

Git 分支操作

知识点

- 添加 SSH 关联授权
- 为 Git 命令设置别名
- git fetch 刷新本地分支信息
- 创建新的本地分支
- 将新分支中的提交推送至远程仓库
- 本地分支跟踪远程分支
- 删除远程分支
- 本地分支的更名与删除

一、添加 SSH 关联授权

上一节操作中每次提交都要手动输入用户名和密码，若想避免这些麻烦，可以在系统中创建 SSH 公私钥，并将公钥放到 GitHub 指定位置。如此操作即可生成 GitHub 账户对于当前系统中的 Git 授权。

终端执行 `ssh-keygen` 命令按几次回车生成公私钥，公私钥存放在家目录下的隐藏目录 `.ssh` 中的两个文件中：

```

shiyanolou:shiyanolou/ (master) $ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/shiyanolou/.ssh/id_rsa):
Created directory '/home/shiyanolou/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/shiyanolou/.ssh/id_rsa.
Your public key has been saved in /home/shiyanolou/.ssh/id_rsa.pub.
The key fingerprint is:
be:33:7b:fc:09:0a:b6:70:76:a1:56:13:9b:e6:c3:62 shiyanolou@73671e15370e
The key's randomart image is:
+--[ RSA 2048 ]-----+

```

```

|
|      .
|      +
|      S
|     B o
|    . E *..
|   B =o+o. .
|  . += .o
|
+-----+

```

```

shiyanolou:shiyanolou/ (master) $ tree ~/.ssh
/home/shiyanolou/.ssh
├── id_rsa
└── id_rsa.pub

