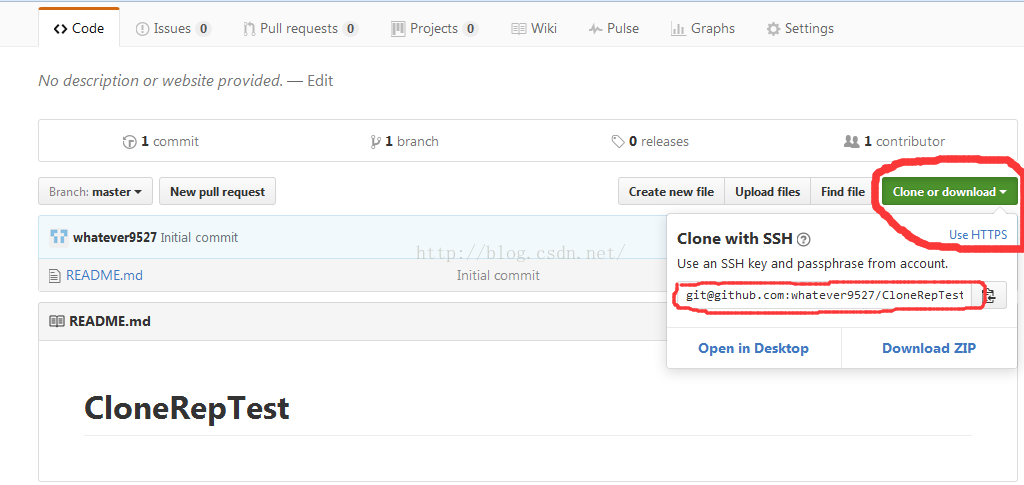
**--Git常用命令**

汇总：

git clone /git pull /git fetch /git status /git stash /git show /git diff /git add /git cherry-pick /git commit /git pull /git push /git log /gitk –all /git clean /git branch /git reset /git merge

详解：

1. git clone：将远程代码克隆岛本地仓库；



git clone <http://www.kernel.org/pub/scm/git/git.git> 将该路径的master分支克隆到了本地。

2. git branch (参数：-r/-a/dev/-d dev/-vv/-m oldname newname)

2.1 git branch 查看本地已经存在的分支，并且在当前分支的前面用"\*"标记。

2.2 git branch –r查看远程版本库分支列表

2.3 git branch –a查看所有分支列表，包括本地和远程

2.4 git branch dev创建名为dev的分支

2.5 git branch –d dev 删除dev分支

2.6 git branch –vv 查看本地分支对应的远程分支

2.7 git branch –m oldname newname 分支重命名

2.8 git branch --set-upstream master origin/next 与origin/nex建立追踪关系。

3. git checkout

3.1操作文件

3.1.1 git checkout filename 放弃单个文件的修改

3.1.2 git checkout . 放弃当前目录下的修改

3.2 操作分支

3.2.1 git checkout master 将分支切换到master分支

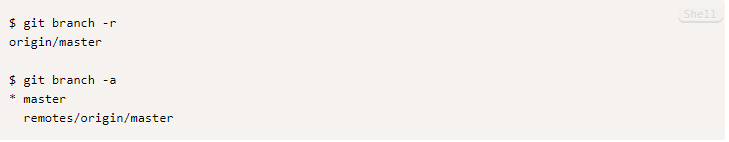
3.2.2 git checkout -b master 如果分支存在则切换分支，若不存在则创建并且切换到maste分支

3. git fetch 从其他存储库下载对象和引用

3.1.1 git fetch –all 下载所有远程存储库中的对象和引用

3.1.2 git fetch origin filename:localfilename 将远程分支filename下载到本地并且设置分支名为localfilename

3.1.3 git fetch origin master 取回origin主机的master分支



4. git pull 取回远程主机某个分支的更新，再与本地的指定分支合并, git pull是git fetch后跟git merge FETCH\_HEAD的缩写。

