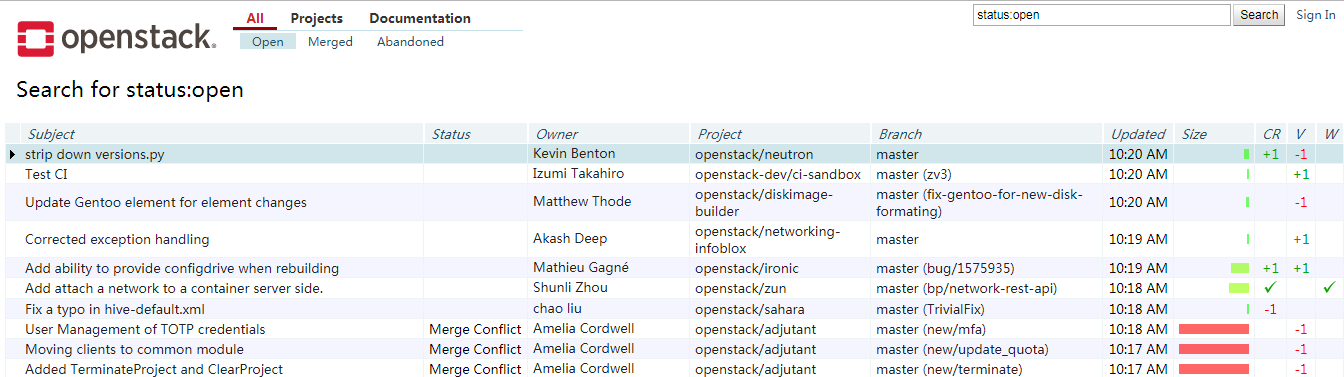
# 自动化测试使用说明

智能产品研究部 Robert/罗水波

# 测试背景

* 为了提高开发效率和测试效率，我们使用git+Jenkins+gerrit的工具组合来进行控制。这些开源工具都是世界上使用非常广泛的工具，在许多知名的顶尖企业中都有很好的应用。

如下图是openstack公司在使用gerrit 进行代码review。https://review.openstack.org/#/q/status:open



* 本文档这里是简要说明主要的内容，还有许多有用的知识和操作需要在不断使用和总结.

主要有：测试的主要原理， developer用户操作步骤，reviewer的用户操作步骤,管理员的操作步骤.另外有一个文档管理系统smaba的使用，也有提到。

* 为什么代码入库前一定要code review?

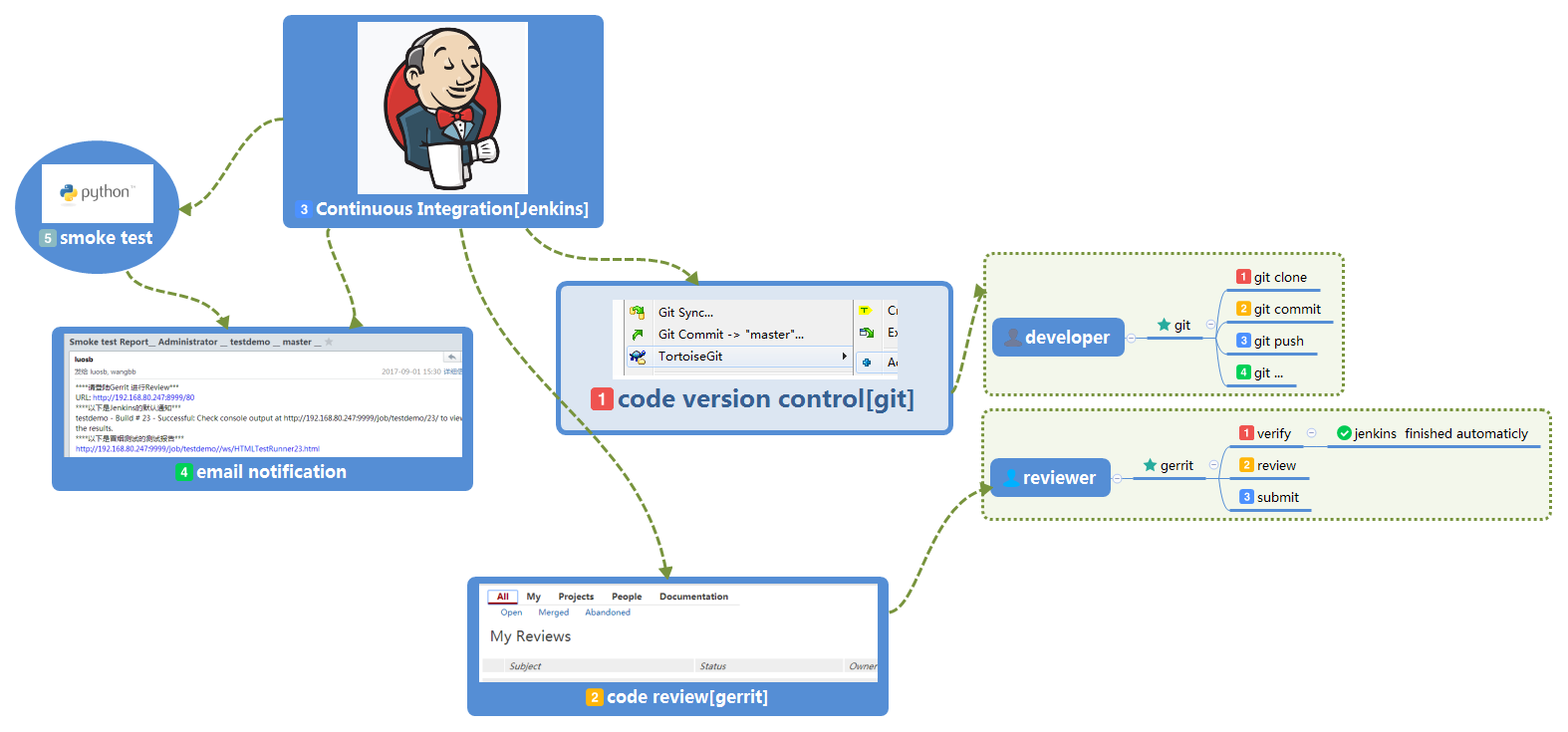
首先，代码审查的最大的功用是纯社会性的。如果你在编程，而且知道将会有同事检查你的代码，你编程态度就完全不一样了。你写出的代码将更加整洁，有更好的注释和程序结构。

