# Git的使用手册

## git基础命令

**1、git init** 将仓库初始化，在项目目录中将会有一个.git的文件出现，这个文件就是git的版本库，这个是隐藏文件

**2、git diff “文件名”** 在修改完文件时，使用这个命令可以查看具体修改了什么内容，以免在修改完文件后忘记修改的内容，在添加之前使用

**3、git add 文件名** 向git暂存区添加一个文件，以便于以后提交

**4、git commit -m “提交描述”** 将暂存区中添加的文件提交到仓库，一次可以提交多个文件

**5、git status** 查看文件的状态，可以查看有哪些文件可以添加，那些可以提交

**6、git log** 可以查看提交日志，可以看到提交的信息（先提交的在前面）：包括版本的id、提交人的信息、提交日期、提交信息，以便于版本管理

**7、git log --pretty=oneline** 简略的输出提交日志，在提交次数很多时使用比较方便，看起来比较清晰

**8、git reset --hard HEAD^** 回退到上一个版本，HEAD表示的是当前版本，HEAD^表示上一个版本，HEAD^^表示上上一个版本，HEAD~100表示前100个版本，也可以将HEAD^换成版本id进行版本还原，版本id可以使用git log命令查看

**9、git reflog** 有时候在回退版本后，使用git log命令无法查看新版本的信息，则可以使用此命令查看所有的版本信息和版本操作信息

**10、git config –global user.name “你的名字”** 设置全局用户名

**11、git config –global user.email “你的邮箱”** 设置全局用户邮箱

上面两条命令主要是为了能让别人看出来是谁提交了什么内容，方便管理

**12、git checkout -- 文件名** 将对文件的修改全部撤销，有两种情况：

- 修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；
- 已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

**13、git checkout 分支名** 用来切换分支

**14、git branch 分支名** 用来创建一个分支

**15、git branch** 用来查看当前存在的分支

**16、git branch -d 分支名** 删除一个分支

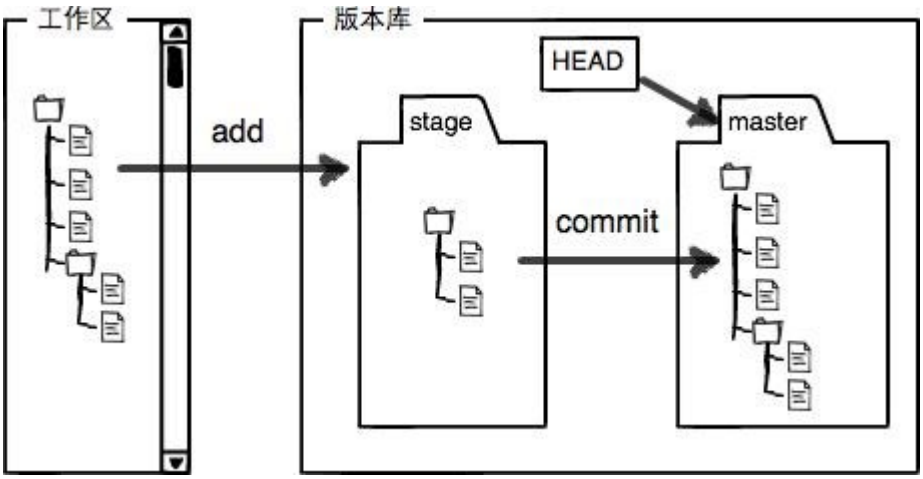
**17、git pull origin 分支名** 拉取远程仓库本地

**18、git push origin 分支名** 推送本地到远程仓库

**19、git tag 版本号** 打标签使用

## 工作区和暂存区

**接下来介绍工作区和暂存区：**



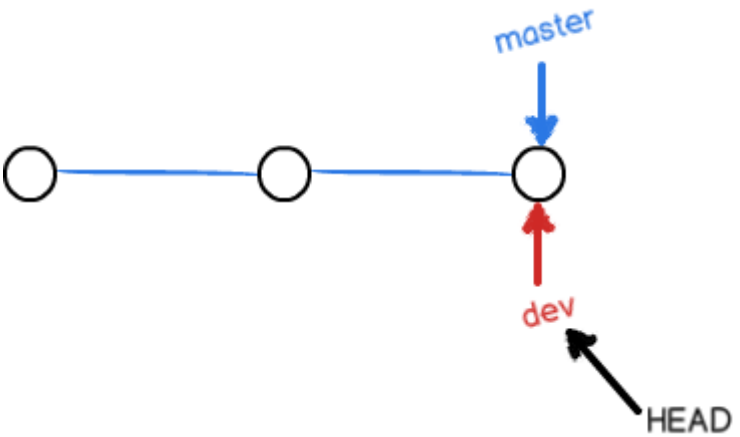

工作区就是你`git init`那个目录，不包括.git文件，统称为工作区。

.git文件就叫做版本库，版本库中的stage就叫做暂存区，使用`git add`时就是将文件的修改加入到了暂存区，使用`git commit`时就是将暂存区中的文件修改提交到当前分支中，`git init`时默认创建一个master分支，也叫作主分支，HEAD指向当前分支，HEAD和分支后面介绍。

