代码托管

常见问题

文档版本 01

发布日期 2023-09-05

华为技术有限公司

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。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 仓库管理

1.1 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 用不同的帐号，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，你也可以把证书保存在任何地方。

步骤4 读写代码。

原本在Web页面上复制的SSH URL，可以直接使用，例如：

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

但是，现在根据你的帐号不同，需要自行替换：

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

----结束

1.2 如何防止软件代码被他人 Copy?

问题现象

- 软件开发的各种行为，在云上进行。
- 随时随地，都能够方便的接入云。

原因分析

- 在CodeArts上写程序的常见流程：
  a. 在CodeArts上，创建一个代码仓库，并确保是最新版本。
  b. 所有程序员都可以将代码下载到本地，在本地开发、调试完成后，提交代码到服务器。
  c. 通过Code Review过程，将程序员的代码，合入到主干。
- 令人担忧的情况：
  程序员总要在本地保留一份代码，我们无法防止，这个程序员笔记本里的代码，被他Copy出去，或者打包以后上传到某个网盘上。
  原因：那是他自己的电脑，在他自己的机器上，他的操作很难防范。

处理方法

- 封掉USB接口、封掉蓝牙接口。
- 在电脑里加装特殊的软件，监控并限制各种上传的行为。
  - 监控特定文件的上传：如何防止改名、压缩、混淆后的文件。

- 监控上传到特定网站：如何防止上传到未知的网络服务、邮件地址、自建服务器。
- 监控所有的电脑操作并记录：只能在代码泄露之后，作为起诉的证据之一，很难在事前防范。

1.3 代码托管服务升级 TLS 协议版本

问题现象

代码托管致力于提供安全、可靠、稳定、快速的代码托管服务，始终将代码资产的安全放在首位。

2018年宣布了TLSv1弱加密标准的弃用。在2019年1月11日以后，将开始禁用以下内容：

- TLSv1：适用于所有HTTPS连接，包括与代码托管的Web、API和Git连接。

原因分析

- 目前只有一小部分流量使用了弃用的算法，且很多客户端将自动转换并开始使用新算法，会有一小部分客户端受到影响。
- 预计其中大多数是不再维护的旧系统，继续使用不推荐使用的算法访问Git / CodeArts Repo API。

处理方法

- 如果您的GitBash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。
- 您也可以使用如下命令检查本地的GitBash的TLS版本：

  ```
  git config http.sslVersion
  ```

  如果是TLSv1.0，则用下面一句命令行更新至TLSv1.2

  ```
  git config --global http.sslVersion tlsv1.2
  ```

表 1-1 已知的不兼容客户端

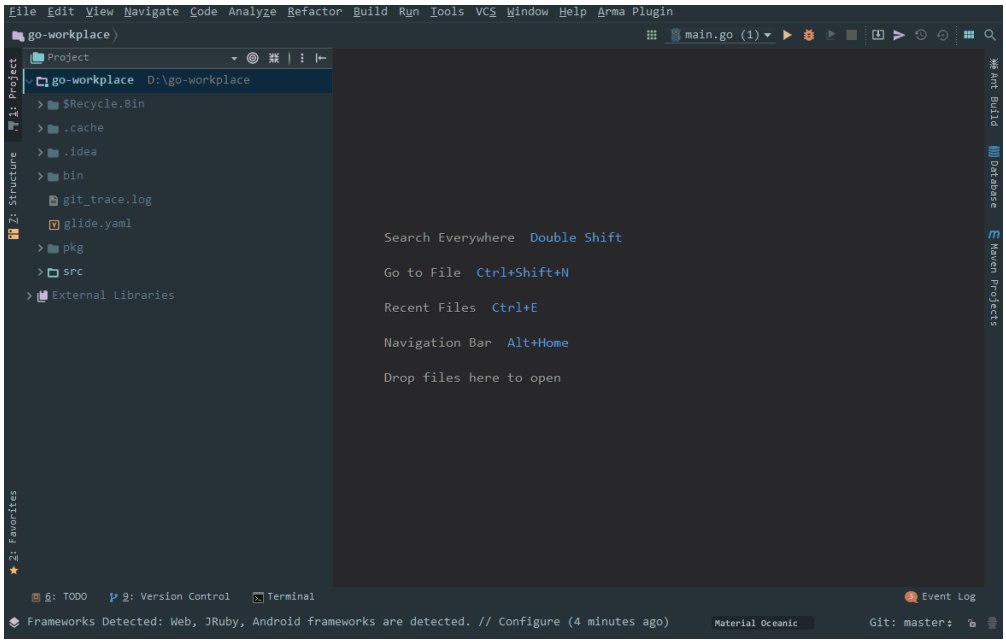
客户端	说明
Git Credential Manager for Windows< v1.14.0	默认不支持TLSv1.2可以通过更新到**最新版Git**客户端来解决，最新版版本的GitBash自带Git-Credential-Manager功能，只需要在安装时勾选即可。
Git on Red Hat 5	• 不支持TLSv1.2； • 建议Red Hat 5的用户升级到更新版本的操作系统。
Git on Red Hat 6	默认不支持，升级版本至6.8（或更高）可支持。
Git on Red Hat 7	默认不支持，升级版本至7.2（或更高）可支持。
JDK7	默认情况下为TLSv1，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。

客户端	说明
JDK6及以下	• 不支持TLSv1.2； • 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）； • Microsoft已更新Visual Studio 2017的最新版本以与TLSv1.2 Git服务器配合使用； • 我们建议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； • Microsoft已在Visual Studio开发人员社区支持论坛上提供了其他指导。