其次，偷懒是人的天性，从节约成本的角度考虑，大家一般会选择在测试之前无限制的Delay.

* 还有硬件自动化测试的背景内容后续增加。

# 测试原理

这里是软件自动化测试的原理图，还有硬件的后续增加。



1. 版本管理工具git ,tortoisegit。
2. 代码review工具 gerrit。
3. 持续集成工具jenkins。
4. 邮件发送通知，邮件中有需要进行review的链接.
5. 冒烟测试python脚本 .

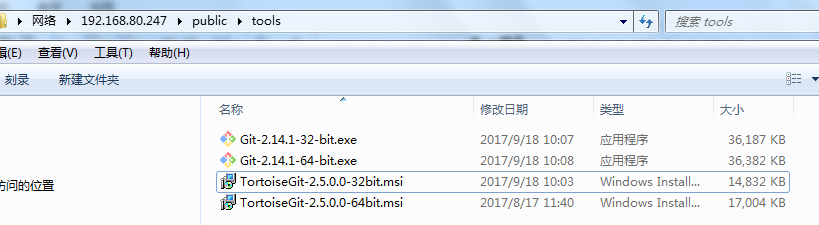
# 用户操作步骤(开发者)

* **以开发者为例**

1. 安装git 和 tortoiseGit
2. 下载方式 file://192.168.80.247/public/tools/

找到对应的系统版本位数,如32位操作系统

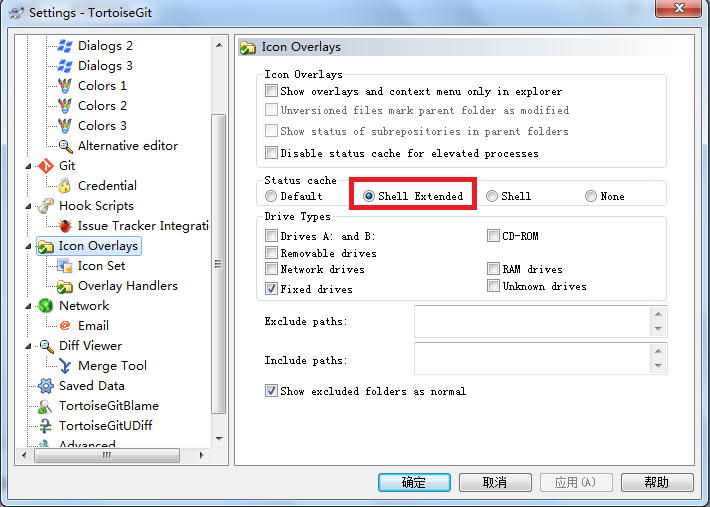
* Git-2.14.1-32-bit.exe
* TortoiseGit-2.5.0.0-32bit.msi



1. 安装git，默认安装，都是直接点下一步，这里不多说。
2. 安装tortoiseGit，默认安装，都是直接点下一步，这里不多说。
3. 最后tortoiseGit要设置两个地方。

