` na etapa 3.
4. Configure o nuget ou o dotnet para usar o endpoint do repositório da etapa 1 e o token de autorização da etapa 2.

> **Note**
>
> O URL de origem deve terminar em `/v3/index.json` para que nuget ou dotnet se conectem com sucesso a um repositório. CodeArtifact

   dotnet

   Usuários de Linux e macOS: como a criptografia não são compatíveis com plataformas que não sejam Windows, você deve adicionar o `--store-password-in-clear-text` sinalizador ao comando a seguir. Observe que isso armazenará sua senha como texto simples no arquivo de configuração.

```
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 atualizar uma fonte existente, use o 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
```

Resultado do exemplo:

```
Package source with Name: domain_name/repo_name added successfully.
```

> **Note**
>
> Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

Consuma NuGet pacotes de CodeArtifact

Depois de configurar NuGet com CodeArtifact, você pode consumir NuGet pacotes que estão armazenados em seu CodeArtifact repositório ou em um de seus repositórios upstream.

Para consumir uma versão de pacote de um CodeArtifact repositório ou de um de seus repositórios upstream com `nuget` ou`dotnet`, execute o comando a seguir substituindo *packageName* pelo nome do pacote que você deseja consumir e *packageSourceName* pelo nome da fonte do seu CodeArtifact repositório no seu arquivo de configuração. NuGet Se você usou o `login` comando para configurar sua NuGet configuração, o nome da fonte é*domain_name/repo_name*.

> **Note**
>
> Quando um pacote é solicitado, o NuGet cliente armazena em cache quais versões desse pacote existem. Devido a esse comportamento, uma instalação pode falhar para um pacote que foi solicitado anteriormente antes que a versão desejada fosse disponibilizada. Para evitar essa falha e instalar com êxito um pacote existente, você pode limpar o NuGet cache

antes de uma instalação com `nuget locals all --clear` ou`dotnet nuget locals all --clear`, ou evitar usar o cache durante `install` os `restore` comandos, fornecendo a `-NoCache` opção para `nuget` ou a `--no-cache` opção para`dotnet`.

dotnet

```
dotnet add package packageName --source packageSourceName
```

nuget

```
nuget install packageName -Source packageSourceName
```

Para instalar uma versão específica de um pacote

dotnet

```
dotnet add package packageName --version 1.0.0 --source packageSourceName
```

nuget

```
nuget install packageName -Version 1.0.0 -Source packageSourceName
```

Consulte Gerenciar pacotes usando a CLI nuget.exe ou Instalar e gerenciar pacotes usando a CLI dotnet na documentação da Microsoft para ver mais informações.

Consuma NuGet pacotes do NuGet domínio.org

Você pode consumir NuGet pacotes de NuGet.org por meio de um CodeArtifact repositório configurando o repositório com uma conexão externa com .org. NuGet Os pacotes consumidos em NuGet.org são ingeridos e armazenados no seu CodeArtifact repositório. Para ver mais informações sobre como adicionar conexões externas, consulte Conectar um CodeArtifact repositório a um repositório público.

Publique NuGet pacotes em CodeArtifact

Depois de configurar NuGet com CodeArtifact, você pode usar `nuget` ou `dotnet` publicar versões de pacotes em CodeArtifact repositórios.

Para enviar uma versão do pacote para um CodeArtifact repositório, execute o comando a seguir com o caminho completo do seu `.nupkg` arquivo e o nome da fonte do seu CodeArtifact repositório no seu arquivo de NuGet configuração. Se você usou o `login` comando para configurar sua NuGet configuração, o nome da fonte é`domain_name/repo_name`.

> **Note**
>
> Você pode criar um NuGet pacote se não tiver um para publicar. Para ver mais informações, consulte Fluxo de trabalho de criação de pacotes na documentação da Microsoft.

dotnet

```
dotnet nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg --source packageSourceName
```

nuget

```
nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg -Source packageSourceName
```

CodeArtifact NuGet Referência do provedor de credenciais

O provedor de CodeArtifact NuGet credenciais facilita a configuração e a autenticação em seus NuGet CodeArtifact repositórios.

CodeArtifact NuGet Comandos do provedor de credenciais

Esta seção inclui a lista de comandos para o provedor de CodeArtifact NuGet credenciais. Esses comandos devem ser prefixados com `dotnet codeartifact-creds` como o seguinte exemplo.

```
dotnet codeartifact-creds command
```

- `configure set profile profile`: configura o provedor de credenciais para usar o perfil fornecido AWS .
- `configure unset profile`: remove o perfil configurado, se definido.
- `install`: copia o provedor de credenciais na pasta `plugins`.
- `install --profile profile`: copia o provedor de credenciais para a `plugins` pasta e o configura para usar o perfil fornecido AWS .

- `uninstall`: desinstala o provedor de credenciais. Isso não remove as alterações no arquivo de configuração.
- `uninstall --delete-configuration`: desinstala o provedor de credenciais e remove todas as alterações no arquivo de configuração.

CodeArtifact NuGet Registros do provedor de credenciais

Para habilitar o registro para o provedor de CodeArtifact NuGet credenciais, você deve definir o arquivo de log em seu ambiente. Os logs do provedor de credenciais contêm informações úteis de depuração, como:

- O AWS perfil usado para fazer conexões
- Quaisquer erros de autenticação
- Se o endpoint fornecido não for um URL CodeArtifact

Definir o arquivo de log do provedor de CodeArtifact NuGet credenciais

```
export AWS_CODEARTIFACT_NUGET_LOGFILE=/path/to/file
```

Depois que o arquivo de log for definido, qualquer comando `codeartifact-creds` anexará sua saída de log ao conteúdo desse arquivo.

CodeArtifact NuGet Versões do Credential Provider

A tabela a seguir contém informações sobre o histórico de versões e links para download do provedor de CodeArtifact NuGet credenciais.

Versão	Alterações	Data de publicação	Link para download (S3)
1.0.2 (mais recente)	Dependências atualizadas	26/06/2024	Baixar v1.0.2
1.0.1	Foi adicionado suporte para perfis net5, net6 e SSO	03/05/2022	Fazer download da v1.0.1

Versão	Alterações	Data de publicação	Link para download (S3)
1.0.0	Lançamento inicial do CodeArtifact NuGet Credential Provider	20/11/2020	Fazer download da v1.0.0

Normalização do nome, versão e nome do ativo do pacote NuGet

O CodeArtifact normaliza os nomes dos pacotes e ativos e as versões dos pacotes antes de armazená-los, o que significa que os nomes ou versões no CodeArtifact podem ser diferentes dos fornecidos quando o pacote ou ativo foi publicado.

Normalização do nome do pacote: o CodeArtifact normaliza os nomes dos pacotes NuGet convertendo todas as letras em minúsculas.

Normalização da versão do pacote: o CodeArtifact normaliza as versões do pacote NuGet usando o mesmo padrão do NuGet. As informações a seguir são dos Números de versão normalizados da documentação do NuGet.

- Os zeros iniciais são removidos dos números de versão:
  - `1.00` é tratado como `1.0`
  - `1.01.1` é tratado como `1.1.1`
  - `1.00.0.1` é tratado como `1.0.0.1`
- Um zero na quarta parte do número da versão será omitido:
  - `1.0.0.0` é tratado como `1.0.0`
  - `1.0.01.0` é tratado como `1.0.1`
- Os metadados de compilação do SemVer 2.0.0 são removidos:
  - `1.0.7+r3456` é tratado como `1.0.7`

Normalização do nome do ativo do pacote: o CodeArtifact constrói o nome do ativo do pacote NuGet a partir do nome e da versão do pacote normalizado.

O nome e a versão do pacote não normalizado podem ser usados com solicitações de API e CLI, pois o CodeArtifact executa a normalização no nome do pacote e nas entradas da versão

dessas solicitações. Por exemplo, as entradas de `--package Newtonsoft.JSON` e `--version 12.0.03.0` seriam normalizadas e retornariam um pacote com o nome normalizado `newtonsoft.json` e a versão `12.0.3`.

Você deve usar o nome do ativo do pacote normalizado nas solicitações de API e CLI, pois o CodeArtifact não executa a normalização na entrada de `--asset`.

Você deve usar nomes e versões normalizados em ARNs.

Para localizar o nome normalizado de um pacote, use o comando `aws codeartifact list-packages`. Para obter mais informações, consulte Listar nomes de pacotes.

Para localizar o nome não normalizado de um pacote, use o comando `aws codeartifact describe-package-version`. O nome não normalizado do pacote é retornado no campo `displayName`. Para obter mais informações, consulte Exiba e atualize os detalhes e dependências da versão do pacote.

Compatibilidade do NuGet

Este guia contém informações sobre a compatibilidade do CodeArtifact com diferentes ferramentas e versões do NuGet.

Tópicos

Compatibilidade geral do NuGet

O AWS CodeArtifact é compatível com o NuGet 4.8 e mais recente.

O AWS CodeArtifact é compatível apenas com a V3 do protocolo HTTP do NuGet. Isso significa que alguns comandos da CLI que dependem da V2 do protocolo não são compatíveis. Consulte a seção Suporte ao comando nuget.exe para obter mais informações.

O AWS CodeArtifact não é compatível com o PowerShellGet 2.x.

Suporte à linha de comando do NuGet

O AWS CodeArtifact é compatível com as ferramentas de CLI NuGet (`nuget.exe`) e .NET Core (`dotnet`).

Suporte ao comando nuget.exe

Uma vez que o CodeArtifact é compatível apenas com a V3 do protocolo HTTP do NuGet, os comandos a seguir não funcionarão quando usados em recursos do CodeArtifact:

- `list`: o comando `nuget list` exibe uma lista de pacotes de uma determinada fonte. Para obter uma lista de pacotes em um repositório do CodeArtifact, você pode usar o comando Listar nomes de pacotes da CLI da AWS.

Usando CodeArtifact com Python

Esses tópicos descrevem como usar `pip` o gerenciador de pacotes Python e `twine` o utilitário de publicação de pacotes Python com. CodeArtifact

Tópicos

Configure e use o pip com CodeArtifact

pip é o instalador de pacotes para pacotes Python. Para usar o pip para instalar pacotes Python do CodeArtifact seu repositório, você deve primeiro configurar o cliente pip com as informações e credenciais do CodeArtifact seu repositório.

O pip só pode ser usado para instalar pacotes Python. Para publicar pacotes Python, você pode usar o twine. Para obter mais informações, consulte Configure e use o twine com CodeArtifact.

Configure o pip com o comando **`login`**

Primeiro, configure suas AWS credenciais para uso com o AWS CLI, conforme descrito emConceitos básicos do CodeArtifact. Em seguida, use o CodeArtifact `login` comando para buscar as credenciais e configurar `pip` com elas.

> **Note**
>
> Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

Para configurar o pip, execute o seguinte comando.

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

`login`busca um token de autorização CodeArtifact usando suas AWS credenciais. O `login` comando será CodeArtifact configurado `pip` para uso com `~/.config/pip/pip.conf` a edição para definir o `index-url` para o repositório especificado pela `--repository` opção.

O período de autorização padrão após chamar o `login` é de 12 horas e o `login` deve ser chamado para atualizar o token periodicamente. Para obter mais informações sobre o token de autorização criado com o comando `login`, consulte Tokens criados com o comando `login`.

Configurar o pip sem o comando login

Se você não puder usar o comando `login` para configurar o `pip`, você pode usar `pip config`.

1. Use o AWS CLI para obter um novo token de autorização.

   > **Note**
   >
   > Se você estiver acessando um repositório em um domínio de sua propriedade, não precisa incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

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

2. Use `pip config` para definir o URL e as credenciais do CodeArtifact registro. O comando a seguir atualizará somente o arquivo de configuração do ambiente atual. Para atualizar o arquivo de configuração de todo o sistema, substitua `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/
   ```

> **Note**
>
> Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

> **Important**
>
> O URL de registro deve terminar com uma barra inclinada (/). Do contrário, você não poderá se conectar ao repositório.

Exemplo de arquivo de configuração do pip

Veja a seguir um exemplo de `pip.conf` arquivo após definir o URL e as credenciais CodeArtifact do registro.

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

Executar o pip

Para executar `pip` comandos, você deve configurar `pip` com CodeArtifact. Para obter mais informações, consulte a documentação a seguir.

1. Siga as etapas na Configurando com AWS CodeArtifact seção para configurar sua AWS conta, ferramentas e permissões.
2. Configure o `twine` seguindo as etapas em Configure e use o twine com CodeArtifact.

Supondo que um pacote esteja presente em seu repositório ou em um de seus repositórios upstream, você pode instalá-lo com `pip install`. Por exemplo, use o seguinte comando para instalar o pacote `requests`.

```
pip install requests
```

Use a `-i` opção de reverter temporariamente a instalação de pacotes do https://pypi.org em vez do seu CodeArtifact repositório.

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

Configure e use o twine com CodeArtifact

O twine é um utilitário de publicação de pacotes para pacotes Python. Para usar o twine para publicar pacotes Python no CodeArtifact seu repositório, você deve primeiro configurar o twine com as informações e credenciais do CodeArtifact seu repositório.

O twine só pode ser usado para publicar pacotes Python. Para instalar pacotes Python, você pode usar o pip. Para obter mais informações, consulte Configure e use o pip com CodeArtifact.

Configure o twine com o comando **`login`**

Primeiro, configure suas AWS credenciais para uso com o AWS CLI, conforme descrito emConceitos básicos do CodeArtifact. Em seguida, use o CodeArtifact `login` comando para buscar as credenciais e configurar o twine com elas.

> **Note**
>
> Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

Para configurar o twine, execute o seguinte comando.

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

`login`busca um token de autorização CodeArtifact usando suas AWS credenciais. O `login` comando configura o twine para uso com CodeArtifact `~/.pypirc` a edição para adicionar o repositório especificado pela `--repository` opção com credenciais.

O período de autorização padrão após chamar o `login` é de 12 horas e o `login` deve ser chamado para atualizar o token periodicamente. Para obter mais informações sobre o token de autorização criado com o comando `login`, consulte Tokens criados com o comando `login`.

Configure o twine sem o comando `login`

Se você não puder usar o comando `login` para configurar o twine, pode usar o arquivo `~/.pypirc` ou as variáveis de ambiente. Para usar o arquivo `~/.pypirc`, adicione as seguintes entradas a ele. A senha deve ser um token de autenticação adquirido pela 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
```

> **Note**
>
> Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

Para usar variáveis de ambiente, faça o seguinte.

> **Note**
>
> Se você estiver acessando um repositório em um domínio de sua propriedade, não precisa incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

```
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`
```

Executar o twine

Antes de usar o twine para publicar ativos do pacote Python, você deve primeiro CodeArtifact configurar permissões e recursos.

1. Siga as etapas na Configurando com AWS CodeArtifact seção para configurar sua AWS conta, ferramentas e permissões.
2. Configure o twine seguindo as etapas em Configure o twine com o comando `login` ou Configure o twine sem o comando `login`.

Depois de configurar o twine, você pode executar comandos `twine`. Use o seguinte comando para publicar os ativos do pacote Python.

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

Para obter informações sobre como criar e empacotar seu aplicativo Python, consulte Gerando arquivos de distribuição no site da Python Packaging Authority.

Normalização do nome do pacote Python

CodeArtifact normaliza os nomes dos pacotes antes de armazená-los, o que significa que os nomes dos pacotes CodeArtifact podem ser diferentes do nome fornecido quando o pacote foi publicado.

Ao realizar a normalização dos pacotes Python, o nome do pacote é colocado em minúsculas e todas as instâncias dos caracteres `.`, `-` e `_` são substituídas por um único caractere `-`. Portanto, os nomes dos pacotes `pigeon_cli` e `pigeon.cli` são normalizados e armazenados como `pigeon-cli`. O nome não normalizado pode ser usado por pip e twine, mas o nome normalizado deve ser usado em solicitações de CodeArtifact CLI ou API (como) e em. `list-package-versions` ARNs Para obter mais informações sobre a normalização de nome de pacote Python, consulte PEP 503 na documentação do Python.

Compatibilidade com o Python

CodeArtifact não suporta PyPI ou. `XML-RPC` `JSON` APIs

CodeArtifact suporta PyPI `Legacy` APIs, exceto a `simple` API. Embora CodeArtifact não seja compatível com o endpoint `/simple/` da API, ele oferece suporte ao `/simple/<project>/` endpoint.

Para obter mais informações, consulte o seguinte no repositório da Python Packaging Authority. GitHub

- API XML-RPC

- API JSON
- API legada

Suporte para comandos pip

As seções a seguir resumem os comandos pip que são suportados pelos CodeArtifact repositórios, além dos comandos específicos que não são suportados.

Tópicos

- Comandos compatíveis que interagem com um repositório
- Comandos do lado do cliente compatíveis

Comandos compatíveis que interagem com um repositório

Esta seção lista os comandos `pip` em que o cliente `pip` faz uma ou mais solicitações ao registro com o qual foi configurado. Foi verificado que esses comandos funcionam corretamente quando invocados em um CodeArtifact repositório.

Command	Description
install	Instalar pacotes.
baixar	Baixar pacotes.

CodeArtifact não implementa`pip search`. Se você configurou `pip` com um CodeArtifact repositório, a execução `pip search` pesquisará e mostrará pacotes do PyPI.

Comandos do lado do cliente compatíveis

Esses comandos não exigem nenhuma interação direta com um repositório, portanto, CodeArtifact não é necessário fazer nada para suportá-los.

Command	Description
uninstall	Desinstalar pacotes.

Command	Description
freeze	Emita pacotes instalados no formato de requisitos.
list	Listar pacotes instalados.
show	Mostre informações sobre os pacotes instalados.
check	Verifique se os pacotes instalados têm dependências compatíveis.
config	Gerencie a configuração local e global.
wheel	Construa rodas conforme suas necessidades.
hash	Calcule hashes de arquivos de pacotes.
completion	Ajuda na conclusão do comando.
debug	Mostre informações úteis para depuração.
help	Mostre ajuda para comandos.

Solicitar pacotes Python de upstreams e conexões externas

Ao importar uma versão do pacote Python do pypi.org, o CodeArtifact importa todos os ativos dessa versão do pacote. Embora a maioria dos pacotes Python contenha um pequeno número de ativos, alguns contêm mais de 100, normalmente para oferecer suporte a várias arquiteturas de hardware e interpretadores de Python.

É comum que novos ativos sejam publicados em pypi.org para uma versão de pacote existente. Por exemplo, alguns projetos publicam novos ativos quando novas versões do Python são lançadas. Quando um pacote Python é instalado pelo CodeArtifact com `pip install`, as versões do pacote retidas no repositório do CodeArtifact são atualizadas para refletir o conjunto mais recente de ativos do pypi.org.

Da mesma forma, se novos ativos estiverem disponíveis para uma versão de pacote em um repositório upstream do CodeArtifact que não estejam presentes no repositório atual do CodeArtifact, eles serão mantidos no repositório atual quando `pip install` é executado.

Versões de pacotes retirados

Algumas versões do pacote em pypi.org são marcadas como retiradas, o que comunica ao instalador do pacote (como pip) que a versão não deve ser instalada, a menos que seja a única que corresponda a um especificador de versão (usando == ou ===). Para obter mais informações, consulte a PEP_592.

Se uma versão de pacote no CodeArtifact foi originalmente obtida de uma conexão externa em pypi.org, ao instalar a versão do pacote de um repositório do CodeArtifact, o CodeArtifact garante que os metadados retirados e atualizados da versão do pacote sejam obtidos em pypi.org.

Como saber se uma versão do pacote foi retirada

Para verificar se uma versão do pacote foi retirada do CodeArtifact, você pode tentar instalá-la com `pip install packageName===packageVersion`. Se a versão do pacote for retirada, será exibida uma mensagem de aviso semelhante à seguinte:

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

Para verificar se uma versão do pacote foi retirada em pypi.org, você pode visitar a lista do pypi.org para a versão do pacote em `https://pypi.org/project/packageName/packageVersion/`.