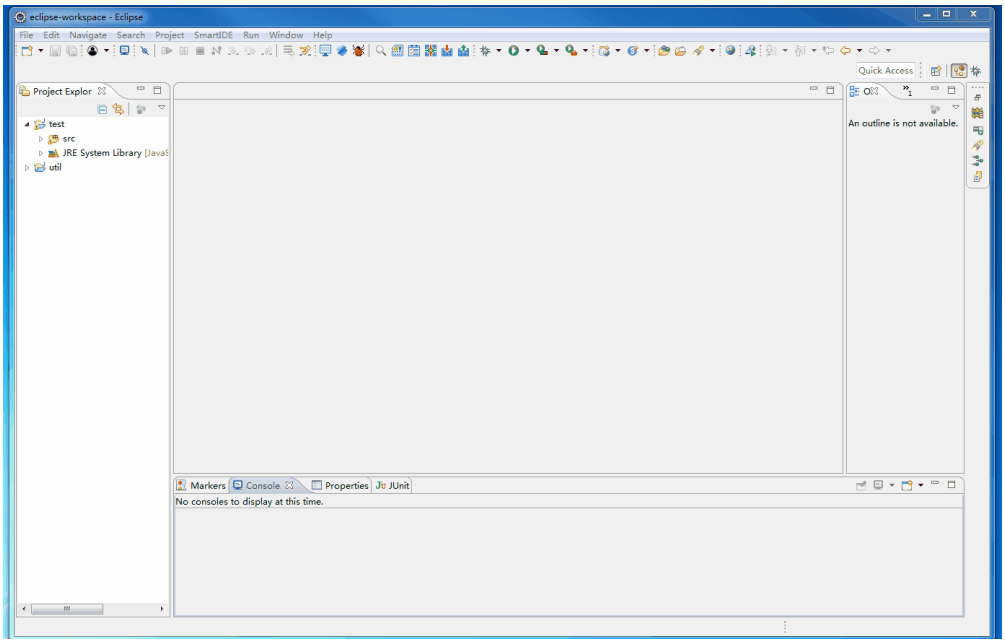
第三方 Git 图形客户端

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



1.4 同一用户创建的多个仓库都需设置 SSH 密钥吗?

不需要。

SSH密钥在电脑和某个帐号下的代码托管服务之间建立安全连接，在一台电脑上配置了SSH密钥并添加公钥到代码托管服务中后，所有该帐号下的代码仓库与这台电脑之间都可以使用该密钥进行连接。

1.5 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。

1.6 移除项目成员是否同时将其从代码仓库删除并删除其所创建的 SSH 密钥?

成员会从代码仓库中删除，但密钥不会。

当用户被管理员从项目中移除后，会自动将该用户从项目下的代码仓库中移除，但SSH密钥不会被删除。此时用户将没有访问该项目和代码的权限，不能再访问项目和代码仓库。

1.7 代码仓库可以相互转换“私有”或者“公开”吗?

可以。

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

2 仓库使用

2.1 是否要同时设置好 SSH/HTTPS 密码才能上传下载代码?

不是。

SSH和HTTPS是使用Git进行代码版本管理的两种连接方式，使用其中任何一种方式都可以进行代码的上传下载，密钥（密码）的设置根据您选择的连接方式设定即可：

- 若选择SSH方式，需要**根据页面提示生成密钥并添加到系统中**。
- 若选择HTTPS方式，可通过代码托管生成密码，**直接获取HTTPS密码**。

2.2 如何保证 SSH 密钥的安全?

1. 在生成SSH密钥时，可以为密钥设置密码以保证安全。
2. 在代码托管中，SSH密钥只能对应一个用户名，且只对该用户可见。

2.3 是否支持对每个分支设置 SSH 密钥?

不支持。

代码仓库并不支持对每个分支添加SSH密钥。

SSH密钥是一个全局的配置，是代码仓库识别用户的方式，所以不支持对于某一个项目或者某一个分支进行密钥设置。

例如，项目中有5个成员，每个人拉取一个分支进行开发，目前是不可以对每个分支设置SSH密钥来限制分支操作权限。

2.4 如何清理仓库资源?

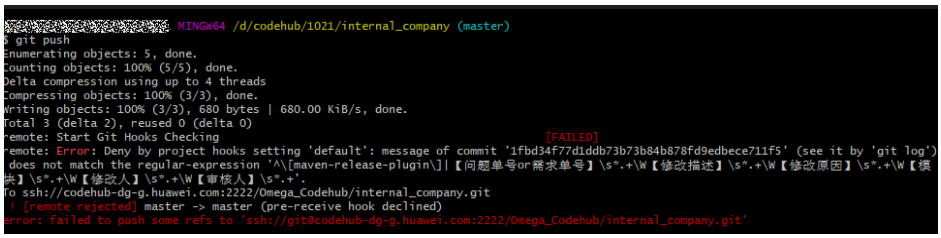
您可使用如下四种方法清理仓库资源：

- 单击仓库详情中的“代码”页签下的“分支”子页签，在控制台分支列表中，选择不需要的分支，单击 ，可以按提示操作，将该分支进行删除。
- 单击仓库详情中的“代码”页签下的“标签”子页签，在控制台的标签列表中，选择不需要的标签，单击 ，可以将此标签从代码托管仓库删除（想从本地删除请clone、pull或本地手动-d删除）。
- 单击仓库详情中的 “设置 > 仓库管理 > 仓库加速”，针对当前仓库运行后台进行整理任务，压缩文件并移除不再使用的对象。
- 单击仓库详情中的 “设置 > 仓库管理 > 子模块设置”，删除不需要的子模块。

2.5 如何修改历史提交信息?