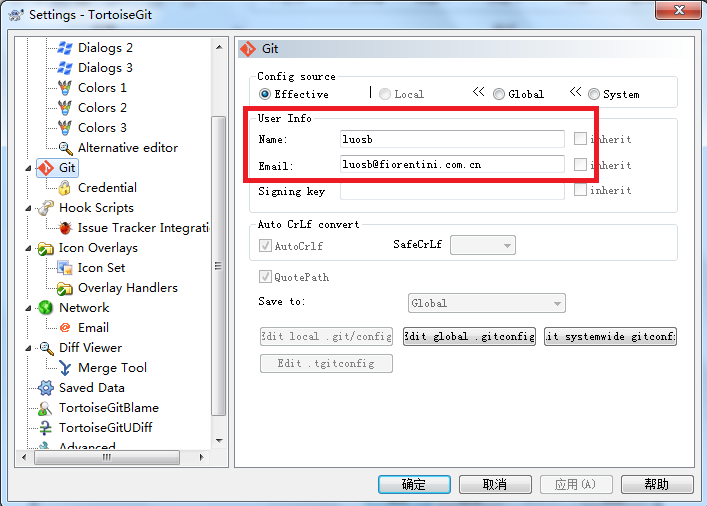
在系统开始菜单，找到TortoiseGit,选择Settings.

* 修改图标icon的状态为 shell Extended，以防在系统中出现文件夹闪烁的情况.

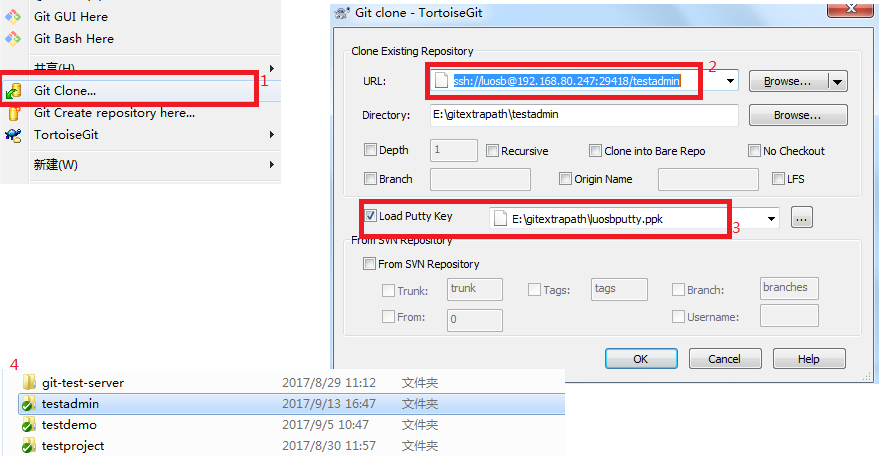


* 确定git使用的user info 是正确的。如果打开是空的，先跳过此步骤。

如下图 name 和Email 填写的是我本人的信息.



1. 复制远程仓库到本地,参考下图



准备工作

在本地D盘目录中，新建一个文件夹test，并且进入该test文件夹。再参考上图。

1. 使用 tortoiseGit 的git clone

如图示数字1部分

1. 输入远程仓库地址

如图示数字2部分

ssh://luosb@192.168.80.247:29418/testadmin

其中luosb是帐户名，需要改成自己的邮箱名字的前缀。[如wangbb@fiorentini.com.cn](mailto:如wangbb@fiorentini.com.cn)，就是wangbb。

testadmin是项目名称,也需要改成对应的项目.如硬件的工程名称是gsmp\_hw



1. 加载密钥 (密钥的生成步骤可以参考系统管理员的操作步骤)

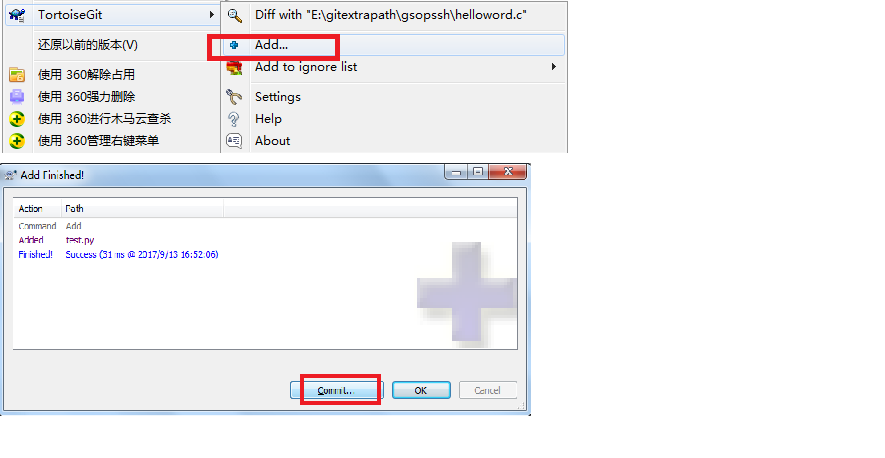
如图示3部分.

