制品仓库

最佳实践

文档版本	01
发布日期	2025-10-30

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 CodeArts Artifact 最佳实践汇总

本文汇总了基于制品仓库（CodeArts Artifact）常见应用场景的操作实践，为每个实践提供详细的方案描述和操作指导，帮助用户深入了解CodeArts Artifact的各个功能。

表 1-1 CodeArts Artifact 最佳实践一览表

最佳实践	说明
通过编译构建任务发布Maven组件并按照版本归档至私有依赖库	相对于开发过程中的“源代码”，制品仓库服务关注和管理开发产生的待部署的软件包。软件包通常是由源码编译构建或打包而成，其中涉及生命周期的元数据（如名称、大小等基本属性、代码库地址、代码分支信息、构建任务、构建者、构建时间）。在开发过程中，软件包会根据不同版本不断生成改进。 软件包及其属性的管理是发布过程管理的基础，也是软件开发过程中的重要资产，而能够及时查看软件包的版本记录也成为开发者面临的诉求。本实践介绍如何通过编译构建任务发布Maven组件并按照版本归档至私有依赖库。
通过编译构建任务发布/获取NPM私有组件	私有依赖库管理各种开发语言对应的私有组件包（开发者通俗称之为私服）。由于不同的开发语言组件通常有不同的归档格式要求，私有依赖库目的就在于管理私有开发语言组件并在企业或团队内共享给其他开发者开发使用。 本实践介绍如何通过编译构建任务发布私有组件到NPM私有依赖库、如何从NPM私有依赖库获取依赖包完成编译构建任务。
通过编译构建任务发布/获取Go私有组件	本实践介绍如何通过编译构建任务发布私有组件到Go私有依赖库、如何从Go私有依赖库获取依赖包完成编译构建任务。
通过编译构建任务发布/获取PyPI私有组件	本实践介绍如何通过编译构建任务发布私有组件到PyPI私有依赖库、如何从PyPI私有依赖库获取依赖包完成编译构建任务。
通过Linux命令行上传/获取RPM私有组件	本实践介绍如何通过Linux命令行上传私有组件到RPM私有依赖库、如何从RPM私有依赖库获取依赖包。

最佳实践	说明
通过Linux命令行上传/获取Debian私有组件	本实践介绍如何通过Linux命令行上传私有组件到Debian私有依赖库、如何从Debian私有依赖库获取依赖包。
迁移Nexus上的仓库数据到CodeArts Artifact私有依赖库	CodeArts Artifact提供了批量迁移工具，方便用户将Nexus上hosted/proxy/group类型仓库数据批量快速迁移至CodeArts Artifact私有依赖库，从而在CodeArts Artifact进行统一高效的操作和维护。
迁移JFrog仓库至私有依赖库	JFrog仓库是一个用于存储和管理软件包的中央存储库，提供了一种集中式的方式来管理软件包，支持各种软件包管理工具，如Maven、Gradle、NPM、NuGet等。CodeArts Artifact的私有依赖库提供了批量迁移工具，支持将JFrog仓库迁移至私有依赖库。本实践介绍如何批量迁移JFrog仓库至私有依赖库。
迁移本地仓库数据至CodeArts Artifact私有依赖库	CodeArts Artifact提供了批量迁移工具，方便用户将本地磁盘中的Maven仓库数据、NPM仓库数据批量快速迁移至CodeArts Artifact私有依赖库中的Maven私有依赖库、NPM私有依赖库，从而在CodeArts Artifact进行统一高效的操作和维护。
CodeArts Artifact权限配置最佳实践	本实践以私有依赖库权限管理为例，介绍在私有依赖库内快速管理权限的方式，实现对单个仓库进行单独权限管理，以及按照项目维度进行权限管理。

2 通过编译构建任务发布 Maven 组件并按照版本归档至私有依赖库

背景信息

相对于开发过程中的“源代码”，制品仓库服务关注和管理开发产生的待部署的软件包。软件包通常是由源码编译构建或打包而成，其中涉及生命周期的元数据（如名称、大小等基本属性、代码库地址、代码分支信息、构建任务、构建者、构建时间）。在开发过程中，软件包会根据不同版本不断生成改进。

软件包及其属性的管理是发布过程管理的基础，也是软件开发过程中的重要资产，而能够及时查看软件包的版本记录也成为开发者面临的诉求。

前提条件

- 该功能需要使用编译构建CodeArts Build服务，需要**购买CodeArts Build套餐**，或者**开通/购买软件开发生产线服务组合套餐**（如果已经购买了软件开发生产线服务组合套餐，则无需再单独购买CodeArts Build套餐）。
- 已有可用项目。如果没有项目，请先**新建CodeArts项目**，例如项目名称为project01。
- 已添加当前账号对当前私有库的权限，请参考**配置私有依赖库权限**。

创建 Maven 类型私有依赖库并关联项目

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 在私有依赖库页面，单击右上角“新建制品仓库”。

步骤3 选择“本地仓库”，输入仓库名称maven01，选择“Maven制品类型”。

步骤4 单击“确定”。新建成功的Maven私有依赖库将显示在仓库视图中。

步骤5 在仓库视图中，单击对应的仓库名称maven01，单击“设置仓库”。

步骤6 选择“项目关联权限”页签，单击对应项目名所在操作列的图标，在弹框中勾选目标私有依赖库的名称maven01。

步骤7 单击“确定”。

----结束

在代码仓库中设置组件的版本

步骤1 使用华为云账号登录华为云控制台页面。

步骤2 单击页面左上角 ☰，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。

步骤3 单击“立即使用”，进入CodeArts服务首页。

步骤4 在顶部菜单选择“服务 > 代码托管”，进入代码托管服务。

步骤5 单击“新建仓库”。

步骤6 选择“归属项目”project01，单击“按模板新建”，单击“下一步”。

步骤7 搜索并勾选“Java Maven Demo”类型模板，单击“下一步”。

步骤8 输入“代码仓库名称”repo01，单击“确定”。

步骤9 进入代码仓库，单击“pom.xml”，查看组件配置。

步骤10 在组件配置页面中，<version>代码行中为当前组件的版本号（默认为1.0）。

单击页面右上方 ✎，可以修改版本号，修改完成后单击“确定”。

----结束

通过编译构建发布 Maven 私有组件到私有依赖库

步骤1 根据在代码仓库中设置组件的版本在代码仓库完成设置组件版本后，单击页面右上角“设置构建”，页面跳转至“新建构建任务”页面。

步骤2 在页面中选择“空白模板”，单击“确定”。

步骤3 单击“点击添加构建步骤”。搜索并添加步骤“Maven构建”。

步骤4 编辑步骤“Maven构建”。

- 工具版本按照实际选择，本文中选择“maven3.5.3-jdk8-open”。
- 找到以下命令行，删除命令行前的#。

```
#mvn deploy -Dmaven.test.skip=true -U -e -X -B
```

找到以下命令行，在命令行前添加#。

```
mvn package -Dmaven.test.skip=true -U -e -X -B
```

- 在“发布依赖包到CodeArts私有依赖库”一栏勾选“配置所有pom”，并在下拉列表中选择与项目已关联的Maven私有依赖库maven01。

步骤5 单击页面右上角“保存并执行”，执行构建任务。

----结束

在 Maven 私有依赖库的版本视图中查看归档的组件

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 选择目标私有依赖库，找到通过构建任务上传的Maven私有组件。

参考**在代码仓库中设置组件的版本**在代码仓库中**设置组件版本**，可将多个版本组件归档至私有依赖库。

步骤3 单击“版本视图”。

在包列表中，可以查看从编译构建中获取软件包的版本数和最新版本。

步骤4 单击“包名”，页面将显示该软件包最新版本的概览信息。

步骤5 选择“文件列表”页签，在列表中可以单击目标组件操作列中的，可将组件下载到本地。

步骤6 用户在本地对组件修改并设置新的版本号后，在目标私有依赖库中，单击“上传组件”，可将最新版本的组件上传至私有依赖库。

版本视图中的包列表显示对应组件最新上传的版本并统计版本归档过的数量。

----结束

3 通过编译构建任务发布/获取 NPM 私有组件

本文档介绍如何通过编译构建任务发布私有组件到NPM私有依赖库、如何从NPM私有依赖库获取依赖包完成编译构建任务。

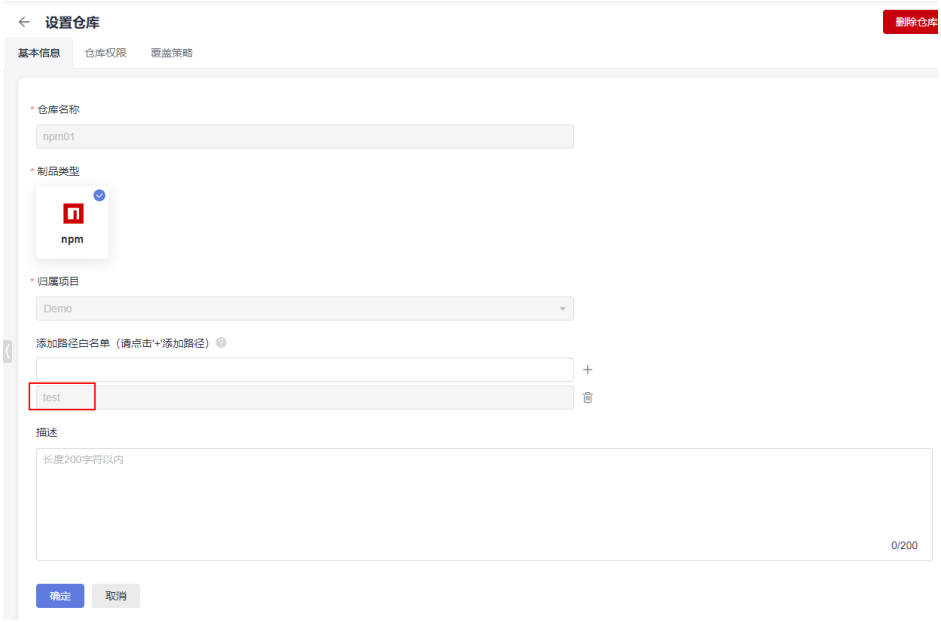
前提条件

- 该功能需要使用编译构建CodeArts Build服务，需要**购买CodeArts Build套餐**，或者**开通/购买软件开发生产线服务组合套餐**（如果已经购买了软件开发生产线服务组合套餐，则无需再单独购买CodeArts Build套餐）。
- 已有可用项目。如果没有项目，请先**新建CodeArts项目**，例如项目名称为project01。
- 已**创建NPM格式私有依赖库**。
- 已添加当前账号对当前私有库的权限，请参考**配置私有依赖库权限**。