问题现象

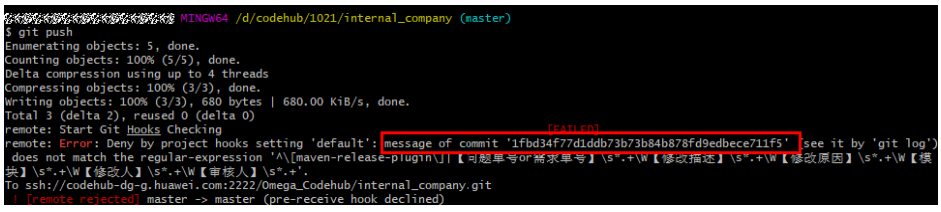
本地Commit时的提交信息有误，出现如下报错。



原因分析

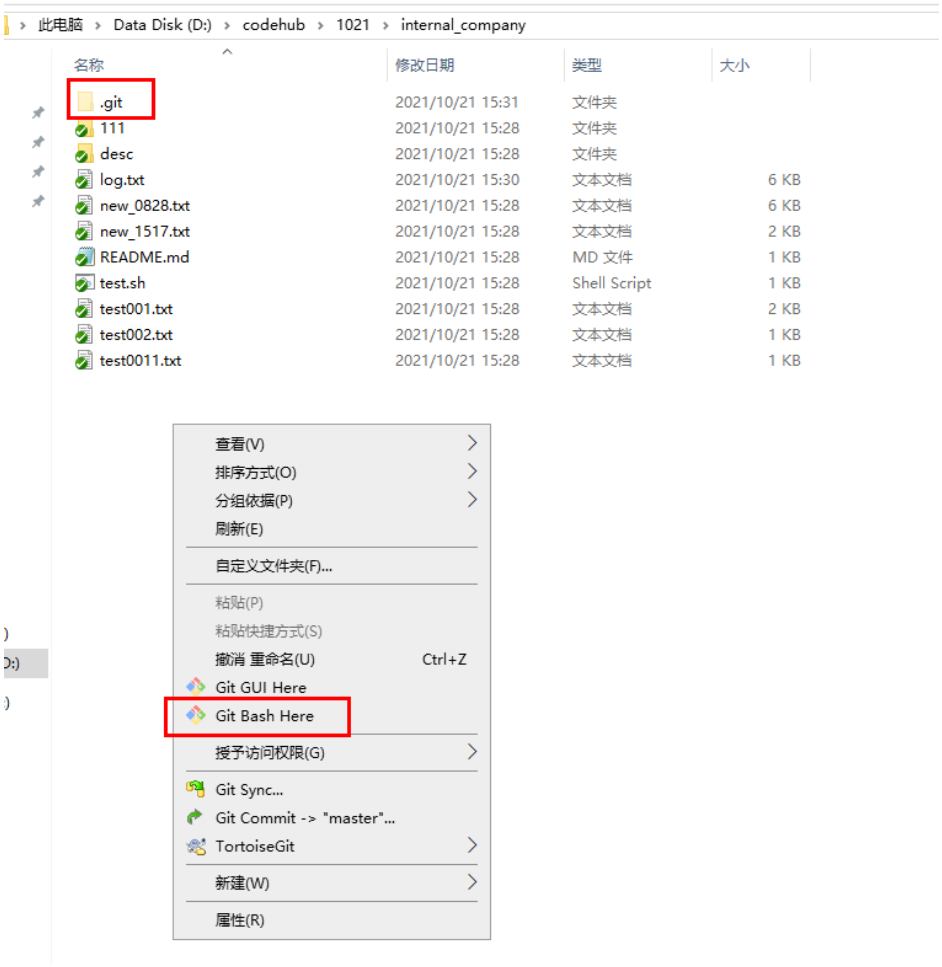
提示信息报错：message of commit，后面跟着一串由数字和小写字母组成的长达40位的字符串。这个字符串代表着这次提交的message填写有误。

即填写错误提交commit ID：1fbd34f77d1ddb73b73b84b878fd9edbece711f5。不符合提交规则：^\[maven-release-plugin\]|【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*.+



解决方案

- 您可执行如下操作修改最新一次书写错误的提交记录信息：
  a. 在本地代码工作空间的根目录，即有.git隐藏文件夹的这一层级，右键选择git bash here打开git bash界面。



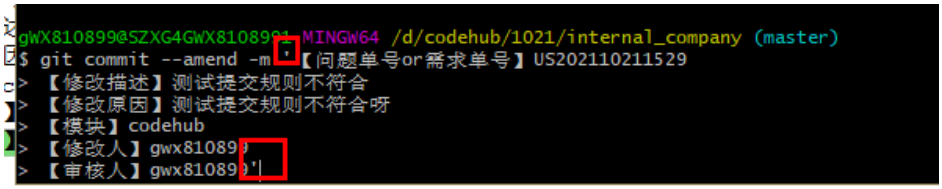
b. 执行如下命令查找最新一次提交记录信息。

```
git log
```

c. 执行如下命令，进入vi文本编辑器界面。

```
git commit --amend
```

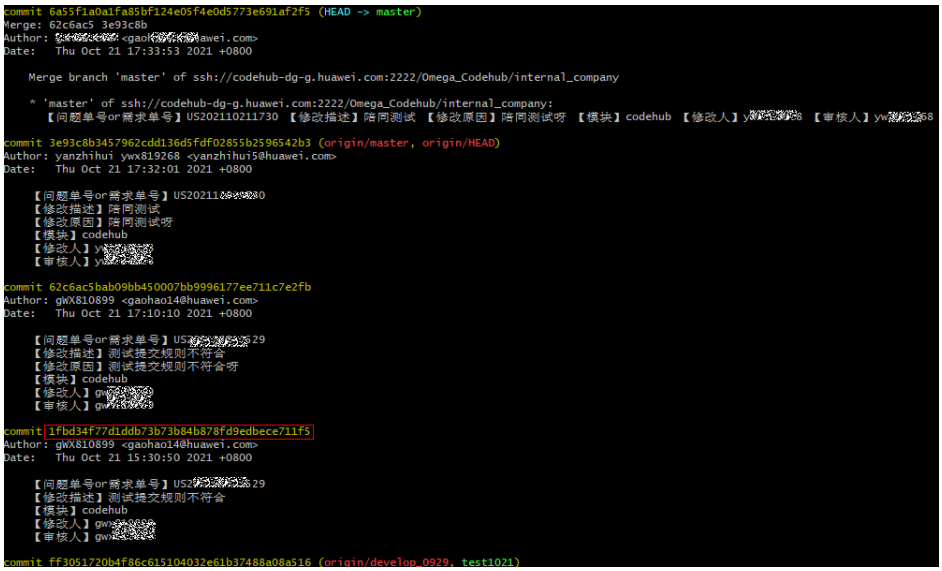
d. 输入git commit --amend -m '，输入第一行信息，回车再输入第二行信息，依次类推，在最后一行末尾加上另一个单引号，回车即可修改成功。



- 您可执行如下操作修改非最新一次书写错误的提交记录信息：

  a. 在git bush界面中执行如下命令，查找错误的提交记录。

  ```
  git log
  ```

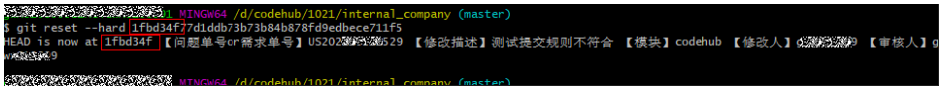
  

  b. 执行如下命令，回退到之前的提交记录。

  ```
  git reset --hard commitID
  ```

  如执行如下命令：

  ```
  git reset --hard 1fbd34f77d1ddb73b73b84b878fd9edbece711f5
  ```

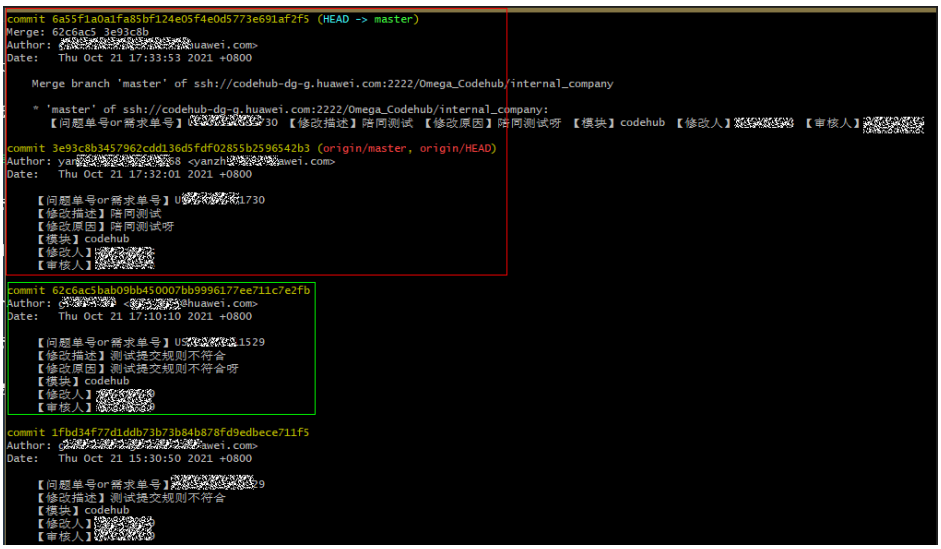
  c. 参考“修改最新一次提交记录message书写错误中2或3”，修改之前的提交记录，回车保存。

