

Git 与 GitHub 简介

知识点

- Git 与 GitHub 的来历
- 在 GitHub 上创建仓库
- 安装 Git、升级 Git 到最新版
- 克隆 GitHub 上的仓库到本地

一、Git 与 GitHub 的来历

Linux 之父 Linus 在 1991 年创建开源的 Linux 操作系统之后，多年来依靠全世界广大热心志愿者的共同建设，经过长足发展，现已成为世界上最大的服务器系统。系统创建之初，代码贡献者将源码文件发送给 Linus，由其手动合并。这种方式维持多年后，代码量已经庞大到人工合并难以为继，于是深恶集中式版本控制系统的 Linus 选择了一个分布式商业版本控制系统 BitKeeper，不过 Linux 社区的建设者们可以免费使用它。BitKeeper 改变了 Linus 对版本控制的认识，同时 Linus 发现 BitKeeper 有一些不足，而且有个关键性的问题使之不能被广泛使用，就是不开源。

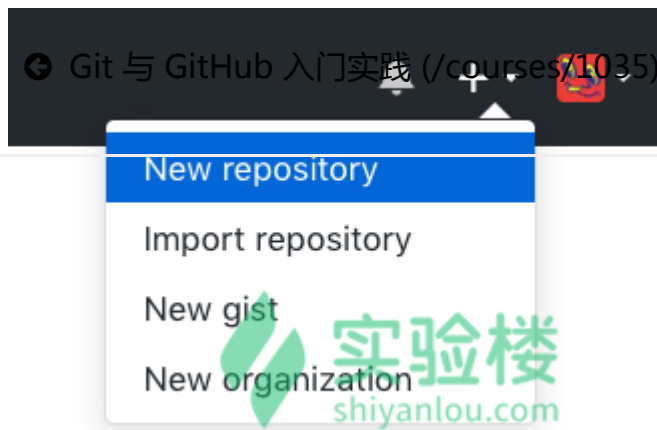
在 2005 年，BitKeeper 所在公司发现 Linux 社区有人企图破解它，BitKeeper 决定收回 Linux 社区的免费使用权。Linus 对此事调节数周无果，找遍了当时已知的各种版本控制系统，没有一个看上眼的，一怒之下决定自己搞一个。Linus 花了十天时间用 C 语言写好了一个开源的版本控制系统，就是著名的 Git。

2007 年旧金山三个年轻人觉得 Git 是个好东西，就搞了一个公司名字叫 GitHub，第二年上线了使用 Ruby 编写的同名网站 GitHub，这是一个基于 Git 的免费代码托管网站（有付费服务）。十年间，该网站迅速蹿红，击败了实力雄厚的 Google Code，成为全世界最受欢迎的代码托管网站。2018 年 6 月，GitHub 被财大气粗的 Microsoft 收购。2019 年 1 月 GitHub 宣布用户可以免费创建私有仓库。根据 2018 年 10 月的 GitHub 年度报告显示，目前有 3100 万开发者创建了 9600 万个项目仓库，有 210 万企业入驻。

本课程将以图文的形式逐步讲解 GitHub 的使用以及 Git 实现版本控制。

二、在 GitHub 上创建仓库

首先，打开 GitHub (<https://github.com/>) 注册个人账户并登录。登录后，在个人主页的右上角点击 New repository 创建新的仓库：



打开页面如下图所示，填入相关信息。注意下图紫色框中有两个下拉按钮，左边的用来选择忽略文件，右边的用来选择所属协议，这两项可以不选，后面的课程会讲到。

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name *

 **Manchangdx** / 

Great repository names are short and memorable. Need inspiration? How about **upgraded-train**.