发布私有组件到 NPM 私有依赖库

步骤1 下载私有依赖库配置文件。

1. 使用华为云账号**访问CodeArts Artifact的私有依赖库**。
2. 选择NPM私有依赖库。单击页面右侧“设置仓库”，记录仓库的路径。

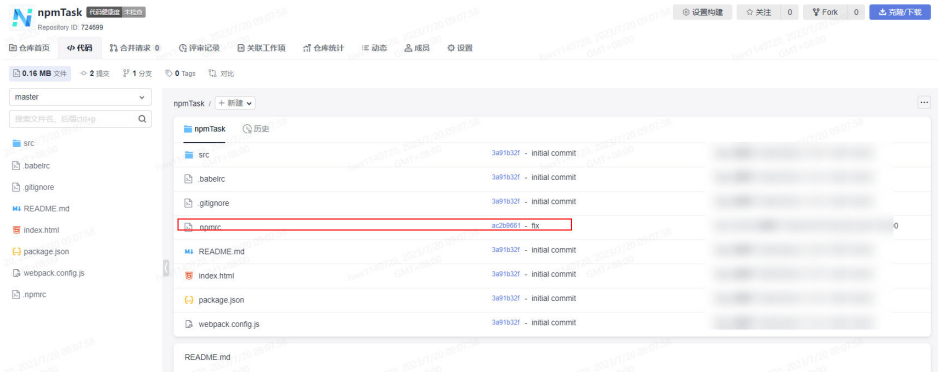


3. 单击“取消”返回私有依赖库页面，单击页面右侧“操作指导”。
4. 在弹框中单击“下载配置文件”。

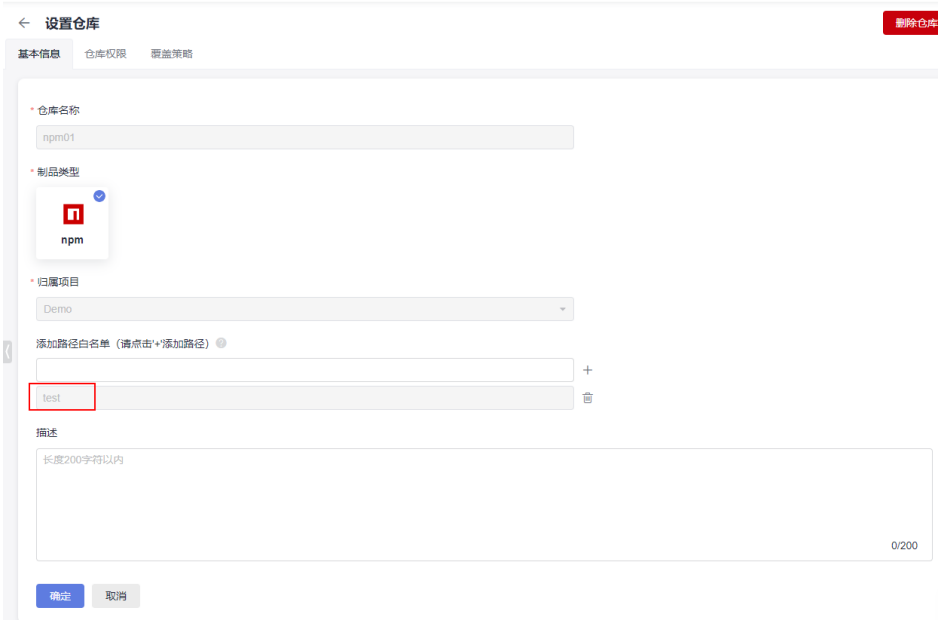
5. 在本地将下载的“npmrc”文件另存为“.npmrc”文件。

步骤2 配置代码仓库。

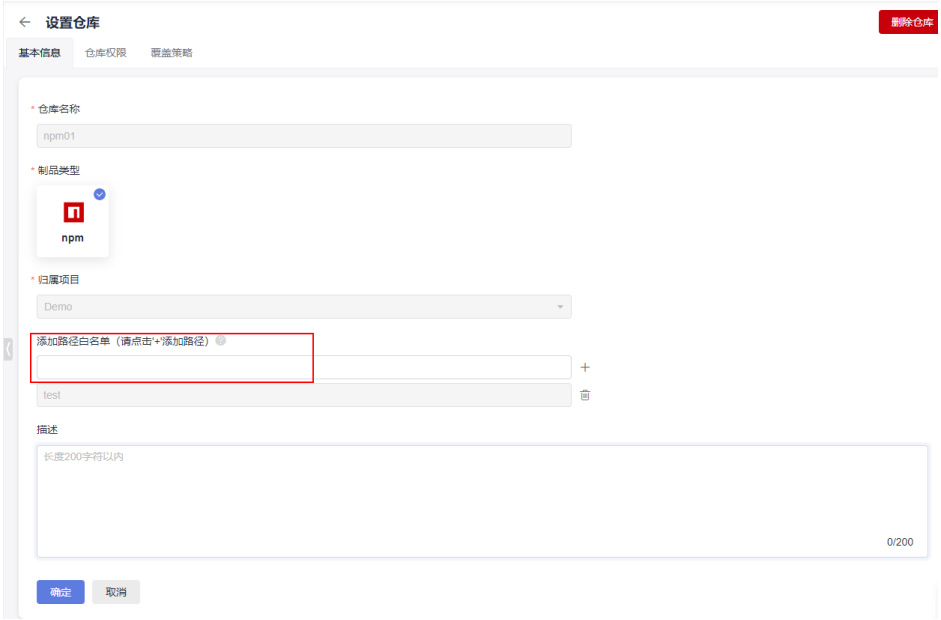
1. 使用华为云账号**登录华为云控制台页面**。
2. 单击页面左上角 ☰ ，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。
3. 单击“立即使用”，进入CodeArts服务首页。
4. 在顶部菜单选择“服务 > 代码托管”，进入代码托管服务。
5. 创建Node.js代码仓库（操作步骤请参考**创建云端仓库**）。本文使用模板“nodejs Webpack Demo”创建代码仓库。
6. 进入代码仓库，将“.npmrc”文件**上传至代码仓库**的根目录中。



7. 在代码仓库中找到“package.json”文件并打开，将在“编辑私有依赖库”页面中记录的路径信息添加到**name**字段对应的值中。



实际操作中，若出现**name**字段的值固定且不便修改的情况，则可以在“编辑私有依赖库”页面将该值配置到“添加路径”字段中。



步骤3 配置并执行编译构建任务。

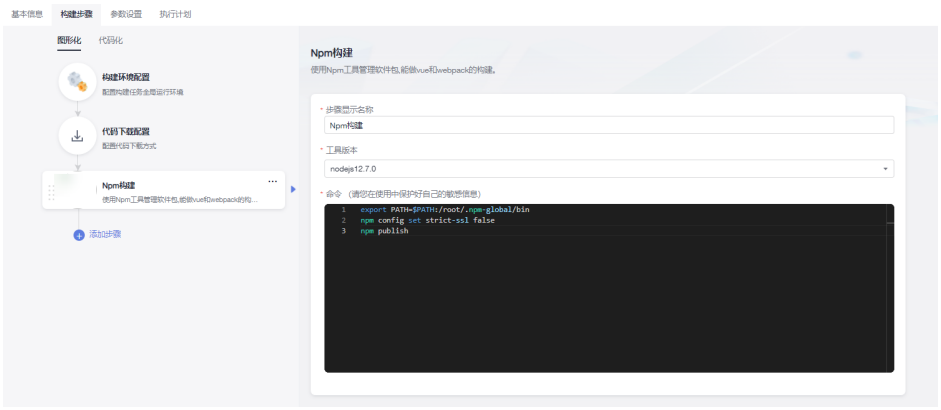
1. 在代码仓库中，单击页面右上角“设置构建”，页面跳转至“新建编译构建任务”页面。

   在页面中选择“空白构建模板”，单击“下一步”。

2. 添加步骤“Npm构建”。

3. 编辑步骤“Npm构建”。
   - 工具版本按照实际选择，本文中选择“nodejs12.7.0”。
   - 删除已有命令行，输入以下命令：

```
export PATH=$PATH:/root/.npm-global/bin
npm config set strict-ssl false
npm publish
```



4. 单击“新建并执行”，启动构建任务执行。
   待任务执行成功时，进入私有依赖库，可找到通过构建任务上传的NPM私有组件。

----结束

从 NPM 私有依赖库获取依赖包

以**发布私有组件到NPM私有依赖库**中发布的NPM私有组件为例，介绍如何从Npm私有依赖库中获取依赖包。

步骤1 配置代码仓库。

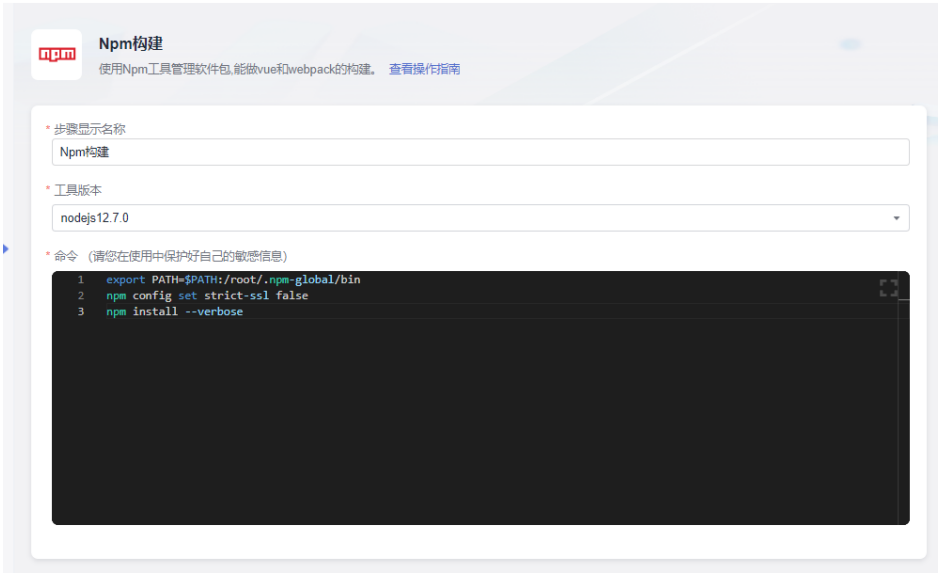
1. 使用华为云账号**登录华为云控制台页面**。
2. 单击页面左上角 ☰ ，在服务列表中选择“开发与运维 > 软件开发生产线 CodeArts”。
3. 单击“立即使用”，进入CodeArts服务首页。
4. 在顶部菜单选择“服务 > 代码托管”，进入代码托管服务。
5. 创建Node.js代码仓库（操作步骤请参考**创建云端仓库**）。本文使用模板“nodejs Webpack Demo”创建代码仓库。
6. 参考**发布私有组件到NPM私有依赖库**，获取“.npmrc”文件并上传至需要使用Npm依赖包的代码仓库根目录中。
7. 在代码仓库中找到“package.json”文件并打开，将依赖包配置到**dependencies**字段中，本文中配置的值为：
   "@test/vue-demo": "^1.0.0"

步骤2 配置并执行编译构建任务。

1. 在代码仓库中，单击页面右上角“设置构建”，页面跳转至“新建编译构建任务”页面。
   在页面中选择“空白构建模板”，单击“下一步”。
2. 添加步骤“Npm构建”。

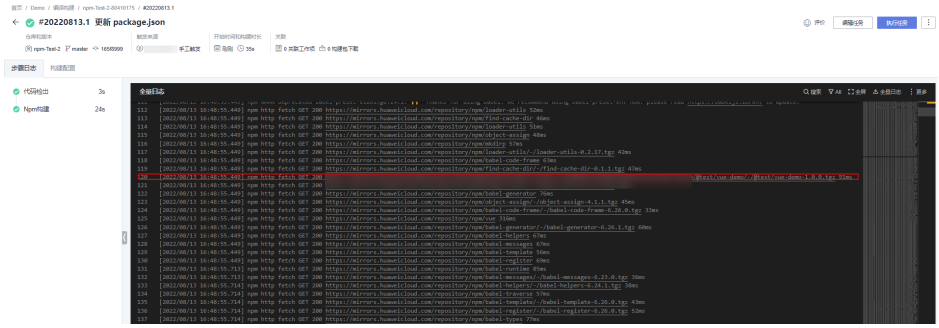
3. 编辑步骤“Npm构建”。
   - 工具版本按照实际选择，本文中选择“nodejs12.7.0”。
   - 删除已有命令行，输入以下命令：

```
export PATH=$PATH:/root/.npm-global/bin
npm config set strict-ssl false
npm install --verbose
```



步骤3 单击“新建并执行”，启动构建任务执行。

待任务执行成功时，查看构建任务详情，在日志中找到类似如下内容，说明编译构建任务从私有依赖库完成了依赖包下载并构建成功。



----结束

NPM 命令简介

在编译构建任务命令行中，还可以配置如下NPM命令，以完成其它功能：

- 删除私有依赖库中已存在的私有组件
  ```
  npm unpublish @scope/packageName@version
  ```
- 获取标签列表
  ```
  npm dist-tag list @scope/packageName
  ```
- 新增标签
  ```
  npm dist-tag add @scope/packageName@version tagName --registry registryUrl --verbose
  ```
- 删除标签
  ```
  npm dist-tag rm @scope/packageName@version tagName --registry registryUrl --verbose
  ```

命令行参数说明：

- scope：私有依赖库路径，查看方法请参考**发布私有组件到NPM私有依赖库**。
- packageName：“package.json”文件中，**name**字段中scope之后的部分。
- version：“package.json”文件中，**version**字段对应的值。
- registryUrl：私有库配置文件中的对应scope的私有库地址url。
- tagName：标签名称。

以**发布私有组件到NPM私有依赖库**发布的私有组件为例：

- **scope**对应的值为“test”。
- **packageName**对应的值为“vue-demo”。
- **version**对应的值为“1.0.0”。

