安装之后命令：

git config –global user.name “Your Name”

git config –global use.email [email@example.com](mailto:email@example.com)

建立本地仓库：

mkdir test

cd test

git init

touch file1

git add file1

git commit –m “add file1”

恢复等操作：

git log

git log –pretty=oneline

git reset –hard HEAD^

git reset –hard xxxxxxx

git reflog

git status

git diff HEAD --file 工作区和版本库里面最新版本的区别

git checkout -- file可以丢弃工作区的修改

git reset HEAD file可以把暂存区的修改撤销掉

rm file 删除本地文件

git status

git rm file 删除版本库中文件

git checkout – file 删除错误，恢复误删文件

连接远程：

Ssh-keygen –t rsa –C [email@example.com](mailto:email@example.com) 将公钥粘入github

github上建立test库

git remote add origin [git@github.com:”name”/test.git](mailto:git@github.com:)

git push –u origin master

克隆远程：

git clone [git@github.com: ”name”/test.git](mailto:git@github.com:xxxx/test.git)

创建分支：

git checkout –v dev 等价于 git branch dev 和 git checkout dev

git branch

git checkout master

git merge dev

git branch -d dev

git log --graph --pretty=oneline --abbrev-commit

git merge --no-ff -m “xxxx” dev 合并禁用fast forward

git branch –D dev 强行删除未合并的分支

BUG分支：

git stash 将当前工作现场存储

git stash list 查看保存的工作区

git stash apply 和 git stash drop

git stash pop

多人协作：

git remote

git remote -v

一方推送：git push origin master git push origin dev

他人clone：git clone [git@github.com:”name”/test.git](mailto:git@github.com:”name”/test.git) 只能看到master分支。因此，需要创 建远程origin的dev分支：git checkout –b dev origin/dev

如果别人在提交dev分支后，自己也更改了该分支中的相同文件，则有冲突： git branch --set-upstream dev origin/dev git pull 解决冲突 提交

删除远程分支：

git branch –a

git push origin –delete dev

标签管理：

git tag v1.0

git tag

git log --pretty=oneline --abbrev-commit

git tag v0.9 xxxxxxx

git show v0.9

git tag -a v0.1 –m “version 0.1” xxxxxxx

git tag –d v0.1

git push origin v0.1

git push origin --tags

删除远程tag：

git tag –d v0.9

git push origin :refs/tags/v0.9

git push origin --delete tag v0.9

获取远程tag：

git fetch origin tag <tagname>

查看远程：(删除不存在对应远程分支的本地分支)

git remote show origin

git remote prune origin

git fetch –p(同上一条同样效果)

忽略文件：

.gitignore文件，放到版本库中

别名管理：

git config --global alias.st status

git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

每个仓库的Git配置文件都放在.git/config文件中

当前用户的Git配置文件放在用户主目录下的一个隐藏文件.gitconfig中

在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；

那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；

你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

所以，团队合作的分支看起来就像这样：

