制品库

快速入门

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2023 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

快速入门
基础操作
Generic 制品库
Docker 制品库
Maven 制品库
npm 制品库
Helm 制品库
PyPI 制品库
Composer 制品库
NuGet 制品库
rpm 制品库
Conan 制品库
Cocoapods 制品库
Go 制品库

版权所有：腾讯云计算（北京）有限责任公司

快速入门
基础操作

最近更新时间：2022-12-28 16:42:50

本文档主要介绍如何快速使用 CODING 制品库。

前提条件

使用 CODING 制品库的前提是，您的腾讯云账号需要开通 CODING DevOps 服务，详情请参见 开通服务。

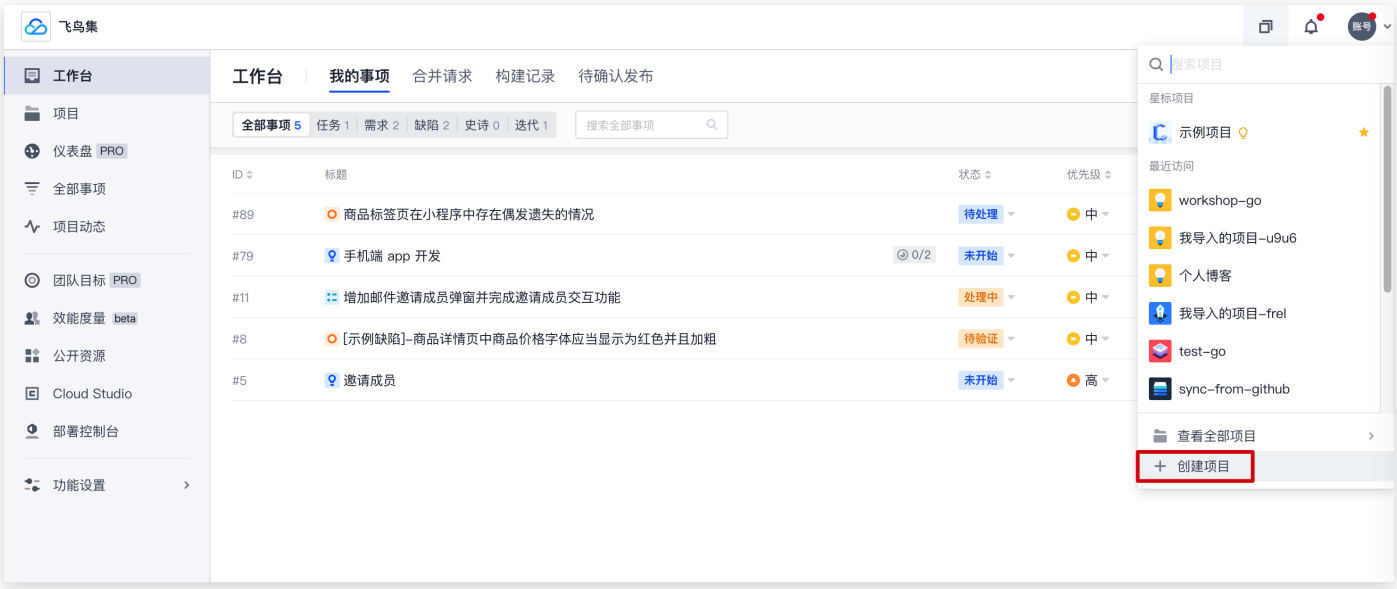
进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

在正式进行制品库的操作前，您可以参考以下内容进行功能初始化。下述的步骤与准备并非必须选项，可以按照实际需求有选择性地阅读。

创建项目

使用制品仓库前需新建项目，选择**DevOps 项目**。



填写名称等关键信息，创建完成后可在左侧菜单中进入制品仓库功能。若功能被隐藏，则需单击左下角的**项目设置 > 功能开关**开启制品库按钮。

新建制品仓库

进入制品仓库后选择仓库类型。

- 提供一个仓库名称
- 权限范围：决定当前创建的制品仓库对不同类型角色的操作权限，默认将对当前项目成员开放**推送**和**拉取**操作。
- 准备就绪，单击**确认新建**。

创建制品库后，您可以参见各个制品类型的快速开始进行制品推拉操作。

- Generic 制品库
- Docker 制品库
- Maven 制品库
- npm 制品库
- Helm 制品库
- PyPI 制品库
- Composer 制品库
- NuGet 制品库

- rpm 制品库
- Conan 制品库
- Cocoapods 制品库

Generic 制品库

最近更新时间：2022-12-28 16:42:55

该文档介绍如何将通用文件类型的制品存储在 CODING 制品库中。其内容包括创建制品库、推送、拉取和删除制品。

进入制品库功能页

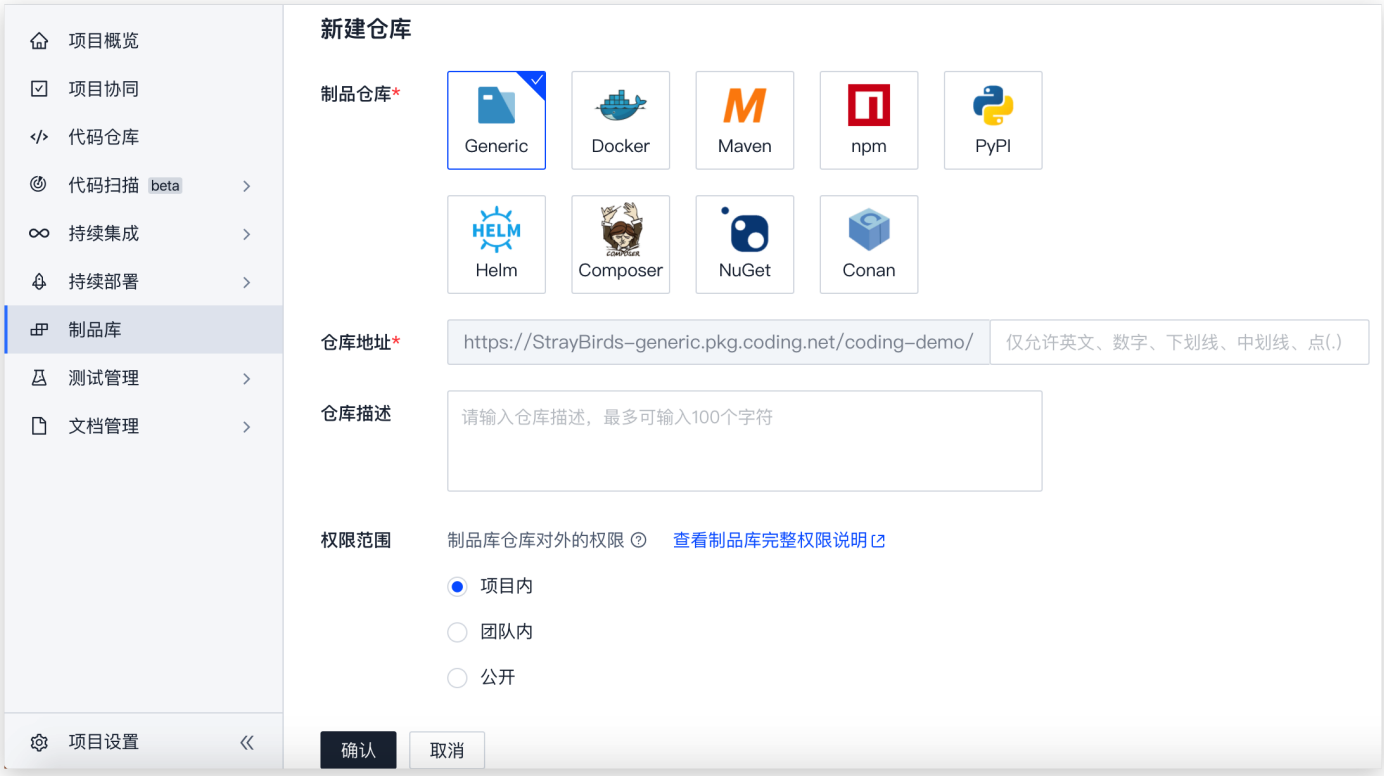
1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

创建制品仓库

进入项目，在左侧栏选择**制品库**，再单击**创建仓库**。

- 仓库类型选中 Generic File
- 提供一个仓库名称
- 提供一段仓库描述（非必填）
- 决定当前创建的制品仓库对不同类型角色的操作权限，默认将对当前项目成员开放**推送**和**拉取**操作。
- 准备就绪，单击**确认新建**

版权所有：腾讯云计算（北京）有限责任公司



推送 Generic 制品

Generic 类型制品库支持两种方式推送制品：

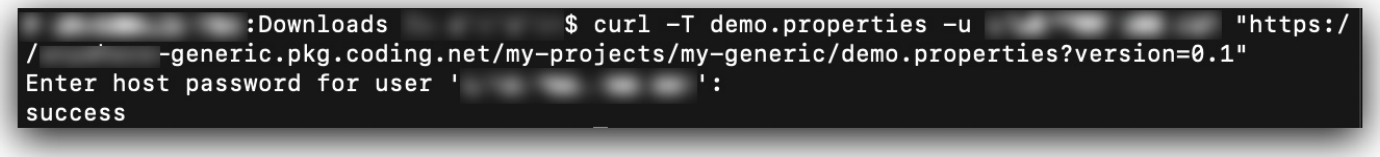
- 命令行方式
- 通过页面直接上传

通过命令行上传

参考**指引**当中的命令行进行推送：

以推送一个文件 demo.properties 为例。

```
curl -T demo.properties -u username "https://********-generic.pkg.coding.net/my-projects/my-generic/demo.properties?version=0.1"
```



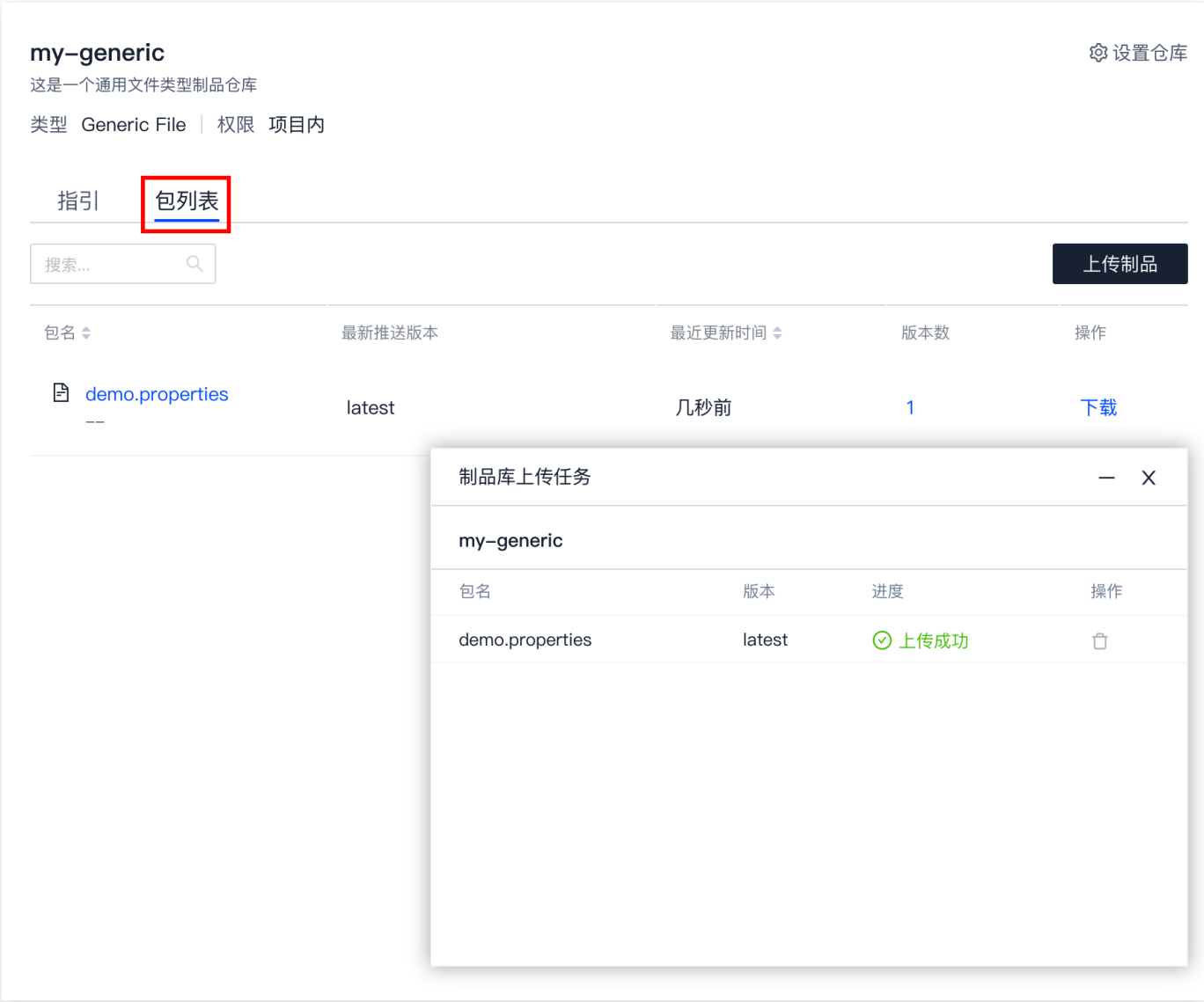
通过页面直接上传

在制品仓库页面，单击**直接上传**或者将文件拖拽至当前页面。

选择一个文件、勾选上传内容的打包形式、填写版本、确认包名，单击**开始上传**。

查看 Generic 制品

上传制品成功后，页面右下角会有弹窗提示任务成功。单击**包列表**即可查看到上传的包信息。



拉取 Generic 制品

Generic 类型制品库支持两种方式拉取制品：

- 命令行方式
- 通过页面直接下载

通过命令行拉取

参考指引当中的命令行进行拉取：

```
curl -L -u username "https://********-generic.pkg.coding.net/my-projects/my-generic/demo.properties?version=0.1" -o demo.properties
```