```

0 directories, 2 files

```
shiyanolou:shiyanolou/ (master) $
```



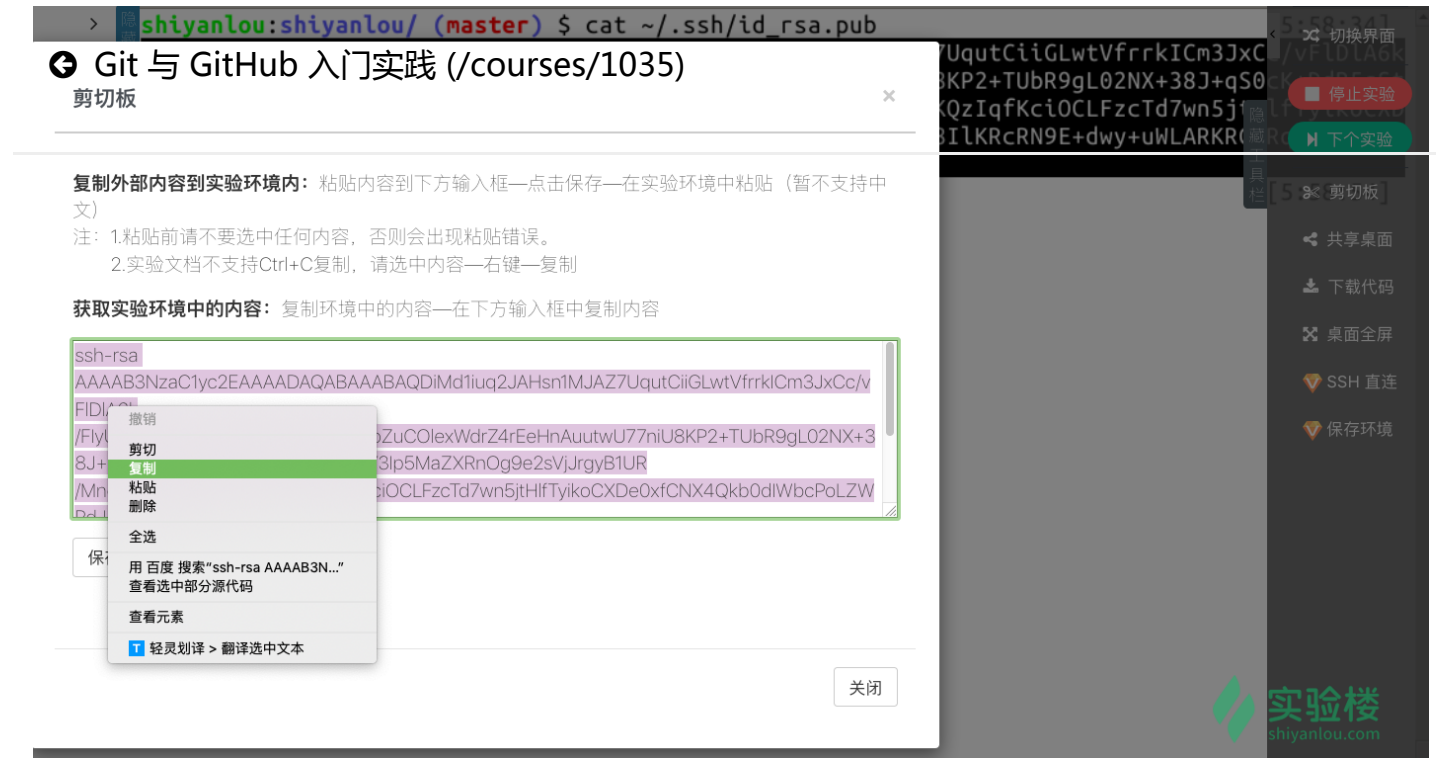
将 ~/.ssh/id_rsa.pub 文件中的公钥内容复制出来，实验环境中可以使用右侧工具栏中的剪切板复制：

```

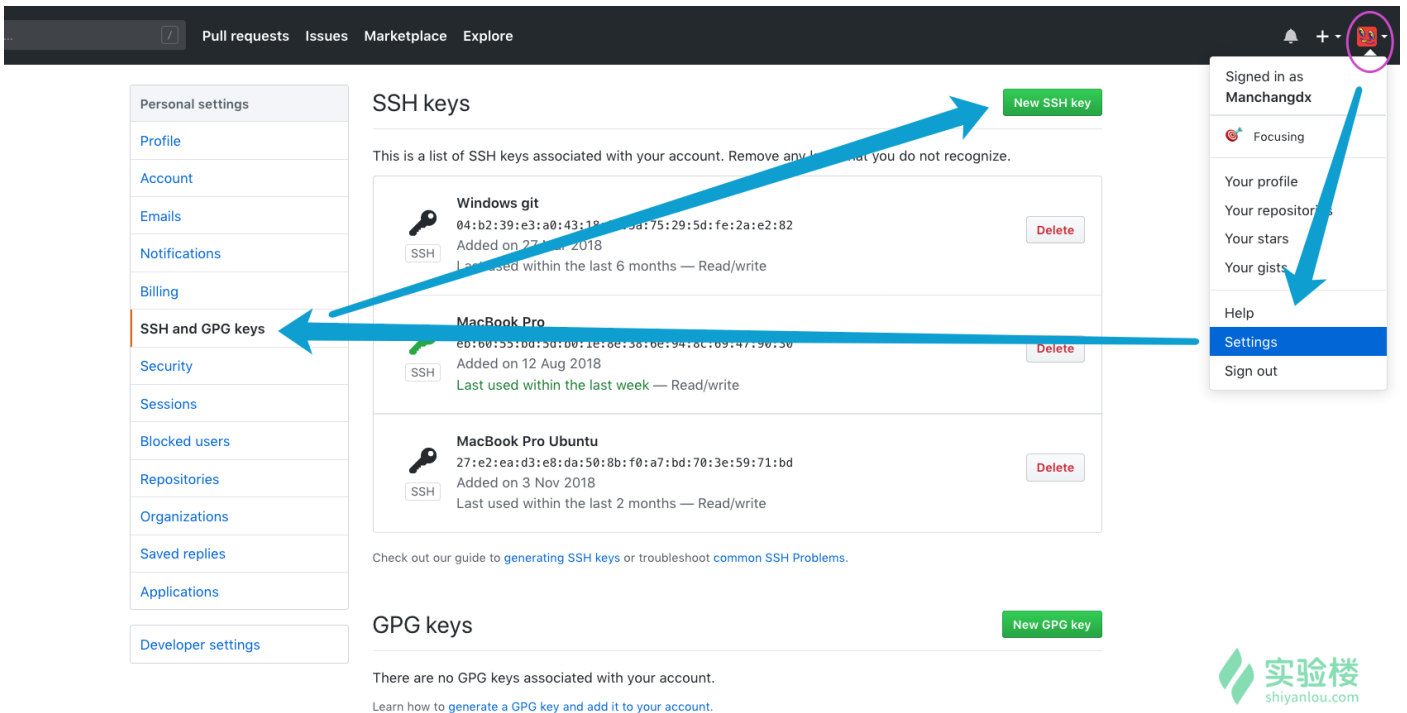
shiyanolou:shiyanolou/ (master) $ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADiMd1iuq2JAHsn1MJAZ7UqtCiiGLwtVfrrkICm
/FIyUnTNXJzdqoy49QMIeVGwEABEpZuCOlexWdrZ4rEeHnAuutwU77niU8KP2+TubR9gL02NX+38J
UXHtVEMUkmY3Ip5MaZXRn0g9e2sVjJrgyB1UR/Mne4J00ZUSC3030qgU5KQzIqfKci0CLFzcTd7wn
e0xfCNX4Qkb0dIWbcPoLZWRdJlwF6gKa9+BA/IVP...q2068I1KRcRN9E+dwY+uWLAR
gckXkd5kjcuCtzSB shiyanolou@73671e15370e
shiyanolou:shiyanolou/ (master) $

```





然后在 GitHub 网页上添加公钥：



Title 自定义，把剪切板中的内容粘贴到 Key 中，点击绿色按钮添加 SSH Key 即可：

SSH keys / Add new

Git 与 GitHub 入门实践 (courses/1035)

