校招必须掌握的git-视频课程

大家在开发项目的过程中，如果直接在本地系统上维护源码目录，经常会碰见下面的问题：

1. 不小心把源代码的目录或文件删了，写了好久的代码没了！
2. 按需求添加新功能，写了好多代码，但净是编译错误，改都改不完，想回到之前的版本，开始大面积删除或者屏蔽代码，很崩溃，如果此时有个代码版本管理工具，该多好！
3. 新功能添加完了，编译运行一切很顺利，功能也正常，但有时候运行会出现以前没见过的运行错误，非必现的，想查看和之前代码的差异，看看都在哪些源文件中修改了代码，该怎么办？
4. 团队开发项目，但是项目成员都不在一起，各自写的代码该如何添加到一块，还能避免错误，不会出现谁把谁的代码给覆盖了？

git介绍

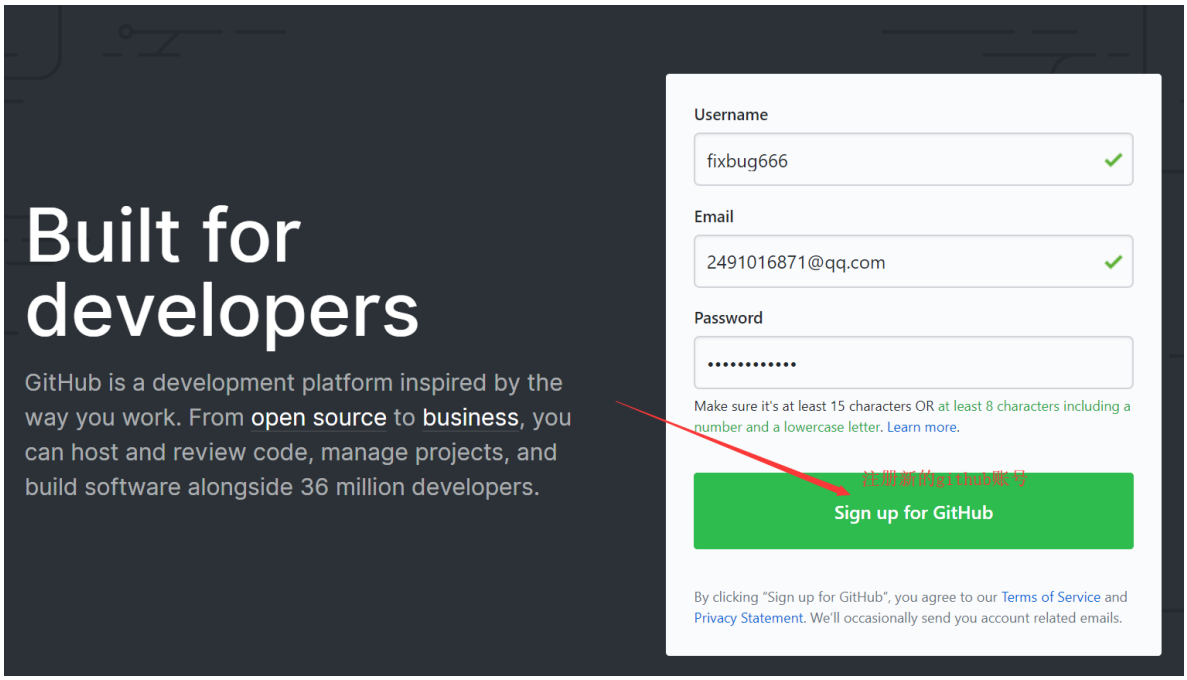
git是目前世界上最先进的分布式版本控制系统（对比集中式版本控制系统SVN），没有之一!

github介绍

一个免费的代码远程托管仓库

GitHub注册新账号

【step 1】：进入github主页 https://github.com/

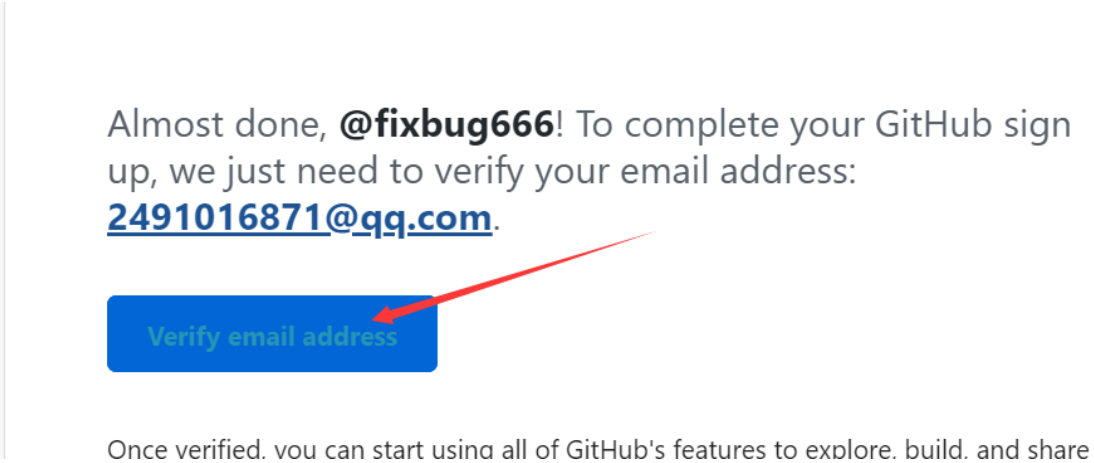


【step 2】：网站为了安全考虑，注册新账号以后，需要验证一下是否是人为操作，你可以跟着页面的提示操作一下。

【step 3】：设置账户，免费的源码托管是GitHub的基石，因此我们使用默认的free账号就行，直接点击页面下方的Continue。

【step 4】：填写你的编程level，你打算用GitHub做什么，然后填写你感兴趣的技术（好好填写完整，会有很多相关项目资料给你推荐！），页面下方点击继续。

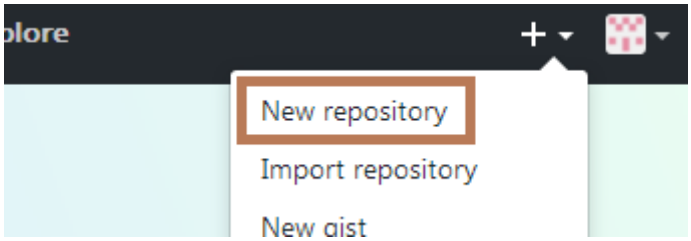
【step 5】：然后提示你去刚注册的邮箱里面，验证一下你的邮箱地址，打开邮件，点击验证就可以了，如下：



