软件工程实验

Git

2021

目录

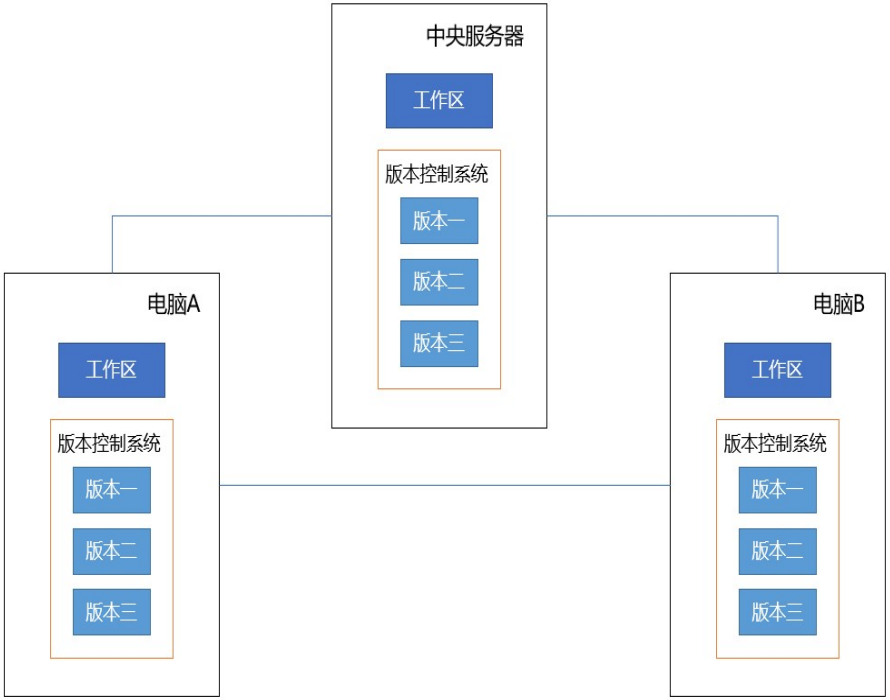
01.

Git简介


git --fast-version-control
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

版本控制系统，是指能随时间的推进记录一系列文件以便于开发者以后想要回退到某个版本的系统，主要分为三类：本地版本控制系统、集中版本控制系统和分布式版本控制系统。

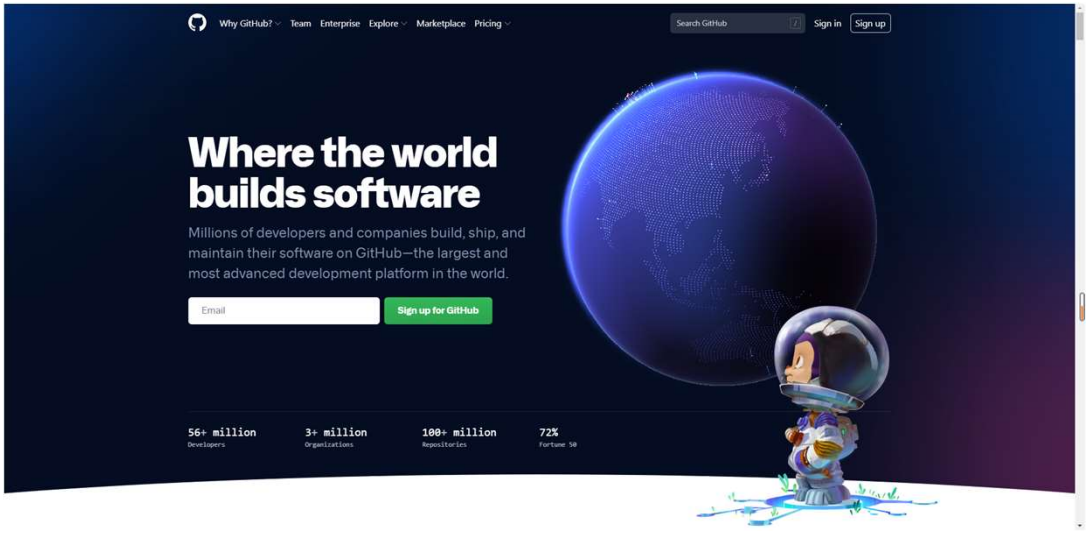
分布式版本控制系统与前两者均不同。首先，在分布式版本控制系统中，系统保存的不是文件变化的差量，而是文件的快照，即把文件的整体复制下来保存，而不关心具体的变化内容。其次，最重要的是分布式版本控制系统是分布式的，开发者从中央服务器拷贝下来代码时，拷贝的是一个完整的版本库，包括历史纪录，提交记录等，这样即使某一台机器宕机也能找到文件的完整备份。



02.

