**GitLab使用手册**

**目录**

[一 Gitlab账号/库申请流程 3](#_Toc388625687)

[1.1 Gitlab账号申请 3](#_Toc388625688)

[1.2 Gitlab库申请 3](#_Toc388625689)

[二 Gitlab登录 3](#_Toc388625690)

[2.1 Gitlab 访问路径 3](#_Toc388625691)

[2.2 Gitlab登录页面 3](#_Toc388625692)

[三 Git环境配置 4](#_Toc388625694)

[3.1创建公钥 4](#_Toc388625695)

[3.1.1 Linux环境 4](#_Toc388625696)

[3.1.2 Windows环境 4](#_Toc388625697)

[3.1.3 MacOS环境 7](#_Toc388625698)

[3.2 Gitlab SSH页面配置 7](#_Toc388625699)

[四 Git操作命令 8](#_Toc388625700)

[4.1 Git本地仓库操作命令 8](#_Toc388625701)

[4.1.1 git init/git clone 初始化库 8](#_Toc388625702)

[4.1.2 git status 查看状态 9](#_Toc388625703)

[4.1.3 git add 添加文件 9](#_Toc388625704)

[4.1.4 git diff 对比文件 9](#_Toc388625705)

[4.1.5 git commit 提交更新 10](#_Toc388625706)

[4.1.6 git rm 移除文件 10](#_Toc388625707)

[4.1.7 git mv 移动文件 11](#_Toc388625708)

[4.1.8 git log 查看提交历史 11](#_Toc388625709)

[4.1.9 git reset 撤销操作 12](#_Toc388625710)

[4.1.10 git branch 创建分支 13](#_Toc388625711)

[4.1.11 git merge 分支合并 13](#_Toc388625712)

[4.1.12 git conflict 解决冲突 14](#_Toc388625713)

[4.1.13 git tag 创建标签 14](#_Toc388625714)

[4.2 Git远程仓库关联操作命令 15](#_Toc388625715)

[4.2.1 git remote 查看远处仓库 15](#_Toc388625716)

[4.2.2 git remote show 查看远程仓库信息 15](#_Toc388625717)

[4.2.3 git remote add 添加远程仓库 15](#_Toc388625718)

[4.2.4 git fetch 从远程仓库抓数据 16](#_Toc388625719)

[4.2.5 git pull 获取最新版本 16](#_Toc388625720)

[4.2.6 git push 推送数据到远程仓库 16](#_Toc388625721)

[4.2.7 git remote rename/rm远程仓库重命名/删除 17](#_Toc388625722)

# 一Gitlab账号/库申请流程

## 1.1 Gitlab账号

使用邮箱前缀、邮箱密码访问即可。

## 1.2 Gitlab库申请

邮件申请。（发邮件给zhaoweiz@、cuixf1@ ）

# 二Gitlab登录

## 2.1 Gitlab访问路径

**URL:**<http://gitlab.rd.chanjet.com/>

## 2.2 Gitlab登录页面



# 三Git环境配置

## 3.1创建公钥

不同操作系统创建公钥的方式不同。以下是三种操作系统的举例。

### 3.1.1 Linux环境

$ssh-keygen //生成公钥

$ cat ~/.ssh/id\_rsa.pub //将公钥文件内容拷贝到Gitlab配置页面中。见3.2节。

**注意：Vim 文件复制粘贴可能会导致内容换行，导致公钥无效。**

### 3.1.2 Windows环境

**需要下载msysgit和tortoisegit客户端**

第一步，下载msysgit并安装。（基本上一路Next。因为我们还要安装tortoisegit，所以不需要选择“Windows explorer integration”。）

URL:<https://code.google.com/p/msysgit/downloads/list?can=2&q=%22Full+installer+for+official+Git+for+Windows%22>

或者公司共享目录：[\\devshare\工具共享\常用软件\git](file:///\\devshare\工具共享\常用软件\git)