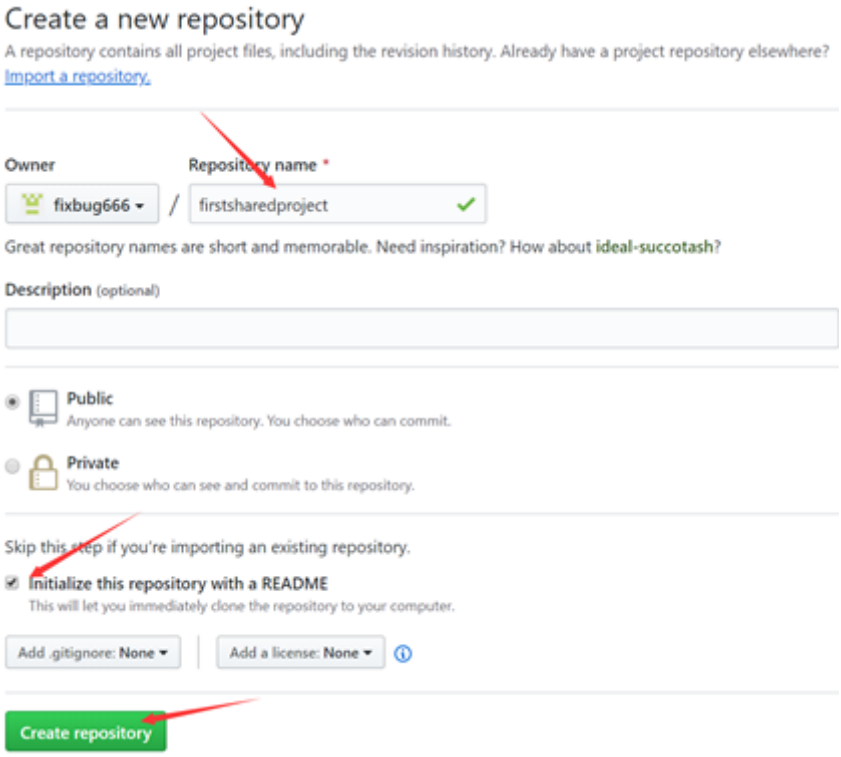
GitHub上创建初始项目

如何在GitHub上创建仓库，创建分支，添加代码，推送修改，拉取代码

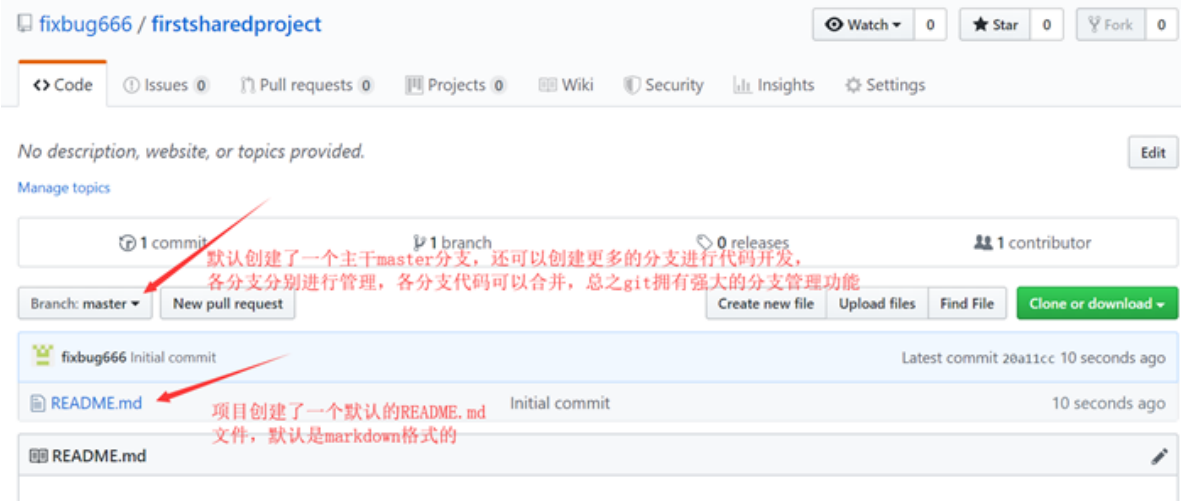
登录GitHub成功以后，在页面的右上角有个+号，点开，选择创建新的代码仓库



填写仓库信息，给仓库命名，写一个简短的介绍，会自动创建一个ReadMe文件！



如下，仓库创建成功！



git本地客户端安装配置

windows

windows平台git客户端

【step 1】双击安装Git-2.18.0-64-bit.exe，可以一路默认安装到底。安装完成后，可以通过git bash启动git客户端命令行

git bash的目录管理实际上都是linux命令，cd,ls,mkdir等都是支持的，你可以自己测试一下。

【step 2】git bash和git hub之间是通过ssh加密传输的，因此需要配置公钥。打开git bash，生成公私密钥，在git hub上进行公钥配置

ssh-keygen -t rsa -C "注册账号的邮箱名字" 生成SSH通信用的公私钥

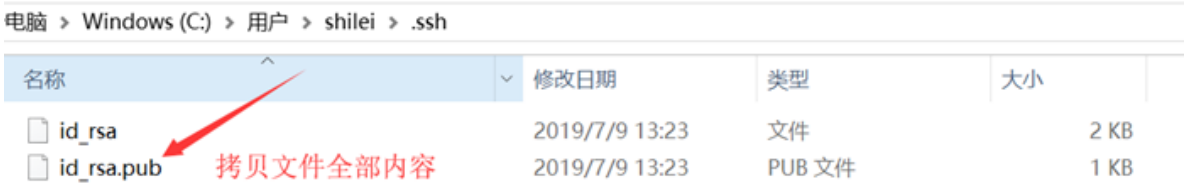
```
shilei@DESKTOP-G21S8HU MINGW64 ~
$ ssh-keygen -t rsa -C "2491016871@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/shilei/.ssh/id_rsa):
Created directory '/c/Users/shilei/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/shilei/.ssh/id_rsa.
Your public key has been saved in /c/Users/shilei/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:HFLb1Dd2+Q+/1xncEJhbh5l7s54i/AgvMlD3yhV9eVI 2491016871@qq.com
The key's randomart image is:
+---[RSA 2048]----+
|         . .. o +.|
|        . +  + X.E|
|       . o . .= Bo|
|        + o ...=++|
|       . S . . o**|
|      .     o   ++|
|       . ..+    . *|
|        o +oo.. =o|
|         o .oo.. .|
+----[SHA256]-----+

shilei@DESKTOP-G21S8HU MINGW64 ~
$
```