通过页面直接下载

在仓库页面，在**包列表**中单击包名，在右侧栏中单击**版本列表**，下载您需要的版本即可。

删除 Generic 制品

需通过命令行删除 Generic 制品。

删除

```
curl -X DELETE -u [redacted]@qq.com "https://[redacted]-generic.pkg.coding.net/new-demo/my-generic/<PACKAGE>?v
```

- `<VERSION>` 为**非必填**项，默认为 **latest**。

参考指引页面中给出的删除指令即可。

```
curl -X DELETE -u username "https://*******-generic.pkg.coding.net/my-projects/my-generic/demo.properties?version=0.1"
```

```
[redacted]:Downloads [redacted] $ curl -X DELETE -u [redacted] "https://[redacted]
-generic.pkg.coding.net/my-projects/my-generic/demo.properties?version=0.1"
Enter host password for user '[redacted]':
success
```

Docker 制品库

最近更新时间：2022-12-28 16:42:59

该文档介绍如何将 Docker 镜像存储在 CODING 制品库中，方便团队在项目进行统一的制品管理与版本控制。下文包含如何进行镜像制作、认证配置与制品推拉。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 Docker。
> - 参见 基础操作 创建制品仓库。
> - 制品仓库选择 Docker 类型。

制作镜像（可选阅读）

本章节提供两种方法快速创建一个 Demo Docker 镜像，若已熟悉 Docker 镜像制作可以跳过本节。

方法一：本地制作镜像

1. 在本地任意目录创建文件，名称为 Dockerfile，并写入以下内容：

```
FROM coding-public-docker.pkg.coding.net/public/docker/nodejs:12
```

2. 在所在目录中调出终端，运行命令构建镜像。

```
docker build -t hello-world .
```

镜像制作成功，tag 默认为 hello-world:latest。您可以参见 Docker 官方手册 自定义 tag 内容，格式为<镜像名>:<版本>。

```
/Volumes/CODING/Docker-learning docker build -t hello-world .
Sending build context to Docker daemon  2.048kB
Step 1/1 : FROM fanvinga/docker-ssrmu
 ---> 90ec15c0d38d
Successfully built 90ec15c0d38d
Successfully tagged hello-world:latest
```

方法二：从 Docker Hub 拉取镜像

1. 在终端中直接执行命令拉取镜像。

```
docker pull hello-world
```

2. 执行命令，查看已拉取的镜像。

```
docker images
```

配置认证信息

当您已在本地完成制品编译后，就可以将制品推送至远端制品仓库。推送之前需在本地配置远端仓库的认证信息。

访问令牌

推荐使用**访问令牌**生成认证的配置信息。

1. 在仓库页面单击**操作指引**。

2. 输入账号的登录密码 / 两步验证码后确认，复制生成的命令。

3. 在本地 Docker 环境中的命令行中粘贴生成后的命令并执行，即可完成认证。

```
➜  ~ docker login -u [redacted] -p [redacted] artifacts-docker.pkg.coding.net

WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
➜  ~
```

推送镜像

以下命令行仅作示例，命令行会因项目差异而发生改变，请复制项目内制品仓库中直接生成的命令。

1. 给 上文 拉取至本地的 hello-world 镜像打标签。

```
docker tag hello-world straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world
```

2. 推送您的 docker 镜像至制品仓库。

```
docker push straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world
```

成功推送后将看到如下内容：

```
/Volumes/CODING/Docker-learning docker push straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world
The push refers to repository [straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world]
cc471758abdf: Pushed
a25d159becc3: Pushed
06cfd7503045: Pushed
beee9f30bc1f: Pushed
latest: digest: sha256:[redacted] size: [redacted]
```

上述操作命令，均会直接显示在操作指引中，可输入替换值后复制命令。

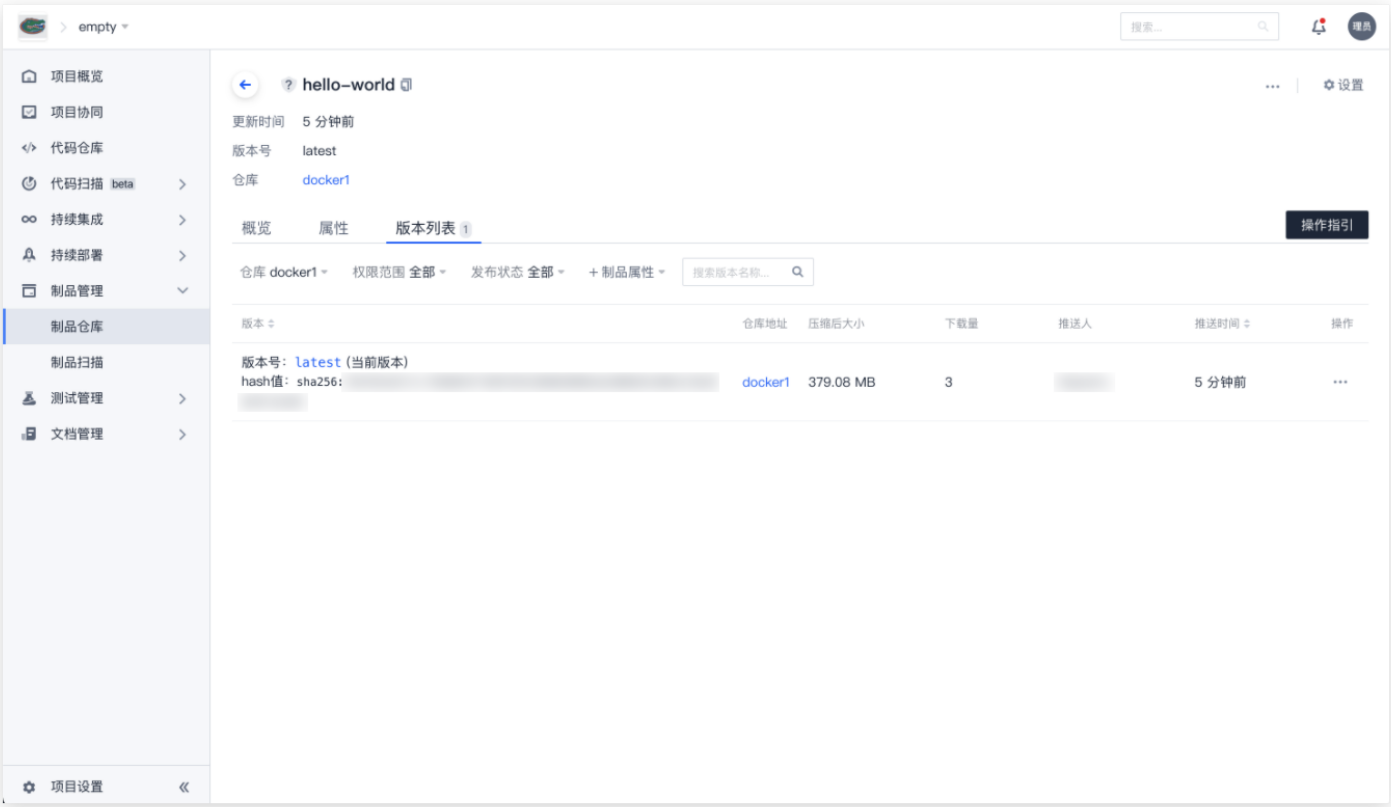
> **注意：**
>
> 更多自动化推送 Docker 镜像的操作，请参见 自动化插件。

查看镜像

推送完毕后，左侧菜单处的**项目概览**会在项目内广播推送动态。

项目的制品列表中，可以看到推送的 **hello-world** 镜像。

单击镜像名，可以在右侧栏查看到该包的完整信息，内含概览、指引、属性、版本列表等信息。



拉取镜像

使用 docker pull 命令可以拉取在 CODING 制品库中托管的 Docker 镜像。指引页面会自动生成相对应的拉取命令。

成功拉取后将会看到如下内容。

Maven 制品库

最近更新时间：2023-08-09 11:12:23

该文档介绍如何将 Maven 类型的制品存储在 CODING 制品库中。其内容包括创建制品库、推送和拉取制品。

进入制品库功能页

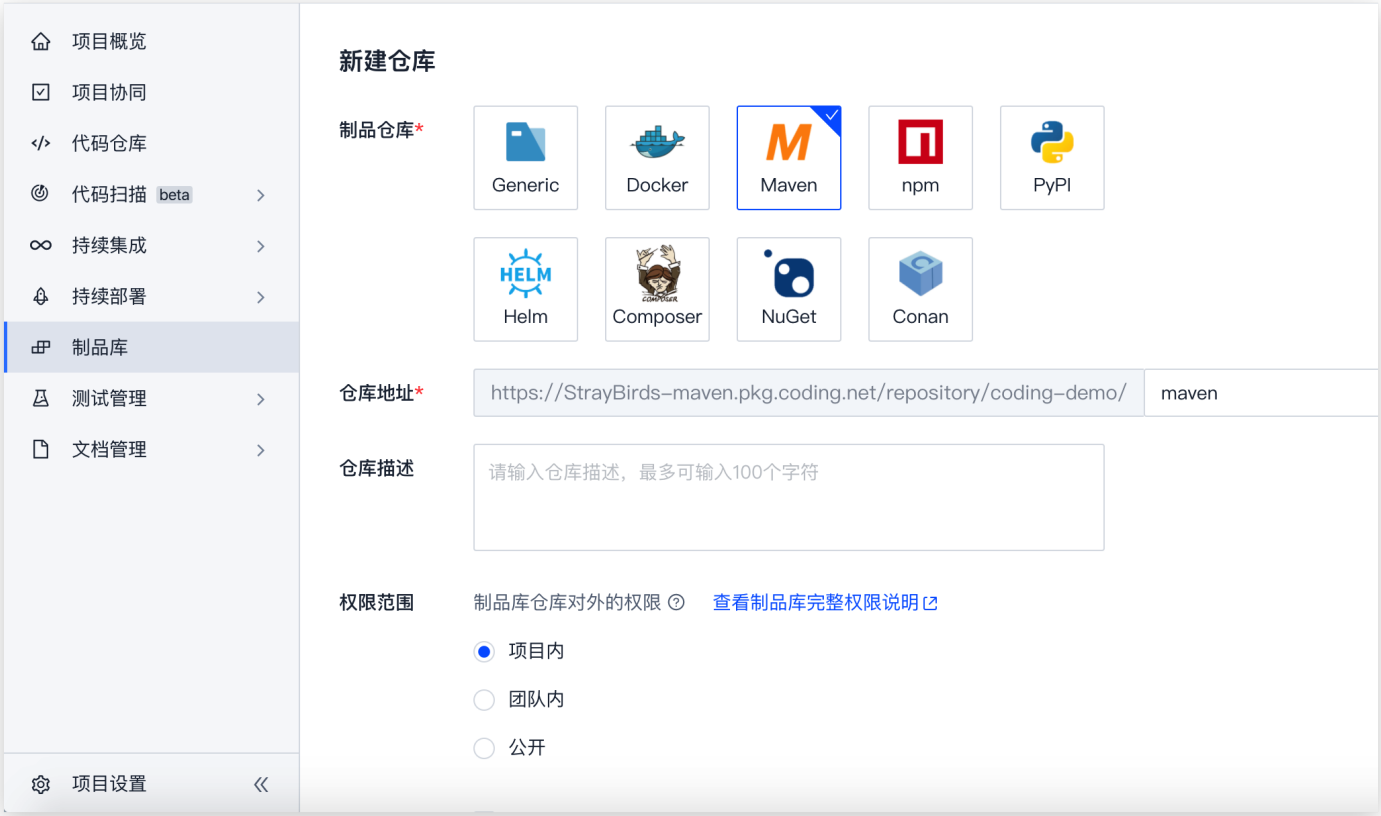
1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

创建制品仓库

进入项目，在左侧栏选择**制品库**，再单击**创建仓库**。

- 仓库类型选中 Maven。
- 提供一个仓库名称。
- 提供一段仓库描述（非必填）。
- 决定当前创建的制品仓库对不同类型角色的操作权限，默认将对当前项目成员开放**推送**和**拉取**操作。
- 准备就绪，单击**确认新建**。

