1. Git初次使用时设置用户名和邮箱：

配置用户名：git config --global user.name “gaoyuhua”

配置邮箱：git config --global user.emaill [gaoyuhua@qq.com](mailto:gaoyuhua@qq.com)

删除错误配置：git config --global –unset “ “

查看配置：git config –list

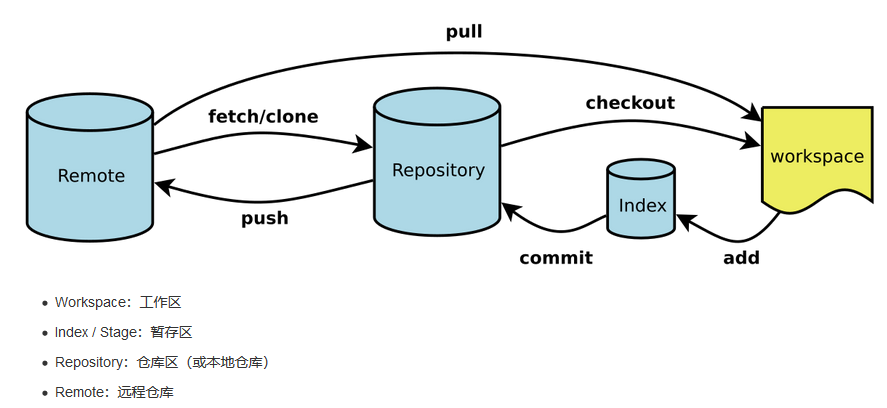
1. 初始化一个空的仓库：git init，会生成一个.git的隐藏文件夹
2. 查看当前目录文件状态：git status
3. 将文件添加到暂存区：git add .(批量添加) git add test.txt
4. 将文件添加到仓储中：git commit –m”备注” git commit –m –c(不添加备注)，直接进行添加提交：git commit –a –m”备注”
5. 查看日志：git log , git long --oneline git log --graph(以图形化查看)
6. 忽略文件：新建一个.gitignore文件
7. 版本回退：git reset --hard Head ^^(回到最近一次提交的版本,若无^就是退回上一次，有^^2个，则退回到第二次)，只是退回，并不会删除
8. 查看有多少分支：git branch 创建分支：git branch dev(创建了一个名为dev的分支) 切换到分支dev：git checkout dev 创建并指定分支：git checkout -b dev
9. 合并git分支,在主分支下将dev合并：git merge dev 在主分支下删除dev分支：git branch –d dev
10. 查看当前的远程仓库：git remote 列出所有的本地分支：git branch 列出所有的远程分支：git branch –r 列出所有的本地分支和远程分支：git branch

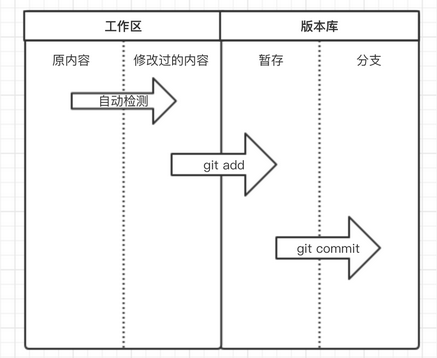
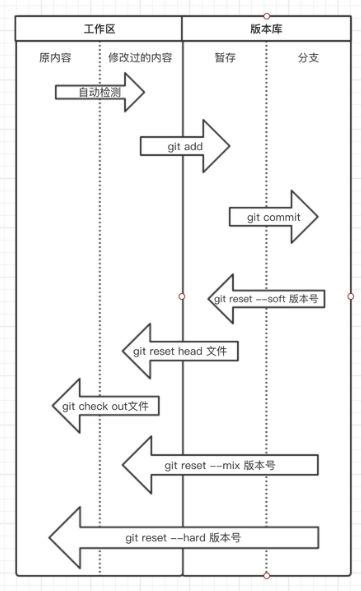
新建一个分支，但是仍然留在当前分支：git branch dev 新建一个分支并切换到该分支：git checkout -b dev 切换到上一个分支：git checkout -

合并指定分支到当前分支：git merge dev 删除分支：git branch -d dev

删除远程分支：git push origin --delete name

1. 远程分支与本地分支：git克隆是会自动建立自己的本地分支master和远程分支origin/master，并且指向origin上的master分支，



1. 
2. 