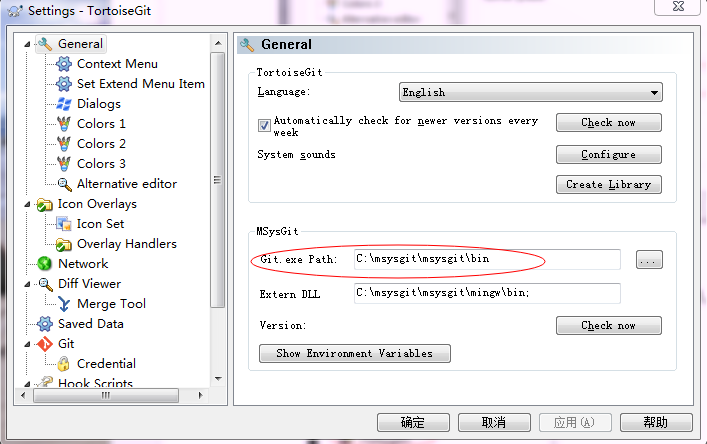
第二步，下载tortoisegit并安装

URL:

<https://code.google.com/p/tortoisegit/wiki/Download?tm=2>

或者公司共享目录：[\\devshare\工具共享\常用软件\git](file:///\\devshare\工具共享\常用软件\git)

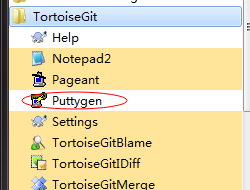
第三步，在tortoisegit上配置msysgit的路径（见下图）



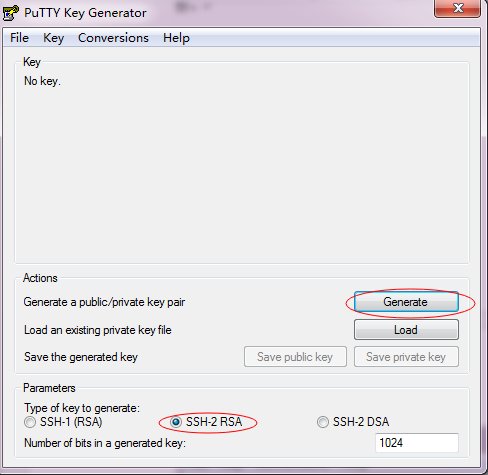
**创建公钥的方式（二者选一即可）**

（1）通过tortoiseGit方式生成公钥

a) 从Windows开始菜单选择TortoiseGit，点击其下的Puttygen工具，该工具用于生成Putty支持的SSH密匙对。

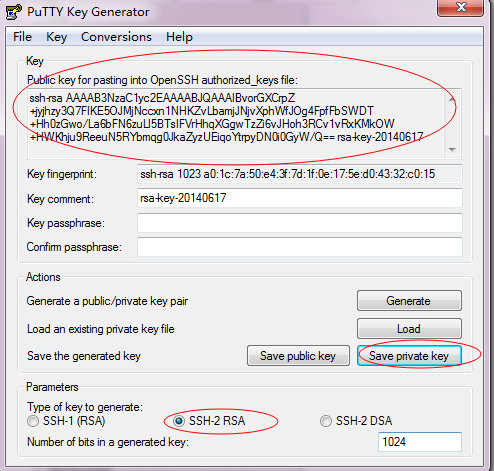


b) 点击Generate按钮，即生成公钥和私钥。将Public key（公匙）和Private key（私匙）都保存到文档中，以免丢失。其中Private key是你的TortoiseGit用来验证用户身份。