4.1 git pull origin next:master 取回origin主机的next分支，与本地的master分支合并

4.2 git pull origin next 与当前分支合并, 实质上，这等同于先做git fetch，再执行git merge

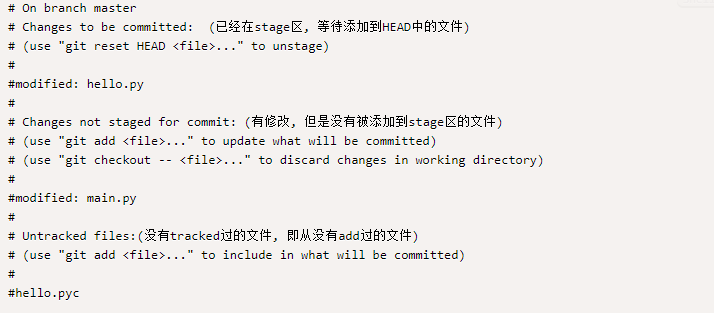
**\*git fetch和git pull的区别**

*git fetch*：相当于是从远程获取最新版本到本地，不会自动合并。

**

5. git status 命令用于显示工作目录和暂存区的状态,但显示已经commit到项目历史中去的信息。看项目历史的信息要使用git log

主要分三类内容：

**

**\*untracked文件：**

一般针对于这些未被跟踪的文件我的做法是执行：

5.\*.1 git clean –fd 命令，直接强制清理掉

5.\*.2 在.gitignore中把要忽略的文件放在其中， 每一个想忽略的文件应该独占一行



6. git add 跟踪当前目录下所有文件并放到暂存区

6.1 git add filename跟踪当前文件并放到暂存区

6.2 git add –A 保存所有的修改, 删除文件后需要执行；

6.3 git add .  保存新的添加和修改，但是不包括删除

6.4 git add -u 保存修改和删除，但是不包括新建文件

7. git stash 将上次commit前的所有修改暂存起来。

7.1 git stash pop 从暂存里面将最新的一条储存数据**剪切**下来

7.2 git stash pop 从暂存里面将最新的一条储存数据**复制**下来（建议）

7.3 git stash list 查看暂存列表细则

8. git commit 主要是将暂存区里的改动给提交到本地的版本库。每次使用git commit 命令我们都会在本地版本库生成一个40位的哈希值，这个哈希值也叫commit-id

8.1 **git commit -m “message” 提交并添加注释信息**

**8.2 git commit –a –m ‘message’** 加的-a参数可以将所有已跟踪文件中的执行修改或删除操作的文件都提交到本地仓库，即使它们没有经过git add添加到暂存区，注意；

注：新加的文件（即没有被git系统管理的文件）是不能被提交到本地仓库的。建议一般不要使用-a参数，正常的提交还是使用git add先将要改动的文件添加到暂存区，再用git commit 提交到本地版本库。

8.3 git commit --amend //也叫追加提交，它可以在不增加一个新的commit-id的情况下将新修改的代码追加到前一次的commit-id中

8.4 git commit –help 查看帮助

9. git reset文件和版本回退

9.1**使用“git reset --hard 目标版本号”命令将版本回退，**撤销的commit中所包含的更改被冲掉；**(不建议使用)**

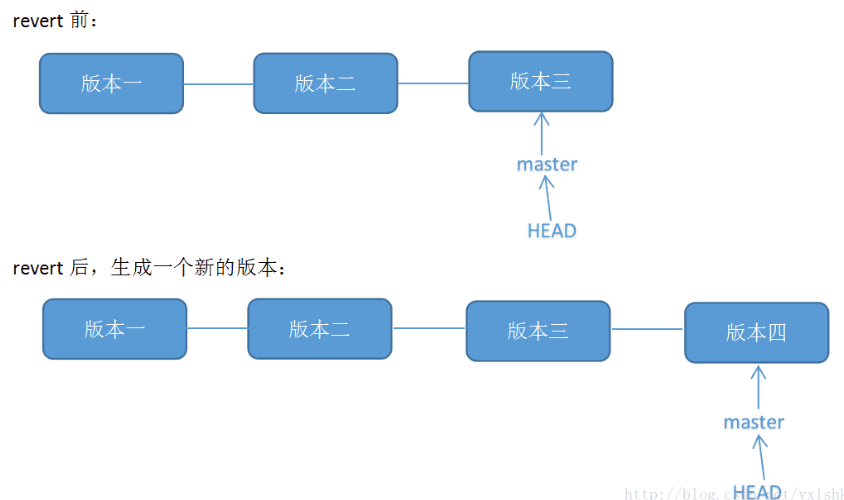
****

**9.2 git reset HEAD filename撤销add 后的指定filename**

**9.3 git reset –-soft （建议）**回退到某个版本，只回退了commit的信息，不会恢复到index file一级。如果还要提交，直接commit即可；

9.4 git reset –mixed 重置, 所有变更将作为local modifications保存在working area中但是并未staged的状态，你可以重新检视然后再做修改和commit

**10. git revert** 通过反做创建一个新的版本，这个版本的内容与我们要回退到的目标版本一样，但是HEAD指针是指向这个新生成的版本，而不是目标版本



10.1 git revert –n 版本号：

具体步骤：

git revert –n 版本号

git add filename 添加需要一并提交的文件

git commit –m ‘message’

git push origin 分支名称

git log 查看本地的版本信息，可见多生成了一个新的版本：



11. git push使用git commit命令将修改从暂存区提交到本地版本库.