**PRO GIT笔记**

1. Git分布式版本控制系统，客户端把代码仓库完整地镜像下来，任何一处协同工作用的服务器发生故障，事后都可以用任何一个镜像出来的本地仓库恢复，因为每次克隆，都是一次对代码仓库的完整备份。
2. 三种状态：已提交（commit），已修改（modified），已暂存（staged）
3. Q:git仓库是指本地仓库还是远程仓库
4. A:git init初始化本地仓库

B:git “mygit\_repo” >bin2.txt在mygit\_repo下加入一个bin2.txt文档

C:git add bin2.txt加入暂存区

D:git commit -m”备注”

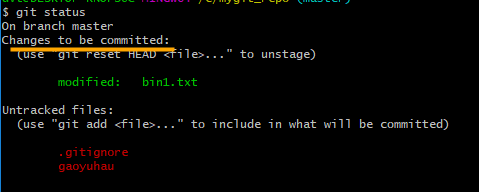
1. 克隆远程仓库

A:git clone https://github.com/libgit2/libgit2(地址)，且创建的名为libgit 的目录，若是自定义本地仓库名字,如mylibgit:

git clone <https://github.com/libgit2/libgit2> mylibgit

1. A:git status查询状态

B:add后，在changes to be committed行下面是已暂存的状态，添加内容到下一次提交中



1. ？？？.gitignore
2. A:git diff可以查看未add的文件修改的内容

B:git diff --cached或者git diff --staged可以查看已经add后进入已暂存的文件修改内容

1. 使用git commit –a –m”备注”可以直接添加提交
2. 使用rm bin1.txt只是从工作目录中删除，使用git status还存在，再次执行git rm bin1.txt就不再纳入版本控制了，且会被记录
3. 移动文件，也是对git的文件重命名，将bin2.txt改名为bin3.txt

A:git mv bin2.txt bin3.txt

B:git commit –m”rename”

1. 日志是最新更新的在上面，

A:git log查看全部日志

B:git log -p -2显示2次的变化

C:git log –stat显示简略统计信息

1. 撤销操作：

A:git commit --amend提交完后发现几个文件没有添加或备注写错，重新提交

B1:git commit –m”initial“

B2:git add forgotten\_file

B3:git commit --amend

提交后发现忘记了暂存某些需要的修改，执行完后第二次提交会将之前不完整的提交覆盖

1. A:git reset HEAD test1.txt可以将已经add的文件返回add前状态，即已经修改但是没有add的状态

B:git checkout – test1.txt可以撤销文件的修改，但这个命令有风险，撤销后不能找回

1. A:git remote查看远程仓库

B:git remote -v查看远程仓库和地址

C:git remote add mingzi URL添加一个新的名字为mingzi的远程仓库

D:git branch –r查看所有远程分支

E:git branch查看所有本地分支

1. A:git fetch [remote\_name]从远程获取数据，须手工合并

B:git pull [remote\_name]从远程抓取并合并

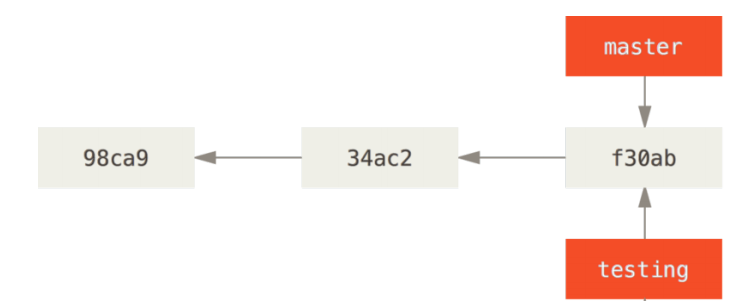
C:git clone会自动设置本地master分支跟踪克隆的远程仓库的master分支，运行git pull会从最初克隆的服务器上抓取数据并合并到当前分支

1. 推送到远程仓库git push origin master将master分支推送到origin服务器：条件是有权限且本地分支是克隆最新的
2. A:git remote show origin显示远程仓库的URL和分支信息
3. Git设置缩写，如将checkout简写co：git config --global alias.co checkout

