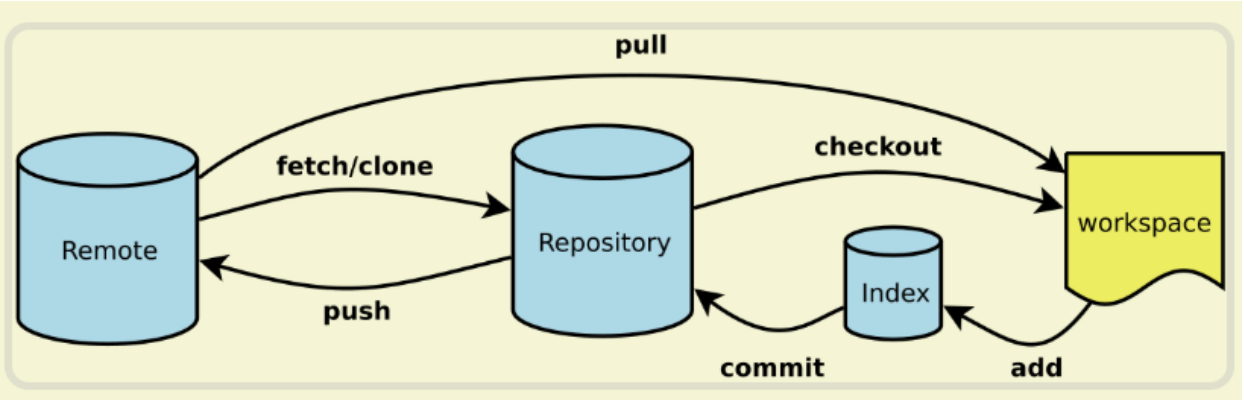
GIT总复习



一、初始化配置

1，本地配置

本地配置将在每次commit时提交，当使用global参数时，对所有提交有效

```
1  //用户名
```

```
$ git config --global user.name ["username"]
//邮箱
$ git config --global user.email ["email"]
```

查看config配置

```
$ git config --list

//查看某一项配置
$ git config [key] //git config user.name

//编辑Git配置文件
$ git config -e [--global]
```

技巧

```
//启用彩色命令行
$ git config --global color.ui auto
//文本编辑器
$ git config --global core.editor [editor name]
```

获取帮助

```
//三种方式获取帮助信息
$ git help <verb>
$ git <verb> --help
$ man git-<verb>

//例如: $ git help config
```

2，初始化仓库

有两种方式初始化，本地或远程

本地新建（一般不用）

```
//本地,通过git bash进入仓库所在的文件夹
$ git init


//新建一个目录，将其初始化为Git代码库
$ git init [project-name]
```

从远端仓库克隆

```
//远程克隆
$ git clone [url]

//例如：$ git clone git@github.com:michaelliao/gitskills.git，克隆一个本地库,则在当前文件夹下会多一个gitskills的文件夹。
```

添加远程仓库（关联）

```
$ git remote add origin git@github.com:michaelliao/learngit.git
```

