# 代码托管

# 常见问题

**文档版本** 01

**发布日期** 2024-11-11

**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。

# 华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

# 目 录



# 1 认证鉴权问题

## 1.1 TLS 协议握手失败并报错"ssl handshake failure"

### 问题现象

本地执行如下命令，与指定服务器建立TLS1.0连接并获取其证书信息。

```
openssl s_client -connect test.com:443 -tls1
```

会出现如下报错信息：

```
CONNECTED(00000003)
140155533838224:error:1409442E:SSL routines:ssl3_read_bytes:tlsv1 alert protocol version:s3_pkt.c:1493:SSL alert number 70
140155533838224:error:1409E0E5:SSL routines:ssl3_write_bytes:ssl handshake failure:s3_pkt.c:659:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 7 bytes and written 0 bytes
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1
    Cipher    : 0000
    Session-ID:
    Session-ID-ctx:
    Master-Key:
    Key-Arg   : None
    Krb5 Principal: None
    PSK identity: None
    PSK identity hint: None
    Start Time: 1720443876
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
---
```

### 原因分析

CodeArts Repo当前支持TLS1.2、TLS1.3协议版本。

## 处理方法

**步骤1** 在Git Bash客户端执行如下命令，查看您的Git版本。

```
git --version
```

**步骤2** 如果您的Git版本低于2.6.0，请将您本地的Git客户端升级到**最新版本**，最新版本的Git客户端默认支持TLSv1.2协议。如果您的Git版本不低于2.6.0，您也可以使用如下命令指定TLS协议的版本。

```
openssl s_client -connect test.com:443 -tls1_2
```

**----结束**

**表 1-1** 已知的不兼容客户端

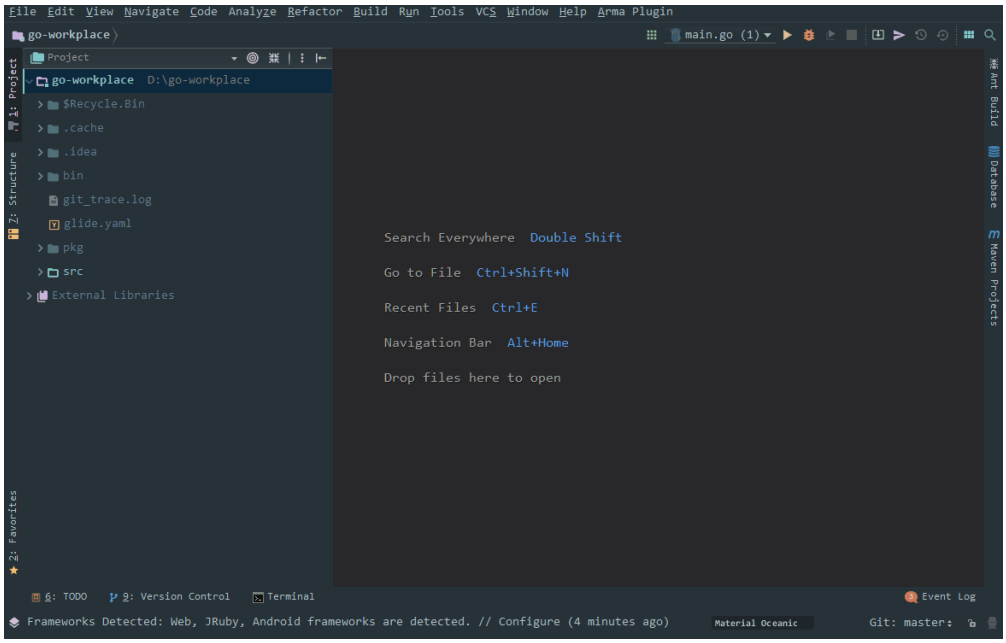
| 客户端 | 说明 |
| --- | --- |
| Git Credential Manager for Windows< v1.14.0 | 默认不支持TLSv1.2可以通过更新到**最新版Git**客户端来解决，最新版版本的GitBash自带Git-Credential-Manager功能，只需要在安装时勾选即可。 |
| Git on Red Hat 5 | • 不支持TLSv1.2；<br>• 建议Red Hat 5的用户升级到更新版本的操作系统。 |
| Git on Red Hat 6 | 默认不支持，升级版本至6.8（或更高）可支持。 |
| Git on Red Hat 7 | 默认不支持，升级版本至7.2（或更高）可支持。 |
| JDK7 | 默认情况下为TLSv1，运行的任何客户端（例如Eclipse自带的JGit这样的流行客户端）都会受到影响，可明确选择TLSv1.2来解决。 |
| JDK6及以下 | • 不支持TLSv1.2；<br>• 建议JDK6及以下的用户升级到较新版本的JDK。 |
| Visual Studio | • Visual Studio附带了特定版本的Git for Windows和Git Credential Manager for Windows（GCM）；<br>• Microsoft已更新Visual Studio 2017的最新版本以与TLSv1.2 Git服务器配合使用；<br>• 建议Visual Studio用户通过单击产品内通知标志或直接从IDE检查更新来升级到最新版本；<br>• Microsoft已在Visual Studio开发人员社区支持论坛上提供了其他指导。 |

## 第三方 Git 图形客户端

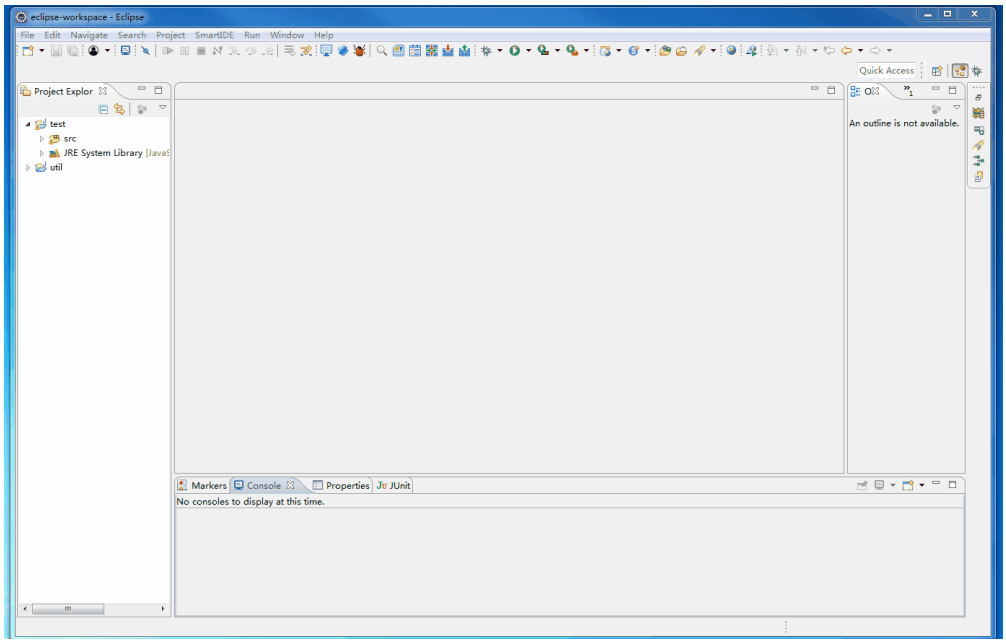
如果您使用的是Eclipse、JetBrains、SourceTree、VSCode、Tower、TortoiseGit等第三方Git客户端时，建议下载最新的GitBash客户端，在工具中设置使用原生(Native)的Git，而不是Build-in类型。

- IDEA设置方式




- Eclipse设置方式




# 1.2 升级 CodeArts Repo 的 SSH 功能

代码托管服务SSH功能下线部分不安全的加密算法。

当前KEX (Key Exchange)和MAC(消息验证码)不再支持以下算法。

KEX (Key Exchange)不支持算法如下：

- diffie-hellman-group18-sha512
- diffie-hellman-group-exchange-sha1
- diffie-hellman-group-exchange-sha256