Personal settings

- Profile
- Account
- Emails
- Notifications
- Billing
- SSH and GPG keys
- Security
- Sessions
- Blocked users
- Repositories

Title

实验楼

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQADiMd1iuq2JAHsn1MJAZ7UqutCiiGLwtVfrrklCm3JxCc/vFDIA6k
/FlyUnTNXJzdqoy49QMieVGwEABEpZuCOlexWdrZ4rEeHnAautwU77niU8KP2+TUbR9gL02NX+38J+qS0cK+Dd
R5qStUXHtVEMUkmY3lp5MaZXRnOg9e2sVjJrgyB1UR
/Mne4JO0ZUsC3O3OqgU5KQzlfKciOCLFzcTd7wn5jtHlfTyikoCXDe0xfCNX4Qkb0dlWbcPoLZWRdJlwF6gKa9
+BA/IVP7xpvQSvu9U9mqQ2068llKRCrN9E+dwY+uWLARKRCARqYd1cfGCggckXkd5kjuCtzSB
shianlou@73671e15370e
```

Add SSH key

实验楼 shiyanlou.com

回到仓库主目录，点击下图所示的绿色按钮，点击紫色框中的 “Use SSH”，然后复制这个链接。

Manchangdx / shiyanlou

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Git 课程测试 Edit

Manage topics

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Create new file Upload files Find file Clone or download

Manchangdx commit file one.txt

README.md commit file one.txt

one.txt commit file one.txt

README.md

shiyanlou

Git 课程测试 Git 操作其实很简单，只需多加练习，在实际应用中逐渐领悟它的设计哲学。

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/Manchangdx/shiyanl

Open in Desktop Download ZIP

实验楼 shiyanlou.com

在实验环境里删除原仓库，使用此链接重新克隆仓库。克隆仓库是需要确认连接，输入 yes 即可：

```

shiyancelou:shiyancelou/ (master) $ cd
shiyancelou:~/ $ git clone git@github.com:Manchangdx/shiyancelou
fatal: 仓库 'git@github.com:Manchangdx/shiyancelou' 不存在
shiyancelou:~/ $ git clone git@github.com:Manchangdx/shiyancelou
正克隆到 'shiyancelou'...
The authenticity of host 'github.com (13.229.188.59)' can't be established.
RSA key fingerprint is 16:27:ac:a5:76:28:3d:36:63:1b:56:4d:eb:df:a6:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,13.229.188.59' (RSA) to the list of known hosts.
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 7 (delta 0), reused 4 (delta 0), pack-reused 0
接收对象中: 100% (7/7), 完成.
shiyancelou:~/ $

```



重要的一点：只有使用这种 git 开头的地址克隆仓库，SSH 关联才会起作用。

使用 SSH 的好处主要有两点：

- 免密码推送，执行 git push 时不再需要输入用户名和密码了；
- 提高数据传输速度。

它不是必须的，比如在实验楼的课程中挑战环境是不可保存的，一次性的，这种环境就没有必要创建 SSH 了，因为相较好处来说，还是太麻烦了。

二、为 Git 命令设置别名

上一节课程中的操作，有些命令的重复度极高，比如 git status 和 git branch -avv 等，Git 可以对这些命令设置别名，以便简化对它们的使用，设置别名的命令是 git config --global alias. [别名] [原命令]，如果原命令中有选项，需要加引号。别名是自定义的，可以随意命名，设置后，原命令和别名具有同等作用。操作如下：

```

shiyancelou:shiyancelou/ (master) $ git config --global alias.st status
shiyancelou:shiyancelou/ (master) $ git config --global alias.br 'branch -avv'
shiyancelou:shiyancelou/ (master) $ git config --global alias.com 'commit -m'
shiyancelou:shiyancelou/ (master) $ cat -n ~/.gitconfig
1  [user]
2      email = 1195581533@qq.com
3      name = Manchangdx
4  [alias]
5      st = status
6      br = branch -avv
7      com = commit -m
shiyancelou:shiyancelou/ (master) $ git st
位于分支 master
您的分支与上游分支 'origin/master' 一致。