版权所有：腾讯云计算（北京）有限责任公司



配置认证信息

在对 CODING 制品仓库进行推送或拉取操作之前，需要配置认证信息。

您有两种方式可以配置认证信息：

- 在 settings.xml 文件当中配置访问令牌。
- 在 settings.xml 文件当中配置 CODING 账户与密码。

推荐使用**访问令牌**生成认证的配置信息。

> 说明：
> 具体 Maven 的 settings.xml 配置文件路径，请参见 常见问题解答。

方式一：使用访问令牌生成配置

1. 在仓库的指引页面单击**使用访问令牌生成配置**，在弹窗提示中输入账号登录密码。

2. 复制生成的配置，添加到您的 settings.xml 相应配置中。

请将下列配置添加到您的 `settings.xml` 相应配置，管理请到 个人访问令牌。

```
<settings>
    <!-- omitted xml -->
    <servers>
        <server>
            <!--仓库 ID-->
            <id>anywhere-coding-demo-my-maven</id>
            <username>my-maven-</username>
            <password></pas
        </server>
    </servers>
</settings>
```

方式二：手动配置账号密码

在仓库的指引页面，复制下列配置，并将下方的 PASSWORD 替换成您的登录密码，填入您的 settings.xml 即可。

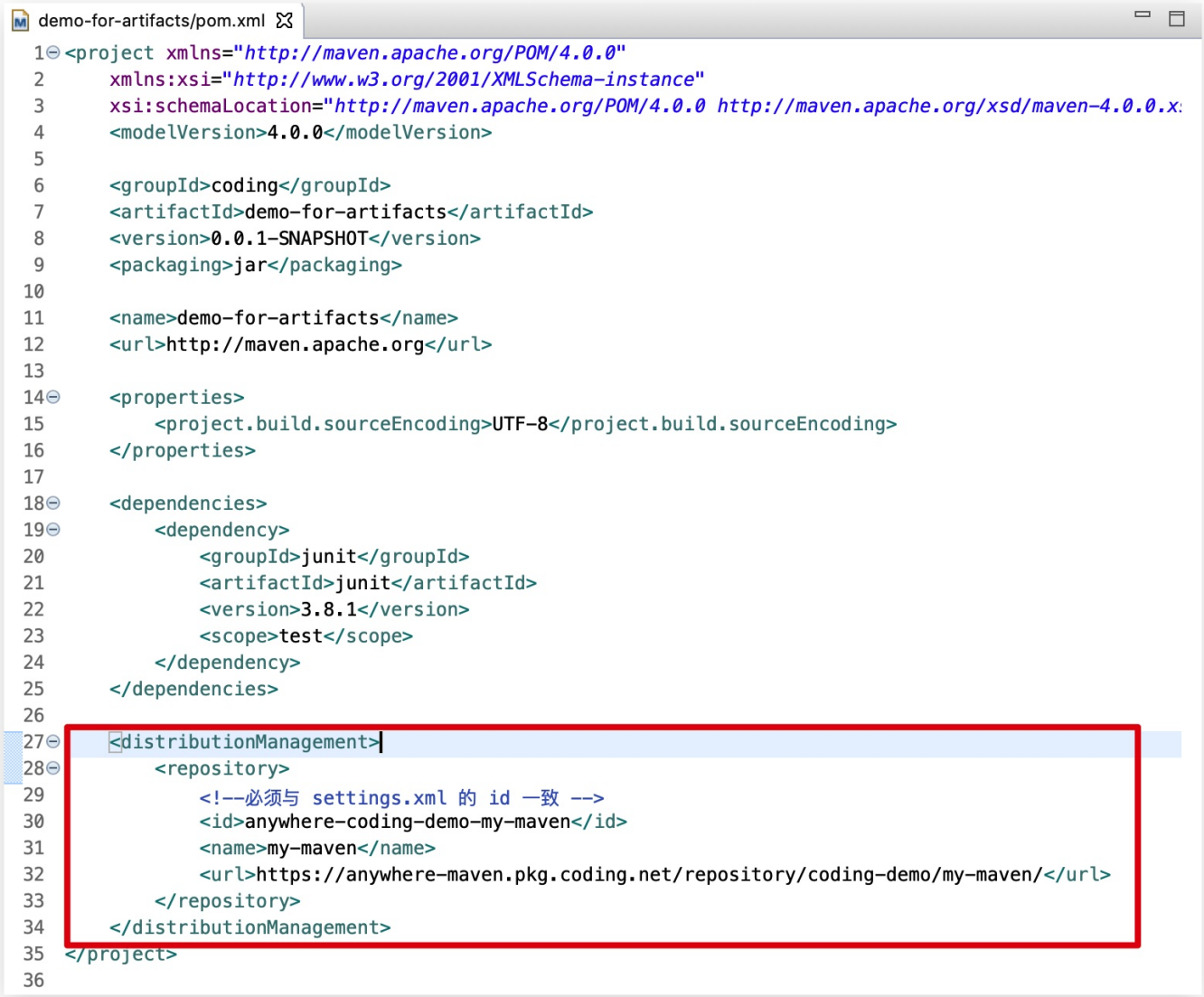
编译 Maven 制品并上传

以一个简单的 demo 为例，我们希望把这个 demo 的 Maven 包推送到上述步骤中创建好的 Maven 仓库中。

```
:0.0.1-SNAPSHOT $ ls -l
total 32
-rw-r--r--  1  staff   220 10 28 13:42 _remote.repositories
-rw-r--r--  1  staff  2254 10 28 13:42 demo-for-artifacts-0.0.1-SNAPSHOT.jar
-rw-r--r--  1  staff   766 10 28 11:03 demo-for-artifacts-0.0.1-SNAPSHOT.pom
-rw-r--r--  1  staff   710 10 28 13:42 maven-metadata-local.xml
:0.0.1-SNAPSHOT $ pwd
/Users/ /.m2/repository/coding/demo-for-artifacts/0.0.1-SNAPSHOT
```

1. 在仓库的指引页面，复制下列配置到项目的 pom.xml 文件当中。

通常一个 Maven 项目当中已有 groupId、artifactId、version 的配置，只将 distributionManagement 拷贝进去即可。



2. 执行 mvn deploy 命令。

```
mvn deploy
```

若提示未找到 settings.xml 文件，在命令末尾加上 -s 参数来设置您放置 settings 文件的路径，代入参数后：

```
mvn deploy -s "/Users/somebody/software/apache-maven-3.6.2/conf/settings.xml"
```

3. mvn 命令提示 build success 后，刷新制品仓库页面，即可看到最新推送上来的制品。

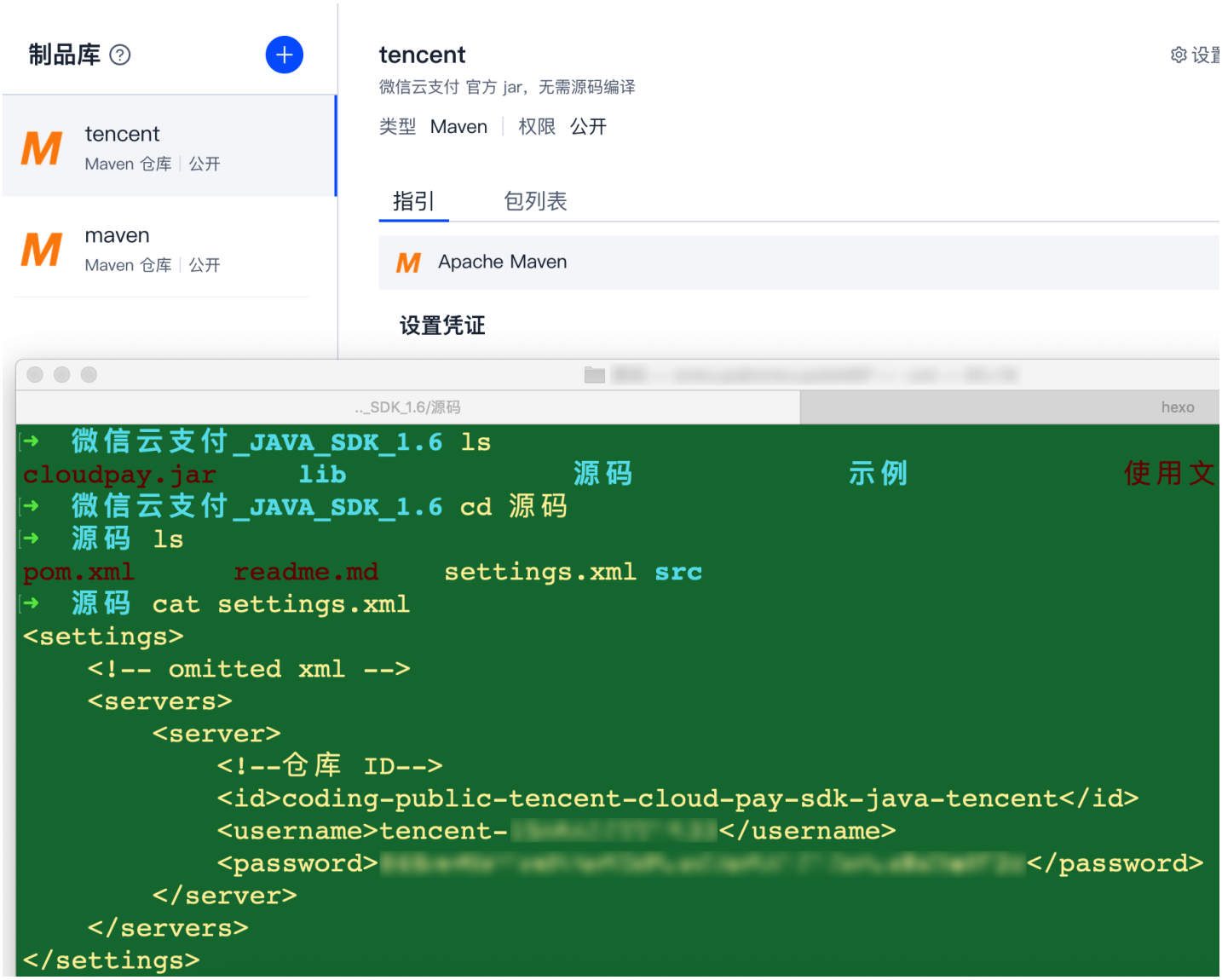
上传无源码的 Maven 包

如果第三方 Maven 包未正规发布到网络仓库，而且仅提供 jar 包，未提供源码或者源码编译报错，那我们可以把 jar 包直接上传到仓库，命令如下：

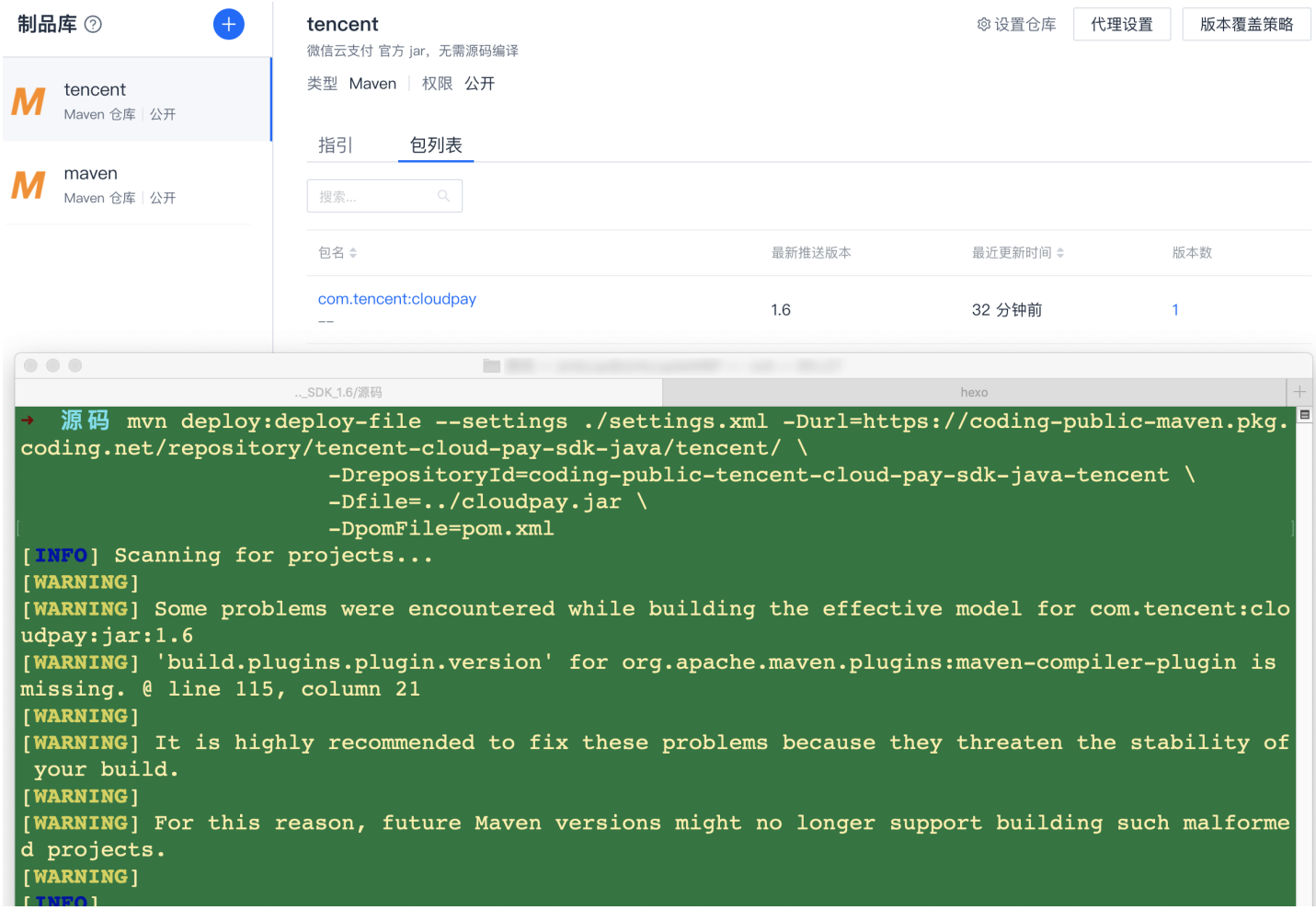
```
mvn deploy:deploy-file -Durl=file://C:\m2-repo \
-DrepositoryId=some.id \
-Dfile=your-artifact-1.0.jar \
[-DpomFile=your-pom.xml] \
[-DgroupId=org.some.group] \
[-DartifactId=your-artifact] \
[-Dversion=1.0] \
[-Dpackaging=jar] \
[-Dclassifier=test] \
[-DgeneratePom=true] \
[-DgeneratePom.description="My Project Description"] \
[-DrepositoryLayout=legacy]
```