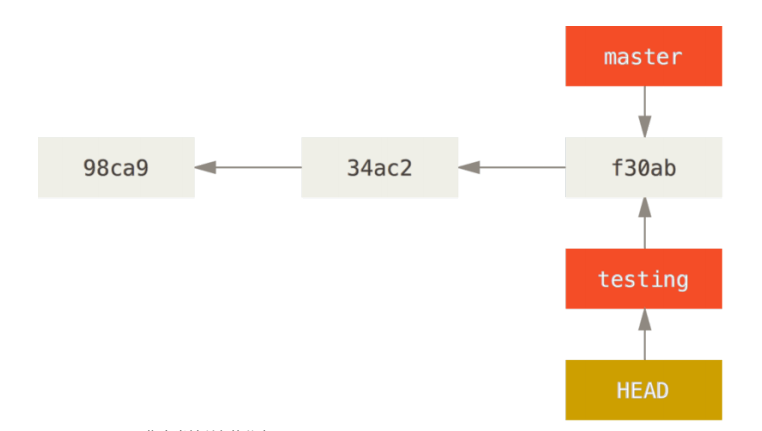
**Important:分支**

**PART1:本地分支**

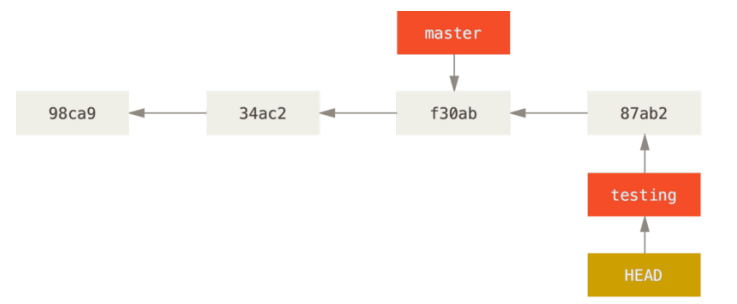
1.新建分支：git branch testing



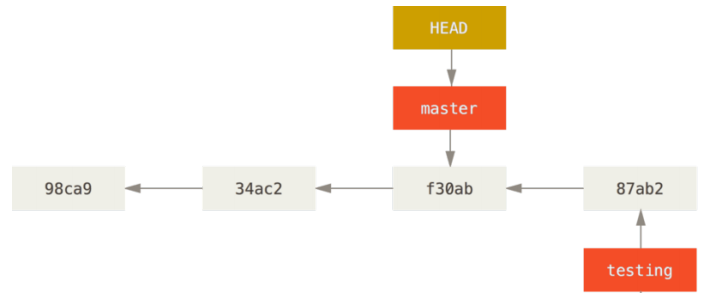
1. 分支切换：git checkout gaoyuhua



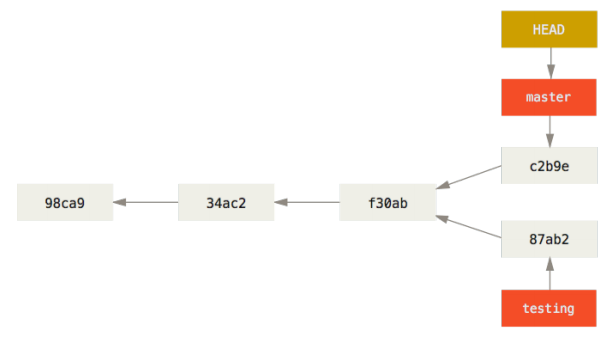
1. 删除分支：git branch -D testing
2. 建立文件（在testing分支下）并修改：vim test.rb 保存并提交，会发现指针随testing向前移动



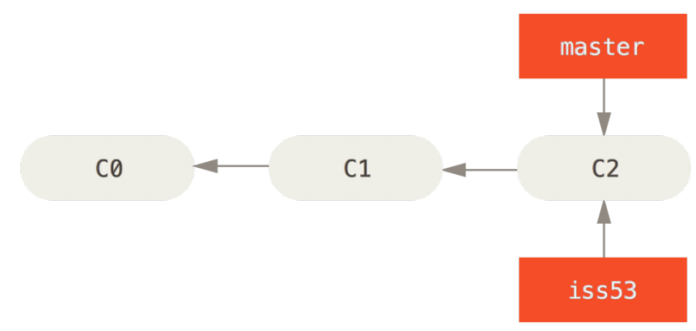
1. 再次切回master分支，会发现指针返回之前一步，回到之前旧的版本



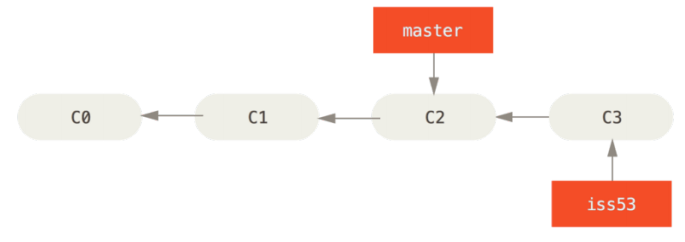
1. Tip：分支切换会改变工作目录的文件，不同的分支下有不同的进度。
2. 建立文件保存提交（在master本地主分支下）：vim test.rb