无文件要提交，干净的工作区
shiyancelou:shiyancelou/ (master) $

```

自己设置的别名要记住，也可以使用 `git config -l` 命令查看配置文件。下面文档中的命令将使用这些别名。

三、Git 分支管理

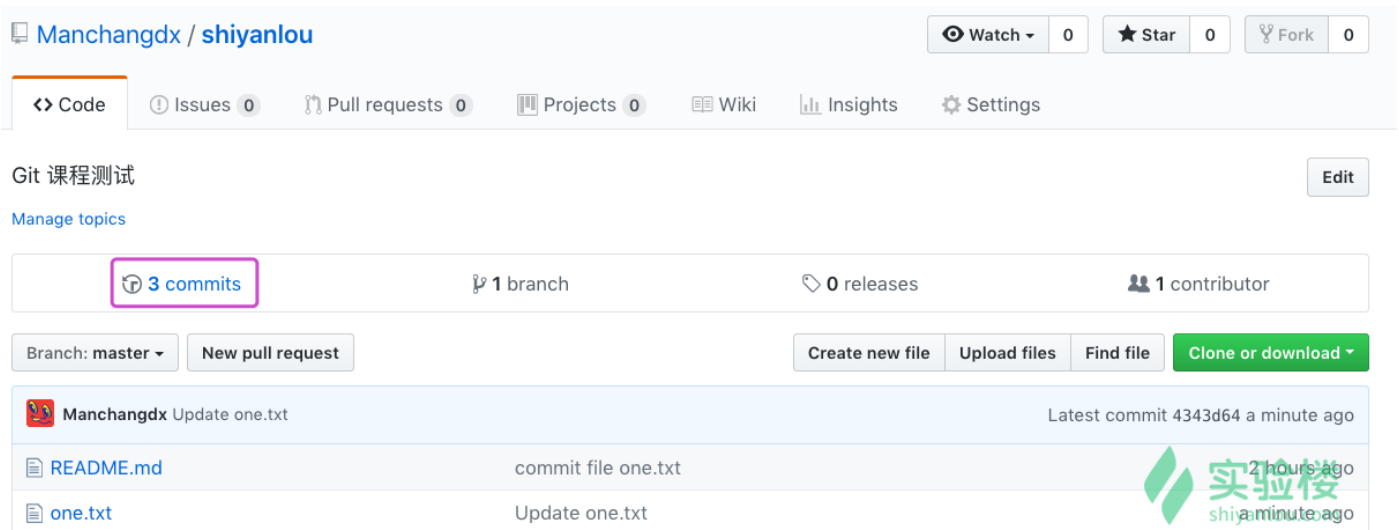
下面介绍 Git 作为分布式版本控制器最强大的功能：分支管理。

3.1 git fetch 刷新本地分支信息

在介绍分支前，先讲解另一个命令 `git fetch`，它的作用是将远程仓库的分支信息拉取到本地仓库，注意，仅仅是更新了本地的远程分支信息，也就是执行 `git branch -avv` 命令时，查看到的 `remotes` 开头的行的分支信息。

举例说明一下，首先我们在 GitHub 页面上对 `one.txt` 文件进行修改并增加一次提交。

提交完成后，提交数变成 3 个，点下图紫色框中的链接可以看到提交记录：



Manchangdx / shiyanlou

Git 与 GitHub 入门实践 (/courses/1035)

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master


Commits on Jan 28, 2019

- Update one.txt
Manchangdx committed 3 minutes ago
Verified 4343d64
- commit file one.txt
Manchangdx committed 2 hours ago
e290005

Commits on Jan 27, 2019

- Initial commit
Manchangdx committed a day ago
Verified 2b96af0

Newer Older

 实验楼
shiyanlou.com

在实验环境中执行 `git fetch` 命令，然后执行 `git branch -avv` 查看分支信息：

```
shiyanlou:shiyanlou/ (master) $ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
展开对象中: 100% (3/3), 完成.
来自 github.com:Manchangdx/shiyanlou
```


```
e290005..4343d64 master -> origin/master
shiyanlou:shiyanlou/ (master) $ git branch -avv
shiyanlou:shiyanlou/ (master) $
```

```
* master e290005 [origin/master: 落后 1] commit file one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/master 4343d64 Update one.txt
(END)
```

可以看到，本地分支 `master` 的版本号无变化，而远程分支已经更新。所以，`fetch` 命令的作用是刷新保存在本地仓库的远程分支信息，此命令需要联网。此时若想使本地 `master` 分支的提交版本为最新，可以执行 `git pull` 命令来拉取远程分支到本地，`pull` 是拉取远程仓库的数据到本地，需要联网，而由于前面执行过 `git fetch` 命令，所以也可以执行 `git rebase origin/master` 命令来实现“使本地 `master` 分支基于远程仓库的 `master` 分支”，`rebase` 命令在后面还会经常用到，这里只需按部就班操作即可：

```
shiyanlou:shiyanlou/ (master) $ git rebase origin/master
首先，回退头指针以便在其上重放您的工作...
快进 master 到 origin/master.
shiyanlou:shiyanlou/ (master) $ git br
shiyanlou:shiyanlou/ (master) $
```

```
* master 4343d64 [origin/master] Update one.txt
Git 与 GitHub 入门实践 (/courses/1035)
remotes/origin/HEAD -> origin/master
remotes/origin/master 4343d64 Update one.txt
(END)
```




可以看到，远程仓库 master 分支、本地仓库的 origin/master 分支、本地仓库的 master 分支已经一致。

3.2 创建新的本地分支

分支在项目开发中作用重大，多人协作时尤其不可或缺。例如一个项目上线了 1.0 版本，研发部门需要开发 1.1、1.2 两个测试版，增加不同的新功能，测试版的代码显然不能在正式版所在的分支上，此时需要新的分支来存放不同版次的代码。再例如实验楼的课程团队在维护课程仓库时，每个人都有各自的分支，在自己的分支上进行修改，然后向 master 分支提 PR (pull request)，最后从 master 分支推送到线上。

首先，克隆远程仓库到本地，进入仓库主目录，执行 `git br` 查看分支信息：