Definir o status de retirado em pacotes privados

O CodeArtifact não é compatível com a configuração de metadados retirados para pacotes publicados diretamente nos repositórios do CodeArtifact.

Por que o CodeArtifact não está buscando os metadados ou ativos retirados mais recentes para uma versão do pacote?

Normalmente, o CodeArtifact garante que, quando uma versão do pacote Python é obtida de um repositório do CodeArtifact, os metadados retirados estejam atualizados com o valor mais recente em pypi.org. Além disso, a lista de ativos na versão do pacote também é mantida atualizada com o conjunto mais recente em pypi.org e em qualquer repositório upstream do CodeArtifact. Isso vale se você estiver instalando a versão do pacote pela primeira vez e o CodeArtifact importá-la do pypi.org

para o seu repositório do CodeArtifact ou se você tiver instalado o pacote antes. No entanto, há casos em que o cliente do gerenciador de pacotes, como pip, não extrai os últimos metadados retirados de pypi.org ou repositórios upstream. Em vez disso, o CodeArtifact retornará os dados que já estão armazenados no repositório. Esta seção descreve as três maneiras pelas quais isso pode ocorrer:

Configuração upstream: se a conexão externa com pypi.org for removida do repositório ou seus upstreams usando disassociate-external-connection, os metadados retirados não serão mais atualizados do pypi.org. Da mesma forma, se você remover um repositório upstream, os ativos do repositório removido e dos upstreams do repositório removido não estarão mais disponíveis para o repositório atual. O mesmo acontece se você usar os controles de origem do pacote CodeArtifact para evitar que novas versões de um pacote específico sejam retiradas; a configuração de `upstream=BLOCK` impedirá que os metadados retirados sejam atualizados.

Status da versão do pacote: se você definir o status de uma versão do pacote para qualquer coisa exceto `Published` ou `Unlisted`, os metadados e ativos retirados da versão do pacote não serão atualizados. Da mesma forma, se você estiver buscando uma versão específica do pacote (digamos `torch 2.0.1`) e a mesma versão do pacote estiver presente em um repositório de upstream com um status que não é `Published` ou `Unlisted`, isso também bloqueará a propagação de metadados e ativos retirados do repositório upstream para o repositório atual. Isso ocorre porque outros status de versão do pacote são uma indicação de que as versões não devem mais ser consumidas em nenhum repositório.

Publicação direta: se você publicar uma versão específica do pacote diretamente em um repositório do CodeArtifact, isso evitará a atualização de metadados e ativos retirados para a versão do pacote de seus repositórios upstream e pypi.org. Por exemplo, digamos que você baixe um ativo da versão do pacote `torch 2.0.1`, como `torch-2.0.1-cp311-none-macosx_11_0_arm64.whl`, usando um navegador da web e, em seguida, faz a publicação no seu repositório do CodeArtifact usando twine como `torch 2.0.1`. O CodeArtifact rastreia se a versão do pacote entrou no domínio por meio da publicação direta no repositório, não de uma conexão externa com pypi.org ou um repositório upstream. Nesse caso, o CodeArtifact não mantém os metadados retirados sincronizados com repositórios upstream ou pypi.org. O mesmo acontece se você publicar o `torch 2.0.1` em um repositório upstream: a presença da versão do pacote bloqueará a propagação de metadados e ativos retirados para repositórios mais abaixo no gráfico de upstream.

Usando CodeArtifact com Ruby

Esses tópicos descrevem como usar as ferramentas RubyGems e o Bundler CodeArtifact para instalar e publicar gems Ruby.

> **Note**
>
> CodeArtifact recomenda o Ruby 3.3 ou posterior e não funciona com o Ruby 2.6 ou mais antigo.

Tópicos

Configure RubyGems e use o Bundler com CodeArtifact

Depois de criar um repositório no CodeArtifact, você pode usar RubyGems (`gem`) e Bundler (`bundle`) para instalar e publicar gems. Este tópico descreve como configurar os gerenciadores de pacotes para se autenticar e usar um CodeArtifact repositório.

Configure RubyGems (**`gem`**) e Bundler (**`bundle`**) com CodeArtifact

Para usar RubyGems (`gem`) ou Bundler (`bundle`) para publicar gems ou consumir gems AWS CodeArtifact, primeiro você precisará configurá-los com as informações do seu CodeArtifact repositório, incluindo credenciais para acessá-las. Siga as etapas em um dos procedimentos a seguir para configurar as ferramentas `gem` e a `bundle` CLI com as informações e credenciais do endpoint CodeArtifact do repositório.

Configurar RubyGems e agrupar usando as instruções do console

Você pode usar as instruções de configuração no console para conectar seus gerenciadores de pacotes Ruby ao seu CodeArtifact repositório. As instruções do console fornecem comandos personalizados que você pode executar para configurar seus gerenciadores de pacotes sem precisar encontrar e preencher suas CodeArtifact informações.

1. Abra o AWS CodeArtifact console em https://console.aws.amazon.com/codesuite/codeartifact/home.
2. No painel de navegação, selecione Repositórios e escolha o repositório que deseja usar para instalar ou enviar gems Ruby.
3. Clique em Visualizar instruções de conexão.
4. Selecione o seu sistema operacional.
5. Escolha o cliente do gerenciador de pacotes Ruby que você deseja configurar com seu CodeArtifact repositório.
6. Siga as instruções geradas para configurar o cliente gerenciador de pacotes para instalar gems Ruby ou publicar gems Ruby no repositório.

Configurar RubyGems e agrupar manualmente

Se você não puder ou não quiser usar as instruções de configuração do console, poderá usar as instruções a seguir para se conectar manualmente aos gerenciadores de pacotes Ruby ao seu CodeArtifact repositório.

1. Em uma linha de comando, use o comando a seguir para buscar um token de CodeArtifact autorização e armazená-lo em uma variável de ambiente.

   - *my_domain*Substitua pelo seu nome de CodeArtifact domínio.
   - *111122223333*Substitua pelo ID da AWS conta do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

   macOS and Linux

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

   Windows

   - Windows (usando o shell de comando padrão):

```
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
```

- Janelas PowerShell:

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

2. Para publicar gems Ruby em seu repositório, use o comando a seguir para buscar o endpoint do seu CodeArtifact repositório e armazená-lo na variável de ambiente. `RUBYGEMS_HOST` A CLI `gem` usa essa variável de ambiente para definir onde os gems são publicados.

> **Note**
>
> Como alternativa, em vez de usar a variável de ambiente `RUBYGEMS_HOST`, você pode fornecer o endpoint do repositório com a opção `--host` ao usar o comando `gem push`.

- *my_domain*Substitua pelo seu nome de CodeArtifact domínio.
- *111122223333*Substitua pelo ID da AWS conta do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
- *my_repo*Substitua pelo nome CodeArtifact do seu repositório.

macOS and Linux

```
export RUBYGEMS_HOST=`aws codeartifact get-repository-endpoint --
domain my_domain --domain-owner 111122223333 --repository my_repo --format ruby
 --query repositoryEndpoint --output text | sed 's:/*$::'`
```

Windows

Os comandos a seguir recuperam o endpoint do repositório, removem o `/` à direita e o armazenam em uma variável de ambiente.

- Windows (usando o shell de comando padrão):

```
for /f %i in ('aws codeartifact get-repository-endpoint --domain my_domain
 --domain-owner 111122223333 --repository my_repo --format ruby --query
 repositoryEndpoint --output text') do set RUBYGEMS_HOST=%i

set RUBYGEMS_HOST=%RUBYGEMS_HOST:~0,-1%
```

- Janelas PowerShell:

```
$env:RUBYGEMS_HOST = (aws codeartifact get-repository-endpoint --
domain my_domain --domain-owner 111122223333 --repository my_repo --format
 ruby --query repositoryEndpoint --output text).TrimEnd("/")
```

O seguinte URL é um exemplo de endpoint de repositório:

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

> **Note**
>
> Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

3. Para publicar gems Ruby em seu repositório, você deve se autenticar CodeArtifact com RubyGems editando seu `~/.gem/credentials` arquivo para incluir seu token de autenticação. Crie um diretório `~/.gem/` e um arquivo `~/.gem/credentials` se o diretório ou arquivo não existir.

   macOS and Linux

   ```
   echo ":codeartifact_api_key: Bearer $CODEARTIFACT_AUTH_TOKEN" >> ~/.gem/
   credentials
   ```

   Windows

   - Windows (usando o shell de comando padrão):

     ```
     echo :codeartifact_api_key: Bearer %CODEARTIFACT_AUTH_TOKEN% >> %USERPROFILE
     %/.gem/credentials
     ```

- Janelas PowerShell:

```
echo ":codeartifact_api_key: Bearer $env:CODEARTIFACT_AUTH_TOKEN" | Add-
Content ~/.gem/credentials
```

4. Para usar `gem` para instalar gems Ruby por meio do seu repositório, é necessário adicionar as informações do endpoint do repositório e o token de autenticação ao seu arquivo `.gemrc`. Você pode adicioná-las ao arquivo global (`~/.gemrc`) ou ao arquivo `.gemrc` do seu projeto. As CodeArtifact informações que você deve adicionar ao `.gemrc` são uma combinação do endpoint do repositório e do token de autenticação. O formato é o seguinte:

```
https://aws:${CODEARTIFACT_AUTH_TOKEN}@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/ruby/my_repo/
```

- Você pode usar a variável de ambiente `CODEARTIFACT_AUTH_TOKEN` definida em uma etapa anterior para o token de autenticação.
- Para buscar o endpoint do repositório, você poderá ler o valor da variável de ambiente `RUBYGEMS_HOST` definida anteriormente ou usar o seguinte comando `get-repository-endpoint`, substituindo os valores conforme necessário:

```
aws codeartifact get-repository-endpoint --domain my_domain --domain-
owner 111122223333 --repository my_repo --format ruby --query repositoryEndpoint
 --output text
```

Após obter o endpoint, use um editor de texto para adicionar `aws:${CODEARTIFACT_AUTH_TOKEN}@` na posição apropriada. Após criar o endpoint do repositório e a string do token de autenticação, adicione-os à seção `:sources:` do seu arquivo `.gemrc` com o comando `echo` da seguinte forma:

> **Warning**
>
> CodeArtifact não suporta a adição de repositórios como fontes usando o `gem sources -add` comando. Você deve adicionar a fonte diretamente ao arquivo.

macOS and Linux

```
echo ":sources:
    - https://aws:
${CODEARTIFACT_AUTH_TOKEN}@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/ruby/my_repo/" > ~/.gemrc
```

Windows

- Windows (usando o shell de comando padrão):

```
echo ":sources:
    - https://aws:%CODEARTIFACT_AUTH_TOKEN
%@my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com/ruby/my_repo/"
 > "%USERPROFILE%\.gemrc"
```

- Janelas PowerShell:

```
echo ":sources:
    - https://aws:
$env:CODEARTIFACT_AUTH_TOKEN@my_domain-111122223333.d.codeartifact.us-
west-2.amazonaws.com/ruby/my_repo/" | Add-Content ~/.gemrc
```

5. Você deve configurar o Bundler com o URL do endpoint do repositório e o token de autenticação executando o seguinte comando `bundle config` ao usar o Bundler:

macOS and Linux

```
bundle config $RUBYGEMS_HOST aws:$CODEARTIFACT_AUTH_TOKEN
```

Windows

- Windows (usando o shell de comando padrão):

```
bundle config %RUBYGEMS_HOST% aws:%CODEARTIFACT_AUTH_TOKEN%
```

- Janelas PowerShell:

```
bundle config $Env:RUBYGEMS_HOST aws:$Env:CODEARTIFACT_AUTH_TOKEN
```

Agora que você configurou RubyGems (`gem`) e Bundler (`bundle`) com seu CodeArtifact repositório, você pode usá-los para publicar e consumir gems Ruby de e para ele.

Instalando gemas Ruby a partir de CodeArtifact

Use os procedimentos a seguir para instalar gems Ruby a partir de um CodeArtifact repositório com as ferramentas ou `gem` CLI`bundle`.

Instalar gems Ruby com **`gem`**

Você pode usar a CLI RubyGems (`gem`) para instalar rapidamente uma versão específica de uma gem do Ruby a partir do seu repositório. CodeArtifact

Para instalar gems Ruby a partir de um repositório com CodeArtifact **`gem`**

1. Caso contrário, siga as etapas Configure RubyGems (`gem`) e Bundler (`bundle`) com CodeArtifact para configurar a `gem` CLI para usar seu CodeArtifact repositório com as credenciais adequadas.

   > **Note**
   >
   > O token de autorização gerado é válido por 12 horas. Você precisará criar um novo se tiverem passado 12 horas desde a criação do token.

2. Use o comando a seguir para instalar gems Ruby a partir de: CodeArtifact

   ```
   gem install my_ruby_gem --version 1.0.0
   ```

Instalar gems Ruby com **`bundle`**

Você pode usar a CLI do Bundler (`bundle`) para instalar os gems Ruby configurados em `Gemfile`.

Para instalar gems Ruby a partir de um repositório com CodeArtifact **`bundle`**

1. Caso contrário, siga as etapas Configure RubyGems (`gem`) e Bundler (`bundle`) com CodeArtifact para configurar a `bundle` CLI para usar seu CodeArtifact repositório com as credenciais adequadas.

> **Note**
>
> O token de autorização gerado é válido por 12 horas. Você precisará criar um novo se tiverem passado 12 horas desde a criação do token.

2. Adicione a URL CodeArtifact do endpoint do repositório ao seu `Gemfile` as `source` para instalar gems Ruby configuradas do seu CodeArtifact repositório e de seus upstreams.

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

gem 'my_ruby_gem'
```

3. Use o seguinte comando para instalar os gems Ruby conforme especificado em `Gemfile`:

```
bundle install
```

Publicando gemas Ruby em CodeArtifact

Use o procedimento a seguir para publicar gems Ruby em um CodeArtifact repositório usando a CLI. `gem`

1. Caso contrário, siga as etapas Configure RubyGems (`gem`) e Bundler (`bundle`) com CodeArtifact para configurar a `gem` CLI para usar seu CodeArtifact repositório com as credenciais adequadas.

> **Note**
>
> O token de autorização gerado é válido por 12 horas. Você precisará criar um novo se tiverem passado 12 horas desde a criação do token.

2. Use o comando a seguir para publicar gems Ruby em um CodeArtifact repositório. Observe que, se você não definiu a variável de `RUBYGEMS_HOST` ambiente, deverá fornecer o endpoint CodeArtifact do repositório na `--host` opção.

```
gem push --key codeartifact_api_key my_ruby_gem-0.0.1.gem
```

RubyGems suporte de comando

CodeArtifact suporta `gem install` os `gem push` comandos e.

> **Note**
>
> O `gem install` comando requer a RubyGems versão 3.5.18 ou anterior quando usado com o. CodeArtifact RubyGems as versões 3.5.19 e posteriores exigem a API Compact Index para resolver gems, que atualmente CodeArtifact não é compatível. Se você estiver usando a RubyGems versão 3.5.19 ou posterior, use o Bundler para instalar gems em vez disso, ou faça o downgrade executando. RubyGems `gem update --system 3.5.18`

CodeArtifact não suporta os seguintes `gem` comandos:

- `gem fetch`
- `gem info --remote`
- `gem list --remote`
- `gem mirror`
- `gem outdated`
- `gem owner`
- `gem query`
- `gem search`
- `gem signin`
- `gem signout`
- `gem sources --add`
- `gem sources --update`
- `gem specification --remote`
- `gem update`
- `gem yank`

Compatibilidade do Bundler

Este guia contém informações sobre a compatibilidade com CodeArtifact o Bundler.

Compatibilidade do Bundler

AWS CodeArtifact recomenda o Bundler 2.4.11 ou superior. Se você houver problemas com a instalação, atualize a CLI do Bundler para a versão mais recente.

Suporte à versão do Bundler

Nas versões do Bundler anteriores à 2.4.11, há um limite de 500 dependências que podem ser definidas no Gemfile antes que o Bundler decida consultar o índice completo, `specs.4.8.gz`. Como CodeArtifact não oferece suporte ao índice completo, especificar mais de 500 dependências não funcionará CodeArtifact ao usar versões do Bundler inferiores à 2.4.11.

Para definir mais de 500 dependências em seu Gemfile com CodeArtifact, atualize o Bundler para a versão 2.4.11 ou superior.

Suporte às operações do Bundler

CodeArtifactO suporte para RubyGems não inclui o Bundler Compact Index APIs (a `/versions` API não é suportada). CodeArtifact só é compatível com a API de dependências.

Como o Compact Index não é compatível, o Bundler resolve gems usando a API Dependencies (`/api/v1/dependencies`), que envia vários nomes de gem em uma única solicitação. Cada nome de gem na solicitação conta como uma solicitação separada em relação à cota de solicitações de leitura por segundo da sua conta. Por exemplo, se o Bundler enviar uma solicitação de dependência contendo 20 nomes de gem, ela contará como 20 solicitações em relação à cota. Isso pode causar limitação em CI/CD ambientes com alta simultaneidade, mesmo quando a contagem de solicitações HTTP parece estar bem abaixo do limite configurado. Se você enfrentar limitação durante a resolução de gemas em Ruby, solicite um aumento de cota para solicitações de leitura por segundo de uma única conta. AWS Para obter mais informações, consulte Cotas em AWS CodeArtifact.

Além disso, CodeArtifact não suporta as várias especificações APIs, como`specs.4.8.gz`.

Usando CodeArtifact com Swift

Esses tópicos descrevem como usar o Swift Package Manager CodeArtifact para instalar e publicar pacotes Swift.

> **Note**
>
> CodeArtifact suporta Swift 5.8 e posterior e Xcode 14.3 e posterior.
> CodeArtifact recomenda o Swift 5.9 e posterior e o Xcode 15 e posterior.

Tópicos

Configure o Swift Package Manager com CodeArtifact

Para usar o Swift Package Manager para publicar pacotes ou consumir pacotes AWS CodeArtifact, primeiro você precisará configurar as credenciais para acessar seu CodeArtifact repositório. O método recomendado para configurar a CLI do Swift Package Manager com CodeArtifact suas credenciais e o endpoint do repositório é usando o comando. `aws codeartifact login` Você também pode configurar o Swift Package Manager manualmente.

Configurar o Swift o com o comando login

Use o `aws codeartifact login` comando para configurar o Swift Package Manager com CodeArtifact.

> **Note**
>
> Para usar o comando login, é necessário utilizar o Swift 5.8 ou posterior e o Swift 5.9 ou posterior é recomendado.

O comando `aws codeartifact login` fará o que segue:

1. Obtenha um token de autenticação CodeArtifact e armazene-o em seu ambiente. O modo como as credenciais são armazenadas depende do sistema operacional do ambiente:

   a. macOS: uma entrada é criada no aplicativo macOS Chavechain.

   b. Linux e Windows: uma entrada é criada no arquivo `~/.netrc`.

   Em todos os sistemas operacionais, se existir uma entrada de credenciais, o comando substituirá a entrada por um token.

2. Busque a URL do endpoint CodeArtifact do repositório e adicione-a ao seu arquivo de configuração do Swift. O comando adiciona o URL do endpoint do repositório ao arquivo de configuração no nível do projeto localizado em `/path/to/project/.swiftpm/configuration/registries.json`.

> **Note**
>
> O comando `aws codeartifact login` aciona os comandos `swift package-registry` que devem ser executados a partir do diretório que contém o arquivo `Package.swift`. Por isso, o comando `aws codeartifact login` deve ser executado dentro do projeto Swift.

Para configurar o Swift o com o comando login

1. Navegue até o diretório do projeto Swift que contém o arquivo `Package.swift` do projeto.
2. Execute o seguinte comando `aws codeartifact login`.

   Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