二、进入工作区编辑

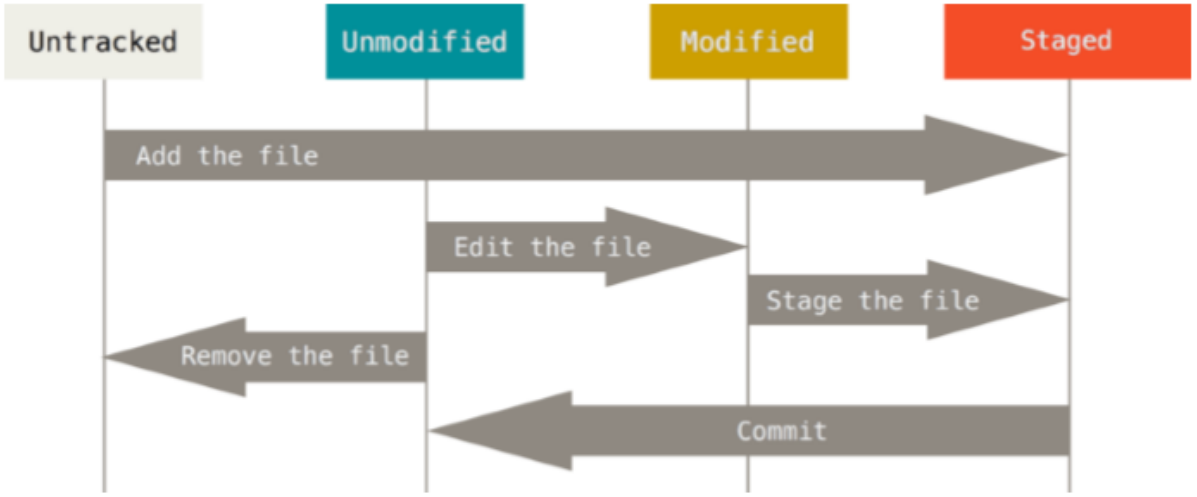


Figure 8. 文件的状态变化周期

1，添加文件到缓存区

add增加了仓库对这个文件的追踪

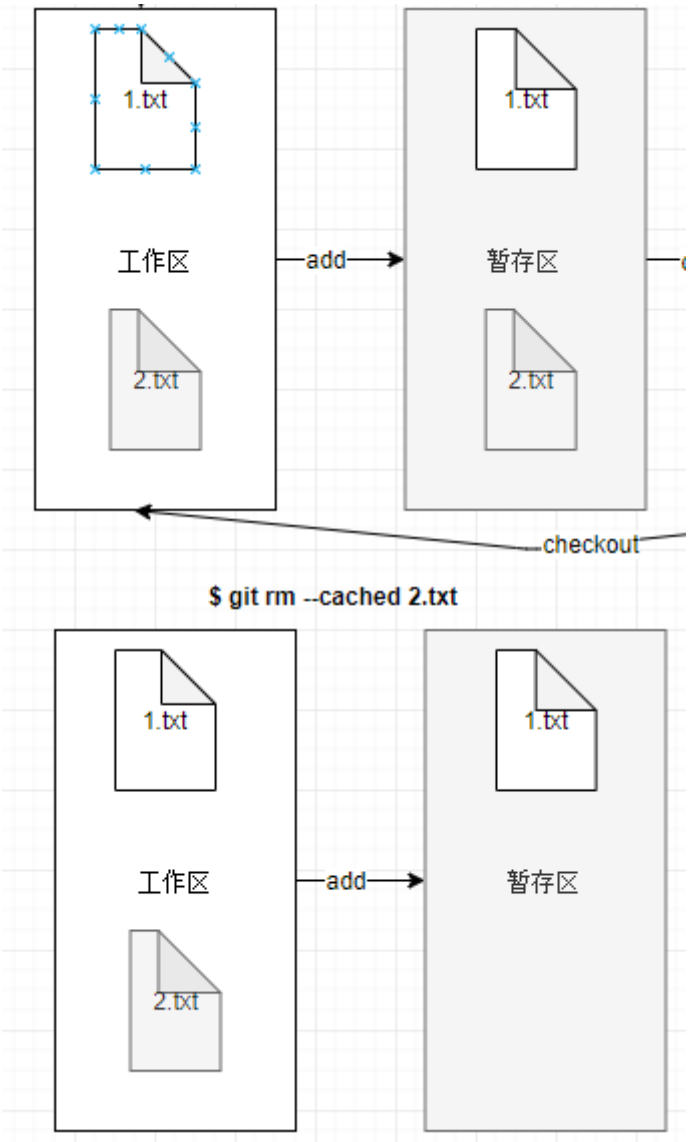
```
//1,添加一个文件
$ git add [filename]
//2,添加多个文件
$ git add [file1] [file2]
//3,增加文件夹
$ git add [dir]
//4,添加所有修改过的文件
$ git add *
//5,增加当前目录下所有文件
 $git add .
```