  d. 找到自己提交的记录，执行cherry-pick。

  说明

  若author是自己，且不含merge操作，则可进行cherry-pick操作。

  如下图所示，绿框中的代表是可以操作cherry-pick的，红框内的不可操作cherry-pick。

e. 执行如下命令挑分支。

```
git cherry-pick commitID
```

如commitID为62c6ac5bab09bb450007bb9996177ee711c7e2fb。

```
git cherry-pick 62c6ac5bab09bb450007bb9996177ee711c7e2fb
```

说明

1. 如果错误信息的提交先前已经push入个人库，需要进行强制推送，即git push -f。
2. 如果错误信息的提交先前未提交成功，直接git push即可。
3. 如果错误信息不是由自己引入，且错误的提交已经合入进主库，那么无需修改，直接将个人库的提交规则临时去掉，推进入库即可。

3 仓库迁移

3.1 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。

4 通用问题

4.1 项目成员为何看不到他人创建的代码仓库?

为了保证代码的安全，代码托管服务中添加了一层仓库成员的管理，需要将项目成员设置为代码仓库成员，才能够看到仓库内容。

进入代码仓库“成员”页面，单击“添加成员”可完成代码仓库成员的添加。

4.2 代码仓库对文件大小是否有限制?

有，具体如下：

- 控制台上传代码时，单个文件不能超过50M，否则会上传失败；（客户端支持上传50M以上的文件）
- 在线修改代码时，单次保存行数不能超过5000行，否则会修改失败；（但是客户端上传不限制行数）
- 每个代码仓库的容量上限是2GB。
- 本地单文件推送大小限制是200M，超出时建议使用SSH协议。

4.3 成员提交的代码行数支持导出吗?

不支持。

目前没有导出功能，可以在代码仓库的“仓库统计”页面中进行查询。

4.4 如何阻止涉密文件被推送到代码仓库?

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

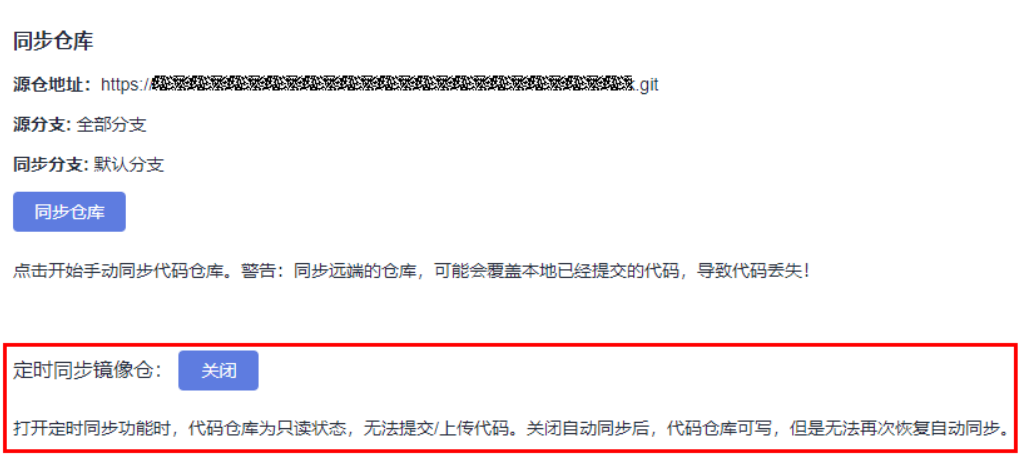
此推送规则不会限制已提交的文件。

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

4.5 为什么没有“同步仓库”功能页签?

Q：为什么我的仓库中“仓库管理”里，没有“同步仓库”这个页签?



A：只有导入的仓库，才有此功能。

4.6 删除 Git 分支的创建者，相关的 Git 代码分支也会自动删除吗?

不会，删除git分支的创建者后，这个分支就会移交给其他仓库成员，比如仓库管理者。可以在设置-操作历史里看到删除过分支的记录，这个层面是针对于整个仓库的动态，要看具体分支的历史要到具体的分支-历史去查看历史操作。

4.7 用户推送二进制文件到代码托管仓库失败?

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，配置功能请参考**提交规则**章节。

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。

5 Git 相关问题

5.1 Git 如何判断是代码仓库管理员?

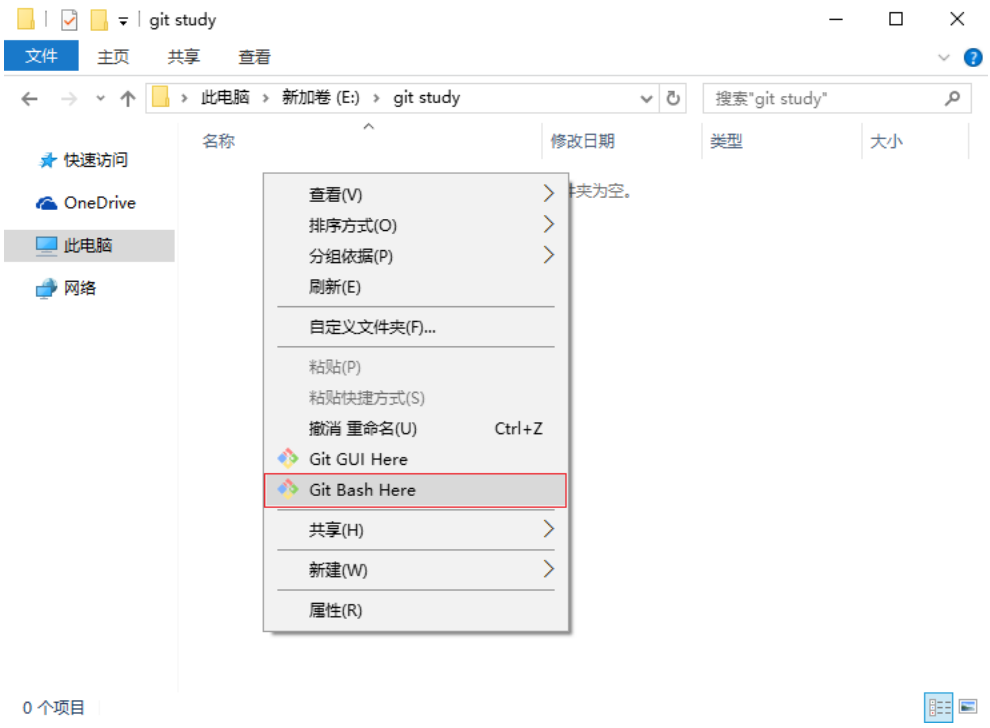
- 若使用HTTPS方式，连接云端仓库时需要输入用户名与密码。

  用户名为HTTPS用户名，格式是：帐号/子帐号，如果当前登录的是华为云主帐号用户，则格式为：帐号/帐号，通过用户名即可判断用户在仓库中的角色。
- 若使用SSH方式，连接云端仓库前需要先配置SSH密钥。

