# Git简介

Git是一款免费、开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。可以有效、高速的处理从很小到非常大的项目版本管理，它是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。



# Git功能特性

从一般开发者的角度来看，git有以下功能：

1、从服务器上克隆完整的Git仓库（包括代码和版本信息）到单机上。

2、在自己的机器上根据不同的开发目的，创建分支，修改代码。

3、在单机上自己创建的分支上提交代码。

4、在单机上合并分支。

5、把服务器上最新版的代码fetch下来，然后跟自己的主分支合并。

6、生成补丁（patch），把补丁发送给主开发者。

7、看主开发者的反馈，如果主开发者发现两个一般开发者之间有冲突（他们之间可以合作解决的冲突），就会要求他们先解决冲突，然后再由其中一个人提交。如果主开发者可以自己解决，或者没有冲突，就通过。

8、一般开发者之间解决冲突的方法，开发者之间可以使用pull 命令解决冲突，解决完冲突之后再向主开发者提交补丁。

# Git和SVN的区别

1. GIT是分布式版本控制系统，SVN是集中式版本控制系统。
2. GIT在离线情况下依然可以提交修改，而SVN不可。

GIT在离线情况下依然可以提交修改,查看版本信息和日志，只需要在合适的时候将代码推送到服务器上。而SVN需要联网才能操作。也就是说在服务器宕机的情况下，GIT依然可以在本地操作，引文本地就相当于服务器上的一个完整备份，而SVN则不行。

1. GIT分支和SVN分支不同。

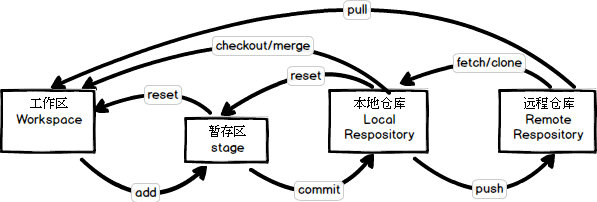
针对一个项目如果有50M，那么三个分支可能会占用150M。而GIT不一样，即使创建10个分支，可能只需要51M，而不是成倍增长。

