

3GIT 版本控制软件

目标

- 理解版本控制能进行代码的托管
- 能够安装好 git 环境
- 记忆 git 工作的基本流程
- 能够使用 git 进行本地的提交
- 能够使用 git 撤销文件
- 理解 git 分支，创建分支，切换分支
- 能够注册 github 账号，并且创建仓库
- 理解多人开发的使用流程，并且能够解决冲突
- 能够使用SSH免密登录
- 能够配置 git 清单文件

版本控制概念

什么是版本管理

版本管理是一种记录文件变化的方式，以便将来查阅特定版本的文件内容。



20220404最新修
改.xlsx



20220501更新.
xlsx



20220502更新.
xlsx



20220502会议使
用.xlsx



20220502文档.
xlsx



20220504最新.
xlsx



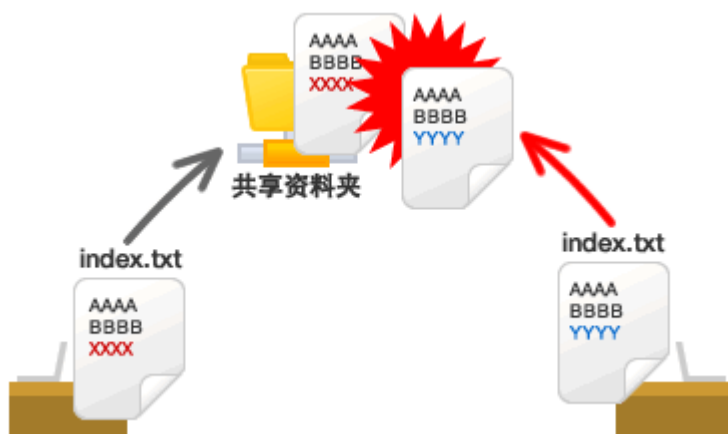
20220504最新复
制版.xlsx



20220608备份
用.xlsx

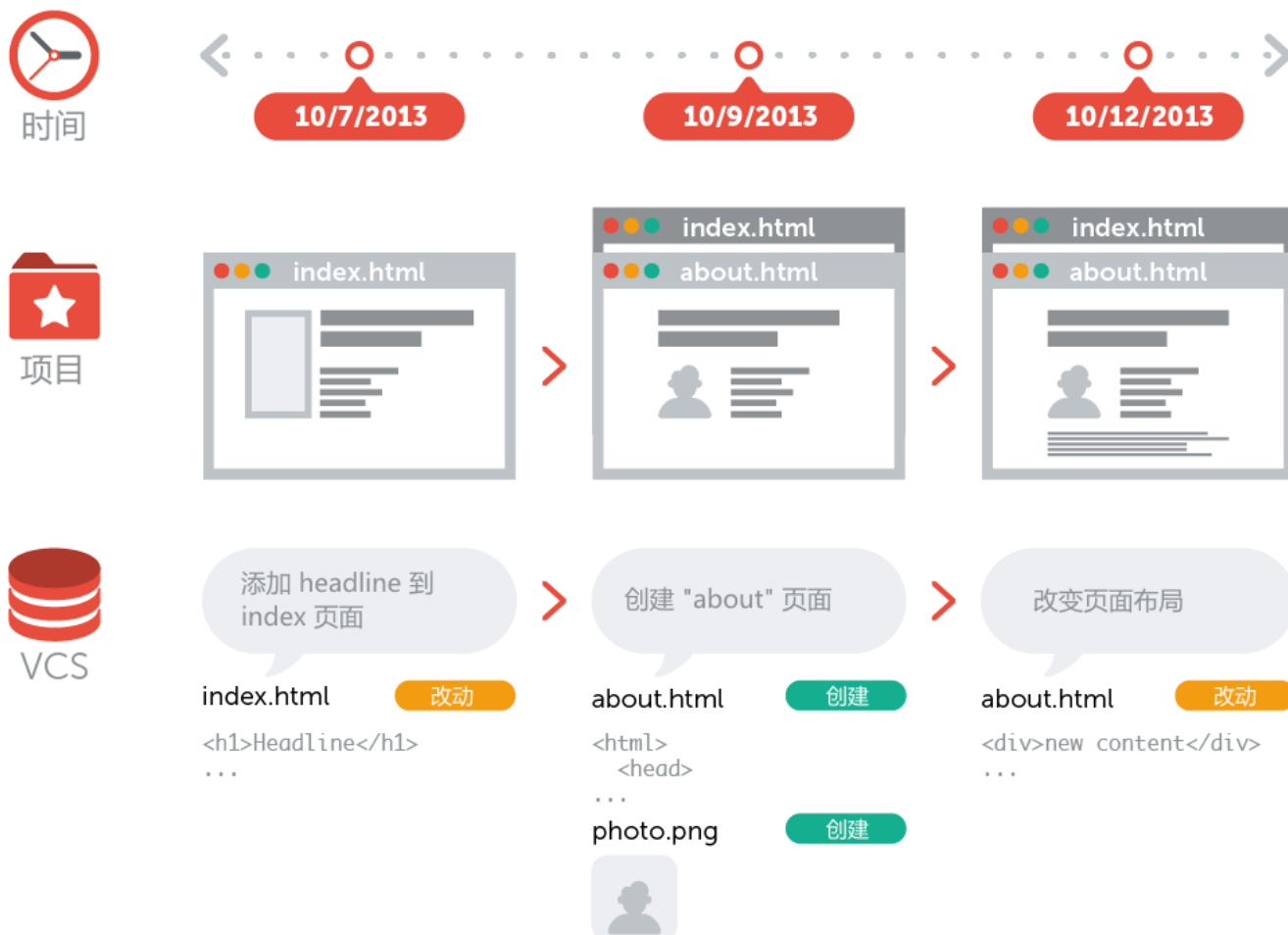
人为维护文档版本的问题

1. 文档数量多且命名不清晰导致文档版本混乱
2. 每次编辑文档需要复制，不方便
3. 多人同时编辑同一个文档，容易产生覆盖



Git 是什么 (★★★)

Git是一个分布式版本管理控制系统（缩写VCS），它可以在任何时间点，将文档的状态作为更新记录保存起来，也可以在任何时间点，将更新记录恢复回来。




小结

- git 是一个版本管理工具
- git 可以帮我们托管代码，每一次提交信息都会被记录在git仓库
- git 可以帮我们管理代码，后续我们都是团队开发，每一个开发一个功能，通过git可以帮我们吧代码进行整合

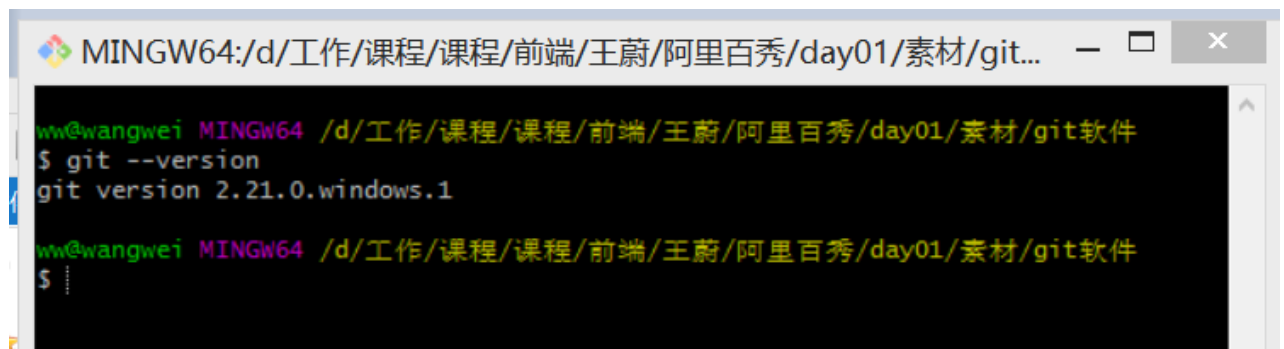
Git 环境搭建 (★★★)

Git 下载安装

- 下载 git 安装包，在 阿里百秀-day01 课件里面的素材文件夹中已经下载好

课件对应目录			阿里百秀 > day01 > 素材 > git软件		
名称			修改日期	类型	
 Git-2.21.0-64-bit.exe			2019/5/13 18:14	应用	
 git-2.21.0-intel-universal-mavericks.dmg			2019/5/13 18:13	好压	

- 双击安装，选择默认配置即可
- 安装完后，右键会出现 git 菜单，选中 Git Bash Here，弹出命令窗口
- 输入命令 `git --version` 查看 git 版本



Git 基本工作流程

git 仓库	暂存区	工作目录
用于存放提交记录	临时存放被修改文件	被Git管理的项目目录(写代码的文件夹)



小结:

- 工作目录：就是我们编写代码的地方

- 暂存区：修改过的文件放在暂存区，方便进行管理
- GIT仓库：代码最终提交存放的位置
- 执行流程： 工作目录中修改的文件 --> 提交到 暂存区 --> GIT仓库

Git 使用前配置

Git 允许多人进行开发，所以我们需要配置用户名和邮箱，这样在 Git 仓库里面就能标明代码是谁进行的提交，方便进行管理

1. 配置提交人姓名： `git config --global user.name 提交人姓名`
2. 配置提交人邮箱： `git config --global user.email 提交人邮箱`
3. 查看git配置信息： `git config --list`

通过命令行的形式进行的配置，也可以通过修改配置文件来进行修改，文件名字叫 `.gitconfig`，位置在 `C:\Users\系统用户名.gitconfig`

注意

1. 如果要对配置信息进行修改，重复上述命令即可。
2. 配置只需要执行一次。

Git 使用 (★★★)

提交文件到 Git 仓库

需要使用到的命令：

1.对"工作目录"的操作

(1)git init` 初始化git仓库

(2)git status` 查看文件状态

(3)git add 文件列表` 追踪文件

2.对"暂存区"的操作

(1)git commit -m 提交信息` 向仓库中提交代码

3.对"git仓库"的操作

(1)git log` 查看提交记录

1.初始化Git 仓库

```
git init
```

此时在对应文件夹中会自动创建一个 `.git` 的隐藏文件夹，里面放的就是git相关配置信息与git仓库

2.查看相关状态

```
git status
```

```
$ git status
On branch master 分支相关信息

No commits yet 当前仓库没有提交记录

Untracked files: 没有被管理起来的文件列表
  (use "git add <file>..." to include in what will be committed)

    index.html

nothing added to commit but untracked files present (use "git add" to track)
ww@wangwei MINGW64 /d/工作/JAVA/workspace/jsSpace/08-阿里百秀/day01/test (master)
```

此时我们index.html 文件还没有被git管理，如果需要被管理起来，需要操作下一个步骤

3.添加文件到git中

```
git add index.html
添加完后，可以查看一下文件状态
git status
```

```
ww@wangwei MINGW64 /d/工作/JAVA/workspace/jsSpace/08-阿里百秀/day01/test (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   index.html 此时变成了绿色，说明文件成功被添加到暂存区

ww@wangwei MINGW64 /d/工作/JAVA/workspace/jsSpace/08-阿里百秀/day01/test (master)
```

此时文件是添加到了暂存区，还没有提交到Git仓库

4.提交到Git仓库

```
git commit -m 第一次提交
```

```
$ git commit -m 第一次提交
[master (root-commit) e4947d3] 第一次提交
1 file changed, 15 insertions(+)
create mode 100644 index.html
```

出现这些信息，代表提交成功；注意，后面要带上提交的日志信息

5.查看提交日志信息

```
git log
```

```
$ git log
commit e4947d356cd3c86389dcc5f308b10376c14e992e (HEAD -> master)
Author: itcast <ittheima@itcast.cn> 提交人姓名
Date: Tue May 14 13:42:48 2019 +0800 提交时间

第一次提交
```

撤销

用暂存区中的文件覆盖工作目录中的文件

场景：当工作目录中添加的一些代码存在一些问题，但是又不记得修改了哪一些地方了，此时就可以用暂存区的文件来覆盖我们工作目录的文件

```
git checkout 文件
```

将文件从暂存区中删除

场景：当我们不小心把其他的一些测试代码或者是一些没用的文件添加到了暂存区后，我们可以通过命令进行移除

```
git rm --cached 文件名
```

恢复git仓库中指定版本的项目

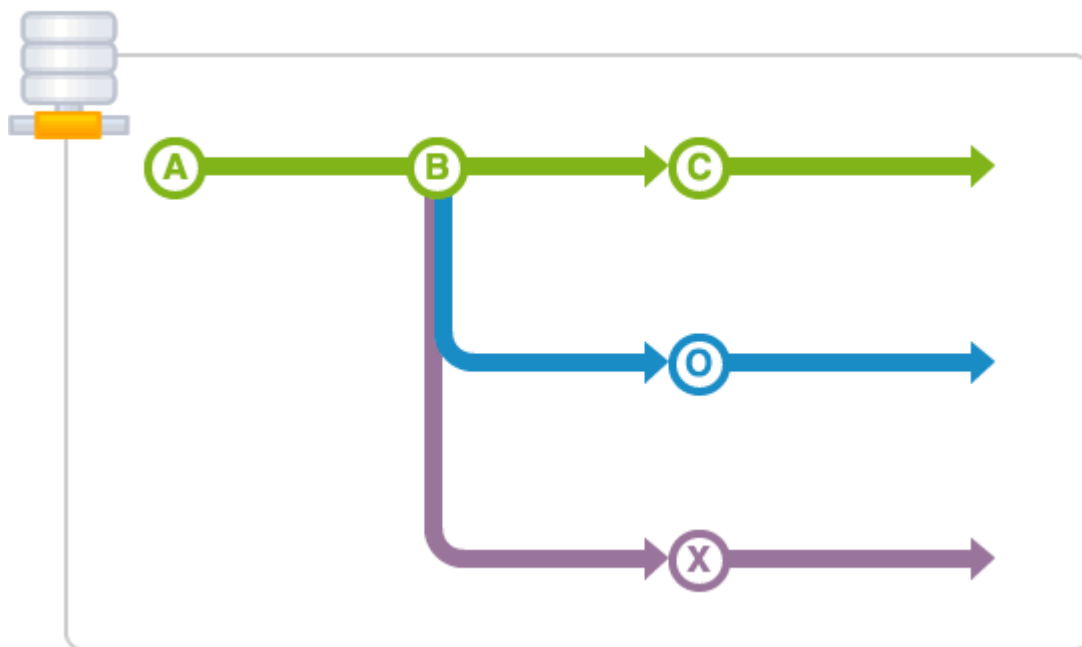
场景：一般是用于恢复到某一个正常的版本

```
git reset --hard 提交ID
```

Git分支 (★★★)

为了便于理解，大家暂时可以认为分支就是当前工作目录中代码的一份副本。

使用分支，可以让我们从开发主线上分离出来，以免影响开发主线。



分支细分

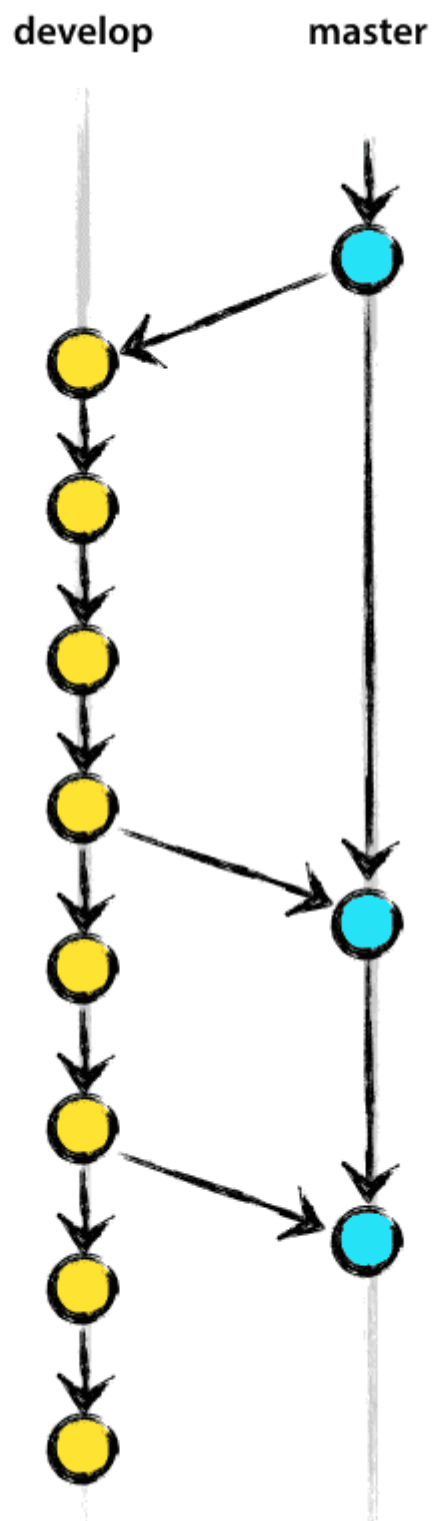
主分支 (master)

第一次向 git 仓库中提交更新记录时自动产生的一个分支。这个属于是主分支，主分支中的代码是很稳定的代码，所以我们在开发的时候一般都不会把代码直接提交到主分支中，主分支中的代码一般都是用于线上的项目；我们一般在开发的时候，会把代码先同步到开发分支上，等功能完成并且测试没有问题了，我们才会同步到主分支上



开发分支 (develop)