2，查看变更信息

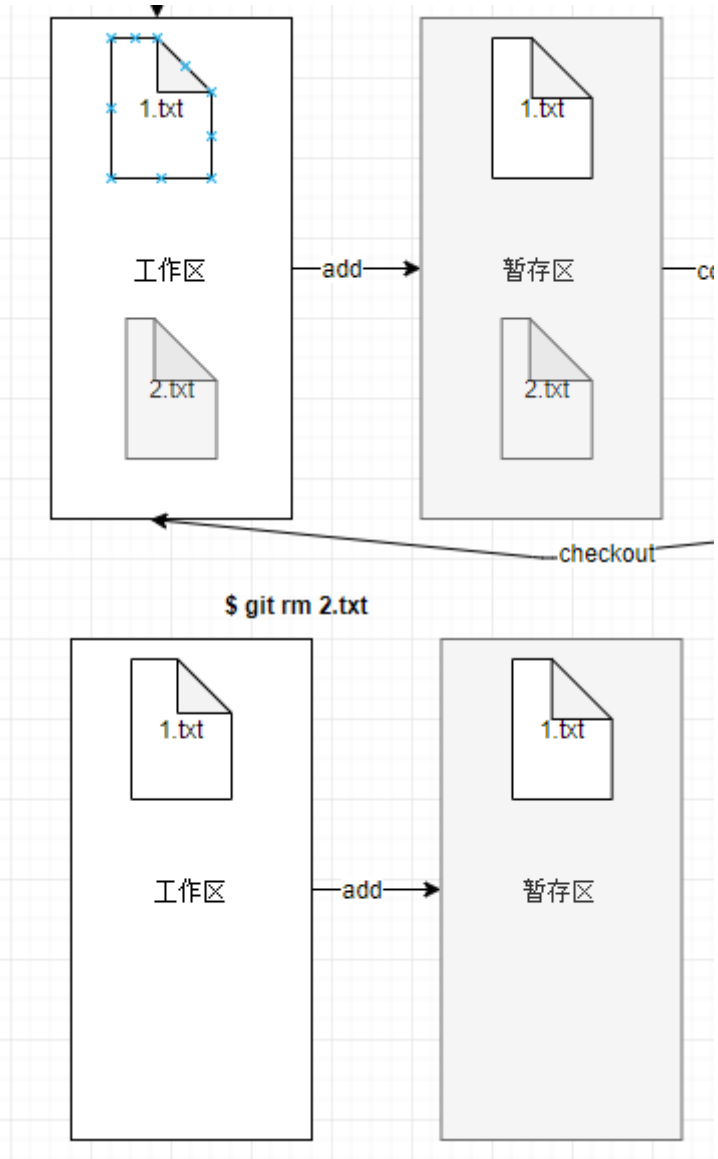
```
//显示有变更的文件(modified文件和unmodified的文件)
```

```
$ git status

//显示暂存区和工作区的差异(未add的和已add的文件差异)
$ git diff

//显示暂存区和上一个commit的差异
$ git diff --cached [file]
```