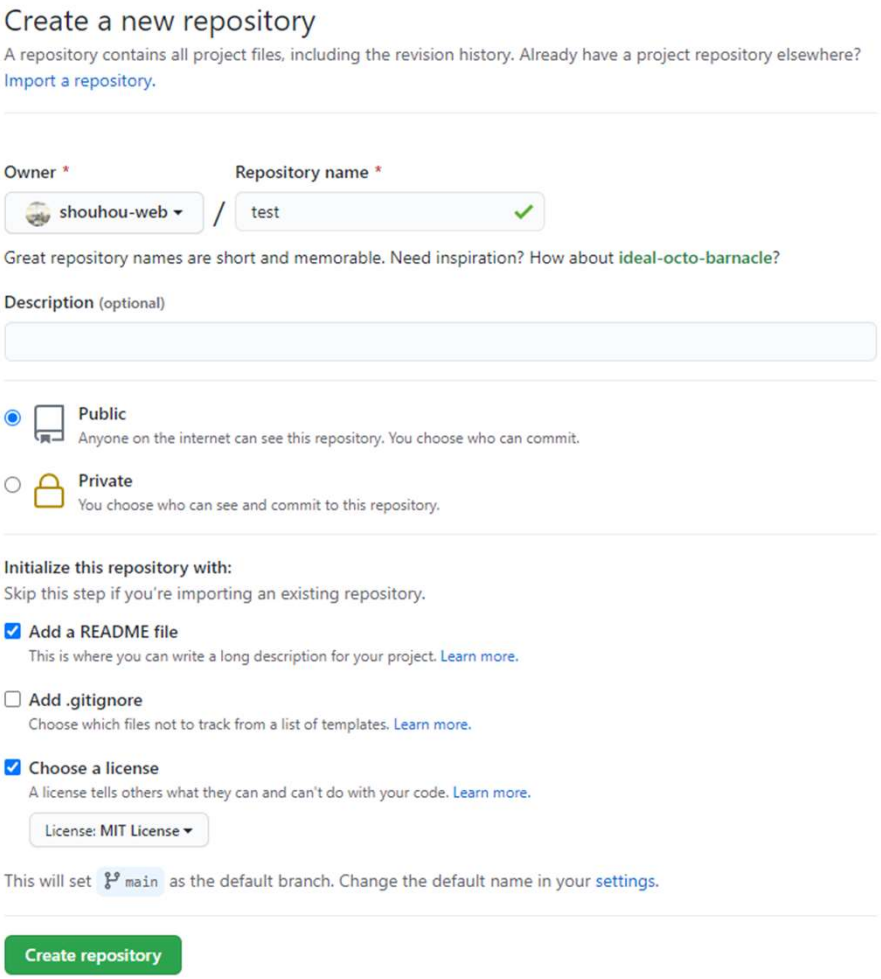
远程仓库的设置

远程仓库的选取

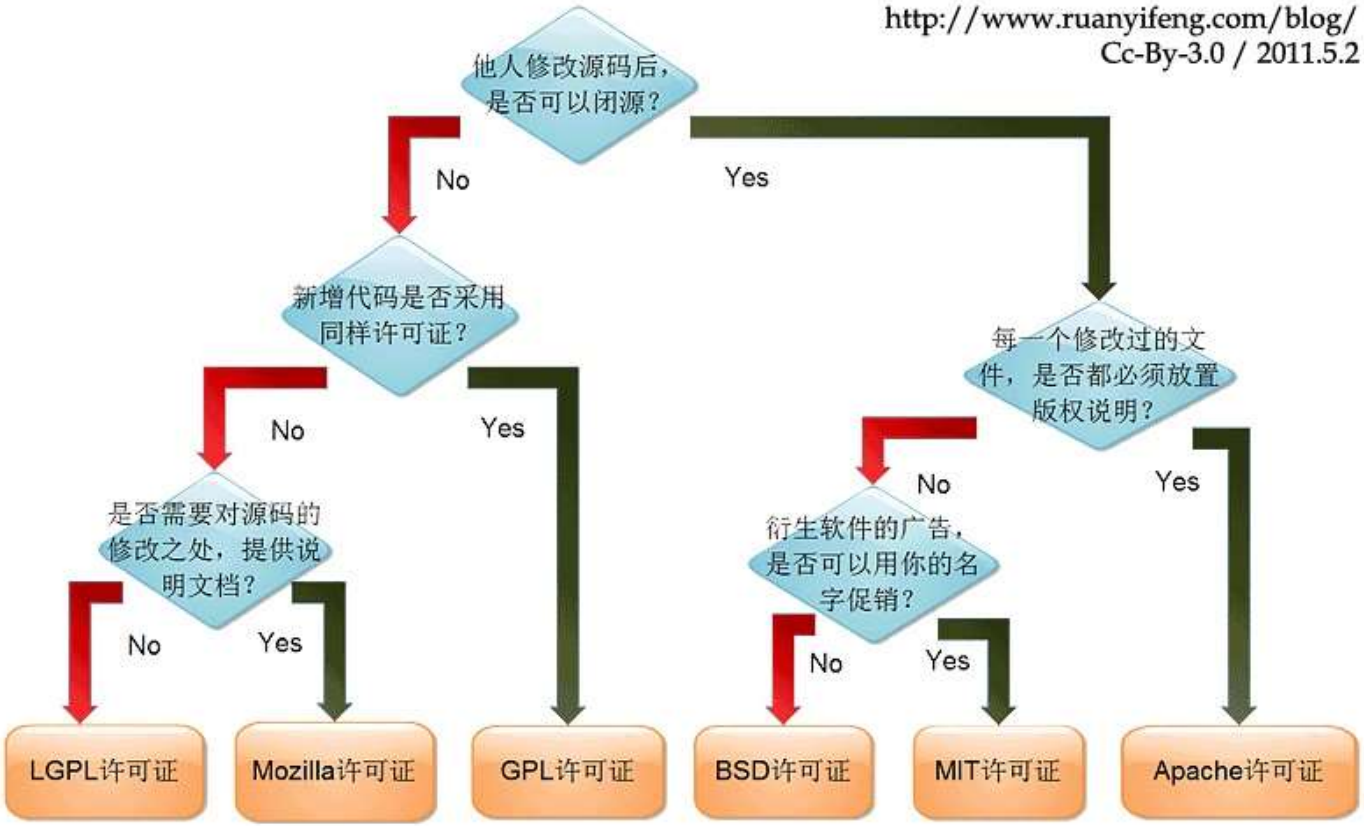


github与gitee均可，以前常常使用github，但是它有可能被墙（），所以这里也介绍gitee，可以自行选择一个使用。（本PPT使用gitee做演示）

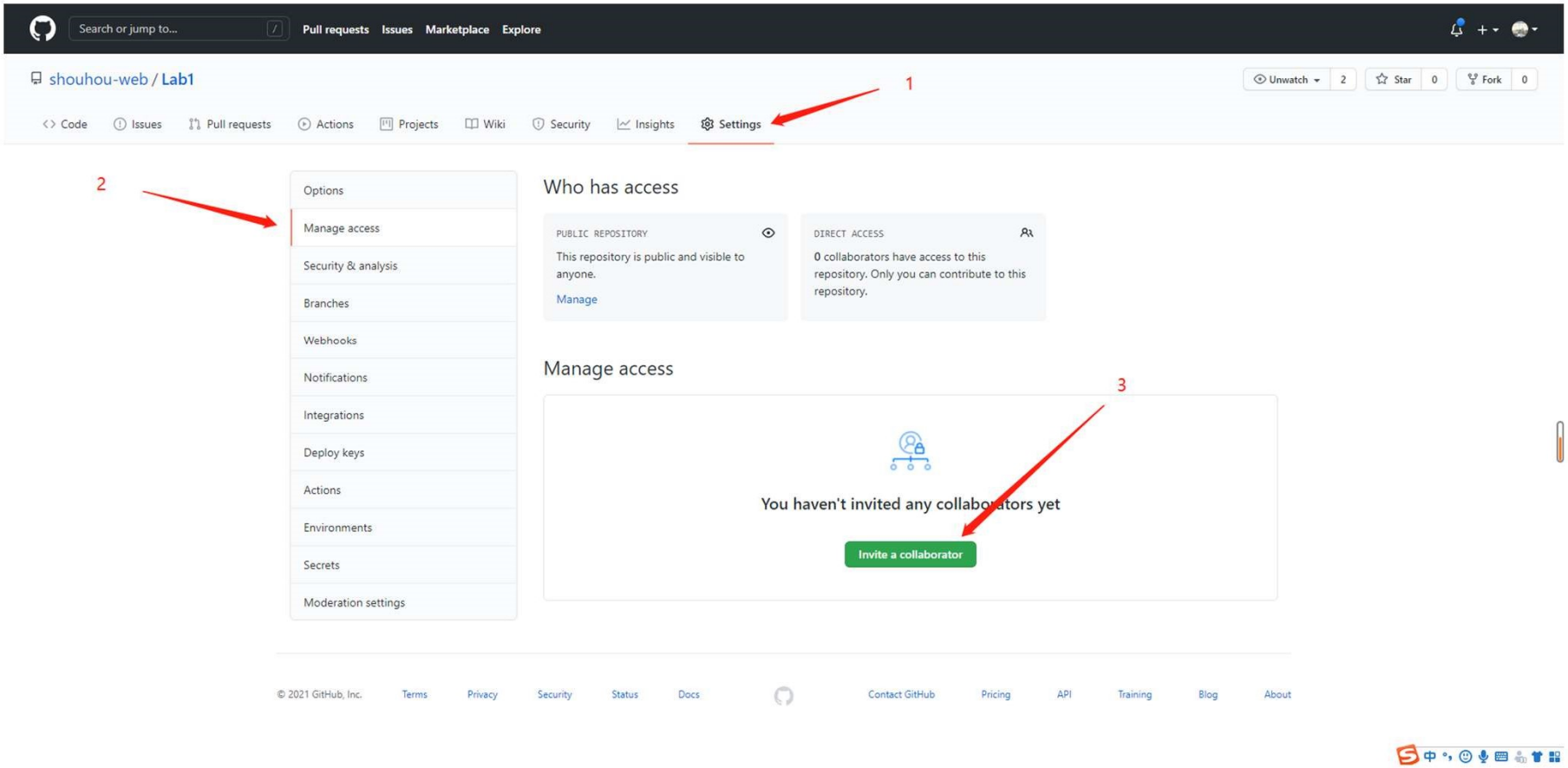
新建一个github/gitee项目



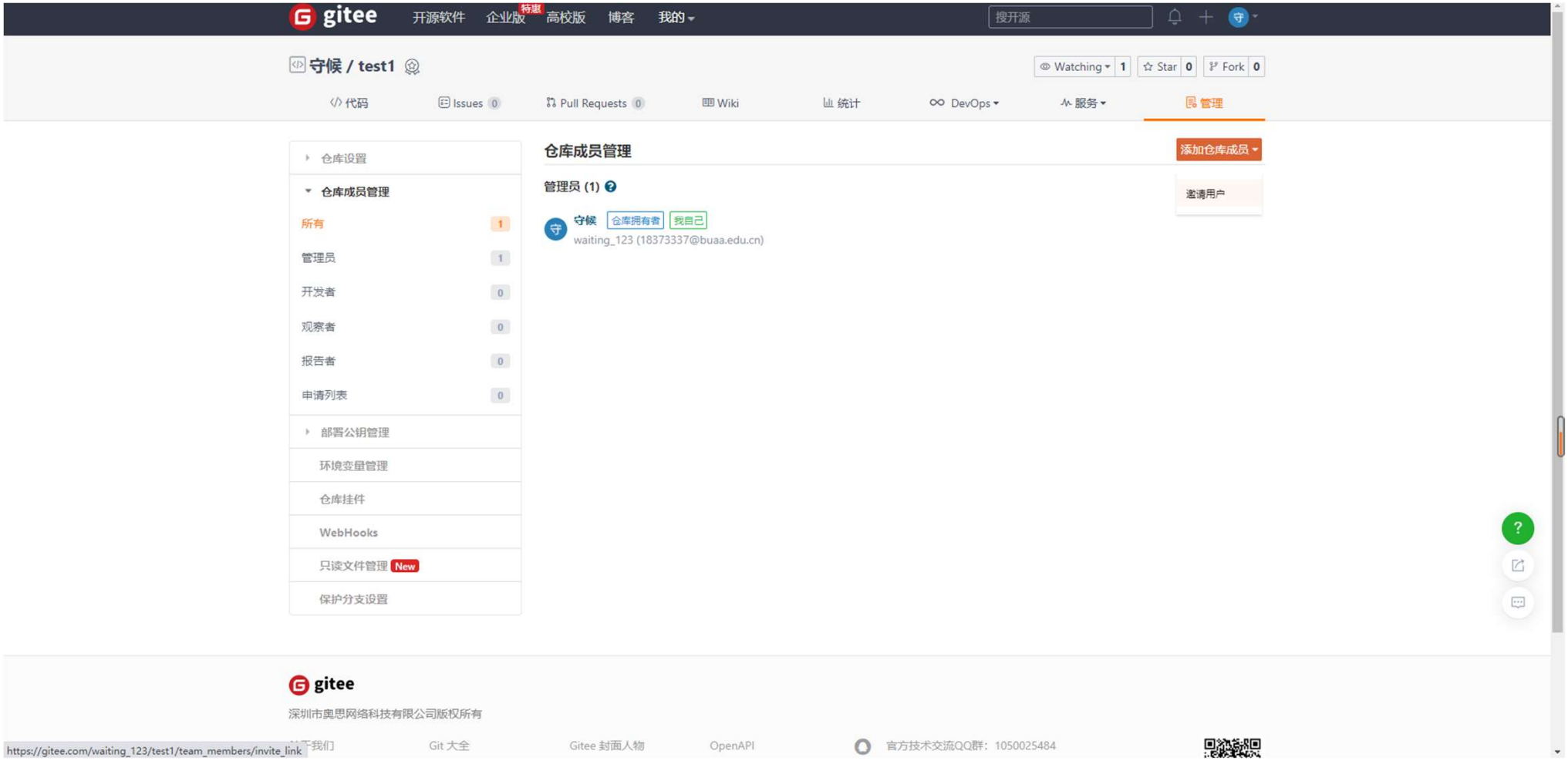
关于MIT协议的选取


http://www.ruanyifeng.com/blog/
Cc-By-3.0 / 2011.5.2
他人修改源码后，是否可以闭源？
No
Yes
新增代码是否采用同样许可证？
No
Yes
