Gerrit使用手册

|  |  |  |  |
| --- | --- | --- | --- |
| 文档编号 |  | | |
| 文档状态 | [√]草稿 [ ]正在修改 [] 正式发布 | | |
| 当前版本 | V1.0 | | |
| 密级 | 内部公开 | | |
| 拟制 |  | 日期 | 2014-08-25 |
| 审核 |  | 日期 |  |
| 批准 |  | 日期 |  |

目录

1 Gerrit审核流程 4

1.1 标准流程 4

1.2 试用流程 6

2 Gerrit访问数据库 6

2.1 gerrit服务端操作 6

2.2 SSH调用接口 6

3 用户管理 7

3.1 规则约定 7

3.2 添加用户 7

3.3 登录 7

3.4 配置邮箱和全名 8

3.5 配置SSH公钥 9

3.6 删除用户 10

3.7 其他说明 10

4 组管理 10

4.1 创建用户组 10

4.2 规则约定 11

4.3 用户组授权 12

5 项目管理 13

5.1 All-Projects项目 13

5.2 创建新项目 13

5.2.1 命令行创建新项目 13

5.2.2 WEB创建新项目 13

5.3 访问数据库 15

5.4 删除新项目 15

6 Gerrit在Eclipse中的使用 15

6.1 创建本地代码库 15

6.2 导入代码到工作空间 19

6.3 eclipse中配置name和email属性 20

6.4 代码提交 22

6.5 其它操作 27

6.5.1 回溯文件 27

6.5.2 git revert 和 git reset的区别 28

6.5.3 关于 git stash 29

6.6 冲突解决 29

6.7 常见问题 29

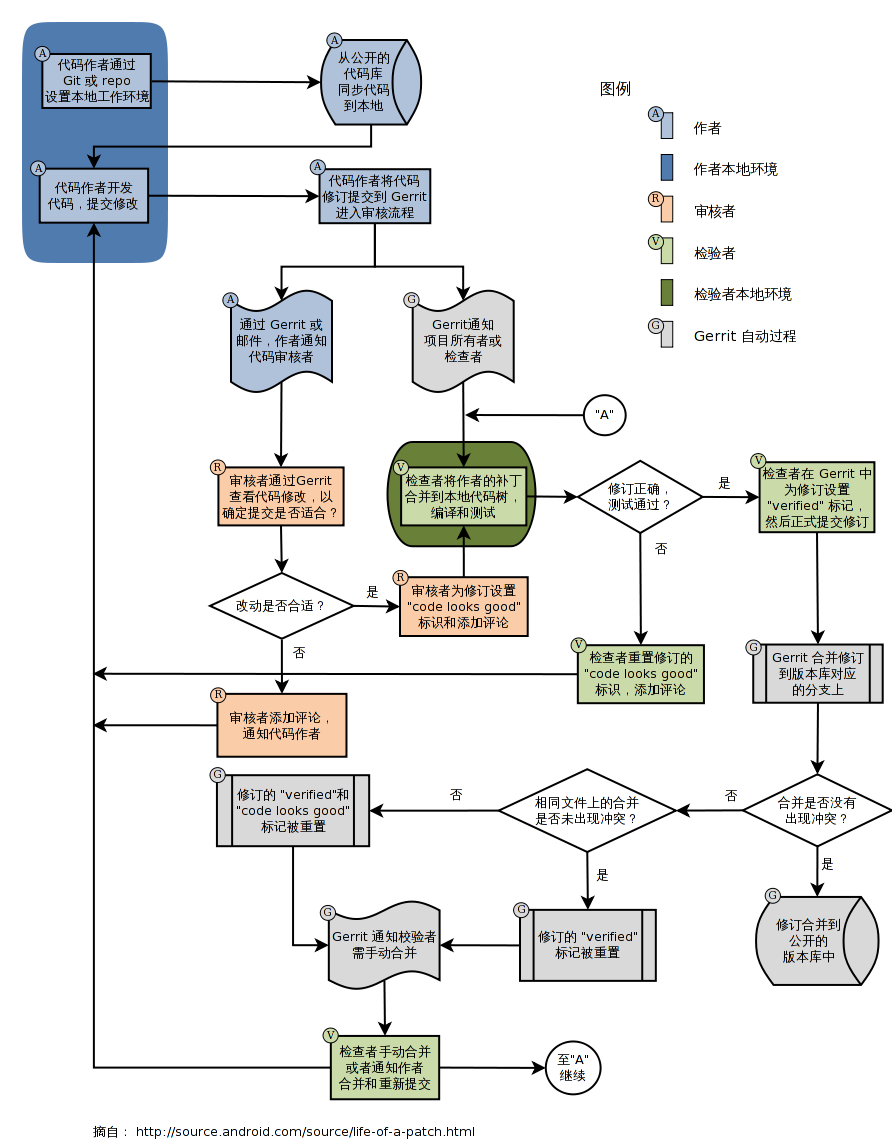
# Gerrit审核流程

## 标准流程

Gerrit 为 Git 引入的代码审核是强制性的，也就是说除非特别的授权设置，向 Git 版本库的推送必须要经过 Gerrit 服务器，修订必须经过代码审核的一套工作流之后，才可能经批准并纳入正式代码库中。

首先贡献者的代码通过 git 命令（或 repo 封装）推送到 Gerrit 管理下的 Git 版本库，推送的提交转化为一个一个的代码审核任务，审核任务可以通过 refs/changes/ 下的引用访问到。代码审核者可以通过 Web 界面查看审核任务、代码变更，通过 Web 界面做出通过代码审核或打回等决定。测试者也可以通过 refs/changes/ 之下的引用获取修订然后对其进行测试，如果测试通过就可以将该评审任务设置为校验通过（verified）。最后经过了审核和校验的修订可以通过Gerrit 界面中的提交动作合并到版本库对应的分支中。

Android 项目网站上有一个代码贡献流程图，详细地介绍了 Gerrit 代码审核服务器的工作流程。翻译后的工作流程图见下图。



## 试用流程

对比标准流程并结合公司实情，现阶段的代码评审有如下要求：

1. 评审流程参考标准流程，检验者的测试环节暂时省略，可直接合并版本；
2. 开发人员的代码审核者为项目经理，项目经理的代码审查者为Michael。
3. 检验者为Michael。