MAC(消息验证码)不支持算法如下：

- hmac-md5
- hmac-md5-96
- hmac-sha1-etm@openssh.com
- hmac-sha1-96-etm@openssh.com
- hmac-md5-etm@openssh.com
- hmac-md5-96-etm@openssh.com
- umac-64@openssh.com
- umac-128@openssh.com

升级您的的代码提交工具至最新版本（如：git bash，eclipse，idea等），新版本默认使用更为安全的算法。

如有疑问请联系技术支持工程师。

# 1.3 在一台电脑上，如何配置多个 SSH Key?

## 场景描述

开发人员通常只会生成一个SSH Key，名字叫**id_rsa**，然后提交到多个不同的网站。

但是也存在另一种需要，在同一个网站上，注册了两个用户名，通常网站不会允许为这两个用户名，配置同一个SSH Key，这时候就会有些麻烦。

## 操作步骤

**步骤1** 在本地Git仓库生成两个不同的SSH Key。

```
ssh-keygen -t rsa -C "email"
Generating public/private rsa key pair.
Enter file in which to save the key (~/.ssh/id_rsa):<不要直接回车，填写自己定义的名字>
Enter passphrase(empty for no passphrase):<不要直接回车，填写自己定义的密码>
```

**说明**

这是第一个关键，如果要生成2个Key，这里写成：**github_1**和**github_2**这样，就生成了2个证书。

**步骤2** 用不同的账号，上传两个不同的证书。

读取*.pub的内容，粘贴到服务网站上。记住对应的用户名。

**步骤3** 编辑**~/.ssh/config**文件。

```
Host dc_1
HostName *******.com
IdentityFile ~/.ssh/dc_1
PreferredAuthentications publickey
User username1
Host dc_2
HostName *******.com
IdentityFile ~/.ssh/dc_2
PreferredAuthentications publickey
User username2
```

要点在于Host与HostName的区别：

- HostName：是填写真实的服务地址。
- Host：是填写别名，后面会用上。
- IdentityFile：填写的是证书的所在位置，您也可以把证书保存在任何地方。

**步骤4** 读写代码。

在Web页面上复制的SSH URL，可以直接使用，例如：

```
git@*****.com:name/repo.git
```

现在根据您的账号不同，需要自行替换：

```
git@dc_1:name/repo.git 或 git@dc_2:name/repo.git
```

**----结束**

# 2 成员权限问题

## 2.1 如何将成员一键添加到所有代码仓

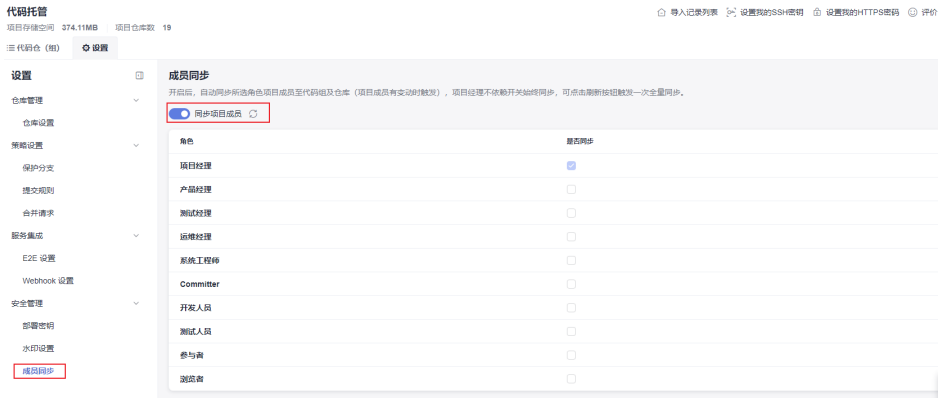
### 场景描述

代码仓库较多，项目组新加入的成员如果在每个仓库都配置一遍会非常繁琐，管理难度非常大。

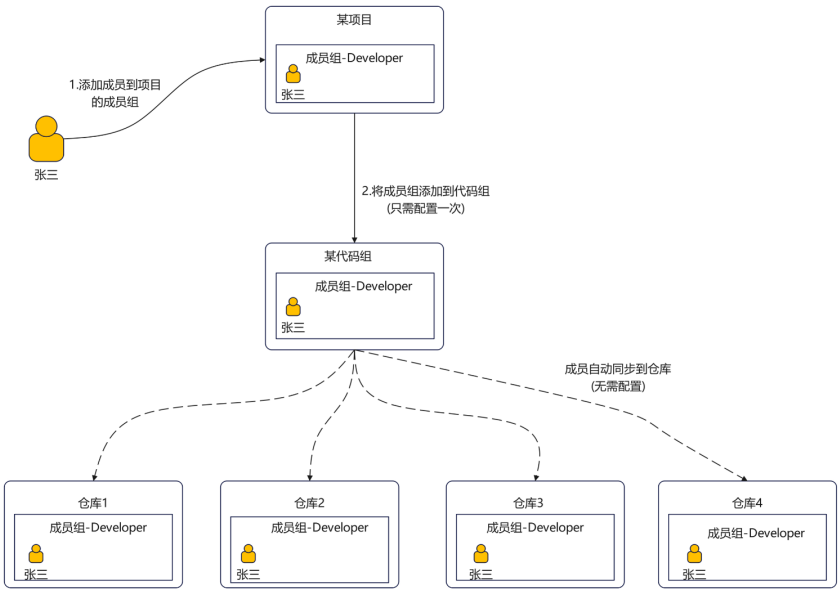
### 解决方案

- 方案一：通过项目代码托管设置-成员同步

开启后，自动同步所选角色项目成员至代码组及仓库（项目成员有变动时触发），项目经理不依赖开关始终同步，可单击刷新按钮触发一次全量同步。




- 方案二：通过成员组进行成员管理。




操作步骤如下：

**步骤1** 在项目里定义成员组。

**步骤2** 通过代码组对仓库进行分层管理，代码组下建仓库。

**步骤3** 将项目的成员组添加到代码组后，项目里的成员组有成员新加入/离开，就会自动同步到仓库。实现成员一键添加到所有代码仓里的述求。

**----结束**

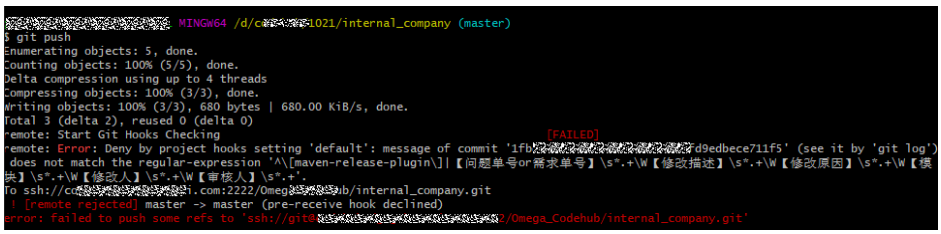
# 3 代码上传下载问题

## 3.1 从本地推送代码仓到 CodeArts Repo 时，报错"Error: Deny by project hooks setting 'default': message of commit"

### 问题现象

如果push代码文件到远端仓时，推送的格式不规范，没有填写单号、修改人等，会出现如下图所示的报错信息。

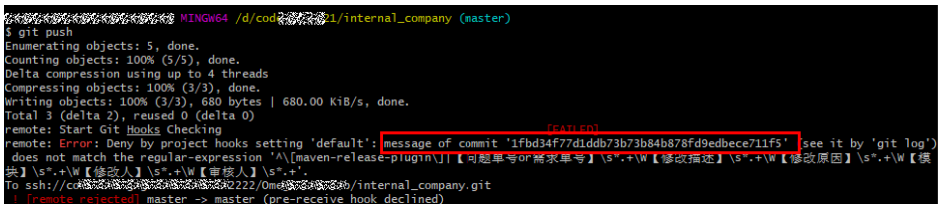
**图 3-1** push 代码时的报错信息




### 原因分析

提交信息不符合规范：^\[maven-release-plugin\]|【问题单号or需求单号】\s*.+\W【修改描述】\s*.+\W【修改原因】\s*.+\W【模块】\s*.+\W【修改人】\s*.+\W【审核人】\s*.+