是否需要对源码的修改之处，提供说明文档？
No
Yes
每一个修改过的文件，是否都必须放置版权说明？
No
Yes
衍生软件的广告，是否可以用你的名字促销？
No
Yes
LGPL许可证
Mozilla许可证
GPL许可证
BSD许可证
MIT许可证
Apache许可证

邀请成员（github）


Search or jump to...
Pull requests
Issues
Marketplace
Explore
shouhou-web / Lab1
Unwatch 2
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
1
2
Options
Manage access
Security & analysis
Branches
Webhooks
Notifications
Integrations
Deploy keys
Actions
Environments
Secrets
Moderation settings
Who has access
PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage
DIRECT ACCESS
0 collaborators have access to this repository. Only you can contribute to this repository.
Manage access
3
You haven't invited any collaborators yet
Invite a collaborator
© 2021 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

邀请成员（gitee）


gitee
开源软件
企业版
特惠
高校版
博客
我的
搜开源
守候 / test1
Watching 1
Star 0
Fork 0
代码
Issues 0
Pull Requests 0
Wiki
统计
DevOps
服务
管理
仓库设置
仓库成员管理
所有 1
管理员 1
开发者 0
观察者 0
报告者 0
申请列表 0
部署公钥管理
环境变量管理
仓库挂件
WebHooks
只读文件管理 New
保护分支设置
仓库成员管理
添加仓库成员
管理员 (1)
邀请用户
守候
仓库拥有者
我自己
waiting_123 (18373337@buaa.edu.cn)
gitee
深圳市奥思网络科技有限公司版权所有
Git 大全
Gitee 封面人物
OpenAPI
官方技术交流QQ群：1050025484
https://gitee.com/waiting_123/test1/team_members/invite_link

03.