因此，删除此组件的命令应为：

```
npm unpublish @test/vue-demo@1.0.0
```


4 通过编译构建任务发布/获取 Go 私有组件

本文档介绍如何通过编译构建任务发布私有组件到Go私有依赖库、如何从Go私有依赖库获取依赖包完成编译构建任务。

前提条件

- 该功能需要使用编译构建CodeArts Build服务，需要**购买CodeArts Build套餐**，或者**开通/购买软件开发生产线服务组合套餐**（如果已经购买了软件开发生产线服务组合套餐，则无需再单独购买CodeArts Build套餐）。
- 已有可用项目。如果没有项目，请先**新建CodeArts项目**，例如项目名称为project01。
- 已**创建GO格式私有依赖库**。
- 已添加当前账号对当前私有库的权限，请参考**配置私有依赖库权限**。

发布私有组件到 Go 私有依赖库

步骤1 下载私有依赖库配置文件。

1. 使用华为云账号**访问CodeArts Artifact的私有依赖库**。
2. 选择Go私有依赖库。单击页面右侧“操作指导”。
3. 在弹框中单击“下载配置文件”。

步骤2 配置代码仓库。

1. 进入代码托管服务。创建Go语言代码仓库（操作步骤请参考**创建云端仓库**）。本文中使用仓库模板“Go Web Demo”创建代码仓库。
2. 准备“go.mod”文件，并**上传至代码仓库**的根目录中。本文中使用的“go.mod”文件如下所示：


```
go.mod
1    module example.com/demo
```

步骤3 配置并执行编译构建任务。

1. 在代码仓库中，单击页面右上角“设置构建”，页面跳转至“新建编译构建任务”页面。

   在页面中选择“空白构建模板”，单击“下一步”。

2. 添加步骤“Go语言构建”。

3. 编辑步骤“Go语言构建”。
   - 工具版本按照实际选择，本文中选择“go-1.13.1”。
   - 删除已有命令行，打开在步骤**步骤1**中下载的配置文件，将文件中的“LINUX下配置go环境变量命令”复制到命令框中。
   - 将配置文件中go上传命令代码段复制到命令框中，并参考**Go Modules打包方式简介**替换命令行中的参数信息（本文打包版本为“v1.0.0”）。
4. 单击“新建并执行”，启动构建任务执行。

   待页面提示“构建成功”时，进入私有依赖库，可找到通过构建任务上传的Go私有组件。

----结束

从 Go 私有依赖库获取依赖包

以**发布私有组件到Go私有依赖库**中发布的Go私有组件为例，介绍如何从Go私有依赖库中获取依赖包。

步骤1 参考**发布私有组件到Go私有依赖库**，下载私有依赖库配置文件。

步骤2 进入代码托管服务，创建Go语言代码仓库（操作步骤请参考**创建云端仓库**）。本文中使用仓库模板“GoWebDemo”创建代码仓库。

步骤3 配置并执行编译构建任务。

1. 在代码仓库中，单击页面右上角“设置构建”，页面跳转至“新建编译构建任务”页面。

   在页面中选择“空白构建模板”，单击“下一步”。

2. 添加步骤“Go语言构建”。
3. 编辑步骤“Go语言构建”。
   - 工具版本按照实际选择，本文中选择“go-1.13.1”。
   - 删除已有命令行，打开已下载的私有依赖库配置文件，将文件中的“LINUX下配置go环境变量命令”代码段复制到命令框中。

- 根据下载版本，选择配置文件中“go下载命令”相应的命令行复制到命令框中，并将“<modulename>”参数值。（本文中为“example.com/demo”）。

步骤4 单击“新建并执行”，启动构建任务执行。

待页面提示“构建成功”时，查看构建任务详情，在日志中找到类似如下内容，说明编译构建任务从私有依赖库完成了依赖包下载并构建成功。

----结束

Go Modules 打包方式简介

本文采用Go Modules打包方式完成Go组件的构建与上传。

打包命令主要包括以下几部分：

1. 在工作目录中创建源文件夹。
   ```
   mkdir -p {module}@{version}
   ```
2. 将代码源拷贝至源文件夹下。
   ```
   cp -rf . {module}@{version}
   ```
3. 压缩组件zip包。
   ```
   zip -D -r [包名] [包根目录名称]
   ```
4. 上传组件zip包与“go.mod”文件到私有依赖库中。
   ```
   curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/{filePath} -T {{localFile}}
   ```

根据打包的版本不同，组件目录结构有以下几种情况：

- v2.0以下版本：目录结构与“go.mod”文件路径相同，无需附加特殊目录结构。
- v2.0以上（包括v2.0）版本：
  - “go.mod”文件中第一行以“/vX”结尾：目录结构需要包含“/vX”。例如，版本为v2.0.1，目录需要增加“v2”。
  - “go.mod”文件中第一行不以“/vN”结尾：目录结构不变，上传文件名需要增加“+incompatible”。

下面分别对不同的版本举例说明：

- **v2.0以下版本打包**。
  以下图所示“go.mod”文件为例。

  go.mod
  ```
  1  module example.com/demo
  ```

  a. 在工作目录中创建源文件夹。
     命令行中，参数“module”的值为“example.com/demo”，参数“version”自定义为1.0.0。因此命令如下：
     ```
     mkdir -p ~/example.com/demo@v1.0.0
     ```
  b. 将代码源拷贝至源文件夹下。
     参数值与上一步一致，命令行如下：
     ```
     cp -rf . ~/example.com/demo@v1.0.0/
     ```
  c. 压缩组件zip包。
     首先，使用以下命令，进入组件zip包所在根目录的上层目录。
     ```
     cd ~
     ```


然后，使用zip命令将代码压缩成组件包。命令行中，“包根目录名称”为“example.com”“包名”自定义为“v1.0.0.zip”，因此命令如下：

```
zip -D -r v1.0.0.zip  example.com/
```

d. 上传组件zip包与“go.mod”文件到私有依赖库中。

命令行中，参数“username”、“password”、“repoUrl”均可通过私有依赖库配置文件获取。

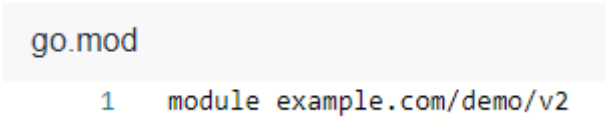
- 对于zip包，参数“filePath”为“example.com/demo/@v/v1.0.0.zip”，“localFile”为“v1.0.0.zip”。
- 对于“go.mod”文件，参数“filePath”为“example.com/demo/@v/v1.0.0.mod”，“localFile”为“example.com/demo@v1.0.0/go.mod”。

因此命令如下（参数username、password、repoUrl请参照私有依赖库配置文件自行修改）：

```
curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/@v/v1.0.0.zip -T v1.0.0.zip
curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/@v/v1.0.0.mod -T example.com/demo@v1.0.0/go.mod
```

- **v2.0以上版本打包，且“go.mod”文件中第一行以“/vX”结尾。**

以下图所示“go.mod”文件为例。



a. 在工作目录中创建源文件夹。

命令行中，参数“module”的值为“example.com/demo/v2”，参数“version”自定义为“2.0.0”。因此命令如下：

```
mkdir -p ~/example.com/demo/v2@v2.0.0
```

b. 将代码源拷贝至源文件夹下。

参数值与上一步一致，命令行如下：

```
cp -rf . ~/example.com/demo/v2@v2.0.0/
```

c. 压缩组件zip包。

首先，使用以下命令，进入组件zip包所在根目录的上层目录。

```
cd ~
```

然后，使用zip命令将代码压缩成组件包。命令行中，“包根目录名称”为“example.com”“包名”自定义为“v2.0.0.zip”，因此命令如下：

```
zip -D -r v2.0.0.zip  example.com/
```

d. 上传组件zip包与“go.mod”文件到私有依赖库中。

命令行中，参数“username”、“password”、“repoUrl”均可通过私有依赖库配置文件获取。

- 对于zip包，参数“filePath”为“example.com/demo/v2/@v/v2.0.0.zip”，“localFile”为“v2.0.0.zip”。
- 对于“go.mod”文件，参数“filePath”为“example.com/demo/v2/@v/v2.0.0.mod”，“localFile”为“example.com/demo/v2@v2.0.0/go.mod”。

因此命令如下（参数username、password、repoUrl请参照私有依赖库配置文件自行修改）：


```
curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/v2/@v/v2.0.0.zip -T v2.0.0.zip
curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/v2/@v/v2.0.0.mod -T example.com/demo/v2@v2.0.0/go.mod
```

- **v2.0以上版本打包，且“go.mod”文件中第一行不以“/vX”结尾。**
  以下图所示“go.mod”文件为例。

  go.mod

  ```
  1  module example.com/demo
  ```

  a. 在工作目录中创建源文件夹。
     命令行中，参数“module”的值为“example.com/demo”，参数“version”自定义为“3.0.0”。因此命令如下：
     ```
     mkdir -p ~/example.com/demo@v3.0.0+incompatible
     ```
  b. 将代码源拷贝至源文件夹下。
     参数值与上一步一致，命令行如下：
     ```
     cp -rf . ~/example.com/demo@v3.0.0+incompatible/
     ```
  c. 压缩组件zip包。
     首先，使用以下命令，进入组件zip包所在根目录的上层目录。
     ```
     cd ~
     ```
     然后，使用zip命令将代码压缩成组件包。命令行中，“包根目录名称”为“example.com”“包名”自定义为“v3.0.0.zip”，因此命令如下：
     ```
     zip -D -r v3.0.0.zip  example.com/
     ```
  d. 上传组件zip包与“go.mod”文件到私有依赖库中。
     命令行中，参数“username”、“password”、“repoUrl”均可通过私有依赖库配置文件获取。
     - 对于zip包，参数“filePath”为“example.com/demo/@v/v3.0.0+incompatible.zip”，“localFile”为“v3.0.0.zip”。
     - 对于“go.mod”文件，参数“filePath”为“example.com/demo/@v/v3.0.0+incompatible.mod”，“localFile”为“example.com/demo@v3.0.0+incompatible/go.mod”。

     因此命令如下（参数username、password、repoUrl请参照私有依赖库配置文件自行修改）：
     ```
     curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/@v/v3.0.0+incompatible.zip -T v3.0.0.zip
     curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/@v/v3.0.0+incompatible.mod -T example.com/demo@v3.0.0+incompatible/go.mod
     ```


5 通过编译构建任务发布/获取 PyPI 私有组件

本文档介绍如何通过编译构建任务发布私有组件到PyPI私有依赖库、如何从PyPI私有依赖库获取依赖包完成编译构建任务。

前提条件

- 该功能需要使用编译构建CodeArts Build服务，需要**购买CodeArts Build套餐**，或者**开通/购买软件开发生产线服务组合套餐**（如果已经购买了软件开发生产线服务组合套餐，则无需再单独购买CodeArts Build套餐）。
- 已有可用项目。如果没有项目，请先**新建CodeArts项目**，例如项目名称为project01。
- 已**创建PyPI格式私有依赖库**。
- 已添加当前账号对当前私有库的权限，请参考**配置私有依赖库权限**。

发布私有组件到 PyPI 私有依赖库

步骤1 下载私有依赖库配置文件。

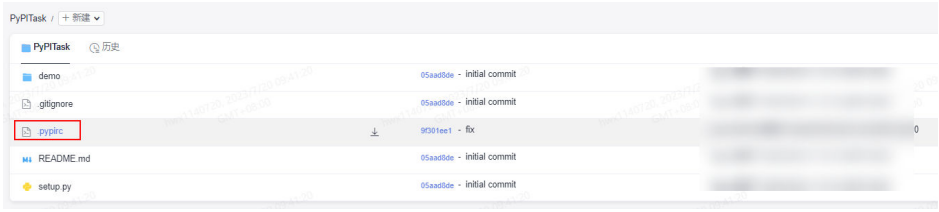
1. 使用华为云账号**访问CodeArts Artifact的私有依赖库**。
2. 选择PyPI私有依赖库。单击页面右侧“操作指导”。
3. 在弹框中找到“发布配置”，单击“下载配置文件”。