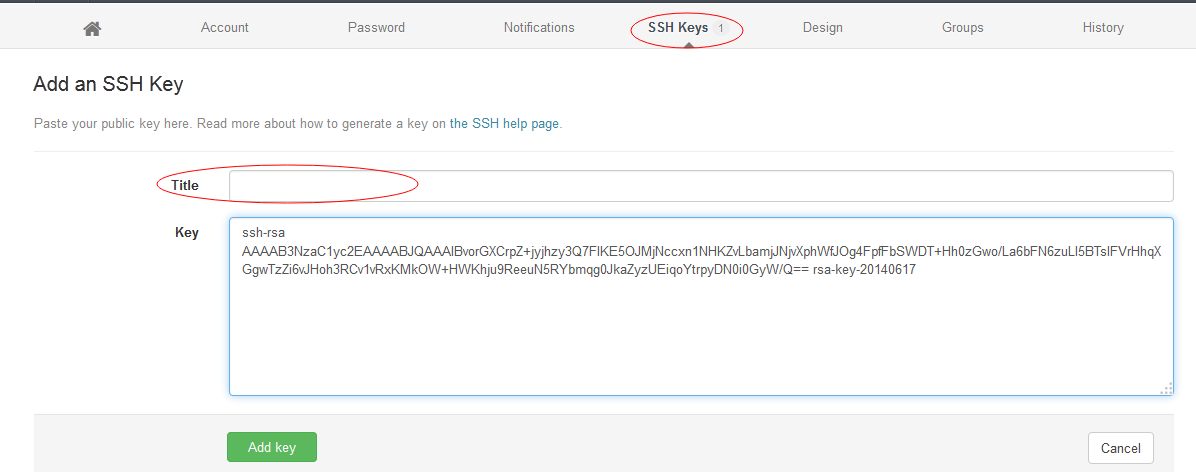
说明：

1.生成key的类型：SSH-2 RSA



说明：

1. 将生成的Public key粘贴到GitLab站点“申请者帐号”里。配置方式见3.2节。（由于tortoisegit生成的公钥粘贴到’gitlab SSH Key’中并不能够自动生成’title’，因此需要自己填写一个’title’）



1. 保存”Save private key”到本地。

（2）通过命令行方式生成公钥

a)点击‘开始’，在搜索中输入‘cmd’。



b)$ssh-keygen //生成公钥

c) 将生成的Public key粘贴到GitLab站点“申请者帐号”里。配置方式见3.2节。

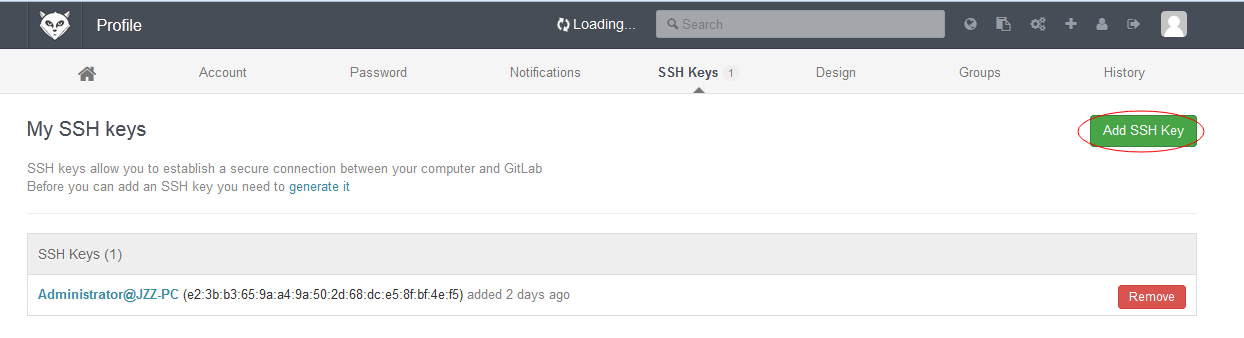
### 3.1.3 MacOS环境

$ssh-keygen //生成公钥

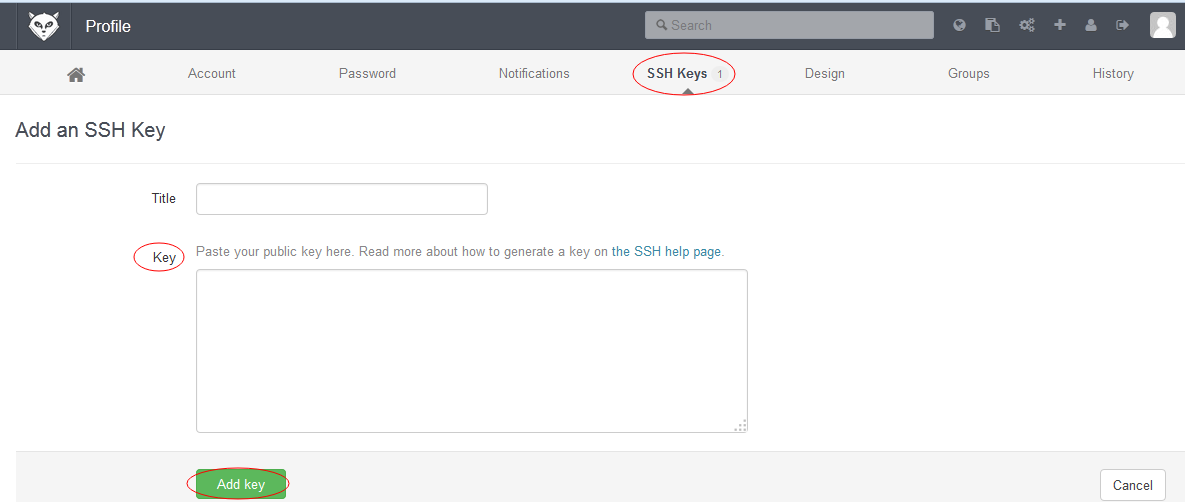
$ cat ~/.ssh/id\_rsa.pub //将公钥文件内容拷贝到Gitlab配置页面中。见3.2节。