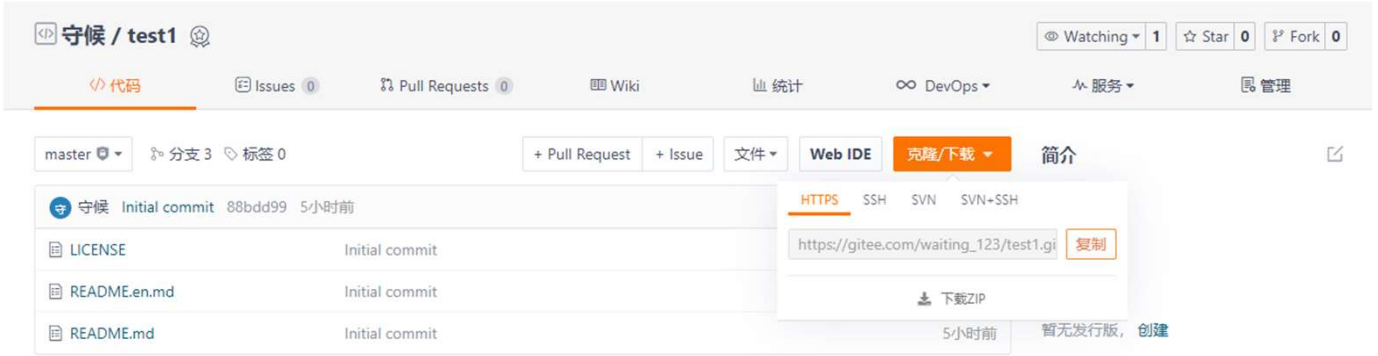
Git 指令

https://www.git-scm.com/download/

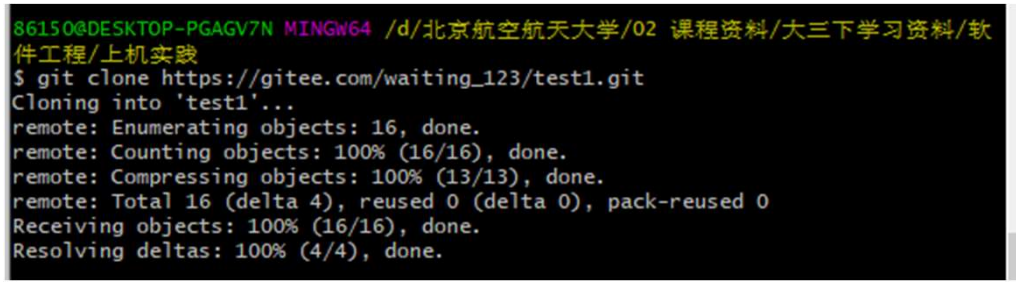
下载安装包进行Git GUI的安装

在本地找一个文件夹
右键，git bash here

然后找到云端的地址复制下来



然后输入“git clone 项目地址”，
开始clone远程仓库到本地



这就是你的本地仓库，
可以放在任何地方：

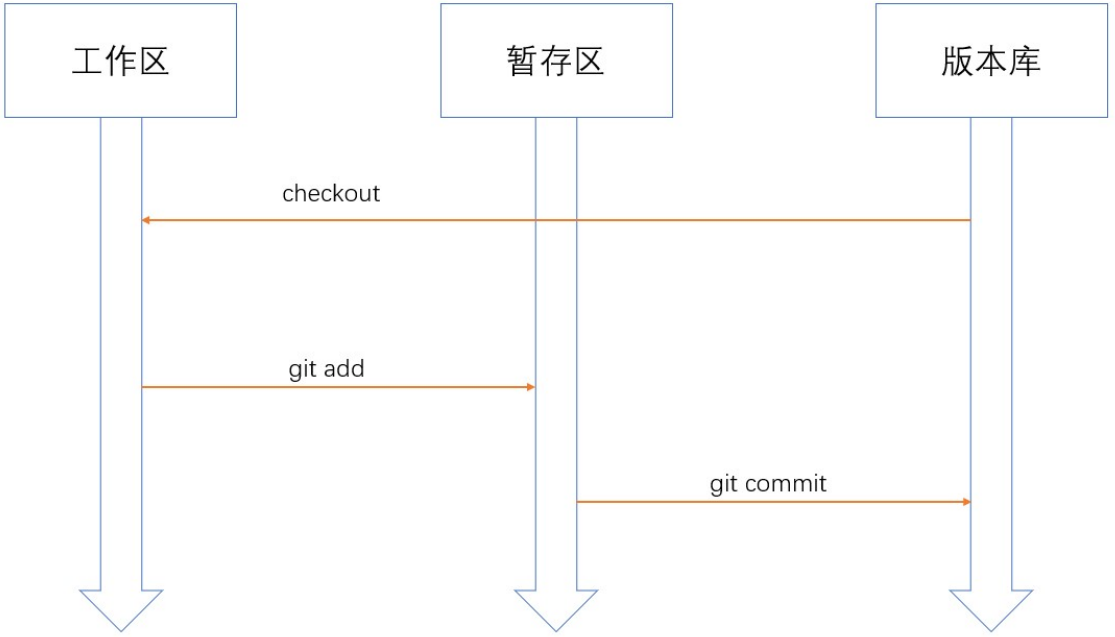
进入这个文件夹（一定要进去！），同样右键，git bash here，
输入“git status”查看状态

```
86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/蓝协小程序/新建文件夹/BuaaVolu
nteer (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

显示这个就正常了，文件夹内应该有正常的所有文件，而且注意提示，
当前位于master分支中，且已经与远程分支origin/master同步

Git有三个工作区域，分别是工作目录、暂存区域和本地仓库。
工作目录是当前进行工作的区域，文件修改但未提交，处于已修改状态（modified）；
暂存区域是运行git add命令后文件保存的区域，也就是下次提交要保存的文件，文件处于已暂存状态（staged）；
本地仓库即版本库，记录了工程提交的完整状态和内容，文件处于已提交状态（committed）



将本地仓库提交到远程

1.将项目下指定文件加入暂存区：

git add [文件名]

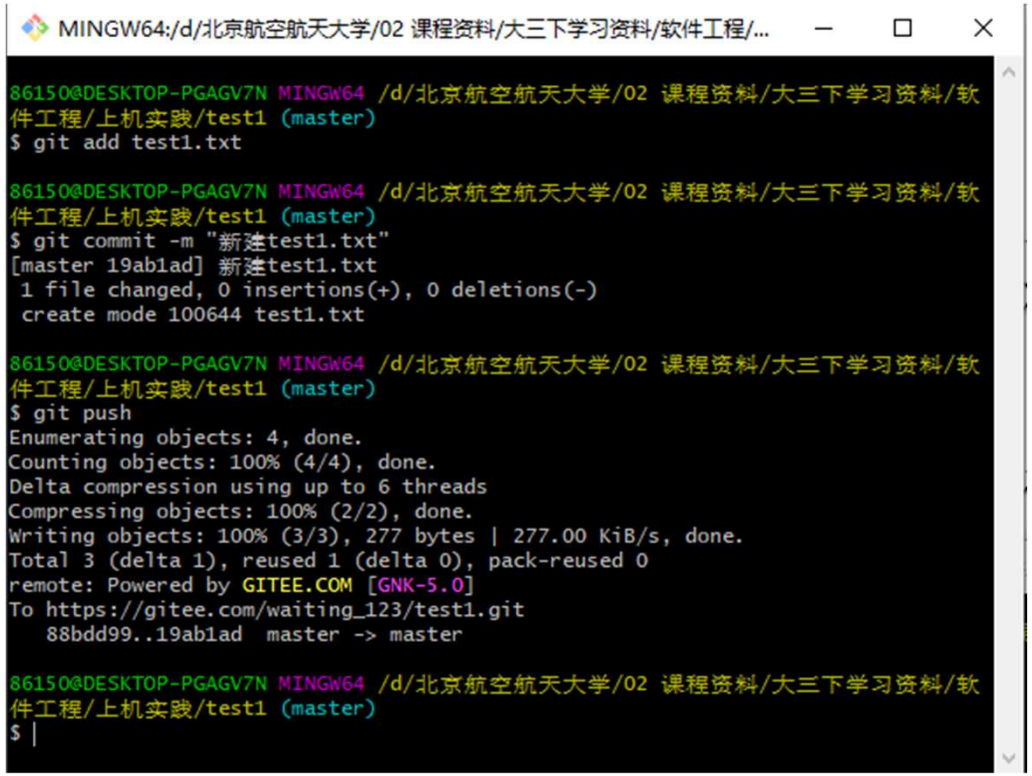
（或使用git add .将项目下所有文件加入暂存区）

2.将暂存区的内容打包成commit，提交到本地仓库：

git commit -m "[提交理由]"

3.将本地仓库的内容提交到远程仓库：

git push



将远程仓库同步到本地（若本地没有关联的分支）

1.将远程仓库内容同步到本地origin

git fetch

2. 新建一个分支指向远程分支（关联）

git checkout -b [分支名] origin/[分支名]

```
86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/02 课程资料/大三下学习资料/软
件工程/上机实践/test1 (master)
$ git fetch
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 14 (delta 5), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (14/14), 1.25 KiB | 3.00 KiB/s, done.
From https://gitee.com/waiting_123/test1
   0499d92..6970e6b  develop    -> origin/develop

86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/02 课程资料/大三下学习资料/软
件工程/上机实践/test1 (master)
$ git checkout -b develop origin/develop
Switched to a new branch 'develop'
Branch 'develop' set up to track remote branch 'develop' from 'origin'.
```

将远程仓库同步到本地（若本地存在与远程关联的分支）

1.将远程仓库内容同步到本地origin

git fetch

2.跳转到本地该分支

git checkout [分支名]

3.合并远程分支到本地分支

git merge [分支名] origin/[分支名]

另：也可以直接在与远程关联的分支使用pull指令，效果等于fetch+merge

git pull

```
86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/02 课程资料/大三下学习资料/软件工程/上机实践/test1 (master)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 273 bytes | 3.00 KiB/s, done.
From https://gitee.com/waiting_123/test1
   6970e6b..d45dbce  develop    -> origin/develop