**图 3-2** push 代码时的提交信息



## 解决方案

- 如果您修改最新一次书写错误的提交记录信息，可执行如下操作：

**步骤1** 执行如下操作，修改最新一次书写错误的提交记录信息。

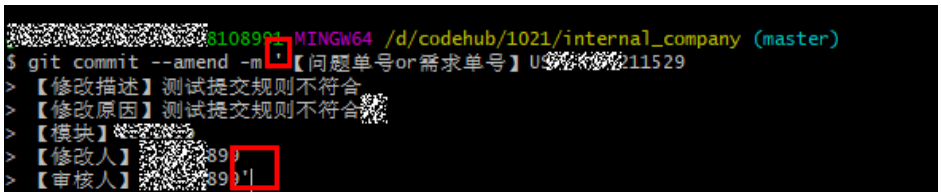
在.git隐藏文件夹的层级，右键选择“Open Git Bash here”打开Git Bash。

**步骤2** 执行**git log**命令，查找到错误的提交记录。

**步骤3** 执行**git commit --amend -m** 命令，进入vi文本编辑器界面。

**步骤4** 执行**git commit --amend -m** '命令，输入第一行信息，回车再输入第二行信息，依次类推，在最后一行末尾加上另一个单引号，回车即可修改成功。

**图 3-3** 修改提交信息




**----结束**

- 如果您修改非最新一次书写错误的提交记录信息，可执行如下操作：

**步骤1** 如果修改非最新一次书写错误的提交记录信息，您可执行如下操作修改倒数第二次提交信息：

在Git Bash界面中执行**git log**命令，查找错误的提交记录。


**步骤2** 执行**git reset --hard commitID**，回退到该Commit ID之前的提交记录。

**步骤3** 执行**步骤3**和**步骤4**，完成提交信息的修改。

**----结束**

# 3.2 用户推送二进制文件到 CodeArts Repo 失败

代码托管支持提交二进制文件管控功能，请查看仓库是否禁止提交二进制文件。

系统不推荐您将二进制文件存放至代码托管仓库，这样会影响代码仓的性能和稳定性。建议您将二进制文件上传到**“制品仓库”**中进行存储。


# 3.3 执行 git push 命令时，报错'origin' does not appear to be a git repository...

## 问题现象

执行如下命令时，出现报错“'origin' does not appear to be a git repository...”。

```
git push --set-upstream origin feature1
```

## 原因分析

原因是远程不存在origin这个仓库名称。

## 处理方法

查看远程仓库名称及路径的相关信息，删除错误的远程仓库名称，再重新添加新的远程仓库。执行如下命令：

**步骤1** 查看远程仓库的详细信息，可看到代码仓库的名称，关联地址。

```
git remote -v
```

**步骤2** 删除错误的origin仓库。

```
git remote remove origin
```

**步骤3** 重新添加远程代码仓库地址。

```
git remote add origin
```

**步骤4** 重新提交代码文件到远程代码仓库的master主干。

```
git push -u origin master
```

**----结束**

# 3.4 在 CentOS 系统下使用 HTTPS 协议克隆代码时，报错 "The requested URL returned error: 401"

## 问题现象

在CentOS系统下使用HTTPS方式克隆代码时，系统报错“The requested URL returned error: 401”。

## 原因分析

CentOS系统自带的Git版本为1.7.1及以下。

## 处理方法

**步骤1** 在Git Bash中执行如下命令，查看系统自带的Git版本。版本为1.7.1及以下，请继续执行步骤2。

```
git --version
```

**步骤2** 执行如下命令，卸载CentOS自带的Git版本。

```
yum remove git
```

**步骤3** 在**Git官网**下载最新版本，并执行如下命令将Git添加到环境变量中。

1. 执行如下命令，从GitHub上下载Git的源代码压缩包，版本号是需要替换为具体的版本号。
   ```
   wget https://github.com/git/git/archive/版本号.tar.gz
   ```
2. 执行如下命令，解压缩源代码压缩包。
   ```
   tar zxvf 版本号.tar.gz
   ```
3. 执行如下命令，进入解压后的Git源代码目录。
   ```
   cd git-版本号
   ```
4. 执行如下命令，生成配置文件。
   ```
   make configure
   ```
5. 执行如下命令，配置Git的安装路径和编码转换库。
   ```
   ./configure --prefix=/usr/local/git --with-iconv=/usr/local/libiconv
   ```
6. 执行如下命令，编译Git源代码和文档。
   ```
   make all doc
   ```
7. 执行如下命令，安装Git及其文档。
   ```
   make install install-doc install-html
   ```
8. 执行如下命令，将Git的可执行文件路径添加到系统环境变量中。
   ```
   echo "export PATH=$PATH:/usr/local/git/bin" >> /etc/bashrc
   ```
9. 执行如下命令，使环境变量生效。
   ```
   source /etc/bashrc
   ```

**----结束**

# 3.5 使用 git pull 拉取 CodeArts Repo 的代码失败，报错 "Merge branch 'master' of https://test.com Please Enter a commit"

## 问题现象

使用git pull命令拉取CodeArts Repo的代码失败，报错“Merge branch 'master' of https://test.com Please enter a commit message to explain why this merge is necessary...”，报错如下图所示。

**图 3-4** 报错提示信息


## 原因分析

原因是CodeArts Repo的代码仓库与您本地仓库内容不一致，拉取代码时会跟本地代码进行合并（merge），弹框提示是否确认本次merge操作，并提交备注信息。

## 处理方法

进入本地仓库的目录中，在Git Bash中执行如下命令：

**步骤1** 执行如下命令，取消合并操作。

```
git merge --abort
```

**步骤2** 执行如下命令，将远程仓库(origin)的 master 分支合并到当前分支。

```
git merge origin/master
```

**----结束**

# 3.6 fatal: refusing to merge unrelated histories

## 问题现象

使用**git pull**命令时，提示“refusing to merge unrelated histories”。

### 原因分析

云端与本地的仓库不同，例如：不同的分支、或不同的仓库等。

### 处理方法

使用命令**git pull origin master --allow-unrelated-histories**进行强制合并。

## 3.7 如何阻止涉密文件被推送到 CodeArts Repo 代码仓库

### 背景信息

- 当用户需要避免上传涉密信息的时候，可以选择在仓库设置中设置提交规则，勾选拒绝包含秘密的提交选项。
- 通过选择复选框防止向代码仓库提交涉密的文件，当文件名与正则表达式匹配时，代码托管服务会阻止用户推送。

**须知**

此推送规则不会限制已提交的文件。

以下列出代码托管服务目前会阻止的文件名称与正则表达式：

| 正则表达式 | 文件名例子 |
| --- | --- |
| (ssh\|config)\/(personal\|server)_(rsa\|dsa\|ed\d+\|ecdsa) | ssh_server_rsa |
| _rsa$ | id_rsa |
| _dsa$ | id_dsa |
| _ed25519$ | id_ed25519 |
| _ecdsa$ | id_ecdsa |
| \.(pem\|key)$ | secret.key privatekey.pem |
| "[._]history$" | .bash_history |

# 4 仓库迁移问题

## 4.1 基于 Git 的远程仓库导入 CodeArts Repo 时，报错“访问远程仓库超时，请检查网络”

### 问题现象

基于Git的远程仓库导入CodeArts Repo时，导入时长超过30分钟，并且提示“访问远程仓库超时，请检查网络”。

### 问题分析

导致该问题的原因可能是：代码仓库过大或者网络不好。

### 解决方案

**步骤1** 从源仓库地址下载仓库。进入要下载的代码仓，复制其HTTPS地址。

**步骤2** 打开Git Bash客户端，执行如下命令，将代码仓库克隆到本地计算机。

```
git clone --bare 源仓库地址
```

**步骤3** 将克隆的代码仓库关联并推送到CodeArts Repo。

1. 进入CodeArts Repo首页，单击“新建仓库”，在“归属项目”下拉框中选择已有的项目或者“新建项目”。
2. 仓库类型选择“普通仓库”，填写对应参数信息并去勾选“允许生成README文件”，设置“选择gitignore”，完成新的代码仓库创建，并自动跳转到该代码仓库首页。
3. 选择右上角的“克隆/下载” > “用HTTPS克隆”，复制HTTPS地址。
4. 打开Git Bash客户端，执行如下命令，将本地的代码仓库推送到上述新建的代码仓库中。