4. 在本地将下载的“pypirc”文件另存为“.pypirc”文件。

步骤2 配置代码仓库。

1. 进入代码托管服务，创建Python代码仓库（操作步骤请参考**新建仓库**）。本文使用模板“Python3 Demo”创建代码仓库。

2. 进入代码仓库，将“.pypirc”文件**上传至代码仓库**的根目录中。



步骤3 配置并执行编译构建任务。

1. 在代码仓库中，单击页面右上角“设置构建”，页面跳转至“新建编译构建任务”页面。

   在页面中选择“空白构建模板”，单击“下一步”。

2. 添加步骤“SetupTool构建”。

3. 编辑步骤“SetupTool构建”。
   - 工具版本按照实际选择，本文中选择“python3.6”。
   - 删除已有命令行，输入以下命令：

```
# 请保证代码根目录下有setup.py文件,下面命令将把工程打为whl包
python setup.py bdist_wheel
# 设置当前项目根目录下的.pypirc文件为配置文件
cp -rf .pypirc ~/
# 上传组件至pypi私有库
twine upload -r pypi dist/*
```

   如果上传时报证书问题，请在上述命令首行添加以下命令，设置环境变量跳过证书校验（twine低于或等于3.8.0版本，request低于或等于2.27版本时可忽略）：

```
export CURL_CA_BUNDLE=""
```

4. 单击“新建并执行”，启动构建任务执行。

   待任务执行成功时，进入私有依赖库，可找到通过构建任务上传的PyPI私有组件。

----结束

从 PyPI 私有依赖库获取依赖包

以**发布私有组件到PyPI私有依赖库**中发布的PyPI私有组件为例，介绍如何从PyPI私有依赖库中获取依赖包。

步骤1 下载私有依赖库配置文件。

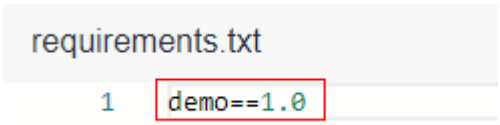
1. 使用华为云账号**访问CodeArts Artifact的私有依赖库**。
2. 选择PyPI私有依赖库，单击页面右侧“操作指导”。
3. 在弹框中找到“下载配置”，单击“下载配置文件”。

4. 在本地将下载的“pip.ini”文件另存为“pip.conf”文件。

步骤2 配置代码仓库。

1. 进入代码托管服务，创建Python代码仓库（操作步骤请参考**新建仓库**）。本文使用模板“Python3 Demo”创建代码仓库。
2. 进入代码仓库，将“pip.conf”文件上传至需要使用PyPI依赖包的代码仓库根目录中。
3. 在代码仓库中找到“requirements.txt”文件并打开（若没有请**新建文件**），将依赖包配置添加到此文件中，本文中配置的值为：

```
demo ==1.0
```



步骤3 配置并执行编译构建任务。

1. 在代码仓库中，单击页面右上角“设置构建”，页面跳转至“新建编译构建任务”页面。

   在页面中选择“空白构建模板”，单击“下一步”。
2. 添加步骤“Setup Tool构建”。

3. 编辑步骤“SetupTool构建”。
   - 工具版本按照实际选择，本文中选择“python3.6”。
   - 删除已有命令行，输入以下命令：

```
# 可以通过此命令设置当前项目根目录下的pip.conf文件为配置文件
export PIP_CONFIG_FILE=./pip.conf
# 下载pypi组件
pip install -r requirements.txt --no-cache-dir
```


步骤4 单击“新建并执行”，启动构建任务执行。

待任务执行成功时，查看构建任务详情，在日志中找到类似如下内容，说明编译构建任务从私有依赖库完成了依赖包下载并构建成功。

----结束

6 通过 Linux 命令行上传/获取 RPM 私有组件

本文档介绍如何Linux命令行上传私有组件到RPM私有依赖库、如何从RPM私有依赖库获取依赖包。

前提条件

- 已有可用的RPM组件。
- 已有可连通公网的Linux系统主机。
- 已**创建RPM格式私有依赖库**。
- 已添加当前账号对当前私有库的权限，请参考**配置私有依赖库权限**。

发布私有组件到 RPM 私有依赖库

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 选择RPM私有依赖库。单击页面右侧“操作指导”。

步骤3 在弹框中单击“下载配置文件”。

步骤4 在Linux主机中执行以下命令，上传RPM组件。

```
curl -u {{user}}:{{password}} -X PUT https://{{repoUrl}}/{{component}}/{{version}}/ -T {{localFile}}
```

其中，“user”、“password”、“repoUrl”来源于**上一步**下载的配置文件中“rpm上传命令”部分。

- user：位于**curl -u**与**-X**之间、“:”之前的字符串。
- password：位于**curl -u**与**-X**之间、“:”之后的字符串。
- repoUrl：“https://”与“/{{component}}”之间的字符串。

“component”、“version”、“localFile”来源于待上传的RPM组件。以组件“hello-0.17.2-54.x86_64.rpm”为例。

- component：软件名称，即“hello”。
- version：软件版本，即“0.17.2”。
- localFile：RPM组件，即“hello-0.17.2-54.x86_64.rpm”。

完整的命令行如下图所示：

```
curl -u                                                                    :                -X PUT
https://devrepo.devcloud.huaweicloud.com/artgalaxy/                                        _rpm_1/hello/0.17.2/ -T hello-0.17.2-54.x86_64.rpm
```

步骤5 命令执行成功，进入私有依赖库，可找到已上传的RPM私有组件。

----结束

从 RPM 私有依赖库获取依赖包

以**发布私有组件到RPM私有依赖库**中发布的RPM私有组件为例，介绍如何从Rpm私有依赖库中获取依赖包。

步骤1 参考**发布私有组件到RPM私有依赖库**，下载Rpm私有依赖库配置文件。

步骤2 打开配置文件，将文件中所有“{{component}}”替换为上传Rpm文件时使用的“{{component}}”值（本文档中该值为“hello”），并删除“rpm上传命令”部分，保存文件。

步骤3 将修改后的配置文件保存到Linux主机的“/etc/yum.repos.d/”目录中。

```
[                          yum.repos.d]# pwd
/etc/yum.repos.d
[                          yum.repos.d]# ll
total 20
-rw-r--r-- 1          737 Mar 12 11:04 cn-north                              _rpm_0.repo
-rw-r--r-- 1          235 Jan 25 23:00
-rw-r--r-- 1          186 Jan 25 22:59
-rw-r--r-- 1          234 Jan 25 23:00
drwxr-xr-x 4         4096 Dec 18 17:18 tmp
```

步骤4 执行以下命令，下载Rpm组件。其中，hello为组件的“component”值，请根据实际情况修改。

```
yum install hello
```

----结束

7 通过 Linux 命令行上传/获取 Debian 私有组件

本文档介绍如何通过Linux命令行上传私有组件到Debian私有依赖库、如何从Debian私有依赖库获取依赖包。

前提条件

- 已有可用的Debian组件。
- 已有可连通公网的Linux系统主机。
- 已**创建Debian格式私有依赖库**。
- 已添加当前账号对当前私有库的权限，请参考**配置私有依赖库权限**。

发布私有组件到 Debian 私有依赖库

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 选择Debian私有依赖库。单击页面右侧“操作指导”。

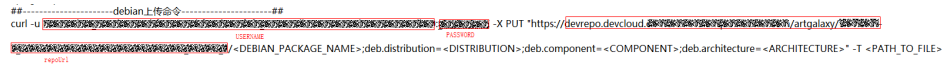
步骤3 在弹框中单击“下载配置文件”。

步骤4 在Linux主机中执行以下命令，上传Debian组件。

```
curl -u <USERNAME>:<PASSWORD> -X PUT "https:// <repoUrl>/
<DEBIAN_PACKAGE_NAME>;deb.distribution=<DISTRIBUTION>;deb.component=<COMPONENT>;deb.architecture=<ARCHITECTURE>" -T <PATH_TO_FILE>
```

其中“USERNAME”、“PASSWORD”、“repoUrl”来源于**步骤3**下载的配置文件中“Debian上传命令”部分。

- USERNAME：上传文件使用的用户名，可以从Debian配置文件中获取，参考示例图片。
- PASSWORD：上传文件使用的密码，可以从Debian配置文件中获取，参考示例图片。
- repoUrl：上传文件使用的url，可以从Debian配置文件中获取，参考示例图片。



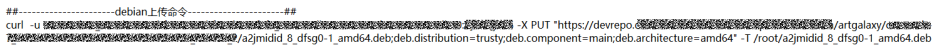
“DEBIAN_PACKAGE_NAME”、“DISTRIBUTION”、“COMPONENT”、“ARCHITECTURE”来源于待上传的Debian组件。

以组件“a2jmidid_8_dfsg0-1_amd64.deb”为例。

- DEBIAN_PACKAGE_NAME：软件包名称，例如：“a2jmidid_8_dfsg0-1_amd64.deb”。

- DISTRIBUTION：发行版本，例如：“trusty”。
- COMPONENT：组件名称，例如：“main”。
- ARCHITECTURE：体系结构，例如：“amd64”。
- PATH_TO_FILE：Debian组件的本地存储路径，例如：“/root/a2jmidid_8_dfsg0-1_amd64.deb”。

  完整的命令如下图所示：



步骤5 命令执行成功，进入私有依赖库，可找到已上传的Debian私有组件。

----结束

从 Debian 私有依赖库获取依赖包

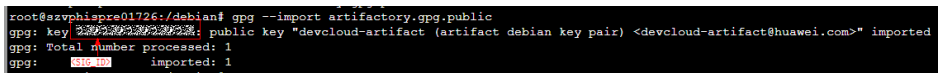
以**发布私有组件到Debian私有依赖库**中发布的Debian私有组件为例，介绍如何从Debian私有依赖库中获取依赖包。

步骤1 参考**发布私有组件到Debian私有依赖库**，下载Debian私有依赖库的“公钥”文件。

步骤2 导入gpg公钥。

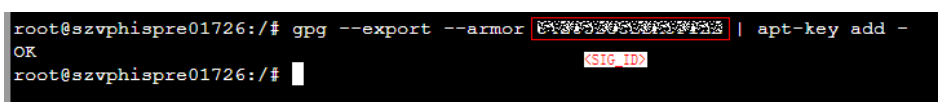
```
gpg --import <PUBLIC_KEY_PATH>
```

PUBLIC_KEY_PATH：Debian公钥的本地存储路径，例如：“artifactory.gpg.public”。



步骤3 apt导入公钥。

```
gpg --export --armor <SIG_ID> | apt-key add -
```



步骤4 apt仓库源添加。

打开配置文件（获取方法参考**发布私有组件到Debian私有依赖库**），将文件中所有“DISTRIBUTION”替换为上传Debian文件时使用的“COMPONENT”值（例如“main”），并根据下载的配置文件sources.list执行仓库源添加。

步骤5 仓库源添加后，使用如下命令更新仓库源。

```
apt-get update
```

步骤6 执行以下命令，下载Debian包。其中a2jmidid为包的“PACKAGE”值，请根据实际情况修改。

```
apt download a2jmidid
```

<PACKAGE>获取方法如下：

下载Debian组件的Packages源数据，以a2jmidid包为例。

----结束

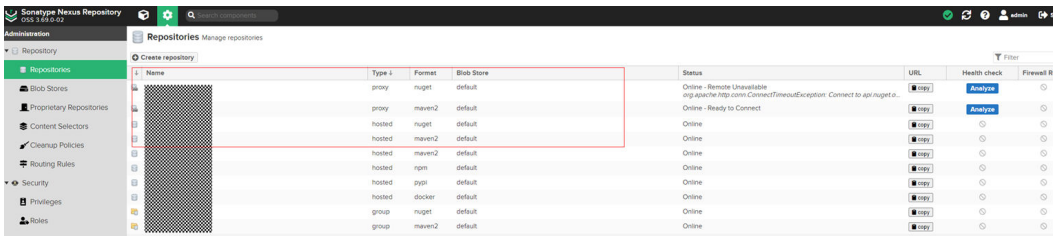
8 迁移 Nexus 中仓库数据到 CodeArts Artifact 私有依赖库

8.1 迁移前准备

CodeArts Artifact提供了批量迁移工具，便于用户将Nexus中hosted/proxy/group类型仓库的数据快速迁移至CodeArts Artifact的私有依赖库，从而实现统一且高效的操作与管理。

确认迁移的仓库类型和仓库语言类型

此处以Nexus3进行举例。登录Nexus3进入管理员页面，查看并确认需要迁移的仓库类型（Type列）和仓库语言类型（Format列），如下图所示。



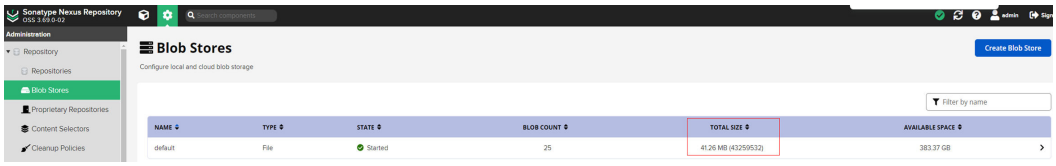
根据需要迁移的仓库类型（Type列）需要使用不同的迁移方式完成迁移，具体操作请参考如下章节：