替换成你的注册的git hub账号的邮箱地址

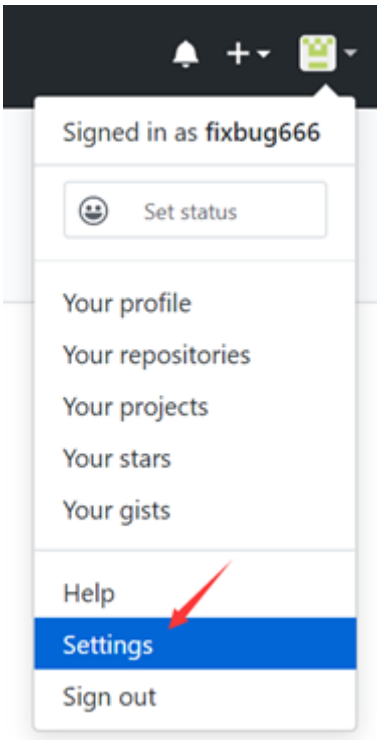
全部回车就可以！

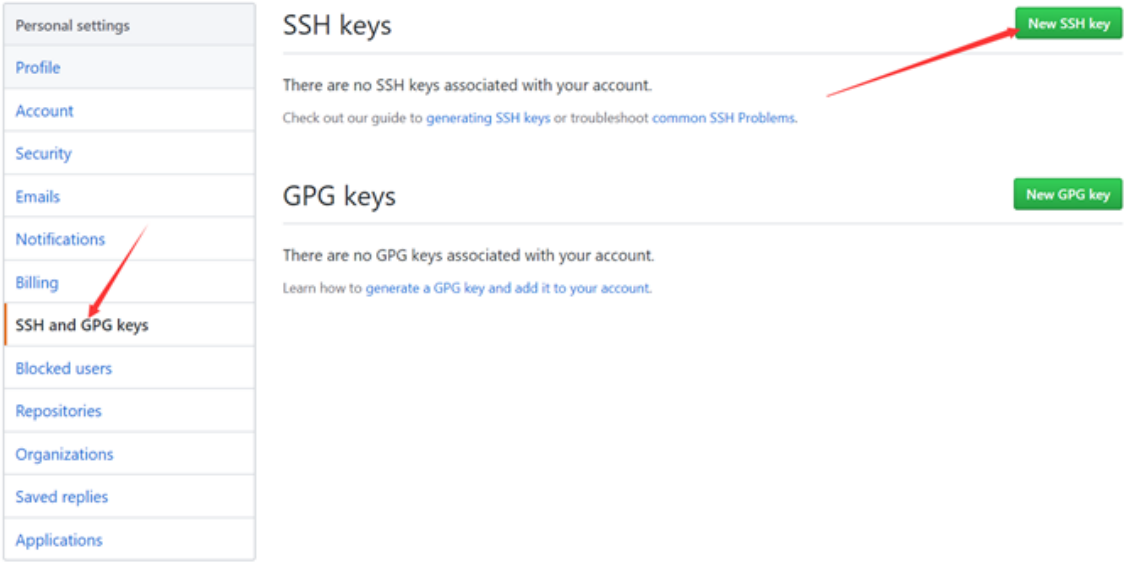
生成的公钥文件，一会儿拷贝该文件内容，在git hub上进行配置

【step 3】在上面图片上标注的路径下，找到id_rsa.pub公钥文件，拷贝文件内容

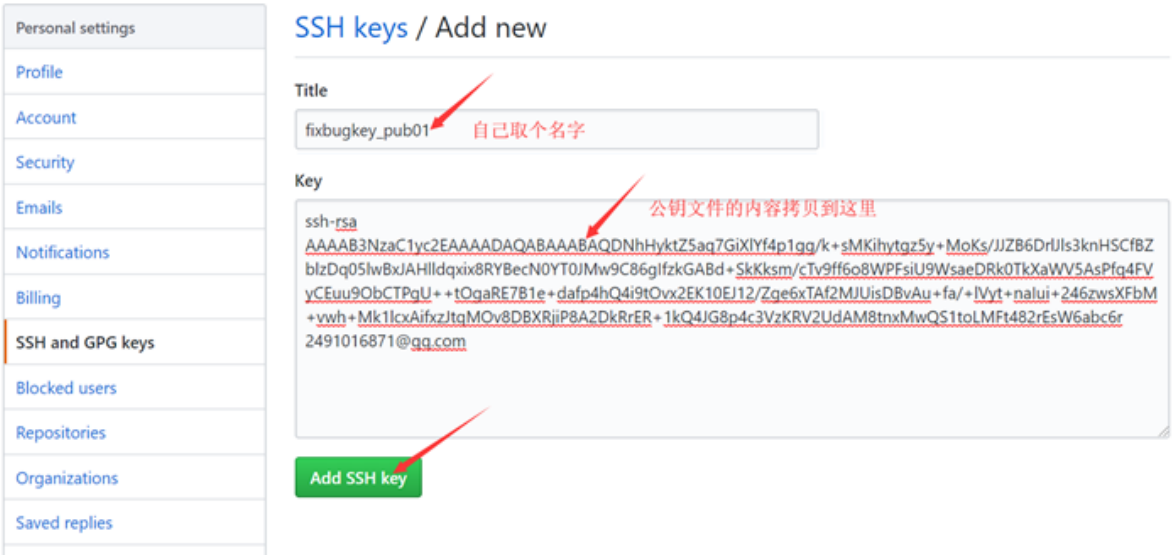


【step 4】登录git hub，点击右上角的头像，进入setting设置页面





点击New SSH key，把之前在id_rsa.pub文件中拷贝的内容，粘贴到输入框中，如下图



添加完成后，需要再次输入密码确认，添加完成！

【step 5】打开git bash，输入以下命令，测试和git hub是否能够通信成功，如下：

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github
$ ssh -T git@github.com
Hi fixbug666! You've successfully authenticated, but GitHub does not provide sh
ell access.
```

上面显示连接git hub成功，如果你出现如下的提示：

```
The authenticity of host 'github.com (52.74.223.119)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
```

直接输入yes就可以！

在上面的提问那里直接输入yes回车就可以了！

【step 6】配置邮箱和用户名，以后你在git hub上提交的任何代码文件，都会附带你的邮箱用户名信息，如下

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码
$ git config --global user.name "fixbug"

shilei@DESKTOP-G21S8HU MINGW64 /d/代码
$ git config --global user.email "2491016871@qq.com"
```

填写你自己的用户名和邮箱

ubuntu

在linux环境下配置git bash和上面的一样，首先在ubuntu上安装git和ssh服务，然后生成公私钥，ssh-keygen -t rsa -C "注册账号的邮箱名字"，生成SSH通信用的公私钥把公钥配置到github上面，如下：

```
tony@tony-virtual-machine:~/github$ ssh-keygen -t rsa -C "2491016871@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tony/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/tony/.ssh/id_rsa.
Your public key has been saved in /home/tony/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:BqdJGKqSSevmLjiKlD83O3vx4+UubnCUy4IFwsERCAk 2491016871@qq.com
The key's randomart image is:
+---[RSA 2048]----+
|E.++=            |
|.. = +           |
| .. o + . .      |
|.+.  . * o       |
|=.    = S .      |
|o .  . = +       |
|.=      *  .     |
|O .. + . =o      |
|*+ .oo* +o+o     |
+----[SHA256]-----+
tony@tony-virtual-machine:~/github$ cat /home/tony/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDZhZNUsuAb1vtTIageFIxnvyFodP0Um3EtLW+/JAYZGXkm2usR3vGLoyq90fSlw/UrBqvywUHZHVMUQNiEVL8e
qwwQD+KHaNMuBiIdN77gT1vHnLFj3+wCHapMhCC43MpIsiV+5mzAfhcnRjDgvA0KhRQDXM8OAZSec/WG4W4bkSJaZyIr3eyJONqSDUdU1CVAdBn6mCAphVf5N3WL
jq5P62cnGWJVTgY0a3bb6LVDGCDFY6al9zsOWCpFrRn2keiChcPPIlIyHowmnGamM8kGC+5fvhUX631A4You46b5j2WhL99kGVEH+C1mVilOR9blG9EvA7zWRIC+
EsyTHcCl 2491016871@qq.com
tony@tony-virtual-machine:~/github$
```

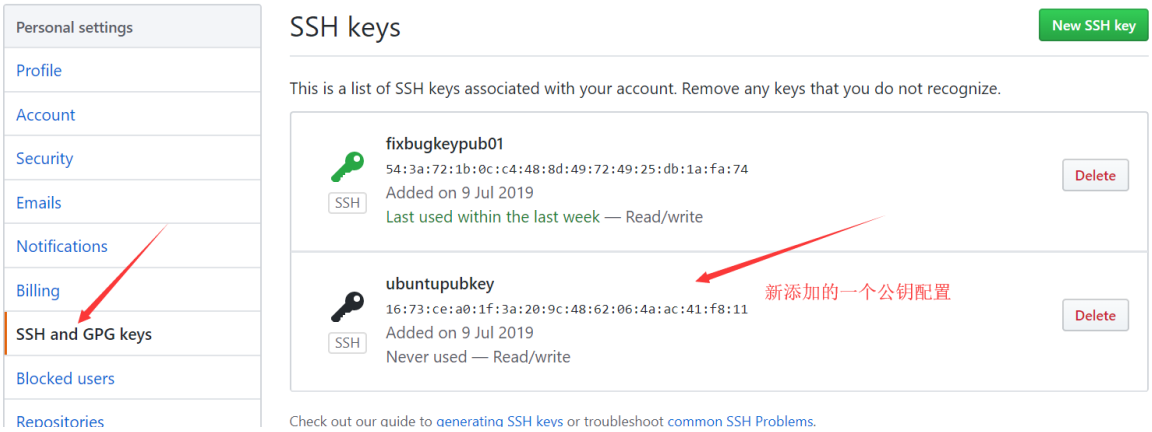
生成公钥文件

这里全部都是输入回车

公钥文件路径

打开公钥文件，把下面文件的内容拷贝一下，在git hub上进行配置

在github上添加了一个公钥配置，如下：



在ubuntu shell下测试能否连接到git hub上面，如下：

```
tony@tony-virtual-machine:~/github$ ssh -T git@github.com
The authenticity of host 'github.com (13.250.177.223)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,13.250.177.223' (RSA) to the list of known hosts.
Hi fixbug666! You've successfully authenticated, but GitHub does not provide shell access.
```