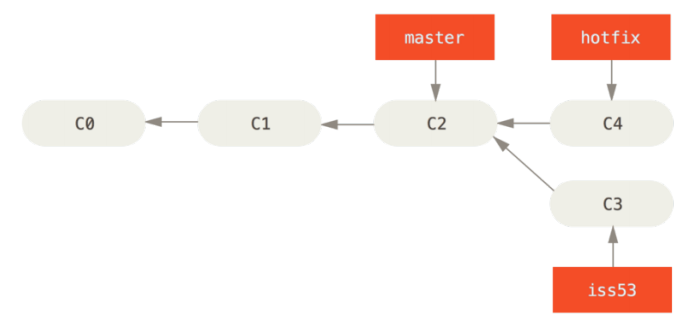
1. 新建分支并切换到分支：git checkout –b iss53////git branch iss53;git checkout iss53



1. 在分支iss53上建立新文件并提交，iss53的版本到达C3



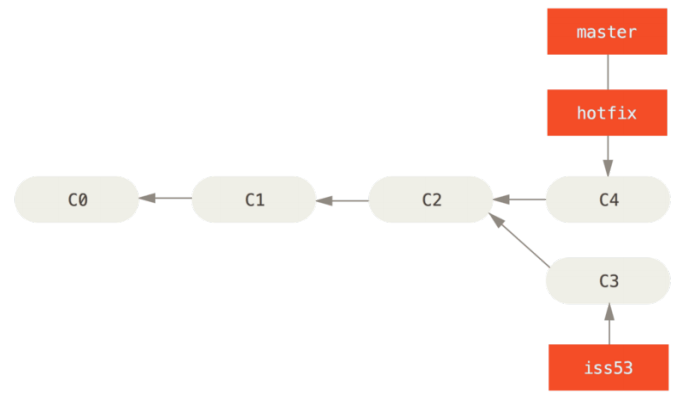
1. 若此时有个bug在之前的版本，那么可以在master分支再次建立分支hotfix，前提是master分支上保持干净的状态，在 hotfix上修改bug，再合并，此时在hotfix上建立新的文档index.html，此时到达版本C4



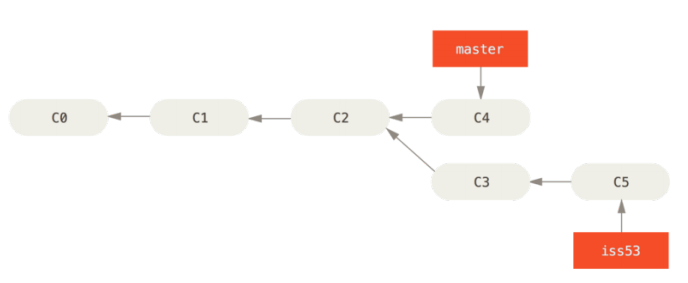
1. 当你确定hotfix上的修改是正确的，可以将hotfix合并到master上

A:切回master分支：git checkout master

B:合并分支hotfix：git merge hotfix



1. 此时hotfix的责任已经完成，返回正常工作，此时需要将hotfix删除：git branch -d hotfix,切换到你工作的分支iss53：git checkout iss53，并修改提交文件index.html,此时iss53到达C5的版本