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

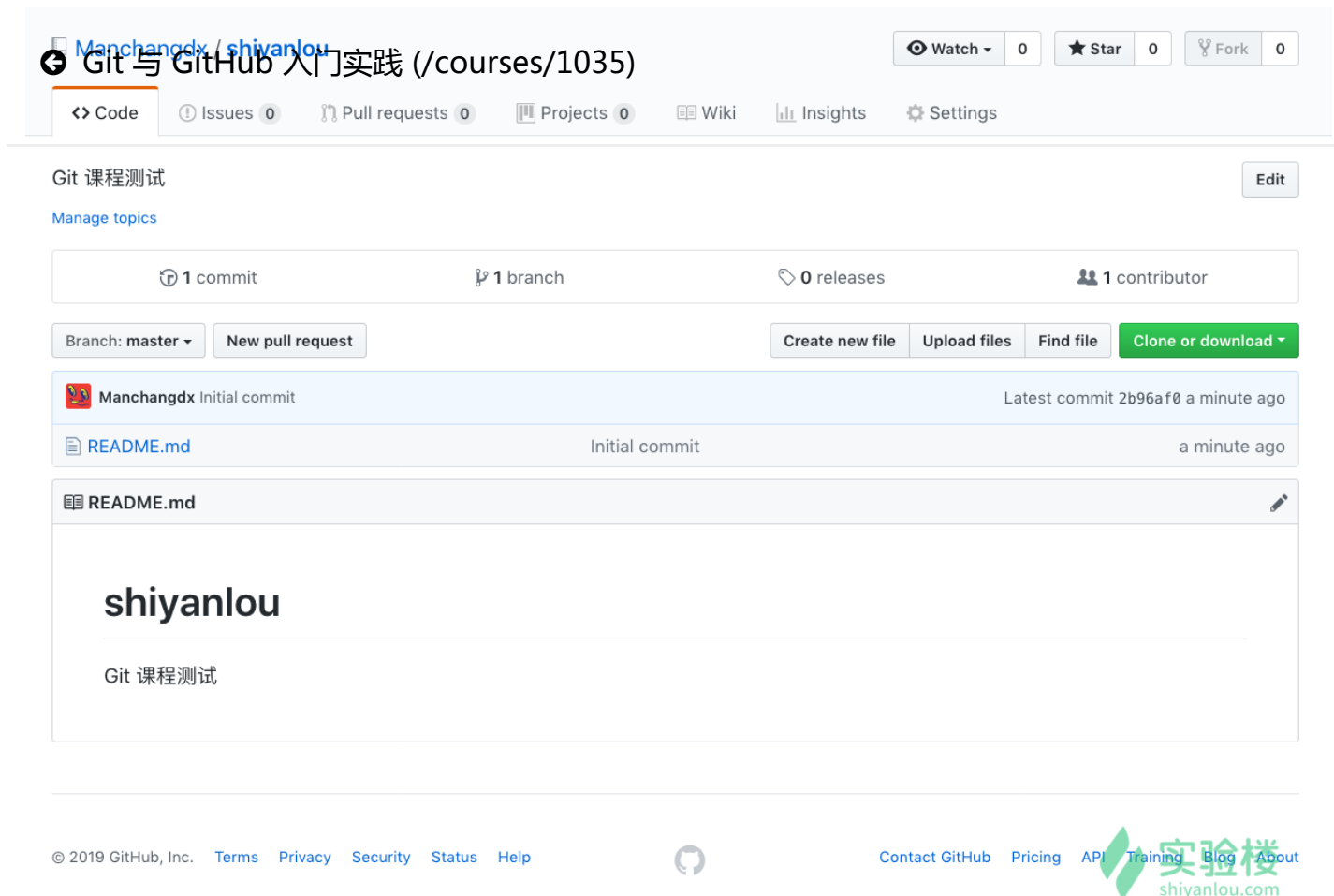
Add a license: **None** ▾



Create repository



点击绿色按钮创建新的仓库，成功后自动跳转到新建仓库的主页面，如下图所示：



The screenshot shows a GitHub repository page for 'Manchangdx / shiyanlou' with the path '/courses/1035'. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The main branch is 'master'. The repository contains a 'README.md' file. The commit history shows an 'Initial commit' by 'Manchangdx' a minute ago. The README content includes the title 'shiyanlou' and the subtitle 'Git 课程测试'. The footer of the page includes copyright information for 2019 GitHub, Inc. and links to Terms, Privacy, Security, Status, Help, Contact GitHub, Pricing, API, Training, Blog, and About. The '实验楼' (Shiyanlou) logo is also present.