# Gerrit访问数据库

之所以要介绍对数据库的访问，是因为在web界面往往无法配置对gerrit的一些设置，需要直接修改数据库。Gerrit提供了两种方式来操作数据库，一种是在gerrit服务器端调用gerrit.war包中的命令入口，另外一种方法是远程SSH调用接口。

## gerrit服务端操作

对于第一种方法，需要再服务端执行，而且如果使用的是H2内置数据库还需要先将gerrit服务停止。总结起来操作步骤如下：

◆先以安装用户的身份进入Gerrit部署目录下；

◆停止gerrit服务；

◆执行命令调用gerrit.war包。

cd /app/gerrit

service gerrit stop

java -jar bin/gerrit.war gsql

当出现“gerrit>”提示符时，就可以输入 SQL 语句操作数据库了。

此种方式访问数据库已验证。

## SSH调用接口

当在 Gerrit 上注册了第一个帐号时，即拥有了管理员帐号，正确为该帐号配置公钥之后，就可以访问 Gerrit 提供的 SSH 登录服务。Gerrit 的 SSH 协议提供访问数据库的第二种方法。下面的命令就是用管理员公钥登录 Gerrit 的 SSH 服务器，操作数据库。虽然演示用的是本机地址（localhost），但是操作远程服务器也是可以的，只要拥有管理员权限。

$ ssh -p 29418 localhost gerrit gsql

运行命令 gerrit gsql 连接 Gerrit 的 SSH 服务。当连接上数据库管理接口后，便出现“gerrit>”提示符，在该提示符下可以输入 SQL 命令。在All-projects里面**Global Capabilities里面给管理员设置access database权限 。此种方式也验证成功。**

# 用户管理

## 规则约定

Gerrit用户名：与OA用户名一致。

密码：123456。

## 添加用户

使用如下命令为用户生成http认证配置文件。

htpasswd -b /app/gerrit/etc/htpasswd.conf michael.zhao 123456

将上述命令按行加入脚本文件，即可实现批量添加用户的目的。

vi addgerrituser.sh

htpasswd -b /app/gerrit/etc/htpasswd.conf jacky.xiao 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf owen.liu 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf gavin.zhang 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf nety.huang 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf james.fu 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf carl.yu 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf randy.cheng 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf eric.li 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf max.liu 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf danie.zhou 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf judy.xia 123456

htpasswd -b /app/gerrit/etc/htpasswd.conf sunlight.fu 123456

## 登录

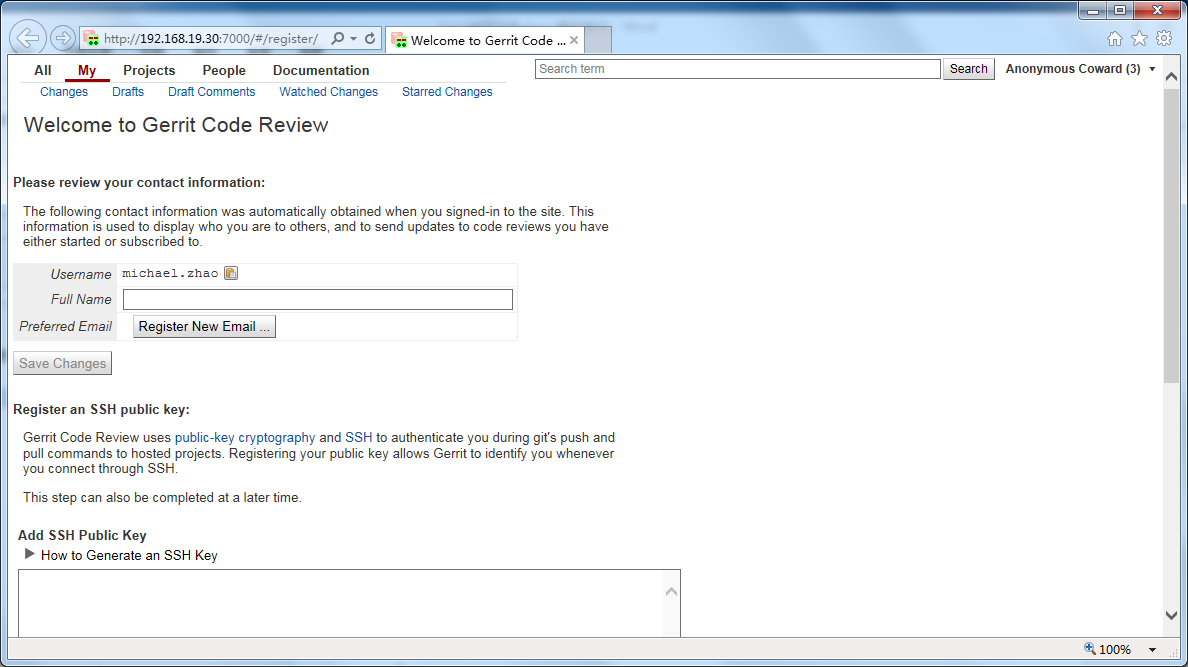
浏览器访问<http://192.168.19.30:7000/>，弹出如下对话框。



输入用户名密码，勾选“记住我的凭据”，点击“确认”登录。成功登录后gerrit会自动按此用户名生成用户。

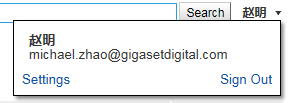
## 配置邮箱和全名

成功登录的界面如下，配置好全名，并注册邮箱。



注册邮箱时会发验证邮件，点击邮件里的链接或复制链接到浏览器以激活邮箱。

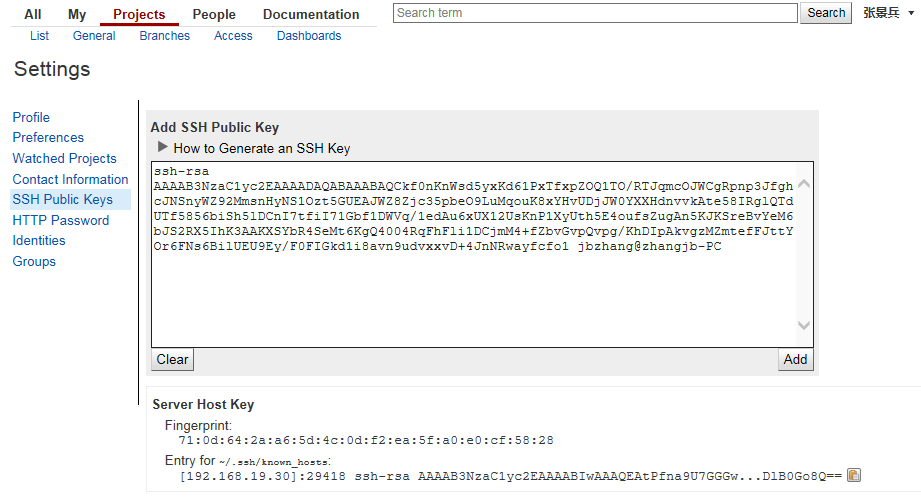
也可通过点击右上角的姓名--Settings—Contact Infomation进入邮箱、全名配置界面，如下图。