1. 此时iss53完成，需要将iss53合并到master分支,若不再需要iss53分支可以进行删除

A:git checkout master

B:git mergr iss53

C:git branch -d iss53

1. A:git branch:可以查看当前所有本地分支列表

B:git branch –v:查看每一个分支最后一次提交

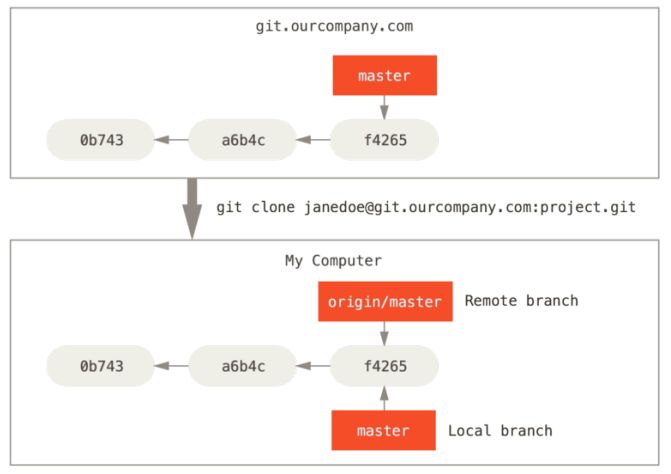
C:git branch --merged:查看哪些分支已经合并到当前分支，此列表下不带\*可以直接删除

D:git branch --no-merged:显示还未进行合并的分支

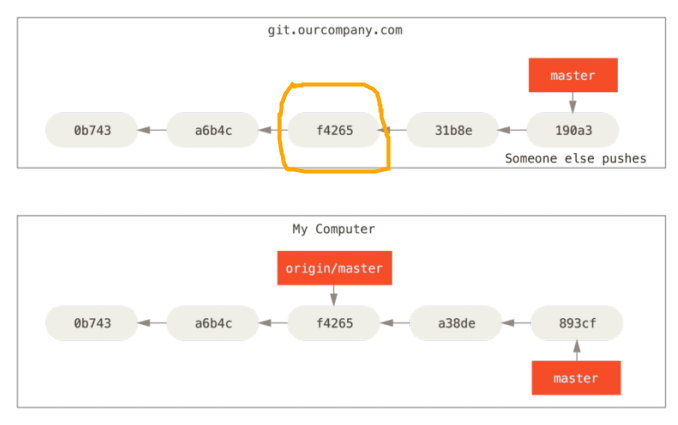
E:若未合并的分支使用git branch -d testing会报错，若要强制删除可使用git branch -D testing

PART2:远程分支

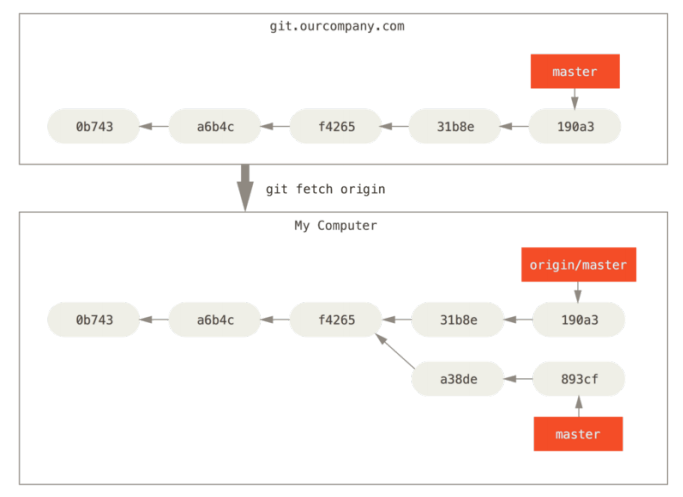
1. 远程仓库名字origin与分支名字master一样，无特殊含义，master是运行git init默认起的分支名字，origin是你运行git clone是默认的远程仓库名字，远程github上有公司的服务器，名字为git.ourcompany.com，在服务器上有版本库，且主分支是master，最新版本是f4265，使用git clone命令copy到计算机上一个名为origin/master的远程分支，同时计算机自动建立一个本地的分支master，且双方都指向版本f4265



1. 若在一段时间内，有人推送提交到git.ourcompany.com，并且更新了它的master分支，你如果再次提交，会向不同的方向前进



1. 使用git fetch origin可以将抓取本地没有的数据并且进行更新



1. 若有自己小组的git服务器，则单独clone下来

A: git remote add teamone https://github.com/gaoyuh/learngitrepo\_team1.git

表示将远程服务器learngitrepo\_team1克隆到本机，且命名为teamone

B:git remote查询远程的仓库

C:git checkout testing将本地分支切换到testing

D:git push teamone testing将testing的内容推送到远程仓库teamone中

1. 删除远程分支：git push origin --delete teamone

服务器上的GIT :