鉴于以上创建3个分支，SVN可能需要20分钟，而GIT创建分支只需要秒级的时间量。

1. GIT把内容按元数据方式存储，而SVN是按文件。
2. GIT没有一个全局的版本号，而SVN有
3. GIT的内容完整性要优于SVN

GIT的内容存储使用的是SHA-1哈希算法，它能够确保代码的完整性，确保在遇到磁盘故障和网问题时降低对版本库的破坏。

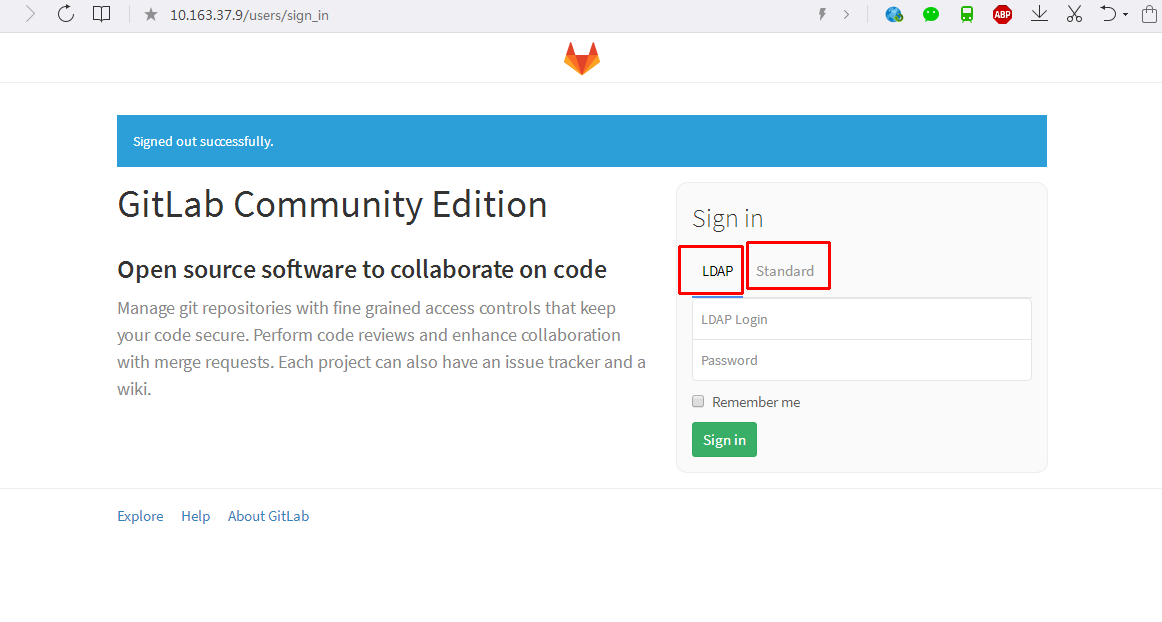
# Git流程图



# 使用Gitlab

## 注册用户和登录

登录<http://10.163.37.9>，有两种登录方式：ldap和standard。

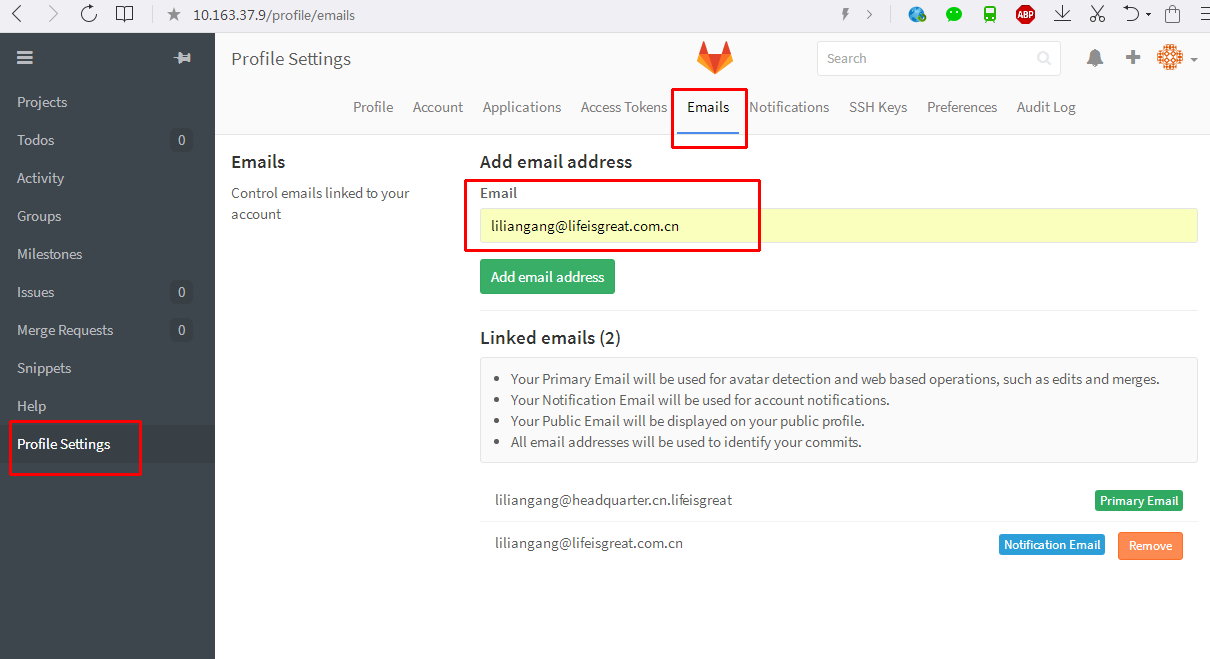


ldap方式是恒大内部人员登录入口，首次登录会系统会自动创建，用户为电脑开机名，密码为电脑登录密码。

standard方式是非恒大内部人员登录入口，如有注册用户需求，可向项目经理申报，由项目经理统一申请。

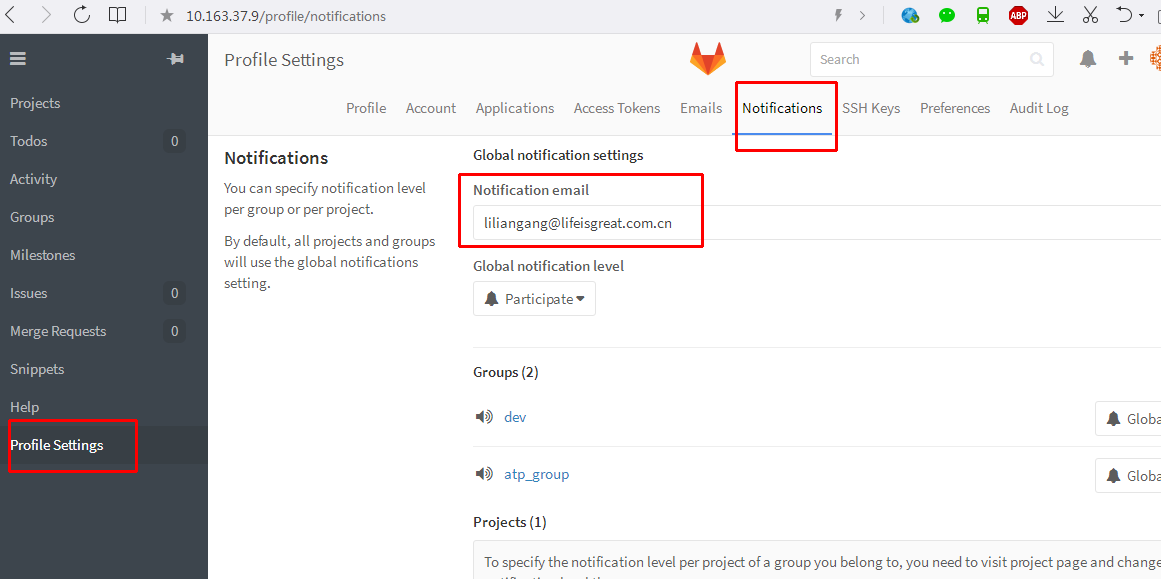
## 添加邮箱

进入个人信息的邮箱标签，添加新邮箱 [username@email.com](mailto:username@email.com)



## 修改接收提示消息的邮箱

（首次登录时，需查看提示消息邮箱是否正确，如不是，需修改）进入个人信息的提示信息标签。如下图，选择一个邮箱作为提示邮箱。



# 使用Git

## 安装Git

下载地址<https://git-scm.com/downloads/>

公盘路径：X:\devtool\Git-2.10.1-64-bit.exe

安装略。

## 配置Git用户信息

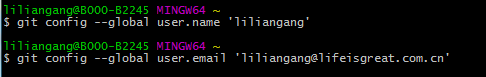
安装完git后，进入git-bash命令窗口，



然后配置用户名和email：

# git config --global user.name 'username'

# git config --global user.email 'name@lifeisgreat.com.cn'



# Git通用开发流程

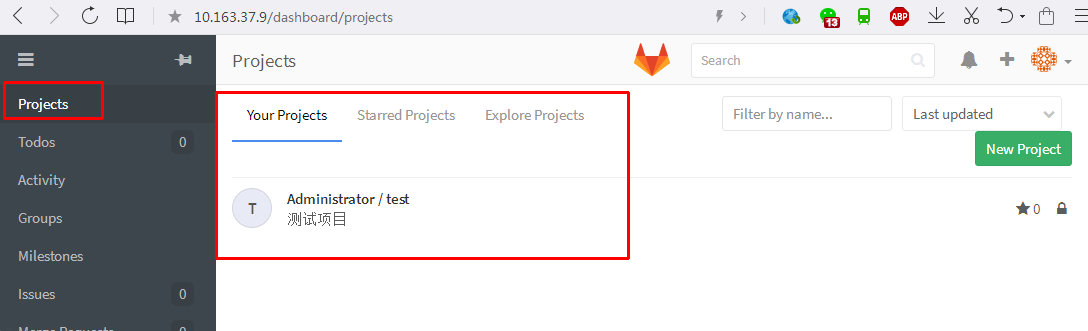
1. 将项目组的远程项目clone克隆到本地local
2. 将本地修改add到stage暂存区
3. 将stage暂存区的修改提交到本地local
4. 将项目组的远程项目pull到本地local
5. 合并merge代码，并再次提交到本地local
6. 将本地local项目push推送到远程库