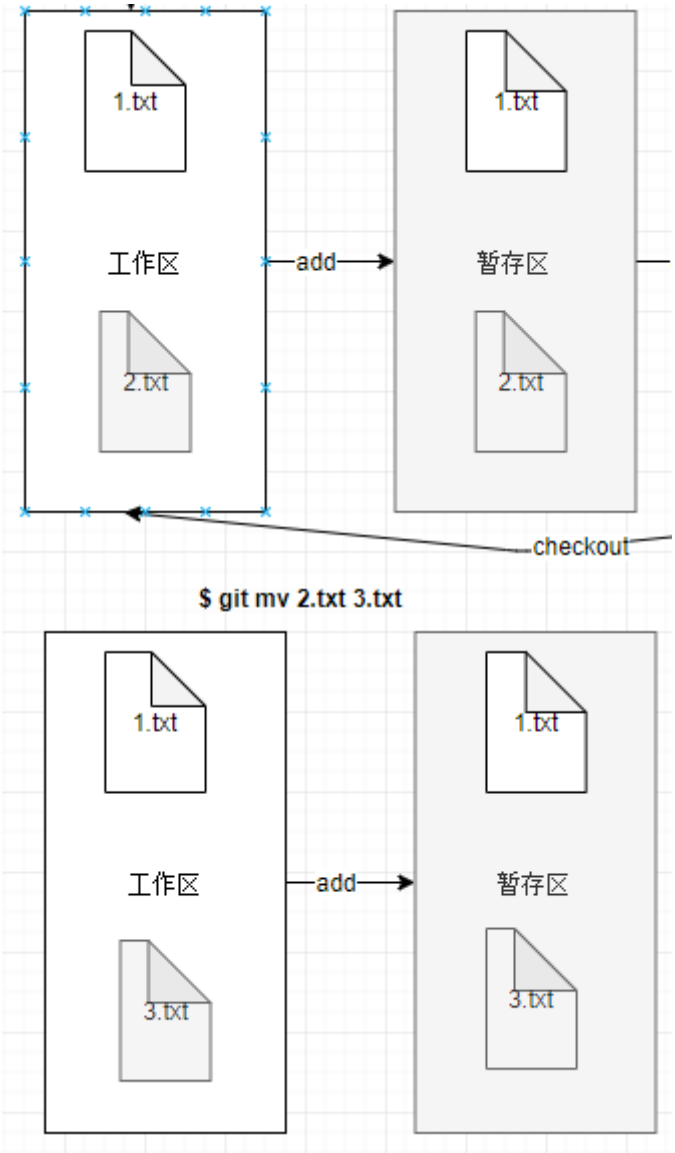
3，从缓存区撤出文件




```
//停止追踪指定文件，但该文件会保留在工作区
$ git rm --cached [file]
```

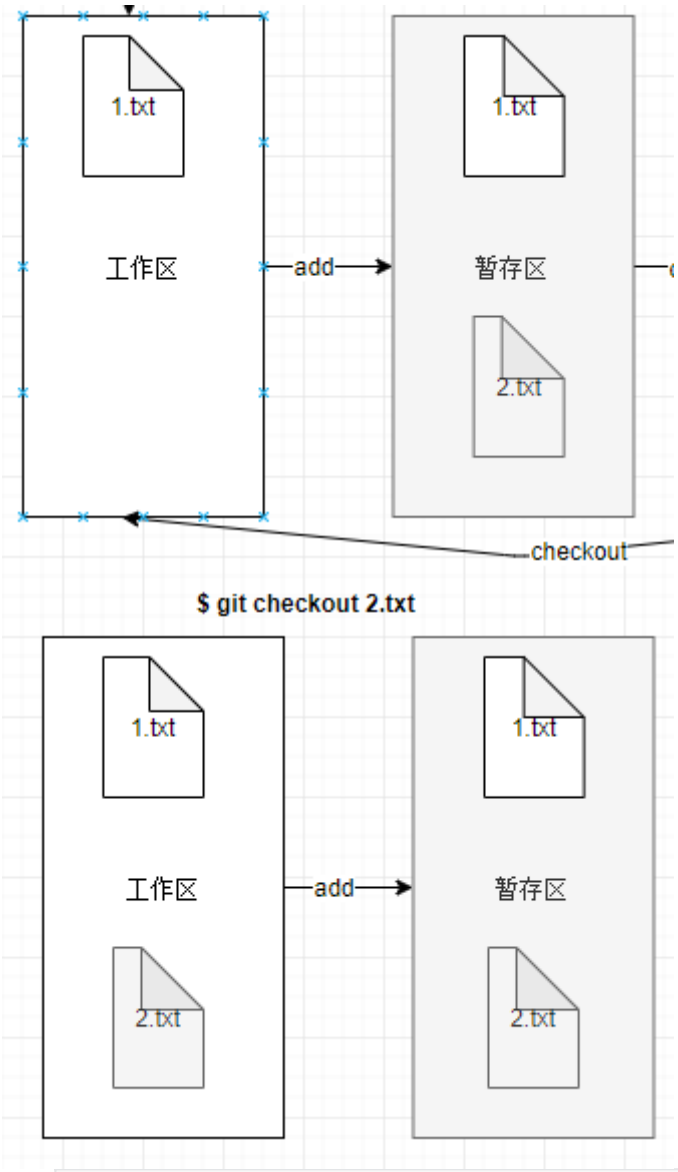



```
//删除工作区文件，并且将这次删除放入暂存区
$ git rm [file1] [file2] ...
```