```
* master 4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/master 4343d64 Update one.txt
(END)
```



执行 `git branch [分支名]` 可以创建新的分支：

```
shianlou:shianlou/ (master) $ git branch dev
shianlou:shianlou/ (master) $ git br
shianlou:shianlou/ (master) $
```

```
dev 4343d64 Update one.txt
* master 4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/master 4343d64 Update one.txt
(END)
```



此命令创建新分支后并未切换到新分支，还是在 master 分支上，执行 `git checkout [分支名]` 切换分支，`checkout` 也是常用命令，先给它设置别名，然后切换分支：


```

shiyanolou:shiyanolou/ (master) $ git config --global alias.ch checkout
shiyanolou:shiyanolou/ (master) $ cat -n ~/.gitconfig
1  [user]
2      email = 1195581533@qq.com
3      name = Manchangdx
4  [alias]
5      st = status
6      br = branch -avv
7      com = commit -m
8      ch = checkout
shiyanolou:shiyanolou/ (master) $ git ch dev
切换到分支 'dev'
shiyanolou:shiyanolou/ (dev) $ █

```



创建新分支还要手动切换太麻烦，介绍另一个常用的命令 `git checkout -b [分支名]` 创建分支并切换到新分支：

```

shiyanolou:shiyanolou/ (dev) $ git ch -b dev1
切换到一个新分支 'dev1'
shiyanolou:shiyanolou/ (dev1) $ git br
shiyanolou:shiyanolou/ (dev1) $ █

```



```

dev          4343d64 Update one.txt
* dev1       4343d64 Update one.txt
master       4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/master 4343d64 Update one.txt
(END) █

```



如上图所示的分支信息，前两行是新建的本地分支信息，它们的版本号与主分支 `master` 一致，这是因为在哪个分支上创建新分支，新分支的提交记录就与哪个分支一致。新建分支并无跟踪任何远程分支，所以没有 `master` 分支中的中括号和括号内的蓝色远程分支名。

假设我们要在当前分支 `dev1` 上开发一个新的功能，需要增加一个文件 `new_func1`，然后生成一个新的提交：

```

shiyanolou:shiyanolou/ (dev1) $ echo '新功能测试 ~~' > new_func1
shiyanolou:shiyanolou/ (dev1*) $ git add .
shiyanolou:shiyanolou/ (dev1*) $ git com '新功能测试 ~~'
[dev1 7c1a41c] 新功能测试 ~~
1 file changed, 1 insertion(+)
create mode 100644 new_func1
shiyanolou:shiyanolou/ (dev1) $ git br
shiyanolou:shiyanolou/ (dev1) $ █

```