## 配置SSH公钥

通过点击右上角的姓名--Settings—SSH Public Keys进入SSH公钥配置界面，如下图。



复制个人电脑id\_rsa.pub文件里的内容，粘贴到文本框内，点击“Add”按钮即可。一个用户可配置多个SSH公钥。

## 删除用户

使用如下命令删除用户。

htpasswd -D /app/gerrit/etc/htpasswd.conf michael.zhao

注：如要修改用户密码，先删除用户，再使用新密码重新添加用户。

## 其他说明

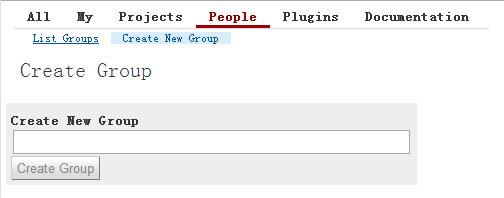
由于采用的是HTTP反向代理认证，所以不支持Sign Out（退出登录），如有切换用户的需求，请先关闭浏览器再打开即可重新登录认证。

# 组管理

## 创建用户组

有两种方法可以实现对评审最终确认的授权，一种是赋予特定用户 Verified 类别中的“+1:Verified”的授权，另外一个方法是赋予特定用户 Code Review 类别中更高级别的授权：“+2:Looks good to me, approved”。要想实现对经过确认的评审任务的提交，还需要赋予特定用户Submit 类别中的“+1: Submit”授权。为此创建两个新的用户组 Reviewer 和 Verifier，并为其赋予相应的授权。

可以通过 Web 界面或命令行创建用户组。如果通过 Web 界面添加用户组，选择“People”菜单下的“Create New Group”子菜单，如下图。



输入用户组名称后，点击“Create Group”按钮，进入创建用户组后的设置页。



其中General选项卡里可以修改用户组的描述和Owners。Members选项卡可为用户组分配用户，也可指定包含其他的用户组。

## 规则约定

现有云服务组团队成员的用户组明细表如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **职能组** | **姓名** | **Registered User** | **Reviewer** | **Verifier** |
| 1 | 云服务组 | michael.zhao |  |  |  |
| 2 | jacky.xiao |  |  |  |
| 3 | owen.liu |  |  |  |
| 4 | gavin.zhang |  |  |  |
| 5 | nety.huang |  |  |  |
| 6 | bing.zhang |  |  |  |
| 7 | james.fu |  |  |  |
| 8 | carl.yu |  |  |  |
| 9 | randy.cheng |  |  |  |
| 10 | eric.li |  |  |  |
| 11 | max.liu |  |  |  |
|  |  |  |  |  |  |

**Registered User：**所有登录用户均属于该组。该组成员可下载源码，提交变更后的代码，并对变更发表意见，但不具备接受或拒绝一个变更的权限。Registered User对于所有有Read权限的项目的任何变更都有发表评论的权力。

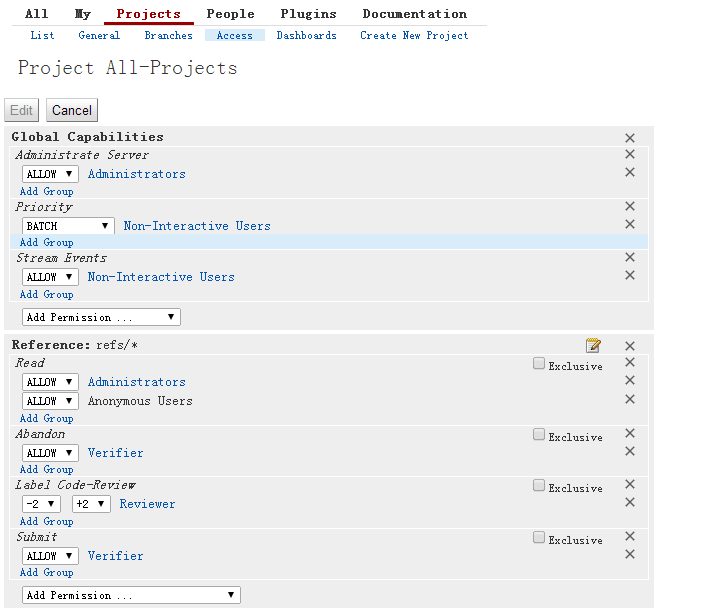
**Reviewer：**该组成员执行代码评审，可接受或拒绝一个变更。

**Verifier：**该组成员可提交合并代码。

注：用户属于多个组，则享受最大化权利的原则。

## 用户组授权

为新的用户组授权，需要访问“Projects”菜单下的“List”子菜单，点击对应的项目进入权限编辑界面。为了简便起见，选择“All Projects”，对其授权的更改可以被其他的所有项目共享。下图是为用户组建立授权过程的页面。



分别为两个新建立的用户组（Reviewer和Verifier）分配授权，如下表。这样，就为 Gerrit 所有的项目设定了可用的评审工作流。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 引用名称 | 类别 | 用户组名称 | 权限范围 |
| 1 | refs/\* | Label Code-Review | Reviewer | -2 --- +2 |
| 2 | refs/\* | Submit | Verifier | ALLOW |

# 项目管理

## All-Projects项目

该项目为Gerrit创建，默认为是所有后建项目的父项目，根据子项目继承父项目权限的规则，All-project的权限被后期建立的所有项目继承。Administrators组内成员可以修改这个项目的权限控制。这个项目的所有权不能被授予给其他组，否则被授权组将近乎拥有和Administrators组内成员相同的访问权限。

## 创建新项目

一个 Gerrit 项目对应于一个同名的 Git 库，同时拥有一套可定制的评审流程。创建一个新的Gerrit 项目就会在对应的版本库根目录下创建 Git 库。管理员可以通过命令行或 Web 界面创建新项目。