```
aws codeartifact login --tool swift --domain my_domain \
--domain-owner 111122223333 --repository my_repo \
[--namespace my_namespace]
```

A `--namespace` opção configura o aplicativo para consumir pacotes do seu CodeArtifact repositório somente se eles estiverem no namespace designado. CodeArtifact namespaces são sinônimos de escopos e são usados para organizar o código em grupos lógicos e evitar colisões de nomes que podem ocorrer quando sua base de código inclui várias bibliotecas.

O período de autorização padrão após chamar o `login` é de 12 horas e o `login` deve ser chamado para atualizar o token periodicamente. Para obter mais informações sobre o token de autorização criado com o comando `login`, consulte Tokens criados com o comando `login`.

Configurar o Swift sem o comando login

Embora seja recomendável configurar o Swift com o comando `aws codeartifact login`, você também pode configurar o Swift Package Manager sem o comando login atualizando manualmente a configuração do Swift Package Manager.

No procedimento a seguir, você usará o AWS CLI para fazer o seguinte:

1. Obtenha um token de autenticação CodeArtifact e armazene-o em seu ambiente. O modo como as credenciais são armazenadas depende do sistema operacional do ambiente:
   a. macOS: uma entrada é criada no aplicativo macOS Chavechain.
   b. Linux e Windows: uma entrada é criada no arquivo `~/.netrc`.
2. Busque o URL do endpoint CodeArtifact do seu repositório.
3. No arquivo de configuração `~/.swiftpm/configuration/registries.json`, adicione uma entrada com o URL do endpoint do repositório e o tipo de autenticação.

Para configurar o Swift sem o comando login

1. Em uma linha de comando, use o comando a seguir para buscar um token de CodeArtifact autorização e armazená-lo em uma variável de ambiente.
   - *`my_domain`*Substitua pelo seu nome de CodeArtifact domínio.
   - *`111122223333`*Substitua pelo ID da AWS conta do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

macOS and Linux

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

Windows

- Windows (usando o shell de comando padrão):

```
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
```

- Janelas PowerShell:

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

2. Obtenha o endpoint do seu CodeArtifact repositório executando o comando a seguir. O endpoint do repositório é usado para direcionar o Swift Package Manager ao seu repositório para consumir ou publicar pacotes.

   - *my_domain*Substitua pelo seu nome de CodeArtifact domínio.
   - *111122223333*Substitua pelo ID da AWS conta do proprietário do domínio. Se você estiver acessando um repositório em um domínio de sua propriedade, não será necessário incluir `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.
   - *my_repo*Substitua pelo nome CodeArtifact do seu repositório.

   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 (usando o shell de comando padrão):

```
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
```

- Janelas PowerShell:

```
$env:CODEARTIFACT_REPO = aws codeartifact get-repository-endpoint --
domain my_domain --domain-owner 111122223333 --repository my_repo --format
 swift --query repositoryEndpoint --output text
```

O URL a seguir é um exemplo de endpoint de repositório.

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

> **Note**
>
> Para usar um endpoint de pilha dupla, use o endpoint `codeartifact.region.on.aws`.

> **Important**
>
> Você deve incluir `login` ao final do endpoint de URL do repositório quando usado para configurar o Swift Package Manager. Isso é feito para você nos comandos deste procedimento.

3. Com esses dois valores armazenados em variáveis de ambiente, envie-os ao Swift usando o comando `swift package-registry login` da seguinte forma:

macOS and Linux

```
swift package-registry login ${CODEARTIFACT_REPO}login --token
 ${CODEARTIFACT_AUTH_TOKEN}
```

Windows

- Windows (usando o shell de comando padrão):

```
swift package-registry login %CODEARTIFACT_REPO%login --token
 %CODEARTIFACT_AUTH_TOKEN%
```

- Janelas PowerShell:

```
swift package-registry login $Env:CODEARTIFACT_REPO+"login" --token
 $Env:CODEARTIFACT_AUTH_TOKEN
```

4. Em seguida, atualize o registro do pacote usado pelo seu aplicativo para que qualquer dependência seja retirada do seu CodeArtifact repositório. Esse comando deve ser executado no diretório do projeto em que você está tentando resolver a dependência do pacote:

   macOS and Linux

   ```
   $ swift package-registry set ${CODEARTIFACT_REPO} [--scope my_scope]
   ```

   Windows

   - Windows (usando o shell de comando padrão):

   ```
   $ swift package-registry set %CODEARTIFACT_REPO% [--scope my_scope]
   ```

   - Janelas PowerShell:

   ```
   $ swift package-registry set $Env:CODEARTIFACT_REPO [--scope my_scope]
   ```

   A `--scope` opção configura o aplicativo para consumir pacotes do seu CodeArtifact repositório somente se eles estiverem no escopo designado. Os escopos são sinônimos de [CodeArtifact]

namespaces e são usados para organizar o código em grupos lógicos e evitar colisões de nomes que podem ocorrer quando sua base de código inclui várias bibliotecas.

5. Você pode confirmar se a configuração foi realizada corretamente ao acessar o conteúdo do arquivo `.swiftpm/configuration/registries.json` no nível do projeto executando o seguinte comando no diretório do projeto:

```
$ 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
}
```

Agora que você configurou o Swift Package Manager com seu CodeArtifact repositório, você pode usá-lo para publicar e consumir pacotes Swift de e para ele. Para obter mais informações, consulte Consumir e publicar pacotes Swift.

Consumir e publicar pacotes Swift

Consumindo pacotes Swift de CodeArtifact

Use o procedimento a seguir para consumir pacotes Swift de um AWS CodeArtifact repositório.

Para consumir pacotes Swift de um CodeArtifact repositório

1. Caso contrário, siga as etapas Configure o Swift Package Manager com CodeArtifact para configurar o Swift Package Manager para usar seu CodeArtifact repositório com as credenciais adequadas.

> **Note**
>
> O token de autorização gerado é válido por 12 horas. Você precisará criar um novo se tiverem passado 12 horas desde a criação do token.

2. Edite o arquivo `Package.swift` na pasta do projeto do aplicativo para atualizar as dependências do pacote a serem usadas no projeto.

   a. Se o arquivo `Package.swift` não contiver a seção `dependencies`, adicione uma.

   b. Na seção `dependencies` do arquivo `Package.swift`, adicione o pacote que você deseja usar adicionando seu identificador de pacote. O identificador do pacote consiste no escopo e no nome do pacote separados por ponto (.). Consulte o trecho de código em uma etapa posterior para ver um exemplo.

   > **Tip**
   >
   > Para encontrar o identificador do pacote, você pode usar o CodeArtifact console. Localize a versão específica do pacote que você deseja usar e consulte as instruções de Atalho de instalação na página de versão do pacote.

   c. Se o arquivo `Package.swift` não contiver a seção `targets`, adicione uma.

   d. Na seção `targets`, adicione os destinos que precisarão usar a dependência.

   A seguir há um trecho de exemplo que mostra a configuração das seções `dependencies` e `targets` em um arquivo `Package.swift`:

```
...
    ],
    dependencies: [
        .package(id: "my_scope.package_name", from: "1.0.0")
    ],
    targets: [
      .target(
         name: "MyApp",
         dependencies: ["package_name"]
      ),...
    ],
...
```

3. Agora que tudo está configurado, use o comando a seguir para baixar as dependências do CodeArtifact pacote.

```
swift package resolve
```

Consumindo pacotes Swift CodeArtifact do Xcode

Use o procedimento a seguir para consumir pacotes Swift de um CodeArtifact repositório no Xcode.

Para consumir pacotes Swift de um CodeArtifact repositório no Xcode

1. Caso contrário, siga as etapas Configure o Swift Package Manager com CodeArtifact para configurar o Swift Package Manager para usar seu CodeArtifact repositório com as credenciais adequadas.

> **Note**
>
> O token de autorização gerado é válido por 12 horas. Você precisará criar um novo se tiverem passado 12 horas desde a criação do token.

2. Adicione os pacotes como uma dependência do projeto no Xcode.
   a. Escolha Arquivo > Adicionar pacotes.
   b. Localize o pacote usando a barra de pesquisa. Sua pesquisa deve estar no formato `package_scope.package_name`.
   c. Quando encontrado, escolha o pacote e clique em Adicionar pacote.
   d. Depois que o pacote for verificado, escolha os produtos do pacote que você deseja adicionar como dependência e escolha Adicionar pacote.

Se você tiver problemas ao usar seu CodeArtifact repositório com o Xcode, consulte Solução de problemas do Swift os problemas comuns e as possíveis correções.

Publicando pacotes Swift em CodeArtifact

CodeArtifact recomenda o Swift 5.9 ou posterior e usa o `swift package-registry publish` comando para publicar pacotes Swift. Se você estiver usando uma versão anterior, deverá usar um comando Curl para publicar pacotes do Swift. CodeArtifact

Publicando CodeArtifact pacotes com o `swift package-registry publish` comando

Use o procedimento a seguir com o Swift 5.9 ou posterior para publicar pacotes Swift em um CodeArtifact repositório com o Swift Package Manager.

1. Caso contrário, siga as etapas Configure o Swift Package Manager com CodeArtifact para configurar o Swift Package Manager para usar seu CodeArtifact repositório com as credenciais adequadas.

   > **Note**
   >
   > O token de autorização gerado é válido por 12 horas. Você precisará criar um novo se tiverem passado 12 horas desde a criação do token.

2. Navegue até o diretório do projeto Swift que contém o arquivo `Package.swift` para seu pacote.
3. Para publicar o pacote, execute o seguinte comando `swift package-registry publish`. O comando cria um arquivo de origem do pacote e o publica no seu CodeArtifact repositório.

   ```
   swift package-registry publish packageScope.packageName packageVersion
   ```

   Por exemplo:

   ```
   swift package-registry publish myScope.myPackage 1.0.0
   ```

4. Você pode confirmar se o pacote foi publicado e existe no repositório acessando o console ou usando o comando `aws codeartifact list-packages` da seguinte forma:

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

   Você pode exibir a versão única do pacote usando o comando `aws codeartifact list-package-versions` da seguinte forma:

   ```
   aws codeartifact list-package-versions --domain my_domain --repository my_repo \
   --format swift --namespace my_scope --package package_name
   ```

Publicando CodeArtifact pacotes com Curl

Embora seja recomendável usar o `swift package-registry publish` comando que vem com o Swift 5.9 ou posterior, você também pode usar o Curl para publicar pacotes do Swift no. CodeArtifact

Use o procedimento a seguir para publicar pacotes Swift em um AWS CodeArtifact repositório com Curl.

1. Caso ainda não tenha feito isso, crie e atualize as variáveis de ambiente `CODEARTIFACT_AUTH_TOKEN` e `CODEARTIFACT_REPO` seguindo as etapas em Configure o Swift Package Manager com CodeArtifact.

   > **Note**
   >
   > O token de autorização é válido por 12 horas. Você precisará atualizar a variável de ambiente `CODEARTIFACT_AUTH_TOKEN` com novas credenciais se tiverem passado 12 horas desde que ela foi criada.

2. Primeiramente, se você não tiver um pacote Swift criado, isso pode ser feito executando os seguintes comandos:

   ```
   mkdir testDir && cd testDir
   swift package init
   git init .
   swift package archive-source
   ```

3. Use o seguinte comando Curl para publicar seu pacote Swift em 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. Você pode confirmar se o pacote foi publicado e existe no repositório acessando o console ou usando o comando `aws codeartifact list-packages` da seguinte forma:

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

Você pode exibir a versão única do pacote usando o comando `aws codeartifact list-package-versions` da seguinte forma:

```
aws codeartifact list-package-versions --domain my_domain --repository my_repo \
--format swift --namespace my_scope --package package_name
```

Buscando pacotes Swift GitHub e republicando em CodeArtifact

Use o procedimento a seguir para buscar um pacote Swift GitHub e republicá-lo em um CodeArtifact repositório.

Para obter um pacote Swift GitHub e republicá-lo no CodeArtifact

1. Caso contrário, siga as etapas Configure o Swift Package Manager com CodeArtifact para configurar o Swift Package Manager para usar seu CodeArtifact repositório com as credenciais adequadas.

> **Note**
>
> O token de autorização gerado é válido por 12 horas. Você precisará criar um novo se tiverem passado 12 horas desde a criação do token.

2. Clone o repositório Git do pacote Swift que você deseja buscar e republicar usando o seguinte comando `git clone`. Para obter informações sobre a clonagem de GitHub repositórios, consulte Clonar um repositório na Documentação. GitHub

```
git clone repoURL
```

3. Navegue até o repositório que você acabou de clonar:

```
cd repoName
```

4. Crie o pacote e publique-o no CodeArtifact.

   a. Recomendado: se você estiver usando o Swift 5.9 ou posterior, poderá usar o `swift package-registry publish` comando a seguir para criar o pacote e publicá-lo no seu CodeArtifact repositório configurado.

   ```
   swift package-registry publish packageScope.packageName versionNumber
   ```

   Por exemplo:

   ```
   swift package-registry publish myScope.myPackage 1.0.0
   ```

   b. Se você estiver usando uma versão do Swift anterior à 5.9, deverá usar o comando `swift archive-source` para criar o pacote e, em seguida, usar um comando Curl para publicá-lo.

      i. Se as variáveis de ambiente `CODEARTIFACT_AUTH_TOKEN` e `CODEARTIFACT_REPO` não tiverem sido configuradas ou caso já tenham se passado mais de 12 horas desde que você realizou essa configuração, siga as etapas em Configurar o Swift sem o comando login.

      ii. Crie o pacote Swift usando o comando `swift package archive-source`:

      ```
      swift package archive-source
      ```

      Se for bem-sucedido, você verá `Created package_name.zip` na linha de comando.

      iii. Use o seguinte comando Curl para publicar o pacote Swift em 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. Você pode confirmar se o pacote foi publicado e existe no repositório acessando o console ou usando o comando `aws codeartifact list-packages` da seguinte forma:

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

   Você pode exibir a versão única do pacote usando o comando `aws codeartifact list-package-versions` da seguinte forma:

   ```
   aws codeartifact list-package-versions --domain my_domain --repository my_repo \
   --format swift --namespace my_scope --package package_name
   ```

Normalização do nome e do namespace do pacote Swift

CodeArtifact normaliza os nomes e namespaces dos pacotes antes de armazená-los, o que significa que os nomes CodeArtifact podem ser diferentes dos fornecidos quando o pacote foi publicado.

Normalização do nome do pacote e do namespace: CodeArtifact normaliza os nomes e namespaces dos pacotes Swift convertendo todas as letras em minúsculas.

Normalização da versão do pacote: CodeArtifact não normaliza as versões do pacote Swift. Observe que CodeArtifact só oferece suporte aos padrões de versão 2.0 do Semantic Versioning. Para obter mais informações sobre o Versionamento Semântico, consulte Versionamento Semântico 2.0.0.

O nome e o namespace do pacote não normalizados podem ser usados com solicitações de API e CLI porque CodeArtifact executam a normalização nas entradas dessas solicitações. Por exemplo, as entradas `--package myPackage` e `--namespace myScope` seriam normalizadas e retornariam um pacote normalizado com o nome `mypackage` e namespace `myscope`.

Você deve usar nomes normalizados em ARNs, como nas políticas do IAM.

Para localizar o nome normalizado de um pacote, use o comando `aws codeartifact list-packages`. Para obter mais informações, consulte Listar nomes de pacotes.

Solução de problemas do Swift

As informações a seguir podem ajudá-lo a solucionar problemas comuns com o Swift e. CodeArtifact

O erro 401 é exibido no Xcode mesmo depois de configurar o Swift Package Manager

Problema: Quando você está tentando adicionar um pacote do seu CodeArtifact repositório como uma dependência ao seu projeto Swift no Xcode, você está recebendo um erro 401 não autorizado mesmo depois de seguir as instruções para conectar o Swift a. CodeArtifact

Possíveis correções: isso pode ser causado por um problema com o aplicativo macOS Keychain, onde suas CodeArtifact credenciais são armazenadas. Para corrigir isso, recomendamos abrir o aplicativo Keychain, excluir todas as CodeArtifact entradas e configurar o Swift Package Manager com seu CodeArtifact repositório novamente seguindo as instruções em. Configure o Swift Package Manager com CodeArtifact

O Xcode trava na máquina de CI devido à solicitação de senha do Keychain

Problema: quando você está tentando extrair pacotes do Swift CodeArtifact como parte de uma compilação do Xcode em um servidor de integração contínua (CI), como com o GitHub Actions, a autenticação com CodeArtifact pode travar e, eventualmente, falhar com uma mensagem de erro semelhante à seguinte:

```
Failed to save credentials for
\'https://my_domain-111122223333.d.codeartifact.us-west-2.amazonaws.com\'
to keychain: status -60008
```

Possíveis correções: isso ocorre porque as credenciais não são salvas no Keychain em máquinas de CI e o Xcode é compatível somente com as credenciais salvas no Keychain. Para corrigir isso, recomendamos criar a entrada do Keychain manualmente, usando as seguintes etapas:

1. Prepare o Keychain.

```
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. Obtenha um token de CodeArtifact autenticação e o endpoint do seu repositório.

```
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. Crie manualmente a entrada do Keychain.

```
SERVER=$(echo $CODEARTIFACT_REPO | sed  's/https:\/\///g' | sed 's/.com.*$/.com/g')
AUTHORIZATION=(-T /usr/bin/security -T /usr/bin/codesign -T /usr/bin/xcodebuild -
T /usr/bin/swift \
               -T /Applications/Xcode-15.2.app/Contents/Developer/usr/bin/
xcodebuild)

security delete-internet-password -a token -s $SERVER -r htps "${KEYCHAIN_NAME}"

security add-internet-password -a token \
                            -s $SERVER \
```