证明ubuntu下git客户端连接github成功，最后配置git提交内容的用户名和邮箱信息：

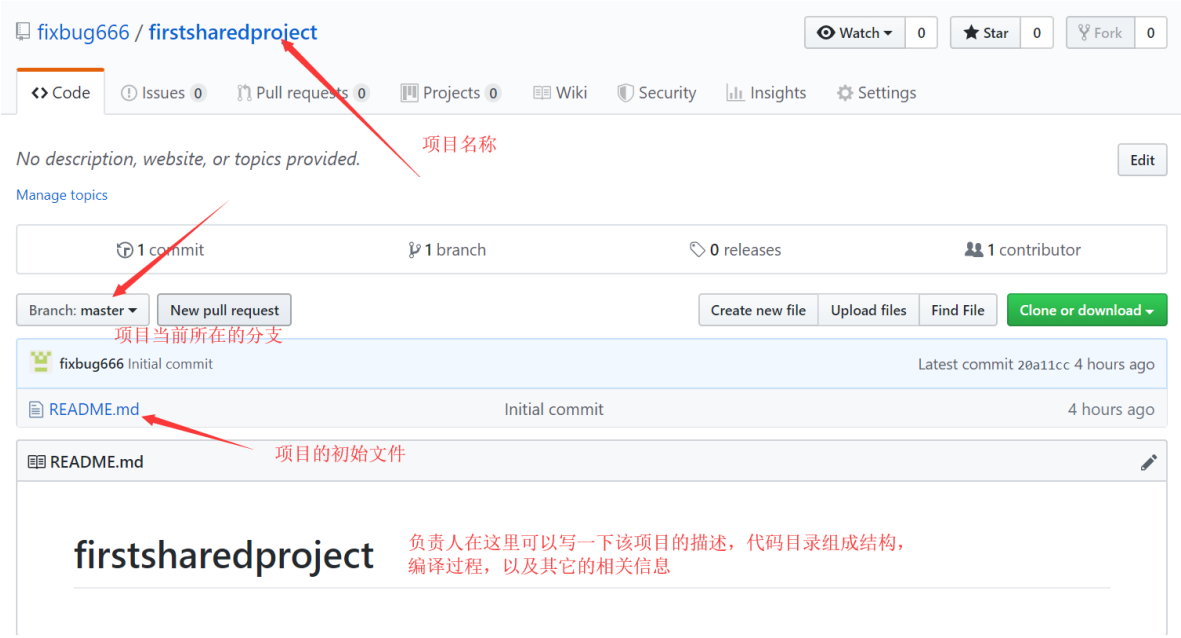
tony@tony-virtual-machine:~/github$ git config --global user.name "ubuntu-fixbug"
tony@tony-virtual-machine:~/github$ git config --global user.email "2491016871@qq.com"

配置完成！

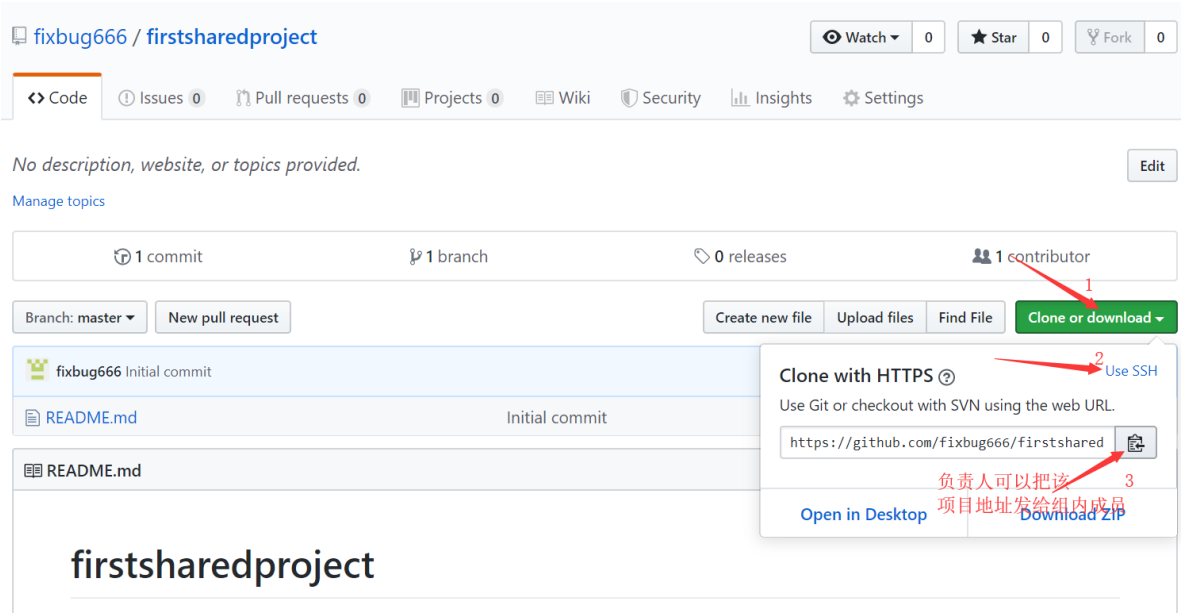
git常用命令介绍

git常用基本操作命令

【step 1】在团队开发新项目时，项目负责人王sir（或者是团队专门负责维护代码仓库的人）先在公司私有的代码仓库上创建了一个项目（我们直接以git hub举例），如下：



【step 2】王sir把上面项目的地址（SSH地址）分享给组内其它成员，大家拿到git地址后，在本地通过git clone把远程仓库上的项目代码拉到本地，如下：



作为项目组成员，可以在本地新建一个目录，专门存放该项目代码，通过git clone拉取远程代码，如下：

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github
$ git clone https://github.com/fixbug666/firstsharedproject.git
Cloning into 'firstsharedproject'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github
$ ls
firstsharedproject/

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github
$ cd firstsharedproject/

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ ls
README.md

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
```

显示拉代码的进度

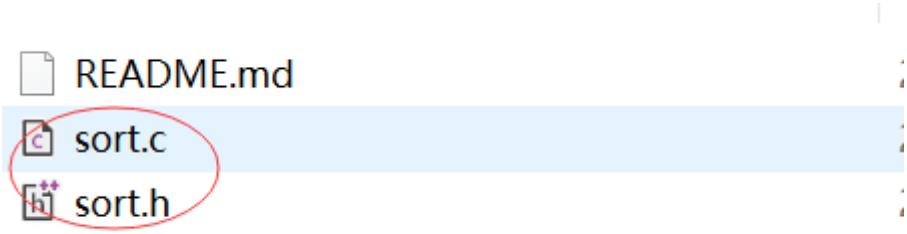
根据远程仓库的代码地址，把代码拉到本地

可以看到项目已经拉到本地了

可以看到，我们现在拉的是远程项目master分支的代码

这是创建项目时的初始文件

【step 3】小张是项目主程，责任重大，现在需要开发一个排序的代码，小张负责写具体算法。小张在firstsharedproject目录下创建了一个sort.c和sort.h文件，分别写了相应代码，如下：



开发完成以后，小张需要把改动的代码提交到远程仓库（github）上去。

【step 4】git add命令可以把本地修改的代码或者文件，添加到本地暂存区（后面讲详细原理），如下：

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ ls
README.md  sort.c  sort.h

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        sort.c
        sort.h

nothing added to commit but untracked files present (use "git add" to track)

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git add .

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   sort.c
        new file:   sort.h


shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
```

可以看到本地目录添加的两个新的代码文件

建议多通过git status命令查看当前git工作区的状态

通过信息提示，可以看到当前我们本地master分支上的代码，新增加了两个代码文件

通过git add .把所有改动的代码或文件添加到暂存区，或者通过git add sort.h一个一个文件添加

说明已经把改动的代码文件正确的添加到了暂存区

【step 5】通过git commit提交命令，把修改的代码文件，从暂存区提交到本地的master分支上去（实际上当我们第一次用git clone拉取远程master分支的代码时，本地也创建了一个master分支，保存用户修改后需要提交的代码，分布式嘛，每个机器上都可以维护一个代码仓库！）

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git commit -m "添加了排序的代码"
[master 510156f] 添加了排序的代码
 2 files changed, 25 insertions(+)
 create mode 100644 sort.c
 create mode 100644 sort.h
```

暂存区的代码提交到本地master分支代码仓库中

【step 6】代码存到小张自己电脑上的master分支不行啊，别人看不到代码修改，所以小张再通过git push命令，把本地master分支上的所有代码，都推送到远程master分支上去了（俗称“合代码”）

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 565 bytes | 282.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/fixbug666/firstsharedproject.git
   20a11cc..510156f  master -> master
```

