## 创建版本库

$ git init

把当前目录变成仓库，多了一个.git的目录

自动创建master分支

$ git add readme.txt

readme.txt是目录下的一个文件，这样git才能找到

git add -A

add所有

$ git commit -m "wrote a readme file"

-m后面是注释

* 如果commit一个.txt文件，显示

[master 01671b3] wrote a readme file

1 file changed, 1 insertion(+)

表示文本文件增加了一行

* 如果提交一个二进制文件

[master 888cc49] commit a picture

1 file changed, 0 insertions(+), 0 deletions(-)

git status

命令查看当前状态

* 所有文件已commit

On branch master

nothing to commit, working tree clean

* 有文件被修改，没有add也没有commit

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: aaa.txt

no changes added to commit (use "git add" and/or "git commit -a")

显示 未经任何处理的changes，也就是没有add过的changes

* 有文件被修改，已经add，没有commit

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: aaa.txt

显示 待commit

git diff readme.txt

修改过但没有add的文件，可以通过diff命令查看具体修改了什么内容

* Untracked files未跟踪文件

在目录里新建一个文件啥都不干就是这个状态

* Changes to be committed

untracked files执行add命令之后的状态

add命令就是把文件修改添加到暂存区stage

* changes not staged for commit

修改文件后未经任何处理的状态

* commit

把暂存区的所有内容提交到当前分支

$ git log

git log命令显示提交日志，也就是历史commit的版本号、时间、作者、注释

commit a6554cb22dc6b92e972174a1bd9706dac60685ac (HEAD -> master)

Author: Nfboys <1033276810@qq.com>

Date: Sat Nov 3 12:49:55 2018 +0800

add GPL

commit 43784d6debff264fb9e1a1d0bb433a90cb580fbe

Author: Nfboys <1033276810@qq.com>

Date: Sat Nov 3 12:48:52 2018 +0800

bbb2

commit 974aa828f5a35afc6d85c4971dcf0dc35f223017

Author: Nfboys <1033276810@qq.com>

Date: Sat Nov 3 12:35:39 2018 +0800

first bbb

最上边是commit id版本号

git log –graph

查看分支合并图

## 版本回退

HEAD表示当前版本，也就是最新的提交a6554cb22dc6b92e972174a1bd9706dac60685ac。上一个版本是HEAD^，上上一个版本是HEAD^^，往上100个版本是HEAD~100。

git reset --hard HEAD^

reset命令回退到上一个版本。亲测不行，报错，git reset –hard HEAD^^可以退回到上一版本

git reset --hard 1094a

回退到某个版本，1094a是版本号前5位，至少写版本号前5位

$ git reflog

reflog记录每一次命令

24e1c23 (HEAD -> master) HEAD@{0}: reset: moving to HEAD^

01671b3 HEAD@{1}: reset: moving to HEAD^

888cc49 HEAD@{2}: reset: moving to HEAD^

e4adb6b HEAD@{3}: commit: addBBB

前面e4adb6b是版本号的前7位，可供reset

* 工作区（Working Directory）

就是根目录，.git文件的父目录

* 版本库（Repository）

.git目录

Git的版本库里存了很多东西，其中最重要的就是称为stage/index的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针HEAD。

把文件往Git版本库里添加的时候分两步执行

第一步git add把文件添加进去，实际上就是把文件修改添加到暂存区；

第二步git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。

修改——add——暂存区——commit——分支

## 撤销修改

git checkout -- readme.txt

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，这里有两种情况：

* readme.txt修改后还没有被放到暂存区，也就是没有git add，则撤销修改就回到和版本库一模一样的状态；
* readme.txt已经通过git add添加到暂存区后，又作了修改，则撤销修改就回到添加到暂存区后的状态。

总之，git checkout命令就是让这个文件回到最近一次git commit或git add时的状态。

* 情况1
* 新建文件readme.txt
* 加一行aaaaaa
* git checkout
* 文件变为空
* 情况2
* 新建文件readme.txt
* 加一行aaaaaa
* git add
* 加一行bbbbb
* git checkout
* 文件变为aaaaa，没有bbbbb

git reset HEAD readme.txt

git reset HEAD <file>可以把暂存区的修改撤销掉（unstage），重新放回工作区：

git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。HEAD表示最新版本。

* 新建readme.txt
* 加一行aaaaaaaa
* git add
* git reset HEAD readme.txt
* 现在readme.txt里还有aaaaaaa，但是暂存区是空的，也就是没有经过git add的状态