  在代码托管中，每个密钥只能对应一个用户名。在连接云端仓库时，将记录密钥与用户名的对应关系，来判断用户在仓库中的角色。

5.2 如何获取代码下载的存储路径?

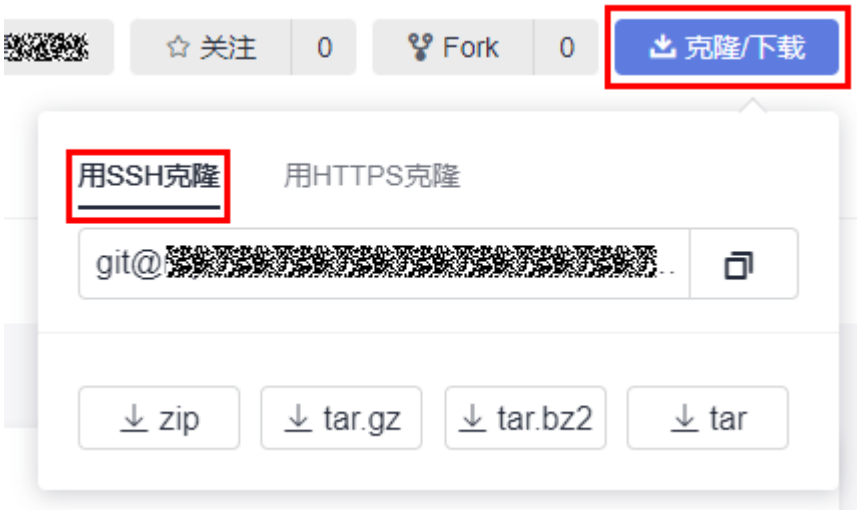
- 若按照默认路径安装Git，通过本地PC上的开始菜单栏打开**Git Bash**，则通常默认的存储路径为“C:/User/XX用户”。
- 若在某个文件夹内通过鼠标右键菜单打开**Git bash Here**，则存储路径就是该文件夹。



5.3 怎样获取代码仓库地址?

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。



步骤3 单击“**用HTTPS克隆**”获取到HTTPS协议地址。

----结束

须知

- 克隆地址仅用于克隆，不支持页面直接访问。
- 可用仓库的页面url进行页面访问。

5.4 复制的仓库地址用在哪些场景?

在连接云端仓库时，需要使用仓库地址，例如：

- 克隆云端代码仓库到本地时，使用命令： git clone 云端仓库url
- 在本地init仓库后，需要与云端仓库连接时，使用命令：git remote add origin 云端仓库url

5.5 代码托管服务是否支持 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，能够更加便捷的进行异地协作。

虽然不支持SVN，但可以将 **SVN代码库导入到代码仓库中**。

5.6 向代码仓库中上传压缩包能否在线解压?

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。

6 常见报错解决方法

6.1 私钥丢失

问题现象

用户由于密钥问题进行**git clone**失败，**git clone**时由于密钥不正确或私钥丢失导致操作失败，无法**clone**代码。密钥验证失败，系统默认转换密码验证，而密码认证方式，前期并未做相关配置，如图6-1所示。

图 6-1 私钥丢失导致认证失败

```
[root@lsw-centos65 ~]# git clone git@codehub                                                    31/master.git
Cloning into 'master'...
The authenticity of host 'codehub-                                    .com (42.202.149.38)' can't be established.
RSA key fingerprint is 82:ed:e8:ee:5c:08:d3:d2:d0:89:71:88:ab:eb:eb:78.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'codehub-                                  .com,42.202.149.38' (RSA) to the list of known hosts.
git@codehub-                                          com's password:
Permission denied, please try again.
git@codehub                                          .com's password:
```

处理方法

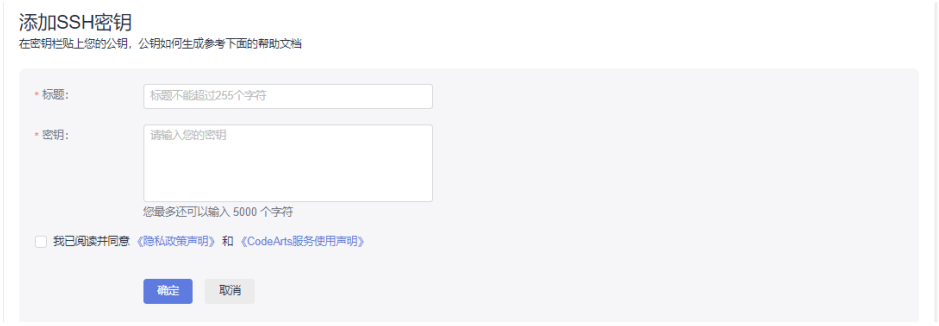
1. 登录您的代码托管服务仓库列表页，单击右上角昵称，单击“个人设置 > SSH密钥管理”，进入页面。

2. 在“SSH密钥管理”页面单击删除原有密钥。
3. 新建一个密钥。
   执行**ssh-keygen -t rsa -C "abc123@***.com"**生成密钥对。
4. 将生成的密钥进行手工复制，执行**cat ~/.ssh/id_rsa.pub**，如图6-2所示。