**在push代码之前一般需要先pull拉取远程项目组代码，然后合并代码，以减少服务器代码冲突**

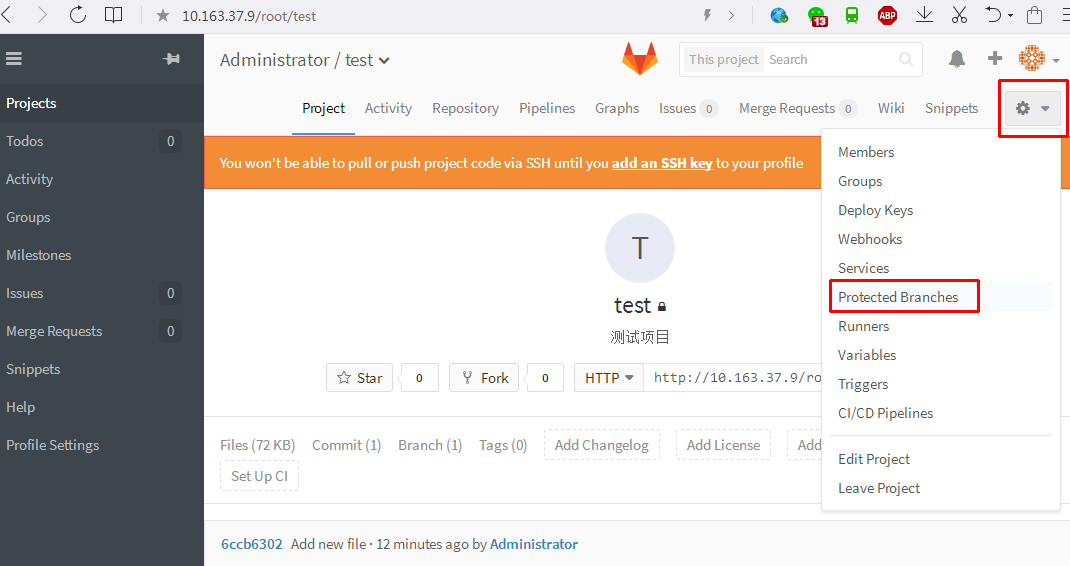
# 项目管理者操作

## 配置项目分支权限

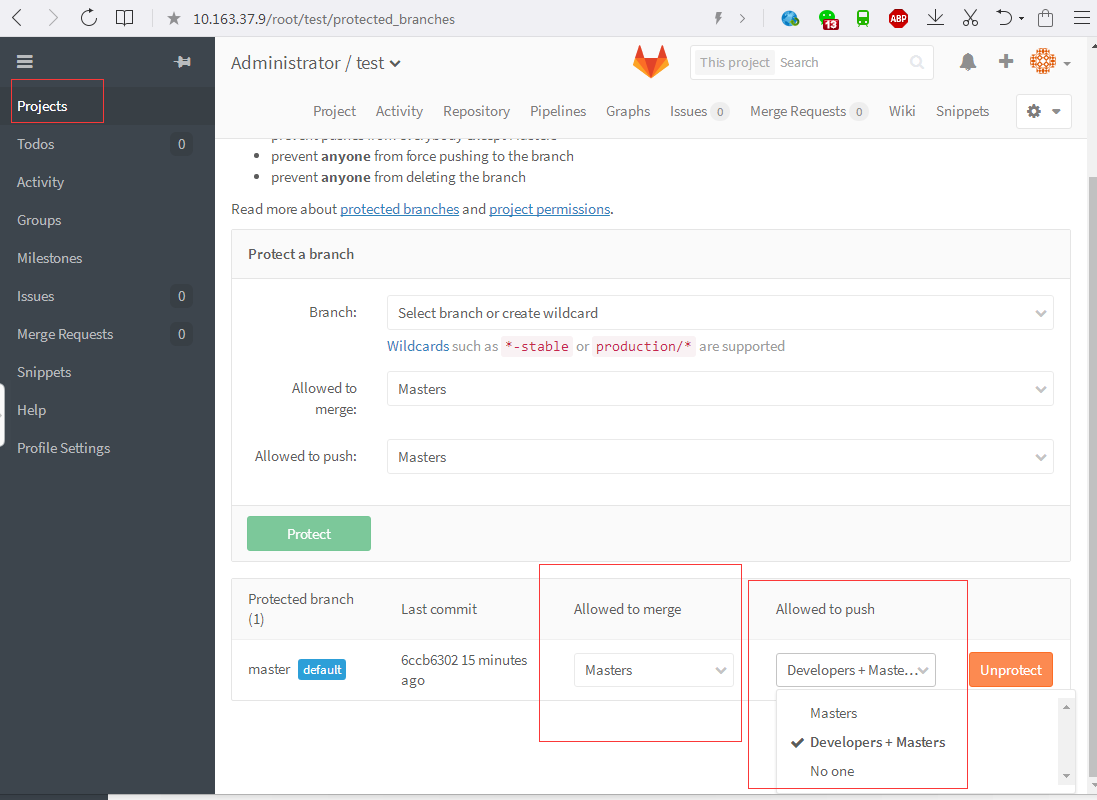
首先登录服务器<http://10.163.37.9> ，进入项目project页面，查看自己的项目，并且进入到项目工程中