如果第三方提供了 pom.xml，可以从中读取 group、artifact、version 等字段，例如微信云支付 Java SDK 使用下列命令：

```
mvn deploy:deploy-file --settings ./settings.xml -Durl=https://coding-public-maven.pkg.coding.net/repository/tencent-cloud-pay-sdk-java/tencent/ \
-DrepositoryId=coding-public-tencent-cloud-pay-sdk-java-tencent \
-Dfile=../cloudpay.jar \
-DpomFile=pom.xml
```



微信云支付 Java SDK 上传 jar 包列表页面。



拉取 Maven 制品

1. 在仓库的指引页面，复制配置到 settings.xml 当中，例如**微信云支付 Java SDK**的配置如下：

```
<settings>
<!-- omitted xml -->
<profiles>
<profile>
<id>Repository Proxy</id>
<activation>
<activeByDefault>true</activeByDefault>
</activation>
<repositories>
<repository>
<id>coding-public-tencent-cloud-pay-sdk-java-tencent</id>
<name>tencent</name>
<url>https://coding-public-maven.pkg.coding.net/repository/tencent-cloud-pay-sdk-java/t
```

```
encent/</url>
<releases>
<enabled>true</enabled>
</releases>
<snapshots>
<enabled>true</enabled>
</snapshots>
</repository>
</repositories>
</profile>
</profiles>
</settings>
```

2. 在您的 Java 项目的 pom.xml 中配置依赖包（dependencies 标签），例如依赖**微信云支付 Java SDK**的配置如下：

```
<project>
<dependencies>
<dependency>
<groupId>com.tencent</groupId>
<artifactId>cloudpay</artifactId>
<version>1.6</version>
</dependency>
</dependencies>
</project>
```

3. 编译项目。

```
mvn install -s ./settings.xml
```

执行过程您可以看到包被正常拉取下来，也可以在执行完成后在本地 maven 缓存看到拉取下来的包：

```
pom.xml src     target
[redacted] demo-for-artifacts-consumer [redacted]$ mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------< coding:demo-for-artifacts-consumer >-----------------
[INFO] Building demo-for-artifacts-consumer 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from anywhere-coding-demo-my-maven: https://anywhere-maven.pkg.coding.net/repos
itory/coding-demo/my-maven/coding/demo-for-artifacts/0.0.1-SNAPSHOT/maven-metadata.xml
Downloaded from anywhere-coding-demo-my-maven: https://anywhere-maven.pkg.coding.net/reposi
tory/coding-demo/my-maven/coding/demo-for-artifacts/0.0.1-SNAPSHOT/maven-metadata.xml (774
B at 332 B/s)
```

npm 制品库

最近更新时间：2023-08-09 11:14:26

该文档介绍如何将 npm 类型制品存储在 CODING 制品库中，方便团队在项目进行统一的制品管理与版本控制。下文包含如何进行制品制作、认证配置与制品推拉。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 Node.js。
> - 参见 基础操作创建制品仓库。
> - 制品仓库选择 npm 类型。

初始化本地 npm 项目（可选阅读）

若您已熟悉 npm 制品的操作，则可以跳过此章节。

1. 新建 Demo 目录作为 npm 的项目地址。

```
mkdir npm-demo
```

2. 初始化 npm 项目。

```
cd npm-demo && npm init
```

根据提示在新增的 package.json 填入该 npm 包的配置文件。参考内容：

```
{
"name": "example",
"version": "1.0.0",
```

```
"description": "",
"main": "index.js",
"author": "",
"license": "MIT"
}
```

3. 新建 .npmrc 文件。

```
touch .npmrc
```

配置认证信息

在对制品进行推送或拉取操作之前，需要配置认证信息。

有两种方式可以配置认证信息：

- 使用配置文件设置凭证。
- 使用交互式命令行设置凭证。

方式一：使用配置文件设置凭证

1. 在制品仓库的指引页面，单击**使用访问令牌生成配置**，在弹窗中输入 CODING 账号的登录密码。

2. 复制弹窗中的配置，将其添加到您项目的 package.json 同一级目录下的 .npmrc 文件。

方式二：使用交互式命令行设置凭证

1. 复制网页上的 npm config 命令设置 npm registry 为当前制品库仓库。

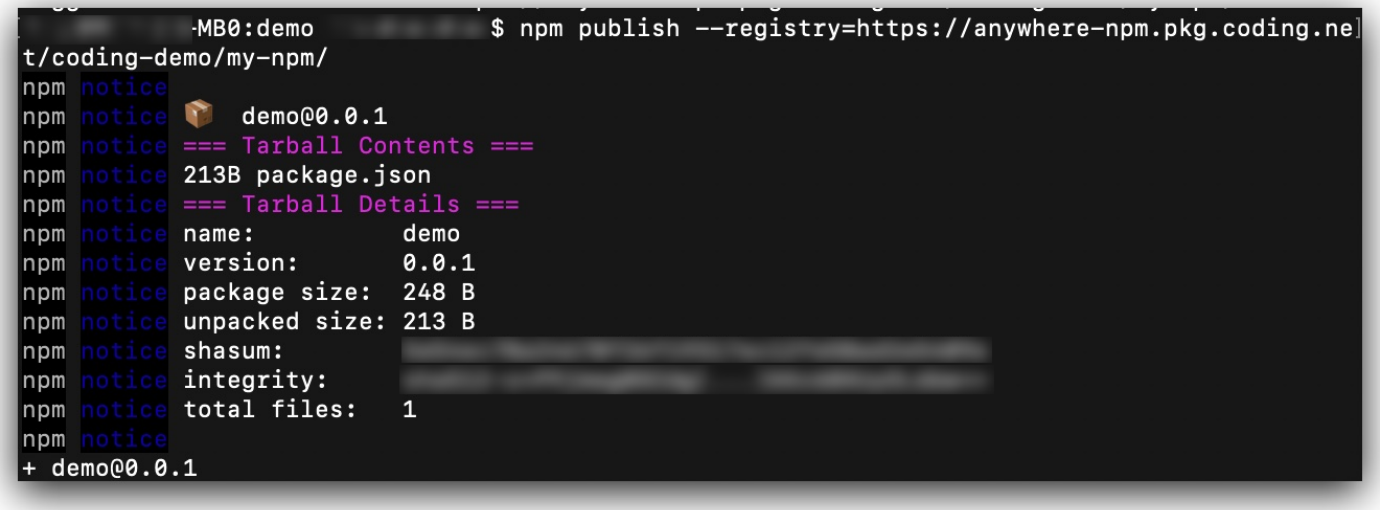
2. 执行 npm login ，按照提示输入账号名、密码、邮箱信息。

```
npm login
```

推送 npm 制品

复制网页上的**推送**命令，将本地制品推送至远端仓库中。

```
npm publish --registry=<推送指引中的仓库地址信息>
```



推送成功后，刷新仓库页面，您可以看到最新推送上来的制品。

拉取 npm 制品

执行网页上的 npm install 命令进行制品库拉取：

npm install <拟拉取的包名> --registry=<拉取指引中的仓库地址信息>

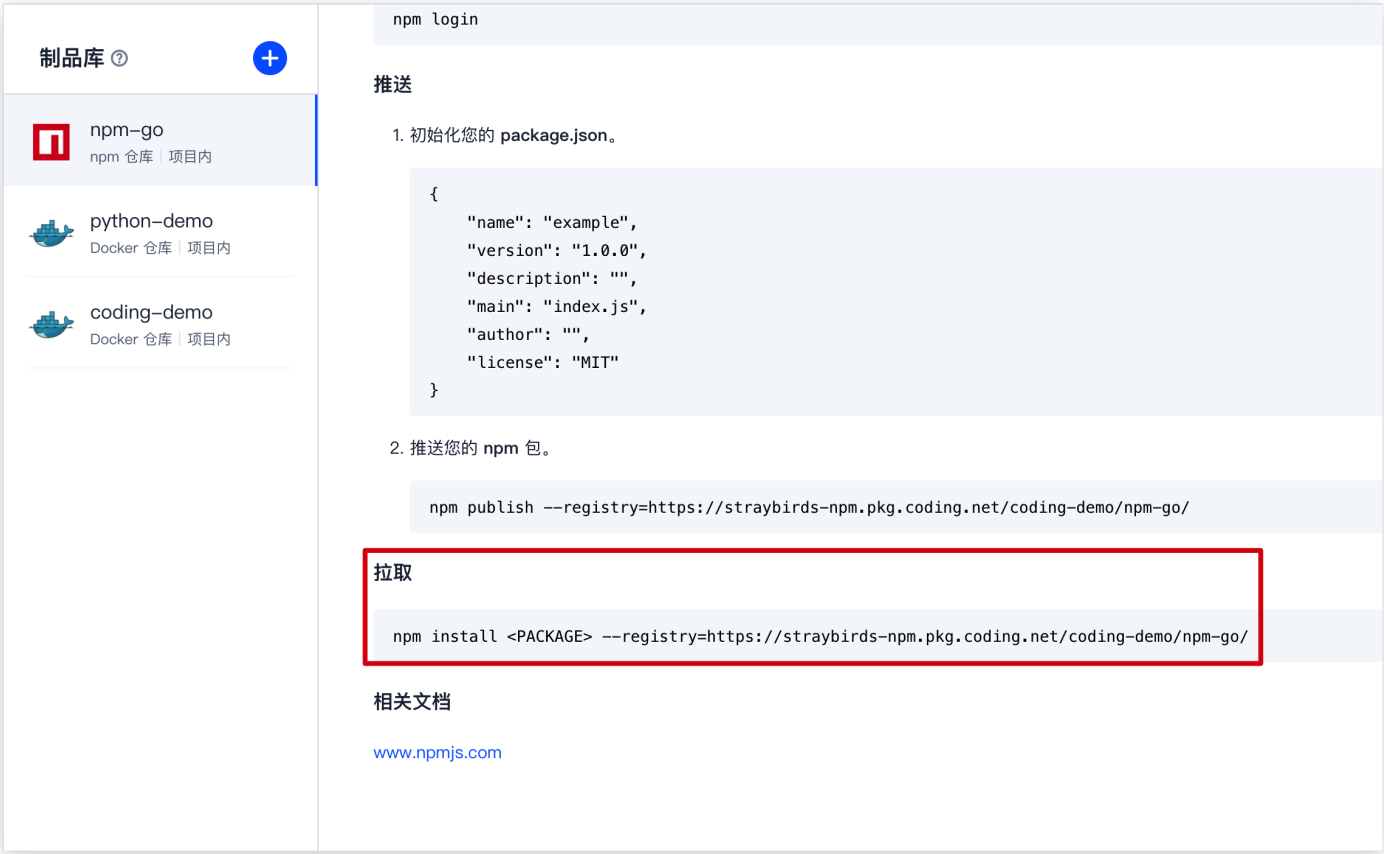
执行完毕后，您可以看到拉取成功的信息提示：

```
/Volumes/CODING/node npm install hello-world --registry=https://straybirds-npm.pkg
.coding.net/coding-demo/npm-go/
npm WARN node@1.0.0 No description
npm WARN node@1.0.0 No repository field.

+ hello-world@0.0.2
updated 1 package in 35.936s

3 packages are looking for funding
  run `npm fund` for details
```