本地master分支的代码推送到远程代码仓库的master分支上

推送进度

左边是本地的分支名称，右边是远程的分支名称

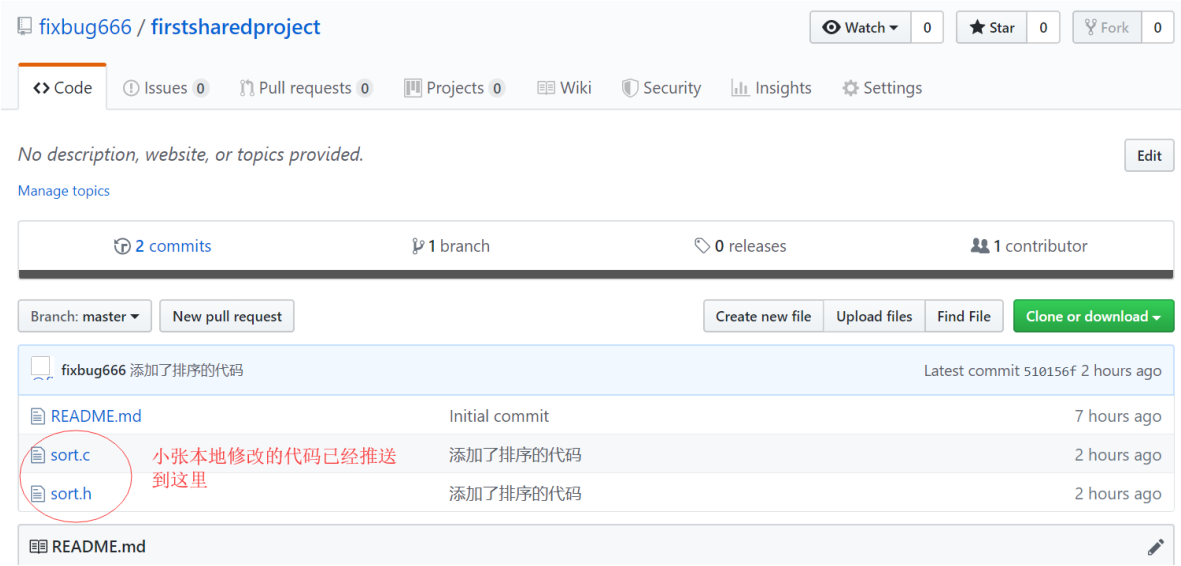
默认的远程仓库名字就是origin，第一次推送会提示输入git hub账号用户名和密码，输入即可。

【step 7】小张输入git log，可以查看修改日志

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git log
commit 510156fb98acabc0725ef44842944046027005e0 (HEAD -> master, origin/master, origin/HEAD)
Author: 小张 <2491016871@qq.com>
Date:   Tue Jul 9 16:18:07 2019 +0800

    添加了排序的代码

commit 20a11cc6ffccbdd2b97c7001a22879d161ad2448
Author: fixbug666 <52686902+fixbug666@users.noreply.github.com>
Date:   Tue Jul 9 11:29:16 2019 +0800

    Initial commit
```

小张的代码提交信息，生成了一个长串的logid，作者姓名和操作的日期以及修改说明信息

【step 8】可以查看git hub上面项目master分支的代码修改，发现sort.c和sort.h都已经推送到git hub远程代码仓库的master分支上了。



【step 9】小张已经把排序的代码推送到远程master分支了，现在给小弟啊亮分配任务，需要写一段测试代码，测试排序函数的正确性，步骤是：

1. 啊亮需要先在git bash上通过git pull命令拉取firstsharedproject项目的最新代码，然后基于最新代码进行修改（注意：当修改代码前，执行一下git pull命令是一个好习惯！）
2. 修改完代码，通过git add、git commit、git push命令把修改推送到git hub远程项目代码仓库中，完成

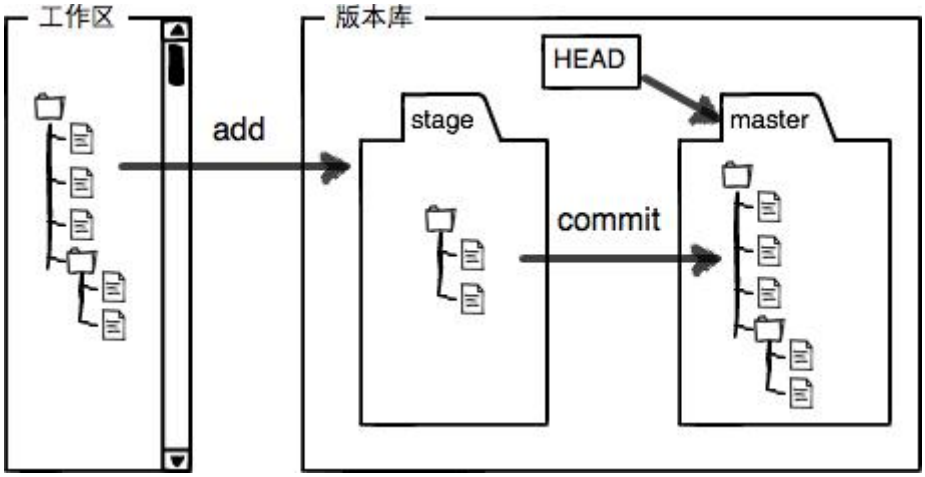
```
tony@tony-virtual-machine:~/github/firstsharedproject$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
展开对象中: 100% (4/4), 完成.
来自 https://github.com/fixbug666/firstsharedproject
   20a11cc..510156f  master     -> origin/master
更新 20a11cc..510156f
Fast-forward
 sort.c | 18 ++++++++++++++++++
 sort.h |  7 +++++++
 2 files changed, 25 insertions(+)
 create mode 100644 sort.c
 create mode 100644 sort.h
tony@tony-virtual-machine:~/github/firstsharedproject$ ls
README.md  sort.c  sort.h
```

```
tony@tony-virtual-machine:~/github/firstsharedproject$ git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。

未跟踪的文件:
  （使用 "git add <文件>..." 以包含要提交的内容）

    testsort.c

提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
tony@tony-virtual-machine:~/github/firstsharedproject$ git add testsort.c
tony@tony-virtual-machine:~/github/firstsharedproject$ git commit -m "增加排序测试代码"
[master 01d94a2] 增加排序测试代码
 1 file changed, 12 insertions(+)
 create mode 100644 testsort.c
tony@tony-virtual-machine:~/github/firstsharedproject$ git push origin master
Username for 'https://github.com': fixbug666
Password for 'https://fixbug666@github.com':
对象计数中: 3, 完成.
Delta compression using up to 8 threads.
压缩对象中: 100% (3/3), 完成.
写入对象中: 100% (3/3), 438 bytes | 438.00 KiB/s, 完成.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/fixbug666/firstsharedproject.git
   23796db..01d94a2  master -> master
```

【step 10】啊亮推送完成以后，可以通过git log查看代码修改日志，会发现所有人对项目的修改信息都记录在列

```
tony@tony-virtual-machine:~/github/firstsharedproject$ git log
commit 01d94a26d8cbb0630a57b510fcea5bfea932748d (HEAD -> master, origin/master,
origin/HEAD)
Author: 啊亮 <2491016871@qq.com>
Date:   Tue Jul 9 19:20:06 2019 +0800

    增加排序测试代码

commit 23796db9be8c8ea924f3e75f41aef8d53b55e235
Author: 小张 <2491016871@qq.com>
Date:   Tue Jul 9 19:12:49 2019 +0800

    修改了sort.h头文件中函数的名字

commit 510156fb98acabc0725ef44842944046027005e0
Author: 小张 <2491016871@qq.com>
Date:   Tue Jul 9 16:18:07 2019 +0800

    添加了排序的代码

commit 20a11cc6ffccbdd2b97c7001a22879d161ad2448
Author: fixbug666 <52686902+fixbug666@users.noreply.github.com>
Date:   Tue Jul 9 11:29:16 2019 +0800

    Initial commit
```

> git clone命令的作用是，可以把指定的远程仓库代码拉取到本地
>
> git add 把git工作区的代码改动添加到暂存区
>
> git commit -m "xxx" 把暂存区的代码提交到本地分支
>
> git push 把本地分支的代码推送（提交）到远程分支上去
>
> git pull 把远程代码拉取到本地
>
> git status 查看当前操作的状态信息
>
> git log 查看代码修改日志
>
> 工作区：当前存放项目代码的目录
>
> 暂存区：git add把工作区修改的内容添加到暂存区当中
>
> 本地仓库：git commit把本地暂存区的修改提交到本地代码仓库分支中（不同分支代表不同的代码版本）
>
> 远程仓库：通过git push把本地仓库的某一个分支上的代码推送到远程仓库的某个分支上
>
> HEAD指针：本地仓库每一个分支上的代码修改都会生成一个commit id信息，HEAD指针指向最近一次的commit提交，通过这个commit id可以进行版本回退



git各阶段版本回退命令

1. 工作区的代码改动不想要了（git add之前）

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

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

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git checkout -- README.md

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

2. git add以后放入暂存区的代码修改不想要了

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md
```