```

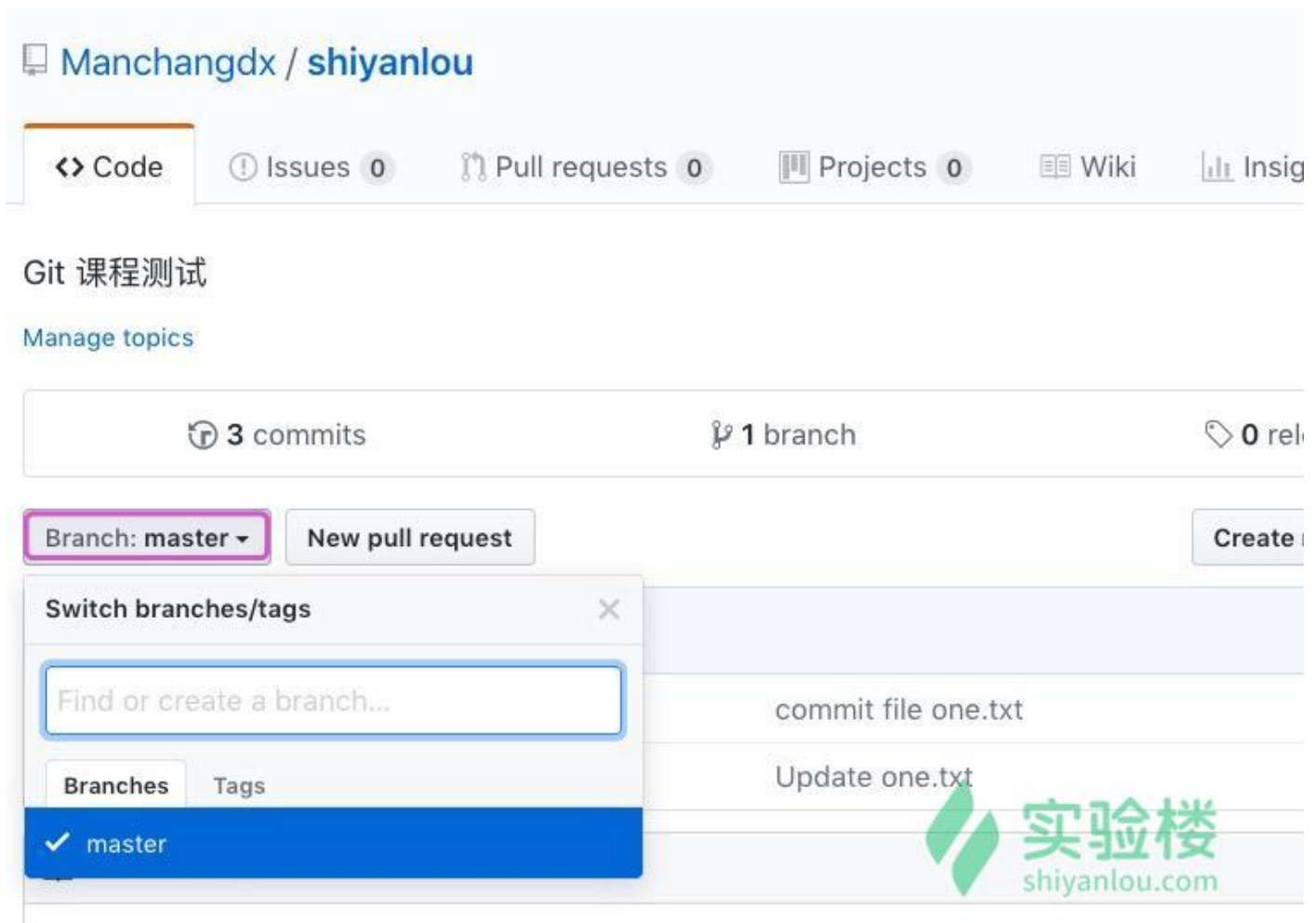
dev
* 4343d64 Update one.txt
dev 新功能测试 ~~
master 4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/master 4343d64 Update one.txt
(END)

```



3.3 将新分支中的提交推送至远程仓库

好，新功能已经写好并提交到了版本区，现在要推送了，推送到哪里呢？正常逻辑当然要推送到远程仓库的同名分支，不过现在远程仓库里只有一个分支：



上图紫色框中是一个下拉按钮，点击后显示仓库中的全部分支，按钮上显示的是当前所在分支。

执行 `git push [主机名] [本地分支名]:[远程分支名]` 即可将本地分支推送到远程仓库的分支中，通常冒号前后的分支名是相同的，如果是相同的，可以省略 `:[远程分支名]`，如果远程分支不存在，会自动创建：

```

shiyanolou:shiyanolou/ (dev1) $ git push origin dev1:dev1
Warning: Permanently added the IP address '13.250.177.223'
to the list of
known hosts.
枚举对象：4，完成。
对象计数中：100% (4/4)，完成。
使用 4 个线程进行压缩
压缩对象中：100% (2/2)，完成。
写入对象中：100% (3/3)，354 bytes | 354.00 KiB/s，完成。
总共 3 （差异 0），复用 0 （差异 0）
remote:
remote: Create a pull request for 'dev1' on GitHub by visiting:
remote:      https://github.com/Manchangdx/shiyanolou/pull/new/dev1
remote:
To github.com:Manchangdx/shiyanolou
 * [new branch]      dev1 -> dev1
shiyanolou:shiyanolou/ (dev1) $

```



上图命令可以简写为 `git push origin dev1`。注意哦，这是我们创建 SSH 关联后第一次执行 `push` 命令，可以看到传输速度有明显的提高，更重要的是，不再需要重复输入用户名和密码了，另外打印信息的第一行是警告信息，因为是这个分支的第一次推送嘛，下次执行推送就不会再出现了。现在执行 `git br` 查看一下分支情况：

```

dev          4343d64 Update one.txt
* dev1       7c1a41c 新功能测试 ~~
master       4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/dev1 7c1a41c 新功能测试 ~~
remotes/origin/master 4343d64 Update one.txt
(END)

```



可以看到，远程分支 `origin/dev1` 的信息已经在本地存在，且与本地同名分支一致。再看下 GitHub 页面的情况：

很好，与预期毫无二致。

📌 Git 与 GitHub 入门实践 (/courses/1035)

3.4 本地分支跟踪远程分支

现在有个问题，当我们再次在 dev1 分支上修改并提交，推送到远程仓库时还是要输入上面的那个长长的命令，好不方便。如果能和 master 分支一样跟踪远程同名分支，就可以直接使用 `git push` 命令推送了。有办法的，执行这个命令 `git branch -u [主机名/远程分支名] [本地分支名]` 将本地分支与远程分支关联，或者说使本地分支跟踪远程分支。如果是设置当前所在分支跟踪远程分支，最后一个参数本地分支名可以省略不写：

```
shiyanolou:shiyanolou/ (dev1) $ git branch -u origin/dev1
分支 'dev1' 设置为跟踪来自 'origin' 的远程分支 'dev1'。
shiyanolou:shiyanolou/ (dev1) $ git br
shiyanolou:shiyanolou/ (dev1) $
```