## 3.2 Gitlab SSH页面配置

Gitlab “Add SSH Key”界面如下：



将3.1节创建的公钥拷贝到”Key”方框中：



# 四Git操作命令

## 4.1 Git本地仓库操作命令

* gitinit/git clone 【初始化库】
* git status 【查看状态】
* git add【添加文件】
* git diff【对比文件】
* git commit【提交更新】
* gitrm【移除文件】
* git mv【移动文件】
* git log 【查看提交历史】
* git reset 【撤销操作】
* git branch 【创建分支】
* git merge 【分支合并】
* git conflict 【解决冲突】
* git tag 【创建标签】

### 4.1.1 gitinit/git clone 初始化库

* 初始化一个新库

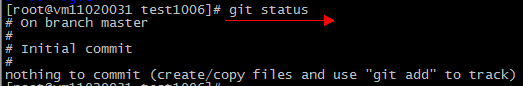
$ gitinit

* 从现有仓库克隆

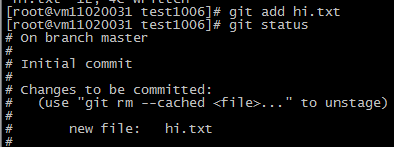
$git clone URL

### 4.1.2 git status 查看状态

* 若没有可跟踪的文件，用git status命令，则会输出：

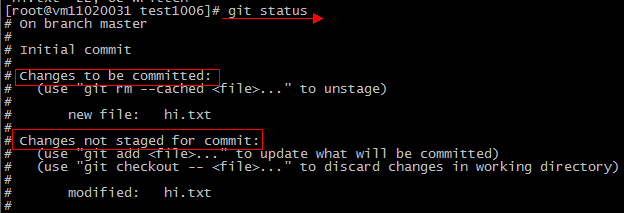
****

* 若编辑一个新文件hi.txt,保存退出。用git add命令跟踪文件后运行git status命令，则输出：



说明:只要在“Changes to be committed”下面，就表示为已暂存状态。

* 若修改已经暂存状态的文件，保存退出后用git status命令，则会输出：



说明：hi.txt 文件出现了两次！一次算未暂存，一次算已暂存。需要将未暂存的文件通过git add添加到已暂存。

### 4.1.3 git add 添加文件

作用：

可以用它开始跟踪新文件；

把已经跟踪的文件放到暂存区

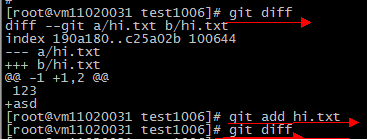
合并时把有冲突的文件标记为已解决状态。

### 4.1.4 git diff 对比文件

作用：

当前作的哪些更新还没有暂存；

有哪些更新已经暂存起来准备好了下次提交。



说明：

有时候一下子暂存了所有更新过的文件后，运行git\_diff后却什么也没有。原因是，git\_diff仅显示还没有暂存起来的改动，而不是这次工作和上次提交之间的差异。

### 4.1.5 git commit 提交更新

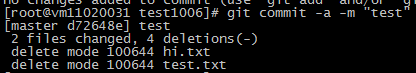
在提交之前，一定要用git status确认还有哪些修改过的或是新建的文件还没有暂存起来。否则，这些没有暂存的文件就不能将最新变化提交到版本库中。