4 克隆成功，生成本地仓库目录

如图示4部分

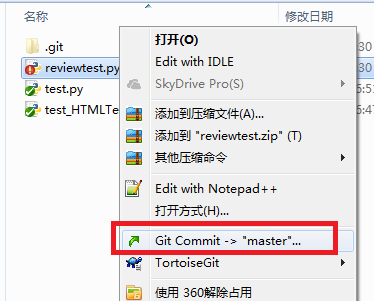
1. 提交到本地仓库
2. 新建的文件提交，需要新增再提交到本地仓库 。

* 进入本地仓库目录下,手动新建一个文件test.py
* 光标选中新建的文件test.py，然后鼠标右键出现tortoiseGit的菜单，选择Add
* 文件选择后进行OK确认
* 最后点commit进行提交



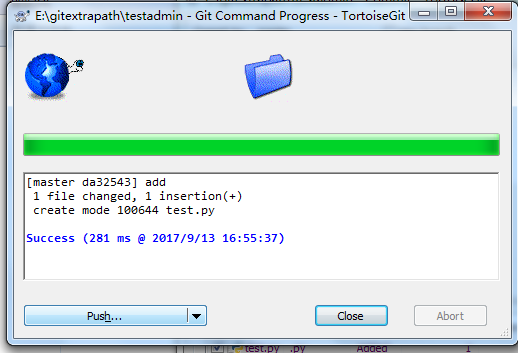
其中commit的过程中要增加的注释十分重要，务必要写的简明干练.

2 还有另一种情况，修改已有的文件进行提交。这个时候可以右键选择git commit。

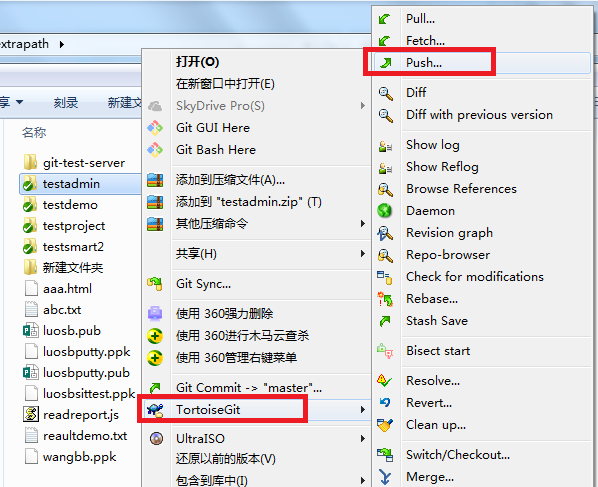


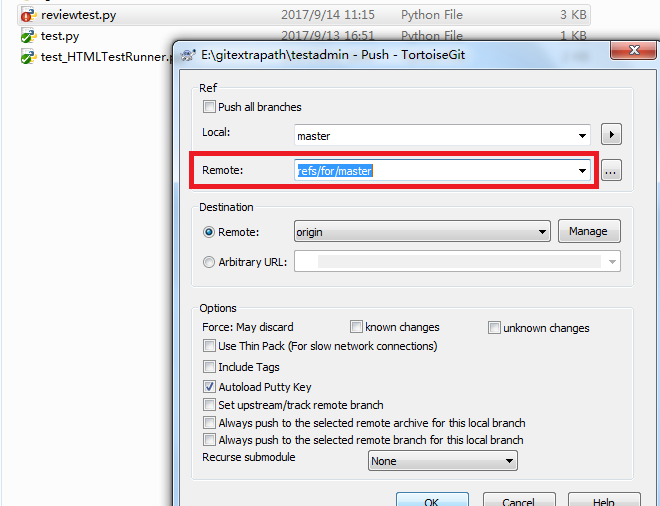
1. 提交到远程仓库

1 在提交到本地仓库之后 ，直接可以点push，就推送到远程仓库了。



2 另一种方法,选择从目录上进行push.