   ```
   git push --mirror 新建的代码仓库的HTTPS地址
   ```

   在执行命令时，需要您输入CoeArts Repo的HTTPS账号和密码。

**----结束**

**说明**

- 如果您本地的代码仓库有分支和标签，将会同步推送到新建的代码仓库中。
- 推送成功后，请到CodeArts Repo的代码仓库详情页，查看您推送的代码仓库是否完整，仍有问题请联系华为云技术支持。

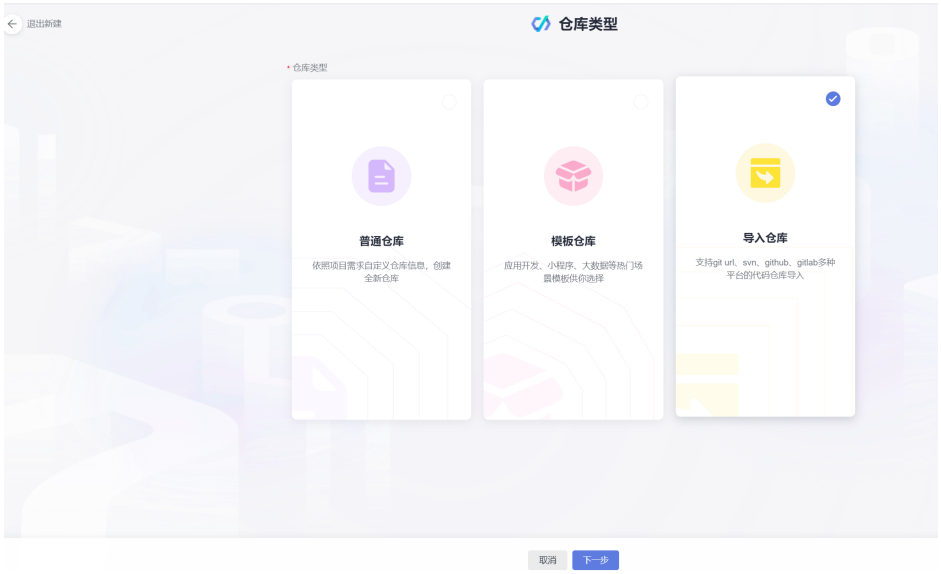
# 4.2 如何迁移 Gitee 仓库

## 迁移流程

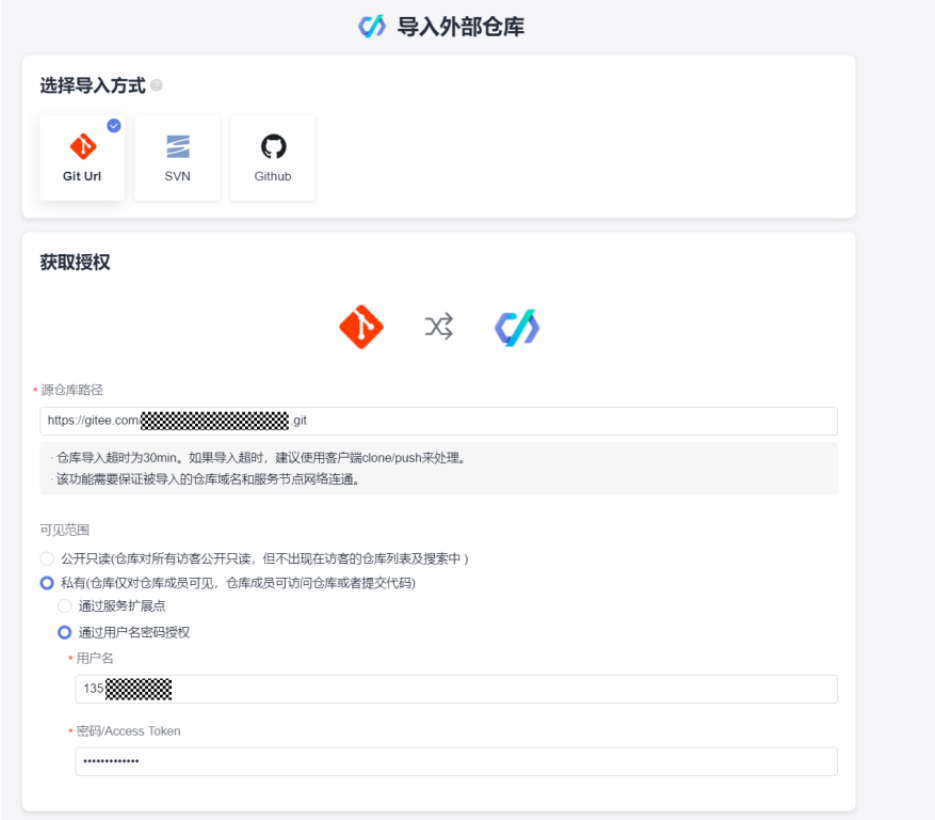

该迁移工具支持迁移仓库文件等相关数据，例如：Branch、Tag、Commit完整提交记录和代码库源文件。不支持迁移Gitee仓库的成员、PR、评论等数据。

## 操作步骤

**步骤1** 新建仓库，选择导入仓库方式。登录并进入到CodeArts Repo，选择“新建仓库 > 导入仓库”。




**步骤2** 导入方式选择“Git Url”， 并填写Gitee的仓库https url地址。




**步骤3** 配置Gitee的用户名和密码。“用户名”：为Gitee登录用户名，一般为手机号，“密码”：为登录Gitee时使用的密码。

**步骤4** 填写仓库信息及初始化配置。


**步骤5** 填写导入后仓库的关键信息。

- 代码组路径： 可选择导入的仓库的根目录路径。
- 仓库名称： 导入后仓库的名称。
- 可见范围: 私有仓库/公开仓库。
- 定时同步：勾选了定时同步，则导入的仓库为镜像仓，仓库无法提交代码，只能从源仓定时同步，仓库将每24小时刷新一次，刷新内容为源仓库24小时前的内容。
- 分支设置: 导入源仓库的默认分支或全部分支。
- 自动创建代码检查任务。