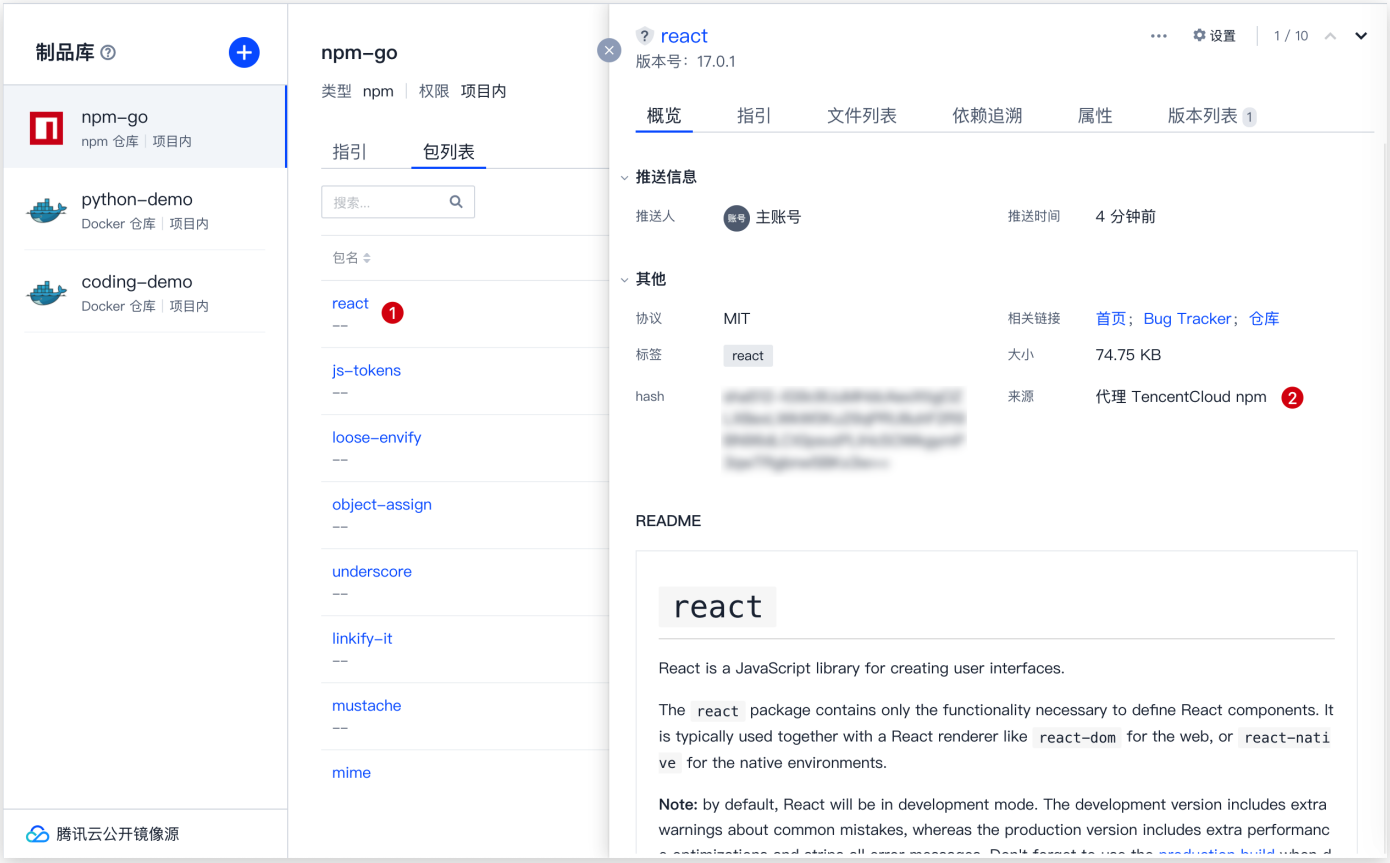
设置代理

当 CODING 私有制品仓库不存在想要拉取的制品时，将尝试从配置的代理地址拉取。您可以添加第三方制品源，用以获取特定仓库中的制品。无需额外设置，CODING 将会按照顺序从上到下依次检索相应的制品包。

使用网页上生成的命令，替换 <package> 的包名，完成拉取。



拉取的制品及依赖会成功拉取到本地，并且还会同步至 CODING 制品仓库中，详情页会显示包的来源。



说明：

代理设置的详细说明，请参见 制品库代理。

Helm 制品库

最近更新时间：2022-12-28 16:43:14

该文档介绍如何将 Helm 类型的制品存储在 CODING 制品库中。其内容包括创建制品库、推送、拉取和删除制品。

进入制品仓库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 Helm。
> - 参见 基础操作 创建制品仓库。
> - 制品仓库选择 Helm 类型。

制作包（可选阅读）

本章节提供两种方法快速创建一个 Helm Chart，若您已熟悉制作方法可跳过本节。

方法一：本地制作镜像

1. 在本地任意目录创建 Helm Chart 并自定义包名。

```
helm create [name]
```

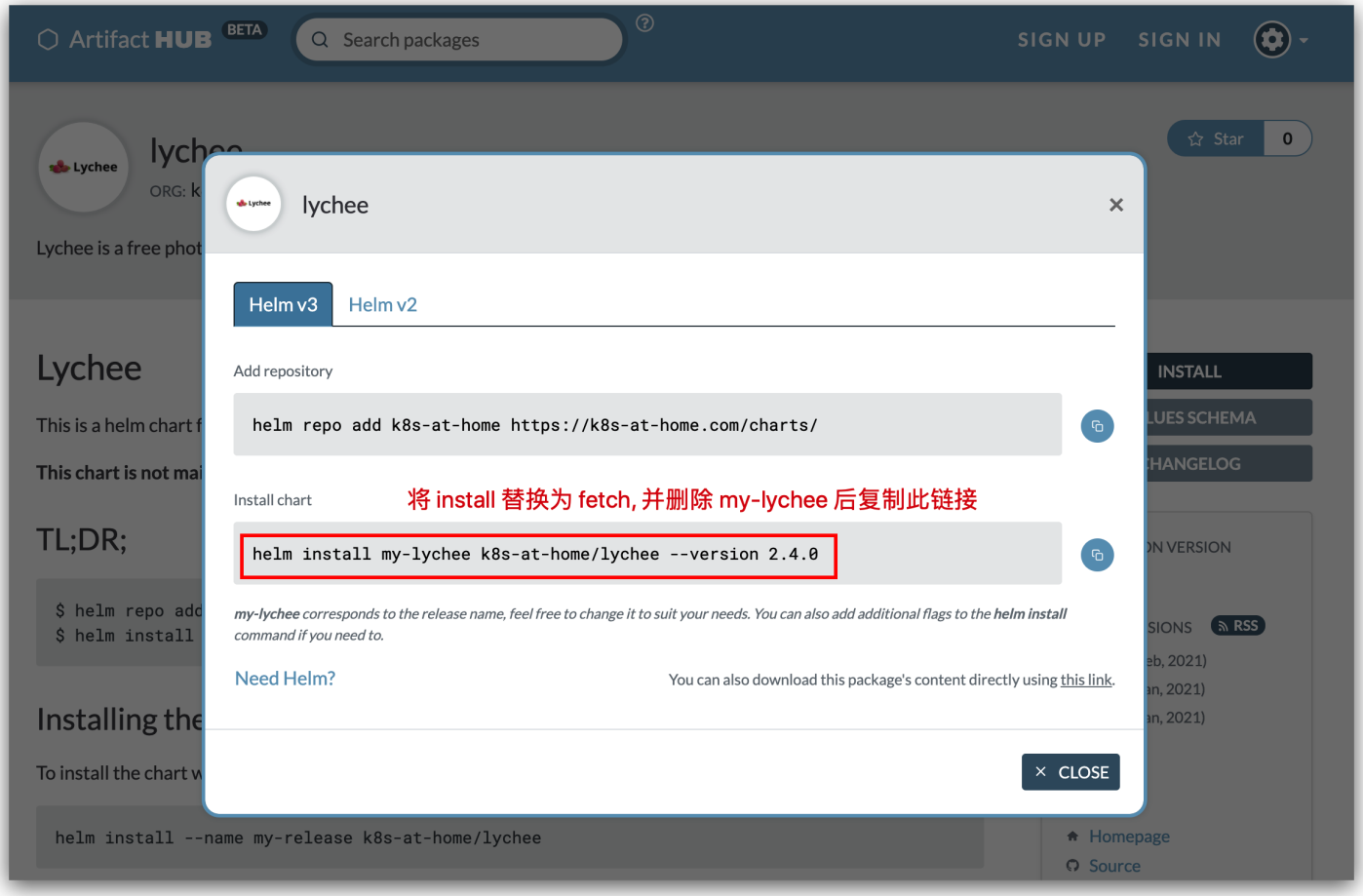
2. 打包。

```
helm package [name]
```

方法二：直接拉取 artifacthub 中的制品

搜索任意 Helm Chart 并在本地自定义目录下运行下载命令。

```
$ helm repo add [远程仓库名] [远程仓库地址]
$ helm fetch [helm chart 在远程仓库的地址>]--version [版本]
```



运行成功后本地会出现相关制品。

```
/Volumes/CODING/制品库写作/helm ls
hello                   hello-0.1.0.tgz         lychee-2.4.0.tgz        sample-golang-app-1.0.2.tgz
```

配置认证信息

在本地完成制品编译后，就可以将制品推送至远端制品仓库。推送之前需在本地配置远端仓库的认证信息。您可以按照**页面指引**，选择 Helm+ cURL 或 Helm + CODING Helm 插件两种方法进行制品推拉。

下文以 CODING 插件为例：

复制页面指引上的命令在本地安装插件后，单击使用访问令牌生成配置按钮并复制命令，使用终端在相应的包目录执行配置命令。

```
/Volumes/CODING/制品库写作/helm ➤ helm plugin install https://e.coding.net/coding-public/helm-push
Downloading and installing helm-push 0.2.0 ...
Downloading https://coding-public-generic.pkg.coding.net/helm-push/release/helm-push_0.2.0_macos-amd64.tgz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   525  100   525    0     0   1185      0 --:--:-- --:--:-- --:--:--  1185
100 9683k  100 9683k    0     0  3557k      0  0:00:02  0:00:02 --:--:-- 5529k
Preparing to install into /Users/adolf/Library/helm/plugins/helm-push
helm-push 0.2.0 installed into /Users/adolf/Library/helm/plugins/helm-push
push chart package to Coding artifact

Usage:
  helm push [chart archive/directory] [repository name] [flags]

Flags:
  -h, --help   help for helm
Installed plugin: push
/Volumes/CODING/制品库写作/helm ➤ helm repo add --username [redacted] --password [redacted]
[redacted] helm-go "https://straybirds-helm.pkg.coding.net/coding-demo/helm-go"
"helm-go" has been added to your repositories
```

推送制品

进入 Helm Chart 所在目录，并执行**页面指引**中的命令，即可把指定的 Helm Chart 推送至制品库：

```
helm push [制品包名.tgz] [推送指引中的仓库信息]
```

推送成功后，刷新仓库页面即可看到最新推送的制品。

拉取制品

如果您的制品仓库有更新，拉取前可以执行命令更新。

```
helm repo update
```

更新后执行拉取命令。

```
helm fetch [拉取指引中的制品信息] --version [版本]
```

PyPI 制品库

最近更新时间：2022-12-28 16:43:19

该文档介绍如何将 PyPI 类型制品存储在 CODING 制品库中，方便团队在项目进行统一的制品管理与版本控制。下文包含如何进行制品制作、认证配置与制品推拉。

进入制品仓库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 Python3。
> - 参见 基础操作 创建制品仓库。
> - 制品仓库选择 PyPI 类型。

初始化本地 PyPI 项目

1. 新建 Demo 目录作为本地 PyPI 包的地址。在终端中运行命令创建 Demo 项目文件夹。

```
mkdir -p demo/example_pkg/__init__.py
```

2. 进入 demo 目录，创建 setup.py 文件。

```
cd demo && touch setup.py
```

3. 在 setup.py 文件中粘贴配置内容。

```
import setuptools

setuptools.setup(
name="example-pkg-YOUR-USERNAME-HERE", # Replace with your own username
```

```
version="0.0.1",
author="Example Author",
author_email="author@example.com",
description="A small example package",
url="https://github.com/pypa/sampleproject",
packages=setuptools.find_packages(),
classifiers=[
"Programming Language :: Python :: 3",
"License :: OSI Approved :: MIT License",
"Operating System :: OS Independent",
],
python_requires='>=3.6',
)
```

4. 安装 setuptools 和 wheel 工具。

```
python3 -m pip install --user --upgrade setuptools wheel
```

5. 打包项目。

```
python3 setup.py sdist bdist_wheel
```

打包项目后，会在 /dist 目录下生成以下两个文件，用于推送到制品仓库:

```
└─npm
├─example_pkg_YOUR_USERNAME_HERE-0.0.1-py3-none-any.whl
├─example_pkg_YOUR_USERNAME_HERE-0.0.1.tar.gz
```

配置制品库认证信息

推送至 CODING 制品库之前，需在本地文件中添加相应的认证信息。您可以通过**自动生成配置**或**手动配置**两种方式进行认证。在进行操作前，请使用命令 cd / 前往根目录，输入 ls -a 查看是否存在 .pypirc 和 pip.conf 文件。

如果没有的话，输入以下命令以新建两个文件。

```
touch .pypirc && touch pip.conf
```

自动生成配置

1. 单击网页上的**使用访问令牌生成配置**，系统会自动帮您生成访问凭据。若需查看个人令牌，则前往**个人账户设置 > 访问令牌**处进行管理。

2. 输入登录密码后确认，得到配置内容。

将配置内容复制进入根目录的 .pypirc 和 pip.conf 文件中。

手动配置

按照网页上的指引提示，复制相关代码至特定文件中。

推送

进入项目目录，如上文中新建的 Demo 目录，复制网页上的命令后在终端执行，即可把 Demo/dist 目录下的所有制品推送至制品库。

```
twine upload -r coding-pypi dist/*
```

```
/Volumes/CODING/pypi/packaging_tutorial ▶ twine upload -r coding-pypi dist/*
Uploading distributions to https://codingcorp-pypi.pkg.coding.net/coding-help-generator/pypi-go
Uploading example_pkg_YOUR_USERNAME_HERE-0.0.1-py3-none-any.whl
100%|████████████████████████████████████████| 4.41k/4.41k [00:00<00:00, 6.96kB/s]
Uploading example-pkg-YOUR-USERNAME-HERE-0.0.1.tar.gz
```