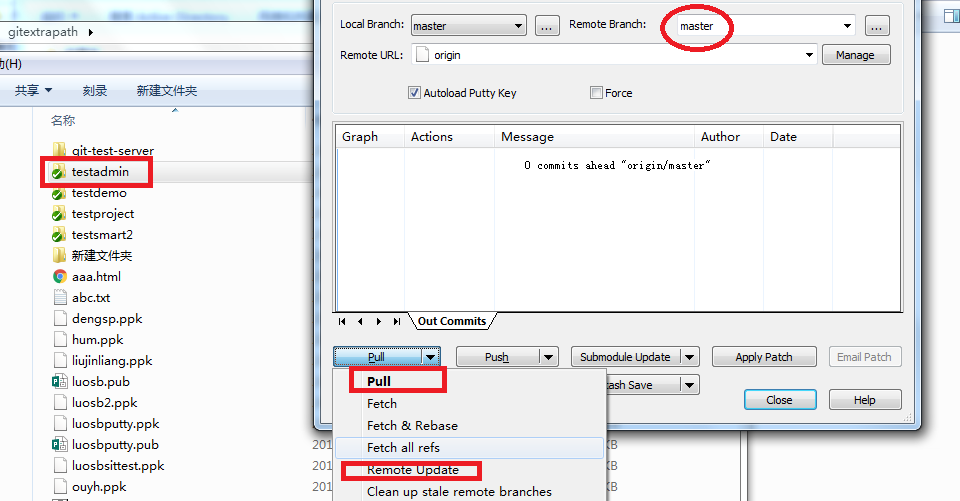




注意：remote 仓库的地址要填写正确。比如refs/for/master

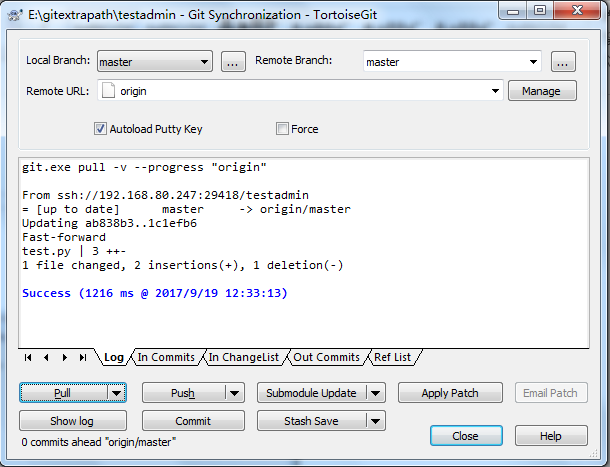
1. 更新本地仓库代码
2. 选择本地仓库的文件夹，鼠标右键可以看到有个[git sysn…]的选项。这里要先进进行remote update，再进行pull .

其中remote branch要选择正确的分支名称，或者手动输入master 分支。

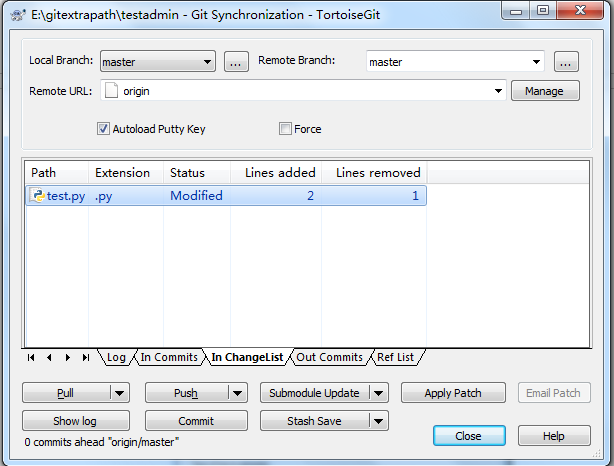


1. 点击Pull后可以获取最新的change。如下图，log 记录中有显示

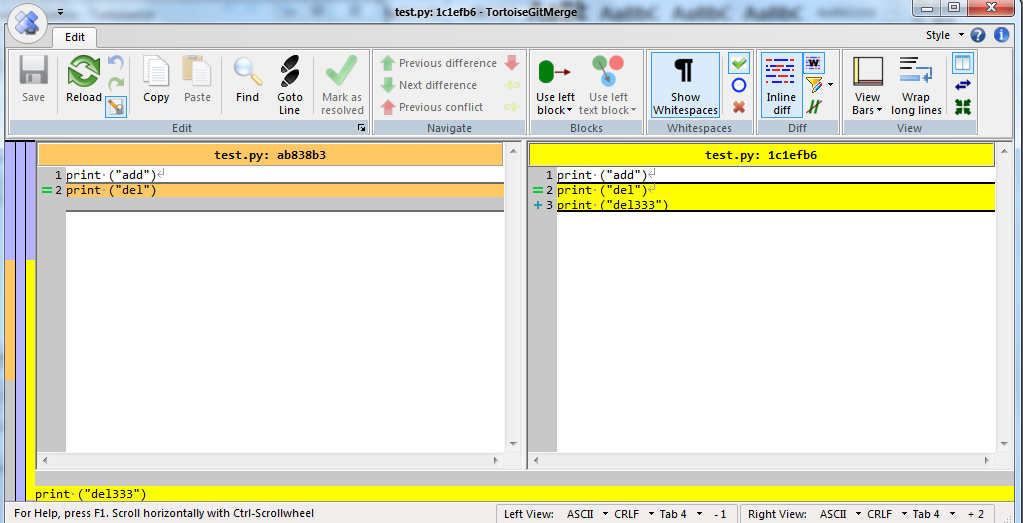
“ 1 file changed “



1. 如果想把文件进行对比可以通过 in ChangeList 这个选项来查看。如下图所示 .



1. 双击这个文件的名字，可以看到远程仓库和本地仓库的不同，这里是多了第三行.如下图.