```
                                -w $CODEARTIFACT_AUTH_TOKEN \
                                -r htps \
                                -U \
                                "${AUTHORIZATION[@]}" \
                                "${KEYCHAIN_NAME}"

security set-internet-password-partition-list \
             -a token \
             -s $SERVER \
             -S "com.apple.swift-
package,com.apple.security,com.apple.dt.Xcode,apple-tool:,apple:,codesign" \
             -k "${KEYCHAIN_PASSWORD}" "${KEYCHAIN_NAME}"

security find-internet-password   "${KEYCHAIN_NAME}"
```

Para obter mais informações sobre esse erro e a solução, consulte https://github.com/apple/swift-package-manager/issues/7236.

Usando CodeArtifact com pacotes genéricos

Esses tópicos mostram como consumir e publicar para pacotes genéricos usando AWS CodeArtifact.

Tópicos

Visão geral dos pacotes genéricos

Usando o formato de `generic` pacote, você pode carregar qualquer tipo de arquivo para criar um pacote em um CodeArtifact repositório. Os pacotes genéricos não estão associados a nenhuma linguagem de programação específica, tipo de arquivo ou ecossistema de gerenciamento de pacotes. Isso pode ser útil para armazenar e controlar versões de artefatos de compilação arbitrários, como instaladores de aplicativos, modelos de machine learning, arquivos de configuração e outros.

Um pacote genérico consiste em um nome de pacote, namespace, versão e um ou mais ativos (ou arquivos). Pacotes genéricos podem existir junto com pacotes de outros formatos em um único CodeArtifact repositório.

Você pode usar o AWS CLI ou SDK para trabalhar com pacotes genéricos. Para obter uma lista completa dos AWS CLI comandos que funcionam com pacotes genéricos, consulteComandos compatíveis com pacotes genéricos.

Restrições de pacotes genéricos

- Eles nunca são obtidos de repositórios upstream. Eles só podem ser obtidos no repositório no qual foram publicados.
- Eles não podem declarar dependências a serem retornadas ListPackageVersionDependenciesou exibidas no. Console de gerenciamento da AWS
- Eles podem armazenar arquivos README e LICENSE, mas não são interpretados por. CodeArtifact As informações nesses arquivos não são retornadas de GetPackageVersionReadmeou DescribePackageVersion, e não aparecem no Console de gerenciamento da AWS.

- Como em todos os pacotes CodeArtifact, há limites para o tamanho do ativo e o número de ativos por pacote. Para obter mais informações sobre limites e cotas em CodeArtifact, consulteCotas em AWS CodeArtifact.
- Os nomes dos ativos que eles contêm devem seguir estas regras:
  - Os nomes dos ativos podem usar letras e números Unicode. Especificamente, são permitidas estas categorias de caracteres Unicode: letra minúscula (`Ll`), letra modificadora (`Lm`), outra letra (`Lo`), letra do título (`Lt`), letra maiúscula (`Lu`), número da letra (`Nl`) e número decimal (`Nd`).
  - Os seguintes caracteres especiais são permitidos: `~!@^&()-_+[]{};,.`
  - Os ativos não podem ser chamados `.` ou `..`
  - Espaços são o único caractere de espaço em branco permitido. Os nomes de ativos não podem começar ou terminar com um caractere de espaço ou incluir espaços consecutivos.

Comandos compatíveis com pacotes genéricos

Você pode usar o AWS CLI ou SDK para trabalhar com pacotes genéricos. Os CodeArtifact comandos a seguir funcionam com pacotes genéricos:

- copy-package-versions (consulte Copiar pacotes entre repositórios)
- delete-package (consulte Excluindo um pacote (AWS CLI))
- delete-package-versions (consulte Excluindo uma versão do pacote (AWS CLI))
- describe-package
- describe-package-version (consulte Exiba e atualize os detalhes e dependências da versão do pacote)
- dispose-package-versions (consulte Descartar as versões do pacote)
- get-package-version-asset (consulte Baixe ativos da versão do pacote)
- list-package-version-assets (consulte Listar ativos da versão do pacote)
- list-package-versions (consulte Listar versões de pacotes)
- list-packages (consulte Listar nomes de pacotes)
- publish-package-version (consulte Publicando um pacote genérico)
- put-package-origin-configuration (consulte Editar controles de origem do pacote)

> **Note**
>
> Você pode usar a configuração de controle de origem `publish` para permitir ou bloquear a publicação de um nome de pacote genérico em um repositório. No entanto, a configuração `upstream` não se aplica a pacotes genéricos porque eles não podem ser obtidos em um repositório upstream.

- update-package-versions-status (consulte Atualizar o status da versão do pacote)

Publicando e consumindo pacotes genéricos

Para publicar uma versão genérica do pacote e seus ativos relacionados, use o comando `publish-package-version`. Você pode listar os ativos de um pacote genérico usando o comando `list-package-version-asset` e baixá-los usando `get-package-version-asset`. O tópico a seguir contém step-by-step instruções para publicar pacotes genéricos ou baixar ativos de pacotes genéricos usando esses comandos.

Publicando um pacote genérico

Um pacote genérico consiste em um nome de pacote, namespace, versão e um ou mais ativos (ou arquivos). Este tópico demonstra como publicar um pacote chamado `my-package`, com o namespace `my-ns`, a versão `1.0.0` e contendo um ativo chamado `asset.tar.gz`.

Pré-requisitos:

- Configure e configure o AWS Command Line Interface com CodeArtifact (consulteConfigurando com AWS CodeArtifact)
- Tenha um CodeArtifact domínio e um repositório (consulteConceitos básicos do uso da AWS CLI)

Para publicar um pacote genérico

1. Use o comando a seguir para gerar o SHA256 hash para cada arquivo que você deseja carregar em uma versão do pacote e coloque o valor em uma variável de ambiente. Esse valor é usado como uma verificação de integridade para conferir se o conteúdo do arquivo não foi alterado após o envio 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. Chame `publish-package-version` para carregar o pacote e criar uma nova versão dele.

> **Note**
>
> Se seu pacote contiver mais de um ativo, você poderá chamar `publish-package-version` uma vez para que cada ativo seja carregado. Inclua o argumento `--unfinished` para cada chamada de `publish-package-version`, exceto ao fazer o carregamento do ativo final. Omitir `--unfinished` definirá o status da versão do pacote como `Published` e impedirá que ativos adicionais sejam enviados para ele.
> Como alternativa, inclua `--unfinished` para cada chamada de `publish-package-version` e, em seguida, defina o status da versão do pacote como `Published` usando o 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%
```

A seguir, é mostrada a saída.

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

Listando ativos de pacotes genéricos

Para listar os ativos contidos em um pacote genérico, use o comando `list-package-version-assets`. Para obter mais informações, consulte Listar ativos da versão do pacote.

O exemplo a seguir lista os ativos da versão `1.0.0` do pacote `my-package`.

Para listar ativos da versão do pacote

- Chame `list-package-version-assets` para listar os ativos contidos em um pacote 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
  ```

  A seguir, é mostrada a saída.

  ```
  {
      "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"
  ```

```
}
```

Baixando ativos de pacotes genéricos

Para baixar os ativos de um pacote genérico, use o comando `get-package-version-asset`. Para obter mais informações, consulte Baixe ativos da versão do pacote.

O exemplo a seguir baixa o ativo `asset.tar.gz` da versão `1.0.0` do pacote `my-package` para o diretório de trabalho atual em um arquivo também chamado `asset.tar.gz`.

Para baixar ativos da versão do pacote

- Chame `get-package-version-asset` para baixar ativos de um pacote 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
  ```

  A seguir, é mostrada a saída.

  ```
  {
      "assetName": "asset.tar.gz",
      "packageVersion": "1.0.0",
      "packageVersionRevision": "REVISION-SAMPLE-1-C7F4S5E9B772FC"
  }
  ```

Usando CodeArtifact com CodeBuild

Esses tópicos descrevem como usar pacotes em um CodeArtifact repositório em um projeto de AWS CodeBuild compilação.

Tópicos

Usando pacotes npm em CodeBuild

As etapas a seguir foram testadas com os sistemas operacionais listados nas imagens do Docker fornecidas pelo CodeBuild.

Configurar permissões com perfis do IAM

Essas etapas são necessárias ao usar pacotes npm a partir do CodeArtifact in CodeBuild.

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Perfis. Na página Funções, edite a função usada pelo seu projeto de CodeBuild compilação. Essa função deve ter as seguintes permissões.

   JSON

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

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

> **Important**
>
> Se você também quiser usar CodeBuild para publicar pacotes, adicione a **`codeartifact:PublishPackageVersion`** permissão.

Para obter informações, consulte Modificar uma função no Guia do usuário do IAM.

Faça login e use o npm

Para usar pacotes npm de CodeBuild, execute o `login` comando na `pre-build` seção do seu projeto `buildspec.yaml` para configurar `npm` a busca de pacotes. CodeArtifact Para obter mais informações, consulte Autenticação com npm.

Depois de `login` ter sido executado com êxito, você pode executar os comandos `npm` na seção `build` para instalar pacotes npm.

Linux

> **Note**
>
> Só é necessário atualizar o AWS CLI with `pip3 install awscli --upgrade --user` se você estiver usando uma CodeBuild imagem mais antiga. Se você estiver usando as versões mais recentes da imagem, poderá remover essa linha.

```
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
```

Usando pacotes Python em CodeBuild

As etapas a seguir foram testadas com os sistemas operacionais listados nas imagens do Docker fornecidas pelo CodeBuild.

Configurar permissões com perfis do IAM

Essas etapas são necessárias ao usar pacotes Python a partir de CodeArtifact dentro. CodeBuild

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Perfis. Na página Funções, edite a função usada pelo seu projeto de CodeBuild compilação. Essa função deve ter as seguintes permissões.

   JSON

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

> **Important**
>
> Se você também quiser usar CodeBuild para publicar pacotes, adicione a **`codeartifact:PublishPackageVersion`** permissão.

Para obter informações, consulte Modificar uma função no Guia do usuário do IAM.

Faça login e use pip ou twine

Para usar pacotes Python do CodeBuild, execute o `login` comando na `pre-build` seção do `buildspec.yaml` arquivo do seu projeto para configurar `pip` a busca de pacotes. CodeArtifact Para obter mais informações, consulte Usando CodeArtifact com Python.

Depois de `login` ter sido executado com êxito, você pode executar os comandos `pip` na seção `build` para instalar ou publicar pacotes Python.

Linux

> **Note**
>
> Só é necessário atualizar o AWS CLI with `pip3 install awscli --upgrade --user` se você estiver usando uma CodeBuild imagem mais antiga. Se você estiver usando as versões mais recentes da imagem, poderá remover essa linha.

Para instalar pacotes Python usando `pip`:

```
pre_build:
  commands:
    - pip3 install awscli --upgrade --user
    - aws codeartifact login --tool pip --domain my_domain --domain-owner 111122223333
 --repository my_repo
build:
  commands:
    - pip install requests
```

Para publicar pacotes Python usando `twine`:

```
pre_build:
  commands:
    - pip3 install awscli --upgrade --user
    - aws codeartifact login --tool twine --domain my_domain --domain-
owner 111122223333 --repository my_repo
```

```
build:
  commands:
    - twine upload --repository codeartifact mypackage
```

Windows

Para instalar pacotes Python usando `pip`:

```
version: 0.2
phases:
  install:
    commands:
      - '[Net.ServicePointManager]::SecurityProtocol = "Tls12"; Invoke-WebRequest
 https://awscli.amazonaws.com/AWSCLIV2.msi -OutFile $env:TEMP/AWSCLIV2.msi'
      - Start-Process -Wait msiexec "/i $env:TEMP\AWSCLIV2.msi /quiet /norestart"
  pre_build:
    commands:
      - '&"C:\Program Files\Amazon\AWSCLIV2\aws" codeartifact login --tool pip --
domain my_domain --domain-owner 111122223333 --repository my_repo'
  build:
    commands:
      - pip install requests
```

Para publicar pacotes Python usando `twine`:

```
version: 0.2
phases:
  install:
    commands:
      - '[Net.ServicePointManager]::SecurityProtocol = "Tls12"; Invoke-WebRequest
 https://awscli.amazonaws.com/AWSCLIV2.msi -OutFile $env:TEMP/AWSCLIV2.msi'
      - Start-Process -Wait msiexec "/i $env:TEMP\AWSCLIV2.msi /quiet /norestart"
  pre_build:
    commands:
      - '&"C:\Program Files\Amazon\AWSCLIV2\aws" codeartifact login --tool twine --
domain my_domain --domain-owner 111122223333 --repository my_repo'
  build:
    commands:
      - twine upload --repository codeartifact mypackage
```

Usando pacotes Maven em CodeBuild

Configurar permissões com perfis do IAM

Essas etapas são necessárias ao usar pacotes Maven a partir de CodeArtifact dentro CodeBuild.

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Perfis. Na página Funções, edite a função usada pelo seu projeto de CodeBuild compilação. Essa função deve ter as seguintes permissões.

   JSON

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

> **Important**
>
> Se você também quiser usar CodeBuild para publicar pacotes, adicione as **`codeartifact:PutPackageMetadata`** permissões **`codeartifact:PublishPackageVersion`** e.

Para obter informações, consulte Modificar uma função no Guia do usuário do IAM.

Usar gradle ou mvn

Para usar pacotes Maven com `gradle` ou`mvn`, armazene o token de CodeArtifact autenticação em uma variável de ambiente, conforme descrito em Passe um token de autenticação em uma variável de ambiente. Veja um exemplo do a seguir:

> **Note**
>
> Só é necessário atualizar o AWS CLI with `pip3 install awscli --upgrade --user` se você estiver usando uma CodeBuild imagem mais antiga. Se você estiver usando as versões mais recentes da imagem, poderá remover essa linha.

```
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 o Gradle:

Se você fez referência à `CODEARTIFACT_AUTH_TOKEN` variável no seu `build.gradle` arquivo Gradle conforme descrito em Usando CodeArtifact com o Gradle, você pode invocar sua compilação do Gradle na seção. `buildspec.yaml` `build`

```
build:
  commands:
```

```
    - gradle build
```

Para usar o mvn:

Você deve configurar seus arquivos de configuração do Maven (`settings.xml`e `pom.xml`) seguindo as instruções em Usando CodeArtifact com mvn.

Usando NuGet pacotes em CodeBuild

As etapas a seguir foram testadas com os sistemas operacionais listados nas imagens do Docker fornecidas pelo CodeBuild.

Tópicos

- Configurar permissões com perfis do IAM
- Consuma NuGet pacotes
- Crie com NuGet pacotes
- Publique NuGet pacotes

Configurar permissões com perfis do IAM

Essas etapas são necessárias ao usar NuGet pacotes de CodeArtifact dentro CodeBuild.

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Perfis. Na página Funções, edite a função usada pelo seu projeto de CodeBuild compilação. Essa função deve ter as seguintes permissões.

   JSON

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

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

> **Important**
>
> Se você também quiser usar CodeBuild para publicar pacotes, adicione a **`codeartifact:PublishPackageVersion`** permissão.

Para obter informações, consulte Modificar uma função no Guia do usuário do IAM.

Consuma NuGet pacotes

Para consumir NuGet pacotes do CodeBuild, inclua o seguinte no `buildspec.yaml` arquivo do seu projeto.

1. Na `install` seção, instale o provedor de CodeArtifact credenciais para configurar ferramentas de linha de comando, como `dotnet` criar `msbuild` e publicar CodeArtifact pacotes.
2. Na `pre-build` seção, adicione seu CodeArtifact repositório à sua NuGet configuração.

Veja os exemplos de `buildspec.yaml` a seguir. Para obter mais informações, consulte Usando o CodeArtifact com NuGet.

Depois que o provedor de credenciais for instalado e a fonte do repositório for adicionada, você poderá executar os comandos da ferramenta NuGet CLI na `build` seção para consumir pacotes. NuGet

Linux

Para consumir NuGet pacotes 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 add package <packageName> --source codeartifact
```

Windows

Para consumir NuGet pacotes 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 add package <packageName> --source codeartifact
```

Crie com NuGet pacotes

Para criar com NuGet pacotes do CodeBuild, inclua o seguinte no `buildspec.yaml` arquivo do seu projeto.

1. Na `install` seção, instale o provedor de CodeArtifact credenciais para configurar ferramentas de linha de comando, como `dotnet` criar `msbuild` e publicar CodeArtifact pacotes.
2. Na `pre-build` seção, adicione seu CodeArtifact repositório à sua NuGet configuração.

Veja os exemplos de `buildspec.yaml` a seguir. Para obter mais informações, consulte Usando o CodeArtifact com NuGet.

Depois que o provedor de credenciais for instalado e a fonte do repositório for adicionada, você poderá executar os comandos da ferramenta NuGet CLI, como `dotnet build` na seção. `build`

Linux

Para criar NuGet pacotes 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 criar NuGet pacotes 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 criar NuGet pacotes 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 criar NuGet pacotes 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
```

Publique NuGet pacotes

Para publicar NuGet pacotes do CodeBuild, inclua o seguinte no `buildspec.yaml` arquivo do seu projeto.

1. Na `install` seção, instale o provedor de CodeArtifact credenciais para configurar ferramentas de linha de comando, como `dotnet` criar `msbuild` e publicar CodeArtifact pacotes.
2. Na `pre-build` seção, adicione seu CodeArtifact repositório à sua NuGet configuração.

Veja os exemplos de `buildspec.yaml` a seguir. Para obter mais informações, consulte Usando o CodeArtifact com NuGet.

Depois que o provedor de credenciais for instalado e a fonte do repositório for adicionada, você poderá executar os comandos da ferramenta NuGet CLI na `build` seção e publicar seus pacotes. NuGet

Linux

Para publicar NuGet pacotes 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 pack -o .
      - dotnet nuget push *.nupkg -s codeartifact
```

Windows

Para publicar NuGet pacotes 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 pack -o .
      - dotnet nuget push *.nupkg -s codeartifact
```

Cache de dependências

Você pode ativar o armazenamento em cache local CodeBuild para reduzir o número de dependências que precisam ser buscadas para cada compilação. CodeArtifact Para obter mais informações, consulte Armazenamento em cache de builds no AWS CodeBuild no Guia do usuário do AWS CodeBuild . Depois de habilitar um cache local personalizado, adicione o diretório de cache ao arquivo `buildspec.yaml` do seu projeto.

Por exemplo, se estiver usando `mvn`, faça o seguinte:

```
cache:
  paths:
    - '/root/.m2/**/*'
```

Para outras ferramentas, use as pastas de cache mostradas nesta tabela.

Ferramenta	Diretório de cache
mvn	`/root/.m2/**/*`
gradle	`/root/.gradle/caches/**/*`
pip	`/root/.cache/pip/**/*`
npm	`/root/.npm/**/*`
nuget	`/root/.nuget/**/*`
yarn (classic)	`/root/.cache/yarn/**/*`

Monitoramento CodeArtifact

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de CodeArtifact suas outras AWS soluções. AWS fornece as seguintes ferramentas de monitoramento para observar CodeArtifact, relatar quando algo está errado e realizar ações automáticas quando apropriado:

- Você pode usar EventBridge a Amazon para automatizar seus AWS serviços e responder automaticamente a eventos do sistema, como problemas de disponibilidade de aplicativos ou alterações de recursos. Os eventos dos AWS serviços são entregues quase EventBridge em tempo real. Você pode escrever regras simples para determinar quais eventos são do seu interesse, e as ações automatizadas a serem tomadas quando um evento corresponder à regra. Para obter mais informações, consulte o Guia EventBridge do usuário da Amazon CodeArtifact formato e exemplo de evento e.
- Você pode usar CloudWatch as métricas da Amazon para visualizar o CodeArtifact uso por operação. CloudWatch as métricas incluem todas as solicitações feitas para CodeArtifact, e as solicitações são mostradas por conta. Você pode visualizar essas métricas em CloudWatch métricas navegando até o namespace Usage/By AWS Resource. AWS Para obter mais informações, consulte Usar CloudWatch métricas da Amazon no Guia CloudWatch do usuário da Amazon.

Tópicos

Monitoramento de CodeArtifact eventos

CodeArtifact é integrado à Amazon EventBridge, um serviço que automatiza e responde a eventos, incluindo alterações em um CodeArtifact repositório. Você pode criar regras para eventos e configurar o que acontece quando um evento corresponde a uma regra. EventBridge era anteriormente chamado de CloudWatch Eventos.

As seguintes ações podem ser acionadas por um evento:

- Invocando uma AWS Lambda função.
- Ativando uma máquina de AWS Step Functions estado.
- Notificação de um tópico do Amazon SNS ou de uma fila do Amazon SQS.
- Iniciando um pipeline em AWS CodePipeline.

CodeArtifact cria um evento quando uma versão do pacote é criada, modificada ou excluída. Veja a seguir exemplos de CodeArtifact eventos:

- Publicar uma nova versão de pacote (por exemplo, executando `npm publish`).
- Adicionar um novo ativo a uma versão de pacote existente (por exemplo, enviando um novo arquivo JAR para um pacote Maven existente).
- Copiar uma versão de pacote de um repositório para outro usando`copy-package-versions`. Para obter mais informações, consulte Copiar pacotes entre repositórios.
- Excluir versões do pacote usando o `delete-package-versions`. Para obter mais informações, consulte Excluir um pacote ou uma versão do pacote.
- Excluir versões de um pacote usando o `delete-package`. Um evento será publicado para cada versão do pacote excluído. Para obter mais informações, consulte Excluir um pacote ou uma versão do pacote.
- Manter uma versão de pacote em um repositório downstream quando ela foi obtida de um repositório upstream. Para obter mais informações, consulte Trabalhando com repositórios upstream em CodeArtifact.
- Ingestão de uma versão de pacote de um repositório externo em um CodeArtifact repositório. Para obter mais informações, consulte Conectar um CodeArtifact repositório a um repositório público.

Os eventos são entregues tanto para a conta proprietária do domínio quanto para a que administra o repositório. Por exemplo, suponha que essa conta `111111111111` seja proprietária do domínio `my_domain`. A conta `222222222222` cria um repositório no `my_domain` chamado `repo2`. Quando uma nova versão do pacote é publicada no`repo2`, ambas as contas recebem os EventBridge eventos. A conta proprietária do domínio (`111111111111`) recebe eventos para todos os repositórios no domínio. Se uma única conta for proprietária tanto do domínio quanto do repositório dentro dela, somente um único evento será entregue.

Os tópicos a seguir descrevem o formato do CodeArtifact evento. Eles mostram como configurar CodeArtifact eventos e como usar eventos com outros AWS serviços. Para obter mais informações, consulte Getting Started with Amazon EventBridge no Guia EventBridge do usuário da Amazon.