推送成功后，刷新仓库页面，您可以看到最新推送的制品。

拉取

根据 PyPI 制品仓库中具体制品的拉取指引可以执行 pip install 拉取制品。

```
pip install <制品名>
```

设置代理

当 CODING 私有制品仓库不存在想要拉取的制品时，将尝试从配置的代理地址拉取。您可以添加第三方制品源，用以获取特定仓库中的制品。无需额外设置，CODING 将会按照顺序从上到下依次检索相应的制品包。

使用网页上生成的命令，替换 <package> 的包名，完成拉取。

拉取的制品及依赖会成功拉取到本地，并且还会同步至 CODING 制品仓库中，详情页会显示包的来源。

代理设置的详细说明，请参见 制品库代理。

Composer 制品库

最近更新时间：2022-12-28 16:43:23

本文档介绍如何快速使用 Composer 制品仓库，方便团队在项目进行统一的制品管理与版本控制。内容包含如何进行制品包制作、认证配置与制品推拉。

进入制品仓库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 Composer。
> - 参见 基础操作 创建制品仓库。
> - 制品仓库选择 Composer 类型。

制作 Composer 包（可选阅读）

安装 Composer

在终端中执行下载 Copmoser 命令。

```
curl -sS https://getcomposer.org/installer | php
```

添加至环境变量，方便全局运行命令。

```
mv composer.phar /usr/local/bin/composer
```

初始化

新建 Demo 目录。

```
mkdir composer-demo && cd composer-demo
```

初始化 Composer 包，按照提示输入初始化信息。

```
composer inits
```

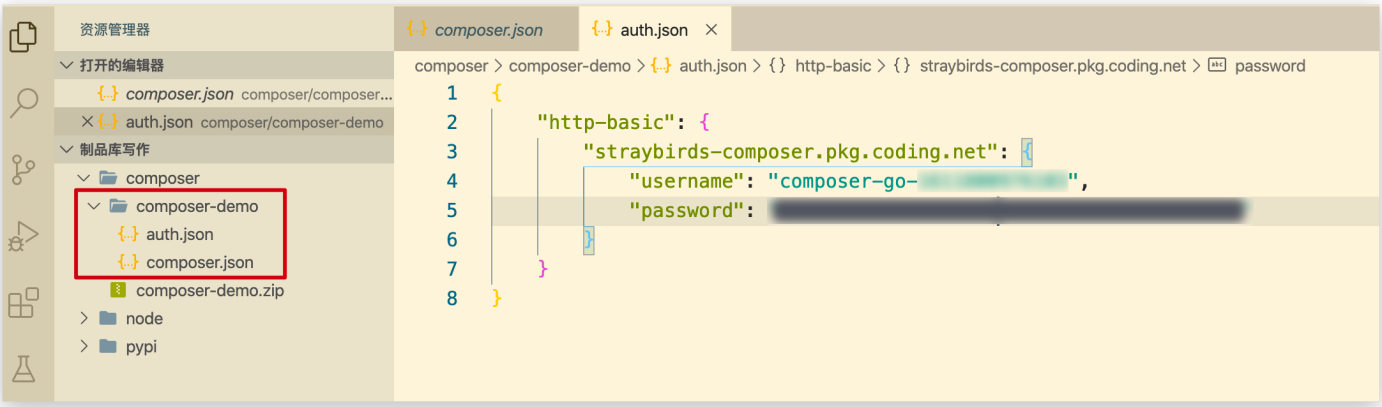
初始化完成后会在同一目录下新增 composer.json 文件作为该 Composer 包的配置文件。

配置认证文件

前往 Composer 包的文件目录，新建 auth.json 文件。您可以使用自动生成配置或手动配置两种方式进行配置。

自动生成配置

单击网页上的**使用访问令牌生成配置**按钮，复制内容后粘贴至 auth.json 文件内。



手动配置

复制网页上的命令，并将其中的［PASSWORD］替换为您的服务密码。

推送

根据 CODING 制品库页面给出的推送命令，将 Composer 包文件打包成 zip 并使用 cURL 等工具推送至仓库。

推送时需按照提示输入您在 auth.json 文件中填写的密码，并将 <version> 替换为拟定的版本号。

```
/Volumes/CODING/制品库写作/composer zip -r composer-demo.zip . -x "./vendor/*"          SIGINT(2) ↵ 9680 10:58:35
updating: composer-demo/ (stored 0%)
updating: composer-demo/auth.json (deflated 31%)
updating: composer-demo/composer.json (deflated 38%)
/Volumes/CODING/制品库写作/composer curl -T composer-demo.zip -u @qq.com "https://straybirds-composer.pkg.coding.net/cod
ing-demo/composer-go?version=0.0.2"
Enter host password for user '@qq.com':
success
```

推送完成后即可在 CODING 制品库看到已推送的包。

拉取

进入 Composer 包的文件目录，设置仓库地址。相关命令已在 CODING 页面生成。

使用 CODING 制品库代替 Composer 官方的源（可选）。

```
composer config repo.packagist false
```

执行拉取命令，将 <package> 包替换为拟拉取的包。

```
composer require < PACKAGE >:< VERSION >
```

输入服务邮箱与密码后成功拉取。

```
/Volumes/CODING/制品库写作/testing composer require adolf/composer-demo:0.0.2        Authentication required (straybirds-composer.pkg.coding.net):
      Username: [redacted]@qq.com
      Password:
https://straybirds-composer.pkg.coding.net/coding-demo/composer-go could not be fully loaded (Invalid credentials for 'https://straybirds-composer.pkg.coding.net/coding-demo/composer-go/packages.json', aborting.), package information was loaded from the local cache and may be out of date
./composer.json has been updated
Running composer update adolf/composer-demo
Loading composer repositories with package information
    Authentication required (straybirds-composer.pkg.coding.net):
      Username: [redacted]@qq.com
      Password:
Do you want to store credentials for straybirds-composer.pkg.coding.net in /Users/adolf/.composer/auth.json ? [Yn] y
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking adolf/composer-demo (0.0.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Downloading adolf/composer-demo (0.0.2)
  - Installing adolf/composer-demo (0.0.2): Extracting archive
Generating autoload files
```

设置代理

当 CODING 私有制品仓库不存在想要拉取的制品时，将尝试从配置的代理地址拉取。您可以添加第三方制品源，用以获取特定仓库中的制品。无需额外设置，CODING 将会按照顺序从上到下依次检索相应的制品包。

代理设置的详细说明，请参见 制品库代理。

NuGet 制品库

最近更新时间：2022-12-28 16:43:27

该文档介绍如何将 NuGet 类型制品存储在 CODING 制品库中，方便团队在项目进行统一的制品管理与版本控制。下文包含制品仓库创建、NuGet 包制作、制品推拉与使用代理等。

进入制品仓库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 NuGet。
> - 参见 基础操作 创建制品仓库。
> - 制品仓库选择 NuGet 类型。

初始化 NuGet 制品（可选阅读）

访问 官网 下载并安装 NuGet。

本地生成

若您已熟悉 NuGet 制品的操作，则可以跳过此章节。

1. 新建 Demo 目录。

```
mkdir nuget-demo && cd nuget-demo
```

2. 创建 .nuspec 包。

```
nuget spec [制品名称]
```

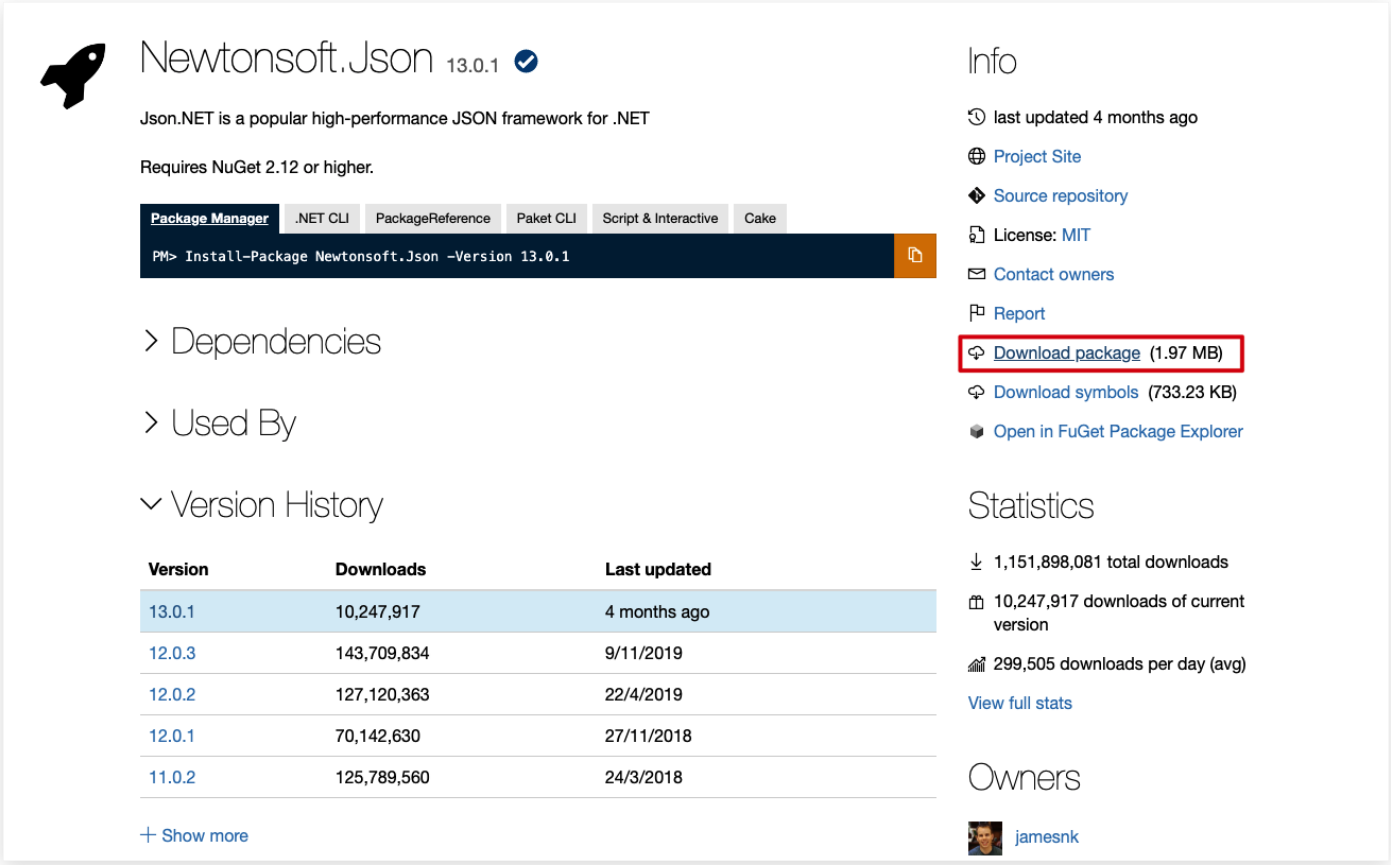
3. 打包制品。

```
nuget pack <制品名称>.nuspec
```