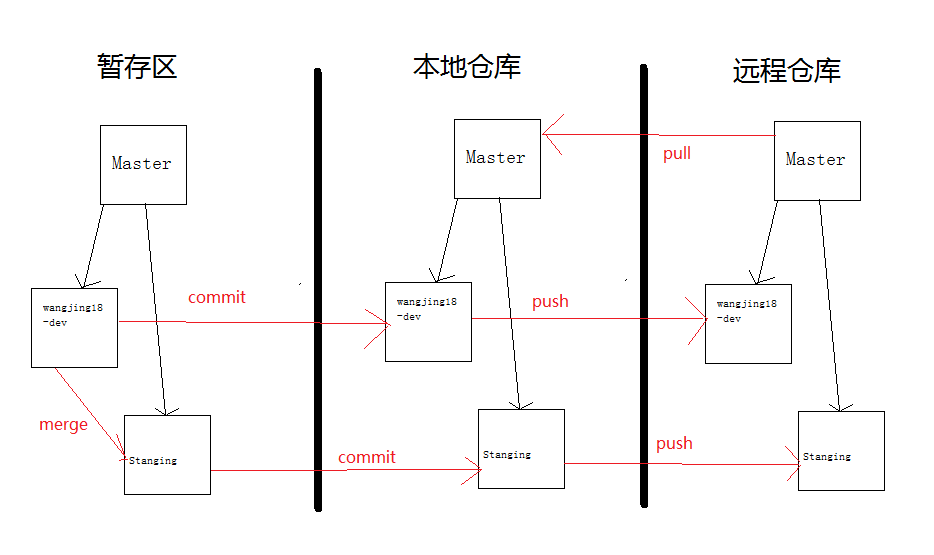


1. 这是git 版本管理代码的存储区域图。

简单来说

* 平时修改的是暂存区。
* Git commit的是到本地仓库
* Git push的是到远程仓库

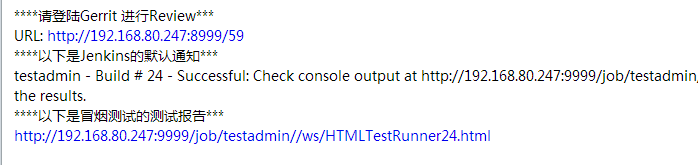
以下是一幅git操作参考图，方便我们来理解。



# 用户操作步骤(审核者)

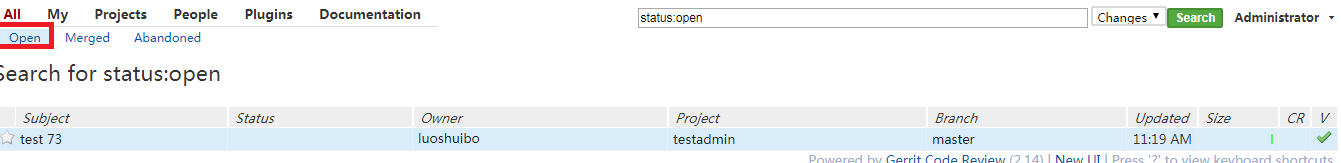
* **以代码审核者为例**

1 登陆邮件打开新的change链接



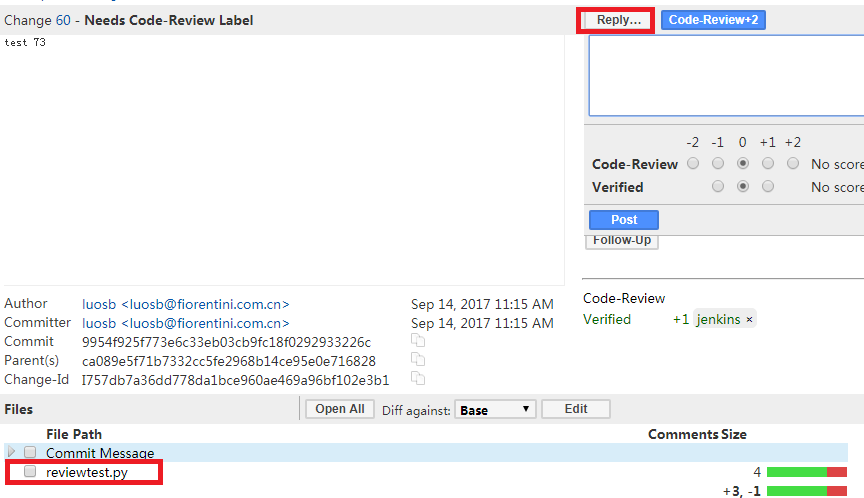
* 打开链接，也可以直接通过gerrit的web ui 界面来访问。其中open状态下的test73就是最新的代码提交。