三、安装 Git、升级 Git 到最新版

接下来，我们就要尝试使用这个仓库。

首先，打开实验环境。实验环境中内置了 Git 版本控制器，无需下载。打开终端使用 `git --version` 命令查看版本：

```
shiyanlou:~/ $ git --version
git version 1.9.1
shiyanlou:~/ $
```

这个版本有点儿低，先给它升升级，Ubuntu 系统中依次执行以下命令即可升级 Git 到最新版。

注意，升级版本这部分操作完全可以略过，升级不是必须的。我如果不是为了写课程，自己用实验环境的话，就不会升级，当前版本与最新版本区别不大。

首先，在终端执行如下命令下载安装 Git 所需的密钥：

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:git-core/ppa
```

```

shiyanolou:dist-packages/ $ sudo add-apt-repository ppa:git-core/ppa
Traceback (most recent call last):
  File "/usr/bin/add-apt-repository", line 11, in <module>
    from softwareproperties.SoftwareProperties import SoftwareProperti
  File "/usr/lib/python3/dist-packages/softwareproperties/SoftwareProp
in <module>
    import apt_pkg
ImportError: No module named 'apt_pkg'

```



出现此报错是因为实验环境中 Python 版本的问题，执行如下命令选择 Python3.4 即可：

```

sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.4 1
sudo update-alternatives --display python3    # 查看可选版本
sudo update-alternatives --config python3     # 选择 python3.4

```

```

shiyanolou:~/ $ sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3
.4 1
shiyanolou:~/ $ sudo update-alternatives --display python3                                [15:14:28]
python3 - 自动模式
链接目前指向 /usr/bin/python3.5
/usr/bin/python3.4 - 优先级 1
/usr/bin/python3.5 - 优先级 60
slave python3m: /usr/bin/python3.5m
目前“最佳”的版本为 /usr/bin/python3.5。
shiyanolou:~/ $ sudo update-alternatives --config python3                               [15:14:54]
有 2 个候选项可用于替换 python3 (提供 /usr/bin/python3)。

  选择      路径                      优先级  状态
-----
* 0         /usr/bin/python3.5              60      自动模式
  1         /usr/bin/python3.4              1       手动模式
  2         /usr/bin/python3.5              60      手动模式

要维持当前值[*]请按回车键，或者键入选择的编号：1
update-alternatives: using /usr/bin/python3.4 to provide /usr/bin/python3 (python3) in 手动
模式
shiyanolou:~/ $

```



重复前一个命令下载秘钥：

```
shiyanolou:~/ $ sudo add-apt-repository ppa:git-core/ppa
The most current stable version of Git for Ubuntu.
```

For release candidates, go to <https://launchpad.net/~git-core/+archive/ubuntu/ppa>
更多信息：<https://launchpad.net/~git-core/+archive/ubuntu/ppa>
按回车继续或者 Ctrl+c 取消添加

```
gpg: 钥匙环 '/tmp/tmpyn3vjdgz/secring.gpg' 已建立
gpg: 钥匙环 '/tmp/tmpyn3vjdgz/pubring.gpg' 已建立
gpg: 下载密钥 'E1DF1F24', 从 hkp 服务器 keyserver.ubuntu.com
gpg: /tmp/tmpyn3vjdgz/trustdb.gpg: 建立了信任度数据库
gpg: 密钥 E1DF1F24: 公钥 "Launchpad PPA for Ubuntu Git Maintaine
gpg: 合计被处理的数量: 1
gpg: 已导入: 1 (RSA: 1)
OK
```



执行 `sudo apt update` 更新源：