86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/02 课程资料/大三下学习资料/软件工程/上机实践/test1 (master)
$ git checkout develop
Switched to branch 'develop'
Your branch is behind 'origin/develop' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/02 课程资料/大三下学习资料/软件工程/上机实践/test1 (develop)
$ git merge develop origin/develop
Updating 6970e6b..d45dbce
Fast-forward
 test1.txt | 1 +
 1 file changed, 1 insertion(+)

86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/02 课程资料/大三下学习资料/软件工程/上机实践/test1 (develop)
$
```

```
86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/02 课程资料/大三下学习资料/软件工程/上机实践/test1 (develop)
$ 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 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 274 bytes | 2.00 KiB/s, done.
From https://gitee.com/waiting_123/test1
   d45dbce..ea2b38b  develop    -> origin/develop
Updating d45dbce..ea2b38b
Fast-forward
 test1.txt | 1 +
 1 file changed, 1 insertion(+)
```

分支合并与解决冲突

1. 合并分支2到分支1

git merge [分支名1] [分支名2]

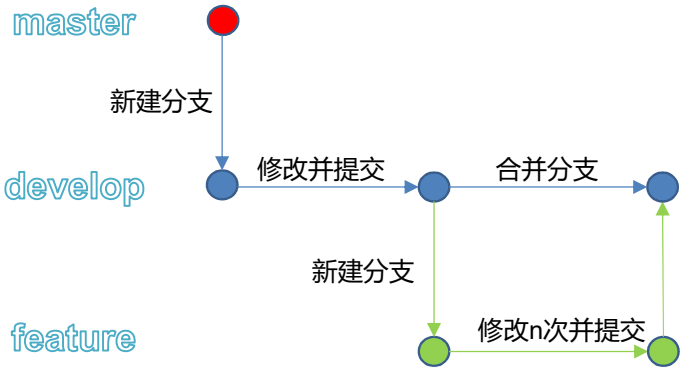
```
86150@DESKTOP-PGAGV7N MINGW64 /d/北京航空航天大学/02 课程资料/大三下学习资料/软件工程/上机实践/test1 (develop)
$ git merge develop origin/develop
Auto-merging test1.txt
CONFLICT (content): Merge conflict in test1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

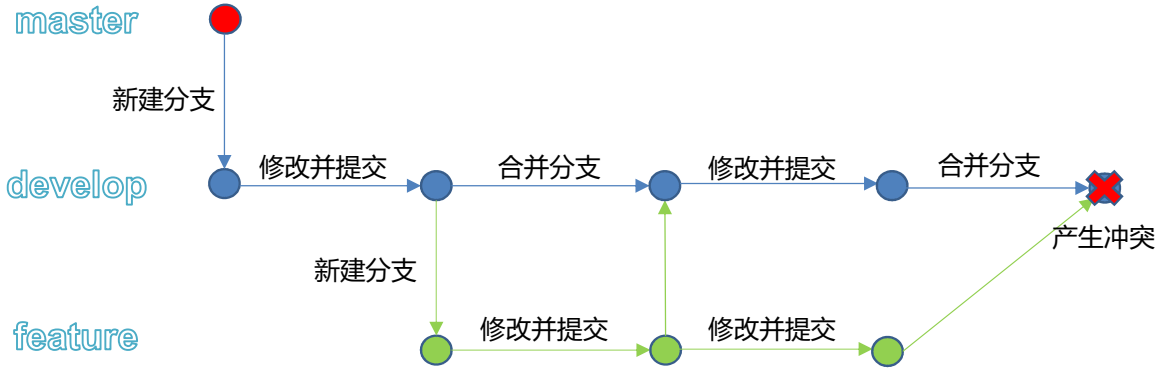
2. 使用vi编辑器填写合并理由

3. 合并过程中（包括git pull自动进行的合并）可能产生冲突，此时打开除记事本外的高级IDE一般就会提供合并冲突的提示，需要手动解决后再add/commit/push

```
233466788
采用当前更改 | 采用传入的更改 | 保留双方更改 | 比较变更
<<<<<<< HEAD (当前更改)
566
=======
788
>>>>>>> origin/develop (传入的更改)
```

为什么会出现冲突?


master
新建分支
develop
修改并提交
合并分支
新建分支
feature
修改n次并提交


master
新建分支
develop
修改并提交
合并分支
修改并提交
合并分支
产生冲突
新建分支
feature
修改并提交
修改并提交

学习参考

- Git的核心概念
  https://lufficc.com/blog/the-core-conception-of-git
- 吕云翔老师的《软件工程项目实训教程》
- 资料中的GitHelp.md