- hosted类型仓库：参考 **迁移Nexus中hosted类型仓库的数据至CodeArts Artifact**进行操作。
- proxy类型仓库：参考**迁移Nexus中proxy类型仓库的数据到CodeArts Artifact**进行操作。
- group类型仓库：参考 **迁移Nexus中group类型仓库的数据到CodeArts Artifact** 进行操作。

确认 CodeArts Artifact 仓库容量

查看Blob Stores中显示的文件数量和大小（如下图所示），评估CodeArts Artifact的仓库剩余容量是否满足需求。如果剩余的CodeArts Artifact的仓库容量不满足使用，请

按需购买制品仓库CodeArts Artifact存储扩展，具体存储容量的查看和购买方法请参考**查看制品仓库存储容量及购买存储扩展**。



确认仓库使用场景

仓库提供给所有项目使用，建议单独创建一个项目，创建项目可以参考**新建CodeArts项目**。

如果已规划好项目，且不同项目下有不同的仓库，则无需创建。

8.2 迁移 Nexus 中 hosted 类型仓库的数据至 CodeArts Artifact

Nexus中hosted类型仓库迁移至CodeArts Artifact本地仓库，使用迁移工具仓库进行迁移。

该迁移工具的原理为：通过将Nexus上的包读取到输入流中，然后调用CodeArts Artifact的接口完成上传，完成对Nexus中hostsed类型仓库的迁移。

前提条件

- 运行环境需要为JDK8，可执行java -version命令查看运行环境。如需安装JDK可参考**安装JRE**。
- 迁移工具所在机器必须与Nexus、CodeArts Artifact服务网络连通。

步骤一：确定需要迁移的原仓库

确认迁移的仓库类型和仓库语言类型，针对hosted类型仓库，使用本章节操作步骤进行迁移。

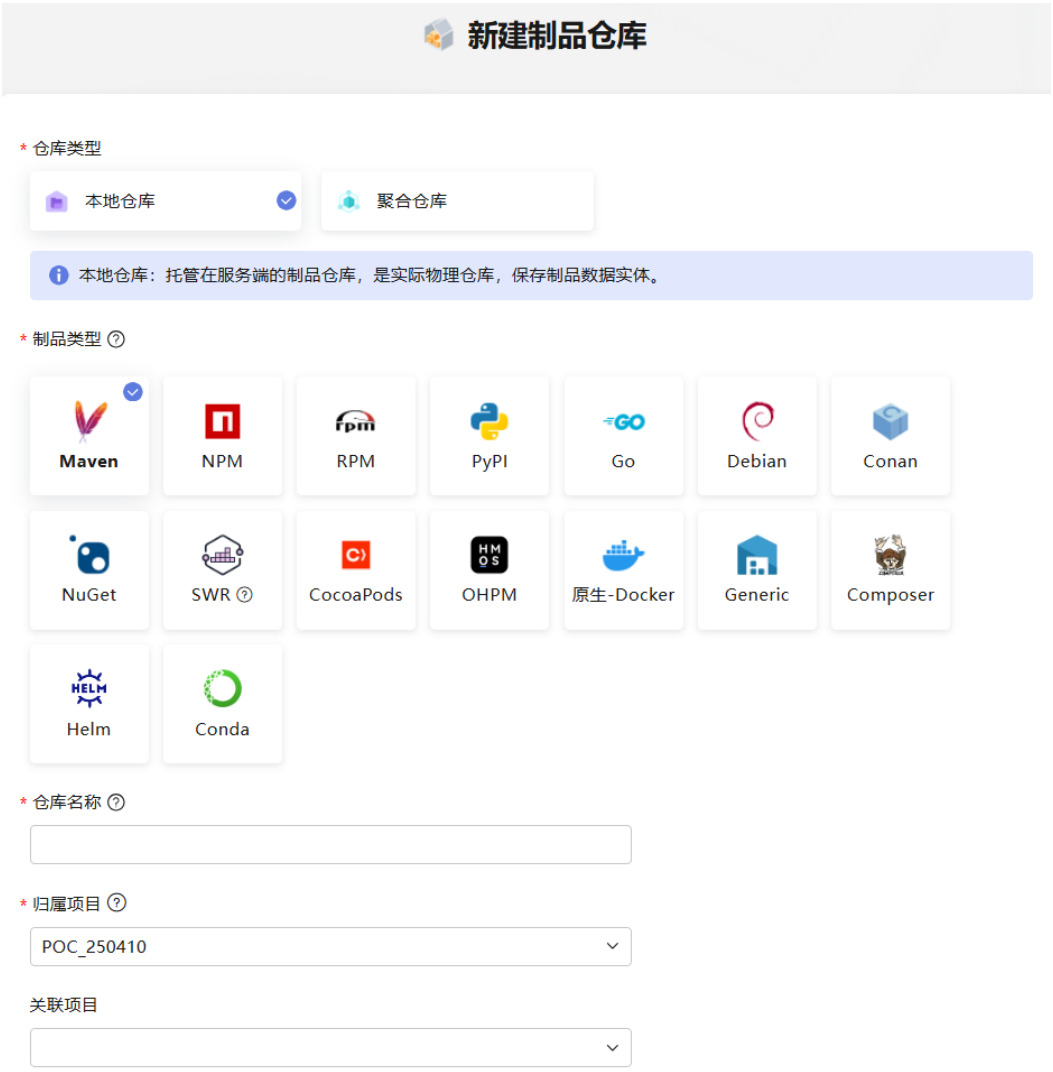
步骤二：在 CodeArts Artifact 创建私有依赖库（本地仓库）

根据**确认迁移的仓库类型和仓库语言类型**所确认的仓库类型，创建目标仓库，此处以创建Maven类型私有依赖库（本地仓库）为例。

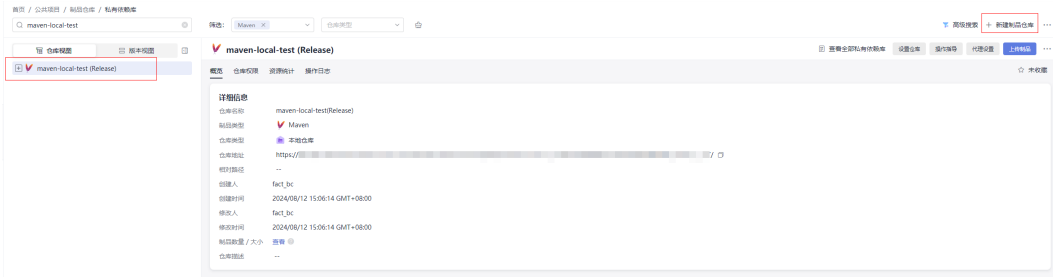
步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 在私有依赖库页面，单击右上方“新建制品仓库”。

步骤3 在“新建制品仓库”页面，仓库类型选择“本地仓库”，制品类型选择Maven，关联项目可以选择此公共项目，单击“确定”。新建成功的Maven私有依赖库将显示在仓库视图中。



步骤4 创建完成后应为此页面，再新建其他仓库，可以选择右上角的“新建制品仓库”。



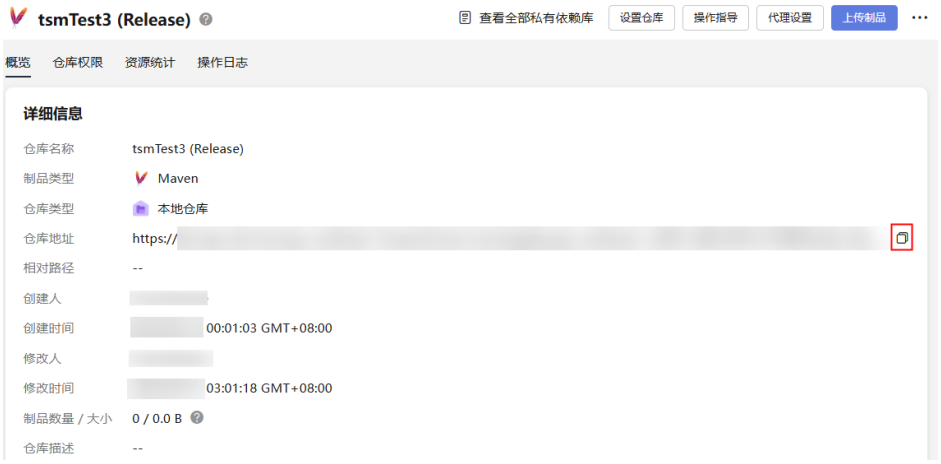
----结束

步骤三：获取创建的私有依赖库（本地仓库）的地址与配置

步骤1 获取私有依赖库地址。

1. 进入**步骤二：在CodeArts Artifact创建私有依赖库（本地仓库）**创建的私有依赖库，选择“仓库视图”。

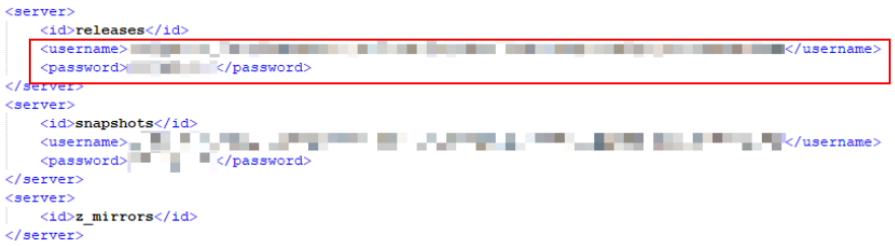
2. 单击仓库名称，右侧页面中仓库的“概览”页签中详细信息显示“仓库地址”。单击即可复制仓库地址。



步骤2 获取私有依赖库配置。

1. 单击页面右上方“操作指导”。
2. 在“操作指导”对话框中单击“下载配置文件”，下载配置文件**settings.xml**至本地。

3. 在本地打开配置文件**settings.xml**文件，在其中搜索并找到如下红框中的用户名与密码。



----结束

步骤四：配置私有依赖库（本地仓库）的迁移工具

步骤1 使用华为云账号访问CodeArts Artifact的私有依赖库。

步骤2 在左侧边栏中单击目标Maven私有依赖库的仓库名称。

步骤3 在仓库页面右上方···，单击“下载迁移工具”将迁移工具压缩包（迁移工具名称relocation-jfrog-20251016.1.jar、配置文件application-nexus.yaml）下载到本地。

步骤4 配置application-nexus.yaml文件中样例（如下表所示），该样例仅包含必须配置的参数，可直接在文件中搜索此参数名称，这些参数统一在relocation下。

表 8-1 application.yaml 关键配置说明

参数名称	样例	配置说明
name	nexus-to-artifact	迁移任务的名称，仅作为展示。
package_type	maven	迁移仓库的语言类型maven npm pypi go。 **须知** maven的私有依赖库分为release和snapshot两种包格式，如果原仓库混合在一起，需要对原仓库两次迁移，分别迁移release和snapshot到不同格式的maven仓库中
migrate_type	nexus3	迁移类型，nexus3/nexus2，根据不同的neuxs版本填写。
save_temp_dir	D:/tmp/xxx/	迁移缓存路径， 迁移maven时必填, "/"结尾, 此路径用户缓存maven的快照版本名称最后上传。
domain	http://{ip}:{port}	原仓库的域名，最后的路径需没有'/'。
repo	test_maven	原仓库名称，Nexus上的仓库名，通过Nexus页面上确认要迁移的仓库的NAME字段获取。
user_name	username	原仓库的用户名。
password	password	原仓库的密码。