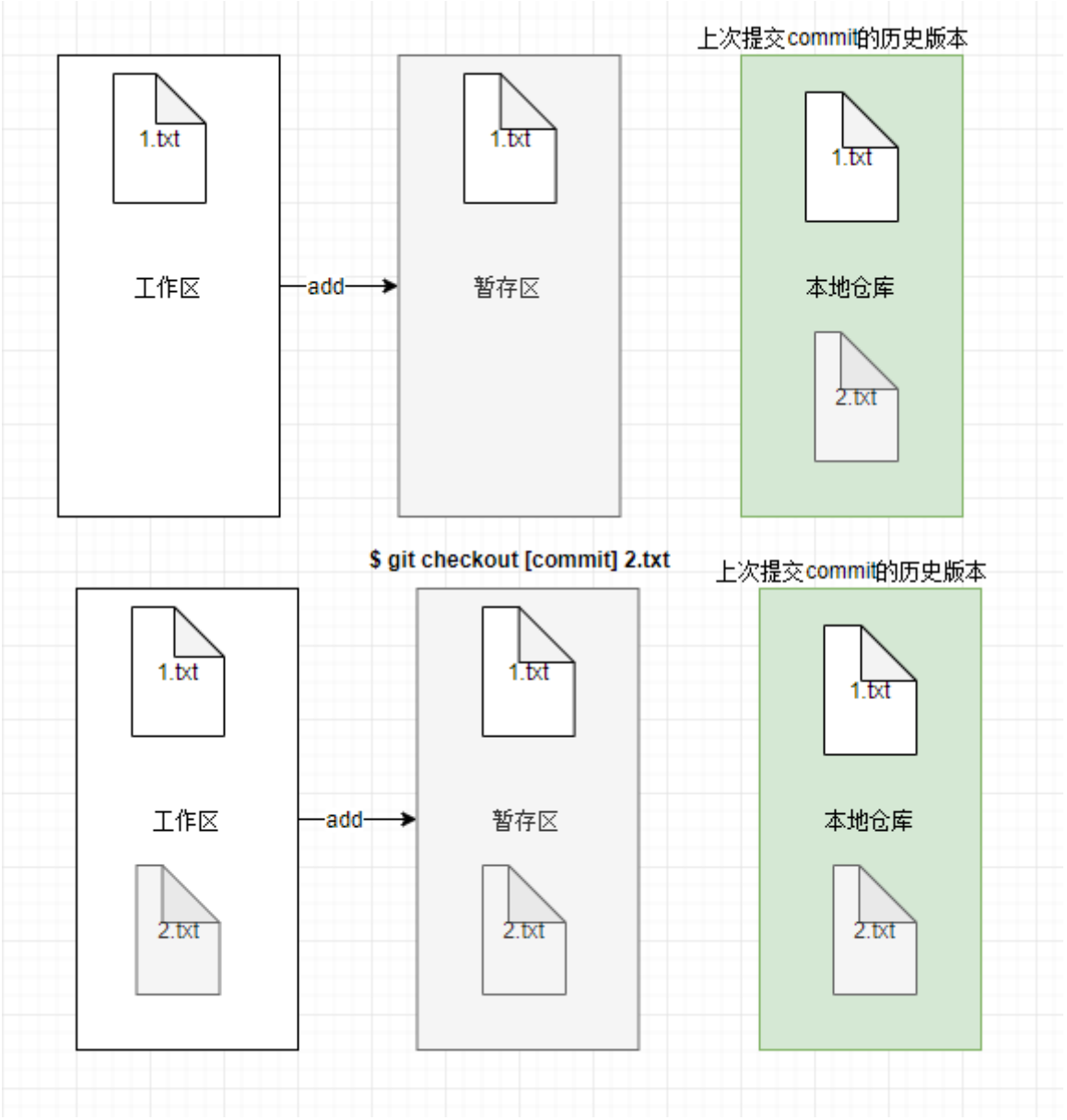


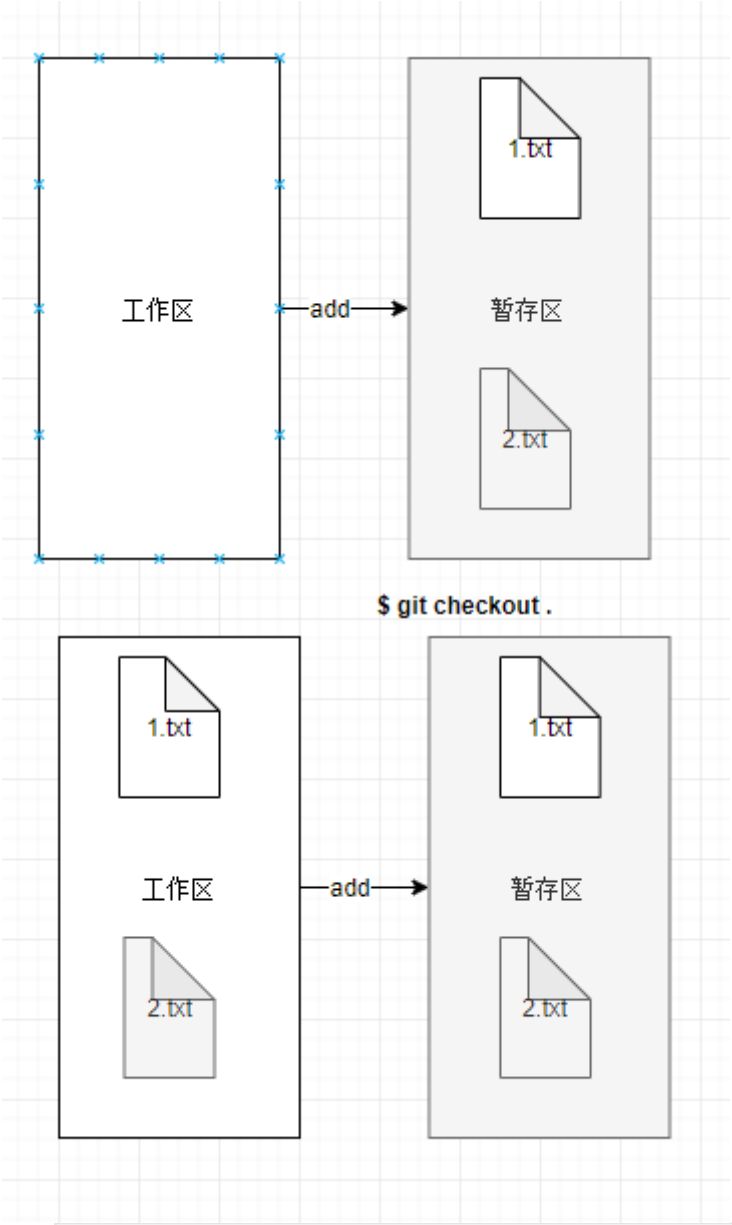

```
//改名文件，并且将这个改名放入暂存区
$ git mv [file-original] [file-renamed]
```

