|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| git修改注释 | | | 1 修改最新的commit  1）还没有push   git commit –amend  执行该命令后，会弹出git的编辑器，直接修改注释  2) 已经push  git commit –amend  弹出的编辑器中修改注释，然后  git pull  git push origin master  或者  git push -f origin master  2修改历史提交  假设要修改当前版本往前三次版本的状态,四次就把HEAD~3改成HEAD~4,以此类推  git rebase -i HEAD~3  假设要从第一个版本开始修改  git rebase -i –root  在弹出的窗口中，把你要修改的commit的pick改成edit，保存退出  然后  git commit –amend  git rebase –continue  如果已经push，那么再执行git push -f origin master | | | |
|  | | | |
| log相关命令 | | | 1 git log –stat  2 git log –pretty=fuller  3 git log –pretty=oneline  4 git log –pretty=raw: 显示tree parent  5 git log –pretty=raw –graph <id>，显示提交的跟踪链  6 git log –oneline 显示更小的提交ID  7 git log –oneline –decorate -4 | | | |
| status相关命令  执行该命令时：  1 根据.git/index文件中记录的时间戳和长度等信息  判断工作区文件是否改变  2 如果工作区文件的时间戳改变了，则打开文件，读取内容，并与更改前的原始文件比较  1）内容被修改，则将新的时间戳记录到.git/index文件中，并且显示改变的内容  2）文件内容没有改变，则将新的时间戳记录到.git/index文件中  3 touch welcome.txt  执行ls -al .git/index，时间戳不变  但如果执行git status -s，则执行ls -al .git/index，则时间戳变成执行git status时的时间戳，说明因为welcome.txt时间戳改变了，需要写入.git/index文件，所以导致.git/index文件时间戳也改变了  这里的时间戳是指.git/index文件的时间戳，这个文件就是用来记录welcome.txt等这些文件的时间戳的 | | | 1 git status -s  MM:  第一个M：版本库与暂存区的比较  第二个M：工作去与暂存区的比较  2 git status -s -b  精简输出并且显示当前工作分支名称 | | | |
| diff相关命令 | | | 版本库 | 工作区 | 暂存区 | 版本库 |
|  | git diff | |  |
| git diff HEAD | |  |  |
|  |  | git diff --cached | |
| reset相关命令  有两种模式  1 git reset <commit> -- <filename/paths>  这种因为包含了具体的路径，不会重置引用，因为包含具体路径意味着只是reset该commit的一部分文件。只重置暂存区中对应的文件  2 git reset [--soft | --mixed | --hard] <commit>  这种由于没有指定具体的文件，只是给了commit，意味着该commit的所有文件都需要被重置，这就需要改变引用指向  第一种用法，不带hard soft等选项  git reset <commit> -- <path>  1) 如果省略commit，则默认是HEAD  2) 有commit，则对应版本库中的commit  git reset – filename  git reset –mixed  这两者区别是后者会改变HEAD，当然这条指令还是之前的HEAD  这种用法是不会改变HEAD，也不会改变工作区，只影响暂存区  第二种用法，带hard soft mixed等这些选项，并且不能带path，否则git会报错  1 git reset –soft HEAD~1  那么，HEAD指针改变，git log显示之前的HEAD提交没了  2 git reflog show master  git reset –soft master@{1}  git log则发现之前的最新提交又回来了  也就是说HEAD指针是指向代码库的，改变HEAD指针，代码库也就改变了，执行git push时就是把HEAD指针指向的提交给push到github上 | | | 1 git reset –hard <commit>  替换引用指向  重置工作区  重置暂存区  2 git reset –soft <commit>  只改变引用指向，不改变暂存区和工作区  3 git reset <commit>  改变引用指向  重置暂存区  4 git reset /git reset HEAD  用HEAD指向的目录树重置暂存区，并且引用指向不变，指向HEAD相当于不变  5 git reset – filename/ git reset HEAD filename  将文件filename的改动撤出暂存区  6 git reset –soft HEAD^  工作区和暂存区不变，但是引用向前回退一次  当对最新的提交说明或者提交改动不满意时，撤销最新提交以便重新提交  git commit –amend. 执行这个命令后，新提交的id被改变，相当于之前不满意的那次提交被撤销  等价于  git reset –soft HEAD^  git commit -e -F .git/COMMIT\_EDITMSG  比如对某个文件做了修改，然后提交了，但是发现修改不正确，则用这个命令撤销最先提交，然后修改好后，再git add/ git commit，这样之前的那次提交就会从版本库中消失，恢复那次提交的办法就是用git reflog命令 | | | |