CodeArtifact formato e exemplo de evento

A seguir estão os campos e as descrições do evento, juntamente com um exemplo de um CodeArtifact evento.

CodeArtifact formato de evento

Todos os CodeArtifact eventos incluem os seguintes campos.

Campo do evento	Description
version	A versão do formato do evento. Atualmente, há apenas uma versão, `0`.
id	Um identificador exclusivo do evento.
tipo de detalhe	O tipo de evento. Determina os campos no objeto `detail`. Atualmente, o único `detail-type` compatível é `CodeArtifact Package Version State Change` .
origem	A origem do evento. Pois CodeArtifact, será`aws.codeartifact` .
account	O AWS ID da conta que recebe o evento.
horário	O horário exato em que o evento foi acionado.
região	A região onde o evento foi acionado.
recursos	Uma lista que contém o ARN do pacote que foi alterado. A lista contém uma entrada. Para obter informações sobre o formato do ARN do pacote, consulte Conceder acesso de gravação aos pacotes.
domainName	O domínio que contém o repositório contendo o pacote.
domainOwner	O ID da AWS conta do proprietário do domínio.

Campo do evento	Description
repositoryName	O repositório que contém o pacote.
repositoryAdministrator	O ID da AWS conta do administrador do repositório.
packageFormat	O formato do pacote que acionou o evento.
packageNamespace	O namespace do pacote que acionou o evento.
packageName	O nome do pacote que acionou o evento.
packageVersion	A versão do pacote que acionou o evento.
packageVersionState	O estado da versão do pacote quando o evento foi acionado. Os valores possíveis são `Unfinished` , `Published` , `Unlisted`, `Archived` e `Disposed`.
packageVersionRevision	Um valor que identifica de maneira exclusiva o estado dos ativos e metadados da versão do pacote quando o evento foi acionado. Se a versão do pacote for modificada (por exemplo, com a adição de outro arquivo JAR a um pacote Maven), `packageVersionRevision` mudará.
changes.assetsAdded	O número de ativos adicionados a um pacote que acionou um evento. Exemplos de um ativo são um arquivo JAR do Maven ou wheel do Python.
changes.assetsRemoved	O número de ativos removidos de um pacote que acionou um evento.
changes.assetsUpdated	O número de ativos modificados no pacote que acionou o evento.

Campo do evento	Description
changes.metadataUpdated	Um valor booleano definido como `true` se o evento incluir metadados modificados em nível de pacote. Por exemplo, um evento pode modificar um arquivo `pom.xml` do Maven.
changes.statusChanged	Um valor booleano definido como `true` se o `packageVersionStatus` do evento for modificado (por exemplo, se `packageVersionStatus` mudar de `Unfinished` para`Published`).
operationType	Descreve o tipo de alto nível da alteração da versão do pacote. Os valores possíveis são `Created`, `Updated` e `Deleted`.
sequenceNumber	Um número inteiro que especifica um número de evento para um pacote. Cada evento em um pacote o incrementa o `sequenceNumber` para que os eventos possam ser dispostos de maneira sequencial. Um evento pode incrementar o `sequenceNumber` em qualquer número inteiro. **Note** EventBridge os eventos podem ser recebidos fora de ordem. `sequenceNumber` pode ser usado para determinar seu pedido real.

Campo do evento	Description
eventDeduplicationId	Um ID usado para diferenciar eventos duplicados. EventBridge Em casos raros, EventBridge pode acionar a mesma regra mais de uma vez para um único evento ou horário agendado. Ou pode invocar mais de uma vez o mesmo alvo para uma determinada regra acionada.

CodeArtifact exemplo de evento

Veja a seguir um exemplo de um CodeArtifact evento que pode ser acionado quando um pacote npm é publicado.


```
{
      "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 um evento para iniciar uma CodePipeline execução

Este exemplo demonstra como configurar uma EventBridge regra da Amazon para que uma AWS CodePipeline execução comece quando uma versão do pacote em um CodeArtifact repositório for publicada, modificada ou excluída.

Tópicos

- Configurar EventBridge permissões
- Crie a EventBridge regra
- Crie o destino da EventBridge regra

Configurar EventBridge permissões

Você deve adicionar permissões para usar EventBridge CodePipeline para invocar a regra que você criou. Para adicionar essas permissões usando o AWS Command Line Interface (AWS CLI), siga a etapa 1 em Criar uma regra de CloudWatch eventos para uma CodeCommit fonte (CLI) no Guia do AWS CodePipeline usuário.

Crie a EventBridge regra

Para criar a regra, use o comando `put-rule` com os parâmetros `--name` e `--event-pattern`. O padrão de evento especifica valores que são comparados com o conteúdo de cada evento. O destino será acionado se o padrão corresponder ao evento. Por exemplo, o padrão a seguir corresponde aos CodeArtifact eventos do `myrepo` repositório no `my_domain` domínio.

```
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"]}}'
```

Crie o destino da EventBridge regra

O comando a seguir adiciona um destino à regra para que, quando um evento corresponda à regra, uma CodePipeline execução seja acionada. Para o parâmetro `RoleArn`, insira o nome do recurso da Amazon (ARN) do perfil criado anteriormente nesse tópico.

```
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'
```

Use um evento para executar uma função do Lambda

Este exemplo mostra como configurar uma EventBridge regra que inicia uma AWS Lambda função quando uma versão do pacote em um CodeArtifact repositório é publicada, modificada ou excluída.

Para obter mais informações, consulte Tutorial: Agendar o uso de AWS Lambda funções EventBridge no Guia EventBridge do usuário da Amazon.

Tópicos

- Crie a EventBridge regra
- Crie o destino da EventBridge regra
- Configurar EventBridge permissões

Crie a EventBridge regra

Para criar uma regra que inicie uma função do Lambda, use o comando `put-rule` com as opções `--name` e `--event-pattern`. O padrão a seguir especifica pacotes npm no escopo do `@types` em qualquer repositório no domínio `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"]}}'
```

Crie o destino da EventBridge regra

O comando a seguir adiciona um destino à regra que executa a função do Lambda quando um evento corresponde à regra. Para o parâmetro `arn`, especifique o nome do recurso da Amazon (ARN) da função do Lambda.

```
aws events put-targets --rule MyCodeArtifactRepoRule --targets \
  Id=1,Arn=arn:aws:lambda:us-west-2:111122223333:function:MyLambdaFunction
```

Configurar EventBridge permissões

Use o comando `add-permission` para conceder permissões para a regra invocar uma função do Lambda. Para o parâmetro `--source-arn`, especifique o ARN da regra criada anteriormente neste exemplo.

```
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
```

Segurança em CodeArtifact

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade que se aplicam CodeArtifact, consulte Serviços da AWS no escopo do programa de conformidade .
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar CodeArtifact. Os tópicos a seguir mostram como configurar para atender CodeArtifact aos seus objetivos de segurança e conformidade. Você também aprende a usar outros serviços da AWS que ajudam você a monitorar e proteger seus CodeArtifact recursos.

Tópicos

Proteção de dados em AWS CodeArtifact

O modelo de responsabilidade AWS compartilhada O modelo se aplica à proteção de dados na AWS CodeArtifact. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Perguntas frequentes sobre privacidade de dados . Para obter informações sobre proteção de dados na Europa, consulte o Centro de Regulamento Geral sobre a Proteção de Dados (RGPD).

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com CodeArtifact ou Serviços da AWS usa o console, a API ou AWS os SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de formato livre

usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Criptografia de dados

A criptografia é uma parte importante da CodeArtifact segurança. Algumas criptografias, como aquelas de dados em trânsito, são fornecidas por padrão e não requerem qualquer ação por parte do cliente. Outras criptografias, como aquelas de dados em repouso, podem ser configuradas durante a criação ou compilação do projeto.

- Criptografia de dados em repouso - Todos os ativos armazenados CodeArtifact são criptografados usando AWS KMS keys (chaves KMS). Isso inclui todos os ativos em todos os pacotes em todos os repositórios. Uma chave do KMS é usada para cada domínio, para criptografar todos os seus ativos. Por padrão, uma chave KMS AWS gerenciada é usada, então você não precisa criar uma chave KMS. Se quiser, é possível usar uma chave do KMS gerenciada pelo cliente, criada e configurada por você. Para obter mais informações, consulte Criar chaves e Conceitos do AWS Key Management Service no Guia do usuário do AWS Key Management Service . Você pode especificar uma chave do KMS gerenciada pelo cliente ao criar um domínio. Para obter mais informações, consulte Trabalhando com domínios em CodeArtifact.
- Criptografia de dados em trânsito - Toda a comunicação entre clientes e entre CodeArtifact CodeArtifact e suas dependências posteriores é protegida usando criptografia TLS.

Privacidade do tráfego

Você pode melhorar a segurança de seus CodeArtifact domínios e dos ativos que eles contêm configurando CodeArtifact para usar uma interface de endpoint de nuvem privada virtual (VPC). Para fazer isso, não é preciso ter um gateway da Internet, de um dispositivo NAT ou de um gateway privado virtual. Para obter mais informações, consulte Trabalhar com endpoints da Amazon VPC. Para obter mais informações sobre endpoints de VPC AWS PrivateLink e VPC, consulte AWS PrivateLinkComo acessar os serviços da AWS por meio de. PrivateLink

Monitoramento CodeArtifact

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de AWS CodeArtifact suas AWS soluções. Você deve coletar dados de monitoramento de todas as partes da sua AWS solução para poder depurar com mais facilidade uma falha de vários

pontos, caso ocorra. AWS fornece o seguinte para monitorar seus CodeArtifact recursos e responder a possíveis incidentes:

Tópicos

Registrando chamadas de CodeArtifact API com AWS CloudTrail

CodeArtifact é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço em CodeArtifact. CloudTrail captura todas as chamadas de API para CodeArtifact eventos, incluindo chamadas de clientes do gerenciador de pacotes.

Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon Simple Storage Service (Amazon S3), incluindo eventos para. CodeArtifact Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita CodeArtifact, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

CodeArtifact informações em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre em CodeArtifact, essa atividade é registrada em um CloudTrail evento junto com outros eventos AWS de serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo dos eventos em sua AWS conta, incluindo eventos para CodeArtifact, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Você também pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para saber mais, consulte os seguintes tópicos:

- Criar uma trilha para a conta da AWS

- CloudTrail Serviços e integrações compatíveis
- Configurando notificações do Amazon SNS para CloudTrail

Quando o CloudTrail registro está ativado em sua AWS conta, as chamadas de API feitas para CodeArtifact ações são rastreadas em arquivos de CloudTrail log, onde são gravadas com outros registros AWS de serviço. CloudTrail determina quando criar e gravar em um novo arquivo com base no período e no tamanho do arquivo.

Todas as CodeArtifact ações são registradas por CloudTrail. Por exemplo, chamadas para as ações `ListRepositories` (nas AWS CLI,`aws codeartifact list-repositories`), `CreateRepository` (`aws codeartifact create-repository`) e `ListPackages` (`aws codeartifact list-packages`) geram entradas nos arquivos de CloudTrail log, além dos comandos do cliente do gerenciador de pacotes. Os comandos do cliente do gerenciador de pacotes normalmente fazem mais de uma solicitação HTTP ao servidor. Cada solicitação gera um evento de CloudTrail registro separado.

Entrega de registros entre contas CloudTrail

Até três contas separadas recebem CloudTrail registros para uma única chamada de API:

- A conta que fez a solicitação, por exemplo, a conta que chamou `GetAuthorizationToken`.
- A conta do administrador do repositório, por exemplo, a conta que administra o repositório que `ListPackages` foi chamada.
- A conta do proprietário do domínio, por exemplo, a conta do proprietário do domínio que contém o repositório no qual uma API foi chamada.

Por APIs exemplo`ListRepositoriesInDomain`, são ações contra um domínio e não contra um repositório específico, somente a conta de chamada e a conta do proprietário do domínio recebem o CloudTrail registro. Por APIs `ListRepositories` isso, não são autorizados contra nenhum recurso, somente a conta do chamador recebe o CloudTrail registro.

Entendendo as entradas do arquivo de CodeArtifact log

CloudTrail os arquivos de log podem conter uma ou mais entradas de log. Cada entrada lista vários eventos com formatação JSON. Um evento de log representa uma única solicitação de qualquer origem e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros de solicitação, e assim por diante. As entradas de log não são um rastreamento de pilha ordenada das chamadas de API pública. Assim, elas não são exibidas em nenhuma ordem específica.

Tópicos

Exemplo: uma entrada de registro para chamar a GetAuthorizationToken API

Uma entrada de log criada por `GetAuthorizationToken` inclui o nome do domínio no 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"
}
```

Exemplo: uma entrada de log para buscar uma versão do pacote npm

As solicitações feitas por todos os clientes do gerenciador de pacotes, incluindo o cliente **`npm`**, têm dados adicionais registrados, incluindo o nome do domínio, nome do repositório e nome do pacote no campo `requestParameters`. O caminho do URL e o método HTTP são registrados no 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"
}
```


Validação de conformidade AWS CodeArtifact

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. Para obter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte a documentação AWS de segurança.

AWS Autenticação e tokens do CodeArtifact

O CodeArtifact exige que os usuários se autentiquem no serviço para publicar ou consumir versões do pacote. Você deve se autenticar no serviço CodeArtifact criando um token de autorização usando suas credenciais do AWS. Para criar um token de autorização, é necessário ter as permissões

corretas. Para obter as permissões necessárias para criar um token de autorização, consulte a entrada `GetAuthorizationToken` no AWS CodeArtifact referência de permissões. Para informações gerais sobre permissões no CodeArtifact, consulte Como AWS CodeArtifact funciona com o IAM.

Para obter um token de autorização do CodeArtifact, você deve chamar a API GetAuthorizationToken. Usando o AWS CLI, você pode chamar `GetAuthorizationToken` com o comando `login` ou `get-authorization-token`.

> **Note**
>
> Os usuários root não podem chamar `GetAuthorizationToken`.

- `aws codeartifact login`: este comando facilita a configuração de gerenciadores de pacotes comuns para usar o CodeArtifact em uma única etapa. Chamar `login` busca um token com `GetAuthorizationToken` e configura seu gerenciador de pacotes com o token e o endpoint correto do repositório do CodeArtifact. Os gerenciadores de pacotes de suporte são os seguintes:
  - dotnet
  - npm
  - nuget
  - pip
  - swift
  - twine
- `aws codeartifact get-authorization-token`: para gerenciadores de pacotes não compatíveis com `login`, você pode chamar `get-authorization-token` diretamente e depois configurar seu gerenciador de pacotes com o token conforme necessário, por exemplo, adicionando-o a um arquivo de configuração ou armazenando-o em uma variável de ambiente.

Os tokens de autenticação do CodeArtifact são válidos por um período padrão de 12 horas. Os tokens podem ser configurados com uma vida útil entre 15 minutos e 12 horas. Quando a vida útil expirar, você deverá obter outro token. A vida útil do token começa após `login` ou `get-authorization-token` após a chamada.

Se `login` ou `get-authorization-token` for chamado enquanto assume uma função, você pode configurar a vida útil do token para ser igual ao tempo restante na duração da sessão da

função definindo o valor de `--duration-seconds` para `0`. Caso contrário, a vida útil do token é independente da duração máxima da sessão da função. Por exemplo, suponha que você chame `sts assume-role` e especifique uma duração de sessão de 15 minutos e, em seguida, chame `login` para obter um token de autorização do CodeArtifact. Nesse caso, o token é válido por todo o período de 12 horas, mesmo que seja maior do que a duração da sessão de 15 minutos. Para obter informações sobre como controlar a duração da sessão, consulte Como usar perfis do IAM no Guia do usuário do IAM.

Tokens criados com o comando **`login`**

O comando `aws codeartifact login` busca um token com `GetAuthorizationToken` e configura seu gerenciador de pacotes com o token e o endpoint correto do repositório do CodeArtifact.

A tabela a seguir descreve os parâmetros para o comando `login`.

Parameter	Obrigatório	Descrição
`--tool`	Sim	O gerenciador de pacotes no qual se autenticar. Os valores possíveis são `dotnet`, `npm`, `nuget`, `pip`, `swift` e `twine`.
`--domain`	Sim	O nome de domínio ao qual o repositório pertence.
`--domain-owner`	Não	O ID do proprietário do domínio. Esse parâmetro é necessário se você estiver acessando um domínio pertencente a uma conta AWS na qual você não está autenticado. Para obter mais informações, consulte Cross-account domínios.
`--repository`	Sim	O nome do repositório no qual se autenticar.
`--duration-seconds`	Não	O tempo, em segundos, durante o qual as informações de login são

Parameter	Obrigatório	Descrição
		válidas. O valor mínimo é 900* e o valor máximo é 43200.
`--namespace`	Não	Associa um namespace à sua ferramenta de repositório.
`--dry-run`	Não	Imprima somente os comandos que seriam executados para conectar sua ferramenta ao seu repositório sem fazer nenhuma alteração na sua configuração.

*Um valor de 0 também é válido ao chamar `login` enquanto assume uma função. Chamar `login` com `--duration-seconds 0` cria um token com uma vida útil igual ao tempo restante na duração da sessão de uma função assumida.

O exemplo a seguir mostra como obter um token de autorização com o comando `login`.

```
aws codeartifact login \
    --tool dotnet | npm | nuget | pip | swift | twine \
    --domain my_domain \
    --domain-owner 111122223333 \
    --repository my_repo
```

Para obter orientações específicas sobre como usar o comando `login` com o npm, consulte Configure e use o npm com CodeArtifact. Para Python, consulte Usando CodeArtifact com Python.

Permissões necessárias para chamar a API **`GetAuthorizationToken`**

As permissões `sts:GetServiceBearerToken` e `codeartifact:GetAuthorizationToken` são necessárias para chamar a API `GetAuthorizationToken` do CodeArtifact.

Para usar um gerenciador de pacotes com um repositório do CodeArtifact, seu usuário ou perfil do IAM deve permitir `sts:GetServiceBearerToken`. Embora `sts:GetServiceBearerToken` possa ser adicionado a uma política de recursos de domínio do CodeArtifact, a permissão não terá efeito nessa política.

Tokens criados com a API `GetAuthorizationToken`

Você pode chamar `get-authorization-token` para obter um token de autorização do CodeArtifact.

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

Você pode alterar por quanto tempo um token é válido usando o argumento `--duration-seconds`. O valor mínimo é 900 e o valor máximo é 43200. O exemplo a seguir cria um 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
```

Se estiver chamando `get-authorization-token` enquanto assume uma função, a vida útil do token é independente da duração máxima da sessão da função. Você pode configurar o token para expirar quando a duração da sessão da função assumida expirar definindo `--duration-seconds` como 0.