参数名称	样例	配置说明
target_domain	https://{domain}/artgalaxy https://{domain}/artgalaxy/api/npm	目标仓库的域名。 通过私有库页面上仓库地址获取**步骤三：获取创建的私有依赖库（本地仓库）的地址与配置**。 • 例如仓库地址为：https://{domain}/artgalaxy/xx-north-xxx_xxxxxxxx_maven_1_388/ 则此处填写：https://{domain}/artgalaxy • 例如仓库地址为：https://{domain}/artgalaxy/api/npm/xx-north-xx_xxxxxxx_npm_6944/ 则此处填写：https://{domain}/artgalaxy/api/npm
target_repo	xx-north-xxx_xxxxxxxx_maven_1_388 xx-north-xx_xxxxxxx_npm_6944	目标仓库名称。 通过仓库地址获取，**步骤三：获取创建的私有依赖库（本地仓库）的地址与配置**。 • 例如仓库地址为：https://{domain}/artgalaxy/xx-north-xxx_xxxxxxxx_maven_1_388/ 则此处填写：xx-north-xxx_xxxxxxxx_maven_1_388 • 例如仓库地址为：https://{domain}/artgalaxy/api/npm/xx-north-xx_xxxxxxx_npm_6944/ 则此处填写：xx-north-xx_xxxxxxx_npm_6944
target_user_name	username	目标仓库用户名，可以通过**步骤三：获取创建的私有依赖库（本地仓库）的地址与配置** 获取。
target_password	password	目标仓库密码，可以通过**步骤三：获取创建的私有依赖库（本地仓库）的地址与配置** 获取。

最简易application-nexus.yaml

```
spring:
  main:
    web-application-type: none
relocation:
  # 迁移任务的名称，仅作为展示
  name: nexus-to-artifact
  # 迁移仓库的语言类型maven npm pypi go等
```

```
package_type: maven
# 迁移类型，支持governance jfrog nexus3/nexus2
migrate_type: "nexus3"
# 迁移缓存路径， 迁移maven时必填, "/"结尾, 此路径用户缓存maven的快照版本名称最后上传
save_temp_dir: "D:/tmp/xxx/"

# 原仓库参数， 仅jfrog nexus场景需要配置
# 原仓库的地址，最后的路径需没有'/'
domain: 'http://{domain}/artifactory'
# 原仓库的仓库名称
repo: 'test-maven'
# 原仓库的用户名
user_name: "username"
# 原仓库的密码
password: "password"

# 目标仓库参数
# 目标仓库地址，最后的路径需没有'/',从页面上获取
target_domain: 'https://{domain}/artgalaxy/xxxx/xxxx'
# 目标仓库的名称
target_repo: xxxxx
# 目标仓库用户名
target_user_name: 'username'
# 目标仓库密码
target_password: 'password'
```

步骤5 全量配置参数application.yaml 参数参考。

```
spring:
  main:
    web-application-type: none
relocation:
  # 迁移程序通用配置
  # 迁移程序核心线程数量
  corePoolSize: 20
  # 迁移程序最大线程池数量
  maxPoolSize: 40
  # 迁移程序线程池队列大小
  queueCapacity: 99999999
  # 迁移程序的最大迁移速度 单位 MB/s，默认值为20
  speed_limit: 20
  # jfrog 迁移场景参数，此参数表示迁移包修改时间从modifiedFrom 到 modifiedTo 时间之间的包（时间戳）
  modifiedFrom:
  modifiedTo:
  # 迁移任务的名称，仅作为展示
  name: jfrog-to-artifact
  # 迁移仓库的语言类型
  package_type: pypi
  # 迁移类型，支持governance jfrog nexus
  migrate_type: "jfrog"
  # 迁移缓存路径， 迁移maven时必填,"/"结尾
  save_temp_dir: "/xxxx/"

  # 原仓库参数， 仅jfrog nexus场景需要配置
  # 原仓库的地址，最后的路径需没有'/'
  domain: 'http://{domain}/artifactory'
  # 原仓库的仓库名称
  repo: 'test-pypi-source'
  # 原仓库的类型，默认为artifactory
  repo_type: artifactory
  # 原仓库的用户名
  user_name: "username"
  # 原仓库的密码
  password: "password"
  # 如果是jfrog场景，此处可填写原仓库的子路径，支持子路径迁移
  source_sub_path:

  # 目标仓库参数
  # 目标仓库地址，最后的路径需没有'/',从页面上获取
  target_domain: 'https://{domain}/artgalaxy/xxxx/xxxx'
```

```
# 目标仓库的名称
target_repo: xxxxx
# 目标仓库的类型，默认为artifactory
target_repo_type: artifactory
# 目标仓库用户名
target_user_name: 'username'
# 目标仓库密码
target_password: 'password'

# governance迁移参数 迁移python无需关注
# governance迁移场景的domain_id
domain_id: test009
# governance场景下是否需要调用governance服务，如果是true将不调用governance
call_governance_use_local: true
# governance 迁移仓库的语言仓类型
governance_type: npm
# governance实体包的存放路径
migrate_local_path: ""
# governance元数据文件，表示需要迁移的元数据文件
migrate_metadata_path: ""
# 回调governance的路径，此路径需要单独设置一个空路径
governance_save_path: ""
# 回调governance的url
governance_url: ""
# 用户的ak，用于回调governance接口
access_key_id: ""
# 用户的sk，用于回调governance接口
secret_access_key: ""
# 是否仅通过governance_save_path中的缓存信息回调governance
only_update_governance: false
# 迁移governance最大的文件数量，此数量为migrate_metadata_path中的元数据数目
migrate_max_num: -1
```

----结束

步骤五：执行迁移

步骤1 执行如下迁移脚本。

```
java -jar relocation-jfrog-20251016.1.jar --spring.config.location=application-nexus.yaml > /log/relocation-jfrog.log 2>&1 &
```

步骤2 进入对应的私有依赖库（本地仓库），查看hosted类型仓库的组件包是否上传成功。

----结束

8.3 迁移 Nexus 中 proxy 类型仓库的数据到 CodeArts Artifact

迁移Nexus中proxy类型仓库的数据至CodeArts Artifact，仅需要完成CodeArts Artifact上的私有依赖库（代理仓）配置即可完成仓库的迁移，后续使用新的代理仓即可。

其原理为：通过配置完成对开源社区仓库或三方依赖仓库的代理，替换Nexus上的proxy类型的仓库。

操作步骤

步骤1 **为私有依赖库聚合仓设置及添加代理。**

步骤2 进入CodeArts Artifact私有依赖库，在左侧边栏中选择已设置对应代理的仓库名称即可。

----结束

8.4 迁移 Nexus 中 group 类型仓库的数据到 CodeArts Artifact

迁移Nexus中group类型仓库的数据至CodeArts Artifact，仅需要完成CodeArts Artifact上的私有依赖库（聚合仓）配置即可完成仓库的迁移，后续使用新的聚合仓即可。

其原理为：通过配置完成对本地仓库代理仓库的聚合，替换Nexus中group类型的仓库。

前提条件

- 已创建聚合仓库，具体操作请参见**新建私有依赖库**创建聚合仓库）。
- 需要配置代理镜像仓网络连通网络环境，请联系环境管理员、运维人员或技术支持配置。

步骤一： 完成对 Nexus 中 group 仓库内需要迁移的 proxy 和 hosted 的迁移

参考**迁移Nexus中hosted类型仓库的数据至CodeArts Artifact**和**迁移Nexus中proxy类型仓库的数据到CodeArts Artifact** 完成对Nexus上group中的hosted和proxy类型仓库的迁移。

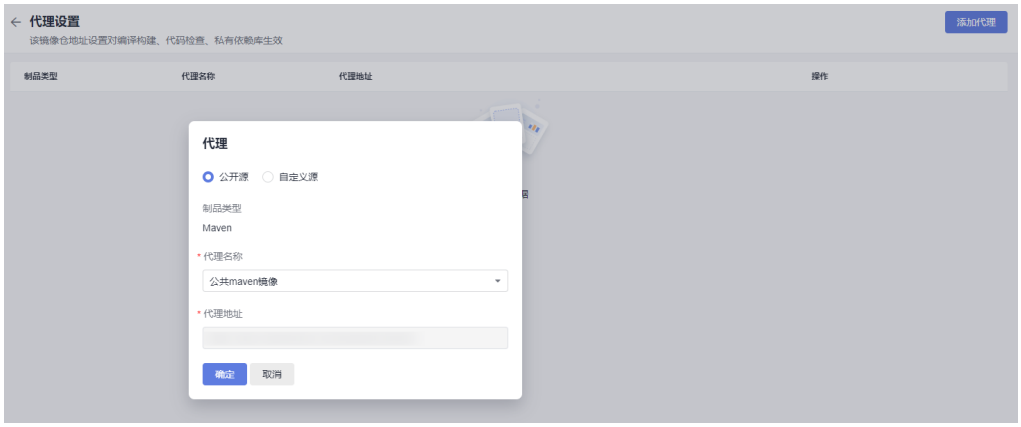
步骤二： 配置聚合仓库

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 在左侧边栏中选择对应聚合仓的仓库名称。

步骤3 单击页面右侧“代理设置”。

步骤4 单击“添加代理”，选择“公开源”或“自定义源”。



用户可以在“自定义源”中选择“第三方仓库”或“华为本地仓库”两种代理类型。

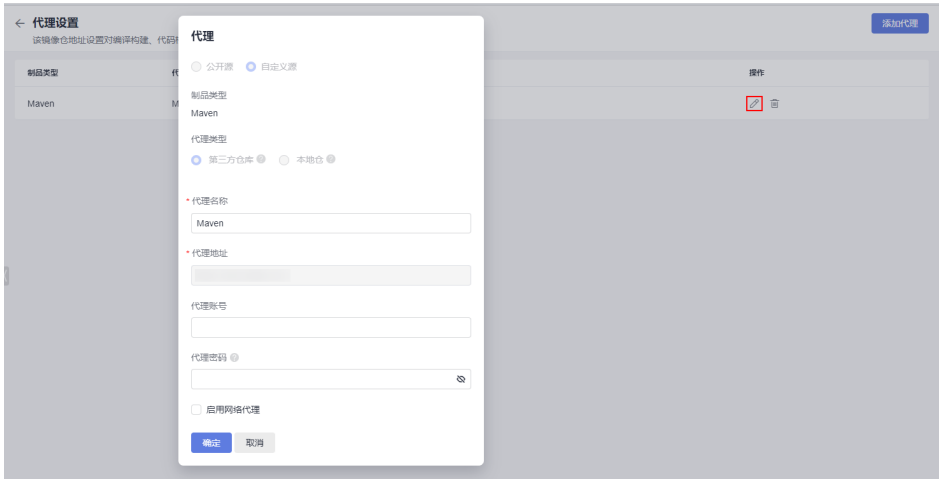
- 第三方仓库（设置第三方仓库或者由用户自行创建的仓库为代理源）

  用户选择第三方仓库后，单击“代理名称”的下拉列表，在下拉列表中选择自定义代理源，对应Nexus中的proxy。
- 华为本地仓库（设置华为本地仓库为代理源，仅能从有权限的仓库中选择），对应Nexus中的hosted类型。

  用户在代理名称的下拉列表中，可以选择私有依赖库中的本地仓库。

步骤5 单击“确定”，完成添加代理。

- 在代理列表中，单击操作列中的，可以修改代理名称、代理账号、代理密码。



说明

用户无法编辑本地仓库的代理源。

- 单击操作列中的，可以删除对应的代理。

----结束

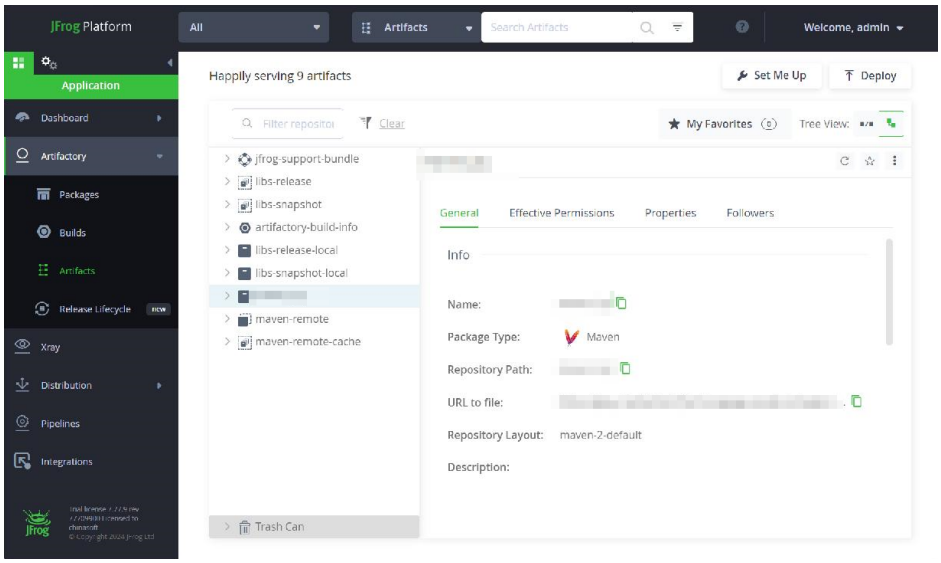
9 迁移 JFrog 仓库软件包至 CodeArts Artifact 私有依赖库