**步骤6** 仓库导入成功。您可在代码托管首页，右上角单击“导入记录列表”，表查看导入状态及失败原因等关键信息。如下图所示，仓库导入成功后，查询仓库已创建成功。


**----结束**

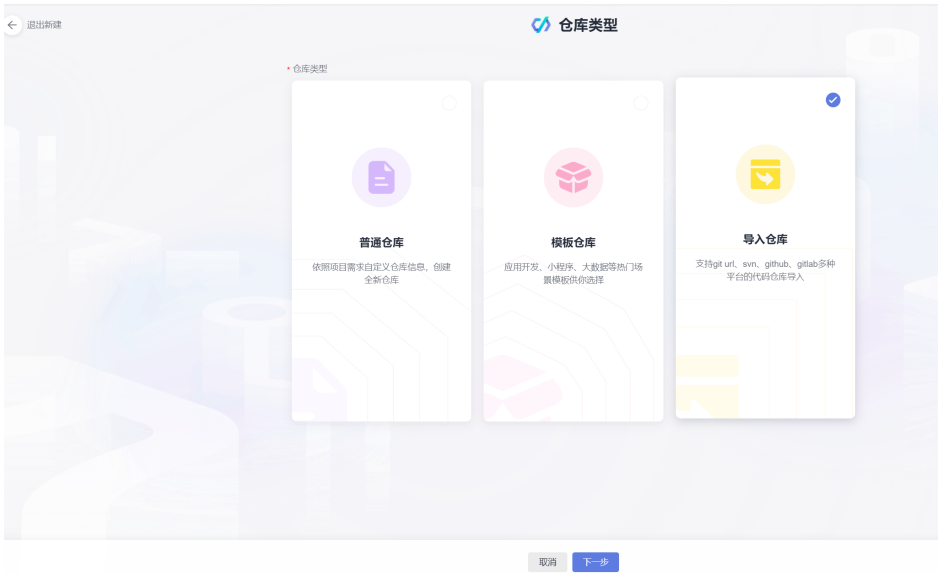
# 4.3 如何迁移 GitLab 仓库

## 迁移流程


该迁移工具支持迁移仓库文件等相关数据，例如：Branch、Tag、Commit完整提交记录和代码库源文件。不支持迁移GitLab仓库的成员、PR、评论等数据。

## 操作步骤

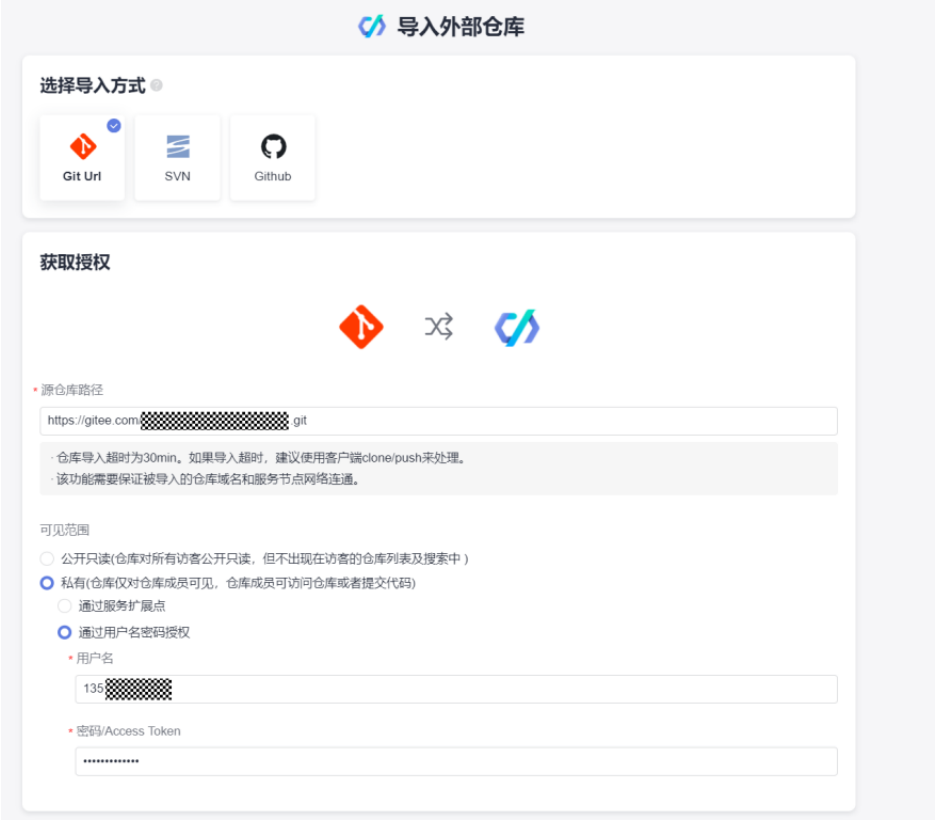
**步骤1** 登录并进入到CodeArts Repo首页，选择“新建仓库 > 导入仓库”。




**步骤2** 导入方式选择“Git Url”， 并填写GitLab的仓库https url地址。

**步骤3** 配置GitLab的用户名和个人访问令牌。

- 用户名: 实际未使用，可任意填一字符串，如test。
- AccessToken：GitLab右上角单击头像，选择“Preferences > User Settings > Access Tokens”，勾选“read_repository”权限。




**步骤4** 填写仓库信息及初始化配置。

**步骤5** 填写导入后仓库的关键信息。

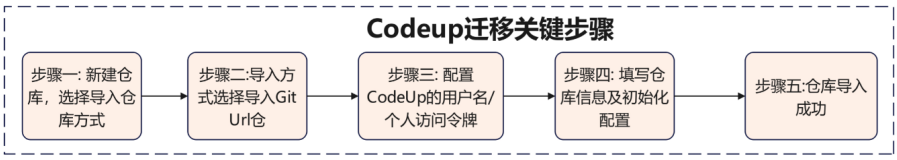
- 代码组路径： 可选择导入的仓库的根目录路径。
- 仓库名称： 导入后仓库的名称。
- 可见范围: 私有仓库/公开仓库。
- 定时同步：勾选了定时同步，则导入的仓库为镜像仓，仓库无法提交代码，只能从源仓定时同步，仓库将每24小时刷新一次，刷新内容为源仓库24小时前的内容。
- 分支设置: 导入源仓库的默认分支或全部分支。
- 自动创建代码检查任务。

