### 远程分支

### 本地分支

***正常的数据推送 和 拉取步骤***

***1. 确保本地分支已经跟踪了远程跟踪分支***

***2. 拉取数据 : git pull***

***3. 上传数据: git push***

一个本地分支去跟踪一个远程跟踪分支

1. 当克隆的时候 会自动生成一个master本地分支(已经跟踪了对应的远程跟踪分支)

2. 在新建其他分支时 可以指定想要跟踪的远程跟踪分支

git checkout -b 本地分支名 远程跟踪分支名

git checkout --track 远程跟踪分支名

3. 将一个已经存在的本地分支 改成 一个跟踪分支

git branch -u 远程跟踪分支名

### 远程跟踪分支

*远程跟踪分支*是远程分支状态的引用。它们是你不能移动的本地分支。当你做任何网络通信操作时，它们会自动移动。它们以 (remote)/(branch) 形式命名

当克隆一个仓库时 ，它通常会自动地创建一个跟踪 origin/master 的 master 分支。

###推送其他分支

1.公开分享一个分支时，需要将其推送到有写入权限的远程仓库上。 本地的分支并不会自动与远程仓库同步 - 你必须显式地推送想要分享的分支。

如果希望和别人一起在名为 serverfix 的分支上工作，你可以像推送第一个分支那样推送它。

命令：git push origin serverfix

2.这里有些工作被简化了。 Git自动将serverfix 分支名字展开为 refs/heads/serverfix:refs/heads/serverfix 。你也可以运行

命令：git push origin serverfix:serverfix，它会做同样的事。

相当于它说，“推送本地的 serverfix 分支，将其作为远程仓库的 serverfix 分支”

3.如果并不想让远程仓库上的分支叫做 serverfix，可以运行以下命令将本地的 serverfix 分支推送到远程仓库上的 awesomebranch 分支。

命令：git push origin serverfix:awesomebranch

4. git fetch origin，不指定的话默认获取master分支

5. 可以运行 git merge origin/serverfix 将这些工作合并到当前所在的分支。

命令：git merge origin/serverfix (其他协作者)

6.如果想要在自己的 serverfix 分支上工作，可以将其建立在远程跟踪分支之上

命令：git checkout -b serverfix origin/serverfix （其他协作者）

###跟踪分支

1. 从一个远程跟踪分支（origin/master）检出一个本地分支会 自动创建一个叫做 “跟踪分支（有时候也叫做 ” “上游分支” ：master）。 只有主分支 并且 克隆时才会自动建跟踪分支.

2.跟踪分支是与远程分支有直接关系的本地分支。如果在一个跟踪分支上输入 git pull，Git能自动地识别去哪个服务器上抓取、合并到哪个分支。

3.如果你愿意的话可以设置其他的跟踪分支，或者不跟踪 master 分支。

命令：git checkout -b [branch] [remotename]/[branch]

命令：git checkout -b serverfix origin/serverfix

命令：git checkout --track origin/serverfix Git 提供了 --track 快捷方式

4.如果想要将本地分支与远程分支设置为不同名字

命令：git checkout -b sf origin/serverfix

5. 设置已有的本地分支跟踪一个刚刚拉取下来的远程分支，或者想要修改正在跟踪的跟踪分支，你可以在任意时间使用-u选项运行git branch来显式地设置

命令“git branch -u origin/serverfix （--set-upstream-to）

6. 查看设置的所有跟踪分支

命令：git branch -vv

### 冲突

1. git本地操作会不会有冲突?

典型合并的时候

2.git远程协作的时候 会不会有冲突?

push

pull

###删除远程分支

git push origin --delete serverfix

//删除远程分支

git remote prune origin --dry-run

//列出仍在远程跟踪但是远程已被删除的无用分支

git remote prune origin

// 清除上面命令列出来的远程跟踪

###pull request 流程

基本流程：

1. 从 master 分支中创建一个新分支 （自己 fork 的项目）

2. 提交一些修改来改进项目 （自己 fork 的项目）

3. 将这个分支推送到 GitHub 上 （自己 fork 的项目）

4. 创建一个合并请求

5. 讨论，根据实际情况继续修改

6. 项目的拥有者合并或关闭你的合并请求