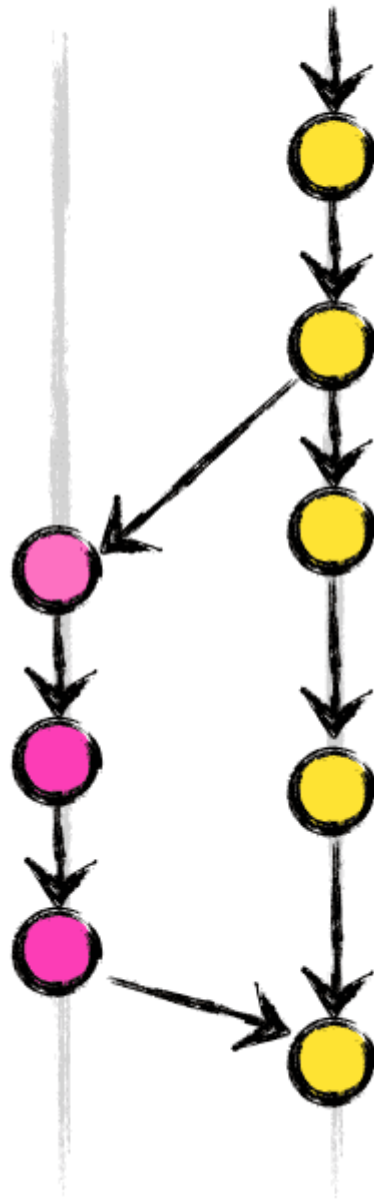
作为开发的分支，基于 master 分支创建，这条分支就是用于我们日常开发的分支



功能分支 (feature)

作为开发具体功能的分支，基于开发分支创建，为了不影响整体的开发项目，我们在实现某一些有难度的功能时候，会创建功能分支，等到功能实现完毕，没有问题了我们才会同步到开发分支上

feature
branches develop



小结

- 主分支是Git自动帮我们创建的 master，这条分支上的代码一般用于都是线上的项目
- 开发分支，在主分支上进行创建，一般用于我们开发的时候使用，也需要尽量保证其稳定性
- 功能分支，在开发分支上进行创建，用于实现某一个难度较大功能的时候使用
- 三者的合并流程为：功能分支 -> 开发分支 -> 主分支

分支相关命令

- `git branch` 查看分支
- `git branch 分支名称` 创建分支
 - 在哪条分支上输入的命令，就是在哪条分支上进行的创建

- `git branch develop` -创建了一个develop的分支
- `git checkout` 分支名称 切换分支
 - `git checkout develop` -切换到develop分支
 - 注意：当切换分支的时候，需要把暂存区里面的文件进行提交，不然会暂存区里面的文件会跟着到切换的这条分支上
 - 当切换会主分支的时候，就看不到其他分支里面的文件了
- `git merge` 来源分支 合并分支
 - 如果当前分支的工作已经完成，就可以合并到到其他分支
 - 需要分清谁要合并谁，例如我们在开发分支上完成了功能，应该合并到主分支上，所以我们要站在主分支角度来进行合并
 - 虽然进行了合并，但是开发分支还是存在
- `git branch -d` 分支名称 删除分支（分支被合并后才允许删除）（-D 强制删除）
 - 分支工作已经完成，就可以进行删除
 - `git branch -d develop`
 - 如果分支没有进行合并，那么默认是不能被删除，这是由于git有分支保护机制
 - 如果想强行删除，把-d 改成-D：`git branch -D develop`

暂时保存更改

在git中，可以暂时提取分支上所有的改动并存储，让开发人员得到一个干净的工作副本，临时转向其他工作。

应用场景：分支的临时切换

- 存储临时改动：`git stash`
- 恢复改动：`git stash pop`

注意：

- 在其他的分支中也能执行恢复改动，但是会把这些文件恢复到当前命令的分支，所以我们在恢复的时候需要注意，我们当时在哪个分支进行的开发

Github

在版本控制系统中，大约90%的操作都是在本地仓库中进行的：暂存，提交，查看状态或者历史记录等等。除此之外，如果仅仅只有你一个人在这个项目里工作，你永远没有机会需要设置一个远程仓库。

只有当你需要和你的开发团队共享数据时，设置一个远程仓库才有意义。你可以把它想象成一个“文件管理服务器”，利用这个服务器可以与开发团队的其他成员进行数据交换。


Github概念


GitHub是一个面向[开源](#)及私有[软件](#)项目的托管平台，因为只支持git 作为唯一的版本库格式进行托管，故名GitHub。

GitHub于2008年4月10日正式上线，除了Git代码仓库托管及基本的 Web管理界面以外，还提供了订阅、讨论组、文本渲染、在线文件编辑器、协作图谱（报表）、代码片段分享（Gist）等功能。目前，其注册用户已经超过350万，托管版本数量也是非常之多，其中不乏知名开源项目 [Ruby on Rails](#)、[jQuery](#)、[python](#) 等。

注册 (★★)

访问[github](#)首页，点击 Sign up 连接。（注册）

 Why GitHub? ▾ Enterprise Explore ▾ Marketplace Pricing ▾

Search GitHub 

Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside 31 million developers.

Username

Pick a username

Email

you@example.com

Password


Create a password


Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

填写用户名、邮箱地址、GitHub登陆密码


 Why GitHub? ▾ Enterprise Explore ▾ Marketplace Pricing ▾


Search GitHub 


Sign in Sign up

Join GitHub

The best way to design, build, and ship software.


 Step 1:
Set up your account

 Step 2:
Choose your plan

 Step 3:
Tailor your experience

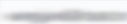

Create your personal account

Username *

itcast-researcher 

This will be your username. You can add the name of your organization later.

Email address *

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password *

.....

Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)


By clicking "Create an account" below, you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.


Create an account


You'll love GitHub

Unlimited public repositories

Unlimited private repositories

 Limitless collaboration

 Frictionless development

 Open source community

选择计划

Welcome to GitHub

You're a few steps away from building better software, @itcast-researcher.

✓

Completed
Set up your account

🔧

Step 2:
Choose your plan

⚙️

Step 3:
Personalize your experience

Choose your plan

With tools developers love and the world's largest open source community, there's no wrong choice.

✓

Free

The basics of GitHub for every developer

\$0

per month

Includes:

∞

Unlimited public and private repositories

✓

3 collaborators for private repositories

✓

Issues and bug tracking

✓

Project management

Are you a [student](#)? Get access to the best developer tools for free with the [GitHub Student Developer Pack](#).

Pro

Pro tools for developers with advanced requirements

\$7

per month

[\(view in HKD\)](#)

Includes:

∞

Unlimited public and private repositories

∞

Unlimited collaborators

✓

Issues and bug tracking

✓

Project management

✓

[Advanced tools and insights](#)

☐ **Help me set up an organization next**
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees. [Learn more about organizations](#)

☐ **Send me updates on GitHub news, offers, and events**
Unsubscribe anytime in your email preferences. [Learn more](#)

Continue

填写 GitHub 问题

/

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

Welcome to GitHub

You'll find endless opportunities to learn, code, and create, @itcast-researcher.

✓

Completed

Set up a personal account

📋

Step 2:

Choose your plan

⚙️

Step 3:

Tailor your experience

How would you describe your level of programming experience?

☐ Totally new to programming

☐ Somewhat experienced

☐ Very experienced

What do you plan to use GitHub for? (check all that apply)

☐ Project Management

☐ Development

☐ Research

☐ Design

☐ School projects

☐ Other (please specify)

Which is closest to how you would describe yourself?

☐ I'm a student

☐ I'm a professional

☐ I'm a hobbyist

☐ Other (please specify)


What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

Submit

skip this step

© 2019 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)