背景信息

JFrog仓库是一个用于存储和管理软件包的中央存储库，提供了一种集中式的方式来管理软件包，支持各种软件包管理工具，如Maven、Gradle、npm、NuGet等。CodeArts Artifact的私有依赖库提供了批量迁移工具，支持将JFrog仓库迁移至私有依赖库。本节介绍如何批量迁移JFrog仓库至私有依赖库。

待迁移的JFrog仓库示例如图9-1所示。

图 9-1 待迁移的 JFrog 仓库



前提条件

- 已有可用项目。如果没有项目，请先**新建CodeArts项目**，例如项目名称为project01。
- 已**创建Maven格式私有依赖库**。
- 依赖Java运行环境，需要安装JRE，请参考**安装JRE**。

步骤一：获取 CodeArts Artifact 的目标私有依赖库地址与配置

步骤1 获取私有依赖库地址。

1. 进入私有依赖库，选择“仓库视图”，并在左侧边栏中选择目标Maven私有依赖库。
2. 单击仓库名称，右侧页面中仓库的“概览”页签中详细信息显示“仓库地址”。单击即可复制仓库地址。

图 9-2 获取私有依赖库地址

步骤2 获取私有依赖库配置。

1. 单击页面右上方“操作指导”。
2. 在“操作指导”对话框中单击“下载配置文件”，下载配置文件**settings.xml**至本地。

图 9-3 下载配置文件

3. 在本地打开配置文件**settings.xml**，在文件中搜索并找到如下红框中的用户名与密码。

```
<server>
    <id>releases</id>
    <username>                                                    </username>
    <password>          </password>
</server>
<server>
    <id>snapshots</id>
    <username>                                                    </username>
    <password>          </password>
</server>
<server>
    <id>z_mirrors</id>
</server>
```

----结束

步骤二：配置迁移工具

步骤1 返回私有依赖库，单击页面右侧 ···并在下拉列表选择“下载迁移工具”，如图9-4所示。

图 9-4 下载迁移工具

步骤2 将迁移工具**MigrateTool.rar**包下载到本地，并执行以下命令，将**MigrateTool.rar**包解压并进入解压后的目录中。

```
unrar x MigrateTool.rar
cd MigrateTool/
```

步骤3 用记事本打开**MigrateTool.rar**包解压后目录中的**application.yaml**文件，配置表9-1所示参数。

表 9-1 配置迁移工具参数

参数名称	参数说明
package_type	JFrog源仓库类型，配置为“maven”。
repo_type	JFrog源仓库类型，配置为“jfrog”。
domain	JFrog源仓库地址，例如“http://*本地JFrog仓库IP.本地JFrog仓库端口*/artifactory”。
repo	需要迁移的JFrog源仓库名称，根据实际名称填写。
user_name	登录JFrog源仓库的账号，根据实际情况填写。

参数名称	参数说明
password	登录JFrog源仓库的密码，根据实际情况填写。
target_repo_type	迁移后的目标仓库类型，配置为“artifactory”。
target_domain	迁移后的目标仓库地址，配置为**图9-5**中“/artgalaxy/”前半段的1的信息。
target_repo	迁移后的目标仓库ID，配置为**图9-5**中“/artgalaxy/”后半段的2的信息。
target_user_name	迁移后的目标仓库账号，配置为从**步骤2.3**中获取的**username**。
target_password	迁移后的目标仓库密码，配置为从**步骤2.3**中获取的**password**。

图 9-5 迁移后的目标仓库详细信息

----结束

步骤三：执行迁移

执行以下命令迁移JFrog仓库至私有依赖库。

```
nohup java -jar /tools/relocation-jfrog.jar --spring.config.additional-location=./application-product.yaml
> /log/relocation-jfrog.log 2>&1 &
```

图 9-6 执行迁移

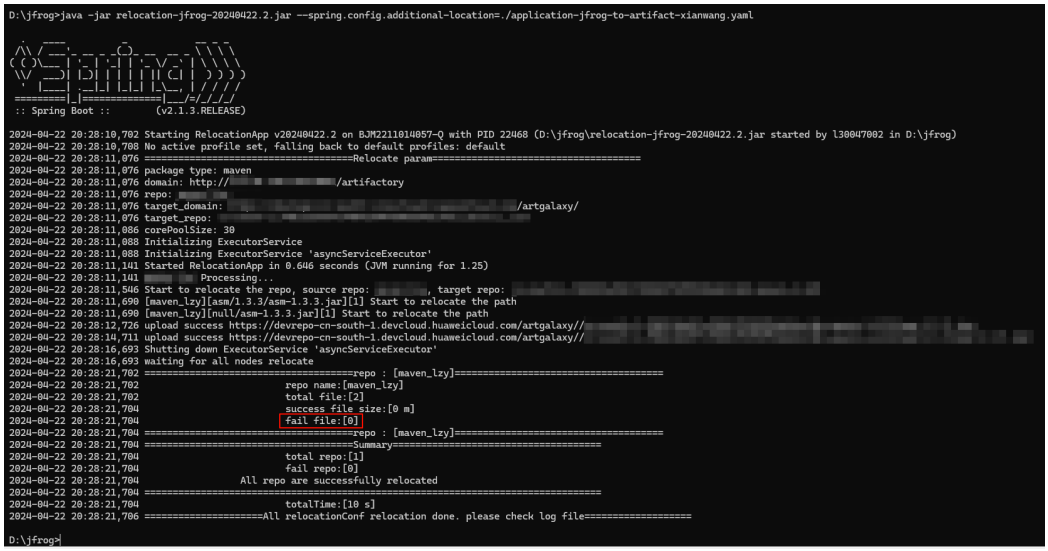


图9-6中“fail file”值为**0**时则表示迁移成功；否则迁移失败，可尝试重新执行迁移或联系客服寻求技术支持。

说明

- 该命令会处于后台运行。
- /tools/relocation-jfrog.jar：指定迁移工具路径。
- --spring.config.additional-location=./application-product.yaml：指定配置文件路径。
- /log/relocation-jfrog.log：指定迁移工具执行日志路径，可通过该日志查看迁移情况。

10 迁移本地仓库数据至 CodeArts Artifact 私有依赖库

10.1 迁移本地仓库数据至 CodeArts Artifact 私有依赖库概述

本地仓库是指在用户计算机上存储的软件包或依赖项的副本。当用户使用Maven、NPM构建工具来管理项目依赖时，这些工具会从远程仓库下载所需的库文件到本地仓库中。而CodeArts Artifact通过对开发过程中产生的依赖组件和最终产物进行存储和推送拉取权限的严格管控，更高效实现团队内协同开发。因此对于用户本地磁盘中已有Maven、NPM仓库数据迁移切换至CodeArts Artifact后即可在CodeArts Artifact进行更为统一高效的操作和运维。为此，CodeArts Artifact提供了批量迁移工具，方便用户将本地磁盘中的Maven仓库数据、NPM仓库数据批量快速迁移至CodeArts Artifact私有依赖库中的Maven私有依赖库、NPM私有依赖库。

约束与限制

仅支持迁移本地Maven仓库、NPM仓库数据至CodeArts Artifact私有依赖库。

准备工作

- 已有可用项目。如果没有项目，请先**新建CodeArts项目**。
- 添加当前账号对当前私有库的权限，请参考**配置私有依赖库权限**。
- 已在CodeArts Artifact**创建Maven、NPM格式私有依赖库**。
- 运行环境为Python3。
- 运行迁移工具所在的本地机器必须和CodeArts Artifact服务网络连通。

10.2 迁移本地 Maven 仓库数据至 CodeArts Artifact 私有依赖库

步骤一：获取 CodeArts Artifact 的目标 Maven 私有依赖库信息

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 在左侧边栏中单击目标Maven私有依赖库的仓库名称，进入仓库详细信息页面，可查看到“仓库地址”。

步骤3 单击仓库地址右侧即可复制该地址。

步骤4 单击页面右上方“操作指导”，在“操作指导”对话框中单击“下载配置文件”，将配置文件settings.xml下载到本地。

在本地打开配置文件，在文件中搜索并找到用户名和密码。

----结束

步骤二：配置迁移工具

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 在左侧边栏中选择目标Maven私有依赖库。

步骤3 单击仓库名称，在页面右上方单击···，然后在下拉选项中单击“下载迁移工具”将迁移工具压缩包“MigrateTool.zip”下载到本地并解压，解压后获取“uploadArtifact.py”（迁移工具）、“artifact.conf”（配置文件）。

步骤4 参考如下样例配置“artifact.conf”文件，该样例中仅列举必要的配置参数，其他参数可删除。

```
[artifact]
packageType = 组件类型，设置为Maven
userInfo = username:password（步骤4中获取的用户名与密码）
repoRelease = 私有依赖库地址（步骤3中获取的私有依赖库仓库地址）
repoSnapshot = 私有仓库地址（步骤3中获取的私有依赖库仓库地址），Maven组件类型为Snapshot时需要配置该参数
srcDir = 需要迁移的Maven本地仓库组件的目录路径，用户自定义，例如：C:\Users\xxxxxx\repository
```

----结束

步骤三：执行迁移

执行**步骤3**获取的迁移工具，即运行如下命令：

```
python uploadArtifact.py
```

步骤四：验证迁移结果

进入CodeArts Artifact对应的目标Maven私有依赖库，查看本地Maven仓库数据是否上传成功。

如果迁移失败，可尝试重新执行迁移步骤或联系客服寻求技术支持。

10.3 迁移本地 NPM 仓库数据至 CodeArts Artifact 私有依赖库

步骤一：获取 CodeArts Artifact 的目标 NPM 私有依赖库信息

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 在左侧边栏中单击目标NPM私有依赖库的仓库名称，进入仓库详细信息页面，可查看到“仓库地址”。

步骤3 单击仓库地址右侧即可复制该地址。

步骤4 单击页面右上方“操作指导”，在弹框中单击“下载配置文件”，将npmrc文件下载到本地。

步骤5 在本地打开配置文件，在文件中找到“_auth”字段的值并进行base64解码。

----结束

步骤二：配置迁移脚本

步骤1 使用华为云账号**访问CodeArts Artifact的私有依赖库**。

步骤2 在左侧边栏中选择目标NPM私有依赖库。

步骤3 单击仓库名称，在页面右上方单击···，然后在下拉选项中单击“下载迁移工具”将迁移工具压缩包“MigrateTool.zip”下载到本地并解压，解压后获取“uploadArtifact.py”（迁移工具）、“artifact.conf”（配置文件）。

步骤4 配置artifact.conf文件中如下样例中的参数，其他参数可删除。

```
[artifact]
packageType = 组件类型，设置为npm
userInfo = npm仓库下的配置文件npmrc中通过base64解密后的_auth字段的值（参见步骤5）
repoRelease = 私有依赖库地址（步骤1中获取的私有依赖库仓库地址）
repoSnapshot = 保留为空
srcDir = 需要迁移的NPM本地仓库组件的目录路径，用户自定义，例如：C:\Users\xxxxxx\repository
```

----结束

步骤三：执行迁移

执行**步骤3**获取的迁移工具，即运行如下命令：

```
python uploadArtifact.py
```

步骤四：验证迁移结果

进入CodeArts Artifact对应的目标NPM私有依赖库，查看本地NPM仓库数据是否上传成功。

如果迁移失败，可尝试重新执行迁移步骤或联系客服寻求技术支持。

11 CodeArts Artifact 权限配置最佳实践

方案概述

制品仓库分为软件发布库和私有依赖库两种类型的仓库。软件发布库提供了项目级的权限管理，私有依赖库提供项目级的权限管理与仓库级的权限管理，详情可参考**配置软件发布库权限**和**配置私有依赖库权限**。

本文以**私有依赖库权限管理**为例，介绍在**私有依赖库**内快速管理权限的方式，实现对单个仓库进行单独权限管理，以及按照项目维度进行权限管理。

约束与限制

- 项目管理者默认拥有全部操作权限，无法修改其权限范围。
- 创建的**自定义角色**在CodeArts Artifact无预置权限，可以联系项目管理者配置角色类型及对应资源的相应操作权限。
- 项目管理者、项目经理、测试经理默认具有“权限配置”的权限，可以配置其他成员角色在私有依赖库的操作权限。其他角色之前如有“权限配置”权限，可以继续在软件发布库内给其他角色配置权限。