## 删除文件

直接在文件管理器中删除一个a.txt之后，要用命令git rm删掉，并且git commit：

git rm a.txt

还可以还原误删的文件在此不解释，有兴趣回头看

## 远程仓库

在github上新建一个repository

假如叫temp

$ git remote add origin [git@github.com:Nfboys/temp.git](mailto:git@github.com:Nfboys/temp.git)

把远方的repository关联到当前的库，远程库的名字就是origin

git pull origin master // 把本地仓库的变化连接到远程仓库主分支

有时候远程库里已经有内容，直接push会报错，所以在add origin和push操作之间加一步pull

git remote

查看远程库的信息

git remote -v

显示形如

origin https://gitlab.com/tjuse/tjqs3.git (fetch)

origin https://gitlab.com/tjuse/tjqs3.git (push)

可以抓取fetch和推送push的远程库地址。如果没有push权限，就看不到push地址

$ git push -u origin master

把本地库所有内容推送到origin

第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。也就是说第一次推送加上-u参数，以后不加。

$ git push origin master

如果先有远程库

git clone git@github.com:Nfboys/temp.git

把远程库下载到本地并关联

## 分支

主分支master

HEAD指向当前分支

git checkout -b b1

创建dev分支，然后切换到dev分支：

$ git branch b1

$ git checkout b1

创建dev分支，然后切换

git switch b1

另一种切换

git switch -c b2

-c参数，如果不存在自动创建新分支

git branch

查看分支

\* b1

master

当前分支前面会标一个\*号

修改文件🡪add🡪commit，都在当前branch上进行，和master互不影响

git checkout master

切回master，文件修改的内容不见了

git branch -d dev

删除分支

### 合并

在master分支

git merge b1

合并b1分支到master分支。git merge用于合并指定分支到当前分支

b1领先master时这样用，HEAD指向b1最新的提交，都是b1修改过后的文件

### 解决冲突

如果在master和b1分支中，同一文件各自有了分别的提交，则产生冲突

* 新建文件mmm.txt

git switch -c b1

* mmm.txt添加一行bbbbbbb

git add mmm.txt

git commit -m "bbb"

git switch master

* mmm.txt添加一行mmmmmmmmaster

git add mmm.txt

git commit -m "mmmmaster"

git merge b1

Auto-merging mmm.txt

CONFLICT (content): Merge conflict in mmm.txt

Automatic merge failed; fix conflicts and then commit the result.

* 此时产生冲突，mmm.txt文件变为

<<<<<<< HEAD

mmmmmmmmmmmmmmaster

=======

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

>>>>>>> b1

<<<<<<< HEAD和=======之间表示当前（master）分支内容

=======和>>>>>>> b1之间表示b1分支新的提交内容

* 本来的master分支变为master|MERGING
* 由于这是在master分支上执行merge操作，所以文件的改动也在master里，b1中的文件不受影响
* 此时

git status

显示

On branch master

You have unmerged paths.

(fix conflicts and run "git commit")

(use "git merge --abort" to abort the merge)

Unmerged paths:

(use "git add <file>..." to mark resolution)

both modified: mmm.txt

no changes added to commit (use "git add" and/or "git commit -a")