[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

验证邮箱

/

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

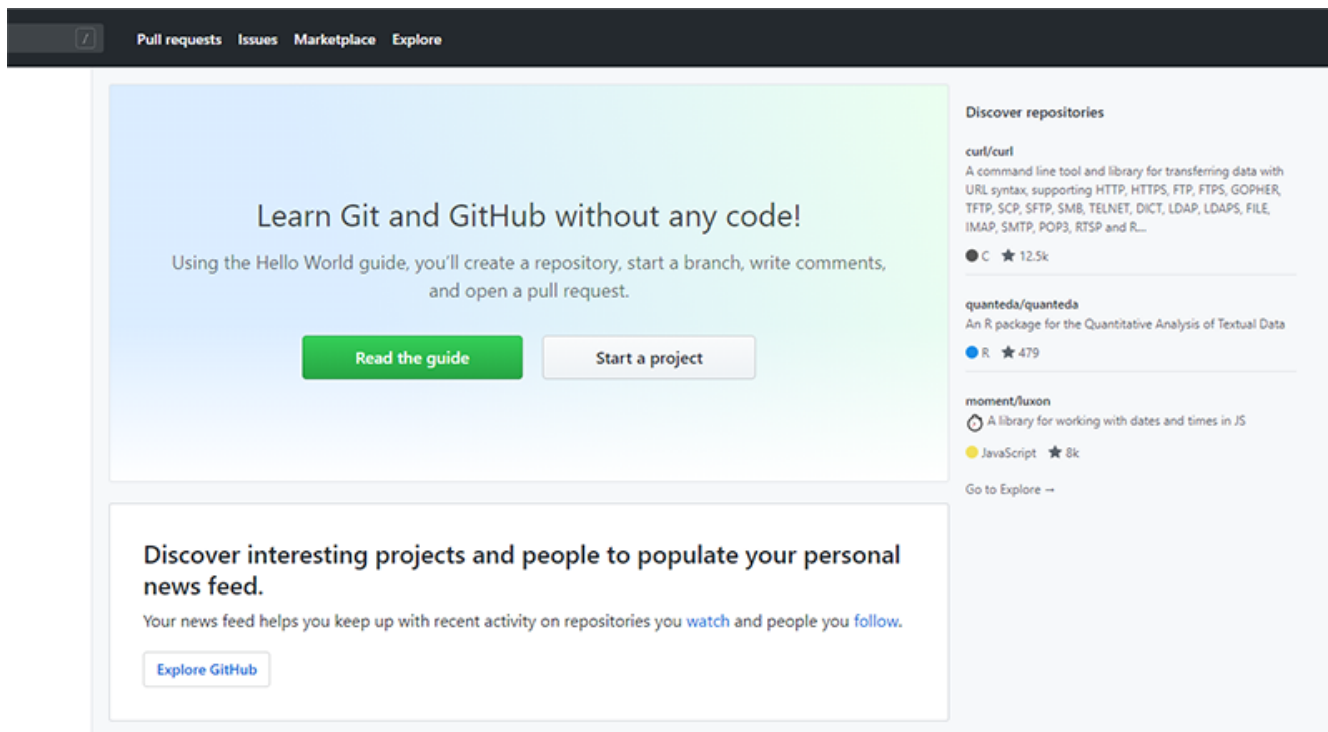


Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.
An email containing verification instructions was sent to **wangjian@itcast.cn**.

Didn't get the email? [Resend verification email](#) or [change your email settings](#).

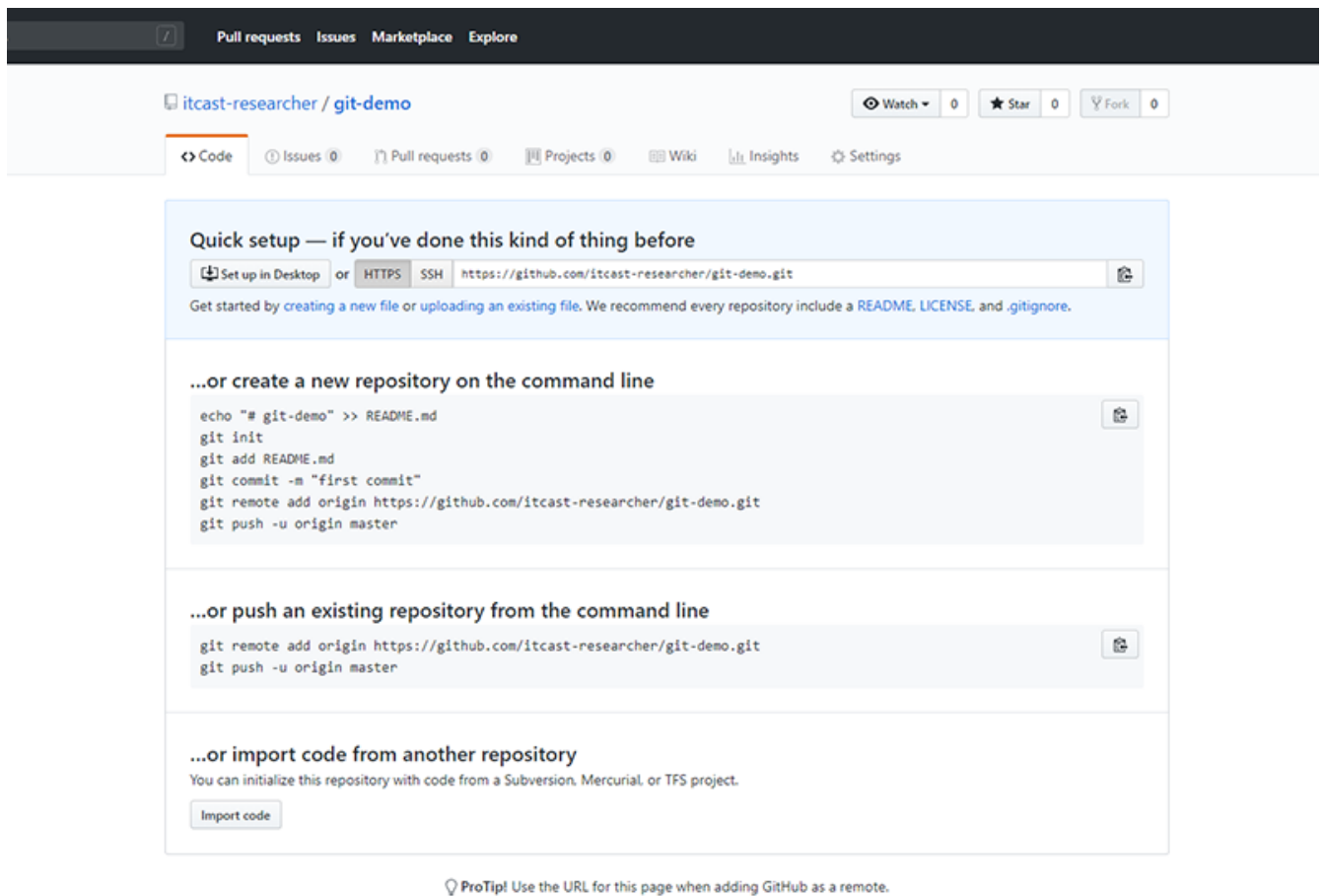
GitHub 个人中心



创建仓库 (★★)