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

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git add .
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.md


shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git reset HEAD
Unstaged changes after reset:
M       README.md

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

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

3. git commit提交到本地仓库的代码不想要了

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git commit -m "修改readme.md文件，添加222222"
[master 0de4add] 修改readme.md文件，添加222222
 1 file changed, 1 insertion(+)

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git log
commit 0de4add575d3c73b03290a6b931ccde87b641629 (HEAD -> master)
Author: 小张 <2491016871@qq.com>
Date:   Wed Jul 10 15:17:39 2019 +0800

    修改readme.md文件，添加222222

commit 216c246594f9e90c32e98a4ccf6b838b3ca3a7c9 (origin/master, origin/HEAD)
Author: 小张 <2491016871@qq.com>
Date:   Wed Jul 10 15:08:23 2019 +0800

    修改readme.md文件，添加111111
```

看上面的git log，每一次commit提交日志都生成一个commit id，如果修改刚提交的修改想回退，那么通过上面的commit id就可以（commit id不需要写全，前几位能区分不同的commit id就可以），命令如下：

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git reset --hard 216c
HEAD is now at 216c246 修改readme.md文件

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git log
commit 216c246594f9e90c32e98a4ccf6b838b3ca3a7c9 (HEAD -> master, origin/master,
origin/HEAD)
Author: 小张 <2491016871@qq.com>
Date:   Wed Jul 10 15:08:23 2019 +0800

    修改readme.md文件

commit 01d94a26d8cbb0630a57b510fcea5bfea932748d
Author: 啊亮 <2491016871@qq.com>
Date:   Tue Jul 9 19:20:06 2019 +0800

    增加排序测试代码
```

实际上，上面的git reset --hard commit_id是把本地仓库分支版本上的HEAD指针进行了移动，实际上没有删除任何内容，如果上面的代码回退你后悔了，可以用git reset --hard继续返回到之前的版本上，但是之前版本的commit id在哪里看呢？可以用git reflog命令，如下：

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git reflog
216c246 (HEAD -> master, origin/master, origin/HEAD) HEAD@{0}: reset: moving to
216c
0de4add HEAD@{1}: reset: moving to 0de4
0de4add HEAD@{2}: commit: 修改readme.md文件，添加222222
216c246 (HEAD -> master, origin/master, origin/HEAD) HEAD@{3}: reset: moving to
HEAD
216c246 (HEAD -> master, origin/master, origin/HEAD) HEAD@{4}: commit: 修改
readme.md文件
01d94a2 HEAD@{5}: pull: Fast-forward
23796db HEAD@{6}: commit: 修改了sort.h头文件中函数的名字
510156f HEAD@{7}: commit: 添加了排序的代码
20a11cc HEAD@{8}: clone: from
https://github.com/fixbug666/firstsharedproject.git

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git reset --hard 0de4
HEAD is now at 0de4add 修改readme.md文件，添加222222

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git log
commit 0de4add575d3c73b03290a6b931ccde87b641629 (HEAD -> master)
Author: 小张 <2491016871@qq.com>
Date:   Wed Jul 10 15:17:39 2019 +0800

    修改readme.md文件，添加222222
```

```
commit 216c246594f9e90c32e98a4ccf6b838b3ca3a7c9 (origin/master, origin/HEAD)
Author: 小张 <2491016871@qq.com>
Date:   Wed Jul 10 15:08:23 2019 +0800

    修改readme.md文件
```

看到了吧，我又回来啦！

4. 远程仓库的代码修改不想要了，有两种方法：

a、git pull，在本地分支最新的代码版本上删除之前修改的，然后重新push到远程代码仓库上

b、在本地分支上通过git reset --hard xxx回退到之前的版本，然后通过git push -f推送覆盖远程代码仓库

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git reset --hard 216c
HEAD is now at 216c246 修改readme.md文件

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git push origin master
To https://github.com/fixbug666/firstsharedproject.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to
'https://github.com/fixbug666/firstsharedproject.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git push -f origin master
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/fixbug666/firstsharedproject.git
 + 0de4add...216c246 master -> master (forced update)

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git log
commit 216c246594f9e90c32e98a4ccf6b838b3ca3a7c9 (HEAD -> master, origin/master,
origin/HEAD)
Author: 小张 <2491016871@qq.com>
Date:   Wed Jul 10 15:08:23 2019 +0800

    修改readme.md文件
```

> git checkout -- 在git add之前，把工作区的代码用版本库中的代码覆盖掉，注意命令中的--不能去掉，否则成切换分支的命令了
>
> git reset HEAD 把git add之后，暂存区的内容全部撤销
>
> git reset --hard commitid 把提交到本地仓库中的代码改动进行回退
>
> git reflog 查看HEAD指针的改动日志
>
> git push -f 强制推送本地仓库代码到远程仓库
>
> git diff HEAD -- 查看工作区file文件和仓库中该文件最新版本的代码有什么区别

git分支版本控制命令

本地分支管理

【step 1】小张让啊亮给他写的冒泡排序进行一下优化，当一趟排完了，发现没有进行任何数据交换，那么就直接结束排序，啊亮想着最好不要在master分支修改代码，还是重新创建一个本地分支吧，写完代码测试好，我再合并到master主干分支上，然后再推送到远程代码仓库中，最为稳妥，master主干分支代码干干净净，省的我改来改去，把原来好的代码改错了。

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git checkout -b sortev01 创建一个新的本地分支sortdev01并切换到该分支
Switched to a new branch 'sortev01'

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortev01)
$ git branch
  master                                                    可以看到，已经切换
* sortev01                                                  到sortdev01分支了
```

git checkout -b sortdev01就是创建一个新的本地分支sortdev01并切换到该分支，从上面的命令可以看到已经切换到sortdev01分支了，git checkout -b相当于是这两个命令（**git branch sortdev01是创建分支，git checkout sortdev01是切换分支**）的合并。

【step 2】啊亮在sortdev01分支上进行的代码修改，测试正确以后，可以切换到master分支上，然后通过git merge命令把sortdev01分支上的代码改动合并到master主干分支上，然后推送到远程的代码仓库当中，如下：

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev01)
$ git add .
                                  在sortdev01分支上修改代码，提交到本地仓库的sortdev01分支上
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev01)
$ git commit -m "修改sort.c文件中的flag为char类型，节省空间"
[sortdev01 9d140ab] 修改sort.c文件中的flag为char类型，节省空间
 1 file changed, 1 insertion(+), 1 deletion(-)

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev01)
$ git checkout master      切换到master分支上
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git merge sortdev01    通过git merge命令把sortdev01分支上的代码修改，合并到master主干分支上来
Updating 7369f5e..9d140ab
Fast-forward
 sort.c | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git push origin master   把本地master分支上的代码改动推送到远程origin仓库的master分支上
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 427 bytes | 427.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/fixbug666/firstsharedproject.git
   7369f5e..9d140ab  master -> master

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
```

如果啊亮做完该任务后，不想要sortdev01分支了，可以通过git branch -d sortdev01删除该分支，所以git在本地仓库中可以让我们创建很多分支，我们可以在分支上瞎折腾，把功能开发测试好了，然后在合并到其它分支当中，git建议多创建使用分支，使用起来非常灵活。