4. 打包完成后即可在本地目录中看到生成的包文件。

在线拉取

单击访问 官网，搜索任意 NuGet 制品并通过在线链接或命令行下载。



通过命令行拉取：

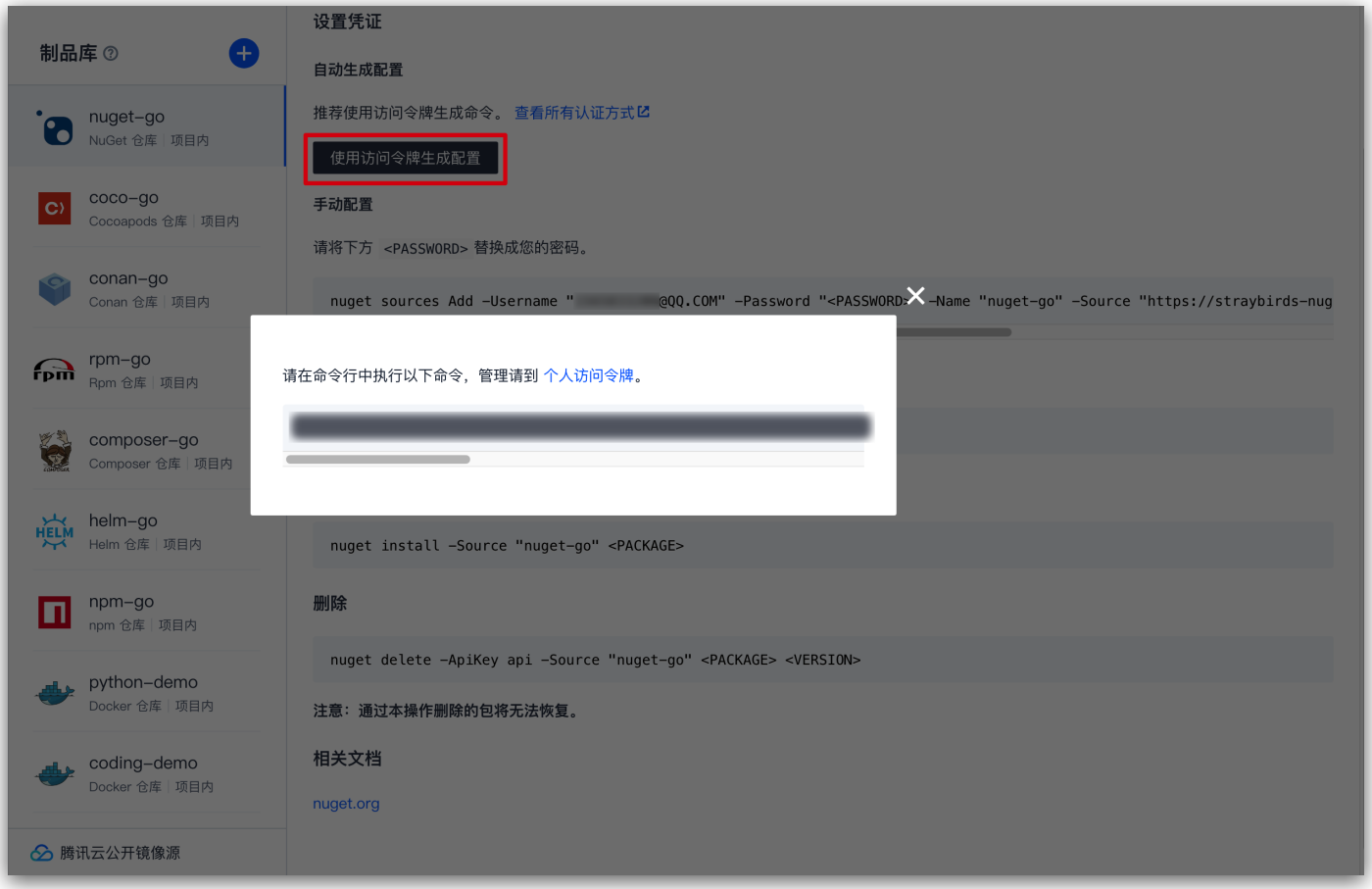
```
nuget install [制品名称] -OutputDirectory packages
```

配置制品仓库认证

您需要在本地配置认证信息，用以访问 CODING 中的 NuGet 类型制品仓库。此处我们使用**自动生成配置**完成认证过程。

单击页面上的**使用访问令牌生成配置**按钮，输入密码后得到配置命令，复制后在需要推送的 NuGet 制品的所在目录执行配置命令即可。此过程的**权限机制**使用到了**个人访问令牌**功能，若希望对其进行控制，详情请参见 访问令

牌。



推送制品

输入命令行，将相应的名称替换为本地内容即可完成推送。

```
nuget push -ApiKey api -Source [推送指引中提供的仓库名称] [本地制品名称].nupkg
```

拉取制品

输入命令行，将相应的名称替换为本地内容即可完成拉取。

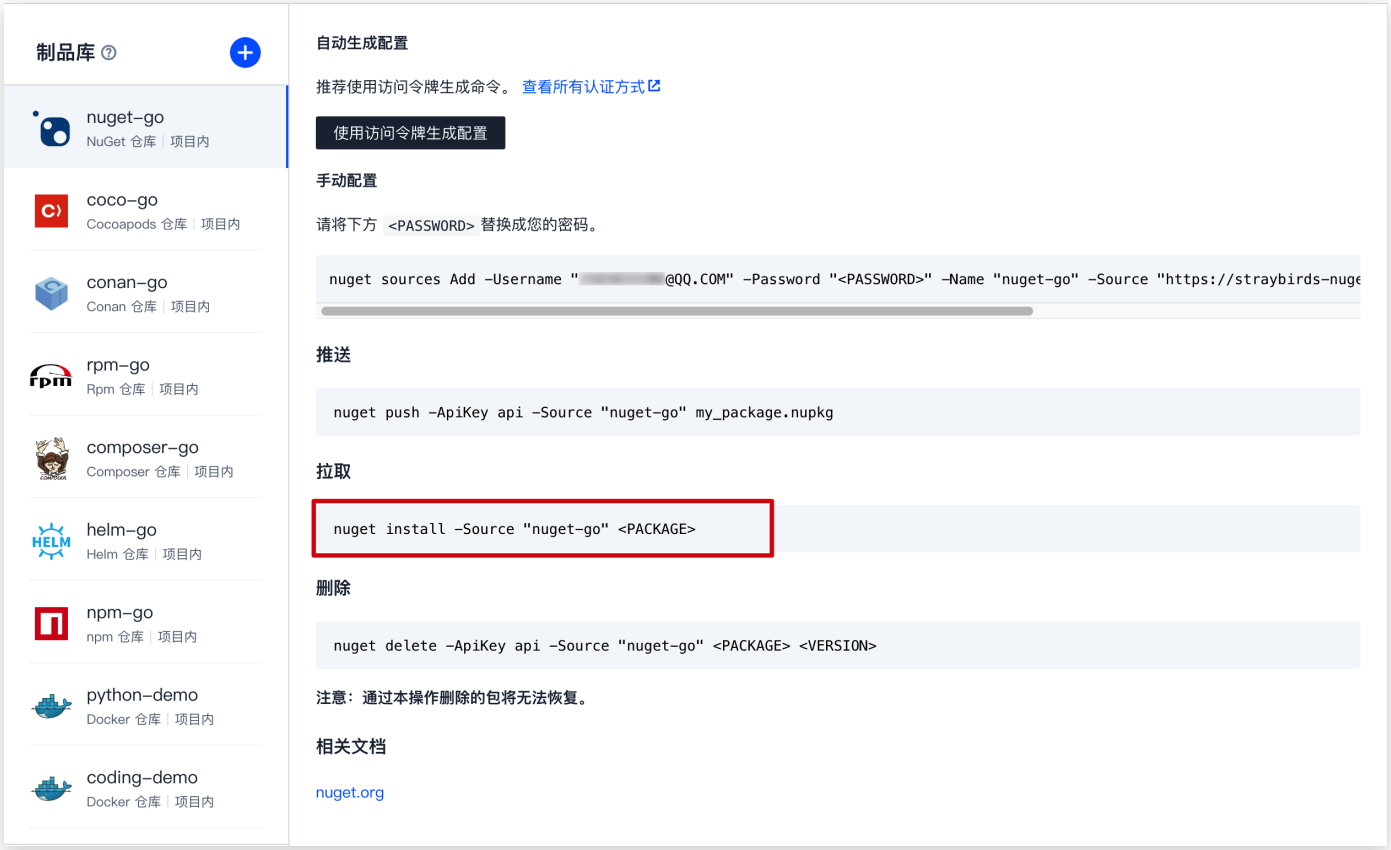
```
nuget install -Source [拉取指引中提供的仓库名称] -Version [制品版本] [制品名称]
```

设置为代理

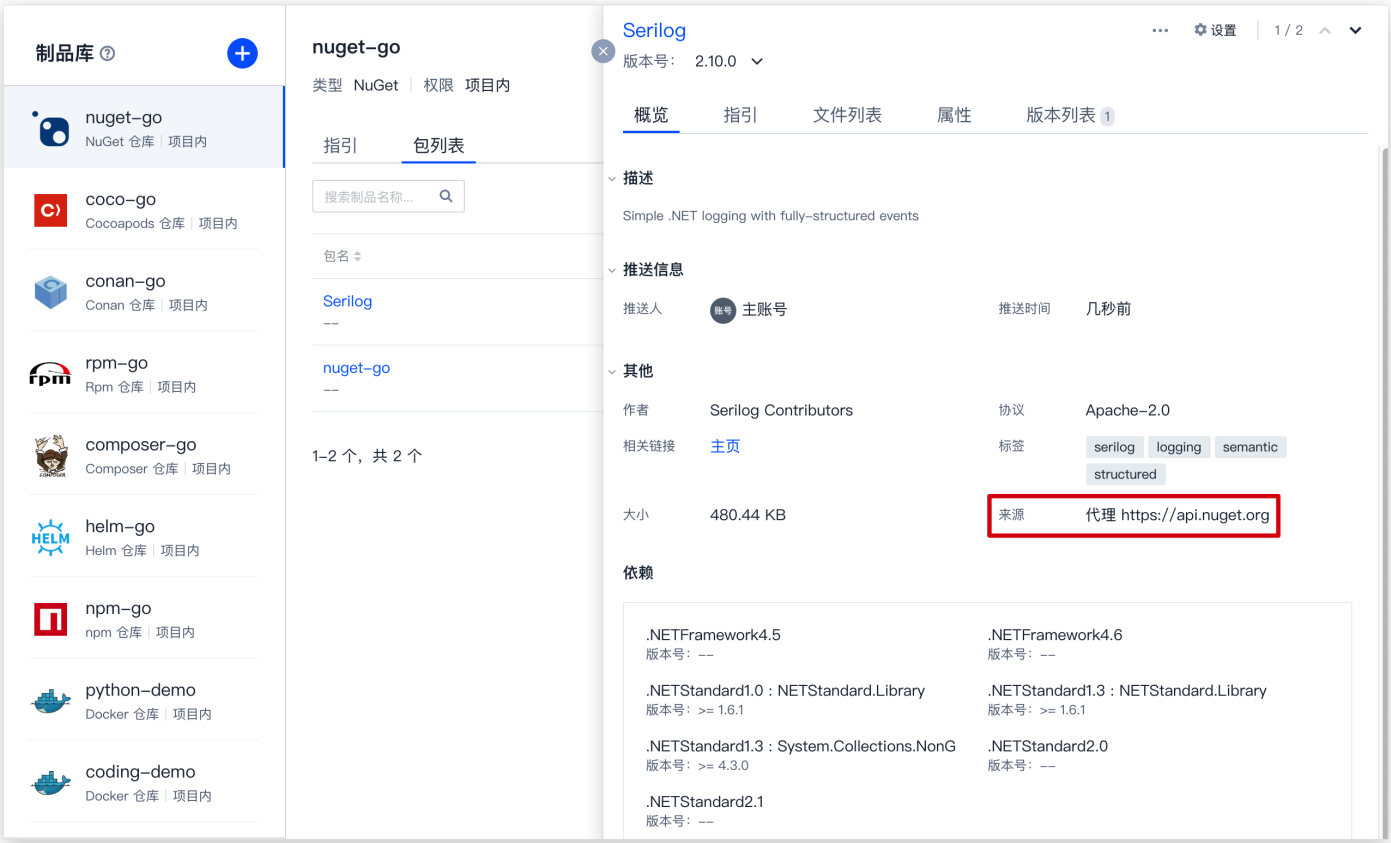
当 CODING 私有制品仓库不存在想要拉取的制品时，将尝试从配置的代理地址拉取。您可以添加第三方制品源，用以获取特定仓库中的制品。无需额外设置，CODING 将会按照顺序从上到下依次检索相应的制品包。

使用命令拉取制品：

```
nuget install -Source [制品仓库名称] -Version [制品版本] [制品名称]
```



拉取的制品及依赖会成功拉取到本地，并且还会同步至 CODING 制品仓库中，详情页会显示包的来源。



> 说明
>
> 若 CODING 制品仓库中没有自动储存由代理拉取的 NuGet 制品，可能由于以下两点问题导致：
>
> 1. 您没有该仓库的推送权限。
> 2. 您的本地缓存中已有该制品。

代理设置的详细说明，请参见 制品库代理。

rpm 制品库

最近更新时间：2022-12-28 16:43:31

该文档介绍如何将 rpm 类型制品存储在 CODING 制品库中，方便团队在项目进行统一的制品管理与版本控制。下文包含如何进行制品制作、认证配置与制品推拉。

进入制品库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - Linux 环境。
> - 参见 基础操作 创建制品仓库。
> - 制品仓库选择 rpm 类型。

初始化本地 rpm 项目

Linux 系统自带 rpm，您可以直接在运行 Linux 系统的终端直接运行命令，若置于其他操作系统，则可以使用 Docker 安装 Centos：

```
docker run -it --name centos centos:8 /bin/bash
```

下载 Demo 项目

进入 rpm 制品下载地址，搜索制品包并下载至本地后进行安装。

例如：

```
wget -N --no-check-certificate "https://www.rpmfind.net/linux/fedora/linux/development/rawhide/Everything/aarch64/os/Packages/h/hello-2.10-5.fc34.aarch64.rpm" && rpm -i hello-2.10-5.fc34.aarch64.rpm
```

配置仓库认证信息

单击页面指引上的使用访问令牌生成配置，系统会帮您自动生成个人令牌作为访问凭证。您可以到个人账户设置 > 访问令牌进行管理。

输入登录密码后，将生成的代码复制至本地的 /etc/yum.repos.d/rpm-go.repo **文件中，如果没有该文件请新**

建。

推送制品

执行 rpm publish 命令推送 rpm 包。

```
curl -u [用户名/邮箱] -X POST [推送指引中提供的仓库地址信息] -T [制品名称].rpm
```