填写仓库基本信息

将本地仓库推送到远程仓库

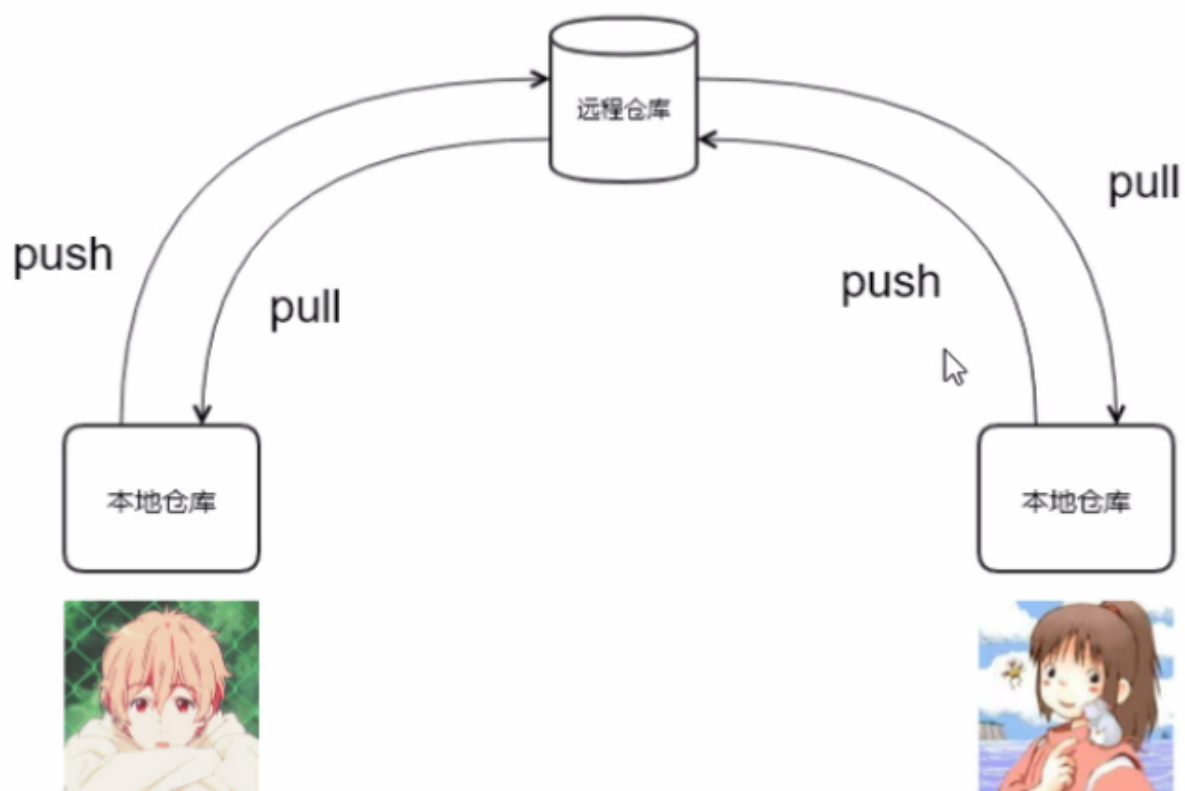
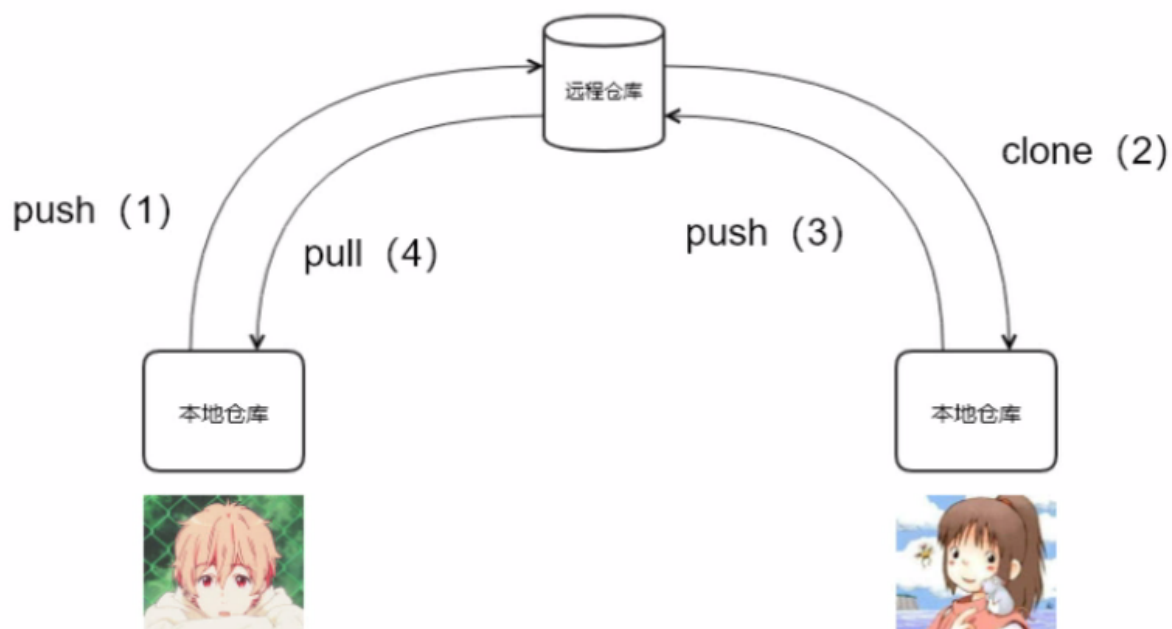


1. git remote add 远程仓库地址别名 远程仓库地址
2. git push -u 远程仓库地址或别名 本地分支名称:
-u 记住推送地址及分支，下次推送只需要输入git push即可

多人协作开发 (★★)

在真实的工作中，都是团队在进行开发项目，一个程序员来负责一个功能模块，最后整合在一起就是完整项目，所以这里就需要多人协作开发了，需要一个服务器来管理所有程序员的代码，每一个程序员把代码都提交到服务器里面，这样，服务器里面的代码就是一个完成的项目了

- A在自己的计算机中创建本地仓库
- A在github中创建远程仓库
- A将本地仓库推送到远程仓库
- B克隆远程仓库到本地进行开发
- B将本地仓库中开发的内容推送到远程仓库
- A将远程仓库中的最新内容拉取到本地



A在自己的计算机中创建本地仓库

```
git init    创建本地仓库
git add index.html  添加一个文件到暂存区
git commit -m A程序员第一次提交
```

A在github中创建远程仓库

参考上面 创建仓库

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH

复制上面的地址

A将本地仓库推送到远程仓库

```
git push https://github.com/自己对应的仓库链接 分支名称
```

给远程仓库配置别名

```
git remote add origin https://github.com/自己对应的仓库链接
这句命令就是给我们远程仓库的地址配置了别名 叫做 origin
后续我们去推送的时候就可以直接使用别名
git push origin master
如果我们提交的时候携带了参数 -u 那么下次提交的时候就只需要输入 git push
git push -u origin master
```

当我们第一次登录了github，那么window会默认帮我们保存用户名跟密码，保存的位置如下



B克隆远程仓库到本地进行开发

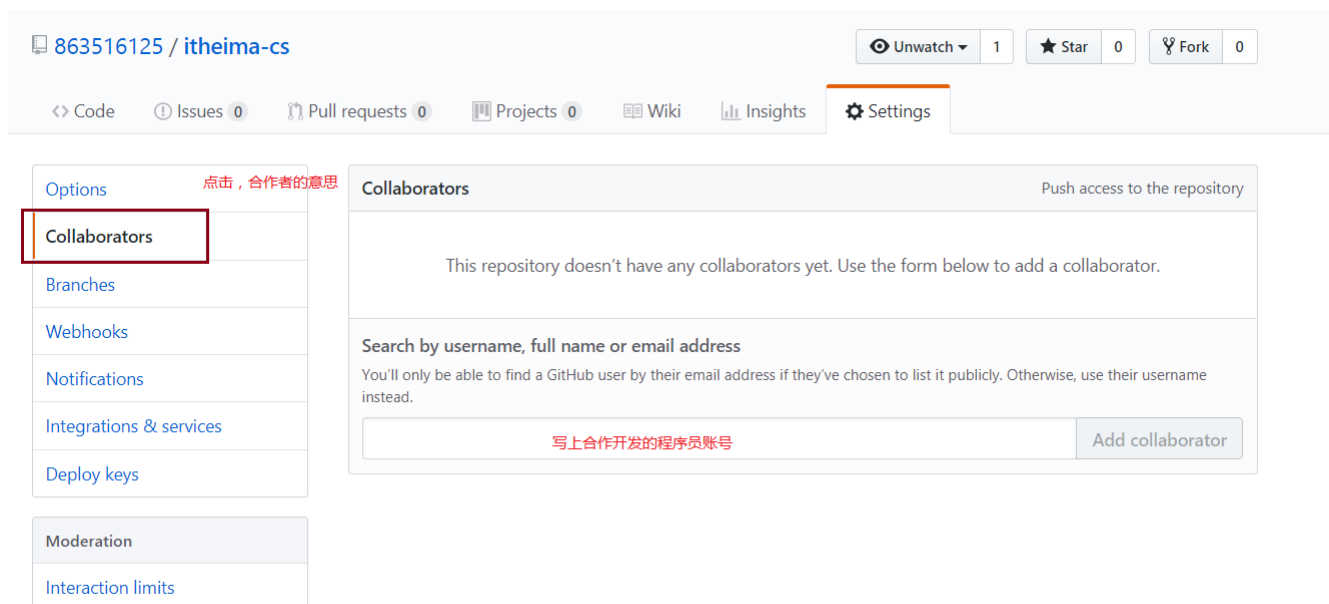
因为A已经创建了远程的仓库，所以程序员B就不需要创建仓库，直接把远程的仓库克隆一份到本地即可

```
git clone 远程仓库地址
```

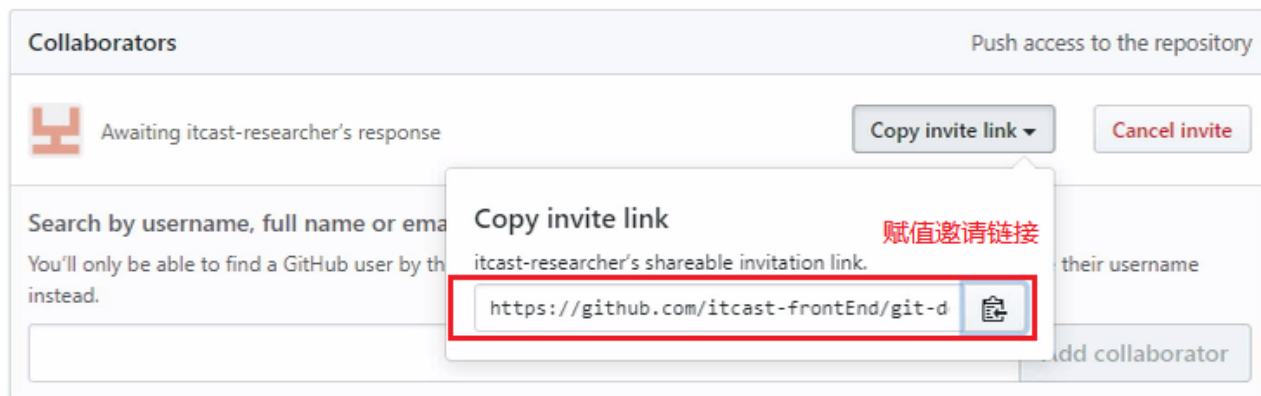
B将本地仓库中开发的内容推送到远程仓库

B程序员此时修改index.html里面的内容
git add index.html 因为修改了内容，需要提交到暂存区
git commit -m 程序员B修改了代码
此时推送到服务器的时候，需要程序员A在github上面添加程序员B