### 命令行创建新项目

管理员可以使用命令行创建新项目。

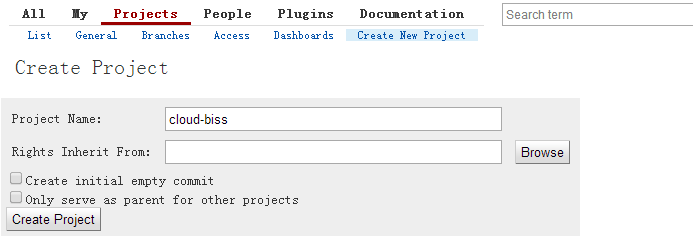
$ ssh -p 29418 localhost gerrit create-project --name cloud-biss

执行 gerrit ls-projects 命令可以看到新项目已经成功创建。

$ ssh -p 29418 localhost gerrit ls-projects

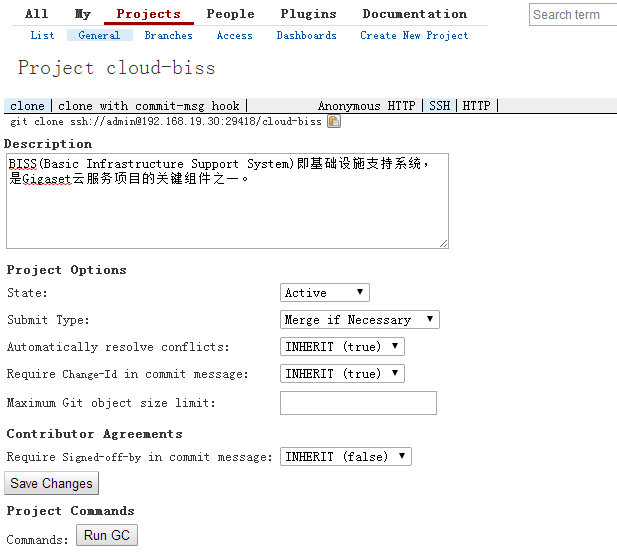
### WEB创建新项目

通过 Web 界面创建新项目，选择“Projects”菜单下的“Create New Project”子菜单，如下图。



需要勾选create initial empty commit

输入项目名，点击“Create Project”按钮，进入创建后的设置页。



可以设置项目描述及项目选项，点击“Save Changes”按钮保存修改后的设置。

“Description”上部有一串字符，它是该项目Git仓库的URI地址，直接复制、粘贴到Eclipse中即可下载源码，有匿名HTTP、SSH和HTTP协议供选择。