实际上，啊亮还可以直接在他的sortdev01分支上，把代码推送到远程origin仓库的master分支，命令如下：

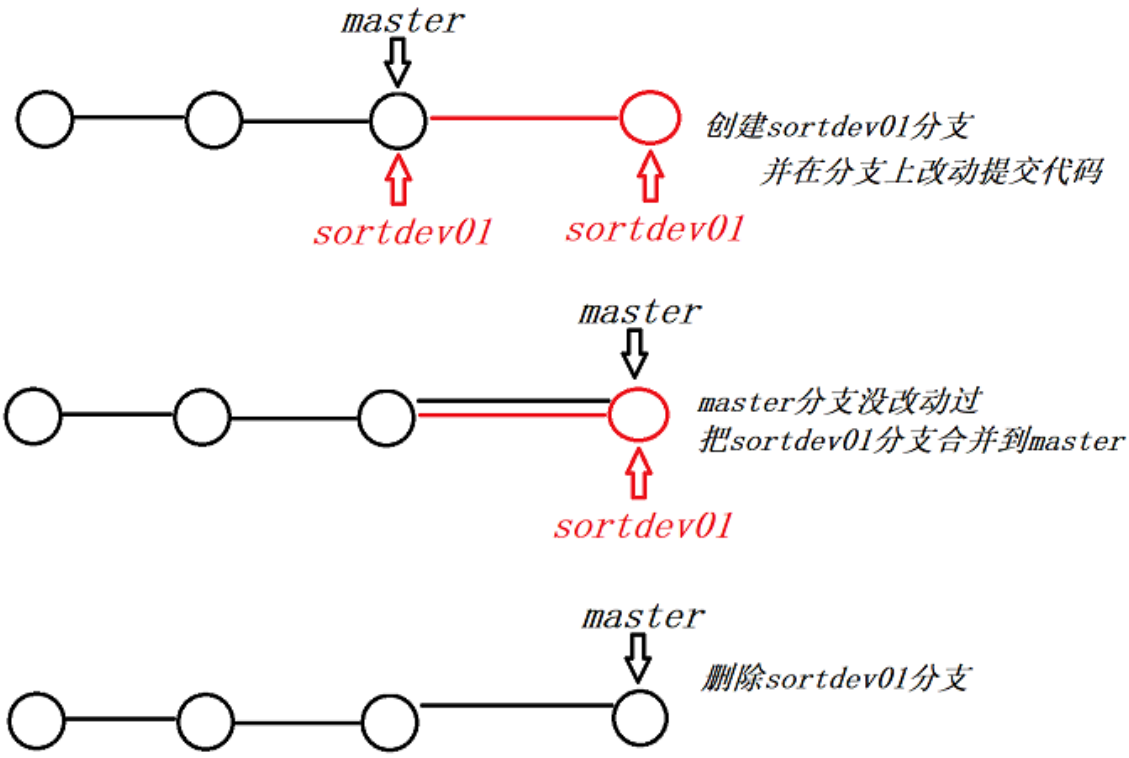
```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev01)
$ git add .

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev01)
$ git commit -m "修改sort.c代码"
[sortdev01 ddd6702] 修改sort.c代码                    在分支上提交代码修改
 1 file changed, 3 insertions(+), 3 deletions(-)

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev01)
$ git push origin sortdev01:master  把sortdev01分支上的代码直接推送到master主干上
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 404 bytes | 404.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/fixbug666/firstsharedproject.git
   9d140ab..ddd6702  sortdev01 -> master

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev01)
$ git checkout master   切换回master分支
Switched to branch 'master'
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)
                    提示你本地的master分支behind落后远程master分支一个提交，因为你刚才在sortdev01分支上给远程
                    master分支推送新的内容了
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git pull   在master主干分支上直接git pull就可以更新到远程origin仓库master分支最新的代码
Updating 9d140ab..ddd6702
Fast-forward
 sort.c | 6 +++---
 1 file changed, 3 insertions(+), 3 deletions(-)

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
```

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git branch -d sortdev01  删除sortdev01分支
Deleted branch sortdev01 (was ddd6702).

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git branch
* master      查看只剩master分支了
```

> 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>`
>
> 删除本地分支：`git branch -d <name>` 如果分支上有更新没有merge，git会提示你merge，强制删除用-D
>
> 本地分支推送到远程分支：git push <远程仓库名> <本地分支名>:<远程分支名>

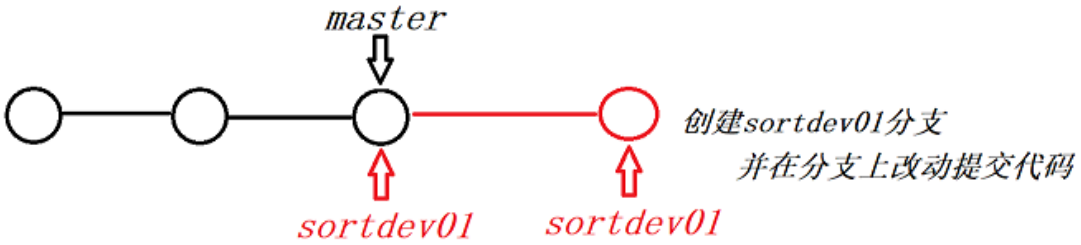
合并分支冲突

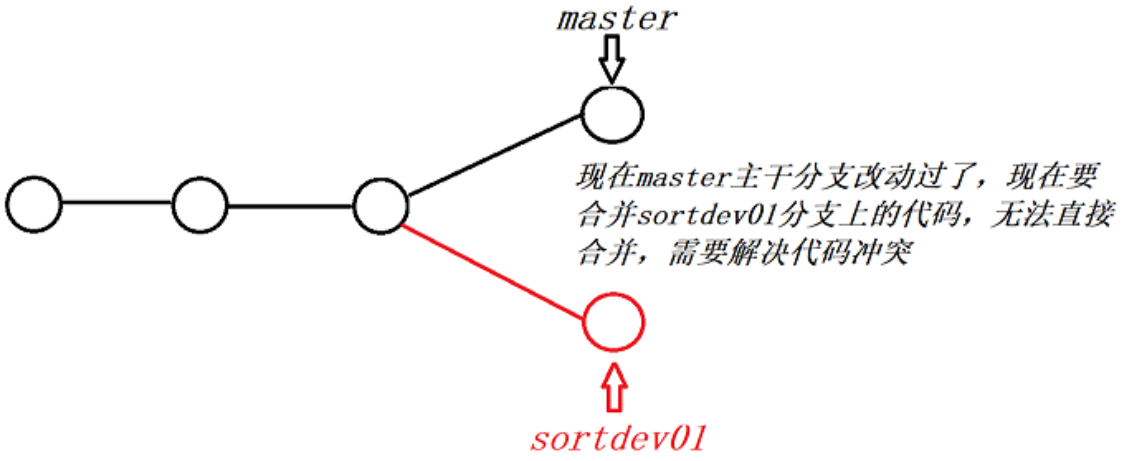
上一小节把sortdev01分支合并到master分支上时一切顺利，是因为sortdev01分支在合并的时候，master分支没有做过任何改动，看图：

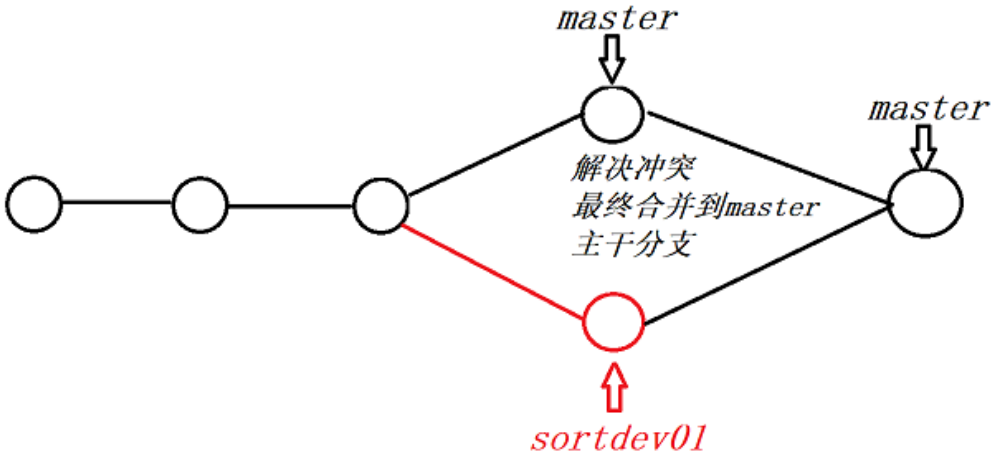


实际上有可能发生这样的情况：

1. 啊亮从master分支创建了一个新的分支sortdev01，进行代码开发测试，提交
2. 小张更新了master分支上的代码
3. 啊亮切换分支到master，git pull同步远程仓库master主干的最新代码，发现有变化
4. 啊亮直接git merge sortdev01就发生冲突了


master
创建sortdev01分支
并在分支上改动提交代码
sortdev01
sortdev01


master
现在master主干分支改动过了，现在要
合并sortdev01分支上的代码，无法直接
合并，需要解决代码冲突
sortdev01


master
master
解决冲突
最终合并到master
主干分支
sortdev01

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git checkout -b sortdev02
Switched to a new branch 'sortdev02'

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev02)
$ vim README.md

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev02)
$ git add .

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev02)
$ git commit -m "readme添加222222"
[sortdev02 ca049aa] readme添加222222
 1 file changed, 1 insertion(+)
```