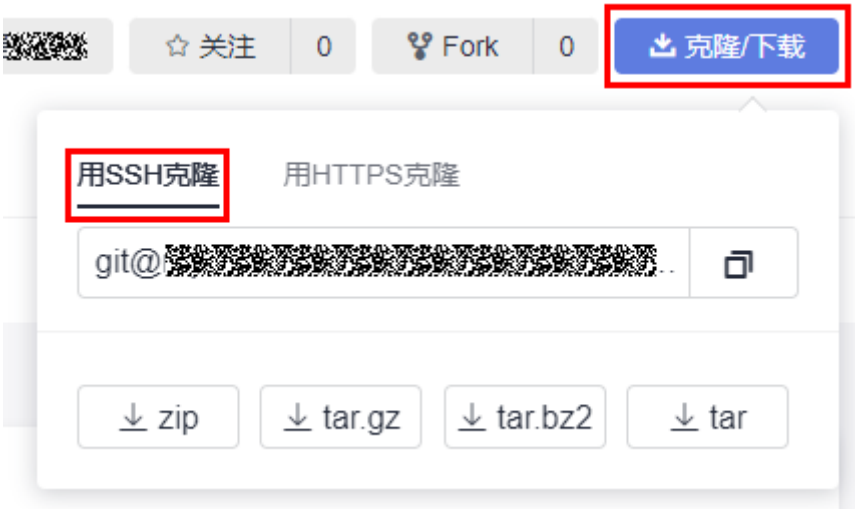
图 6-2 复制密钥

```
[root@      ~]# cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAr0LxUV0Q/XGs98lk7O2cE+xa1j4krOcH5TeGk0dWe98DqwVGjXedrqNyQUQazTB9YGKR0THYSFLBq8y2xuxIRhhn3gwKxUzd2kRFIk92R
7K04xtvnqlTWcZsLkfrALAOIct1rCLuILu38yX26OWjCiDsEWIC5TWfs/LNhNLiQOQekh9xe6537LkN+fpdj8Oa9i9gXJB+bazu8aPs6IPQKzxW02I6zQIS5jSp5oM3lCS4fkm6fwV5cA
E08Jqv7J6lb9Zw0B+PBx2vB9boE+rs/ZWbBEMezsesuHUsJ6NgIvI0M+VHcb9LSIj8qxK0hDUUyrp20B8+v56AkoBwK8D2nw==                    .com
[root@h      ~]#
```

5. 在“SSH密钥管理”页面，单击“添加SSH密钥”添加新的SSH密钥，如图所示。



6. 在代码托管首页单击代码仓库所在行“SSH”复制SSH协议地址，如图所示。



7. 进行代码Clone。
   执行**git clone 代码库SSH协议地址**，如图6-3所示。

图 6-3 clone 代码

```
[root@        ~]# git clone git@codehub-                                                    1/master.git
Initialized empty Git repository in /root/master/.git/
\^Hremote: Counting objects: 16, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 16 (delta 2), reused 0 (delta 0)
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (2/2), done.
```

6.2 此密钥已存在，请重新生成密钥

问题现象

添加SSH密钥时，提示“此密钥已存在，请重新生成密钥”。

原因分析

每个密钥只能对应一个用户名，通常是由于密钥在其它帐户下被添加过。

处理方法

- 找到添加过该密钥的用户，删除密钥。
- 重新生成一次SSH密钥。

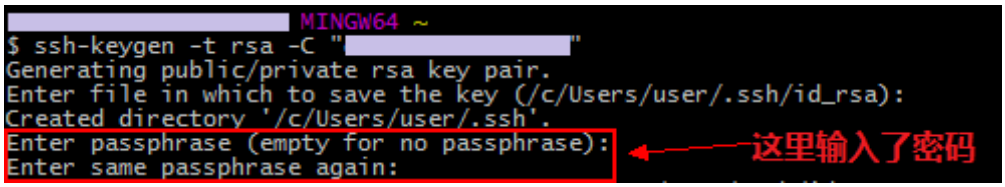
6.3 使用 SSH 协议克隆代码时一直提示输入密码

问题现象

使用SSH协议克隆代码时，一直提示输入密码。

原因分析

由于在生成SSH密钥时，添加了密码，需要输入的是SSH密钥的密码。



处理方法

- 输入在生成SSH密钥时设置的密码。
- 删掉SSH密钥，重新生成一次，在上图标记出的位置按回车键，不输入任何内容。

6.4 使用 HTTPS 方式克隆代码时，输入登录密码报错

问题现象

使用HTTPS方式克隆代码时，输入登录密码后一直报错。

原因分析

密码错误。

处理方法

登录您的代码托管服务仓库列表页，单击右上角昵称，单击“个人设置 > HTTPS密码管理”，进入页面，重置HTTPS密码即可。

说明

1. 使用HTTPS方式连接代码仓库时，用户名要输入完整用户名（xxx/xxx）。
2. 邮箱验证码通过单击“发送邮箱验证码”获取。
3. 忘记密码，设置新的HTTPS密码即可。

6.5 提交代码之后，仓库找不到文件

问题现象

使用git commit提交代码之后，仓库中找不到文件。

处理方法

git commit命令是将本地修改过的文件提交到本地库中，若想推送到云端仓库中，需要使用**git push**命令。

6.6 pathspec XXX did not match any files

问题现象

使用**git add**命令时，提示“pathspec XXX did not match any files”。

```
MINGW64 /d/GitStudy (master)
$ git add README.md
fatal: pathspec 'README.md' did not match any files
```