git clone http://192.168.19.30:7000/cloud-biss

git clone ssh://admin@192.168.19.30:29418/cloud-biss

git clone <http://admin@192.168.19.30:7000/cloud-biss>

## 访问数据库

在服务器端执行，先将 Gerrit服务停止。以安装用户的身份进入 Gerrit 部署目录下，再执行命令调用 gerrit.war 包，如下：

cd /app/gerrit

sh bin/gerrit.sh stop

java -jar bin/gerrit.war gsql

当出现“gerrit>”提示符时，就可以输入 SQL 语句操作数据库了。

## 删除新项目

进入Gerrit目录的git仓库目录，删除项目名.git。

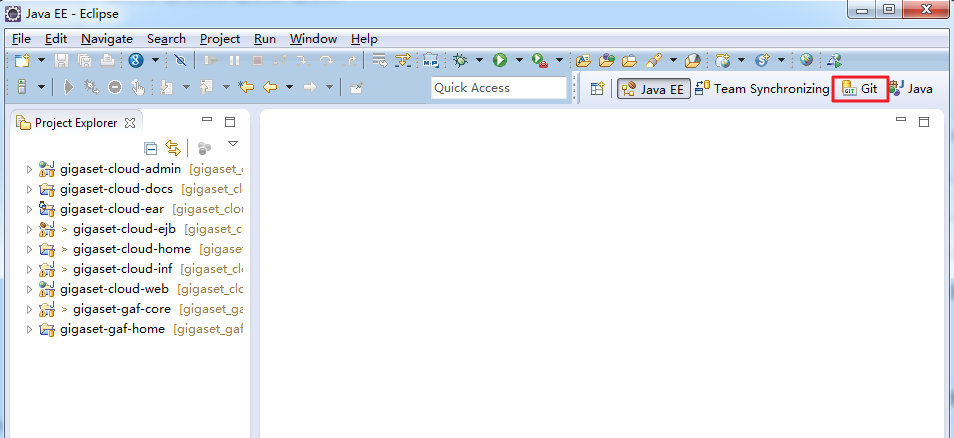
cd /app/gerrit/git

rm -rf cloud-biss.git

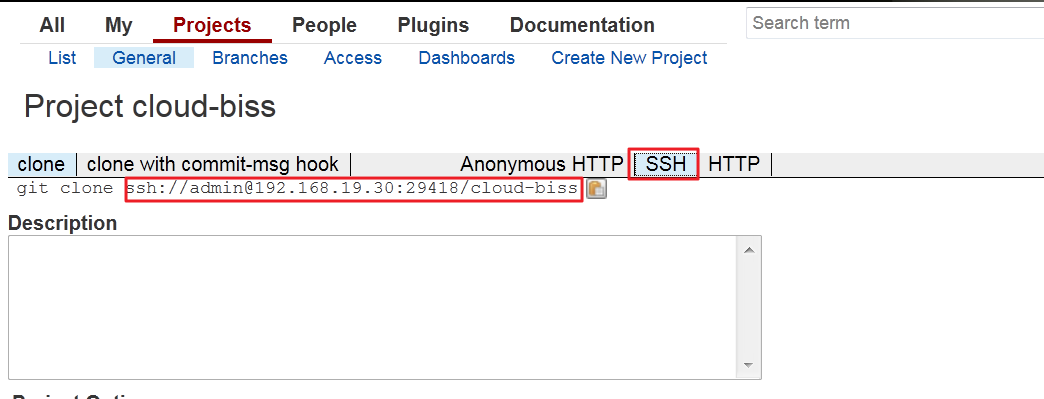
# Gerrit在Eclipse中的使用

## 创建本地代码库

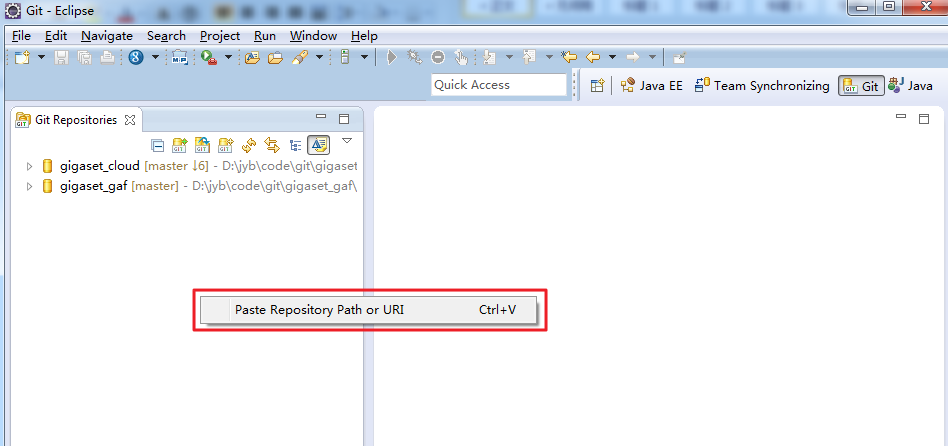
打开git仓库管理视图



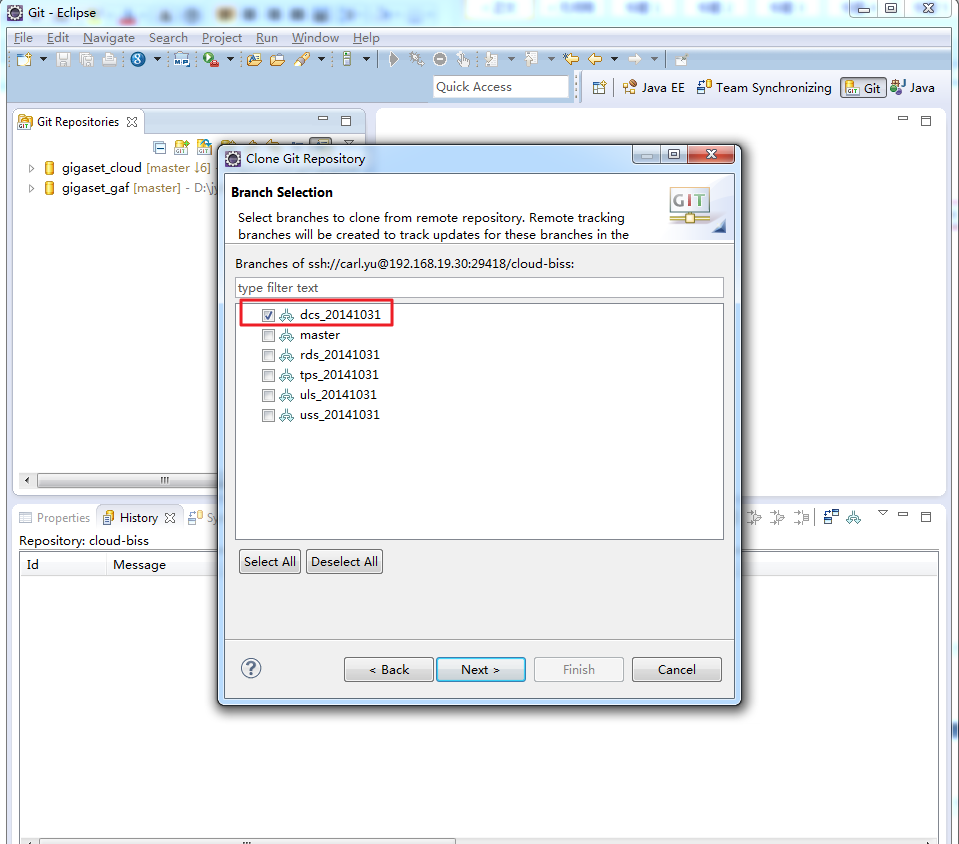
登陆到gerrit web端，URL为：http://192.168.19.30:7000 。复制SSH地址