4,恢复文件




```
// 恢复暂存区的指定文件到工作区
$ git checkout [file]
```


上次提交commit的历史版本
1.txt
1.txt
1.txt
工作区
add
暂存区
本地仓库
2.txt
$ git checkout [commit] 2.txt
上次提交commit的历史版本
1.txt
1.txt
1.txt
工作区
add
暂存区
本地仓库
2.txt
2.txt
2.txt




```
// 恢复暂存区的所有文件到工作区
$ git checkout .
```

三、本地代码提交

1，提交到仓库

```
//提交暂存区到仓库区
$ git commit -m [message]

//提交暂存区的指定文件到仓库区
$ git commit [file1] [file2] ... -m [message]

//提交工作区自上次commit之后的变化，直接到仓库区
$ git commit -a
```

```
9
10 //提交时显示所有diff信息
11 $ git commit -v
12
13 //(用于替换本次提交有误的信息)使用一次新的commit，替代上一次提交，如果代码没有任何新变化，则用来改写上一次commit的提交信息
14 $ git commit --amend -m [message]
15
16 //(用于补交部分内容)重做上一次commit，并包括指定文件的新变化
17 $ git commit --amend [file1] [file2] ...
```

2，查看版本历史

```
1 //显示当前分支的版本历史
2 $ git log
3
4 //显示commit历史，以及每次commit发生变更的文件
5 $ git log --stat
6
7 //搜索提交历史，根据关键词
8 $ git log -S [keyword]
9
10 //显示某个commit之后的所有变动，每个commit占据一行
11 $ git log [tag] HEAD --pretty=format:%s
12
13 //显示某个commit之后的所有变动，其"提交说明"必须符合搜索条件
14 $ git log [tag] HEAD --grep feature
15
16 //显示某个文件的版本历史，包括文件改名
17 $ git log --follow [file]
18 $ git whatchanged [file]
19
20 //显示指定文件相关的每一次diff
21 $ git log -p [file]
22
23 //显示过去5次提交
24 $ git log -5 --pretty --oneline
25
26 //显示所有提交过的用户，按提交次数排序
27 $ git shortlog -sn
28
29 //显示指定文件是什么人在什么时间修改过
```

```
$ git blame [file]
```

3，文件差异比较

```
//(比较当前暂存和工作区)显示暂存区和工作区的差异
$ git diff

//（比较当前暂存和上个提交）显示暂存区和上一个commit的差异
$ git diff --cached [file]

//（比较工作区和上个提交）显示工作区与当前分支最新commit之间的差异
$ git diff HEAD

 //显示两次提交之间的差异
 $ git diff [first-branch]...[second-branch]

 //显示今天你写了多少行代码
 $ git diff --shortstat "@{0 day ago}"
```

4，其他查询

```
//显示某次提交的元数据和内容变化
$ git show [commit]

//显示某次提交发生变化的文件
$ git show --name-only [commit]

//显示某次提交时，某个文件的内容
$ git show [commit]:[filename]

 //显示当前分支的最近几次提交
 $ git reflog
```

5，版本回退

```
//（回退到上次提交）重置暂存区与工作区，与上一次commit保持一致
$ git reset --hard HEAD
```

```
3
4 //（回退到某次提交）重置当前分支的HEAD为指定commit，同时重置暂存区和工作区，与指定commit一致
5 $ git reset --hard [某版本号前几位]
6
7 //重置暂存区的指定文件，与上一次commit保持一致，但工作区不变
8 $ git reset [file]
9
10 //重置当前分支的指针为指定commit，同时重置暂存区，但工作区不变
11 $ git reset [commit]
12
13 //重置当前HEAD为指定commit，但保持暂存区和工作区不变
14 $ git reset --keep [commit]
15
16 //新建一个commit，用来撤销指定commit （后者的所有变化都将被前者抵消，并且应用到当前分支）
17 $ git revert [commit]
```