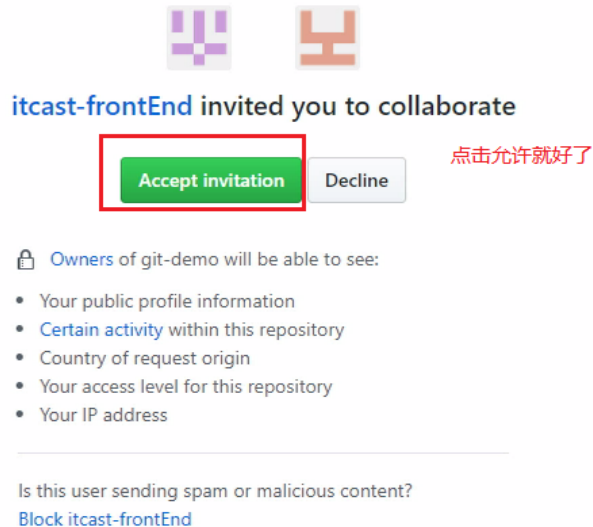
程序员A进入github官网，点击 settings，点击Collaborators



点击 add collaborator 就会被添加进去



把邀请链接发送给程序员B，程序员B需要登录github，然后对链接进行访问



需要在凭据里面把程序员A的账号进行删除

可以在命令行里面进行push了

之前克隆了远程仓库，就把别名配置一起进行了克隆

`git push origin master`

A将远程仓库中的最新内容拉取到本地

拉取利用pull命令，拉取是读操作，不需要校验身份

```
git pull origin master
```

冲突的解决 (★★★)

多人开发肯定会出现代码冲突的情况，冲突情况的产生，是多个人同时修改了一个文件，例如，A修改了index.html文件，B也修改了index.html文件；A进行了提交，那么B就不能提交了

```
$ git push origin master
To https://github.com/itcast-frontEnd/git-demo.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/itcast-frontEnd/git-demo.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.
mongoose@DESKTOP-RO96G8U MINGW64 /c/course/AliBaixiu/day01/code/B/git-demo (master)
```

程序员B需要先把服务器的最新代码拉取到本地，当程序员B输入了 pull 命令后，命令行里面会进行提示

```

mongoose@DESKTOP-RO96G8U MINGW64 /c/course/AliBaixiu/day01/code/B/git-demo (master)
$ git pull origin master
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/itcast-frontEnd/git-demo
* branch          master      -> FETCH_HEAD
  b8edb7b..cf5ece8  master      -> origin/master
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.

mongoose@DESKTOP-RO96G8U MINGW64 /c/course/AliBaixiu/day01/code/B/git-demo (master|MER
GING)
$ |

```

此时文件里面内容就会发生变化

```

我是index.html文件
<<<<<< HEAD      冲突开始
111111111111111111 我是程序员B做出的修改
=====          分割线
111111111111111111 我是程序员A做出的修改
>>>>>> cf5ece87f20885167373188c8f8ac35af7d13395
222222222222222222
333333333333333333
我是程序员B添加的内容

```

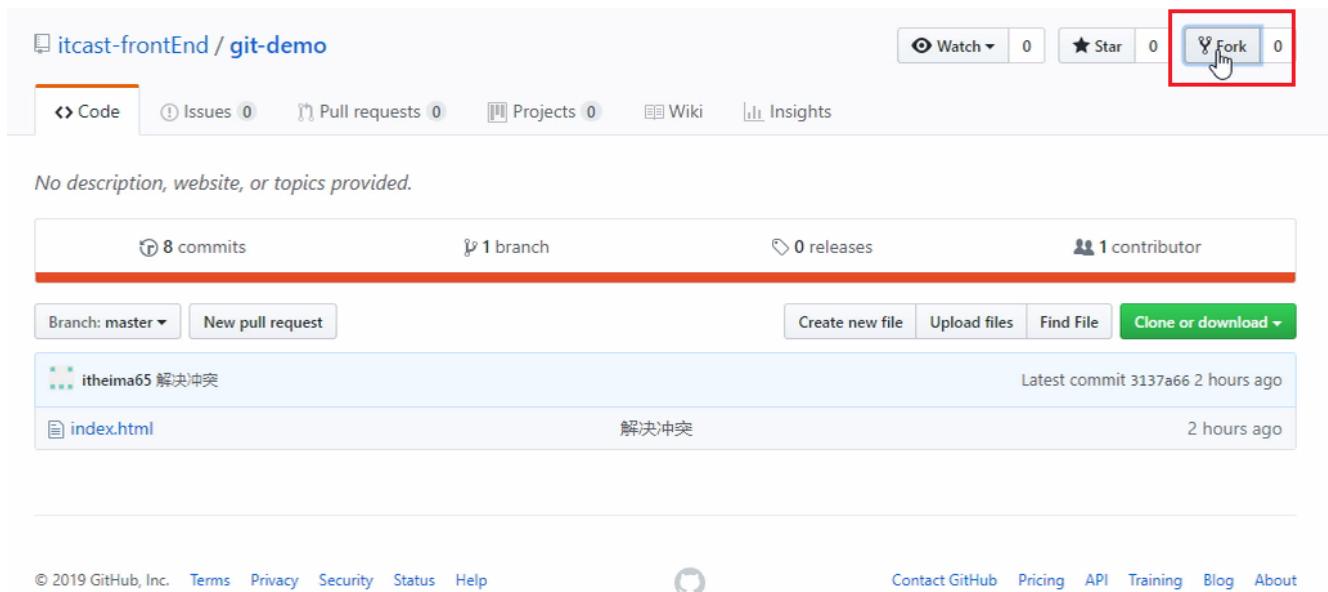
程序员B就需要去多余的代码，然后把里面的内容改成需要的内容就可以了

git commit -m 日志

git push origin master

非团队协作开发 (★★)

一般在使用一些开源框架的时候会遇到，当我们去使用别人写好的一些框架，觉得里面实现的功能不完善，或者你有更优的实现方式，就可以利用github上面 Fork来进行拷贝到自己的账户中



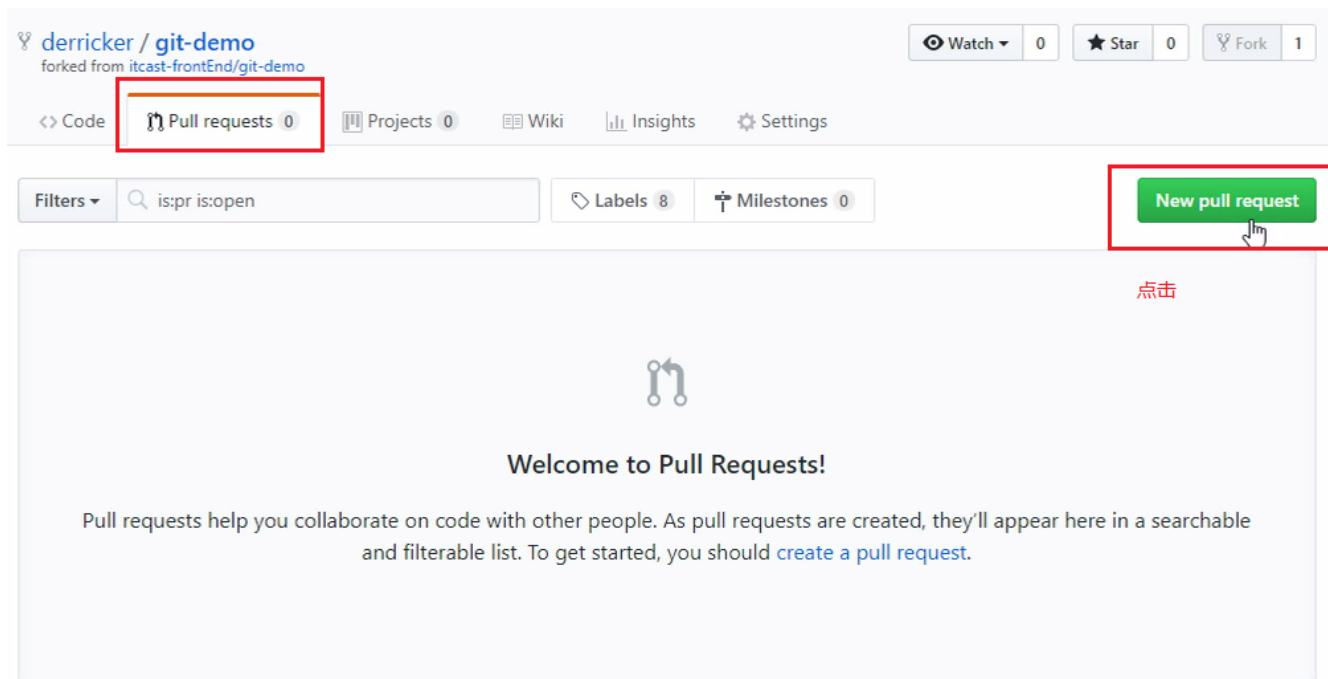
fork完后，我们就可以把服务器里面的代码clone到本地

```
git clone 仓库地址
```