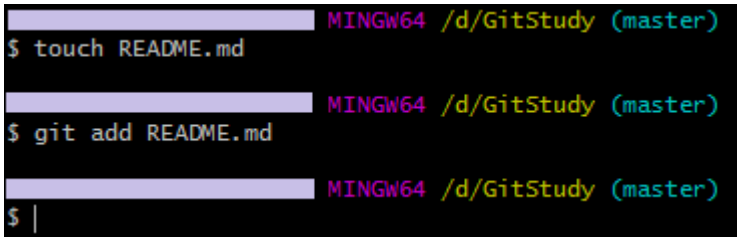
原因分析

本地并不存在“README.md”文件。**git add**命令是将已经存在的文件添加到暂存区，并没有创建文件的功能。

处理方法

方法一：在仓库中手动创建“README.md”文件，再进行**add**操作。

方法二：使用命令**touch README.md**创建文件，再进行**add**操作。



6.7 Transport Error: cannot get remote repository refs. XXX.git: cannot open git-upload-pack

问题现象

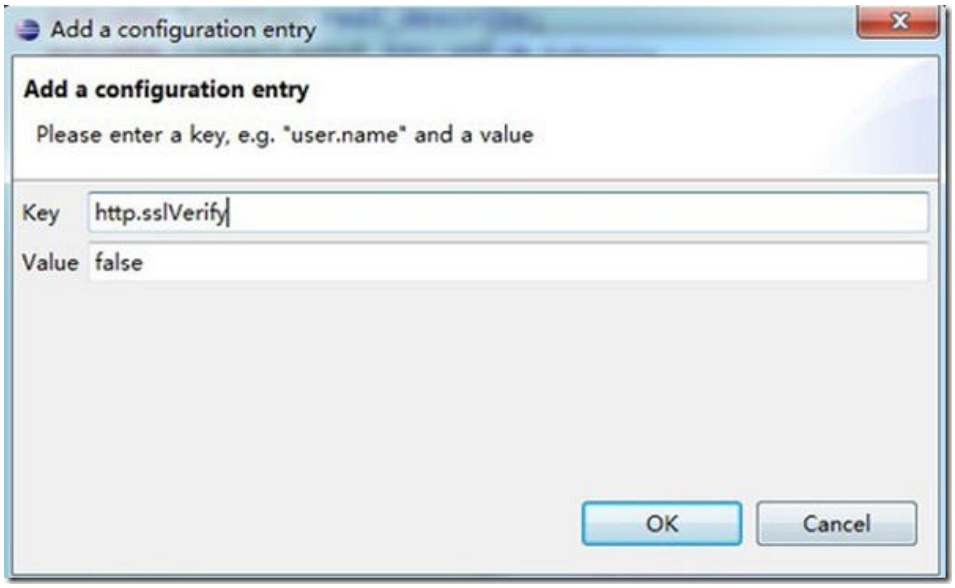
在Eclipse使用HTTPS方式连接云端仓库时，提示“Transport Error: cannot get remote repository refs. XXX.git: cannot open git-upload-pack”。

原因分析

由于Eclipse中Egit插件的配置问题。

处理方法

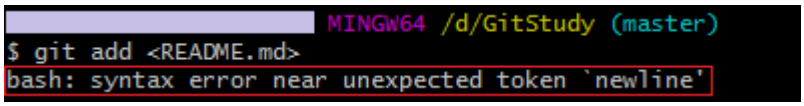
1. 在本地开发工具Eclipse中，选择“Windows > Preferences > Team > Git > Configuration > User Settings”。
2. 单击“Add Entry”，添加一个键值对：**http.sslVerify=false**



6.8 syntax error near unexpected token `newline'

问题现象

使用**git add**命令时，提示“syntax error near unexpected token `newline'”。



原因分析

命令中有无效的占位符“<>”。

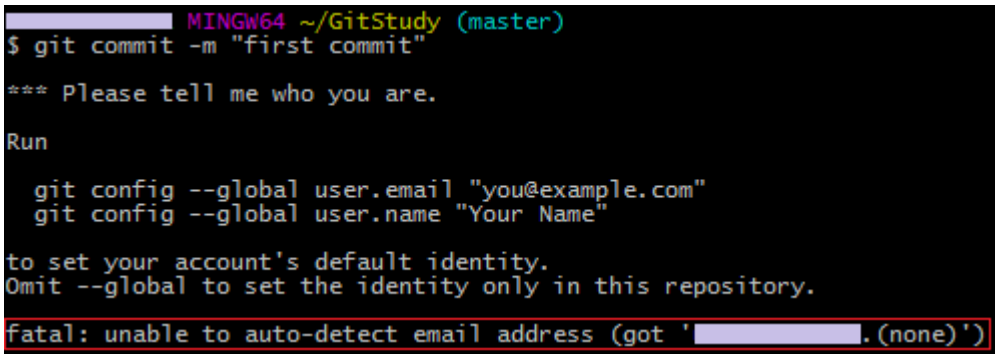
处理方法

将“<>”符号去掉即可，即输入命令： **git add README.md**。

6.9 unable to auto-detect email address

问题现象

使用**git commit**命令后，提示“unable to auto-detect email address”。



原因分析

由于没有进行用户名、邮箱设置所致。

处理方法

1. 执行下面两行命令完整设置。

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

2. 运行**git commit**即可。

6.10 fatal:Authentication failed

问题现象

使用HTTPS方式克隆代码，即**git clone**时，提示“fatal:Authentication failed”。

原因分析

提示“Authentication failed”认证失败，通常是由于HTTPS用户名密码填写错误导致。

处理方法

请检查填写的用户名密码是否正确。

6.11 'origin' does not appear to be a git repository

问题现象

向云端仓库推送代码时，提示“'origin' does not appear to be a git repository”。

原因分析

云端的代码仓库没有初始化成功。

处理方法

使用 **git push -u origin master**命令进行强制关联并推送。

6.12 You are not allowed to push code to protected branches on this project

问题现象

向代码仓库推送代码时，提示“You are not allowed to push code to protected branches on this project”。

原因分析

该分支为受保护分支，用户没有权限推送代码到这个分支。

处理方法 1：修改分支保护设置

进入代码仓库详情页，选择“设置 > 策略设置 > 保护分支”，解除对该分支的保护。具体操作请参考《**用户指南**》中“**编辑保护分支**”。

图 6-4 编辑保护分支

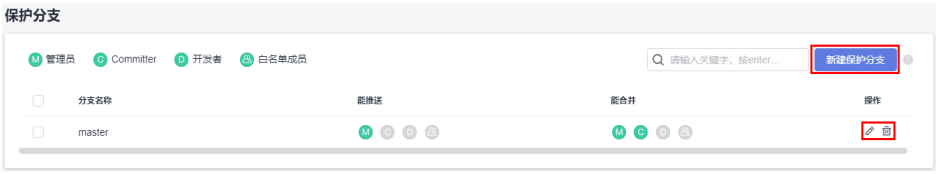


图 6-5 设置白名单

处理方法 2：修改仓库成员角色

进入代码仓库详情页，单击“成员”页签，搜索目标用户，单击设置管理员图标，修改仓库成员角色，将该成员设置为仓库管理员。具体操作请参考《**用户指南**》中“**配置成员管理**”。

图 6-6 成员列表

6.13 Not a git respository

问题现象

使用**git add**命令时，提示“Not a git respository”。

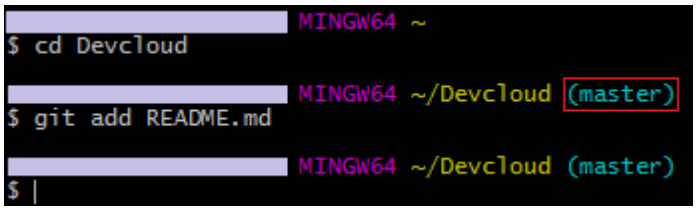
```
MINGW64 ~
$ git add README.md
fatal: Not a git repository (or any of the parent directories): .git
```