1. A:cd ~/.ssh ls 可以查看是否有公钥私钥，其中id\_rsa是私钥，id\_rsa.pub是公钥

B:ssh-ketgen若无公钥私钥文件，可确认密钥的存储位置

1. 首先，远程仓库为空时，建立一个readme文件，克隆远程仓库并将远程仓库取名为origin

A:echo “#team-“ >>README.md

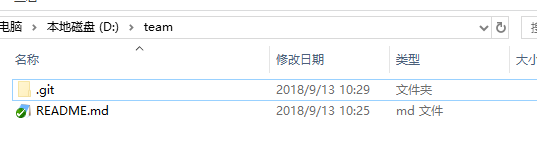
B:git init

C:git add README.md

D:git commit –m”first commit”

E:git remote add origin https://github.com/teamdvt1/team-.git

F:git branch本地建立了一个本地的主分支，名为master



1. 建立本地分支teting,并切换到分支

A:git branch teting

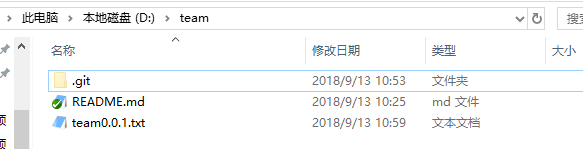
B:git checkout teting

C:或者直接建立切换：git checkout -b teting

因为是直接在master分支建立的，所以和master分支的内容相同

1. 在teting分支新建文件并提交

A:echo “team” >team0.0.1.txt

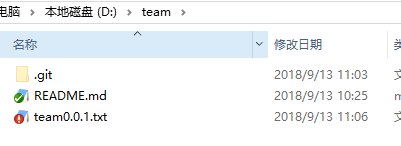
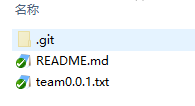


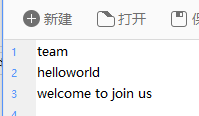
B:git add team0.0.1.txt

C:git commit –m”add team0.0.1.txt”



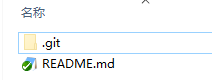
D:增加team0.0.1文件的内容，vim team0.0.1.txt git add team0.0.1.txt git commit –m”add helloworld”



1. 此时切换回本地master分支，此时master分支并不存在team0.0.1文档，所以将teting分支的内容合并到本地主分支master上

A:git checkout master首先要切换到master分支

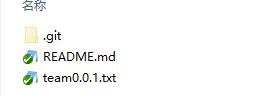
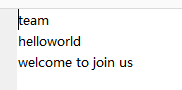


B:git mergr teting将teting分支的内容合并到master分支上

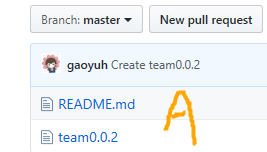
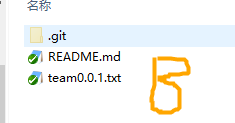
C:git branch -d teting若不再需要teting分支可以删除

Tips：若teting分支没有合并是使用c命令是会提示❌，可以选择合并，再次提交，也可使用git branch –D teting强制删除

此时master分支内容，team0.0.1文件内容如下：

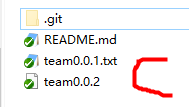
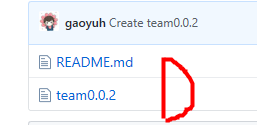
 

1. 将本地没有的文件，存在与远程仓库的文件抓取下来，此时远程仓库内容为A,本地仓库为B

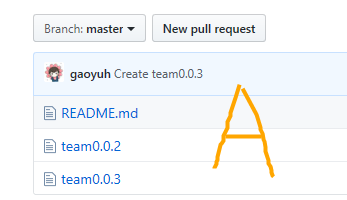
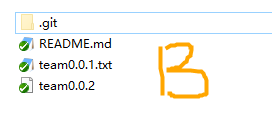
 

运行git fetch origin可以将本地没有的而远程存在的目录下载下来，但是并不会出现在team文件夹中，因为这仅仅是将远程的放到本地为远程仓库设置的独特的位置，你所能看到的仅仅是你在gitbush中设置的分支所在的目录下，若让它出现在你的分支下，还要将他们进行合并。