```
aws codeartifact get-authorization-token \
    --domain my_domain \
    --domain-owner 111122223333 \
    --query authorizationToken \
    --output text \
    --duration-seconds 0
```

Consulte a documentação a seguir para obter mais informações:

- Para obter orientação sobre tokens e variáveis de ambiente, consulte Passar um token de autenticação usando uma variável de ambiente.
- Para usuários do Python, consulte Configurar o pip sem o comando login ou Configure e use o twine com CodeArtifact.

- Para usuários do Maven, consulte Usar o CodeArtifact com o Gradle ou Use CodeArtifact com mvn.
- Para usuários do npm, consulte Configuração do npm sem usar o comando login.

Passar um token de autenticação usando uma variável de ambiente

O CodeArtifact AWS usa tokens de autorização fornecidos pela API `GetAuthorizationToken` para autenticar e autorizar solicitações de ferramentas de criação, como Maven e Gradle. Para mais informações sobre esses tokens de autenticação, consulte Tokens criados com a API `GetAuthorizationToken`.

Você pode armazenar esses tokens de autenticação em uma variável de ambiente que pode ser lida por uma ferramenta de compilação para obter o token necessário para buscar pacotes de um repositório do CodeArtifact ou publicar pacotes nele.

Por motivos de segurança, essa abordagem é preferível a armazenar o token em um arquivo onde ele possa ser lido por outros usuários ou processos, ou acidentalmente verificado no controle de origem.

1. Configure suas credenciais AWS conforme descrito em Instale ou atualize e, em seguida, configure o AWS CLI.
2. Defina a `CODEARTIFACT_AUTH_TOKEN` variável de ambiente:

> **Note**
>
> Em alguns cenários, você não precisa incluir o argumento `--domain-owner`. Para obter mais informações, consulte Cross-account domínios.

- macOS ou Linux:

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

- Windows (usando o shell de comando padrão):

```
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
```

Revogando tokens de autorização do CodeArtifact

Quando um usuário autenticado cria um token para acessar os recursos do CodeArtifact, esse token dura até que seu período de acesso personalizável termine. O período de acesso padrão é de 12 horas. Em alguns casos, é possível revogar o acesso a um token antes que o período de acesso expire. Você pode revogar o acesso aos recursos do CodeArtifact seguindo estas instruções.

Se você criou o token de acesso usando credenciais de segurança temporárias, como funções assumidas ou acesso de usuário federado, pode revogar o acesso atualizando uma política do IAM para negar acesso. Para obter informações, consulte Desativando as permissões de credenciais de segurança temporárias no Guia do usuário do IAM.

Se você usou credenciais de usuário do IAM de longo prazo para criar o token de acesso, deverá modificar a política do usuário para negar o acesso ou excluir o usuário do IAM. Para obter mais informações, consulte Alteração de permissões para um usuário do IAM ou Excluindo um usuário do IAM.

Resiliência em AWS CodeArtifact

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. CodeArtifact A AWS opera em várias zonas de disponibilidade e armazena dados e metadados de artefatos no Amazon S3 e no Amazon DynamoDB. Seus dados criptografados são armazenados com redundância em várias instalações e diversos dispositivos em cada instalação, fazendo com que sejam altamente disponíveis e altamente duráveis.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Segurança da infraestrutura em AWS CodeArtifact

Como serviço gerenciado, AWS CodeArtifact é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar CodeArtifact pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Ataques de substituição de dependências

Os gerenciadores de pacotes simplificam o processo de criação de pacotes e compartilhamento do código reutilizável. Esses pacotes podem ser pacotes privados desenvolvidos por uma organização para uso em seus aplicativos ou públicos, geralmente pacotes de código aberto, desenvolvidos fora de uma organização e distribuídos por repositórios públicos de pacotes. Quando solicitam pacotes, os desenvolvedores dependem do gerenciador de pacotes para buscar novas versões de suas dependências. Os ataques de substituição de dependência, também conhecidos como ataques de confusão de dependência, exploram o fato de que um gerenciador de pacotes normalmente não tem como distinguir as versões legítimas de um pacote de versões maliciosas.

Os ataques de substituição de dependência pertencem a um subconjunto de truques conhecido como ataques à cadeia de suprimentos de software. Um ataque à cadeia de suprimentos de software é um ataque que tira proveito de vulnerabilidades em qualquer ponto da cadeia de suprimentos de software.

Um ataque de substituição de dependência pode ter como alvo qualquer pessoa que use tanto pacotes desenvolvidos internamente quanto aqueles obtidos de repositórios públicos. Os invasores

identificam nomes de pacotes internos e, depois, colocam estrategicamente códigos maliciosos com o mesmo nome em repositórios públicos de pacotes. Normalmente, o código malicioso é publicado em um pacote com um número de versão alto. Os gerenciadores de pacotes buscam o código malicioso desses feeds públicos porque acreditam que os pacotes maliciosos são as versões mais recentes do pacote. Isso causa uma “substituição” ou “confusão” entre o pacote desejado e o pacote malicioso, fazendo com que o código seja comprometido.

Para evitar ataques de substituição de dependências, o AWS CodeArtifact fornece controles de origem de pacotes. Os controles de origem de pacote são configurações que controlam como os pacotes podem ser adicionados aos seus repositórios. Os controles podem ser usados para garantir que as versões do pacote não possam ser publicadas diretamente no seu repositório e ingeridas de fontes públicas, protegendo você contra ataques de substituição de dependências. Os controles de origem podem ser configurados em pacotes individuais e em vários pacotes ao definir controles de origem em grupos de pacotes. Para obter mais informações sobre os controles de origem de pacotes e como alterá-los, consulte Editar controles de origem do pacote e Controles de origem do grupo de pacotes.

Identity and Access Management para AWS CodeArtifact

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar CodeArtifact os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte Solução de problemas AWS CodeArtifact de identidade e acesso).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte Como AWS CodeArtifact funciona com o IAM)
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte Exemplos de políticas baseadas em identidade para AWS CodeArtifact)

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários humanos usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório corporativo, provedor de identidade da web ou Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem funções que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos Centro de Identidade do AWS IAM. Para saber mais, consulte O que é o IAM Identity Center? no Guia do usuário do Centro de Identidade do AWS IAM .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você anexa a uma identidade (usuário, grupo ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.

- Políticas de controle de serviço (SCPs) — Especifique as permissões máximas para uma organização ou unidade organizacional em AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs) — Defina o máximo de permissões disponíveis para recursos em suas contas. Para obter mais informações, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como AWS CodeArtifact funciona com o IAM

Antes de usar o IAM para gerenciar o acesso CodeArtifact, saiba com quais recursos do IAM estão disponíveis para uso CodeArtifact.

Recursos do IAM que você pode usar com AWS CodeArtifact

Recurso do IAM	CodeArtifact apoio
Políticas baseadas em identidade	Sim
Políticas baseadas em atributos	Sim
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Não
ACLs	Não

Recurso do IAM	CodeArtifact apoio
ABAC (tags em políticas)	Parcial
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Não
Funções vinculadas ao serviço	Não

Para ter uma visão de alto nível de como CodeArtifact e outros AWS serviços funcionam com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade para CodeArtifact

Compatível com políticas baseadas em identidade: sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que podem ser anexados a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elemento de política JSON do IAM no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para CodeArtifact

Para ver exemplos de políticas CodeArtifact baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS CodeArtifact

Políticas baseadas em recursos dentro CodeArtifact

Compatível com políticas baseadas em recursos: sim

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. É necessário especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Ações políticas para CodeArtifact

Compatível com ações de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de CodeArtifact ações, consulte Ações definidas por AWS CodeArtifact na Referência de Autorização de Serviço.

As ações de política CodeArtifact usam o seguinte prefixo antes da ação:

```
codeartifact
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "codeartifact:action1",
      "codeartifact:action2"
         ]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "codeartifact:Describe*"
```

Para ver exemplos de políticas CodeArtifact baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS CodeArtifact

Recursos políticos para CodeArtifact

Compatível com recursos de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de CodeArtifact recursos e seus ARNs, consulte Recursos definidos por AWS CodeArtifact na Referência de Autorização de Serviço. Para saber com quais ações é possível especificar o ARN de cada atributo, consulte Ações definidas pelo AWS CodeArtifact. Para ver exemplos de especificação de CodeArtifact recursos ARNs em políticas, consulteAWS CodeArtifact recursos e operações.

Chaves de condição de política para CodeArtifact

Compatível com chaves de condição de política específicas de serviço: não

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

> **Note**
>
> AWS CodeArtifact não suporta as seguintes chaves de contexto de condição AWS global:
>
> - Referer
> - UserAgent

Para ver uma lista de chaves de CodeArtifact condição, consulte Chaves de condição AWS CodeArtifact na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas por AWS CodeArtifact.

Para ver exemplos de políticas CodeArtifact baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS CodeArtifact

ACLs in CodeArtifact

Suportes ACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com CodeArtifact

Compatível com ABAC (tags em políticas): parcial

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos chamados de tags. Você pode anexar tags a entidades e AWS recursos do IAM e, em seguida, criar políticas ABAC para permitir operações quando a tag do diretor corresponder à tag no recurso.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para saber mais sobre o ABAC, consulte Definir permissões com autorização do ABAC no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso por atributo (ABAC) no Guia do usuário do IAM.

Para obter mais informações sobre a marcação de CodeArtifact recursos, incluindo exemplos de políticas baseadas em identidade para limitar o acesso a um recurso com base nas tags desse recurso, consulte. Usando tags para controlar o acesso aos CodeArtifact recursos

Usando credenciais temporárias com CodeArtifact

Compatível com credenciais temporárias: sim

As credenciais temporárias fornecem acesso de curto prazo aos AWS recursos e são criadas automaticamente quando você usa a federação ou troca de funções. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para ter mais informações, consulte Credenciais de segurança temporárias no IAM e Serviços da Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Permissões principais entre serviços para CodeArtifact

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

As sessões de acesso direto (FAS) usam as permissões do principal chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) de fazer solicitações aos serviços posteriores. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.

Há duas ações de CodeArtifact API que exigem que o responsável pela chamada tenha permissões em outros serviços:

1. `GetAuthorizationToken` requer `sts:GetServiceBearerToken` junto com `codeartifact:GetAuthorizationToken`.
2. `CreateDomain`, ao fornecer uma chave de criptografia não padrão, requer ambas `kms:DescribeKey` e `kms:CreateGrant` na chave KMS junto de `codeartifact:CreateDomain`.

Para obter mais informações sobre as permissões e os recursos necessários para ações em CodeArtifact, consulteAWS CodeArtifact referência de permissões.

Funções de serviço para CodeArtifact

Compatível com perfis de serviço: não

O perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber mais, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper CodeArtifact a funcionalidade. Edite as funções de serviço somente quando CodeArtifact fornecer orientação para fazer isso.

Funções vinculadas a serviços para CodeArtifact

Compatível com perfis vinculados ao serviço: Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir o perfil de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Perfil vinculado ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Exemplos de políticas baseadas em identidade para AWS CodeArtifact

Por padrão, usuários e perfis não têm permissão para criar ou modificar recursos do CodeArtifact. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte Criar políticas do IAM (console) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos por CodeArtifact, incluindo o formato do ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição AWS CodeArtifact na Referência de Autorização de Serviço.

Tópicos

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir CodeArtifact recursos em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando

SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.

- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usando o CodeArtifact console

Para acessar o AWS CodeArtifact console, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os CodeArtifact recursos em seu Conta da AWS. Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam à operação de API que estiverem tentando executar.

Para garantir que usuários e funções ainda possam usar o CodeArtifact console, anexe também a política `AWSCodeArtifactAdminAccess` ou a política `AWSCodeArtifactReadOnlyAccess` AWS gerenciada às entidades. Para saber mais, consulte Adicionar permissões a um usuário no Guia do usuário do IAM.

Políticas gerenciadas (predefinidas) pela AWS para AWS CodeArtifact

AWS aborda muitos casos de uso comuns fornecendo políticas autônomas do IAM que são criadas e administradas pela AWS. Essas políticas AWS gerenciadas concedem as permissões necessárias para casos de uso comuns, para que você não precise investigar quais permissões são necessárias. Para obter mais informações, consulte Políticas gerenciadas pela AWS no Guia do usuário do IAM.

As políticas AWS gerenciadas a seguir, que você pode anexar aos usuários em sua conta, são específicas de AWS CodeArtifact.

- `AWSCodeArtifactAdminAccess`— Fornece acesso total à CodeArtifact inclusão de permissões para administrar CodeArtifact domínios.

  JSON

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

- `AWSCodeArtifactReadOnlyAccess`— Fornece acesso somente para leitura a. CodeArtifact

  JSON

```
{
```

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

Para criar e gerenciar funções CodeArtifact de serviço, você também deve anexar a política AWS gerenciada chamada`IAMFullAccess`.

Você também pode criar as próprias políticas do IAM personalizadas a fim de conceder permissões para ações e recursos do CodeArtifact. Você pode anexar essas políticas personalizadas a usuários ou grupos do IAM que exijam essas permissões.

Permitir que os usuários visualizem as próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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 um usuário obtenha informações sobre repositórios e domínios

A política a seguir permite que um usuário ou função do IAM liste e descreva qualquer tipo de CodeArtifact recurso, incluindo domínios, repositórios, pacotes e ativos. A política também inclui a `codeArtifact:ReadFromRepository` permissão, que permite ao diretor buscar pacotes de um CodeArtifact repositório. Ele não permite a criação de novos domínios ou repositórios e não permite a publicação de novos pacotes.

As permissões `codeartifact:GetAuthorizationToken` e `sts:GetServiceBearerToken` são necessárias para chamar a API `GetAuthorizationToken`.

JSON

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

Permitir que um usuário obtenha informações sobre domínios específicos

O exemplo a seguir mostra uma política de permissões que permite que um usuário listar domínios apenas na região `us-east-2`, para a conta `123456789012`, para qualquer domínio que comece com o nome `my`.

JSON

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

```
            "Effect": "Allow",
            "Action": "codeartifact:ListDomains",
            "Resource": "arn:aws:codeartifact:us-east-2:111122223333:domain/my*"

        }
    ]
}
```

Permitir que um usuário obtenha informações sobre repositórios específicos

Veja a seguir um exemplo de uma política de permissões que permite ao usuário obter informações sobre repositórios que terminam com `test`, incluindo informações sobre os pacotes neles contidos. O usuário não poderá publicar, criar ou excluir recursos.

As permissões `codeartifact:GetAuthorizationToken` e `sts:GetServiceBearerToken` são necessárias para chamar a API `GetAuthorizationToken`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "codeartifact:List*",
            "codeartifact:Describe*",
            "codeartifact:Get*",
            "codeartifact:Read*"
         ],
         "Resource": "arn:aws:codeartifact:*:*:repository/*/*test"
      },
      {
         "Effect": "Allow",
         "Action": [
            "codeartifact:List*",
            "codeartifact:Describe*"
         ],
         "Resource": "arn:aws:codeartifact:*:*:package/*/*test/*/*/*"
      },
      {
```

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

Limitar a duração do token de autorização

Os usuários devem se autenticar CodeArtifact com tokens de autorização para publicar ou consumir versões do pacote. Os tokens de autorização são válidos somente durante a vida útil configurada. Os tokens têm uma vida útil padrão de 12 horas. Para obter mais informações sobre a tokens de autorização, consulte AWS Autenticação e tokens do CodeArtifact.

Ao buscar um token, os usuários podem configurar a vida útil dele. Os valores válidos para a vida útil de um token de autorização são `0` e qualquer número entre `900` (15 minutos) e `43200` (12 horas). Um valor de `0` criará um token com uma duração igual às credenciais temporárias do perfil do usuário.

Os administradores podem limitar os valores válidos durante a vida útil de um token de autorização usando a chave de condição `sts:DurationSeconds` na política de permissões anexada ao usuário ou grupo. Se o usuário tentar criar um token de autorização com uma vida útil fora dos valores válidos, a criação do token falhará.

O exemplo de políticas a seguir limita as possíveis durações de um token de autorização criado pelos CodeArtifact usuários.

Exemplo de política: limite a vida útil do token a exatamente 12 horas (43.200 segundos)

Com essa política, os usuários só poderão criar tokens de autorização com uma vida útil de 12 horas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "codeartifact:*",
            "Resource": "*"
        },
        {
            "Sid": "sts",
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "NumericEquals": {
                    "sts:DurationSeconds": 43200
                },
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
                }
            }
        }
    ]
}
```

Exemplo de política: limite a vida útil do token entre 15 minutos e 1 hora ou igual ao período de credenciais temporárias do usuário

Com essa política, os usuários poderão criar tokens válidos entre 15 minutos e 1 hora. Os usuários também poderão criar um token que dure a duração das credenciais temporárias de do perfil especificando `0` para `--durationSeconds`.

JSON

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

```
        {
            "Effect": "Allow",
            "Action": "codeartifact:*",
            "Resource": "*"
        },
        {
            "Sid": "sts",
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*",
            "Condition": {
                "NumericLessThanEquals": {
                    "sts:DurationSeconds": 3600
                },
                "StringEquals": {
                    "sts:AWSServiceName": "codeartifact.amazonaws.com"
                }
            }
        }
    ]
}
```

Usando tags para controlar o acesso aos CodeArtifact recursos

As condições nas declarações de política de usuário do IAM fazem parte da sintaxe que você usa para especificar as permissões para os recursos exigidos pelas CodeArtifact ações. O uso de tags em condições é uma forma de controlar o acesso a recursos e solicitações. Para obter informações sobre a marcação de CodeArtifact recursos, consulteMarcando atributos. Este tópico discute o controle de acesso com base em tags.

Ao criar políticas do IAM, você pode definir permissões granulares concedendo acesso a recursos específicos. À medida que o número de recursos que você gerencia aumenta, essa tarefa se torna mais difícil. Atribuir etiquetas a recursos e usá-las em condições de declaração de política pode facilitar essa tarefa. Você concede acesso em massa a qualquer recurso utilizando determinada etiqueta. Depois, você a aplica repetidamente a recursos relevantes durante a criação ou posteriormente.

As tags podem ser anexadas ao recurso ou passadas na solicitação para serviços compatíveis com tags. Em CodeArtifact, os recursos podem ter tags e algumas ações podem incluir tags. Ao criar uma política do IAM, você poderá usar chaves de condição de tag para controlar:

- Quais usuários podem executar ações em um recurso de domínio ou repositório, com base nas tags que o recurso já tem.
- Quais tags podem ser transmitidas na solicitação de uma ação.
- Se chaves de tags específicas podem ser usadas em uma solicitação.

Para obter a sintaxe e a semântica completas das chaves de condição de tag, consulte Controlar o acesso usando tags no Guia do usuário do IAM.

> **Important**
>
> Ao usar tags em recursos para limitar ações, as tags devem estar no recurso no qual a ação opera. Por exemplo, para negar permissões `DescribeRepository` com tags, estas devem estar em cada repositório e não no domínio. Consulte AWS CodeArtifact referência de permissões para obter uma lista de ações CodeArtifact e em quais recursos elas operam.

Exemplos de controle de acesso baseado em tags

Os exemplos a seguir demonstram como especificar condições de tag em políticas para usuários do CodeArtifact .

Example 1: Limitar ações com base em tags na solicitação

A política de usuário `AWSCodeArtifactAdminAccess` gerenciado dá aos usuários permissão ilimitada para realizar qualquer CodeArtifact ação em qualquer recurso.

A política a seguir limita esse poder e nega a usuários não autorizados permissão para criar repositórios, a menos que a solicitação contenha determinadas tags. Para fazer isso, ela nega a ação `CreateRepository` se a solicitação não especificar uma tag chamada `costcenter` com um dos valores `1` ou `2`. O administrador de um cliente deve anexar essa política do IAM a usuários não autorizados do IAM, além da política de usuário gerenciada.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "codeartifact:CreateRepository",
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/costcenter": "true"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": "codeartifact:CreateRepository",
      "Resource": "*",
      "Condition": {
          "ForAnyValue:StringNotEquals": {
              "aws:RequestTag/costcenter": [
                  "1",
                  "2"
              ]
          }
      }
  }
  ]
}
```

Example 2: Limitar ações com base em tags de recursos

A política de usuário `AWSCodeArtifactAdminAccess` gerenciado dá aos usuários permissão ilimitada para realizar qualquer CodeArtifact ação em qualquer recurso.