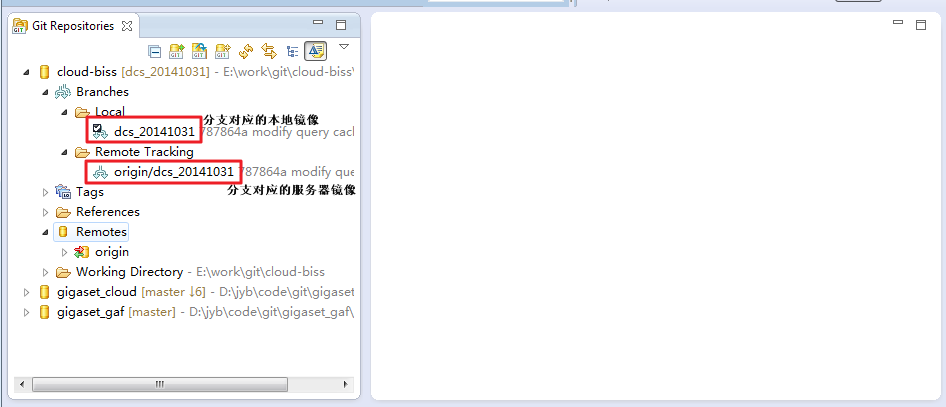
在git仓库管理视图右键单击，粘贴ssh地址



下一步，在弹出的分支选择窗口选择需要下载的分支



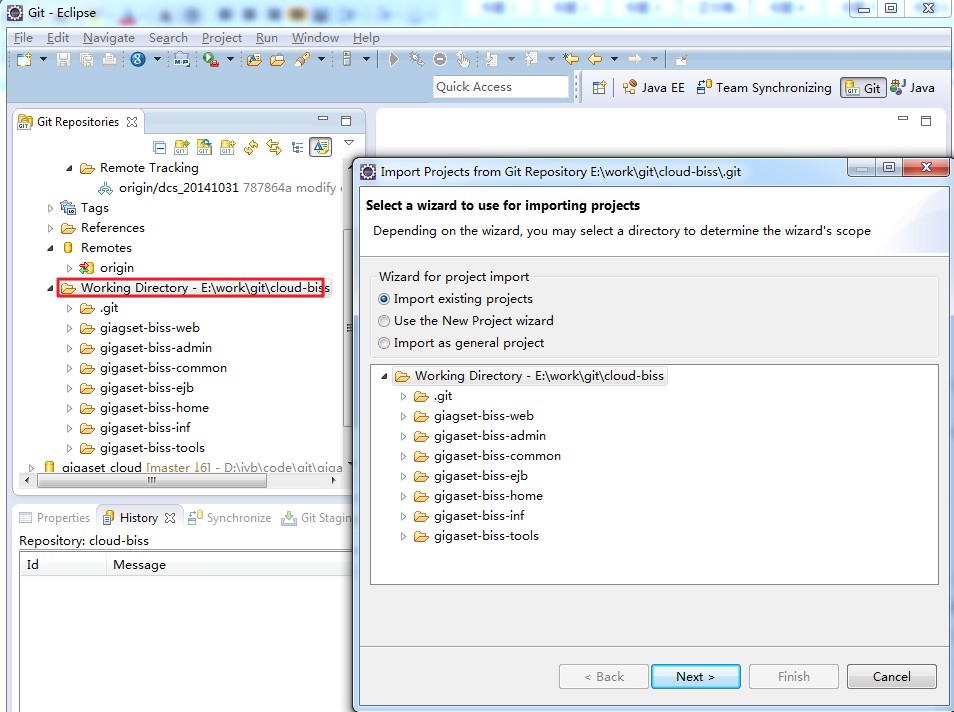
分支目录说明



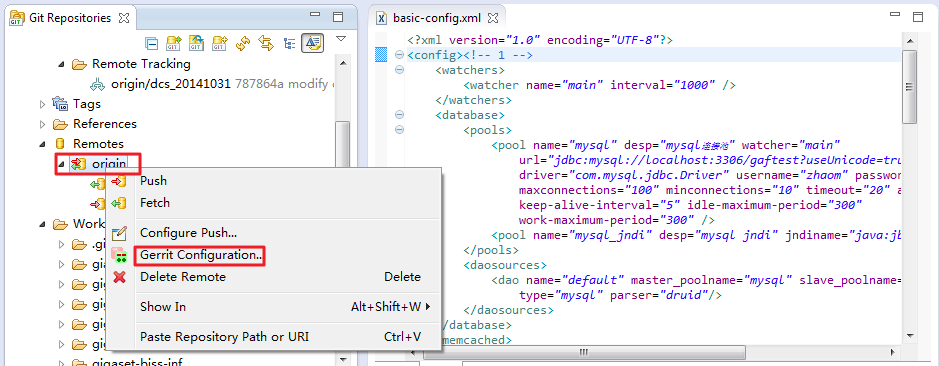
注：双击可切换eclipse工作空间对应的分支

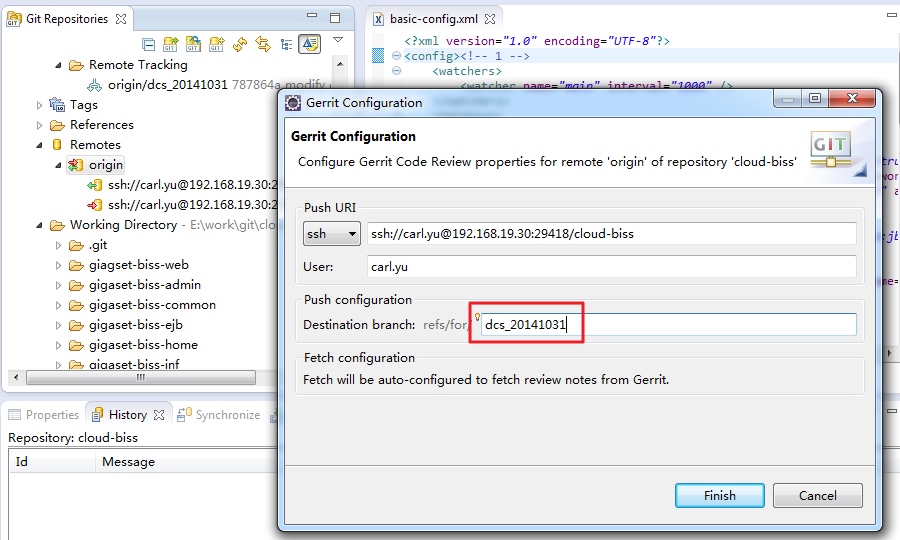
## 导入代码到工作空间

在Working Directory右键导入工程到eclipse



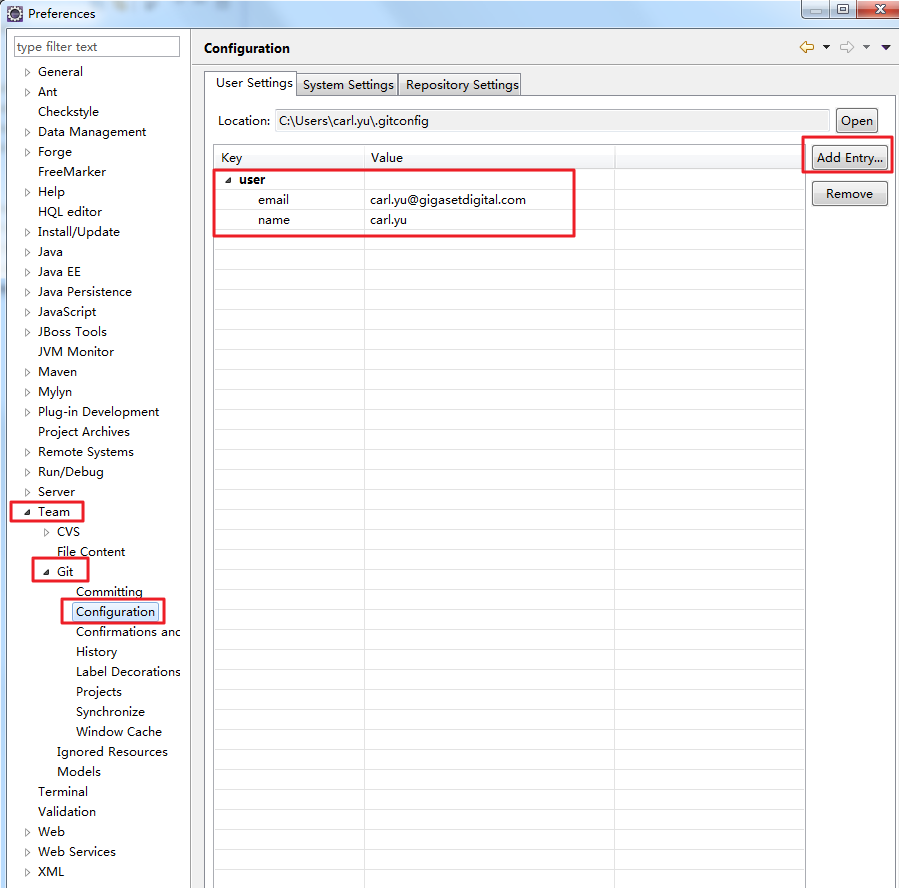
在origin右键点击，配置代码的提交分支



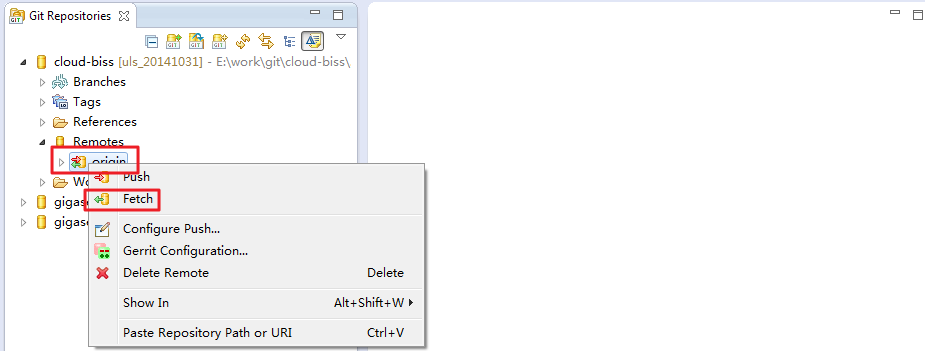


## eclipse中配置name和email属性

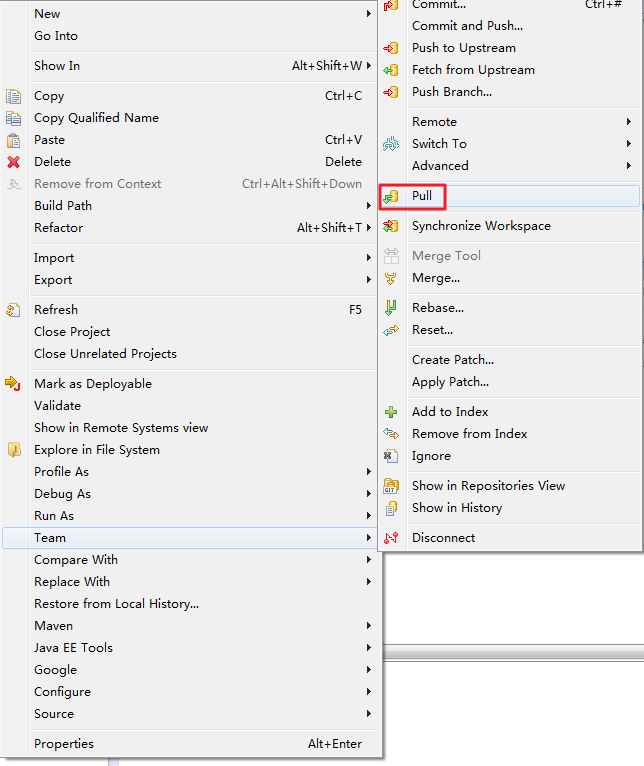
打开Windows-->Preference，找到git属性配置窗口，如图：



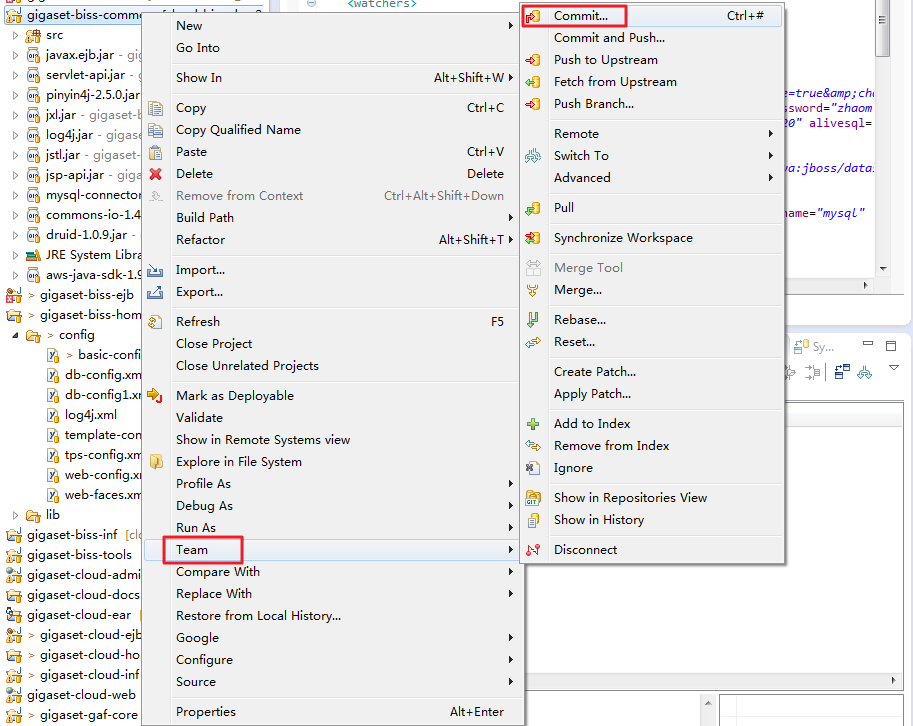
## 代码提交



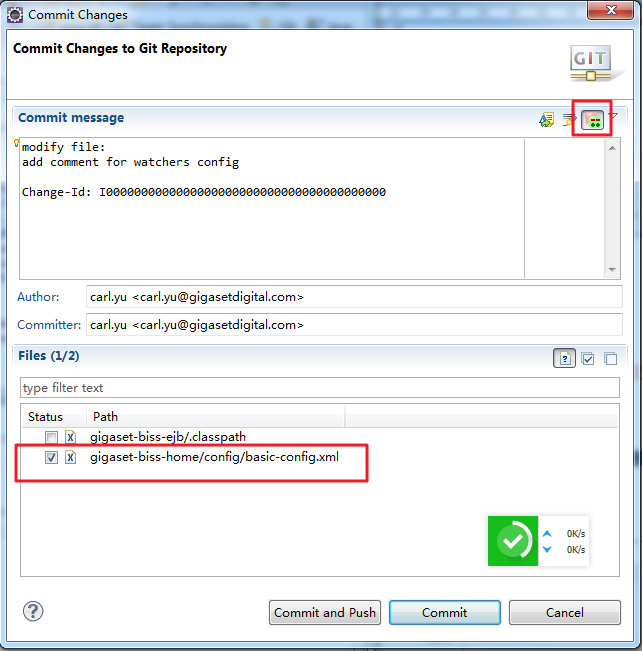
代码提交前需要先下载远程服务器代码，进行更新合并



代码写完后，右键单击工程，提交代码



写提交注释



提交注释规范：

1. 注释内容尽量用英文写。
2. 提交代码注释内容分为如下几类

**上传初始工程**

init project:project name

**新增文件**

add file:filename

description the file

**新增功能**

add function:function name

description the function

**修改功能代码**

modify function:function name

description why do you modify the functon

**修复bug**

fix bug:bug name

describe what's wrong with the bug

说明：

◆ 首行写注释类别及对应名称。

◆ 空一行之后进行相应描述。

◆ 如果代码是针对某个项目的改动，要以 项目名称+:+空格 开头，例如 D508: Add software navigator bar。

例：

modify: S100D: Cofigure GPIO31 and GPIO33 to INPUT PULL UP.