进入项目分支权限管理页面。

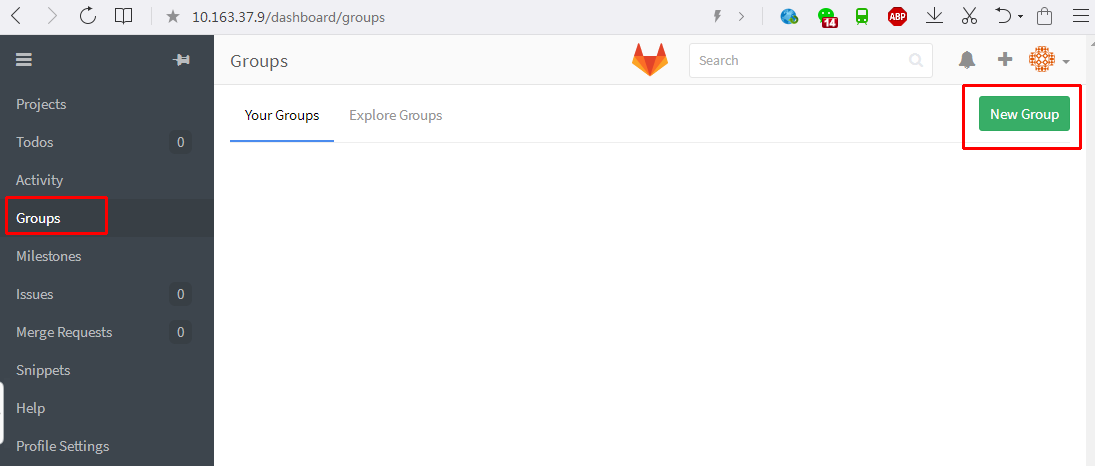


设置项目分支权限，目前推荐开发角色也可以推送push项目。

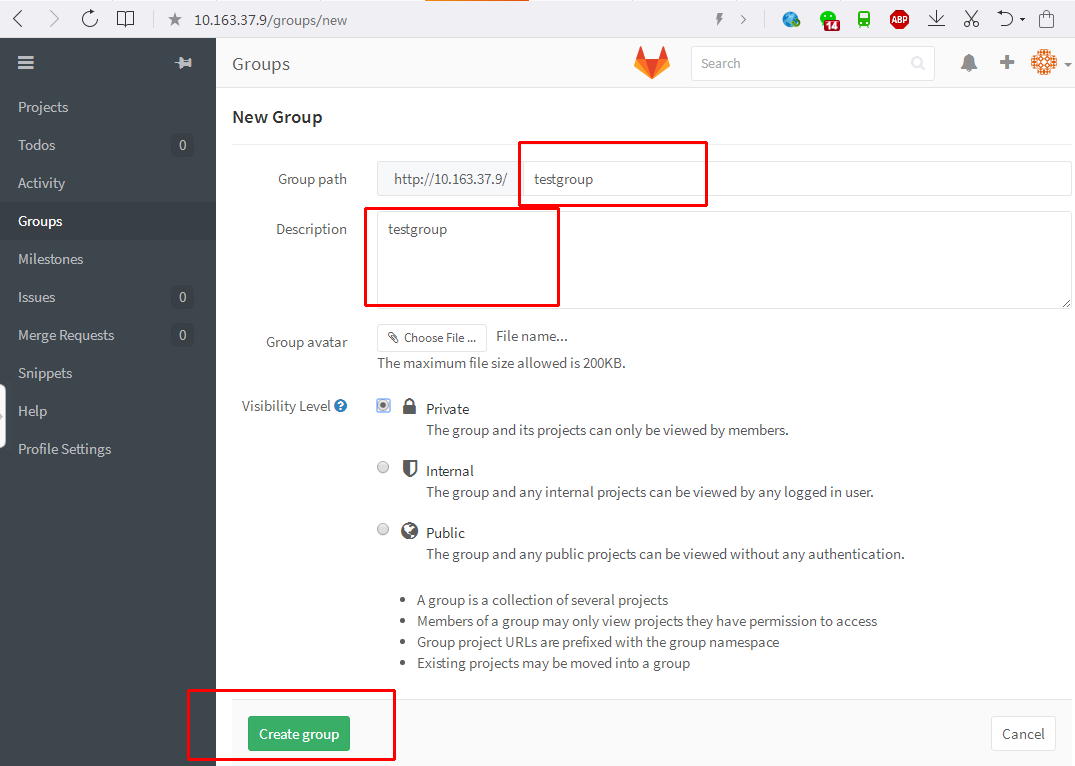


## 新增组Group

进入组管理页面，点击新增组按钮。

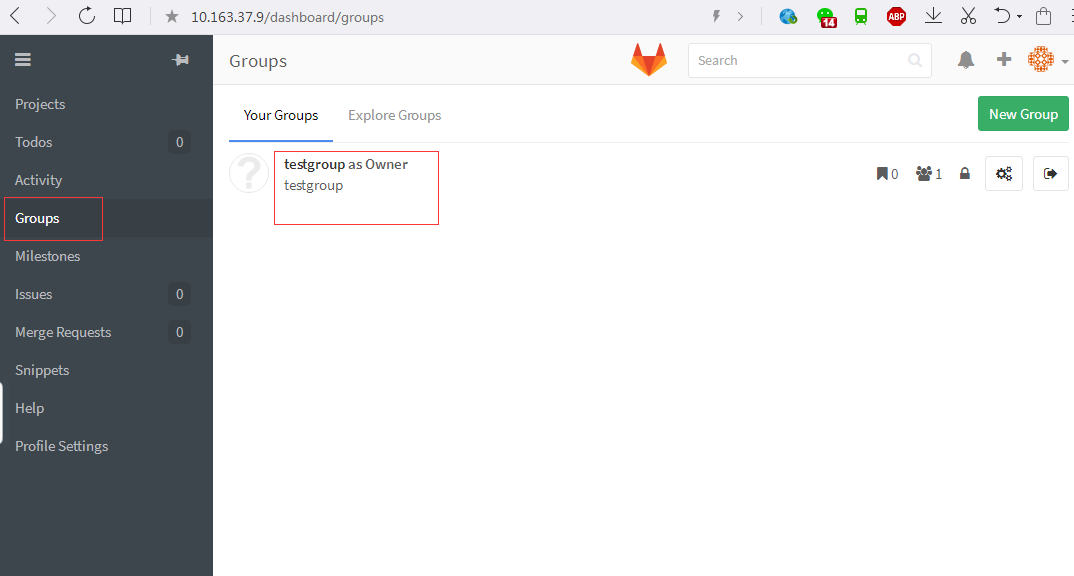


填写相应组信息，保存。

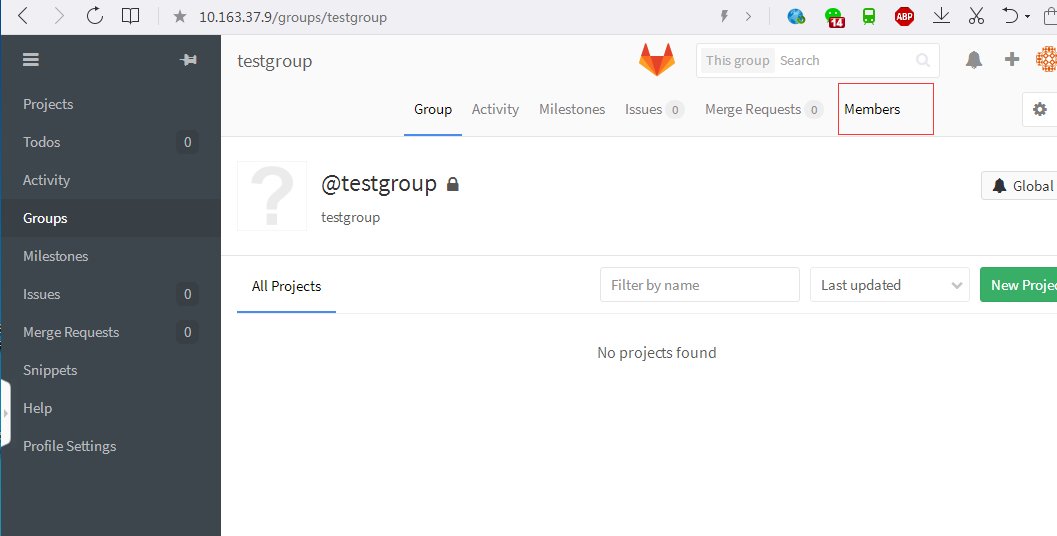


## 添加成员到组中

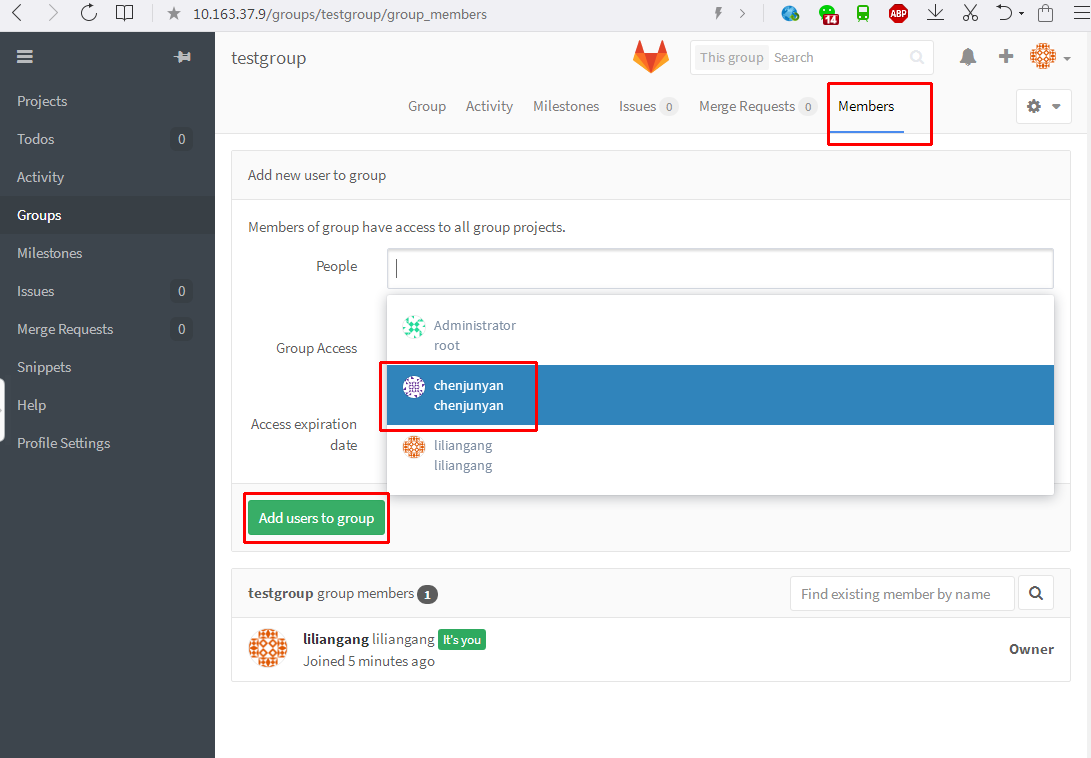
进入组界面



进入组成员编辑界面

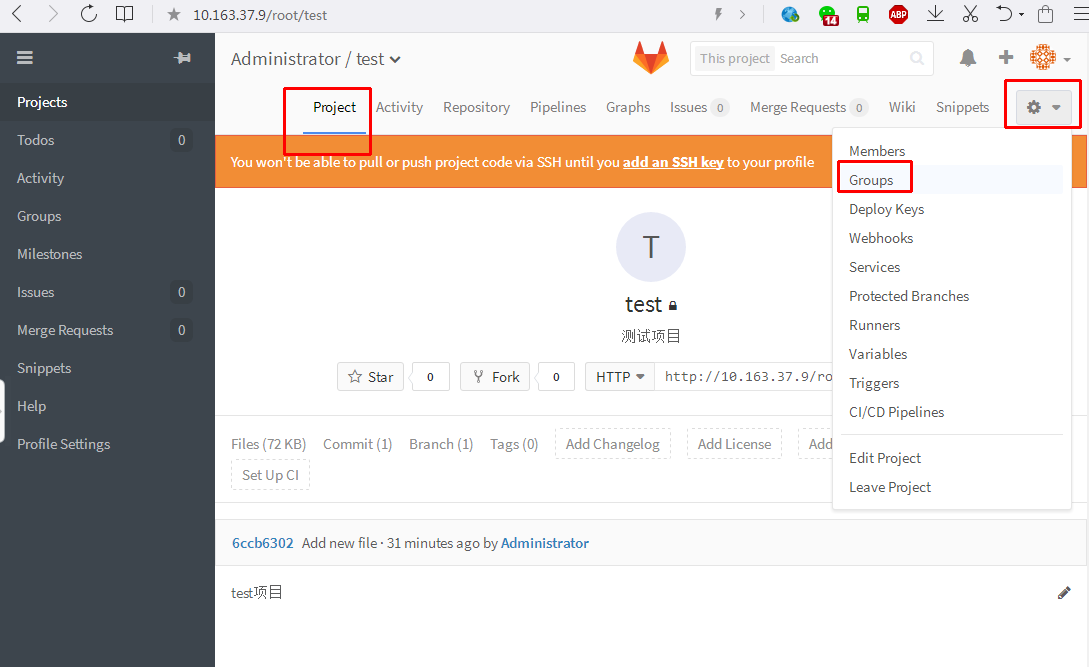


选择成员加入到自己组中即可。

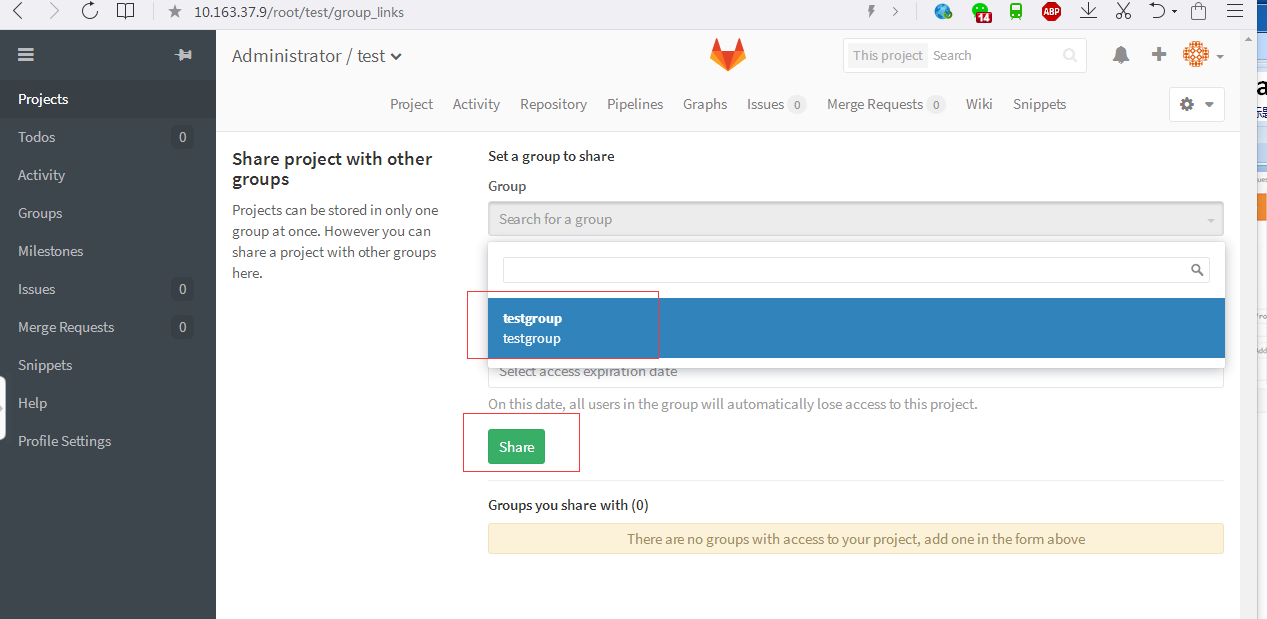