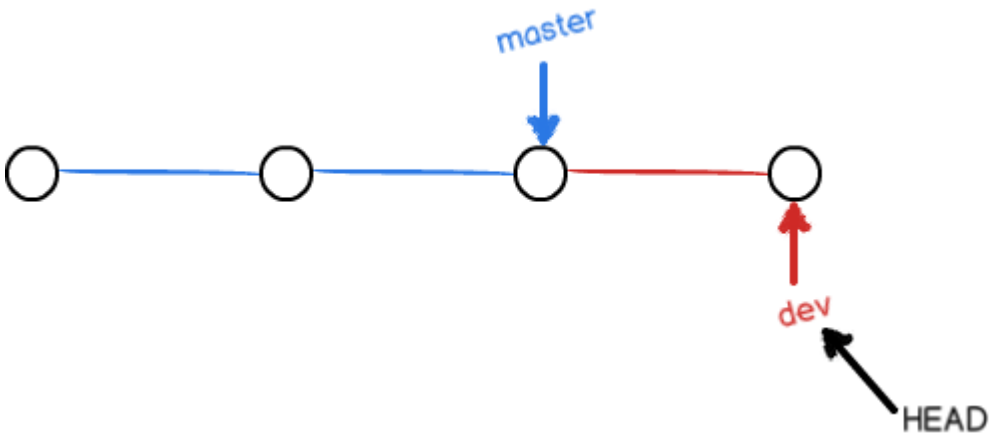
## 分支

**分支在实际中的作用**：假设你准备开发一个新功能，但是需要两周才能完成，第一周你写了50%的代码，如果立刻提交，由于代码还没写完，不完整的代码库会导致别人不能干活了。如果等代码全部写完再一次提交，又存在丢失每天进度的巨大风险。现在有了分支，就不用怕了。你创建了一个属于你自己的分支，别人看不到，还继续在原来的分支上正常工作，而你在自己的分支上干活，想提交就提交，直到开发完毕后，再一次性合并到原来的分支上，这样，既安全，又不影响别人工作。

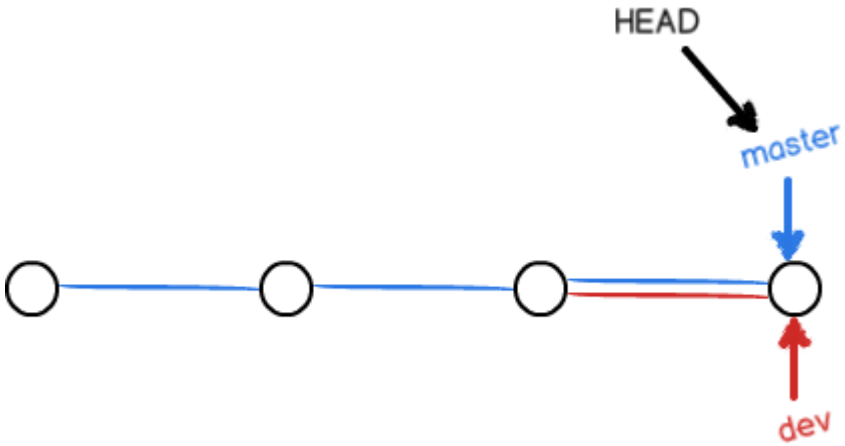
**分支实际上是一个指针，初始为master，指向最新的提交，HEAD指向当前的分支，dev表示新建一个dev分支：**