使用git pull origin master将远程的仓库拉下来，此时本地master分支是C,远程仓库是D

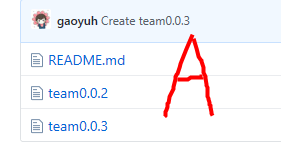
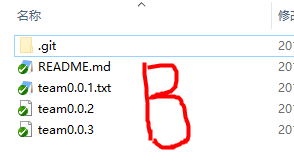
* 1. 使用另一种方法进行抓取文件，远程上新建一个team0.0.3文件，此时远程为A，本地为B

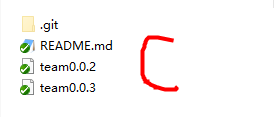
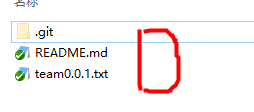
 

A:git fetch origin master:dev表示将远程master分支的内容（本地master中不存在）抓取下来，并把这个内容放在本地重新建立一个名为dev的分支中

B:git diff dev可以比较dev中的内容与主分支master的区别（前提是在master中进行比较）

C：git merge dev将dev分支的内容与master进行合并，此时远程分支为A，本地master分支为B，本地dev分支为C，本地teting分支为D

D:本地的工作大致已经完成，此时可以将dev分支进行删除git branch –d dev

使用git branch可以显示本地分支，git remote可以显示远程分支

* 1. 在远程仓库中建立分支pmo\_team，并在github里抓取下来

A: git fetch origin pmo\_team:dev将远程分支pmo\_team抓取下来放到新建的本地分支dev中。

B:git checkout dev切换到dev分支

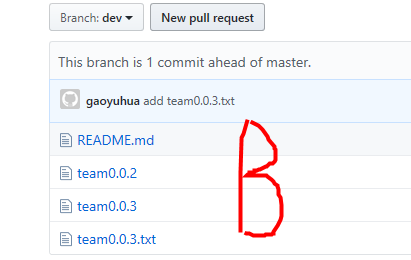
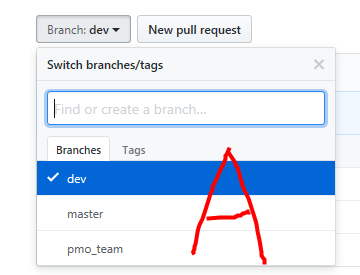
C:vim team0.0.3.txt在dev中新增文件team0.0.3.txt

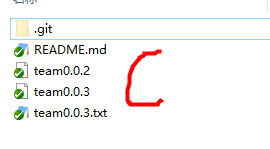
D:git add team0.0.3.txt

E:git commit –m”add team0.0.3.txt”提交

F:git push origin dev在远程仓库建立一个名为dev的分支

Tips：若在远程建立一个pmo\_team1的分支，可以在本地抓取时也名为pmo\_team1，此时若修改本地pmo\_team1，可以通过push命令直接将本地的pmo\_team1文件直接推送到pmo\_team1中，若远程分支上没有与之对应的分支名字，则会重新建立一个分支，与之对应，进行推送。此时远程分支为A,B，本地dev分支为C





1. A:git branch –r查看远程所有仓库
2. 添加**忽略文件**

A:在本地新建文件team0.0.5.txt,team0.0.6.txt,team0.0.7或者echo “team” >team0.0.5.txt

B:git status查看文件状态

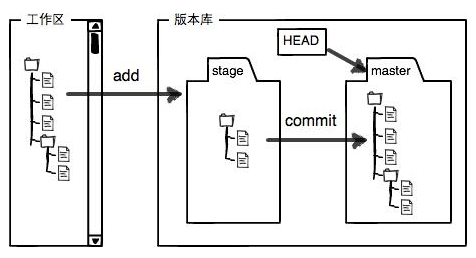
C：vim .gitignore新建文件.gitignore并编辑，将team0.0.5.txt,team0.0.6.txt,team0.0.7名字编辑到,gitignore文件中，则该三个文件会被忽略，再将.gitignore文件添加提交即可

1. **版本退回**

A:git reset --hard HEAD^返回上一版本

B:git reset --hard HEAD^^返回上2个版本

C:git reset --hard HEAD~100返回上100个版本

1. **Vim编辑是卡顿：**可能是vim锁屏了，VIM组合键 Ctrl+S（锁屏） 和 Ctrl+Q（解锁）
2. 
3. 删除文件

A:rm team1.txt在本地工作目录中删除

B:git rm team1.txt在版本库里删除