```

dev          4343d64 Update one.txt
* dev1       7c1a41c [origin/dev1] 新功能测试 ~~
master      4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/dev1 7c1a41c 新功能测试 ~~
remotes/origin/master 4343d64 Update one.txt
(END)
```



这个命令的 `-u` 选项是 `--set-upstream` 的缩写。可不可以让本地分支跟踪远程非同名分支呢？可以的，尽管几乎遇不到这种自找麻烦的需求。可不可以撤销本地分支对远程分支的跟踪呢？也是可以的，执行 `git branch --unset-upstream [分支名]` 即可撤销该分支对远程分支的跟踪，同样地，如果撤销当前所在的分支的跟踪，分支名可以省略不写：

```
shiyanolou:shiyanolou/ (dev1) $ git branch --unset-upstream
shiyanolou:shiyanolou/ (dev1) $ git br
shiyanolou:shiyanolou/ (dev1) $
```



```

dev          4343d64 Update one.txt
* dev1       7c1a41c 新功能测试 ~~
master      4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/dev1 7c1a41c 新功能测试 ~~
remotes/origin/master 4343d64 Update one.txt
(END)
```



问题又来了，前面的操作是先将本地分支推送到远程仓库，使远程仓库创建新分支，然后再执行命令使本地分支跟踪远程分支，有没有办法在推送时就自动跟踪远程分支呢？有的，在推送的时候，加个 `--set-upstream` 或其简写 `-u` 选项即可，现在切换到 dev 分支试一下这个命令：


```

shiyanolou:shiyanolou/ (dev) $ git push -u origin dev
Warning: Permanently added the RSA host key for IP address '52.74.223.
known hosts.
总共 0 (差异 0) , 复用 0 (差异 0)
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:      https://github.com/Manchangdx/shiyanolou/pull/new/dev
remote:
To github.com:Manchangdx/shiyanolou
 * [new branch]      dev -> dev
分支 'dev' 设置为跟踪来自 'origin' 的远程分支 'dev'。
shiyanolou:shiyanolou/ (dev) $ git br
shiyanolou:shiyanolou/ (dev) $ █

```



```

* dev          4343d64 [origin/dev] Update one.txt
dev1          7c1a41c 新功能测试~~
master       4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/dev  4343d64 Update one.txt
remotes/origin/dev1 7c1a41c 新功能测试~~
remotes/origin/master 4343d64 Update one.txt
(END) █

```



3.5 删除远程分支

接下来，介绍一下删除分支的方法。

首先，删除远程分支，使用 `git push [主机名] :[远程分支名]`，如果一次性删除多个，可以这样：`git push [主机名] :[远程分支名] :[远程分支名] :[远程分支名]`。此命令的原理是将空分支推送到远程分支，结果自然就是远程分支被删除。另一个删除远程分支的命令：`git push [主机名] -delete [远程分支名]`。删除远程分支的命令可以在任意本地分支中执行。两个命令分别试一下：

```

shiyanolou:shiyanolou/ (dev) $ git push origin :dev
To github.com:Manchangdx/shiyanolou
 - [deleted]      dev
shiyanolou:shiyanolou/ (dev) $ git push origin --delete dev1
To github.com:Manchangdx/shiyanolou
 - [deleted]      dev1
shiyanolou:shiyanolou/ (dev) $ git br
shiyanolou:shiyanolou/ (dev) $ █

```