## 将项目分享给组

进入项目管理界面，点击组Group按钮。

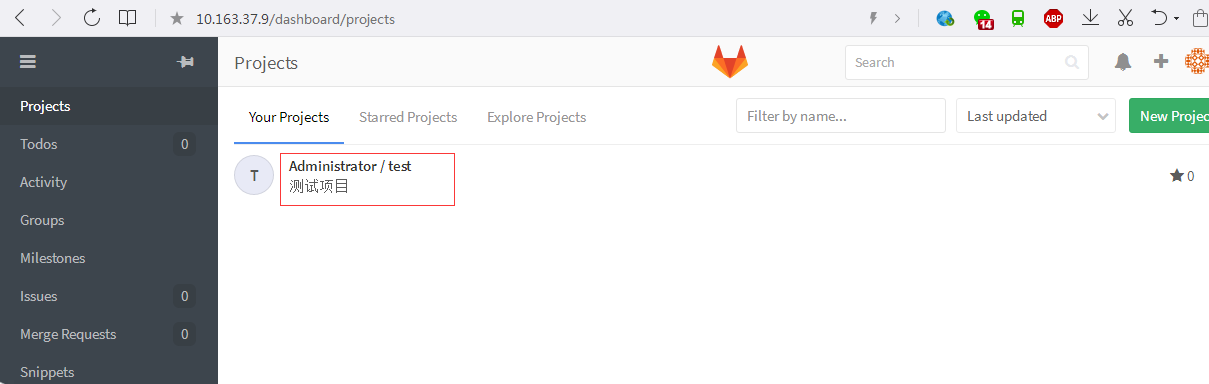


将项目分享给组，以便组成员都有权限操作项目

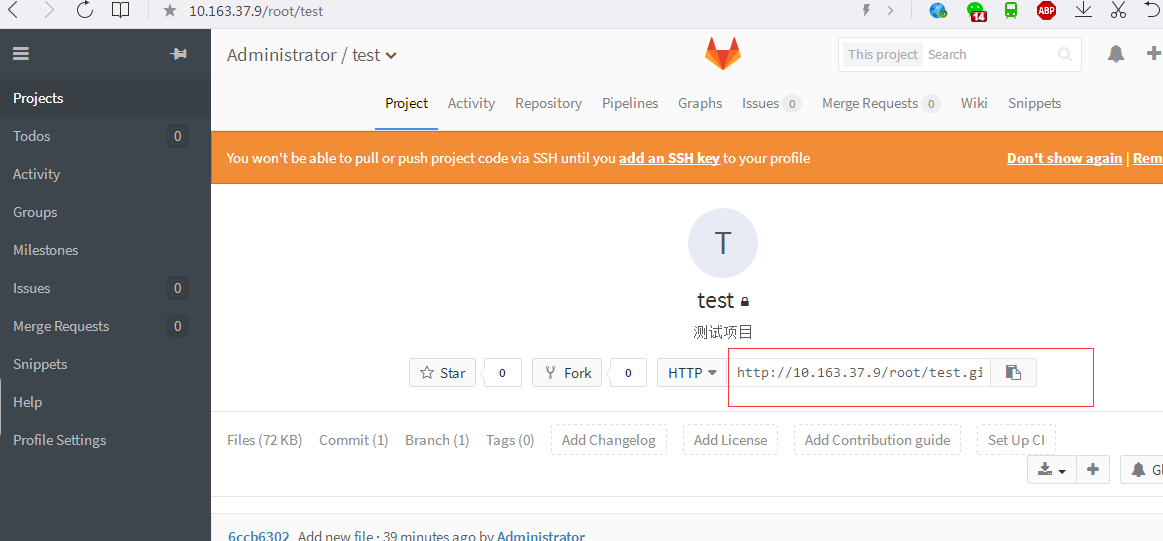


# 开发人员操作

首先登录到gitlab，进入到自己的项目管理界面，这里举例test项目。



复制项目url

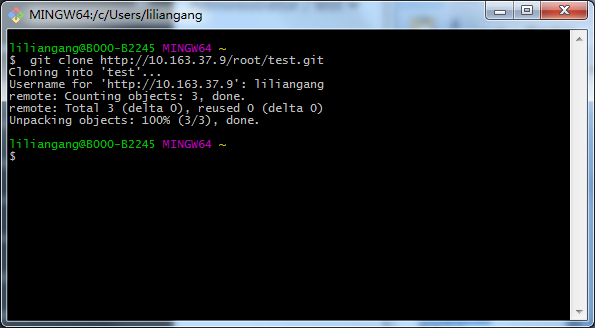


之后打开本地系统的git bash命令窗口界面

## 克隆（clone）项目

将远程库test项目克隆（clone）到本地库。 这一步只需要第一次操作即可。

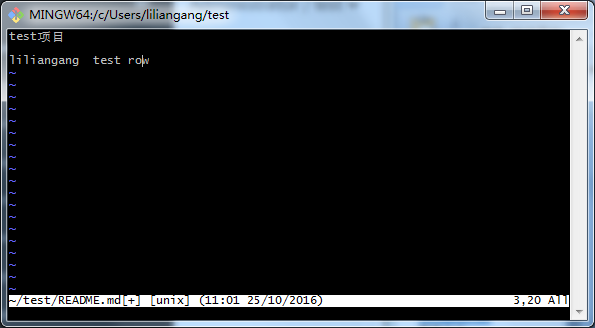
$ git clone [http://10.163.37.9/root/test.git](http://10.163.37.9/liliangang/test.git)