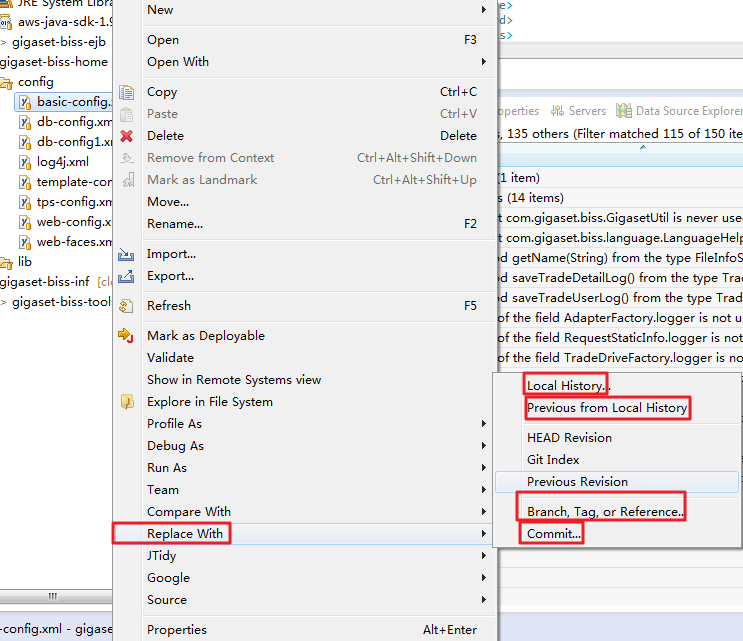
OEMSBL would check VOL+ and VOL- to enter download mode,

becase we only checked the low status, so we need set these two

GPIO to input pull up, otherwise, we can't get right status.

## 其它操作

### 回溯文件



### git revert 和 git reset的区别

1. git revert是用一次新的commit来回滚之前的commit，git reset是直接删除指定的commit。

2. 在回滚这一操作上看，效果差不多。但是在日后继续merge以前的老版本时有区别。因为git revert是用一次逆向的commit“中和”之前的提交，因此日后合并老的branch时，导致这部分改变不会再次出现，但是git reset是之间把某些commit在某个branch上删除，因而和老的branch再次merge时，这些被回滚的commit应该还会被引入。

3. git reset 是把HEAD向后移动了一下，而git revert是HEAD继续前进，只是新的commit的内容和要revert的内容正好相反，能够抵消要被revert的内容。

* **git reset [--hard|soft|mixed|merge|keep] [<commit>或HEAD]**：将当前的分支重设（reset）到指定的<commit>或者HEAD（默认，如果不显示指定commit，默认是HEAD，即最新的一次提交），并且根据[mode]有可能更新index和working directory。mode的取值可以是hard、soft、mixed、merged、keep。下面来详细说明每种模式的意义和效果。

**1、hard(慎用)**

重设index和working tree,所有改变都会被丢弃，包括文件的修改、新增、删除等操作，并把HEAD指向<commit>，

因此通过git log查看版本提交记录，被reset的版本记录会被丢弃，但可以通过git reflog查看

**2、soft**

不重设index和working tree,仅仅将HEAD指向<commit>,表示已经commit的文件会取消commit,

通过git status查看，文件会处于待commit状态“Changes to be committed”

**3、mixed(默认)**

重设index,但不重设working tree,表示已经被add的文件，被取消add，