11.1. git push –u origin master当前分支与多个主机存在追踪关系，则可以使用 -u 参数指定一个默认主机，这样后面就可以不加任何参数使用git push

11.2. git push –u origin master 当遇到这种情况就是不管是否存在对应的远程分支，将本地的所有分支都推送到远程主机

11.3. git push --force origin git push的时候需要本地先git pull更新到跟服务器版本一致，如果本地版本库比远程服务器上的低，那么一般会提示你git pull更新，如果一定要提交，那么可以使用这个命令

11.4. git push 的时候不会推送分支，如果一定要推送标签的话那么可以使用这个命令

12. git show 显示各种类型对象的相关信息

12.1. git show commit\_id 显示当前版本需要的内容要点。

12.2 git show commit\_di ^{tree} 显示当前tree对象的目录结构。

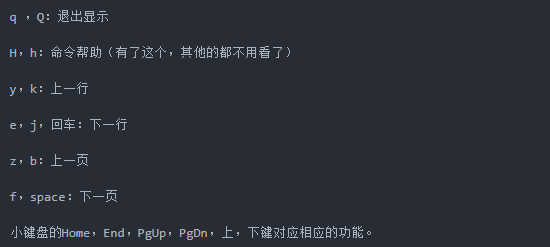
12.3 git show tag标签 查看有tag标签的提交信息

13.git diff 用于比较两次修改的差异

13.1. git diff 不加参数即默认比较工作区与暂存区

13.2. git diff –-cached 比较暂存区与最新本地版本库中最近一次commit的内容

13.3．git diff HEAD[<path>...]  如果HEAD指向的是master分支，那么HEAD还可以换成master



14.git cherry-pick可以选择某一个分支中的一个或几个commit(s)来进行操作

15. git log

15.1. git log –p按补丁显示每个更新间的差异

15.2. git log –-stat显示每次更新的修改文件的统计信息

15.3. git log --shortstat：只显示--stat中最后的行数添加修改删除统计

15.4. git log  --name-only：尽在已修改的提交信息后显示文件清单

15.5. git log --name-status：显示新增、修改和删除的文件清单

15.6. git log --abbrev-commit：仅显示SHA-1的前几个字符，而非所有的40个字符

15.7. git log --relative-date：使用较短的相对时间显示（例如："two weeks ago"）

15.8. git log  --graph：显示ASCII图形表示的分支合并历史

15.9. git log  --pretty：使用其他格式显示历史提交信息

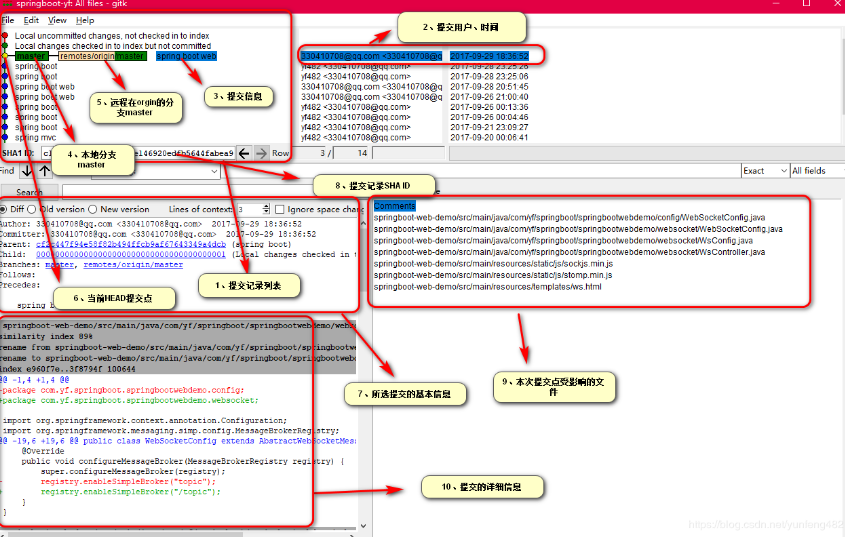
15.10. git log -–author=’name’ :显示指定提交人的所有提交记录

15.11. git log –since=’3 weeks ago’:显示三周之前的提交记录

15.12. 除了可以使用 - -since , 也可以使用 - - after , - -util , - -before ， 取值方式相同

16. gitk –-all显示存储库或选定提交点的更改信息。这包括可视化commit图，显示与每个commit相关的信息，以及每个修订的树中的文件

会调用一个窗口：



17. git rebase 合并，但git rebase可以得到一个更加简洁的提交历史

18. git merge合并，采用git merge dev处理提交log是按照时间戳先后顺序的

19. git clean命令用来从你的工作目录中删除所有没有tracked过的文件

19.1. git clean –n是一次clean的演习, 告诉你哪些文件会被删除. 记住他不会真正的删除文件, 只是一个提醒

19.2. git clean -f删除当前目录下所有没有track过的文件. 他不会删除.gitignore文件里面指定的文件夹和文件, 不管这些文件有没有被track过

19.3. git clean -f <path>删除指定路径下的没有被track过的文件

19.4. git clean –df删除指定路径下的没有被track过的文件

19.5. git clean –xf删除当前目录下所有没有track过的文件. 不管他是否是.gitignore文件里面指定的文件夹和文件