| 删除暂存区，工作区不影响 | | | git rm –cached <file> | | | |
| checkout相关命令  跟reset命令一样，当包含具体的文件时，不会改变HEAD头指针  两种用法，不会改变HEAD  1 想要暂存区的覆盖工作区  git checkout – filename  git checkout -- .  git checkout .  2 想要版本库中的覆盖工作区  git checkout <commit> filename  注意，暂存区也会被覆盖 | | | 1 git checkout <commit> [--] <path>  如果省略<path>，则从暂存区index进行检出，而reset默认是HEAD  reset一般是用于重置暂存区，而检出命令主要是覆盖工作区  1）如果省略<commit>，那么用暂存区中的文件覆盖工作区中的文件  2）如果有<commit>，那么用指定commit中的文件（版本库）覆盖暂存区和工作区中对应的文件  2 git checkout <branch>  改变HEAD头指针。只要用来切换分支。如果省略<branch>，则表示对工作区进行状态检查  3 git checkout [-m] [-b] <new branch> <start\_point>  创建和切换到新的分支，新的分支从<start\_point>指定的提交开始创建 | | | |
| 查看工作区、暂存区、版本库目录树 | | 1 版本库  git ls-tree -l HEAD  2 暂存区  git ls-files -s  或者  git write-tree  git ls-tree -l <commit id>  3 工作区  ls | | | | |
| git tag -m "xxx" tag\_name | | 里程碑 | | | | |
| git describe | | 显示离当前提交最近的tag | | | | |
| cat-file命令 | 1 git cat-file -t <id> 显示id的类型：commit/tree  2 git cat-file -p <id>  id是tree id，则显示该树下的文件  id是非tree id，则显示该提交的相关信息，如tree parent author | | | | | |
| reflog命令 | 1 tail -5 .git/logs/refs/heads/master  2 git reflog show master    3 git reset –hard master@{1}  4 git reflog -1 这是HEAD头指针的变迁记录，HEAD@ | | | | | |
| stash命令 | 1 git stash  保存工作区和暂存区状态  2 git stash list  查看保存的进度  3 git stash pop  恢复进度 | | | | | |
| clean命令 | 1 git clean -nd  测试运行查看哪些文件和目录会被删除  2 git clean -fd  强制删除多余的目录和文件 | | | | | |
| 恢复删除的文件  如果工作区的文件被删除了，则还是可以恢复回来 | 1 git cat-file -p HEAD~1:welcome.txt > welcome.txt  2 git show HEAD~1:welcome.txt > welcome.txt  3 git checkout HEAD~1 – welcome.txt  不能用 git reset –hard HEAD~1，这样HEAD指针也会随着改变 | | | | | |
| 移动文件 | 1 git mv welcome.txt README  git commit -m "change name" | | | | | |
| git add | 1 git add -u 将文件的修改、删除，添加到暂存区  2 git add 将文件的修改、新建，添加到暂存区  3 git add -A 将文件的修改、删除、新建，添加到暂存区 | | | | | |
| github | 1 注册github账号  2 ssh-keygen -t rsa -C "your\_email@youremail.com"  3在~/下生成.ssh文件夹，进去，打开id\_rsa.pub，复制里面的key。  回到github上，进入 Account Settings（账户配置），左边选择SSH Keys，Add SSH Key,title随便填，粘贴在你电脑上生成的key。  4 ssh -T git@github.com  验证是否成功  5 git config --global user.name "your name"  git config --global user.email "your\_email@youremail.com" | | | | | |
| 删除文件 | 1 git rm xx xxx xxxx  git commit  2 rm xx  git add -u  git commit | | | | | |
|  |  | | | | | |
|  |  | | | | | |