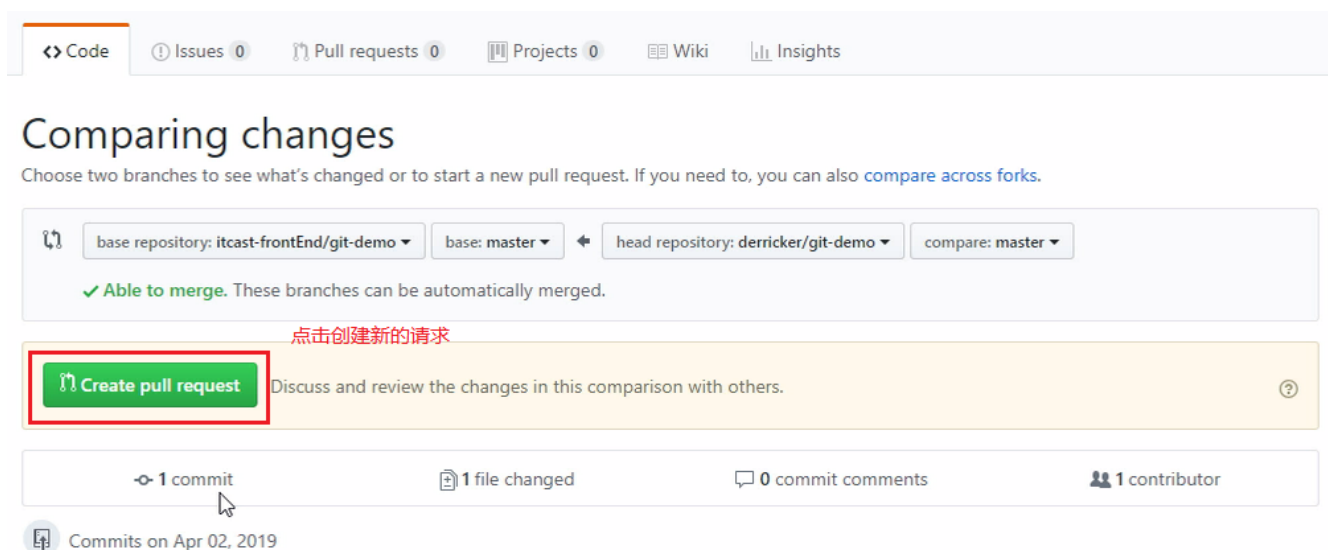
后续就可以进行修改，添加到暂存区，然后提交到git仓库，推送到服务器

```
git add 文件
git commit -m 日志信息
git push 仓库地址
```

此时已经把代码改好，并且推送到了自己的github账户中，此时我们需要把修改好的代码推送给原作者



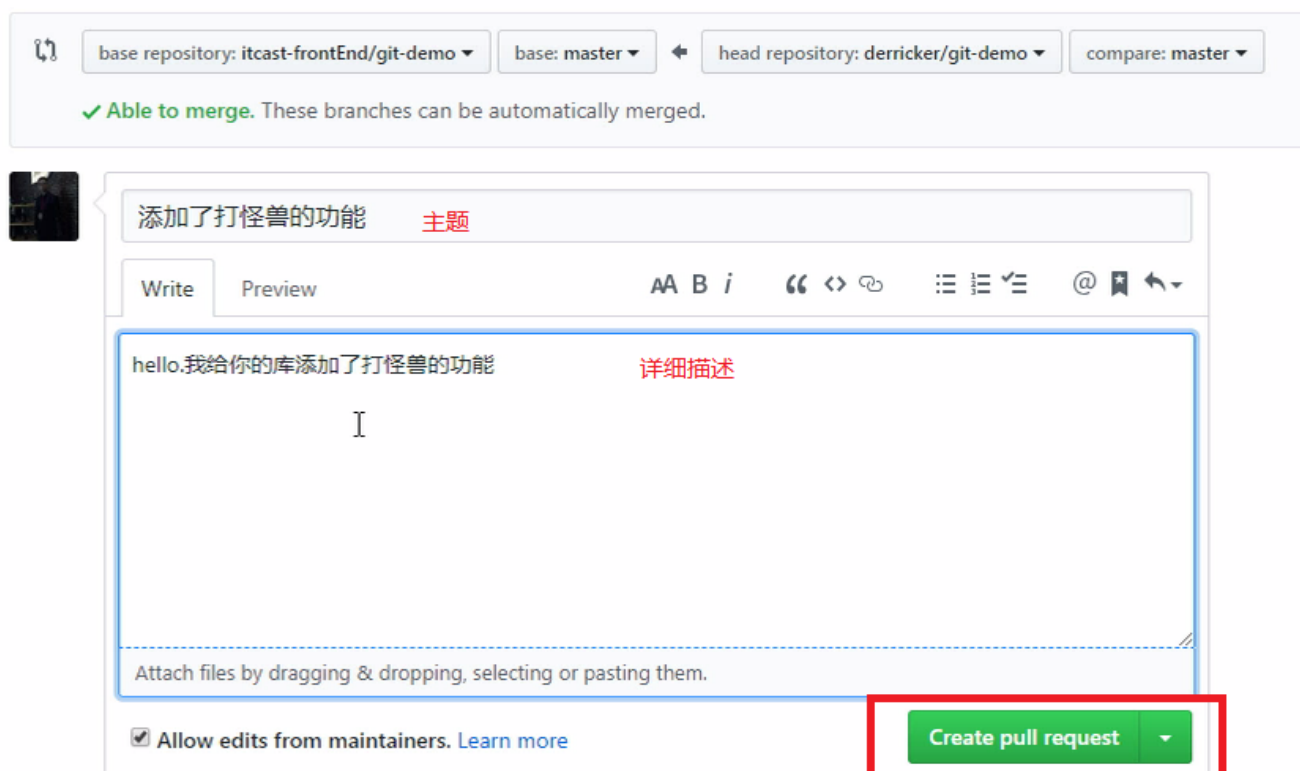
点击 New pull requests 跳转到创建新请求的页面



点击按钮会跳转到发送页面，填写主题和描述，点击create pull request即可，此时原作者就能收到信息

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



原作者如果想看一些代码的变化，可以参照下图

添加了打怪兽的功能 #1

[Edit](#)

Open derricker wants to merge 1 commit into `itcast-frontEnd:master` from `derricker:master`

Conversation 2

Commits 1

Checks 0

Files changed 1

原作者可以通过这个选项来看到代码的变化

Changes from all commits ▾ File filter... ▾ Jump to... ▾ +3 -1

Diff settings ▾

Review changes ▾

4		index.html	
		@@ -3,4 +3,6 @@	
3	3	11111111111111111111 我是程序员A做出的修改	
4	4	22222222222222222222	
5	5	33333333333333333333	
6	-	我是程序员B添加的内容	
6	+	我是程序员B添加的内容	
7	+		
8	+	我是程序员C添加的内容	

原作者觉得修改不错，可以进行合并

Conversation 2

Commits 1

Checks 0

Files changed 1

+3 -1

 derricker commented 3 minutes ago First-time contributor

hello.我给你的库添加了打怪兽的功能

程序员C做出了修改 7ad6de6

Add more commits by pushing to the `master` branch on `derricker/git-demo`. Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style. This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request

通过这个按钮就可以进行合并

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Unsubscribe

You're receiving notifications because you commented.