原因分析

当前所处的目录并非代码仓库目录。

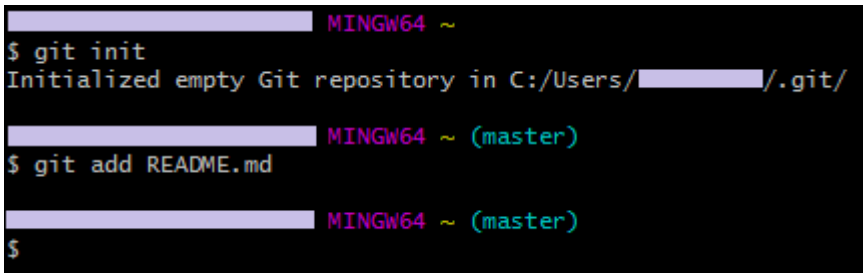
处理方法 1

需要找到本地代码仓库目录地址，通过命令： **cd 仓库名称**，切换到代码仓库中再进行**add**操作。



处理方法 2

使用**git init**命令将当前目录初始化为代码仓库，再进行**add**操作。



6.14 src refspec master does not match any

问题现象

向代码仓库推送代码时，提示“src refspec master does not match any”。

原因分析

本地的版本库中没有任何可供上传的文件。

处理方法

使用add与commit命令，将修改后的文件提交至暂存区中，再使用push命令推送至云端代码仓库中。

6.15 destination path 'XXX' already exists and is not an empty directory

问题现象

从云端克隆代码时，提示“destination path 'XXX' already exists and is not an empty directory”。

原因分析

所克隆的代码仓库已存在并且不为空。

处理方法

克隆项目至新目录。

1. 进入目录：
   ```
   cd 本地代码仓库目录
   ```
2. 克隆云端代码仓库到临时目录tmp中：
   ```
   git clone --no-checkout 云端代码仓库地址tmp
   ```
3. 将tmp目录下的 .git 目录移到当前目录：
   ```
   mv tmp/.git 本地代码仓库目录名
   ```
4. 删除tmp目录：
   ```
   rmdir tmp
   ```
5. 回退至上个版本：
   ```
   git reset --hard HEAD
   ```

6.16 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“The requested URL returned error: 401”。

原因分析

通常是Git版本的问题。

处理方法

通过**git --version**命令查看系统自带的版本，**Centos6.5**自带的Git版本通常是1.7.1。

1. 卸载Centos自带的Git1.7.1。
   ```
   # yum remove git
   ```
2. 到**Git官网**下载最新版本，并将Git添加到环境变量中。
   ```
   # wget https://github.com/git/git/archive/版本号.tar.gz
   # tar zxvf 版本号.tar.gz
   # cd git-版本号
   # make configure
   # ./configure --prefix=/usr/local/git --with-iconv=/usr/local/libiconv
   # make all doc
   ```



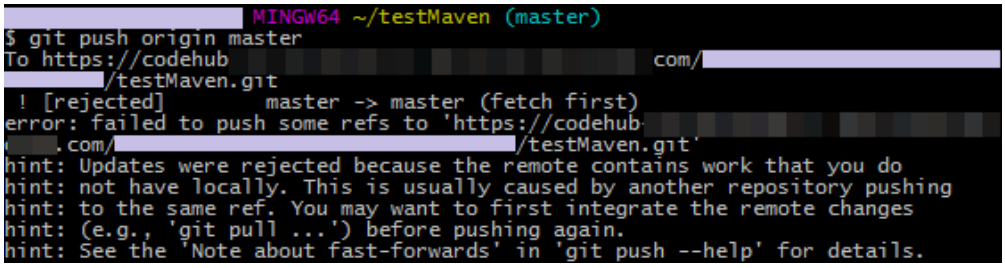
```
# make install install-doc install-html
# echo "export PATH=$PATH:/usr/local/git/bin" >> /etc/bashrc
# source /etc/bashrc
```

6.17 向代码仓库推送代码失败

问题现象

向代码仓库推送代码时，系统提示提交失败，提交失败提示信息如下图所示。

图 6-7 提示信息



原因分析

从提示信息（hint）看，由于云端代码仓库有变化，与本地仓库代码不一致，所以从本地提交代码的操作被拒绝。

处理方法

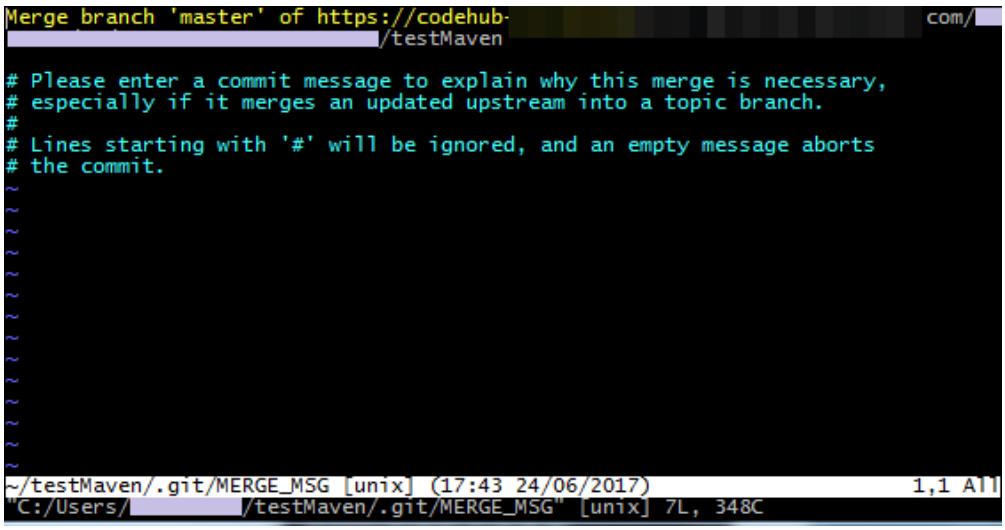
先使用 **git pull**命令拉取云端仓库，与本地仓库合并，再使用push命令推送代码。

6.18 使用 git pull 拉取代码失败

问题现象

使用**git pull**命令拉取代码失败，无法继续操作，拉取代码失败信息如下图所示。

图 6-8 提示信息



原因分析

由于云端仓库与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），弹框是提示是否确认本次merge操作，并提交备注信息。

处理方法

在Git客户端中进行以下操作：

1. 按键盘字母**i**进入**insert**模式。
2. 输入确认信息。
3. 输入**:wq**，按回车键保存退出。

6.19 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

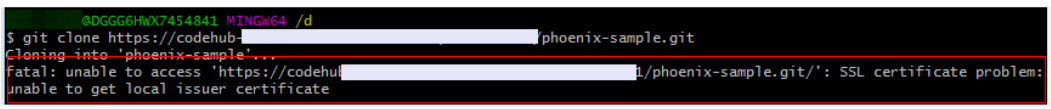
使用命令 **git pull origin master --allow-unrelated-histories**进行强制合并。

6.20 SSL certificate problem

问题现象

克隆代码时，提示“SSL certificate problem: Invalid certificate chain”或“SSL certificate problem: unable to get local issuer certificate”，如图6-9所示。

图 6-9 问题现象



原因分析

SSL证书问题：无法获取本地颁发者证书。

处理方法

通常是由于用户处于Proxy之后的内网环境所致，简单方案为：

```
git config --global http.sslVerify false
```