## 对项目做修改

对本地库test项目做修改，比如修改READ.md文件内容。编辑工具可以选择其他工具。

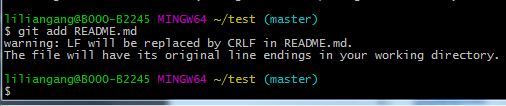




## 添加（add）修改到暂存区

添加修改到暂存区

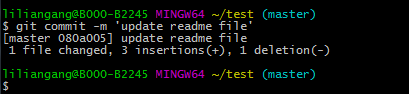
$ git add READ.md



## 提交（commit）修改到本地库

提交（commit）修改到本地库。

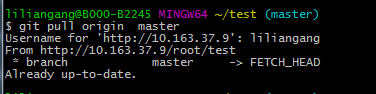
$ git commit -m '修改readme文件‘



## 拉取（pull）远程库到本地库

在push之前一定要先做pull操作，这样能保证在降低代码在服务器上冲突概率。拉取远程库到本地库

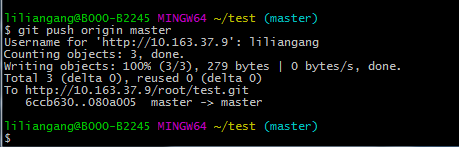
$ git pull origin master



## 推送（push）本地库修改到远程库

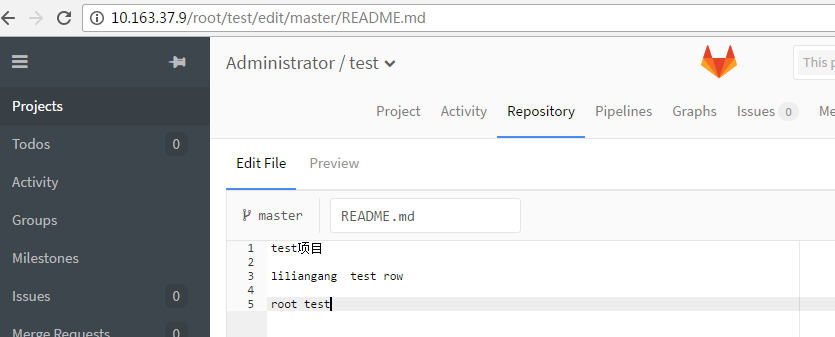
如果没有冲突，即可直接将修改推送（push）到远程库，否则需要先解决冲突。

$ git push origin master

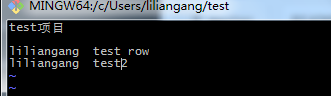


## 解决冲突

假如现在root在README.md文件中添加root test文字。



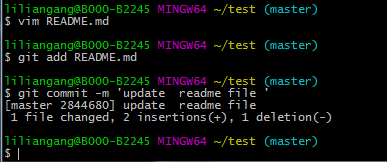
我也在README.md中添加liliangang test2文字。



并且在本地提交了。

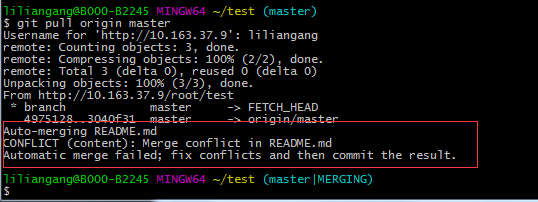
$ git add README.md

$ git commit -m 'update readme file '