**步骤6** 仓库导入成功。您可在导入记录列表查看导入状态及失败原因等关键信息。如下图所示，仓库导入成功后，查询仓库已创建成功。


**----结束**

# 4.4 如何迁移 Codeup 仓库

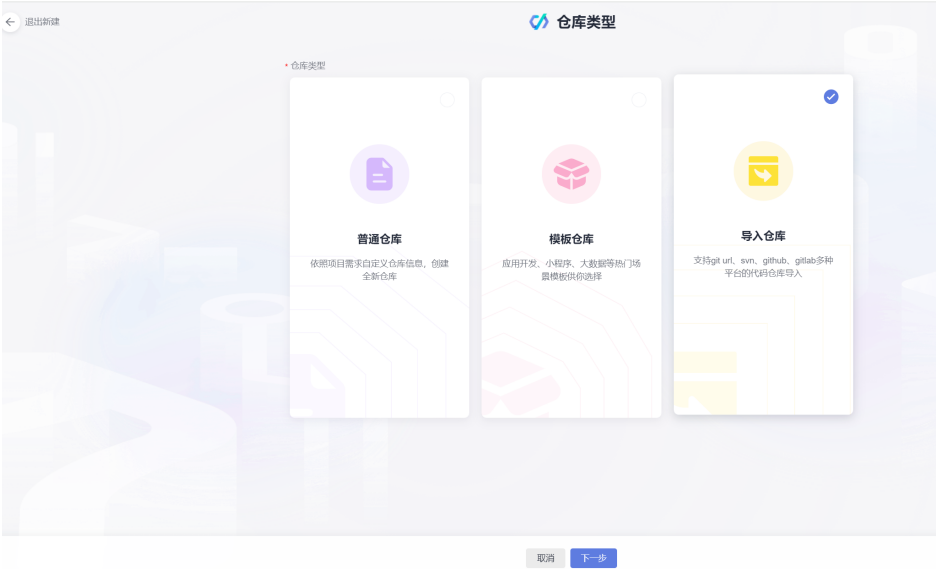
## 迁移流程



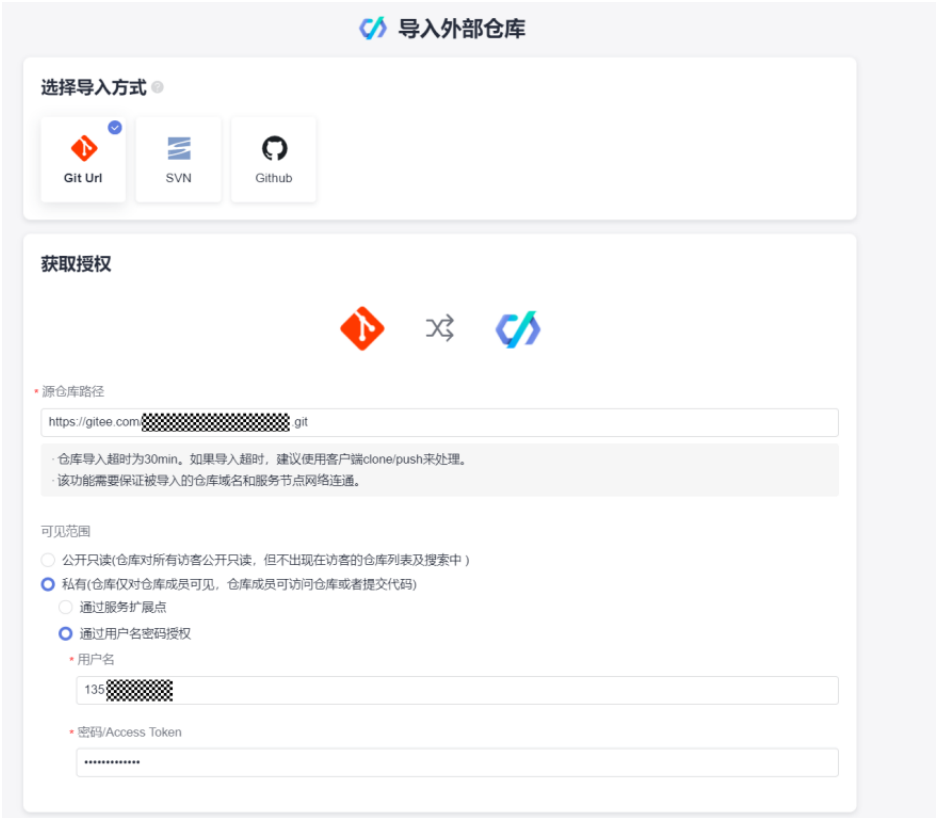

该迁移工具支持迁移仓库文件等相关数据，例如：Branch、Tag、Commit完整提交记录和代码库源文件。不支持迁移Codeup仓库的成员、合并请求、评论等数据。

## 操作步骤

**步骤1** 新建仓库，选择导入仓库方式。登录并进入到CodeArts Repo, 选择“新建仓库 > 导入仓库”。




**步骤2** 导入方式选择导入“Git Url”。并填写Codeup的仓库https url地址。



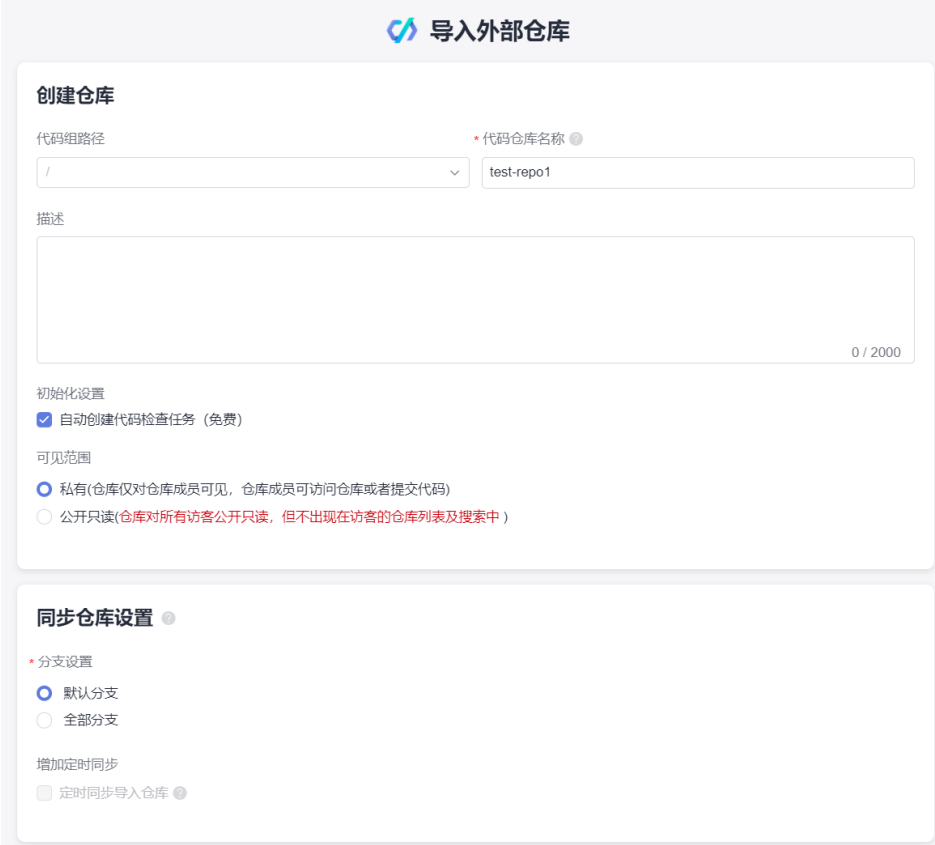

**步骤3** 配置Codeup的“用户名”和“个人访问令牌”。用户名: Codeup登录用户名，一般为手机号。AccessToken: “Codeup个人设置 > 个人访问令牌 > 新建访问令牌”。勾选代码管理的代码仓库只读权限。

**步骤4** 填写仓库信息及初始化配置。

- 代码组路径： 可选择导入的仓库的根目录路径。

- 仓库名称： 导入后仓库的名称。
- 可见范围: 私有仓库/公开仓库。
- 定时同步：勾选了定时同步，则导入的仓库为镜像仓，仓库无法提交代码，只能从源仓定时同步，仓库将每24小时刷新一次，刷新内容为源仓库24小时前的内容。
- 分支设置: 导入源仓库的默认分支或全部分支。
- 自动创建代码检查任务。