前提条件

- 已**开通并授权使用CodeArts Artifact**。
- 已**新建CodeArts项目**，选择“Scrum”模板，命名为“Scrum”。
- 已将用户加入CodeArts项目“Scrum”使其成为项目成员，并为其配置角色，具体操作请参见**添加CodeArts项目成员**。
- 需要具有“权限配置”操作权限（项目管理员、项目经理、测试经理角色默认具有）才可配置其他成员角色在私有依赖库的的操作权限。
- 在项目名为“Scrum”的项目中，已创建名称为“pypi_test”的私有依赖库，具体操作请参见**创建私有依赖库**。

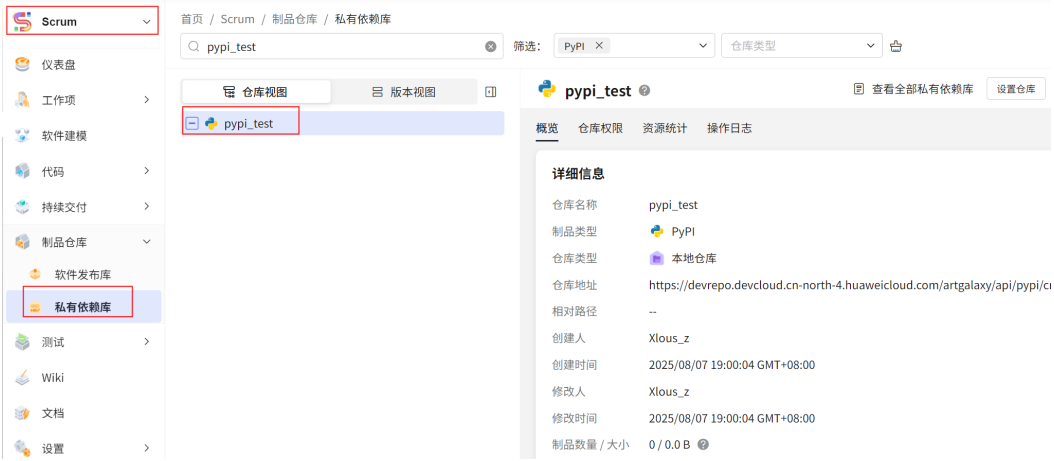
配置私有依赖库项目级权限

制品仓库服务支持在项目下统一配置项目各角色对当前项目下私有依赖库的默认操作权限（可参考**配置私有依赖库项目级权限**）。

步骤1 具有“权限配置”权限的用户**访问CodeArts Artifact的私有依赖库**。

步骤2 选择“私有依赖库”页签，页面展示服务下已创建的所有私有依赖库，如下图所示。

步骤3 在私有依赖库列表中，单击仓库名称“pypi_test”跳转到该仓库所在项目的私有依赖库详情页面，如下图所示。

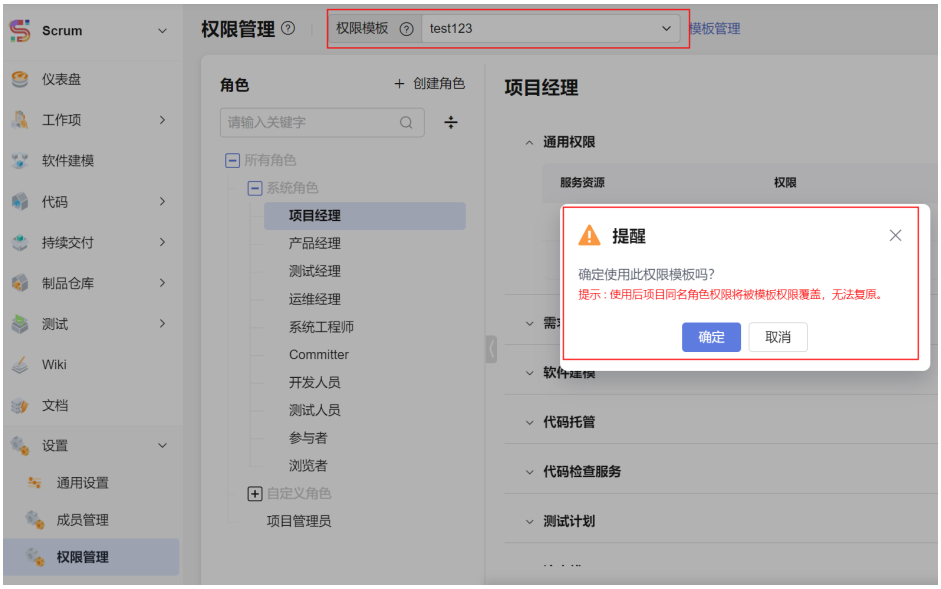


步骤4 在左侧导航栏选择“设置 > 权限管理”，进入项目权限管理页面。可通过以下两种方式进行项目级权限配置。

- 方式一：在“角色”列表中单击需要配置权限的角色，在右侧服务权限列表中选择“制品仓库”，然后单击页面底部“编辑”进入权限编辑状态，根据需要勾选或取消勾选相应权限，单击“保存”，完成权限配置。

- 方式二：在页面顶部“权限模板”下拉框中选择要应用的模板（权限模板的创建和管理方法请参见**管理CodeArts项目权限模板**），在弹框中单击“确定”，即可

复用权限模板中的模板进行一键配置权限，如下图所示。**使用权限模板后项目中同名角色的权限将被模板权限覆盖，无法复原，请谨慎操作**。



----结束

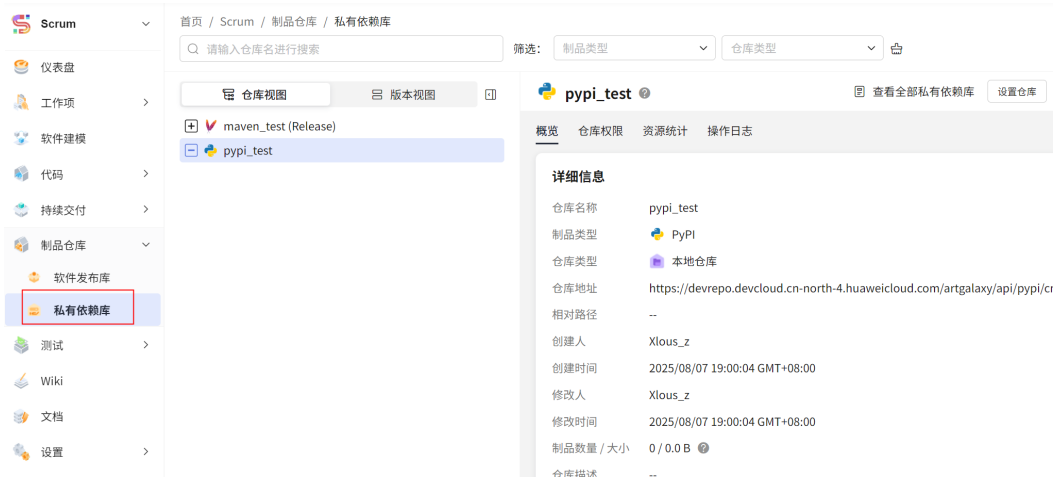
配置私有依赖库仓库级权限

制品仓库服务除了支持在项目下统一配置项目各角色对当前项目下私有依赖库的默认操作权限（可参考**配置私有依赖库项目级权限**）外，也支持单独配置对应私有依赖库的仓库权限。

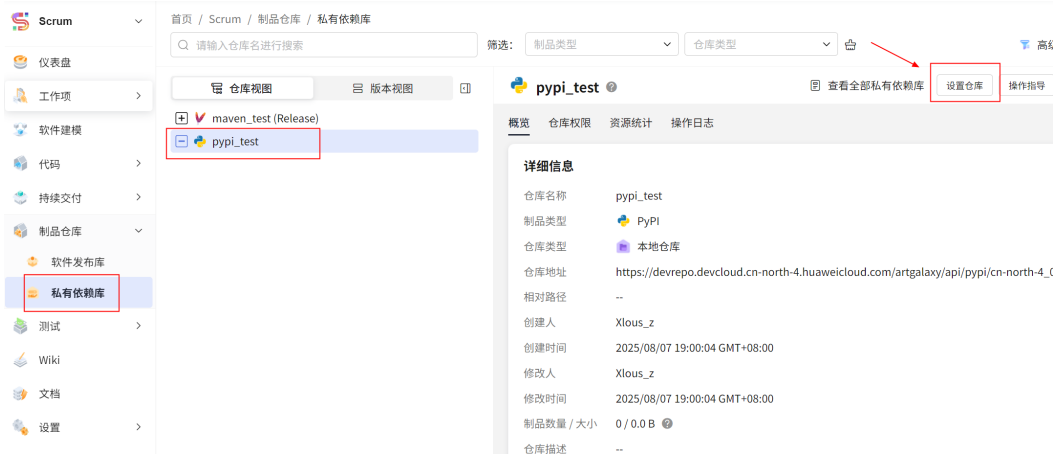
- 新建成功的私有依赖库默认对接项目下“设置 > 权限管理”的角色权限，即通过**私有依赖库项目级权限配置**修改后的角色权限，会同步到私有依赖库的仓库权限。
- 当用户没有在对应私有库下修改相关角色的仓库权限，通过**私有依赖库项目级权限配置**修改该角色的权限会同步到对应私有依赖库的仓库权限。
- 当用户在对应私有库下修改了相关角色的仓库权限，通过**私有依赖库项目级权限配置**修改该角色的权限将不会同步到对应私有依赖库的仓库权限，请在仓库下进行后续该角色的权限修改。

步骤1 **访问CodeArts Artifact的私有依赖库**。

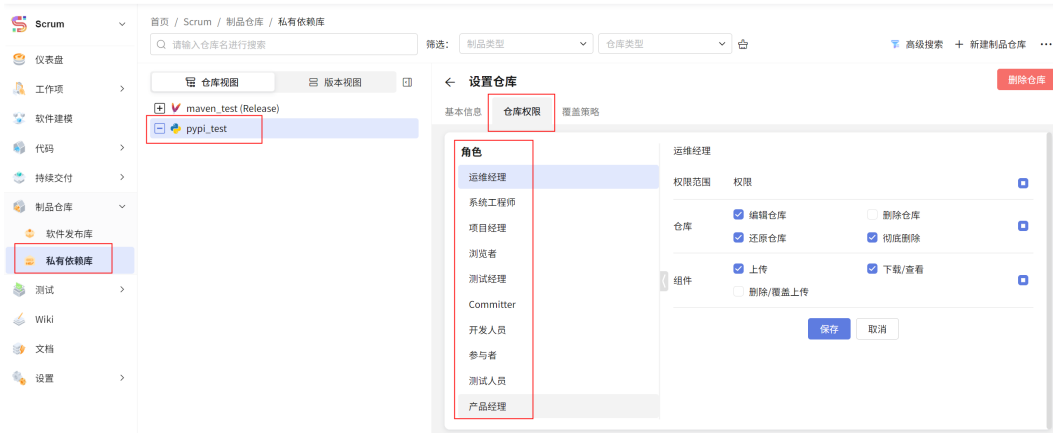
步骤2 在仓库列表中选择目标仓库。



步骤3 在页面右侧单击“设置仓库”。



步骤4 选择“仓库权限”页签，当前项目下的角色显示在页面中。



步骤5 在角色列表中，单击需要修改权限的角色，勾选或取消勾选相关权限。

- 新建成功的私有依赖库默认对接项目下“设置 > 权限管理”的角色权限，即通过**私有依赖库项目级权限配置**修改后的角色权限，会同步到私有依赖库的仓库权限。

- 当用户没有在对应私有库下修改相关角色的仓库权限，通过**私有依赖库项目级权限配置**修改该角色的权限会同步到对应私有依赖库的仓库权限。
- 当用户在对应私有库下修改了相关角色的仓库权限，通过**私有依赖库项目级权限配置**修改该角色的权限将不会同步到对应私有依赖库的仓库权限，请在仓库下进行后续该角色的权限修改。

步骤6 单击“保存”，私有依赖库仓库级权限配置完成。

项目中各成员角色**访问CodeArts Artifact的私有依赖库**，即可在私有依赖库进行已拥有权限的相应操作。

----结束