**如果在div分支上有了新提交，如下图，红线就表示div分支新提交的内容：**

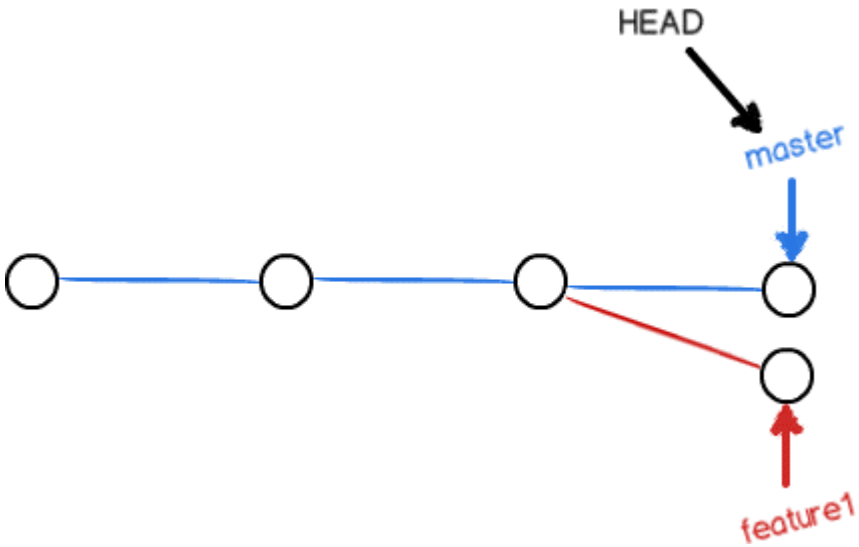



将master分支和dev分支合并，如下图：

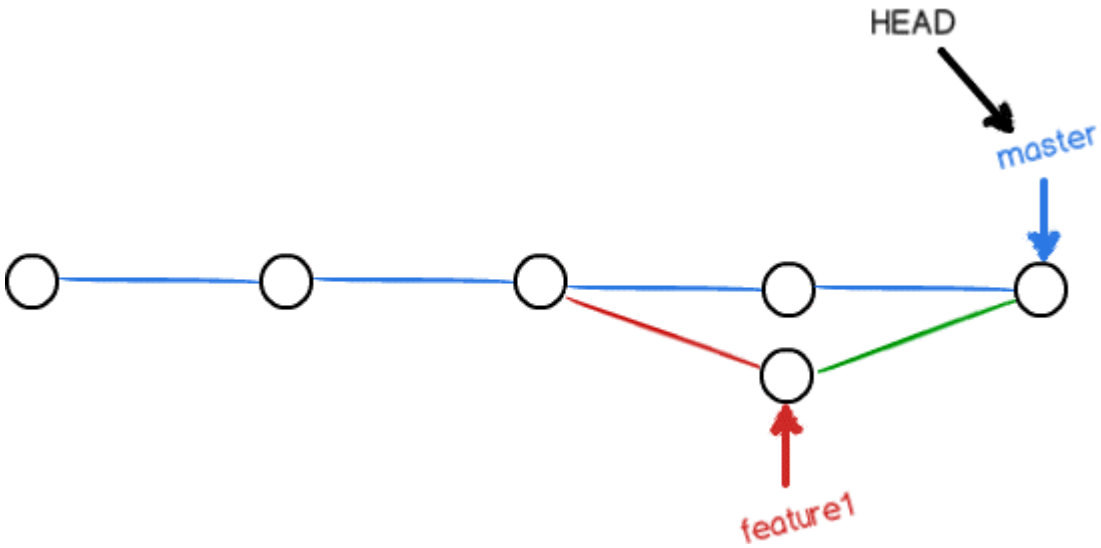



## 分支合并冲突

关于分支合并时的冲突，假如新建了一个feature1分支，在feature1和master上同时都对readme.txt进行了修改，并依次进行了add和commit，如下图：



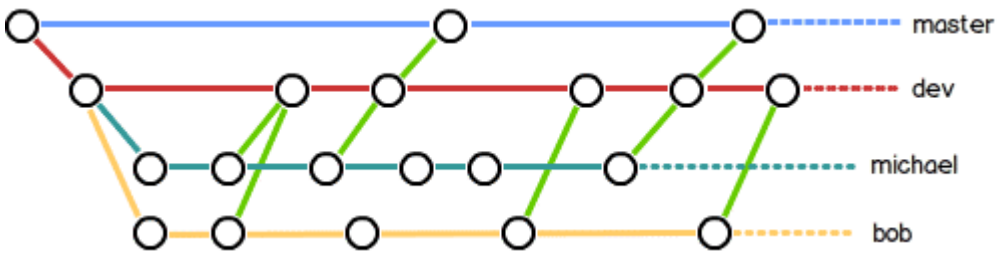

这时候在合并时就会出现冲突，自动合并失败，可以通过git status进行查看，我们需要手动合并，打开冲突的文件，Git用 `<<<<<<<` ， `=======` ， `>>>>>>>` 标记出不同分支的内容，删除不需要的内容，再进行add和commit就变成了下面的：




**这时master分支就将dev中修改的提交合并过来了，dev中readme.txt还是原来的没有变。**

## 分支策略

**在实际开发中，分支策略应该按照master分支为稳定分支，用来发布新版本，不在上面开发，新建一个dev分支当做开发分支，而且每个人都有自己的分支，往dev上合并就行了，发布时将dev和master合并，如下图：**




**待续……**