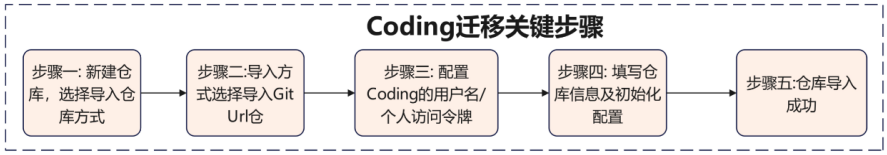

**步骤5** 仓库导入成功。您可在导入记录列表查看导入状态及失败原因等关键信息。仓库导入成功后，查询仓库已创建成功。


**----结束**

# 4.5 如何迁移 Coding 仓库

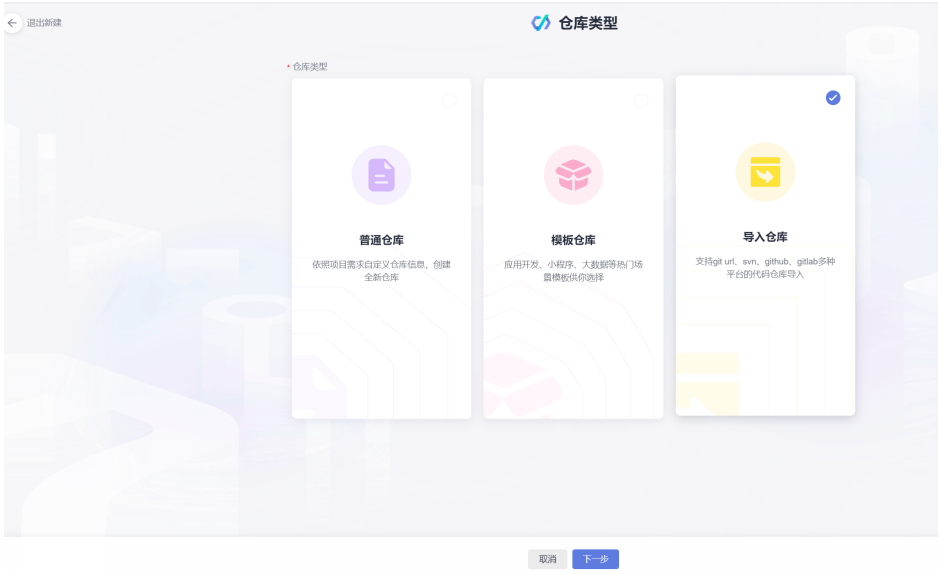
## 迁移说明



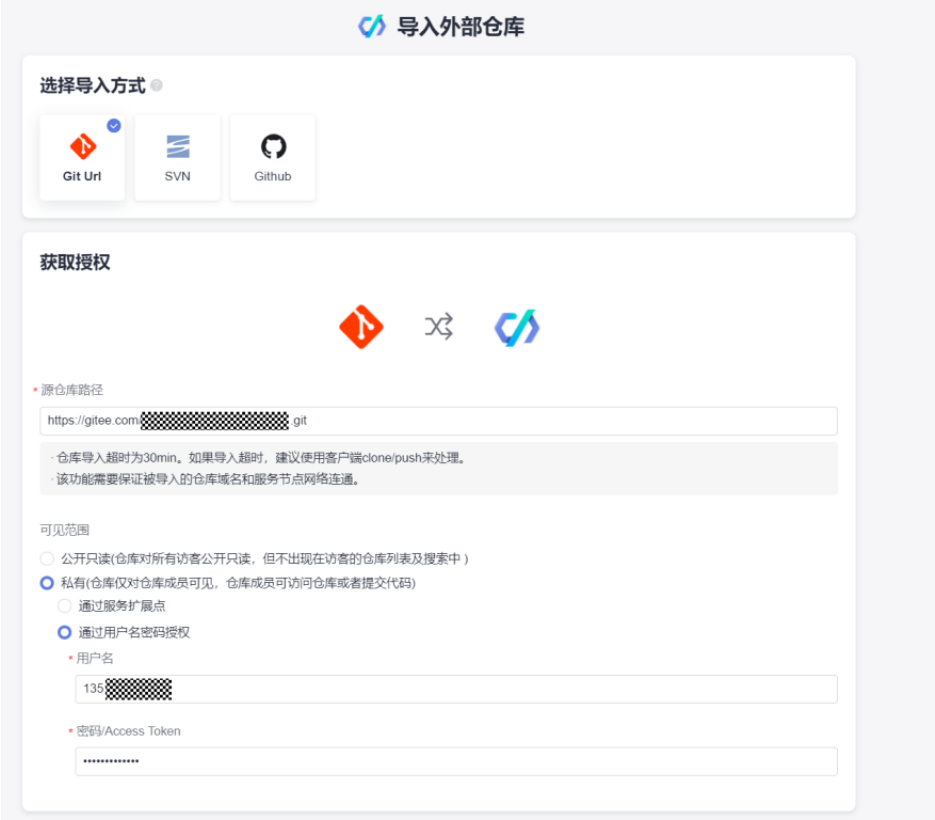

该迁移工具支持仓库文件等相关数据，例如Branch、Tag、Commit完整提交记录和代码库源文件。不支持迁移Coding仓库的成员、合并请求、评论等数据。

## 操作步骤

**步骤1** 新建仓库，选择导入仓库方式。登录并进入到CodeArts Repo, 选择“新建仓库->导入仓库”。




**步骤2** 导入方式选择导入“Git Url”仓，并填写Coding的仓库https url地址。

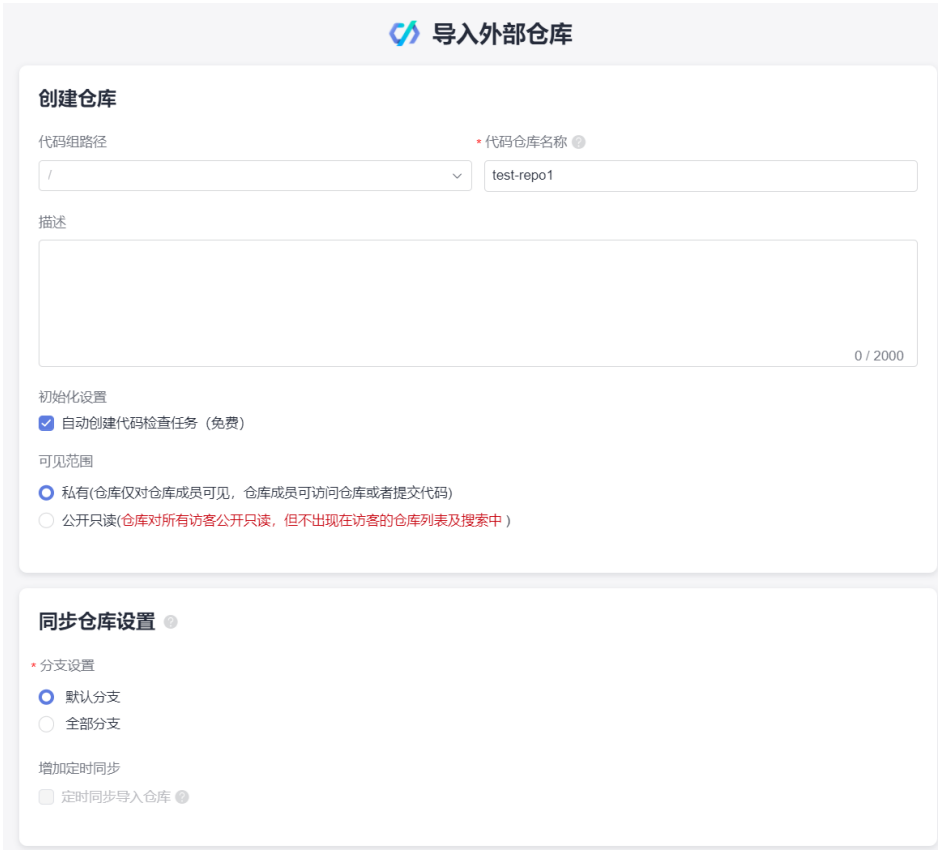



**步骤3** 配置Coding的“用户名”和“个人访问令牌”。用户名: 为Coding登录用户名，一般为手机号。AccessToken: “Coding个人账户设置->访问令牌->新建访问令牌”，勾选代码仓库只读权限。

**步骤4** 填写仓库信息及初始化配置。

- 代码组路径： 可选择导入的仓库的根目录路径。
- 仓库名称： 导入后仓库的名称。
- 可见范围: 私有仓库/公开仓库。
- 定时同步：勾选了定时同步，则导入的仓库为镜像仓，仓库无法提交代码，只能从源仓定时同步，仓库将每24小时刷新一次，刷新内容为源仓库24小时前的内容。
- 分支设置: 导入源仓库的默认分支或全部分支
- 自动创建代码检查任务。




**步骤5** 仓库导入成功。您可在导入记录列表查看导入状态及失败原因等关键信息。仓库导入成功后，查询仓库已创建成功。


**----结束**

# 5 合并请求问题

## 5.1 合入合并请求时，提示“无权限”

### 问题现象

在合入合并请求详情页，单击“合入”，提示“无权限”。

### 原因分析

您需要同时拥有目标分支的代码“提交”和MR“合并”权限。

### 处理方法

- 如果目标分支是普通分支，确认您是否同时拥有这两个权限，如果没有，请联系仓库管理员添加权限。
- 如果目标分支是保护分支，请进入代码仓库“设置”页，选择“策略设置 > 保护分支”，确认您是否同时拥有该“保护分支”的“推送”和“合并”权限，如果没有，请联系仓库管理员修改“保护分支”配置。

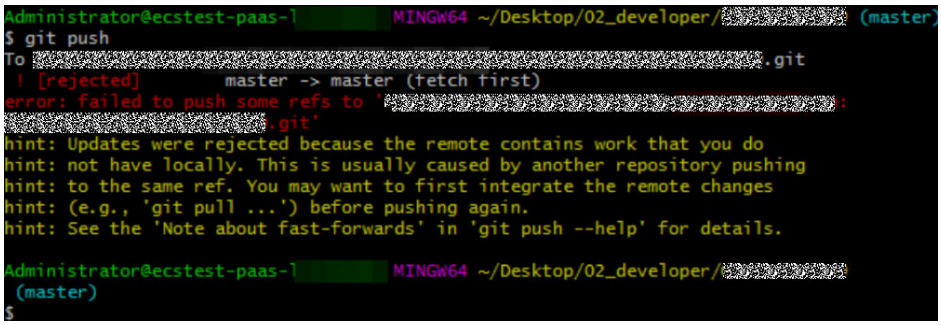
## 5.2 在本地提交合并请求时，报错"failed to push some refs to '....git'"

### 问题现象

在多人团队使用CodeArts Repo时，可能出现两个人同时修改同一行代码，这时在推送（push）代码到CodeArts Repo时会出现代码提交冲突并推送失败，并报错"failed to push some refs to '....git'"，如下图所示。