```
shiyanolou:~/ $ sudo apt update
忽略 http://mirrors.aliyuncs.com trusty InRelease
获取: 1 http://mirrors.aliyuncs.com trusty-updates InRelease [65.9 kB]
获取: 2 http://mirrors.aliyuncs.com trusty-security InRelease [65.9 kB]
忽略 http://mirrors.aliyuncs.com trusty/mongodb-org/3.4 InRelease
命中 http://mirrors.aliyuncs.com trusty Release.gpg
获取: 3 http://mirrors.aliyuncs.com trusty/mongodb-org/3.4 Release.gpg [801 B]
命中 http://mirrors.aliyuncs.com trusty Release
获取: 4 http://mirrors.aliyuncs.com trusty/mongodb-org/3.4 Release [2,495 B]
获取: 5 http://mirrors.aliyuncs.com trusty-updates/main Sources [526 kB]
获取: 6 http://mirrors.aliyuncs.com trusty-updates/restricted Sources [6,449 B]
获取: 7 http://mirrors.aliyuncs.com trusty-updates/universe Sources [285 kB]
获取: 8 http://mirrors.aliyuncs.com trusty-updates/main amd64 Packages [1,414 kB]
命中 http://ppa.launchpad.net trusty InRelease
错误 http://mirrors.aliyuncs.com trusty/mongodb-org/3.4 Release
```



执行 `sudo apt install -y git` 重新安装 Git：

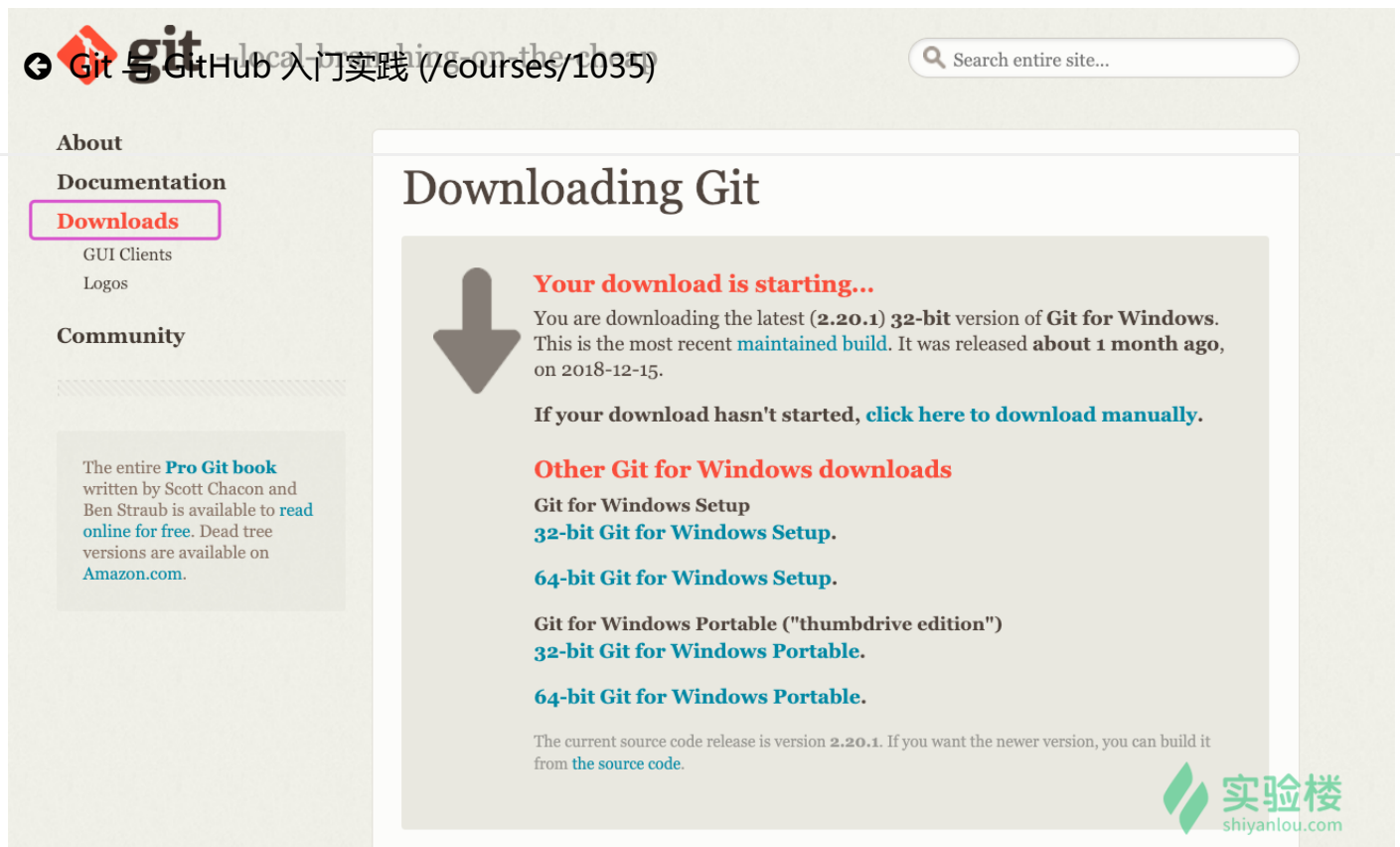
```
shiyanolou:~/ $ sudo apt install -y git
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
```



再次查看 Git 版本，升级成功：