http://192.168.80.247:8999

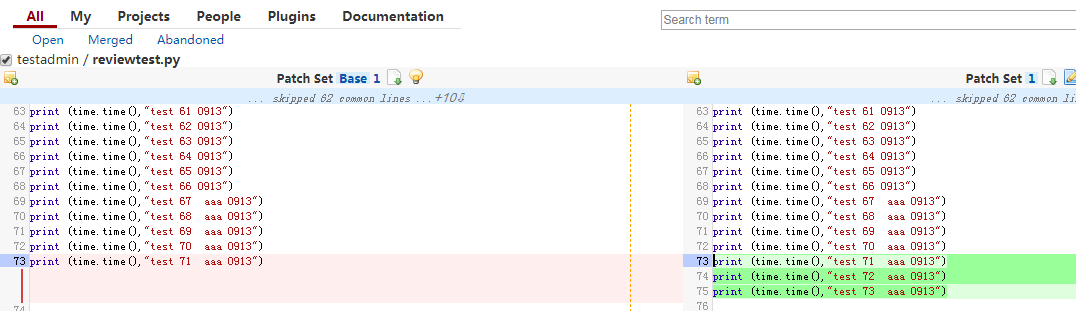


2 打开新更改代码文件。这里双击reviewtest.py文件，就可以看到内容了。

其中reply 中可以写一些描述，比如哪些是需要改进的。

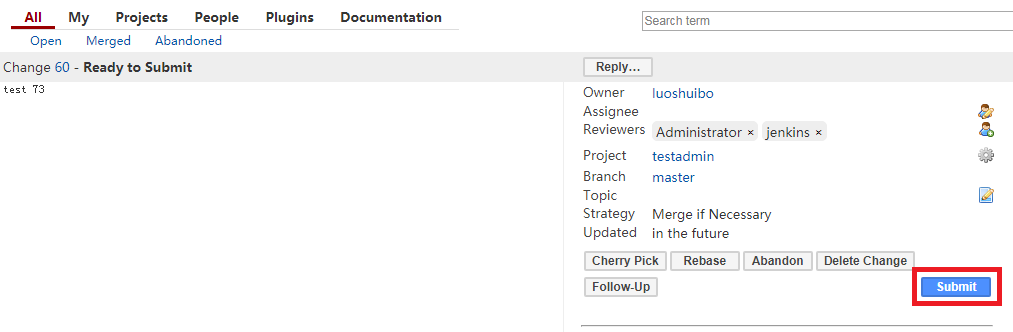


3 打开文件后，可以进行比对，这样子就可以清晰明了的进行review。如下图，可以看出右边后面两行是开发者最新提交的代码。

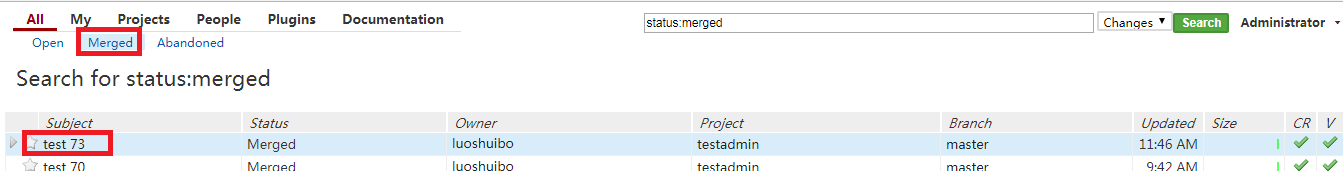


4 Code-review+2是一个快捷review的按钮，相当于不写评论直接review 通过。

5 submit提交。完成一个新代码的最终的更新.



6 确认submit提交已经成功。如图这个 test 73的change 状态已经由open改到Merged。

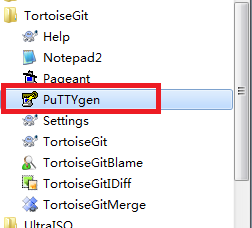


# 用户操作步骤(系统管理员)

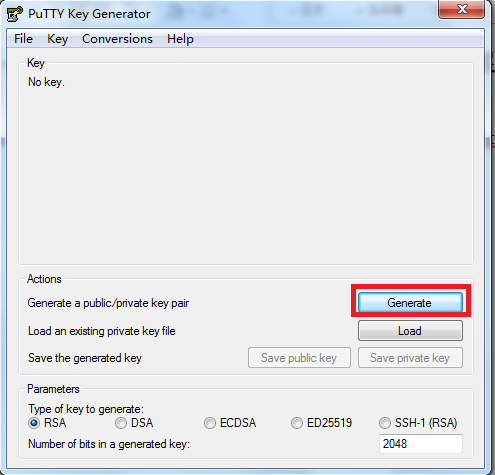
* **增加用户的ssh key (也可以开发人员自己生成，发给系统管理员)**

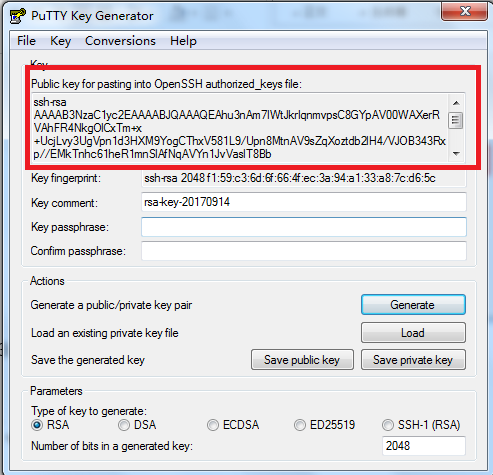
1. 生成开发者ssh key 。这里使用tortoisegit 自带的puttygen来生成。