通过git status查看，文件会处于待添加索引状态 “Changes not staged for commit”

**4、merge**

重设index，重设working tree中发生变化的文件，但是保留index和working tree不一致的文件

**5、keep**

重设index，重设working tree中发生变化的文件

### 关于 git stash

git stash命令简而言之就是帮助开发人员暂时搁置当前已做的改动，倒退到改动前的状态，进行其他的必要操作（比如发布，或者解决一个bug，或者branch，等等），之后还可以重新载入之前搁置的改动

需要找回之前搁置的改动继续先前的工作了？

git stash apply 即可。

也可以用 git stash list 来查看所有的搁置版本（可能搁置了很多次，最好不要这样，容易搞混）

在出现一个搁置栈的情况下，比如如果你想找回栈中的第2个，可以用 git stash apply stash@{1}

如果想找回第1个，可以用 git stash pop

如果想删除一个stash，git stash drop <id>

删除所有stash，git stash clear

## 冲突解决

## 常见问题

remote: ERROR: missing Change-Id in commit message footer）

1. 把remote的hooks/commit-msg 的文件拷贝到本地的git目录的.git/hooks/ 文件夹

例如：

scp -p -P 29418 randy.cheng@192.168.19.30:hooks/commit-msg /f/cloud-biss/.git/hooks/

另外，通过show history查看历史记录是否存在没有写提交注释的commit,将提交恢复到上一次，加上注释，再一次提交。如果有没有change-id的提交，也需要恢复到相应的提交，然后加上chang-id，再次提交。

2.用命行的方式把代码提交到gerrit 进行review

gerrit不允许直接将本地修改同步到远程仓库。客户机必须先push到远程仓库的refs/for/\*分支上 (\*代表的是分支名)

例如，我要把dcs\_20141031提交到gerrit进行代码审核，可以这样执行命令

$ git push ssh://randy.cheng@192.168.19.30:29418/cloud-biss HEAD:refs/for/dcs\_20141031

其它参考文档：http://blog.sina.com.cn/s/blog\_4b5039210102e5o5.html