啊亮在sortdev02分支上修改了readme文件
在原来的111111基础上又添加了222222

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (sortdev02)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/fixbug666/firstsharedproject
   ddd6702..deb53e2  master     -> origin/master
Updating ddd6702..deb53e2
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

因为小张在master更新了代码，把readme改成
111111
333333　所以啊亮的master更新了代码

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git merge sortdev02
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

现在master合并sortdev02分支的代码，发生冲突

111111 111111
333333 222222

这样的文件不知道怎么合并了，需要我们手动处理冲突！！！

通过git diff命令查看一下README.MD文件的冲突，在master主干上解决冲突，并提交远程仓

库

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master|MERGING)
$ git diff -- README.md
diff --cc README.md
index 62e0871,06ec666..0000000
--- a/README.md
+++ b/README.md
@@@ -1,2 -1,2 +1,6 @@@
  111111
++<<<<<<< HEAD
 +333333
++=======
+ 222222
++>>>>>>> sortdev02
```

这里提示master上更新了333333，sortdev02分支更新了222222，需要手动处理冲突
我们自己手动把文件内容改成
111111
222222
333333
就可以了！

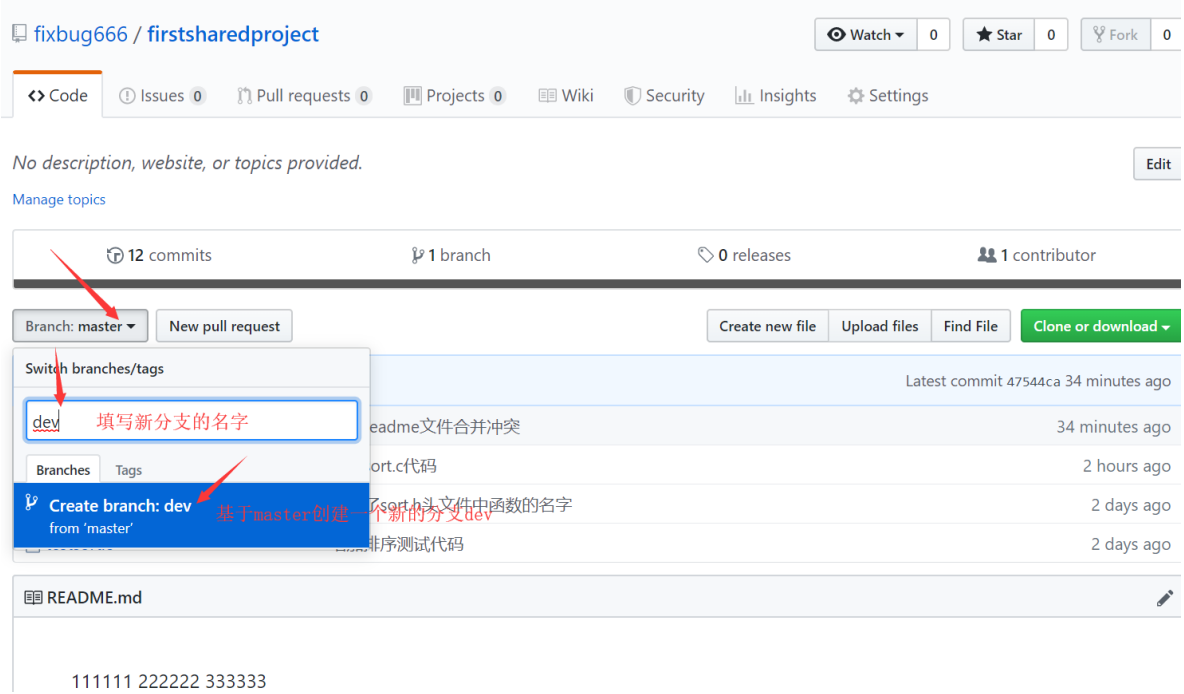
然后再通过git add、git commit、git push把修改推送到远程origin仓库的master主干分支上，冲突就解决完了。

远程分支管理

我们在github上创建一个代码仓库，默认就拉了一个master主干分支，我们在master主干分支上开发了一期项目以后，再进行二期开发的时候，可以再拉一个dev分支出来，大家都在dev分支上开发，此时项目管理员可以把master分支的写权限关掉，因为一期功能开发验证完成，很稳定，此时可以拉取master代码，不能再push推送代码到master分支。

一般远程分支的创建都是由项目管理员来创建的，其它员工没有创建远程分支的权限，而且每一个远程分支的读写权限也都是由管理员来控制的。

【step 1】由王sir在github上创建一个dev分支，来继续开发新的功能



【step 2】小张在git bash上通过git pull更新仓库内容

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git pull
From https://github.com/fixbug666/firstsharedproject
 * [new branch]      dev        -> origin/dev
Already up to date.  这里有提示，远程origin仓库有一个新的分支dev

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git branch  查看本地分支
* master
  sortdev02

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git branch -r   查看远程分支
  origin/HEAD -> origin/master
  origin/dev
  origin/master
```

【step 3】小张自己的git bash上创建了一个本地的localdev分支，注意在本地创建的分支，需要设置跟踪哪个远程分支（拉远程仓库的主干分支，默认就在本地创建了一个master分支，并追踪了远程的origin/master分支，但其它分支的追踪关系，就需要自己设置了），这样在localdev分支推送代码的时候就简单了

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (master)
$ git checkout -b localdev origin/dev  创建并切换到新分支localdev，并让localdev追踪远程仓库
Switched to a new branch 'localdev'    origin/dev分支
Branch 'localdev' set up to track remote branch 'dev' from 'origin'.

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (localdev)
$ git branch -vv  可以通过该命令查看本地分支和远程分支的追踪关系
* localdev  47544ca [origin/dev] 解决readme文件合并冲突
  master    47544ca [origin/master] 解决readme文件合并冲突
  sortdev02 ca049aa readme添加222222
```

可以看到本地master追踪远程的origin/master，本地localdev分支追踪远程的origin/dev分支，都有对应关系。

【step 4】小张在本地localdev分支修改代码，直接推送到远程origin/dev分支中

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (localdev)
$ git add .

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (localdev)
$ git commit -m "给readme添加444444"
[localdev 010be0a] 给readme添加444444
 1 file changed, 1 insertion(+)
```

```
shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (localdev)
$ git push origin localdev:dev
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 356 bytes | 356.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/fixbug666/firstsharedproject.git
   47544ca..010be0a  localdev -> dev

shilei@DESKTOP-G21S8HU MINGW64 /d/代码/github/firstsharedproject (localdev)
```

把本地localdev分支的代码推送到远程origin仓库的dev分支上，
如果本地分支的名字也是dev，那么命令可以简写为
git push origin dev

> 创建远程分支和删除远程分支一般员工是没有权限的，所以此处的命令就不罗列了，大家感兴趣可以在网上查阅，这个操作只能由管理员来执行。如果是搭建自己的git私服代码托管，那就可以随便折腾了
>
> 查看远程仓库名称：git remote　一般远程仓库默认的名字是origin
>
> 查看本地分支：git branch
>
> 查看远程分支：git branch -r
>
> 查看本地分支和远程分支的追踪关系：git branch -vv
>
> 创建本地分支并指定追踪哪个远程分支：git checkout -b <本地分支名> <远程仓库名>/<远程分支名>
>
> 设置已经存在的本地分支追踪哪个远程分支：git branch -u <远程仓库名>/<远程分支名>