参数介绍：

git commit –m 使用-m 参数后跟提交说明的方式，在一行命令中提交更新

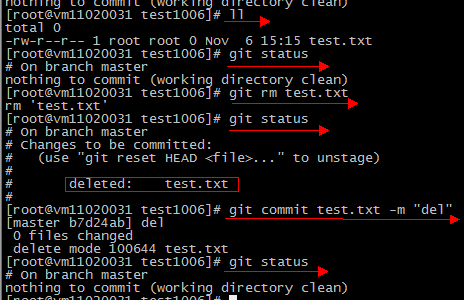


git commit –a 自动把所有已经跟踪过的文件暂存起来一并提交，跳过使用暂存区域提交更新



### 4.1.6 gitrm移除文件

删除已提交的某个文件，需要先用gitrm命令从暂存区域中移除这个文件，然后再用git commit 命令提交。



### 4.1.7 git mv 移动文件

Git中的重命名某个文件，仓库中存储的元数据并不会体现出这是一次改名操作。

文件改名并查看当前状态：



运行git mv就相当于运行了下面三条命令：

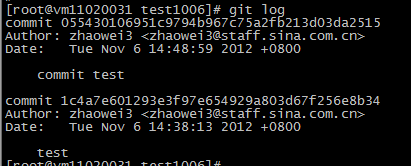
$mv test\_from.txt test\_to.txt

$gitrm test\_from.txt

$git add test\_to.txt

### 4.1.8 git log 查看提交历史

每次更新都有一个SHA-1 校验和、作者的名字和电子邮件地址、提交时间，最后缩进一个段落显示提交说明。



常用查找历史日志的选项：

-p:项展开显示每次提交的内容差异；

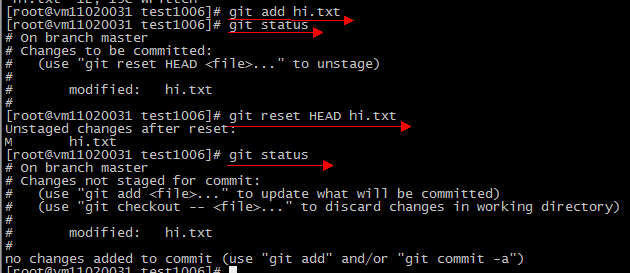
-n(数字)：仅显示最近的n次更新。

在做代码审查，或者要快速浏览其他协作者提交的更新都作了哪些改动时，就可以用这个选项。

### 4.1.9 git reset 撤销操作

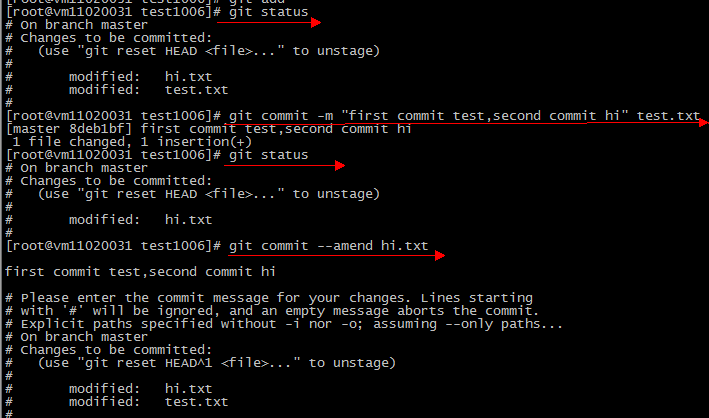
* 取消已暂存的文件

若不小心将某个文件添加/删除到了暂存区域，可以用“git reset HEAD <file>...”命令取消暂存。



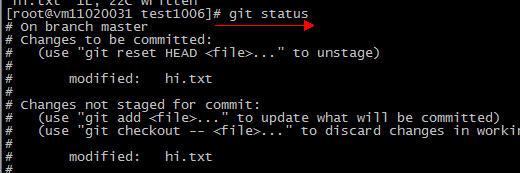
* 修改最后一次提交

提交后发现漏掉了一些文件，想要撤销刚才的提交。若刚才提交完后没有做任何改动，可以使用—amend选项重新提交。

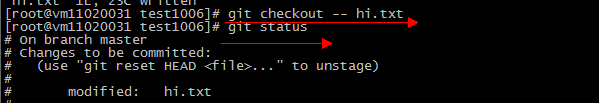


* 取消工作目录中已经修改的文件

若已经添加到暂存区域的文件又进行了修改，但这些改动又觉得是没有必要，则可以用“git checkout -- <file>...”恢复到修改前的版本。

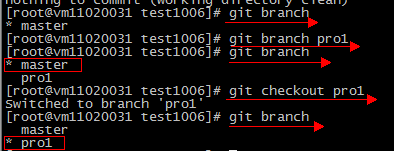


**$vim hi.txt**



### 4.1.10 git branch 创建分支

创建一个test分支，然后将目录切换到test分支下面。



注意：test分支前的“\*”字符：表示当前所在的分支。

git checkout –b 分支名

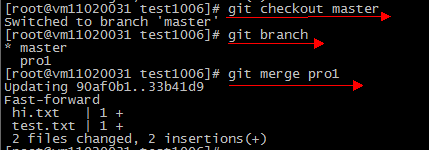
相当于：

git branch 分支名

git checkout 分支名

### 4.1.11 git merge分支合并

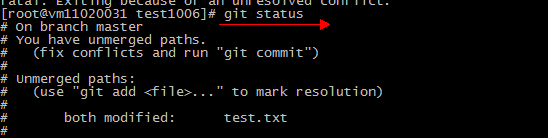
首先查看当前目录所有分支清单，然后切换到master分支目录下，最后将pro1分支内容合并到master上。