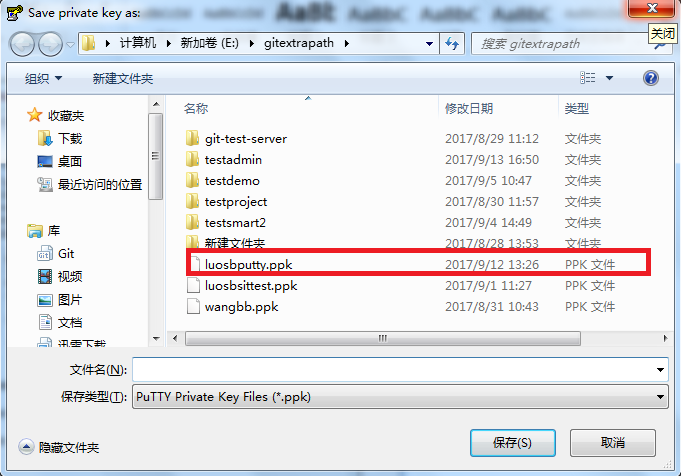
在系统开始菜单，找到PuTTYgen



1. 这里点击Generate并不能自动生成key。需要用鼠标摇晃，才能生成。

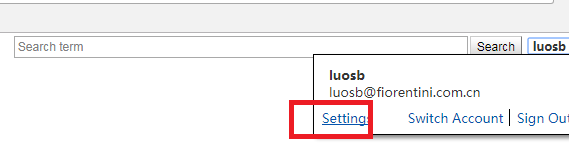


1. 这里需要的是ssh-rsa格式的密钥
2. 生成public key 要保存为private key的格式。
3. 最终生成的是一个是.ppk后缀的文件 。

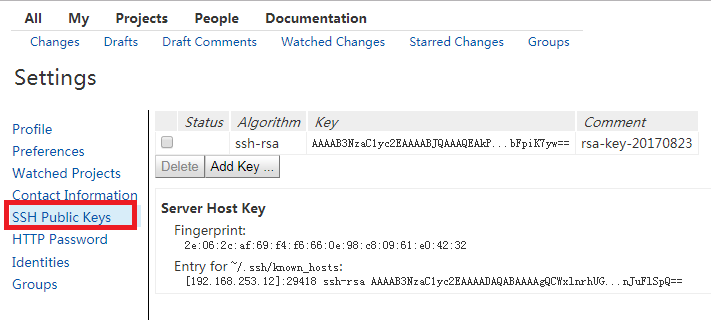


1. 管理员再把拿到的ssh-key添加到gerrit对应的帐户中.

打开gerrit的web ui界面，登陆luosb帐户。



1. 在[settings]中选择[SSH Public Keys],然后add key。填完会在右侧增加一条记录。



1. 如果是Linux 系统那可以用ssh-keygen工具

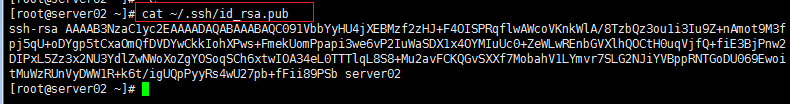
参考命令 ssh-keygen -t rsa -C "luosb"

这条命令会在~/.ssh/目录下生成id\_rsa和id\_rsa.pub文件，其中id\_rsa.pub文件就是我们需要的ssh-key.

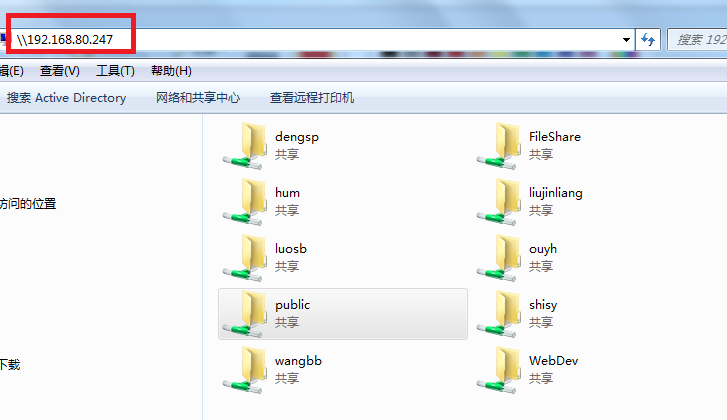


如下图可以看到ssh-key的内容

参考命令cat ~/.ssh/id\_rsa.pub



# 文档管理系统samba



* 使用方式，平时在办公区有内网的情况下,在资源管理器，地址栏输入[\\192.168.80.247](file:///\\192.168.80.247)即可.如上图所示。
* 其中 public是公共的文件，用来给大家平时做共享文件用的。
* 其中然后每个人有个单独的文件夹,比如ouyh，目前没有密码。
* Fileshare 和 webdev 是保留的

# 其他有待讨论的疑问.

1. code review 的频率 一天一次,比如早7:00 发一次
2. 假如每次变更发邮件,不必通知所有人.
3. 增加短信通知功能 ,是否可以没有网络通信的阻碍。