git log --graph --pretty=oneline --abbrev-commit

可以查看分支合并情况

# 创建版本库

（1）

mkdir learngit //创建一个空的文件夹。

cd learngit //进入到这个文件夹。

Pwd // 显示这个文件夹的绝对路径？

（2）

创建git仓库

Git init

# 设置账号

$ git config --global user.name "Your Name"

$ git config --global user.email "email@example.com"

# 添加工作区文件到缓存区

Git add readme.txt

# 提交commit

Git commit –m “xxxxxx” //提交说明

# 查看仓库当前状态

Git status

# 查看文件修改，Different

Git diff readme.txt

# 查看提交记录，Log

Git log

Git log –pretty=oneline可以将Log简化成一行。

# 回退版本

$ git re**set** --hard HEAD^

一个^就是回退一个版本，^^两个就是两个版本。

如果很多，可以HEAD~100,往前回退100个版本。

Git reset –hard XXXXXX可以回到指定版本。

# 查看所有操作记录

Git reflog

# 撤销操作

Git checkout – filename 把工作区的东西撤销

令中的--很重要，没有--，就变成了“切换到另一个分支”的命令，我们在后面的分支管理中会再次遇到git checkout命令

# 撤销暂存区

Git reset Head file

撤销暂存区到工作区状态，然后可以用上一行命令，回复成原样

# 删除文件

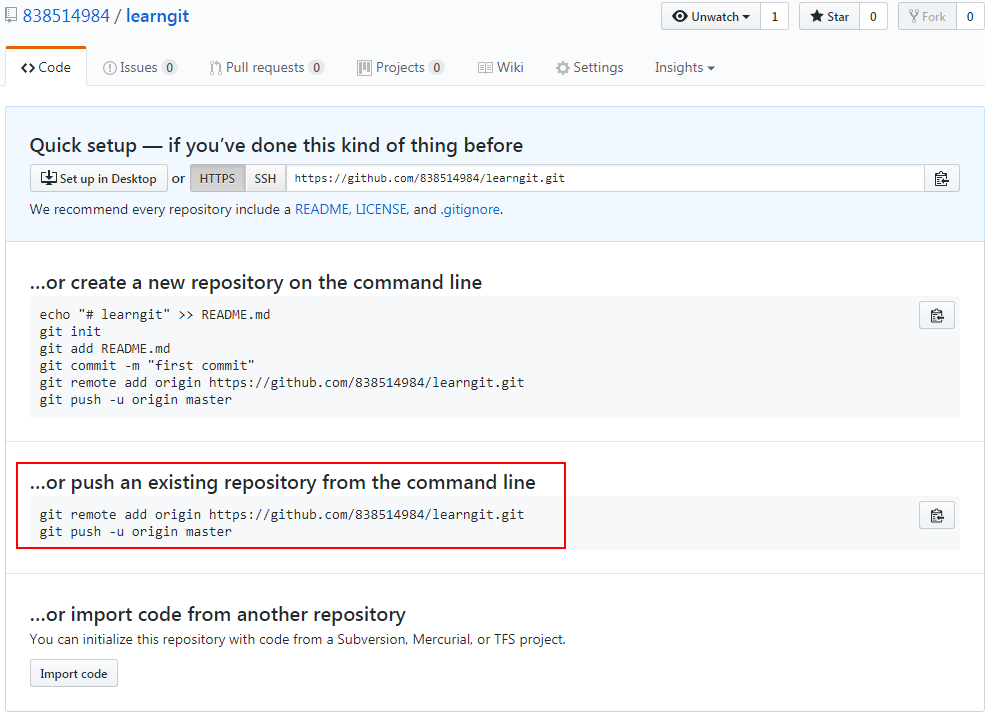
Rm file 删除文件

Git rm file 相当于删除后又add。

# 添加远程库

配置好公钥和私钥；

在github上面创建一个仓库



然后按提示添加。

创建keygen，ssh

Ssh-keygen –t rsa –C “youremail@example.com”

生成两个文件，将公钥放到github上面去。

后续可以用git push进行push。

要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.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；

使用http协议，每次都需要输入口令。

使用服务的话，就不需要每次输入口令

# 管理分支

Git鼓励大量使用分支：

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

# 普通的Merge

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

git merge --no-ff -m "merge with no-ff" dev

因为这样合并要创建一个新的commit，所以加上message

# 储藏和恢复工作区

可以用git stash把工作区储藏起来。可以多次储存。

Git stash list可以查看储存的列表。

一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；

另一种方式是用git stash pop，恢复的同时把stash内容也删了：

# 删除未完成合并的分支

Git branch –D branchname

# 多人协作

### 推送分支

推送分支，就是把该分支上的所有本地提交推送到远程库。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：

$ git push origin master

如果要推送其他分支，比如dev，就改成：

$ git push origin dev

1. 首先，可以试图用git push origin branch-name推送自己的修改；
2. 如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；
3. 如果合并有冲突，则解决冲突，并在本地提交；
4. 没有冲突或者解决掉冲突后，再用git push origin branch-name推送就能成功！

如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream branch-name origin/branch-name。

这就是多人协作的工作模式，一旦熟悉了，就非常简单。

### 小结

* 查看远程库信息，使用git remote -v；
* 本地新建的分支如果不推送到远程，对其他人就是不可见的；
* 从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；
* 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
* 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
* 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。

# 标签

* 命令git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id；
* git tag -a <tagname> -m "blablabla..."可以指定标签信息；后面加commit特定号码
* git tag -s <tagname> -m "blablabla..."可以用PGP签名标签；
* 命令git tag可以查看所有标签。

Git show tagname可以查看制定的标签信息

# 删除标签

* 命令git push origin <tagname>可以推送一个本地标签；
* 命令git push origin --tags可以推送全部未推送过的本地标签；
* 命令git tag -d <tagname>可以删除一个本地标签；
* 命令git push origin :refs/tags/<tagname>可以删除一个远程标签。

# Github fork

Github上面可以随意fork，可以给自己提交。不然在官方的项目上面只有request pull，不然没有权限提交。

# Git ignore

Git目录下面创建一个.gitignore文件，里面填写需要忽略的文件名称。

# 配置别名

git config --global **alias**.st status

代表 st 代替status

配置Git的时候，加上--global是针对当前用户起作用的，如果不加，那只针对当前的仓库起作用。

配置文件放哪了？每个仓库的Git配置文件都放在.git/config文件中：

$ cat .git/config

[core]

repositoryformatversion = 0

filemode = **true**

bare = **false**

logallrefupdates = **true**

ignorecase = **true**

precomposeunicode = **true**

[remote "origin"]

url = git@github.com:michaelliao/learngit.git

fetch = +refs/heads/\*:refs/remotes/origin/\*

[branch "master"]

remote = origin

merge = refs/heads/master

[**alias**]

last = log -1

别名就在[alias]后面，要删除别名，直接把对应的行删掉即可。

而当前用户的Git配置文件放在用户主目录下的一个隐藏文件.gitconfig中：

$ cat .gitconfig

[**alias**]

co = checkout

ci = commit

br = branch

st = status

[user]

name = Your Name

email = your@email.com