**图 5-1** push 代码时报错信息图




## 原因分析

同一个文件同一个位置被同时修改，即本地仓与CodeArts Repo代码仓库存在差异，会产生合并冲突。

## 解决方案

当代码提交冲突产生时，您可以将远程代码仓库拉取（pull）到本地仓库的工作区，这时Git会将可以合并的修改内容进行合并，并将不能合并的文件内容进行提示，开发者只需要对提示的冲突内容进行修改即可再次推送到远程仓库（add → commit → push），这样操作便可以解决合并请求冲突。

在修改冲突文件时应该考虑清楚，必要时要与冲突方联系协商解决，避免覆盖他人代码。

**说明**

git pull可以理解为 git fetch 的操作 + git merge的操作，其详细说明如下：

```
git fetch origin master #从远程主机的master分支拉取最新内容
git merge FETCH_HEAD    #将拉取下来的最新内容合并到当前所在的分支中
```

在merge的时候，会将有冲突不能合并的内容做出提示。

## 如何在代码托管服务的控制台上解决分支合并冲突?

- **在线解决冲突**（推荐在代码量较小或涉及冲突的代码量较小的情况下使用）
  a. 单击“在线解决冲突”，弹出如下图所示的代码冲突。
     此页面中您可以直接选择“应用源分支”或“应用目标分支”来选择一方的修改作为最终修复后的内容。


  b. 当情况较复杂，简单的直接覆盖无法解决问题时，可单击 进入“手动编辑”模式，如下图所示。


c. 在上述页面中手动修改代码以解决冲突，并进行提交即可。

**说明**

提交时注意需要填写提交信息。

上图中“<<<<”、“>>>>”、“====”等所在行是冲突展现与分割符，在修改代码解决冲突时，要注意将其删除。

# 5.3 在本地解决代码文件冲突

## 问题现象

在合入合并请求详情页，显示“代码合并冲突未解决”。

## 原因分析

同一个文件，被两个人修改，导致在合入合并请求时报冲突。

## 解决方案

**步骤1** 更新代码，并切换到此合并请求源分支。

```
git fetch origin
git checkout -b feature_010 origin/feature_010
```

**步骤2** 将目标分支合入源分支。

```
git merge origin/master
```

**步骤3** 根据提示，在本地手动解决冲突。

**步骤4** 解决完冲突后提交到远端仓库。

```
git add .
git commit -m '提交信息'
git push origin feature_010
```

**步骤5** 刷新界面，继续检视该合并请求。

**----结束**

# 6 Fork 同步问题

## 6.1 如何从主库同步代码到个人 Fork 出来的派生库

### 问题现象

在CodeArts Repo上可以将主库代码仓Fork到个人其它项目下，此功能有助于协同开发，但在使用Fork模式开发时，可能会遇到问题：当主库（源项目）更新后，Fork库并不会一起更新，需要自己手动更新。

您可以通过如下操作将主库代码同步到个人Fork出来的派生库。

例如：

主仓地址: https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/AlphaGo/TestService.git

Fork仓地址: https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/Roger/TestService.git

分支名：master

### 解决方案

**步骤1** Clone个人空间里的Fork仓库到本地。

```
git clone https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/Roger/TestService.git
cd TestService
```

**步骤2** 增加远程原始仓（主库仓）到本地（可以用 git remote -v 命令查看远程仓列表）

```
git remote -v
```

**步骤3** 如果没有远程原始仓，则需要增加：

```
git remote -v
origin  https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/Roger/TestService.git (fetch)
origin  https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/Roger/TestService.git (push)
```

**步骤4** 查看确认远程仓列表。

```
git remote -v
origin  https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/Roger/TestService.git (fetch)
origin  https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/Roger/TestService.git (push)
main https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/AlphaGo/TestService.git (fetch)
main https://test.com/f2e22eeb1b8c43cfb764765f5e3ff039/AlphaGo/TestService.git (push)
```

**步骤5** 获取原始仓（主库仓）的branch分支最新代码到本地，合并两个版本的代码。

```
git pull main master
```

**步骤6** 把合并后的最新代码同步到fork仓上。

```
git push origin master
```

**----结束**

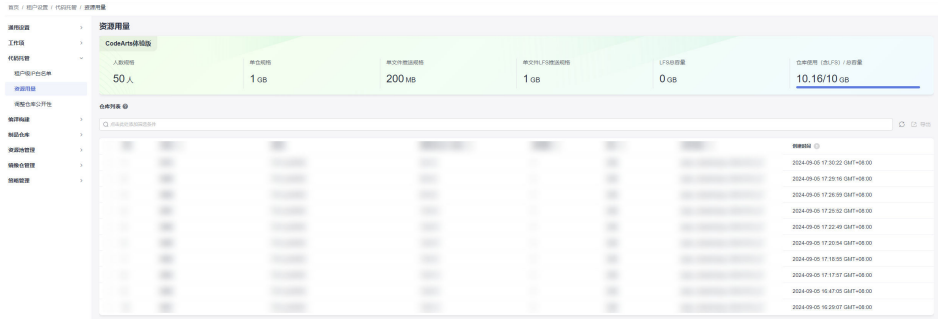
  

# 7 仓库容量问题

## 7.1 仓库剩余容量不足

### 场景描述

如果配置了容量预警通知并打开通知，当仓库容量达到一定预警值时，会通知预警，您可以单击头像，选择“租户设置 > 代码托管 > 资源用量”，查看容量使用情况。如下图所示，仓库容量当前已超出仓库最大容量10GB，目前已使用10.16GB。




### 原因分析

可能存在某些文件占用仓库空间。

### 解决方案

- 方法一，压缩仓库，降低仓库使用容量。操作步骤如下：

**步骤1** 单击头像，选择“租户设置 > 代码托管 > 资源用量”，查看租户下容量占用比较大的代码仓库。

**步骤2** 进入占用量较大的代码仓库首页，选择“设置 > 仓库管理 > 仓库加速”，对仓库容量占用比较大的仓库进行压缩, 降低仓库容量，如下图所示。


**----结束**

- 方法二，购买存储扩容包，对仓库容量进行扩容。选择，“华为云定价 > 价格计算器 > 开发与运维”选择需要购买的区域如下图所示。

# 8 常见问题汇总

## 代码托管是否支持 SVN

不支持。代码托管服务提供基于Git的分布式版本控制管理服务，能够更加便捷地进行异地协作。

## 代码托管是否支持外部 Git 源的一键导入

支持。

目前代码托管支持一键导入外部Git源，支持的外部Git源包括：

- bitbucket.org
- code.aliyun.com
- coding.net
- git.qcloud.com
- gitee.com
- github.com
- gitlab.com
- visualstudio.com
- xiaolvyun.baidu.com

## 代码托管是否支持批量下载多个仓库

不支持。代码托管暂不支持批量下载或上传多个代码仓库，需要对每个代码仓库逐一操作。管理员要对本地仓库做备份，可以自行通过Shell或者批处理命令实现多个仓库下载。

## 代码托管如何获取代码下载的存储路径

- 如果按照默认路径安装Git，通过本地PC上的开始菜单栏打开Git Bash，则通常默认的存储路径为“C:/User/XX用户”。
- 如果您当前在某个文件夹内，鼠标右键菜单打开“Git bash Here”，则存储路径就是该文件夹。

## 代码托管如何获取代码仓库的克隆地址

**步骤1** 进入代码托管首页，单击仓库列表中的仓库名进入仓库详情页。

**步骤2** 单击右侧导航栏“克隆/下载”按钮，单击“用SSH克隆”获取到SSH协议地址。单击“用HTTPS克隆”获取到HTTPS协议地址。

**----结束**

## 代码仓库中上传压缩包，是否支持在线解压

不能。暂不支持在线解压缩，建议在本地解压之后使用Git命令上传。

## 代码仓库是否支持相互转换“私有”或者“公开”

可以。进入代码仓库详情页面，在“设置”页签中选择“基本设置 > 仓库信息 > 可见范围”进行设置。

## 所有用户是否可以使用同一个 SSH 密钥上传下载代码

不可以。SSH密钥在电脑和代码托管服务之间建立安全连接，不同的用户通常使用不同的电脑，在使用SSH方式连接代码仓库前需要在自己电脑配置各自的SSH密钥。