```

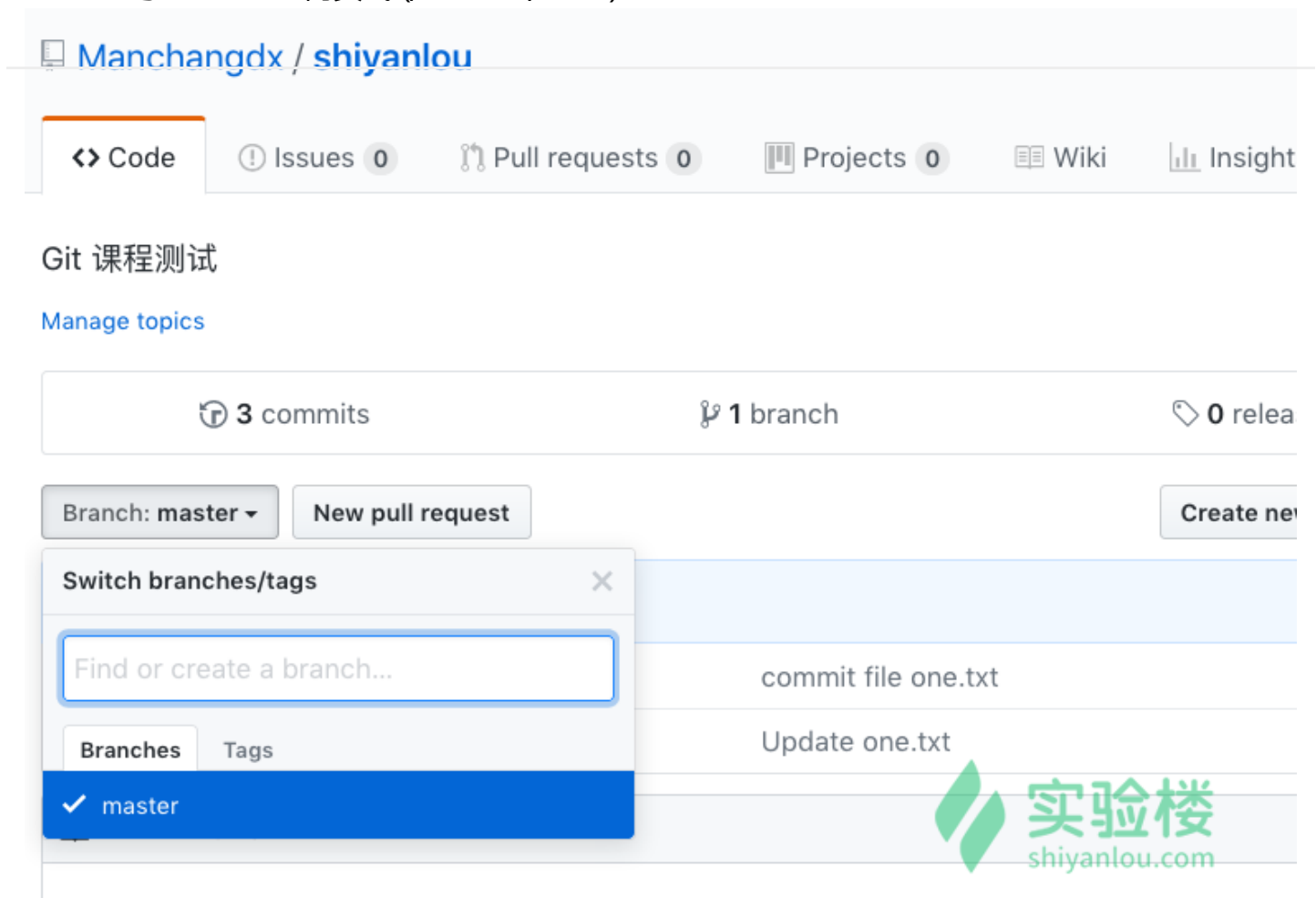
* dev          4343d64 [origin/dev: 丢失] Update one.txt
dev1          7c1a41c 新功能测试~~
master       4343d64 [origin/master] Update one.txt
remotes/origin/HEAD -> origin/master
remotes/origin/master 4343d64 Update one.txt
(END) █

```



可以看到本地仓库已经没有远程分支 dev 和 dev1 的分支信息。查看 GitHub 仓库页面：

👉 Git 与 GitHub 入门实践 (/courses/1035)



也只剩 master 一个分支。操作成功。

3.6 本地分支的更名与删除

回到实验环境，使用 `git branch -D [分支名]` 删除本地分支，同样地，此命令也可以一次删除多个，将需要删除的分支名罗列在命令后面即可。在此之前，先介绍一个极少用到的命令：给本地分支改名 `git branch -m [原分支名] [新分支名]`，若修改当前所在分支的名字，原分支名可以省略不写：

```
shiyanlou:shiyanlou/ (dev) $ git branch -m ved
shiyanlou:shiyanlou/ (ved) $
```

好，现在要一次性删除本地分支 ved 和 dev1。需要注意的一点：当前所在的分支不能被删除。切换到 master 分支，然后执行 `git branch -D ved dev1` 命令：


```
shiyanolou:shiyanolou/ (ved) $ git ch master
```

切换到分支 master

您的分支与上游分支 'origin/master' 一致。

```
shiyanolou:shiyanolou/ (master) $ git branch -D ved dev1
```

已删除分支 ved (曾为 4343d64)。

已删除分支 dev1 (曾为 7c1a41c)。

```
shiyanolou:shiyanolou/ (master) $
```



执行 `git branch -avv` 查看当前仓库分支状态：

```
* master          4343d64 [origin/master] Update one.txt
  remotes/origin/HEAD -> origin/master
  remotes/origin/master 4343d64 Update one.txt
(END)
```



很好，一切都回到了课程开始时的样子，就像什么都没有发生。本节课程就到这里。

四、总结

本节实验主要讲解了以下知识点：

- 添加 SSH 关联授权
- 为 Git 命令设置别名
- `git fetch` 刷新本地分支信息
- 创建新的本地分支
- 将新分支中的提交推送至远程仓库
- 本地分支跟踪远程分支
- 删除远程分支
- 本地分支的更名与删除

希望大家通过本节课程的学习，真正掌握创建分支、关联远程分支、撤销关联、删除分支等必备技能。下节课程我们将学习多人协作中的 Git & GitHub 操作流程。

*本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。

上一节：Git 基础操作 (/courses/1035/labs/9805/document)

下一节：多人协作 GitHub 部分 (/courses/1035/labs/9817/document)