**总结一下ubuntu下github常用的命令，设置部分跳过，假设repository的名字叫hello-world：**

**1.创建一个新的repository：**

**先在github上创建并写好相关名字，描述。**

**$cd ~/hello-world //到hello-world目录**

**$git init //初始化**

**$git add . //把所有文件加入到索引（不想把所有文件加入，可以用gitignore或add 具体文件)**

**$git commit //提交到本地仓库，然后会填写更新日志( -m “更新日志”也可)**

**$git remote add origin git@github.com:WadeLeng/hello-world.git //增加到remote**

**$git push origin master //push到github上**

**2.更新项目（新加了文件）：**

**$cd ~/hello-world**

**$git add . //这样可以自动判断新加了哪些文件，或者手动加入文件名字**

**$git commit //提交到本地仓库**

**$git push origin master //不是新创建的，不用再add 到remote上了**

**3.更新项目（没新加文件，只有删除或者修改文件）：**

**$cd ~/hello-world**

**$git commit -a //记录删除或修改了哪些文件**

**$git push origin master //提交到github**

**4.忽略一些文件，比如\*.o等:**

**$cd ~/hello-world**

**$vim .gitignore //把文件类型加入到.gitignore中，保存**

**然后就可以git add . 能自动过滤这种文件**

**5.clone代码到本地：**

**$git clone git@github.com:WadeLeng/hello-world.git**

**假如本地已经存在了代码，而仓库里有更新，把更改的合并到本地的项目：**

**$git fetch origin //获取远程更新**

**$git merge origin/master //把更新的内容合并到本地分支**

**6.撤销**

**$git reset**

**7.删除**

**$git rm \* // 不是用rm**

**//——————————常见错误———————————–**

**1.$ git remote add origin git@github.com:WadeLeng/hello-world.git**

**错误提示：fatal: remote origin already exists.**

**解决办法：$ git remote rm origin**

**然后在执行：$ git remote add origin git@github.com:WadeLeng/hello-world.git 就不会报错误了**

**2. $ git push origin master**

**错误提示：error:failed to push som refs to**

**解决办法：$ git pull origin master //先把远程服务器github上面的文件拉先来，再push 上去。**

**//—————————————**

**初始化一个Git仓库**，使用git init命令。

**添加文件到Git仓库**，分两步：

第一步，使用命令git add <file>，注意，可反复多次使用，添加多个文件；

第二步，使用命令git commit，完成。

**要随时掌握工作区的状态**，使用git status命令。

如果git status告诉你有文件被修改过，用git diff可以查看修改内容。

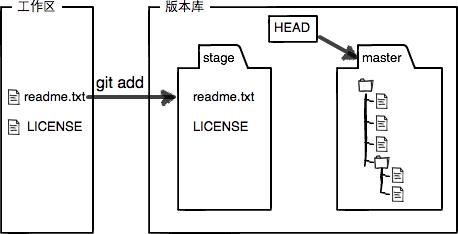
**版本回退**

HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id。上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。

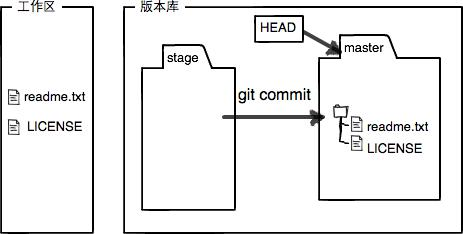
穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。

要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。

Git管理的是修改，当你用git add命令后，在工作区的第一次修改被放入暂存区，准备提交



git commit只负责把暂存区的修改提交



提交后，用git diff HEAD -- readme.txt命令可以查看工作区和版本库里面最新版本的区别

**撤销修改**

命令git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销（在add之前）

命令git reset HEAD file可以把暂存区的修改撤销掉（unstage），重新放回工作区（在commit之前），再用命令git checkout -- readme.txt丢弃工作区的修改

**删除该文件**，那就用命令git rm删掉，并且git commit

总结：提交代码的更改一共分2个阶段

1).从工作目录，提交到stage。

2).从stage提交到master。

从工作目录提交到stage，需要用add或者rm命令，只提交到stage，而没有提交到master，是不会自动同步到master的。

从stage提交到master用commit命令。

退回也是要分两步，一个是从master退回到stage，然后再从stage退回到工作目录。

对于还没有提交到stage的，可以从stage用checkout命令退回，这一步会取stage中的文件状态，覆盖掉工作目录中文件的状态，跟master完全没关系。

对于已经到达stage的，想把state中的文件状态用master中的覆盖掉，就用reset命令，这样就把stage中修改用master的状态覆盖掉了，完全跟工作目录没关系

**要关联一个远程库**，使用命令git remote add origin git@server-name:path/repo-name.git；如git remote add origin [git@github.com:GitHdu/js.git](mailto:git@github.com:GitHdu/js.git)

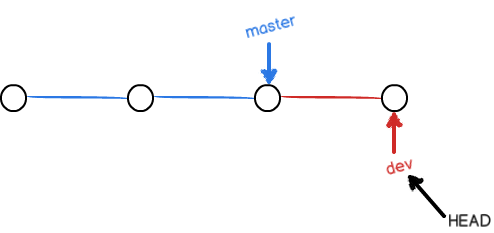
关联后，使用命令git push -u origin master第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

去除关联git remote rm origin

从**远程库克隆**git clone [git@github.com:GitHdu/js.git](mailto:git@github.com:GitHdu/js.git)

创建dev**分支**，然后切换到dev分支：git checkout -b dev，git checkout命令加上-b参数表示创建并切换，然后，用git branch命令查看当前分支

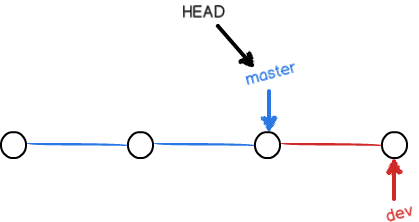


然后，我们就可以在dev分支上正常提交，比如对readme.txt做个修改，然后提交：

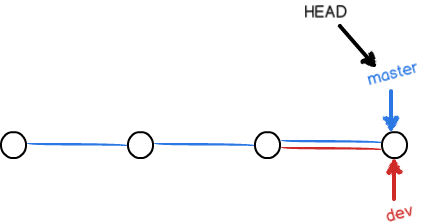
git add readme.txt

git commit -m "branch test"；

现在，dev分支的工作完成，我们就可以切换回master分支：git checkout master



现在，我们把dev分支的工作成果合并到master分支上：git merge dev



合并完成后，就可以放心地删除dev分支了：git branch -d dev

删除后，查看branch，就只剩下master分支了：git branch

