#### 1.1 工作区/暂存区（stage）/版本库（respository）

工作区：就是我们电脑的工程目录，我们在这个工程目录里进行代码文件的编写、修改、删除等操作。

暂存区：我们在工作区内完成的内容通过add命令添加到暂存区内，表示我们需要把这些完成的内容交给git管理。

版本库：又名仓库，可以理解为一个目录，目录内的所有文件都被git管理，每个文件的修改，删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻还可以将文件”还原”。使用commit命令就是将暂存区中的文件（所有add的完成的内容）提交到本机的版本库里。就相当于将文件交给git托管。

这三者代表着代码文件的不同所在：编码人员在工作区内完成编码工作🡪将完成内容add到暂存区🡪最后将所有完成内容commit到版本库，交由git托管。

## 1.2 init/add/commit等基本编辑文件命令

git init :

（1）在所在工程目录执行

（2）创建版本库，表示该目录中的文件及目录会通过git进行管理（即分为工作区、暂存区、仓库）。

git add +文件相对路径:

（1）在所在工程目录执行

（2）将修改后的文件提交到暂存区，待交由git托管。

git commit:

（1）在所在工程目录执行

（2）将暂存区内的所有待托管的文件提交到版本库，交由git托管。

（3）add和commit一般是结合在一起的两步，用于提交修改。也可多次add，然后最后一次提交修改。

git reset –hard HEAD^/~100/版本号：

（1）在所在工程目录执行, ^代表回退到上一个版本，其个数代表回退的版本次数。~100中代表前100个版本，也可直接通过版本号来进行回退（一般这种在当前版本已经回退到前某个版本，又需要回退到当前版本时使用，通过git reflog命令获取版本号）

（2）将版本库里的当前版本进行回退

（3）add和commit一般是结合在一起的两步，用于提交修改。也可多次add，然后最后一次提交修改。

git checkout - filename:

（1）在所在工程目录执行，将工作区中filename文件的内容修改到和版本库（暂存区中没有该文件待提交内容）或暂存区中一致，就是检出的意思。

（2）如果某个文件在工作区中已删除，但在版本库中仍然存在，及删除动作尚未commit的情况下，该命令可以在工作区恢复文件，文件内容和版本库一致。

### 1.3分支

分支概念： Git把所有的提交串成一条时间线，这条时间线就是一个分支。Git初始时默认有一条时间线，这个分支叫主分支，即master分支。HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

分支策略：首先master主分支应该是非常稳定的，也就是用来发布新版本，一般情况下不允许在上面干活，干活一般情况下在新建的dev分支上干活，干完后，比如上要发布，或者说dev分支代码稳定后可以合并到主分支master上来。

创建分支： git branch 分支名 ;创建分支就意味着将当前所在分支工程复制一份到新建的分支中。

切换分支： git checkout 分支名

创建并切换分支：git checkout –b 分支名

查看分支：git branch

合并某分支A到当前分支B中：git merge name；一般来说合并存在于A是由B分支而得的情况下。所有的合并都是合并的版本库里的已托管内容，与工作区无关。

删除分支：git branch –d name

合并解决冲突：类似SVN，修改有冲突文件，解决后再提交到当前分支（add+commit）

Bug分支工作流程：

在开发中，会经常碰到bug问题，那么有了bug就需要修复，在Git中，分支是很强大的，每个bug都可以通过一个临时分支来修复，修复完成后，合并分支，然后将临时的分支删除掉。

（1）git stash ；隐藏当前分支的工作现场，因为很大可能当前分支工作区要修改的内容还未修改完，肯定不能commit，但又需要首先修复主版本的bug，所以将当前分支的工作区保存起来。

（2）切换到主版本分支，创建bug分支，并切换

（3）修改完bug，合并bug分支到主版本

（4）删除bug分支

（5）切换到工作版本分支

（6）git stash pop；恢复的同时删除stash内容。继续工作。

### 1.3远程仓库（github/其他git远程仓库）

可以将Github看作一台管理着成千上万交由git托管的项目的主机，我们可以将自己本地的项目版本库和github主机上的某个项目版本库关联起来，这样就可以拉取或者推送修改。其他人也可申请修改。 github主机上的版本库可以起着中转的作用。即所有人都可以通过github上的版本进行互动，所以叫分布式版本管理。

1：添加远程仓库

我们已经在本地创建了一个Git仓库后，又想在github创建一个Git仓库，并且希望这两个仓库进行远程同步，这样github的仓库可以作为备份，又可以其他人通过该仓库来协作。

（1）首先在github上“create a new respo”，创建一个新的空仓库“test”

（2）在本地仓库目录运行“git remote add origin <https://github.com/YourGithubName/testgit.git>”

表示将本地仓库和远程仓库关联起来

（3）将本地仓库内容推送到远程“git push –u origin master ”. 实际上是把当前分支master推送到远程。由于远程库是空的，我们第一次推送master分支时，加上了 –u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。推送成功后，可以立刻在github页面中看到远程库的内容已经和本地一模一样

（4）从现在起，只要本地作了提交，就可以通过如下命令：

git push origin master

把本地master分支的最新修改推送到github上

2：拉取远程仓库

上面我们了解了先有本地库，后有远程库时候，如何关联远程库。现在我们想，假如远程库有新的内容了，我想克隆到本地来 如何克隆。

（1）在本地目录下直接 “git clone <https://github.com/GithubName/RespoName>”即可。

（2）从远程库克隆时候，实际上Git自动把本地的master分支和远程的master分支对应起来了，并且远程库的默认名称是origin，即已经自动关联起来。

### 1.4 多人协作

**1：推送分支**（将修改推送到远程仓库请求合并）

推送分支就是把该分支上所有本地提交到远程库中，推送时，要指定本地分支，这样，Git就会把该分支推送到远程库对应的远程分支上：

使用命令 git push origin master（push意味推送；origin意味远程仓库的别名，默认为origin；master表示推送的分支名，即将本地的master分支推送到远程的master分支合并）。

如果我们现在要推送到其他分支，比如dev分支上，我们还是那个命令

git push origin dev

那么一般情况下，那些分支要推送呢？

（1）master分支是主分支，因此要时刻与远程同步。

（2）一些修复bug分支不需要推送到远程去，可以先合并到主分支上，然后把主分支master推送到远程去。

**2：抓取分支**

多人协作工作模式一般是这样的：

首先，可以试图用git push origin branch-name推送自己的修改.

如果推送失败，则因为远程分支比你的本地更新早，需要先用git pull试图合并。

如果合并有冲突，则需要解决冲突，并在本地提交。再用git push origin branch-name推送。