3 participants

Lock conversation

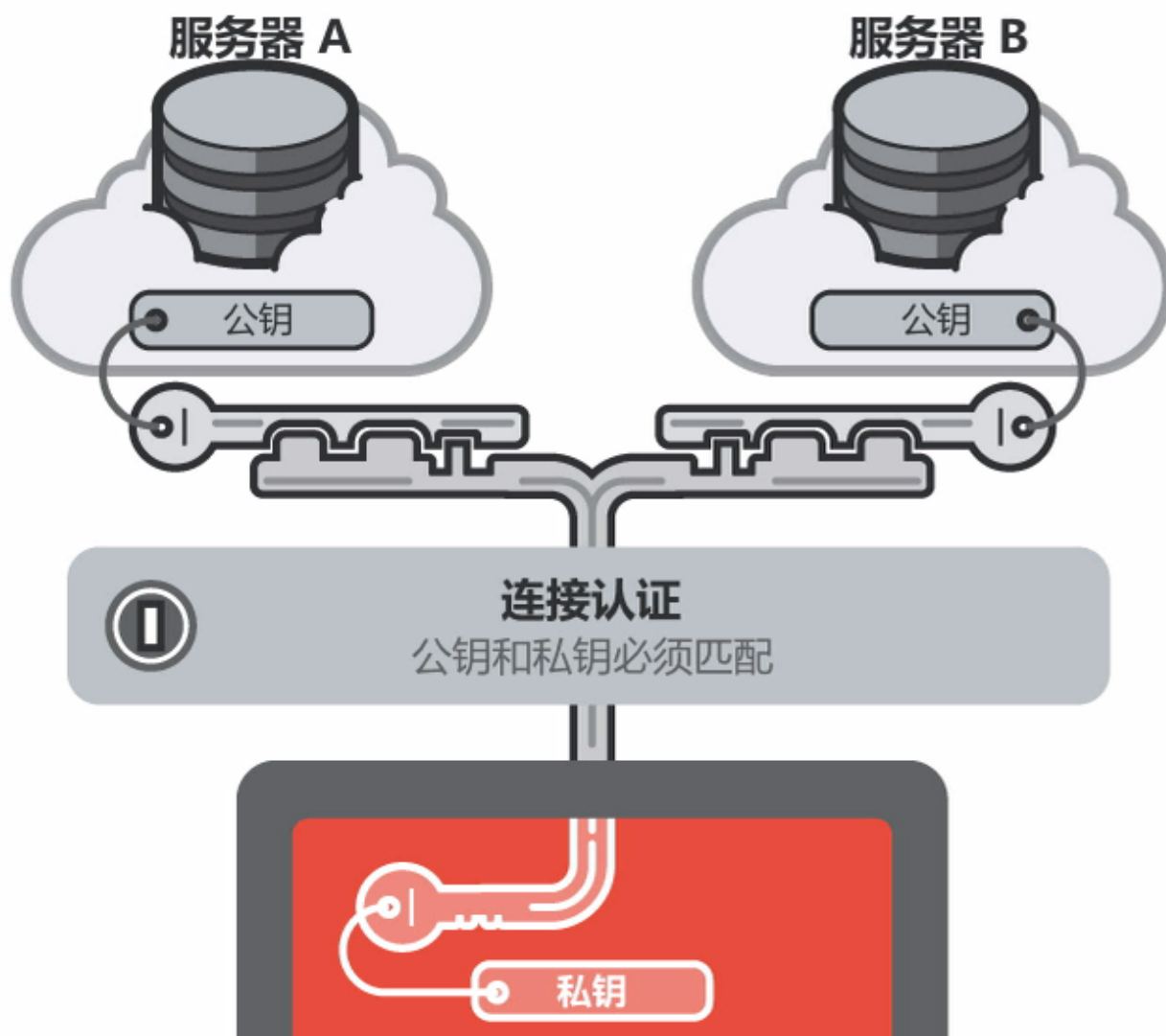
Git 高级用法 (★★★)

SSH免登录

利用SSH协议去进行提交，可以进行免登录操作，实现其原理用的是公钥与私钥，公钥跟私钥是成对呈现，打个比喻：公钥就是一个门锁，私钥就是打开门锁的钥匙，公钥保存在github的账户中（远程服务器中），私钥保存在客户端，当客户端进行push的时候，会携带私钥，远程服务器就会去比对是否匹配，如果匹配，那么push成功

公钥：保存在github账户中，也就是在远程服务器中

私钥：保存在客户端

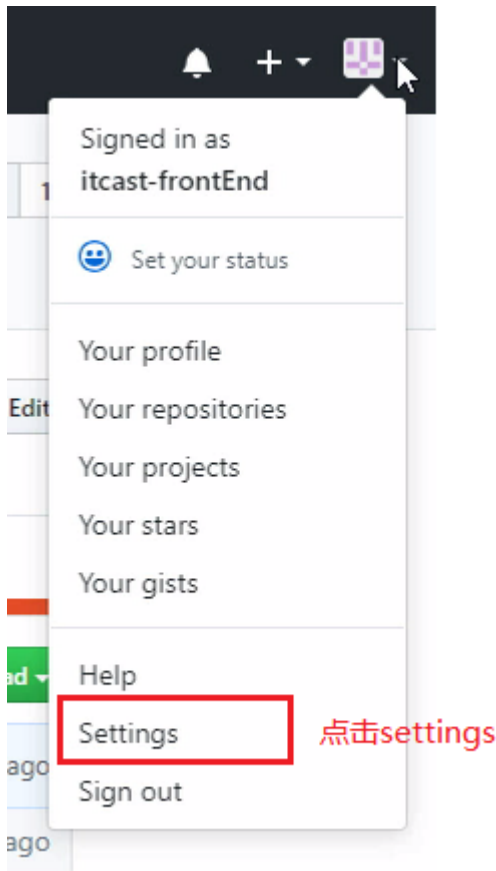


生成密钥

ssh-keygen 生成的密钥在C:\Users\当前用户名称\.ssh 文件夹里面

在github中添加公钥

找到 `id_rsa.pub` 文件，这个就是公钥，复制里面的内容，到github上，点击用户头像，选择settings



Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys

SSH keys

点击SSH key

New SSH key

There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH Problems](#).

GPG keys

New GPG key

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and [add it to your account](#).

SSH keys / Add new

Title

Key

ssh-rsa 把 id_rsa.pub 里面的内容copy到这里
AAAAB3NzaC1yc2EAAAADAQABAAQDPBMGT3oqMEy6rTo6ndRzM986rUKbYQ3ViBhc8Mdg4gSQdx/XKt9i0R9z6
2awVbquBsYOOtrIGzA7xylpbzQknkS2u8tOATHuxdRtmMnziXrs1EcctRUj/YOU3nUj0VTARAQWfhget/cmYCf1Ovf6zPo
LzuhQa8V9JysydBzas/eAjx03UBCqcHCdaVH8FdOlqItXIKNkD/6krSBRL71CNx6tc4QU/CuR6aLe31Sb5P234yaWUFFQ+
OZxhSL1Kmul2XMJjvd8EWT8KYz+Mx6yv0Gt8neqNhA4GrIQ1UqLP/hGVJjwin+zLV1UvLv/TXtQu6lBGoBrPrLUfKc6UEBJ
L ww@wangwei

Add SSH key

点击添加就好

此时就可以通过github上面ssh的协议地址进行push代码了

小结

- SSH免密登录主要用到了公钥跟私钥，公对钥就像门锁，私钥就像门锁对应的钥匙
- 公钥放在 github 远程服务端
- 私钥放在本地客户端
- 利用 ssh-keygen 可以生成一对密钥，这一对密钥生成放在 C:\\Users\\当前用户名称\\.ssh 文件夹里面
- 打开 id_rsa.pub，把里面的内容进行复制，到 github 上面进行添加公钥，把内容复制到里面即可；公钥在 github 上是可以配置多个的，用于多个用户进行免密登录
- 在 github 上把 ssh 协议地址进行拷贝
- 本地利用push命令进行提交即可，git 会默认帮我们携带私钥，在提交的过程中，它会自动去匹配服务器上的公钥和客户端的私钥，如果匹配成功，那么就能提交，如果失败，不能提交

Git 忽略清单

将不需要被 git 管理的文件名字添加到此文件中，在执行 git 命令的时候，git 就会忽略这些文件

git 忽略清单文件名称叫：.gitignore

仓库的详细说明

在仓库根目录添加一个 readme.md 文件即可，在里面写上内容，push到服务器后，默认在 github 里面就能看到仓库的详细说明了

Git 相关命令整理

```
git config --global user.name //配置姓名
git config --global user.email //配置邮箱
git config --list //查看配置信息
git init //初始化本地仓库
git status //查看文件状态
git add 文件名 //添加某个文件到暂存区，如果写 . 代表当前文件夹下所有的文件、
git commit -m 日志说明 //提交到本地仓库
git log //查看提交记录
git checkout 文件名 //撤销，让暂存区文件覆盖工作区间文件
git rm --cached 文件名 //在暂存区移除相应文件
git reset --hard 提交ID //恢复到指定版本
git branch //查看分支
git branch develop //创建分支
git checkout 分支名 //切换分支
git merge //合并分支
git branch -d 分支名称 //删除分支
git clone 地址 //克隆远程仓库
git push 地址 分支名 //往服务器推送
```

```
git pull 地址 //将服务器代码拉取到本地
git remote add 名称 地址 //给地址取别名
git push -u origin master //-u的参数让git记录信息，下次只需要 git push 就能进行提交
ssh-keygen //生成一对密钥
```