GIT 日常操作命令

1、**git** 三种状态

已提交(committed)　　已修改(modified)　　已暂存(staged)

2、获取项目的 **git** 仓库

git init　　//从当前目录初始化
git clone gitosis@192.168.2.4:soul/soul　　//从现有仓库克隆

3、配置 **git** 仓库相关信息

git config user.name "yayun.tian"
git config user.email yayun.tian@soulinfo.com
…….

4、检查当前文件状态

git status

5、跟踪新文件

git add <filename>

gir add --all　　//一次性跟踪当前目录全部文件

注：可以创建一个名为*.gitignore* 的文件,列出要忽略的文件模式，例如*:*

 # 此为注释 – 将被 git 忽略

**.a　　# 忽略所有 .a 结尾的文件*

!lib.a　　# 但 lib.a 除外

/TODO　　# 仅仅忽略项目根目录下的 TODO 文件,不包括 subdir/TODO

build/　　# 忽略 build/ 目录下的所有文件

doc/.txt　　# 会忽略 doc/notes.txt 但不包括 doc/server/arch.txt*

6、移除文件

git rm <filename>　　//移除暂存区的内容而且工作目录也连带删除

注：如果仅仅想删除暂存区可以加上—*cached* 选项

7、提交

先用 git diff --color 查看确认修改

git commit -a -m “comment”　　//comment 代表此次修改的注释

git commit --amend　　//修改最后一次提交

如果忘记暂存某些文件就提交了，可以暂存后使用 amend 再次提交

注：加上*-a*表示把已经修改的文件全部提交，省去暂存的步骤

8、查看提交历史

git log

-p //展开显示每次提交的差异

-n //仅仅显示最近的 n 次提交

--stat //仅显示每次修改的概要信息

注：图形化工具查阅提交历史 *gitk*

9、对比操作

git diff //显示还没有暂存起来的改动

git diff --cached //已经暂存起来的文件和上次提交时的快照之间的差异

git diff HEAD //查看当前工作目录和已经提交的差别

git diff --stat //查看文件改动的大概信息

10、补丁操作

生成补丁方式 1 git diff > 补丁名称

git apply 补丁名称

生成补丁方式 2 git format-patch HEAD ~

patch -p1 < 补丁名称

git am 补丁名称

注：对于 *git diff* 生成的 *patch*，你可以用 *git apply --check* 查看补丁是否能够干净顺利地应用到当前分支中，生成的 *Patch* 兼容性强。

11、恢复命令 **reset**

git reset HEAD <file> //取消已经暂存的文件

git reset --mixed commit //默认方式，它回退到某个版本，只保留源码，回退 commit 和 index 信息（git diff 可以查看到修改内容）

git reset --soft commit //回退到某个版本，只回退了 commit 的信息，如果还要提交，直接 commit 即可（git diff --cached 可以查看到修改内容）

git reset --hard commit //彻底回退到某个版本，源码也回退

12、分支操作

查看分支 git branch(本地) git branch -a（全部）

创建分支 git branch testing

切换分支 git checkout testing

重命名 git branch -m <old_branch_name> <new_branch_name>

新建分支并切换 git checkout -b <branch_name>

git checkout -b <branch_name> origin/branch

合并分支 git merge 待合并分支 需要合并分支

删除以前的分支 git branch -d iss53 //强制删除 -D

git branch 查看当前的分支--v 、 --merge、 —no-merge 选项可以使用

13、远程仓库操作

查看远程仓库 git remote（简单查看） git remote -v（详细查看）

添加远程仓库 git remote add <名称> gitosis@192.168.2.4:soul/soul-1

重命名远程仓库 git remote rename <old_name> <new_name>

将远程分支更新到本地 git fetch //不会 merge

git pull //直接 merge 到本地

推送到远程分支 git push 远程仓库名 分支名

推送本地所有分支 git push --all（默认 origin） git push --all <仓库名>

删除远程分支 git push origin ： <分支名>