A seguinte política limita esse poder e nega a usuários não autorizados permissão para realizar ações nos repositórios de domínios específicos. Para fazer isso, ela negará ações específicas se o recurso tiver uma tag denominada `Key1` com um dos valores `Value1` ou `Value2`. (A chave de condição `aws:ResourceTag` é usada para controlar o acesso a recursos com base nas tags

desses recursos.) O administrador de um cliente deve anexar essa política do IAM a usuários não autorizados do IAM, além da política de usuário gerenciada.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codeartifact:TagResource",
        "codeartifact:UntagResource",
        "codeartifact:DescribeDomain",
        "codeartifact:DescribeRepository",
        "codeartifact:PutDomainPermissionsPolicy",
        "codeartifact:PutRepositoryPermissionsPolicy",
        "codeartifact:ListRepositoriesInDomain",
        "codeartifact:UpdateRepository",
        "codeartifact:ReadFromRepository",
        "codeartifact:ListPackages",
        "codeartifact:ListTagsForResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Key1": ["Value1", "Value2"]
        }
      }
    }
  ]
}
```

Example 3: Permitir ações com base em tags de recursos

A política a seguir concede aos usuários permissão para realizar ações e obter informações sobre repositórios e pacotes em CodeArtifact.

Para fazer isso, ela permitirá ações específicas se o repositório tiver uma tag denominada `Key1` com o valor `Value1`. (A `aws:RequestTag` chave de condição é usada para controlar quais tags podem ser transmitidas em uma solicitação do IAM.) A `aws:TagKeys` condição garante que a chave

de tag faça diferenciação de letras maiúsculas e minúsculas. Essa política é útil para usuários do IAM que não têm a política de usuário gerenciada `AWSCodeArtifactAdminAccess` anexada. A política gerenciada dá aos usuários permissão ilimitada para realizar qualquer CodeArtifact ação em qualquer recurso.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:UpdateRepository",
        "codeartifact:DeleteRepository",
        "codeartifact:ListPackages"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

Example 4: Permitir ações com base em tags na solicitação

A política a seguir concede aos usuários permissão para criar repositórios em domínios especificados em. CodeArtifact

Para fazer isso, ela permitirá as ações `CreateRepository` e `TagResource` se a API de criar recurso na solicitação especificar uma tag denominada `Key1` com o valor `Value1`. (A `aws:RequestTag` chave de condição é usada para controlar quais tags podem ser transmitidas em uma solicitação do IAM.) A `aws:TagKeys` condição garante que a chave de tag faça diferenciação de letras maiúsculas e minúsculas. Essa política é útil para usuários do IAM que não têm a política de usuário gerenciada `AWSCodeArtifactAdminAccess` anexada. A política gerenciada dá aos usuários permissão ilimitada para realizar qualquer CodeArtifact ação em qualquer recurso.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartifact:CreateRepository",
        "codeartifact:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Key1": "Value1"
        }
      }
    }
  ]
}
```

AWS CodeArtifact referência de permissões

AWS CodeArtifact recursos e operações

Em AWS CodeArtifact, o recurso principal é um domínio. Em uma política, você usa um Amazon Resource Name (ARN) para identificar o recurso a que a política se aplica. Os repositórios também são recursos e estão ARNs associados a eles. Para obter mais informações, consulte Amazon Resource Names (ARNs) no Referência geral da Amazon Web Services.

Tipo de atributo	Formato ARN
Domínio	`arn:aws:codeartifact:` *`region-ID`* `:`*`account-ID`*`:domain/`*`my_domain`*
Repositório	`arn:aws:codeartifact:` *`region-ID`* `:`*`account-ID`*`:repository/` *`my_domain`* `/`*`my_repo`*

Tipo de atributo	Formato ARN
Grupo de pacotes	`arn:aws:codeartifact:` *region-ID* `:` *account-ID* `:package-group/` *my_domain* `/` *encoded_package_group_pattern*
Pacote com namespace	`arn:aws:codeartifact:` *region-ID* `:` *account-ID* `:package/` *my_domain* `/` *my_repo* `/` *package-format* `/` *namespace* `/` *my_package*
Pacote sem namespace	`arn:aws:codeartifact:` *region-ID* `:` *account-ID* `:package/` *my_domain* `/` *my_repo* `/` *package-format* `//` *my_package*
Todos os CodeArtifact recursos	`arn:aws:codeartifact:*`
Todos os CodeArtifact recursos pertencentes à conta especificada na região especificada da AWS	`arn:aws:codeartifact:` *region-ID* `:` *account-ID* `:*`

O ARN de recurso que você especifica depende a qual ação ou ações deseja controlar o acesso.

Você pode indicar um domínio específico (*myDomain*) em sua declaração usando seu ARN da seguinte forma.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain"
```

Você pode indicar um repositório específico (*myRepo*) em sua declaração usando seu ARN da seguinte forma.

```
"Resource": "arn:aws:codeartifact:us-east-2:123456789012:domain/myDomain/myRepo"
```

Para especificar vários recursos em uma única declaração, separe-os ARNs com vírgulas. A instrução a seguir se aplica a todos os pacotes e repositórios em um domínio 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**
>
> Muitos AWS serviços tratam dois pontos (:) ou uma barra invertida (/) como o mesmo caractere em. ARNs No entanto, CodeArtifact usa uma correspondência exata nos padrões e regras dos recursos. Certifique-se de usar os caracteres corretos ao criar padrões de evento, de modo que eles correspondam à sintaxe ARN no recurso.

AWS CodeArtifact Operações e permissões de API

Use a tabela a seguir como uma referência ao configurar o controle de acesso e escrever políticas de permissões que você pode anexar a uma identidade do IAM (políticas baseadas em identidade).

Você pode usar chaves AWS de condição abrangentes em suas AWS CodeArtifact políticas para expressar condições. Para obter uma lista, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Você especifica as ações no campo `Action` das políticas. Para especificar uma ação, use o prefixo `codeartifact:` seguido pelo nome da operação API (por exemplo, `codeartifact:CreateDomain` e `codeartifact:AssociateExternalConnection`). Para especificar várias ações em uma única declaração, separe-as com vírgulas (por exemplo, `"Action": [ "codeartifact:CreateDomain", "codeartifact:AssociateExternalConnection" ]`).

Usando caracteres curinga

Você especifica um ARN, com ou sem um caractere curinga (*), como o valor do recurso no campo `Resource` das políticas. Você pode usar um curinga para especificar várias ações ou recursos. Por exemplo, `codeartifact:*` especifica todas as CodeArtifact ações e `codeartifact:Describe*` especifica todas as CodeArtifact ações que começam com a palavra. `Describe`

Package group ARNs

> **Note**
>
> Esta seção sobre como a codificação de grupos ARNs e padrões de pacotes é informativa. É recomendável copiar ARNs do console ou buscar ARNs usando a `DescribePackageGroup` API em vez de codificar padrões e construir. ARNs

As políticas do IAM usam o caractere curinga, `*`, para corresponder a várias ações do IAM ou a vários recursos. Os padrões de grupos de pacotes também usam o caractere `*`. Para escrever com mais facilidade políticas do IAM que correspondam a um único grupo de pacotes, o formato ARN do grupo de pacotes usa uma versão codificada do padrão do grupo de pacotes.

Especificamente, o formato ARN do grupo de pacotes é o seguinte:

```
arn:aws:codeartifact:region:account-ID:package-
group/my_domain/encoded_package_group_pattern
```

Onde o padrão do grupo de pacotes codificado é o padrão do grupo de pacotes, com certos caracteres especiais substituídos por seus valores codificados em porcentagem. A seguinte lista contém os caracteres e seus valores correspondentes codificados em porcentagem:

- `*` : `%2a`
- `$` : `%24`
- `%` : `%25`

Por exemplo, o ARN de um grupo de pacotes raiz de um domínio, (`/*`), seria:

```
arn:aws:codeartifact:us-east-1:111122223333:package-group/my_domain/%2a
```

Observe que os caracteres não incluídos na lista não podem ser codificados e ARNs diferenciam maiúsculas de minúsculas, portanto, `*` devem ser codificados como ou não. `%2a` `%2A`

Solução de problemas AWS CodeArtifact de identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com CodeArtifact um IAM.

Tópicos

Não estou autorizado a realizar uma ação em CodeArtifact

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo *`my-example-widget`* fictício, mas não tem as permissões `codeartifact:`*`GetWidget`* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codeartifact:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso *`my-example-widget`* usando a ação `codeartifact:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha Conta da AWS acessem meus CodeArtifact recursos

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se é CodeArtifact compatível com esses recursos, consulteComo AWS CodeArtifact funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.

- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Trabalhar com endpoints da Amazon VPC

Você pode configurar CodeArtifact para usar uma interface de nuvem privada virtual (VPC) endpoint para melhorar a segurança da sua VPC.

Os VPC endpoints usam AWS PrivateLink, um serviço que possibilita o acesso por CodeArtifact APIs meio de endereços IP privados. AWS PrivateLink restringe todo o tráfego de rede entre sua VPC CodeArtifact e a rede da AWS. Ao usar um endpoint da VPC de interface, não é necessário utilizar um gateway da Internet, um dispositivo NAT ou um gateway privado virtual. Para obter mais informações, consulte Endpoints da VPC no Guia do usuário da Amazon Virtual Private Cloud.

> **Important**
>
> - Os VPC endpoints não oferecem suporte a solicitações entre regiões.AWS Certifique-se de criar seu endpoint na mesma AWS região para a qual planeja emitir suas chamadas de API. CodeArtifact
> - Os endpoints da VPC oferecem suporte somente a DNS fornecidos pela Amazon por meio do Amazon Route 53. Se quiser usar seu próprio DNS, poderá usar o encaminhamento de DNS condicional. Para obter mais informações, consulte Conjuntos de opções DHCP no Guia do usuário da Amazon Virtual Private Cloud.
> - O grupo de segurança anexado ao endpoint da VPC deve permitir conexões de entrada na porta 443 na sub-rede privada da VPC.

Tópicos

Crie endpoints VPC para CodeArtifact

Para criar endpoints de nuvem privada virtual (VPC) para CodeArtifact, use o comando Amazon EC2. `create-vpc-endpoint` AWS CLI Para obter mais informações, consulte Endpoints da VPC de interface (AWS PrivateLink) no Guia do usuário da Amazon Virtual Private Cloud.

São necessários dois VPC endpoints para que todas as solicitações CodeArtifact estejam na rede. AWS O primeiro endpoint é usado para chamar CodeArtifact APIs (por exemplo, `GetAuthorizationToken` e`CreateRepository`).

```
com.amazonaws.region.codeartifact.api
```

O segundo endpoint é usado para acessar CodeArtifact repositórios usando gerenciadores de pacotes e ferramentas de criação (por exemplo, npm e Gradle).

```
com.amazonaws.region.codeartifact.repositories
```

O comando a seguir cria um endpoint para acessar CodeArtifact repositórios.

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --vpc-endpoint-type Interface \
  --service-name com.amazonaws.region.codeartifact.api --subnet-ids subnetid \
  --security-group-ids groupid --private-dns-enabled
```

O comando a seguir cria um endpoint para acessar os gerenciadores de pacotes e as ferramentas de compilação.

```
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**
>
> Ao criar o endpoint `codeartifact.repositories`, você deve criar um nome de host DNS privado usando a opção `--private-dns-enabled`. Se você não puder ou não quiser criar um nome de host DNS privado ao criar o `codeartifact.repositories` endpoint, siga uma etapa extra de configuração para usar seu gerenciador de pacotes a partir de CodeArtifact uma VPC. Consulte Usar o endpoint `codeartifact.repositories` sem DNS privado para obter mais informações.

Depois de criar os VPC endpoints, talvez você precise fazer mais configurações com as regras do grupo de segurança para usar os endpoints. CodeArtifact Para obter mais informações sobre grupos de segurança da Amazon VPC, consulte Grupos de segurança.

Se você estiver tendo problemas para se conectar CodeArtifact, você pode usar a ferramenta VPC Reachability Analyzer para depurar o problema. Para mais informações, consulte O que é o VPC Reachability Analyzer?

Criar o endpoint de gateway do Amazon S3

CodeArtifact usa o Amazon Simple Storage Service (Amazon S3) para armazenar ativos de pacotes. Para extrair pacotes CodeArtifact, você deve criar um endpoint de gateway para o Amazon S3. Quando seu processo de compilação ou implantação baixa pacotes CodeArtifact, ele deve acessar CodeArtifact para obter os metadados do pacote e o Amazon S3 para baixar os ativos do pacote (por exemplo, arquivos `.jar` Maven).

> **Note**
>
> Não é necessário um endpoint do Amazon S3 ao usar os formatos de pacote Python ou Swift.

Para criar o endpoint do gateway Amazon S3 para CodeArtifact, use o comando Amazon EC2. `create-vpc-endpoint` AWS CLI Ao criar o endpoint, você deve selecionar as tabelas de rotas para a VPC. Para obter mais informações, consulte Endpoints da VPC de gateway no Guia do usuário da Amazon Virtual Private Cloud.

O comando a seguir cria um endpoint do Amazon S3.

```
aws ec2 create-vpc-endpoint --vpc-id vpcid --service-name com.amazonaws.region.s3 \
  --route-table-ids routetableid
```

Permissões mínimas de bucket do Amazon S3 para AWS CodeArtifact

O endpoint de gateway do Amazon S3 usa um documento de política do IAM para limitar o acesso ao serviço. Para permitir somente as permissões mínimas do bucket do Amazon S3 CodeArtifact, restrinja o acesso ao bucket CodeArtifact do Amazon S3 usado quando você cria o documento de política do IAM para o endpoint.

A tabela a seguir descreve os buckets do Amazon S3 que você deve referenciar em suas políticas para permitir o acesso CodeArtifact em cada região.

Região	ARN do bucket do 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

Você pode usar o `aws codeartifact describe-domain` comando para buscar o bucket do Amazon S3 usado por CodeArtifact um domínio.

```
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"
  }
}
```

Exemplo

O exemplo a seguir ilustra como fornecer acesso aos buckets do Amazon S3 necessários CodeArtifact para operações na região. `us-east-1` Para outras regiões, atualize a entrada `Resource` com o ARN de permissão correto para sua região com base na tabela acima.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::assets-193858265520-us-east-1/*"]
    }
  ]
}
```

Uso CodeArtifact a partir de uma VPC

Se você não puder ou não quiser habilitar o DNS privado no seu `com.amazonaws.region.codeartifact.repositories` VPC endpoint que você criouCrie endpoints VPC para CodeArtifact, você deve usar uma configuração diferente para o endpoint de repositórios usar a partir de uma VPC. CodeArtifact Siga as instruções em Usar o endpoint `codeartifact.repositories` sem DNS privado para configurar CodeArtifact se o

`com.amazonaws.region.codeartifact.repositories` endpoint não tem o DNS privado habilitado.

Usar o endpoint **`codeartifact.repositories`** sem DNS privado

Se você não puder ou não quiser habilitar o DNS privado no seu `com.amazonaws.region.codeartifact.repositories` VPC endpoint criado Crie endpoints VPC para CodeArtifact em, siga estas instruções para configurar seu gerenciador de pacotes com a URL correta. CodeArtifact

1. Execute o comando a seguir para localizar o endpoint da VPC a ser usado para substituir o nome do host.

   ```
   $ aws ec2 describe-vpc-endpoints --filters Name=service-
   name,Values=com.amazonaws.region.codeartifact.repositories \
     --query 'VpcEndpoints[*].DnsEntries[*].DnsName'
   ```

   A saída é semelhante à seguinte.

   ```
   [
     [
       "vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-west-2.vpce.amazonaws.com"
     ]
   ]
   ```

2. Atualize o caminho do VPC endpoint para incluir o formato do pacote, seu nome de CodeArtifact domínio e CodeArtifact nome do repositório. Veja o exemplo a seguir.

   ```
   https://vpce-0743fe535b883ffff-76ddffff.d.codeartifact.us-
   west-2.vpce.amazonaws.com/format/d/domain_name-domain_owner/repo_name
   ```

   Substitua os campos a seguir do endpoint de exemplo.

   - *`format`*: substitua por um formato de CodeArtifact pacote válido, por exemplo, `npm` ou`pypi`.
   - *`domain_name`*: substitua pelo CodeArtifact domínio que contém o CodeArtifact repositório que hospeda seus pacotes.
   - *`domain_owner`*: substitua pelo ID do proprietário do CodeArtifact domínio, por exemplo,`111122223333`.
   - *`repo_name`*: substitua pelo CodeArtifact repositório que hospeda seus pacotes.

O URL a seguir é um exemplo de endpoint do repositório do npm.

```
https://vpce-0dc4daf7fca331ed6-et36qa1d.d.codeartifact.us-
west-2.vpce.amazonaws.com/npm/d/domainName-111122223333/repoName
```

3. Configure o gerenciador de pacotes para usar o endpoint da VPC atualizado da etapa anterior. Você deve configurar o gerenciador de pacotes sem usar o CodeArtifact `login` comando. Para obter instruções de configuração para cada formato de pacote, consulte as documentações a seguir.

   - npm: Configuração do npm sem usar o comando login
   - nuget: configurar o nuget ou dotnet sem o comando login
   - pip: Configurar o pip sem o comando login
   - twine: Configure e use o twine com CodeArtifact
   - Gradle: Usar o CodeArtifact com o Gradle
   - mvn: Use CodeArtifact com mvn

Crie uma política de VPC endpoint para CodeArtifact

Para criar uma política de VPC endpoint para CodeArtifact, especifique o seguinte:

- A entidade principal que pode realizar ações.
- As ações que podem ser executadas.
- Os recursos que podem ter ações executadas neles.

O exemplo de política a seguir especifica que os diretores na conta 123456789012 podem chamar a API e buscar pacotes de um repositório. `GetAuthorizationToken` CodeArtifact

```
{
  "Statement": [
    {
      "Action": [
        "codeartifact:GetAuthorizationToken",
        "codeartifact:GetRepositoryEndpoint",
        "codeartifact:ReadFromRepository",
        "sts:GetServiceBearerToken"
```

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

Criação de recursos do CodeArtifact com AWS CloudFormation

O CodeArtifact é integrado ao AWS CloudFormation, um serviço que ajuda você a modelar e configurar seus recursos da AWS, para que você possa passar menos tempo criando e gerenciando seus recursos e sua infraestrutura. Você cria um modelo que descreve todos os recursos AWS que deseja, e o CloudFormation se encarrega de provisionar e configurar esses recursos para você.

Quando você usa o CloudFormation, é possível reutilizar seu modelo para configurar seus recursos do CodeArtifact repetidamente e de forma consistente. Basta descrever seus recursos uma vez e, depois, provisionar os mesmos recursos repetidamente em várias contas e regiões da AWS.

CodeArtifact e modelos CloudFormation

Para provisionar e configurar recursos para o CodeArtifact e serviços relacionados, você deve entender os modelos do CloudFormation. Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os atributos que você deseja provisionar nas suas pilhas CloudFormation. Se não estiver familiarizado com o JSON ou o YAML, é possível usar o CloudFormation Designer para ajudar a começar a usar os modelos do CloudFormation. Para mais informações, consulte O que é o AWS CloudFormation Designer? no Guia do usuário do AWS CloudFormation.