```
shiyanolou:~/ $ git --version
git version 2.20.1
shiyanolou:~/ $
```

在 Windows 系统中可以安装 Git for Windows 客户端 (<https://git-scm.com/download/win>)



Git - GitHub 入门实践 (/courses/1035)

Search entire site...

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloading Git

Your download is starting...

You are downloading the latest (**2.20.1**) **32-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **about 1 month ago**, on 2018-12-15.

If your download hasn't started, [click here to download manually](#).

Other Git for Windows downloads

Git for Windows Setup

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Git for Windows Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

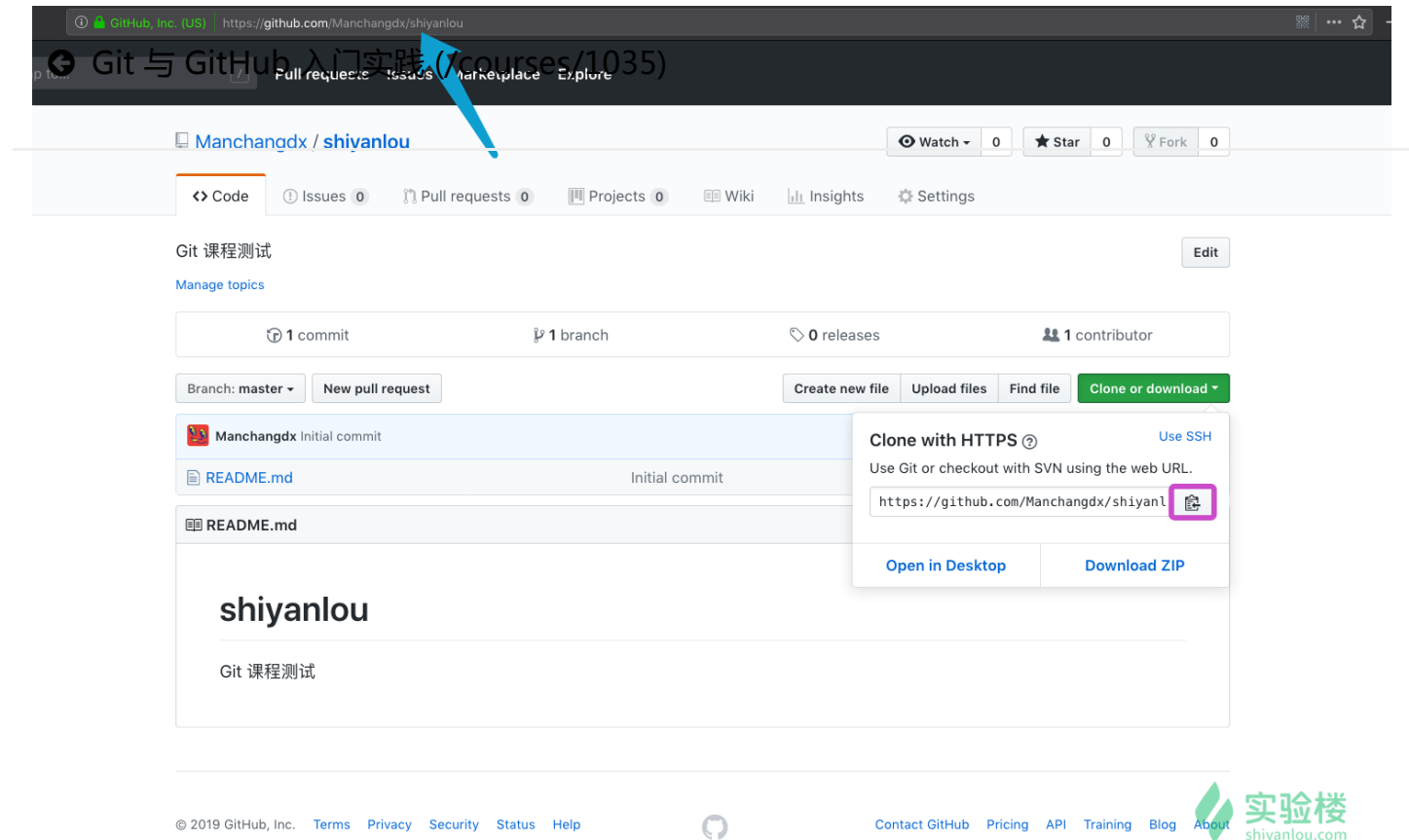
The current source code release is version **2.20.1**. If you want the newer version, you can build it from [the source code](#).

实验楼
shiyanlou.com

四、克隆 GitHub 上的仓库到本地

现在克隆前面我们在 GitHub 上创建的仓库，使用 `git clone + [仓库地址]` 命令即可，这是标准的克隆仓库命令。

点击下图绿色按钮，再点击紫色框中的按钮即可复制仓库地址，当然复制上面地址栏中的内容也是一样的。



克隆仓库到本地：