### 4.1.12 git conflict 解决冲突

* 如果修改了两个待合并的分支里同一个文件的同一个部分，则会出现如下报错：





* 如何编辑冲突：

首先查看当前目录下的状态（git status），然后编辑冲突文件(vim test.txt),再将冲突文件置为解决状态（git add test.txt），最后将合并的文件提交给(git commit –m “注释”)。

冲突文件test.txt，打开后可以看到：

<<<<<<< HEAD

bra1-11111222

=======

bra1-qwerea

>>>>>>> pro1

说明：======= 隔开的上半部分，是HEAD（即master分支）中的内容，下半部分是在test分支中的内容。

### 4.1.13 git tag 创建标签

* 列出所有标签

git tag

* 搜索标签

git tag –l 标签名

* 创建标签

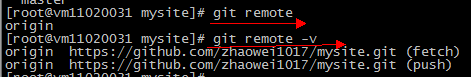
git tag –a 标签名–m 注释

## 4.2 Git远程仓库关联操作命令

* git remote【查看远处仓库】
* git remote show 【查看远程仓库信息】
* git remote add 【添加远程仓库】
* git fetch 【从远程仓库抓数据】
* git pull 【获取最新版本】
* git push 【推送数据到远程仓库】
* git remote rename/rm【远程仓库重命名/删除】

### 4.2.1 git remote 查看远处仓库

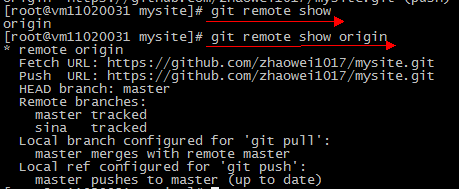
用git remote 命令，查看当前配置有哪些远程仓库。加上参数-v，可以列出所有的远程仓库。



参数介绍：

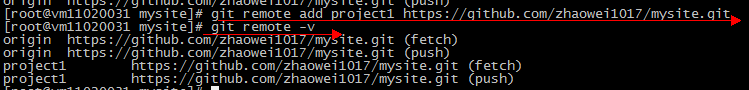
-v：可以列出所有的远程仓库。

### 4.2.2 git remote show 查看远程仓库信息



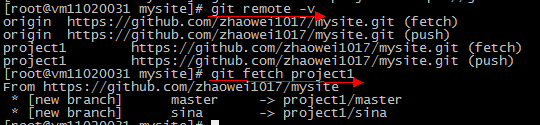
### 4.2.3 git remote add 添加远程仓库

若要添加一个新的远程仓库，运行git remote add [shortname] [url]



### 4.2.4 git fetch 从远程仓库抓数据

命令：$git fetch proname



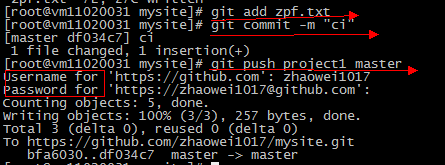
### 4.2.5 git pull 获取最新版本



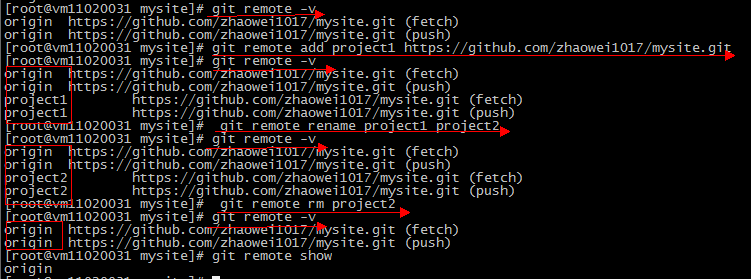
### 4.2.6 git push 推送数据到远程仓库

如果要把本地的内容推送到Git服务器maste分支上.

命令:$ git push origin master



### 4.2.7 git remote rename/rm远程仓库重命名/删除



注意，对远程仓库的重命名，也会使对应的分支名称发生变化，原来的pb/master 分支现在成了paul/master。

# 五Windows Git操作