### 1,在工作目录中初始化新仓库

$ git init

### 2,检查当前文件状态

$ git status

### 3,跟踪新文件

将文件放入暂存区

$ git add README //跟踪readme文件

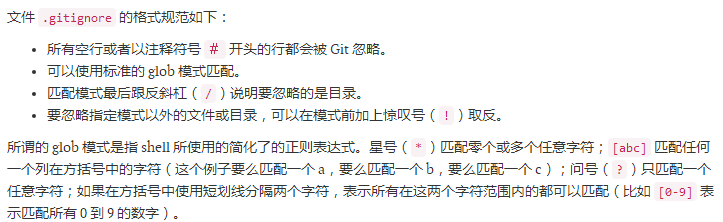
$ git add . //跟踪目录下的所有文件

### 4，忽略某些文件

$ cat .gitignore

\*.[oa] //第一行告诉 Git 忽略所有以 .o 或 .a 结尾的文件。

\*~ //第二行告诉 Git 忽略所有以波浪符（~）结尾的文件，





### 5，查看已暂存和未暂存的更新

git status 的显示比较简单，仅仅是列出了修改过的文件，

可以用 git diff 命令，显示还没有暂存起来的改动

若要看已经暂存起来的文件和上次提交时的快照之间的差异，可以用 git diff --cached 命令。

### 6，跳过使用暂存区域

git commit -a -m 'added new benchmarks'

### 7，移除文件

直接运行git rm XXXX 就会【在所有已跟踪文件种】删除匹配的文件并删除远程跟踪；

【git rm -f XXXX -f（ force ） 强制删除指令，以防误删除文件后丢失修改的内容】

如果想删除文件并移除远程跟踪时：

如果删除之前目标文档没有修改过，则使用 git rm XXXX

如果删除之前修改过，还没有git add . :则使用 git rm -f XXXX

如果删除之前已经git add . 则使用git rm -f XXXX

如果删除之前已经git commit , 则使用 git rm XXXX

如果不想删除文件，但想移除远程跟踪，并稍后在.gitignore中添加相应修改时：

则在相应的步骤中加上 --cached  选项

$ git rm --cached readme.txt //后面可以列出文件或者目录的名字以便稍后在

$ git rm log/\\*.log //删除所有 log/ 目录下扩展名为 .log 的文件。

$ git rm \*~ //只删除当前目录下的匹配文件，

$ git rm \\*~ //递归删除当前目录及其子目录中所有 ~ 结尾的文件。

【指令中加上 \ 则会有递归查询并操作的效果】

## 8，查看提交历史

需求：

按提交时间列出所有的更新运行：git log

展开显示每次提交的内容差异：git log -p

仅展开显示最近2次提交的内容差异：git log -p -2

仅展开单词层面的对比，比行层面的对比：git log -p --word -diff

列出所有最近两周内的提交：git log --since=2.weeks

### 撤销操作

提交(commit)后，想撤消刚才的提交操作

Eg:

$ git commit -m 'initial commit'

$ git add forgotten\_file

$ git commit --amend

上面的三条命令最终只是产生一个提交，第二个提交命令修正了第一个的提交内容

### 10，跳过使用暂存区域

使用git add XXX 后，可以使用 git reset HEAD <file>... 的方式取消暂存。

$ git reset HEAD benchmarks.rb

实测：直接输入git reset 则默认撤销全部暂存；直接输入git reset benchmarks.rb则也可以达到撤销此文件的效果

### 对远程库的操作

查看当前配置有哪些远程仓库，可以用 git remote 命令；

git remote：它会列出每个远程库的简短名字

git remote：它会列出每个远程库的简短名字

$ git remote : 它会列出每个远程库的简短名字

$ git remote -v //显示当前的clone地址

$ git remote add [shortname] [url] //添加远程仓库

$ git fetch [remote-name] //从远程仓库抓取数据（eg:git fetch origin）

$ git push [remote-name] [branch-name] //推送数据到远程仓库

$ git remote show [remote-name] // 查看某个远程仓库的详细信息

$ git remote rename [旧仓库名] [新仓库名] //修改远程仓库的名字

$ git remote rm  [仓库名] //移除远程仓库名

1,【有一点很重要，需要记住，fetch 命令只是将远端的数据拉到本地仓库，并不自动合并到当前工作分支，只有当你确实准备好了，才能手工合并。】

2,注意，对远程仓库的重命名，也会使对应的分支名称发生变化，原来的 pb/master 分支现在成了 paul/master。

## 11，打标签

$ git tag //列出现有标签

$ git tag -l 'v1.4.2.\*' 如果你只对 1.4.2 系列的版本感兴趣

## 12，新建并切换到该分支

$ git checkout -b iss53

这相当于执行下面这两条命令：

$ git branch iss53

$ git checkout iss53

## 13，合并分支

$ git merge dev 默认合并，会将dev分支上最近修改的commit记录全都合并到master上

$ git merge --squash dev 压缩式合并，会将dev分支合并到master上来，commit记录只保留一条。 【git commit -a -m “压缩式合并的记录”】

## 14，分支信息查看

$ git branch // 查看本地分支

$ git branch -a // 查看本地分支和远程分支

$ git branch -v //查看各个分支最后一个提交对象的信息

$ git branch --merged // 查看哪些分支已被并入当前分支

$ git branch --no-merged // 查看哪些分支没有被并入当前分支

## 删除分支

$ git branch -d hotfix 只有当hotfix上的更改全都commit过了才能删除

$ git branch -D hotfix 强制删除分支hotfix；没有commit过也能删除

## 推送和删除本地分支

git push (远程仓库名) (分支名)：

git push origin serverfix:serverfix可以简写为 git push origin serverfix

上传我本地的 serverfix 分支到远程仓库中去，取名为 serverfix 分支

git push origin :serverfix

上传一个空的分支到远程仓库中去，将替换掉:serverfix分支---即删除serverfix分支

$ git checkout -b serverfix origin/serverfix //操作A

//新建的 serverfix 本地分支，其内容同远程分支 origin/serverfix 一致，

$ git checkout --track origin/serverfix

//1.6.2 以上版本的 Git，还可以用 --track 选项简化操作A

## 分支的衍合

$ git checkout experiment

$ git rebase master