推送成功后，刷新仓库页面，您可以看到最新推送上来的制品。

拉取制品

运行页面指引上的命令，完成拉取操作。

制品代理

rpm 仓库已有默认代理地址，可以自定义配置其他地址。

配置需要代理的远程仓库地址，拉取仓库中的制品至本地后，将自动备份至 CODING 制品仓库列表。

> **说明**
>
> 如果 rpm 制品仓库中没有储存代理的 rpm 制品，可能是因为以下两点原因：
>
> - 您没有该仓库的推送权限。
> - 您的本地缓存中已有该制品包。

Conan 制品库

最近更新时间：2022-12-28 16:43:36

该文档介绍如何将 conan 类型制品存储在 CODING 制品库中，方便团队在项目进行统一的制品管理与版本控制。下文包含如何进行制品制作、认证配置与制品推拉。

进入制品仓库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 Python3。
> - 参见 基础操作 创建制品仓库。
> - 制品仓库选择 conan 类型。

安装 Conan

使用 pip 安装，需要 python3.5 及以上的版本。

```
pip install
```

使用 brew 安装。

```
brew install conan
```

新建 conan 包

在本地新建 Demo 目录。

```
mkdir mypkg && cd mypkg
```

创建 Demo 项目。

```
conan new hello/0.1 -t
```

将项目打包成二进制包。

```
conan create . demo/testing
```

如果执行报错 /bin/sh: cmake: command not found，需要执行命令安装 cmake。

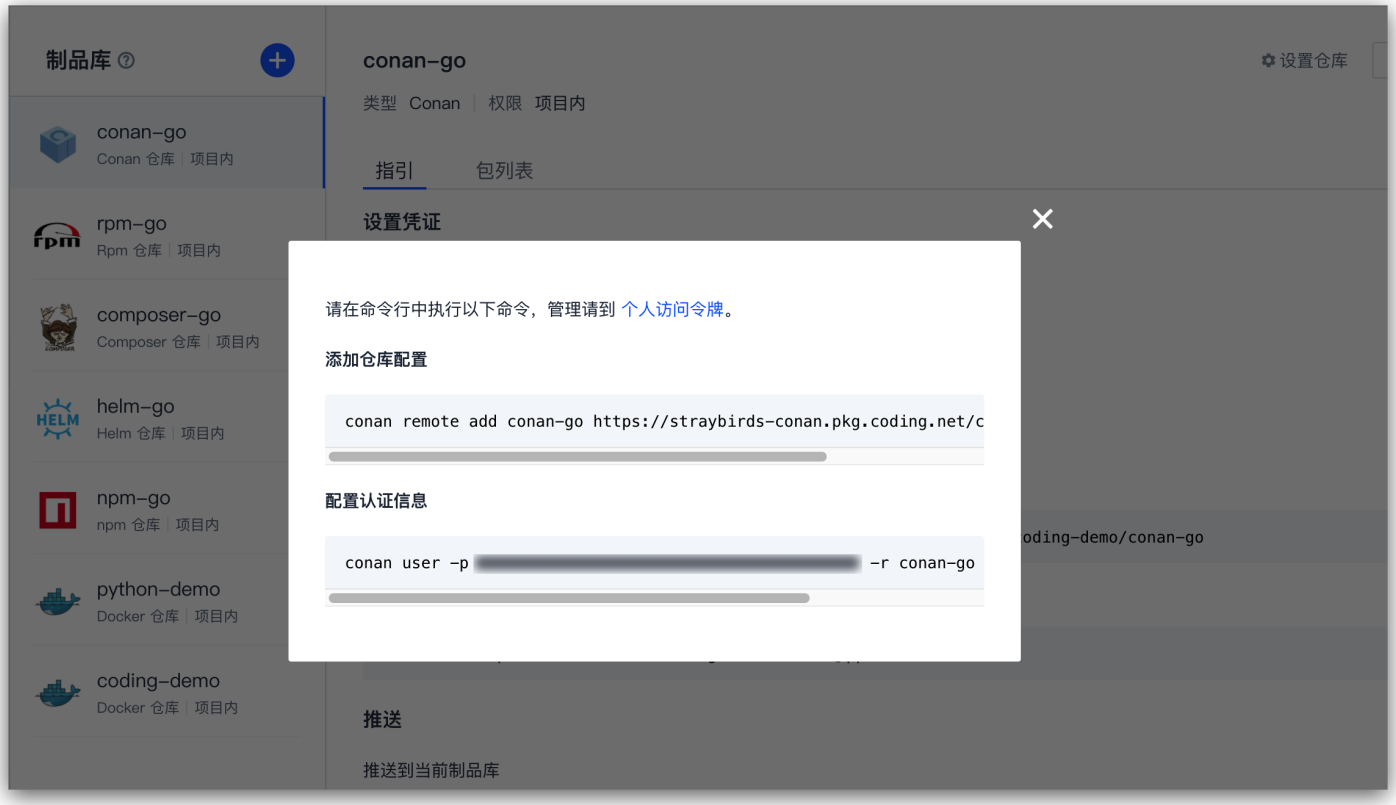
```
$ pip3 install cmake
# 或
$ brew install cmake
```

配置仓库认证信息

您可以选择自动生成配置或手动配置，下文以自动生成配置为例。

单击**页面指引**上的**使用访问令牌生成配置**，系统会帮您自动生成个人令牌作为访问凭证。您可以到**个人账户设置** > **访问令牌**进行管理。

按照提示执行命令。



推送制品

运行页面指引上的命令行，将变量替换为拟推送的制品名称与版本号。

```
conan upload [包名称]/[自定义版本号] --all -r=conan-go
```

拉取制品

执行页面指引中的拉取命令，从当前仓库拉取制品。

```
conan install [包名称]/[自定义版本号]@ -r conan-go
```

Cocoapods 制品库

最近更新时间：2022-12-28 16:43:41

该文档介绍如何将 Cocoapods 类型制品存储在 CODING 制品库中，方便团队在项目进行统一的制品管理与版本控制。下文包含如何进行制品制作、认证配置与制品推拉。

进入制品仓库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 Cocoapods。
> - 参见 基础操作 创建制品仓库。
> - 制品仓库选择 Cocoapods 类型。

安装

执行命令安装 Cocoapods。

```
$ sudo gem install cocoapods
-- 或
$ brew install cocoapods
```

新建 Demo 项目

本节对如何快速创建一个 Demo Cocoapods 制品 提供指引，若您已熟悉 Cocoapods 制品制作可以跳过本节。

在任意目录执行创建命令，并根据命令行提示选择需要的示例模板：

```
pod lib create <自定义 pod 名称>
```

该命令会将 Cocoapods 官方的示例代码从 GitHub 仓库克隆至本地。

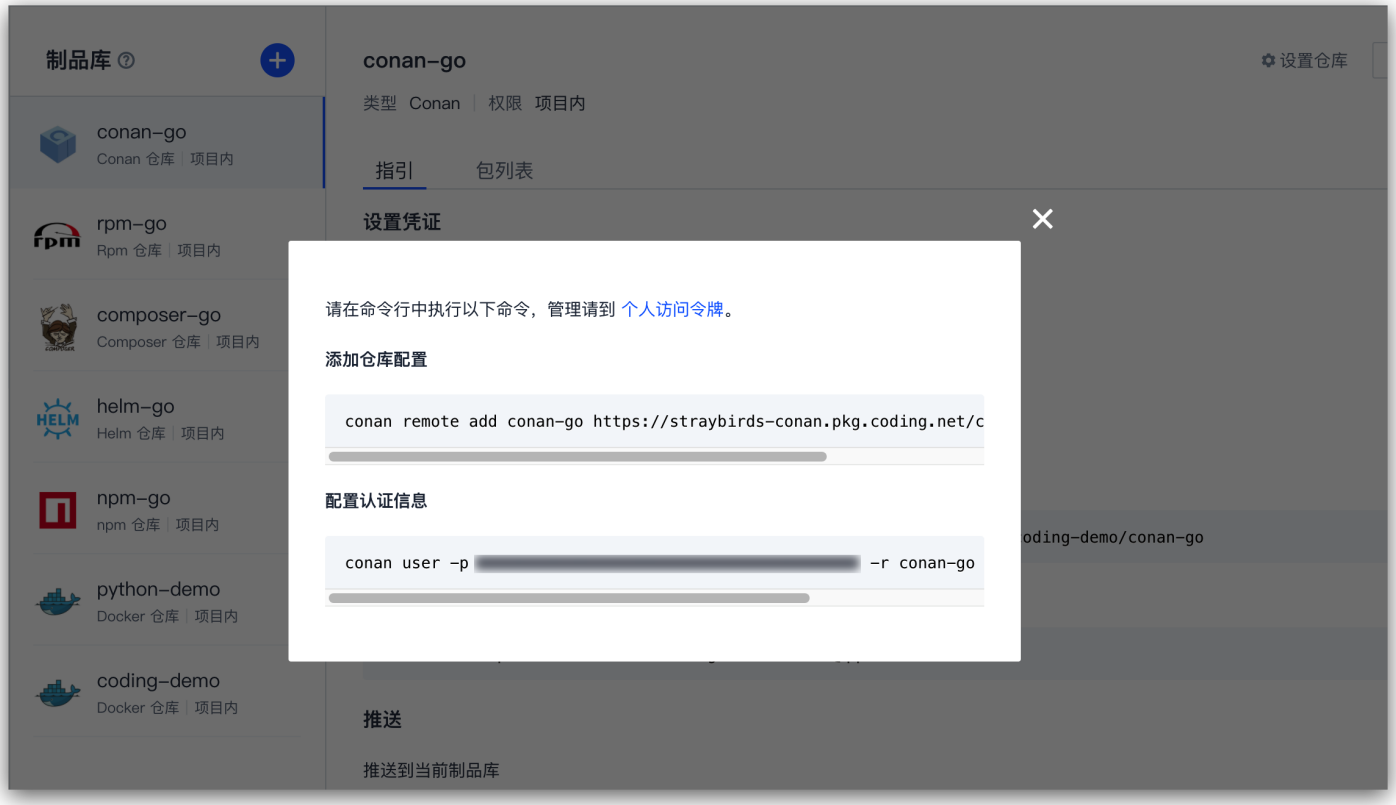
配置认证信息

首先安装 CODING Cocoapods 插件。

```
sudo gem install cocoapods-coding-ar
```

接下来您可以选择自动生成配置或手动配置，下文以自动生成配置为例。单击**页面指引**上的**使用访问令牌生成配置**，系统会帮您自动生成个人令牌作为访问凭证。您可以到**个人账户设置** > **访问令牌**进行管理。

复制命令至您的 ~/.netrc 文件中。若没有该文件，运行 vim ~/.netrc 新建并粘贴内容。



推送制品

按照页面指引执行命令。

拉取制品

根据 Cocoapods 制品仓库中具体制品的拉取指引可以拉取指定 Cocoapods 制品。

1. 修改 Podfile 文件配置。

```
source '<拉取指引中提供的仓库地址>'
target 'MyApp' do
pod '<制品名称>', '~> <版本>'
end
```

2. 执行拉取命令。

```
pod install
```

Go 制品库

最近更新时间：2023-07-28 17:58:02

该文档介绍如何将 Go 类型制品存储在 CODING 制品库中，方便团队在项目进行统一的制品管理与版本控制。下文包含 Go 制品仓库创建、本地打包 Go 制品、制品推拉使用等功能介绍。

进入制品仓库功能页

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 单击左侧菜单栏的**制品管理**，进入制品仓库功能页面。

准备工作

> ⚠ 注意：
>
> 阅读该篇文档需要准备好以下内容：
>
> - 安装 Go 1.13 以上版本。
> - 参见 基础操作 创建项目、新建制品仓库。
> - 制品仓库选择 Go 类型。

配置认证信息

在对制品进行推送或拉取操作之前，需要配置认证信息。

1. 在制品仓库的**仓库管理**页签，单击**操作指引**。

2. 在**操作指引**的**配置凭据**页面输入个人账号的登录密码，单击**生成个人令牌作为凭证**。

3. 根据操作系统添加 GOPROXY 配置，复制弹窗中的命令在终端执行。

推送 Go 制品

使用 CLI 工具推送 Go 制品。

1. 在**操作指引**的**推送**页面，根据不同操作系统复制并在终端执行对应的命令，完成 CLI 工具下载安装。

2. 输入**制品模块**、**制品版本**，系统自动生成推送命令。根据不同操作系统，复制并在终端执行对应的命令，完成 Go 制品包推送至制品仓库。

制品包推送成功后终端界面显示如下信息。

制品仓库页面制品列表中能查看已经推送成功的 Go 制品。

拉取 Go 制品

使用以下任意一种方式拉取 Go 制品。

- 在**操作指引**的**推送**页面，输入 Go 制品的 module 信息，复制并在终端执行命令来拉取 Go 制品。

> **说明：**

- 填写的**制品模块**信息必须与 Go 制品 go.mod 文件中 moudle 信息保持一致，否则会导致拉取 Go 制品库失败。
- 在非 HTTPS 环境使用 go get 命令时，要求 Go proxy 不附带鉴权信息，且必须是公开仓库。

- 在终端的 go.mod 文件所在目录下执行以下命令，拉取依赖制品。

制品包拉取成功后终端界面显示如下信息。

```
➜  croc git:(main) ✗ go get github.com/schollz/croc/v9
go: added github.com/cespare/xxhash v1.1.0
```