```
shiyanlou:~/ $ ll
总用量 12K
drwxrwxr-x 20 shiyanlou shiyanlou 4.0K 8月 21 2017 anaconda3
drwxrwxr-x 9 shiyanlou shiyanlou 4.0K 1月 27 02:19 Code
drwxrwxr-x 3 shiyanlou shiyanlou 4.0K 1月 17 16:59 Desktop
shiyanlou:~/ $ git clone https://github.com/Manchangdx/shiyanlou.git
正克隆到 'shiyanlou'...
remote: Enumerating objects: 3, done.
remote: Counting 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.
shiyanlou:~/ $ ll
总用量 16K
drwxrwxr-x 20 shiyanlou shiyanlou 4.0K 8月 21 2017 anaconda3
drwxrwxr-x 9 shiyanlou shiyanlou 4.0K 1月 27 02:19 Code
drwxrwxr-x 3 shiyanlou shiyanlou 4.0K 1月 17 16:59 Desktop
drwxrwxr-x 3 shiyanlou shiyanlou 4.0K 1月 27 02:20 shiyanlou
```

进入仓库主目录，如下图所示，仓库主目录中有个 `.git` 隐藏目录，它里面包含了仓库的全部信息，删掉这个目录，仓库就变成普通的目录了。进入到仓库目录中，命令行前缀发生了一些变化，出现了红色的 `master`，它就是当前所在的分支名：

```

shiyanolou:~/ $ cd shiyanolou
shiyanolou:shiyanolou/ (master) $ ls -al
总用量 16
drwxrwxr-x  3 shiyanolou shiyanolou 4096  1月 27 02:20 .
drwxr-xr-x 21 shiyanolou shiyanolou 4096  1月 27 02:39 ..
drwxrwxr-x  8 shiyanolou shiyanolou 4096  1月 27 02:39 .git
-rw-rw-r--  1 shiyanolou shiyanolou   29  1月 27 02:20 README.md
shiyanolou:shiyanolou/ (master) $

```

当我们在 GitHub 上创建一个仓库时，同时生成了仓库的默认主机名 origin，并创建了默认分支 master。GitHub 可以看成是免费的 Git 服务器，在 GitHub 上创建仓库，会自动生成一个仓库地址，主机就是指代这个仓库，主机名就等于这个仓库地址。克隆一个 GitHub 仓库（也叫远程仓库）到本地，本地仓库则会自动关联到这个远程仓库，执行 `git remote -v` 命令可以查看本地仓库所关联的远程仓库信息：