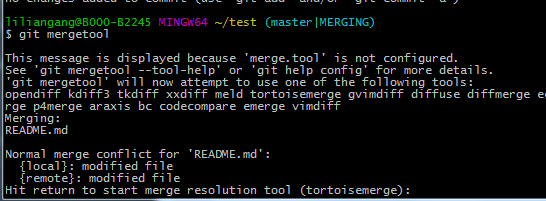
这时我去pull远程库的最新版本，将出现冲突，如下图。

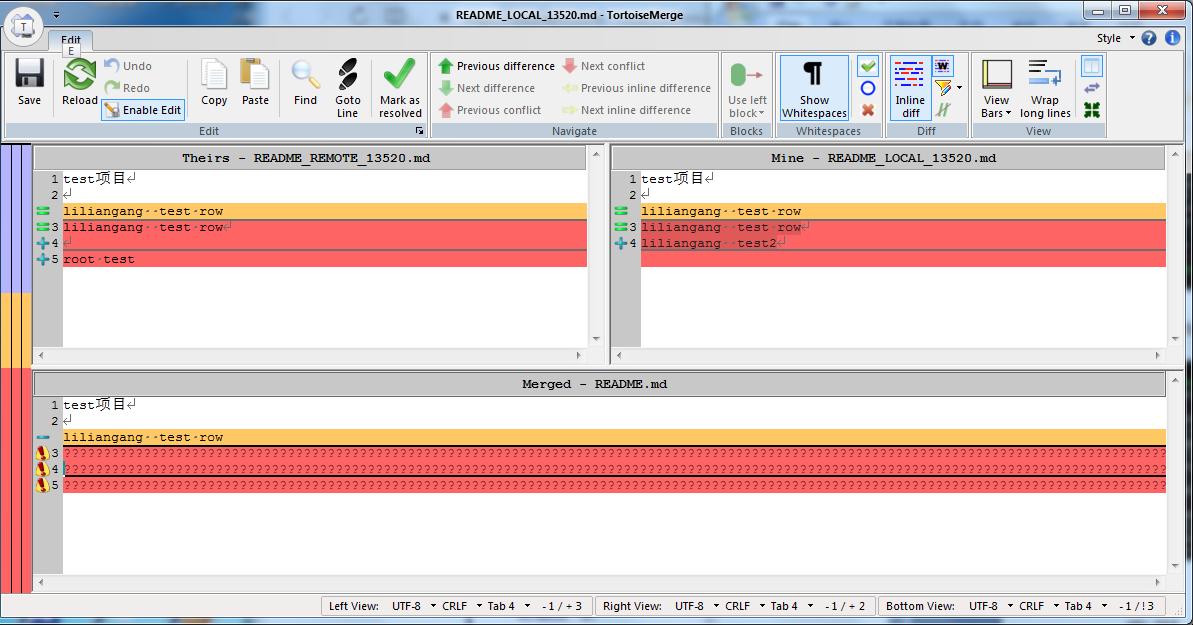
$ git pull origin master



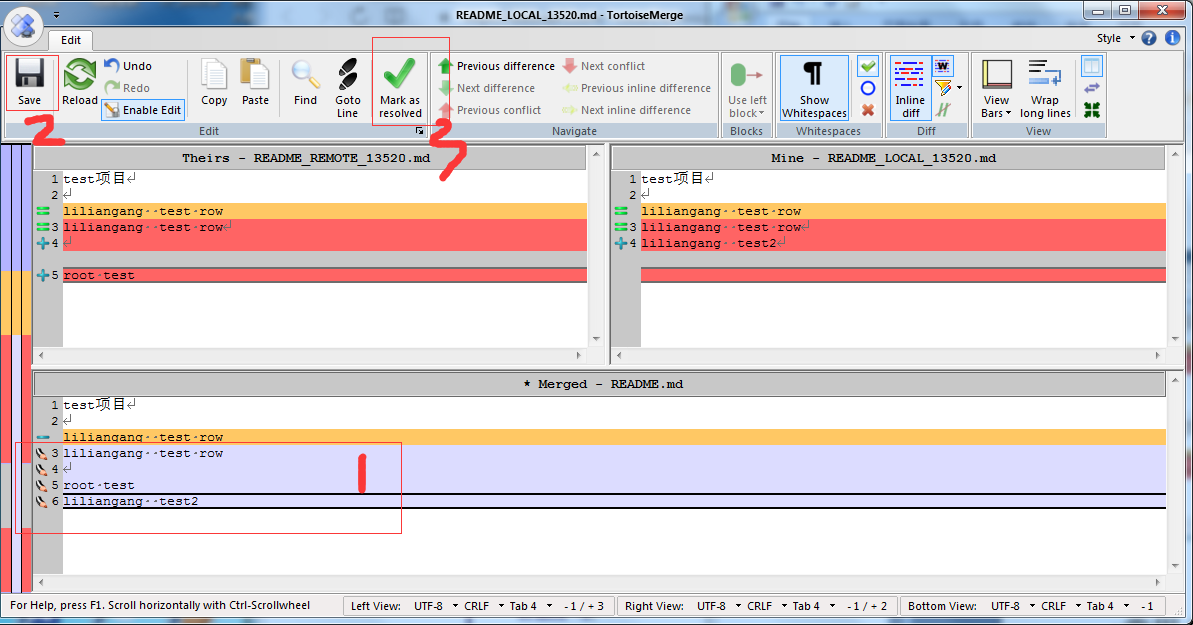
可通过mergetool可视化工具去解决冲突，如下图。

$ git mergetool





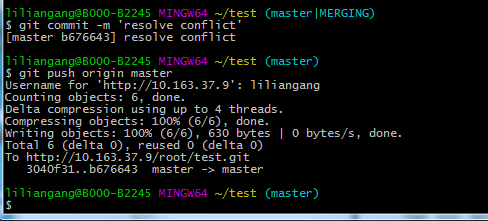
最后合并如下，并标记为已解决。



最后提交解决冲突，并推送到远程服务器。

$ git commit -m 'resolve conflict'

$ git push origin master



# 常用命令

* 初始化当前目录为git仓库

$ git init

* 克隆远程项目，如果不填本地路径，则克隆在当前目录下

$ git clone <http://url/xxx/xx.git> <本地路径>

* 查看当前分支状态（status）

$ git status

* 添加修改文件到暂存区（stage）

$ git add filename

* 提交到本地库

$ git commit –m ‘remark’

* 撤销提交（commit）

$ git reset HEAD^

* 撤销暂存区（stage）修改

$ git reset filename

* 拉取远程代码到本地，并合并

$ git pull origin master

* 抓取origin远程的master分支到本地库

$ git fetch origin master

* 将origin上的master分支合并到当前分支

$ git merge origin/master

* 推送master分支的修改到远程库

$ git push origin master

* 创建分支

$ git branch dev

* 查看所有分支情况

$ git branch –a

* 切换到dev分支

$ git checkout dev

* 设置远程分支简称为remote

$ git remote add root <http://xxx/xxx/xxx.git>

* 查看远程分支情况

$ git remote –v

* 查看提交日志

$ git log

* 查看提交日志，日志已被格式化

$ git log --pretty=oneline