O CodeArtifact é compatível com a criação de domínios, repositórios e grupos de pacotes no CloudFormation. Para obter mais informações, como exemplos de modelos JSON e YAML, consulte os seguintes tópicos no Guia do usuário do CloudFormation:

- AWS::CodeArtifact::Domain
- AWS::CodeArtifact::Repository
- AWS::CodeArtifact::PackageGroup

Impedir a exclusão de recursos do CodeArtifact

Os repositórios do CodeArtifact contêm dependências críticas de aplicações que podem não ser fáceis de recriar se perdidas. Para proteger os recursos do CodeArtifact contra a exclusão acidental ao gerenciar recursos do CodeArtifact com o CloudFormation, inclua os atributos `DeletionPolicy`

e `UpdateRetainPolicy` com um valor de `Retain` em todos os domínios e repositórios. Isso evitará a exclusão se o recurso for removido do modelo de pilha ou se a pilha inteira for excluída acidentalmente. O trecho do YAML a seguir mostra um domínio e um repositório básicos com estes 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
```

Consulte mais informações sobre esses atributos em DeletionPolicy e UpdateReplacePolicy no Guia do usuário do AWS CloudFormation.

Saiba mais sobre o CloudFormation

Para saber mais sobre o CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- AWS CloudFormation Guia do usuário do
- AWS CloudFormation Guia do Usuário da Interface de Linha de Comando

Solução de problemas do AWS CodeArtifact

As informações a seguir podem ajudar a solucionar problemas comuns com o CodeArtifact.

Para obter informações de solução de problemas específicas ao formato, consulte os seguintes tópicos:

- Solução de problemas do Maven
- Solução de problemas do Swift

Não consigo visualizar as notificações

Problema: quando você está no console do Developer Tools e escolhe Notifications(Notificações) em Settings (Configurações), um erro de permissões é exibido.

Possíveis correções: como as notificações são um atributo do console de Ferramentas do Desenvolvedor, o CodeArtifact não é compatível com elas no momento. Nenhuma das políticas gerenciadas do CodeArtifact inclui permissões que permitem aos usuários visualizar ou gerenciar notificações. Se você usa outros serviços no console de Ferramentas do Desenvolvedor e esses serviços oferecem suporte a notificações, as políticas gerenciadas desses serviços incluem as permissões necessárias para visualizar e gerenciar notificações desses serviços.

Marcando atributos

Uma tag é um rótulo de atributo personalizado que você ou a AWS atribui a um recurso da AWS. Cada tag da AWS tem duas partes:

- Uma chave de tag (por exemplo `CostCenter`, `Environment`, `Project` ou `Secret`). Chaves de tag fazem distinção entre maiúsculas e minúsculas.
- Um campo opcional conhecido como um valor de tag (por exemplo, `111122223333`, `Production` ou um nome de equipe). Omitir o valor da tag é o mesmo que usar uma string vazia. Como chaves de tag, os valores das tags diferenciam maiúsculas de minúsculas.

Juntos, esses são conhecidos como pares de chave-valor.

As tags ajudam a identificar e organizar os recursos da AWS. Muitos serviços da AWS oferecem suporte à marcação para que você possa atribuir a mesma tag a recursos de diferentes serviços para indicar que os recursos estão relacionados. Por exemplo, você pode atribuir a mesma tag a um repositório do que você atribui a um projeto do AWS CodeBuild.

Consulte dicas e práticas recomendadas para o uso de tags no whitepaper Práticas recomendadas para marcação de recursos da AWS.

É possível marcar os seguintes tipos de recursos no CodeArtifact:

- Marcar um repositório em CodeArtifact
- Marcar um domínio em CodeArtifact

Você pode usar o console, AWS CLI, as APIs do CodeArtifact ou os SDKs AWS para:

- Adicione tags a um domínio ou repositório ao criá-lo*.
- Adicione, gerencie e remova tags de um domínio ou repositório.

* Você não pode adicionar tags a um domínio ou repositório ao criá-lo no console.

Além de identificar, organizar e monitorar seus recursos com etiquetas, você pode usar etiquetas em políticas do IAM para ajudar a controlar quem pode visualizar e interagir com o seu recurso. Para obter exemplos de políticas de acesso baseadas em tags, consulte Usando tags para controlar o acesso aos CodeArtifact recursos.

Alocação de custos do CodeArtifact com tags

É possível usar tags para alocar custos de armazenamento e solicitação no CodeArtifact.

Alocação de custos de armazenamento de dados no CodeArtifact

Os custos de armazenamento de dados estão vinculados aos domínios, portanto, para alocar seus custos de armazenamento do CodeArtifact, você pode usar qualquer tag aplicada aos seus domínios. Para obter informações sobre como adicionar tags a domínios, consulte Marcar um domínio em CodeArtifact.

Alocação de custos de solicitação no CodeArtifact

A maior parte do uso de solicitações está vinculada aos repositórios, portanto, para alocar seus custos de solicitações do CodeArtifact, você pode usar qualquer tag aplicada aos seus repositórios. Para obter informações sobre como adicionar tags a repositórios, consulte Marcar um repositório em CodeArtifact.

Alguns tipos de solicitação estão associados a domínios em vez de repositórios, portanto, o uso da solicitação e os custos relacionados às solicitações serão alocados às tags no domínio. A melhor maneira de determinar se um tipo de solicitação está associado a um domínio ou repositório é usar as Ações definidas pela tabela AWS do CodeArtifact na Referência de Autorização de Serviço. Encontre o tipo de solicitação na coluna Ações e veja o valor na coluna Tipos de recursos correspondente. Se o tipo de recurso for domínio, as solicitações desse tipo serão cobradas do domínio. Se o tipo de recurso for repositório ou pacote, as solicitações desse tipo serão cobradas no repositório. Algumas ações mostram os dois tipos de recursos. Para essas ações, o recurso faturado depende do valor passado na solicitação.

Cotas em AWS CodeArtifact

A tabela a seguir descreve as cotas de recursos em CodeArtifact. Para ver as cotas de recursos junto com a lista de endpoints de serviço CodeArtifact, consulte cotas AWS de serviço no. Referência geral da Amazon Web Services

Você pode solicitar um aumento da cota de serviço para as seguintes cotas CodeArtifact de recursos. Para obter mais informações sobre como solicitar um aumento de Service Quotas, consulte AWS Service Quotas.

Nome	Padrão	Ajustá	Description
Tamanho do arquivo do ativo	Cada região com suporte: 5 gigabytes	Sim	O tamanho máximo do arquivo por ativo.
Ativos por versão de pacote	Cada região compatível: 150	Não	Número máximo de ativos por versão do pacote.
CopyPackageVersions solicitações por segundo	Cada região compatível: 5	Sim	O número máximo de chamadas que podem ser feitas CopyPacka geVersions por segundo.
Transmissões diretas por repositório	Cada região com suporte: 10	Não	Número máximo de repositórios upstream diretos permitidos para um repositório.
Domínios por conta AWS	Cada região com suporte: 10	Sim	O número máximo de domínios que podem ser criados por AWS conta.
GetAuthorizationToken solicitações por segundo	Cada região compatível: 40	Sim	O número máximo de tokens de autorizaç ão recuperados por segundo.

Nome	Padrão	Ajustá	Description
GetPackageVersionAsset solicitações por segundo	Cada região compatível: 50	Sim	O número máximo de chamadas que podem ser feitas GetPackageVersionAsset por segundo.
ListPackageVersionAssets solicitações por segundo	Cada região compatível: 200	Sim	O número máximo de chamadas que podem ser feitas ListPackageVersionAssets por segundo.
ListPackageVersions solicitações por segundo	Cada região compatível: 200	Sim	O número máximo de chamadas que podem ser feitas ListPackageVersions por segundo.
ListPackages solicitações por segundo	Cada região compatível: 200	Sim	O número máximo de chamadas que podem ser feitas ListPackages por segundo.
PublishPackageVersion solicitações por segundo	Cada região com suporte: 10	Sim	O número máximo de chamadas que podem ser feitas PublishPackageVersion por segundo.
Leia solicitações por segundo de uma única AWS conta	Cada região compatível: 800	Sim	O número máximo de solicitações de leitura de uma AWS conta por segundo.
Repositórios por domínio	Cada região com suporte: 1.000	Sim	O número máximo de repositórios que pode ser criado por domínio.

Nome	Padrão	Ajustá	Description
Solicitações por segundo usando um único token de autenticação	Cada região compatível: 1.200	Não	O número máximo de solicitações por segundo usando um único token de autenticação.
Solicitações sem token de autenticação por endereço IP	Cada região compatível: 600	Não	O número máximo de solicitações por segundo sem um token de autenticação de um único endereço IP.
Repositórios upstream pesquisados	Cada região compatível: 25	Não	O número máximo de repositórios upstream pesquisados ao resolver um pacote.
Escreva solicitações por segundo a partir de uma única AWS conta	Cada região compatível: 100	Sim	O número máximo de solicitações de gravação de uma AWS conta por segundo.

Note

Em geral, cada solicitação de leitura feita para CodeArtifact conta como uma solicitação contabilizada em uma cota. No entanto, para o formato de pacote Ruby, uma única solicitação de leitura para a operação `/api/v1/dependencies` pode solicitar dados sobre vários pacotes.

Por exemplo, a solicitação pode ser semelhante a `https://${CODEARTIFACT_REPO_ENDPOINT}/api/v1/dependencies?gems=gem1,gem2.gem3`. Nesse exemplo, a solicitação conta como três solicitações em relação à cota.

Observe que as várias solicitações se aplicam somente às cotas de serviço, não ao faturamento. No exemplo, haverá cobrança somente por uma solicitação, embora ela conte como três solicitações para a cota de serviço. Para CI/CD ambientes que executam várias `bundle install` operações simultâneas, a taxa efetiva de solicitações pode

ser significativamente maior do que a contagem de solicitações HTTP. Se você enfrentar limitação durante a resolução de gemas em Ruby, solicite um aumento de cota para solicitações de leitura por segundo de uma única conta. AWS

AWS CodeArtifact histórico do documento do guia do usuário

A tabela a seguir descreve mudanças importantes na documentação do CodeArtifact.

Alteração	Descrição	Data
Documentação adicionada para configurar e usar o Cargo com CodeArtifact	CodeArtifact agora suporta caixas de carga. Documentação adicionada com orientação sobre como configurar o Cargo para usar CodeArtifact repositórios. Para obter mais informações, consulte Usar o CodeArtifact com Cargo.	20 de junho de 2024
Documentação adicionada para configurar e usar o Ruby com CodeArtifact	CodeArtifact agora suporta gemas Ruby. Documentação adicionada com orientação sobre como configurar gerenciadores de pacotes Ruby para usar CodeArtifact repositórios. Para obter mais informações, consulte Usando CodeArtifact com Ruby.	30 de abril de 2024
Foi adicionado um exemplo de política de chaves para criar domínios com uma chave gerenciada pelo AWS KMS cliente	Foi adicionado um exemplo de política de chaves que pode ser usado para criar uma chave KMS gerenciada pelo cliente para criptografar ativos em CodeArtifact domínios. Para obter mais informações, consulte Exemplo AWS KMS política chave.	18 de abril de 2024

Adição de documentação para oferecer suporte ao lançamento de grupos de pacotes.	Foi adicionada documentação sobre como gerenciar e usar grupos de pacotes no CodeArtifact. Para obter mais informações, consulte Trabalhar com grupos de pacotes no CodeArtifact.	21 de março de 2024
Adição de mais gerenciadores de pacotes válidos à documentação sobre o comando aws codeartifact login.	Foi adicionado `dotnet`, `nuget` e `swift` à lista de gerenciadores de pacotes válidos para usar com o comando `aws codeartifact login` . Para obter mais informações, consulte AWS Autenticação e tokens do CodeArtifact.	18 de fevereiro de 2024
Adição de uma entrada à documentação de solução de problemas do Swift sobre o Xcode travando em máquinas de CI	Foram adicionadas informações, incluindo uma solução, sobre um problema que pode fazer com que o Xcode trave em máquinas de CI devido à solicitação de senha do Keychain. Para obter mais informações, consulte O Xcode trava na máquina de CI devido à solicitação de senha do Keychain.	6 de fevereiro de 2024

Informações adicionadas sobre como solucionar problemas de lentidão do tempo de instalação de pacotes npm com o npm 8.x ou posterior	Foram adicionadas informações sobre como contornar os lentos tempos de instalação do pacote npm CodeArtifact, o que pode causar tempos de compilação lentos. Para obter mais informações, consulte Solução de problemas de instalações lentas com npm 8.x ou posterior.	29 de dezembro de 2023
Informações atualizadas sobre o comportamento dos ativos e metadados do pacote Python no CodeArtifact	Informações atualizadas sobre como CodeArtifact os repositórios retêm e atualizam os ativos e metadados da versão do pacote Python. Para obter mais informações, consulte Solicitar pacotes Python de upstreams e conexões externas.	14 de dezembro de 2023
Documentação reorganizada sobre monitoramento CodeArtifact	Informações reorganizadas sobre CodeArtifact eventos de monitoramento e informações adicionadas sobre a visualização de CodeArtifact solicitações com CloudWatch métricas da Amazon. Para obter mais informações, consulte Monitoramento CodeArtifact.	14 de dezembro de 2023

Foram adicionadas mais informações sobre o gerenciamento de CodeArtifact recursos com CloudFormation	Foram adicionadas referências e links para a documentação sobre o gerenciamento de CodeArtifact recursos com CloudFormation, incluindo uma seção sobre como impedir a exclusão de CodeArtifact recursos gerenciados com CloudFormation. Para obter mais informações, consulte Impedir a exclusão de recursos do CodeArtifact.	7 de dezembro de 2023
Foi adicionada documentação detalhando CodeArtifact o suporte do AWS KMS External Key Stores (XKS)	Foi adicionada uma seção com informações sobre CodeArtifact o suporte às chaves KMS, incluindo o uso de chaves XKS com. CodeArtifact Para obter mais informações, consulte Tipos de AWS KMS teclas suportadas em CodeArtifact.	31 de outubro de 2023
Atualização da documentação existente e adição de uma nova documentação de solução de problemas	Adição de um tópico de solução de problemas do Maven e inclusão de links para a documentação de solução de problemas do Swift e do Maven no tópico geral de solução de problemas. Para obter mais informações, consulte Solução de problemas do AWS CodeArtifact.	28 de setembro de 2023

Atualização da documentação para incluir o comando publish do Swift Package Manager	O Swift 5.9 introduziu um comando `swift package-registry publish` para criar e publicar um pacote Swift em um repositório de pacotes. Atualização da documentação do Swift para incluir instruções para usar esse comando. Para obter mais informações, consulte Usando CodeArtifact com Swift.	25 de setembro de 2023
Documentação adicionada para configuração CodeArtifact com Swift	CodeArtifact agora suporta pacotes Swift. Foi adicionada documentação com orientações sobre como configurar o Swift para usar CodeArtifact repositórios. Para obter mais informações, consulte Usando CodeArtifact com Swift.	20 de setembro de 2023
Orientação adicional sobre como CodeArtifact lidar com versões retiradas do pacote Python	Foi adicionada documentação com informações sobre como saber se uma versão de pacote Python foi retirada, como CodeArtifact lidar com versões de pacotes retiradas e respostas a perguntas comuns. Para obter mais informações, consulte Versões de pacotes retirados.	2 de agosto de 2023

Correção do comando da linha de comando na documentação do Yarn	Corrigido um comando de linha de comando incorreto que busca um token de CodeArtifact autorização e o armazena em uma variável de ambiente na documentação do Yarn.	20 de julho de 2023
Pequenas adições e correções de bugs na documentação do Python	Adição de informações de pip e twine em suas respectivas documentações e correção do que acontecia ao usar o comando `codeartifact login` com twine. Para obter mais informações, consulte Configure e use o pip com CodeArtifact e Configure e use o twine com CodeArtifact.	14 de julho de 2023
Comandos dotnet incorretos corrigidos na documentação CodeBuild	Correção dos comandos `dotnet add package` na documentação do Usando NuGet pacotes em CodeBuild.	13 de julho de 2023
AWS Identity and Access Management Documentação AWS CodeArtifact e atualização	Revisou o IAM na CodeArtifact documentação para adicionar clareza e consistência à documentação de outros serviços. AWS Consulte Identity and Access Management para AWS CodeArtifact.	24 de maio de 2023

Adição de informações sobre as versões do pacote Python retiradas	Foram adicionadas informaçõ es sobre como CodeArtifact retém os metadados retirados da versão do pacote Python. Para obter mais informações, consulte. Versões de pacotes retirados	11 de abril de 2023
Adição de informações sobre o suporte ao Clojure	Adição de informações sobre o suporte ao Clojure, incluindo o gerenciamento de dependências para projetos do Clojure. Para obter mais informações, consulte Usar o CodeArtifact com deps.edn.	21 de março de 2023
Adição de informações sobre a publicação de pacotes genéricos	Adição de informações sobre pacotes genéricos e como publicar e baixar o conteúdo do pacote com o AWS CLI. Para obter mais informações, consulte Usando CodeArtif act com pacotes genéricos , Publicando e consumind o pacotes genéricos e Comandos compatíveis com pacotes genéricos.	10 de março de 2023
Adição de informações sobre limites de tamanho de ativos para publicação	Adição de uma seção à publicação de pacote, para explicar os limites de tamanho para publicação de ativos.	21 de junho de 2022

Refatoramento da documentação da conexão externa	Moveu a documentação da conexão externa e a reorganizou para focar no objetivo final do usuário, que é conectar seu CodeArtifact repositório a repositórios de pacotes públicos. Também foram adicionadas mais orientações e informações sobre os diferentes métodos para alcançar essa meta. Para obter mais informações, consulte Conectar um CodeArtifact repositório a um repositório público.	9 de maio de 2022
Atualizou as informações do CodeArtifact evento para Amazon CloudWatch Events	Adição de mais informações ao campo `account` e adição do campo `repositoryAdministrator` . Para obter mais informações, consulte CodeArtifact formato e exemplo de evento.	7 de março de 2022
Instruções de configuração adicionadas para uso CodeArtifact em uma VPC sem DNS privado	Se você não puder ou não quiser habilitar o DNS privado em seu `codeartifact.repositories` VPC endpoint, você deve usar uma configuração diferente para o endpoint de repositórios usar a partir de uma VPC. CodeArtifact Consulte Usar o endpoint `codeartifact.repositories` sem DNS privado para obter mais informações.	8 de fevereiro de 2022

Adição da documentação detalhada para atualizar o status das versões do pacote	Expansão da documentação de status da versão do pacote de atualização em seu próprio tópico. Foi adicionada documentação para atualizar o status da versão de um pacote, incluindo as permissões necessárias do IAM, exemplos de AWS CLI comandos para vários cenários e possíveis erros. Consulte Atualizar o status da versão do pacote para obter mais informações.	1º de setembro de 2021
Atualização da documentação das versões do pacote de cópias com informações de permissões mais detalhadas	Foram adicionadas mais informações sobre as permissões necessárias de IAM e políticas baseadas em recursos para chamar o `aws codeartifact copy-package-versions` comando para copiar versões de pacotes de um repositório para outro dentro do mesmo domínio em. CodeArtifact Junto com as informações adicionais, agora há exemplos das políticas baseadas em recursos exigidas para o repositório de origem e destino. Consulte Permissões obrigatórias do IAM para copiar pacotes para obter mais informações.	25 de agosto de 2021

Atualização da documentação para executar uma compilação do Gradle no IntelliJ IDEA	Atualizou a documentação para executar uma compilação do Gradle no IntelliJ IDEA com etapas para configurar o Gradle para buscar plug-ins. CodeArtifact Também foi adicionada uma opção para criar um novo token de CodeArtifact autorização para cada nova execução com uma chamada embutida para`aws codeartifact get-authorization-token` . Consulte Executar uma compilação do Gradle no IntelliJ IDEA para obter mais informações.	23 de agosto de 2021
Documentação adicionada para configurar e usar o Yarn com AWS CodeArtifact	Foi adicionada documentação para configurar e usar o Yarn 1.X e o Yarn 2.X para gerenciar pacotes npm com o. CodeArtifact Consulte Configurar e usar o Yarn com o CodeArtifact para obter mais informações.	30 de julho de 2021

AWS CodeArtifact agora suporta NuGet pacotes	CodeArtifact os usuários agora podem publicar e consumir NuGet pacotes. Foi adicionada documentação para configura r e usar o Visual Studio e as ferramentas de linha de NuGet comando, como `nuget` e `dotnet` com CodeArtifact repositórios. Consulte Usando o CodeArtifact com NuGet para obter mais informações.	19 de novembro de 2020
Marcando recursos em AWS CodeArtifact	Foi adicionada documenta ção sobre a marcação de repositórios e domínios no. AWS CodeArtifact Consulte Marcando atributos.	30 de outubro de 2020
CodeArtifact agora suporta CloudFormation	CodeArtifact os usuários agora podem usar CloudForm ation modelos para criar CodeArtifact repositórios e domínios. Para obter mais informações e começar a usar, consulte Criação de recursos do CodeArtifact com AWS CloudFormation.	8 de outubro de 2020

Adicione informações sobre a criação de endpoints do gateway Amazon S3 para usar com CodeArtifact o Amazon VPC	Foram adicionadas informações sobre a criação de endpoints do gateway Amazon S3 com o comando Amazon EC2. AWS CLI Essa documentação também contém informações sobre as permissões específicas que CodeArtifact precisam ser usadas com ambientes Amazon VPC. Consulte Criar o endpoint de gateway do Amazon S3.	12 de agosto de 2020
Publicação de artefatos do Maven com curl e publicação de artefatos Maven de terceiros	Adição de orientação para Publicar com curl e Publicar artefatos de terceiros.	10 de agosto de 2020
Versão de disponibilidade geral (GA)	Versão inicial do Guia CodeArtifact do usuário.	10 de junho de 2020

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.