```

shiyanolou:shiyanolou/ (master) $ git remote -v
origin  https://github.com/Manchangdx/shiyanolou.git (fetch)
origin  https://github.com/Manchangdx/shiyanolou.git (push)
shiyanolou:shiyanolou/ (master) $

```

Git 要求对本地仓库关联的每个远程主机都必须指定一个主机名（默认为 origin），用于本地仓库识别自己关联的主机，`git remote` 命令就用于管理本地仓库所关联的主机，一个本地仓库可以关联任意多个主机（即远程仓库）。

克隆远程仓库到本地时，还可以使用 `-o` 选项修改主机名，在地址后面加上一个字段作为本地仓库的主目录名，举例如下：

```

shiyanolou:shiyanolou/ (master) $ cd [3:05:23]
shiyanolou:~/ $ git clone -o originnn https://github.com/Manchangdx/shiyanolou.git haha
正克隆到 'haha'...
remote: Enumerating objects: 3, done.
remote: Counting 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.
shiyanolou:~/ $ ll [3:05:52]
总用量 20K
drwxrwxr-x 20 shiyanolou shiyanolou 4.0K  8月 21  2017 anaconda3
drwxrwxr-x  9 shiyanolou shiyanolou 4.0K  1月 27 02:19 Code
drwxrwxr-x  3 shiyanolou shiyanolou 4.0K  1月 17 16:59 Desktop
drwxrwxr-x  3 shiyanolou shiyanolou 4.0K  1月 27 03:05 haha
drwxrwxr-x  3 shiyanolou shiyanolou 4.0K  1月 27 02:20 shiyanolou
shiyanolou:~/ $ cd haha [3:05:56]
shiyanolou:haha/ (master) $ git remote -v [3:05:59]
originnn  https://github.com/Manchangdx/shiyanolou.git (fetch)
originnn  https://github.com/Manchangdx/shiyanolou.git (push)
shiyanolou:haha/ (master) $ [3:06:02]

```

另一个在其它 Git 教程中常见的命令 `git init`，它会把当前所在目录变成一个本地仓库，因为有 GitHub 的存在，这个命令在我们的生产生活中用到的次数应该是零，除非你想费时费力自己搭建服务器。操作截图如下：


```
shiyanolou:~/ $ git init
已初始化 Git 仓库
shiyanolou:~/ $ ls -la
.          .codebox      .ipython      .vim
..         .config        .lessht       .viminfo
anaconda3 .dbus          .local        .viminfo.tmp
.app.py.swp .dbus         .mozilla      .viminfz.tmp
.a.py.swp   Desktop      .mysql_history .vimrc
.bash_history .gconf       .npm          .vnc
.bash_logout .gemrc       .oh-my-zsh    .Xauthority
.bashrc     .gitconfig   .pip          .zcompdump
.bashrc-anaconda3.bak haha         .profile      .zcompdump-7367:
.cache      .hushlogin   .pydistutils.cfg .zsh_history
.client.py.swp .ICEauthority .scim         .zshrc
Code        .idlerc      shiyanolou    .zsh-update
shiyanolou:~/ (master*) $ rm -rf .git
shiyanolou:~/ $
```



五、总结

本节实验主要介绍了：

- Git 和 GitHub 的来历
- 使用 GitHub 创建仓库
- 安装 / 更新 Git
- 使用 Git 克隆远程仓库到本地

本节主要介绍了 Git 与 GitHub 概述，以及创建仓库、克隆仓库到本地的基本操作，相对比较简单。大家在学习时重复操作几次即可熟练掌握。下一节将学习 Git 的使用流程，完成一次修改、提交、推送操作。

*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。

下一节：Git 基础操作 (/courses/1035/labs/9805/document)