* 此时需要手动解决冲突，比如删掉其他行，保留mmmmmaster和bbbbbbbbbbb两行

然后

* git add
* git commit

冲突解决了，master | MERGING也变回master

当然b1分支不受影响，还是只有bbbbbbbbbb这行

在master分支中，可以

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

查看分支合并情况

### 远程仓库

$ git remote

$ git remote -v

查看远程库的信息

git switch -c b1

git add 🡪 git commit

git push origin b1

在b1分支做修改然后提交，push到b1分支上，远程仓库自动新建b1分支，然后同步修改

在b1分支上，往master分支push东西

git push origin master

Everything up-to-date

测试得没有对master分支造成影响

也没有对远程当前的b1分支造成影响，所以这样操作没什么用

在另外一个文件夹克隆远程仓库，模拟另外一个用户

如果用另外一台电脑模拟，好像需要SSH权限

默认情况下，刚刚clone的项目只能看到master分支

git checkout -b b1 origin/b1

本地创建b1分支，关联远程的b1分支。这样就可以用

git push origin b1

命令在b1上提交修改

不能像这样

git checkout -b b2 origin/b1

在本地起名b2分支，然后往b1分支上推送。这样做会报错

src refspec b1 does not match any

error: failed to push some refs to 'https://github.com/Nfboys/testRep1.git'

所以本地分支必需和要推送的远程分支一个名字

用户1在b1分支上推送了修改，在readme.md加了一行ccc

此时用户2想推送自己的修改，在readme.md加了一行xxx，失败，报错

To https://github.com/Nfboys/testRep1.git

! [rejected] b1 -> b1 (fetch first)

error: failed to push some refs to 'https://github.com/Nfboys/testRep1.git'

hint: Updates were rejected because the remote contains work that you do

hint: not have locally. This is usually caused by another repository pushing

hint: to the same ref. You may want to first integrate the remote changes

hint: (e.g., 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

用户1和用户2修改了同一个文件，提交的推送有冲突

解决方法：先用git pull把最新的提交从origin/b1抓下来，然后在本地合并，解决冲突，再推送：

用户2在b1分支：

git pull

拉取下来后readme.md为

<<<<<<< HEAD

xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

=======

ccccccc

>>>>>>> 25040a02742ce863ede6fc3bed26559a5eb44a83

此时分支变更为b1|MERGING

手动解决冲突，重新上传，上传成功

此时用户1的b1分支readme.md还是比较旧的，需要拉取最新的提交

git pull

报错

There is no tracking information for the current branch.

Please specify which branch you want to merge with.

See git-pull(1) for details.

git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

git branch --set-upstream-to=origin/<branch> b1

没有指定本地b1分支跟踪信息，请说明想要合并的远程分支

* 解决方法1

指定要拉取的远程分支

git pull origin b1

* 解决方法2

git branch --set-upstream-to=origin/b1 b1

设置关联，然后git pull，拉取成功。这样一劳永逸，以后就不用带参数了，直接pull

这种多人协作方式，所有开发者都在同一分支上提交，必然会造成冲突，每次都需要重新手动合并。正常的开发应该是每个人在不同的分支上提交，然后pull request合并到master

如果本地分支b1的提交落后于master，想拉取master的最新版

* 思路1：直接在b1上

git pull origin master

git push origin b1

亲测成功

* 思路2：把本地的master先和远程的同步，然后合并master和本地的b1，然后b1再push到远程的b1

在master

git pull

git switch b1

git merge master

git push origin b1

亲测成功

### 远程仓库解决冲突

远程仓库merge分支一般通过手动申请pull request，由github管理员进行审核通过

如果b1分支领先master分支，则将b1分支合并到master上没有冲突，可以自动合并

README.md为空

master分支加一行，mmmmmmm，然后add🡪commit🡪push

b1分支加一行，bbbbbbb，然后add🡪commit🡪push

此时在github上pull request，则不能自动合并。因为两个分支的最新commit是平行关系，没有谁领先谁，而且都修改了README.md

GitHub解决方法：

保证本地有两个分支，master和b1，分别和最新的远程库保持同步

切换到b1分支

git merge master

因为本地merge和远程pull request的原理是一样的，所以这样merge会把问题显示出来在本地

此时merge失败，因为有冲突，而且b1分支变为b1|MERGING

再打开README.md，冲突的地方就显示出来了

手动解决冲突，然后就在b1分支add🡪commit🡪push

这样远程的b1分支就领先了master分支一次提交，而且冲突解决

pull request也自动变为可以进行的状态

BitBucket解决方法：

用一个好像是git pull origin HEAD的方法拉取远程的库，然后就能看到冲突的地方，手动修改再add🡪commit🡪push即可