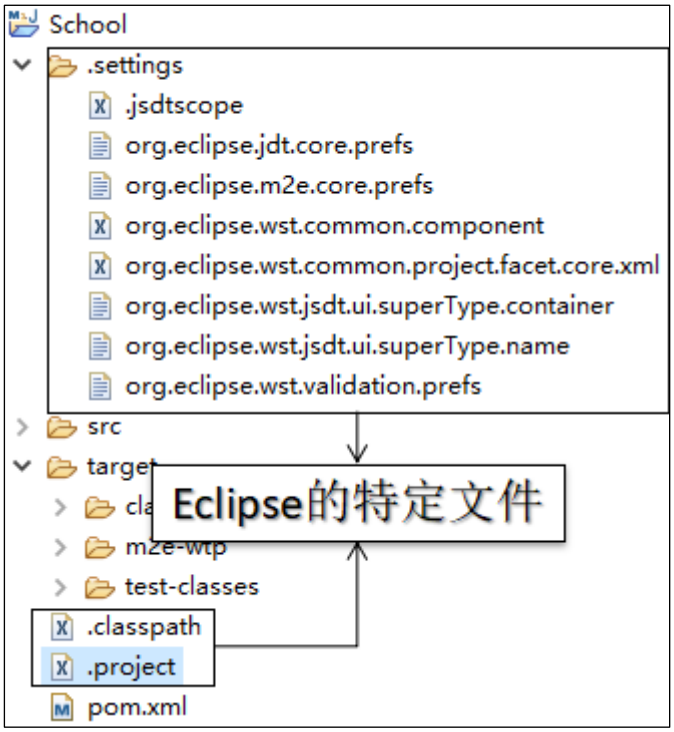
接下来再往远程仓库 push 东西的时候使用 SSH 连接就不需要登录了。

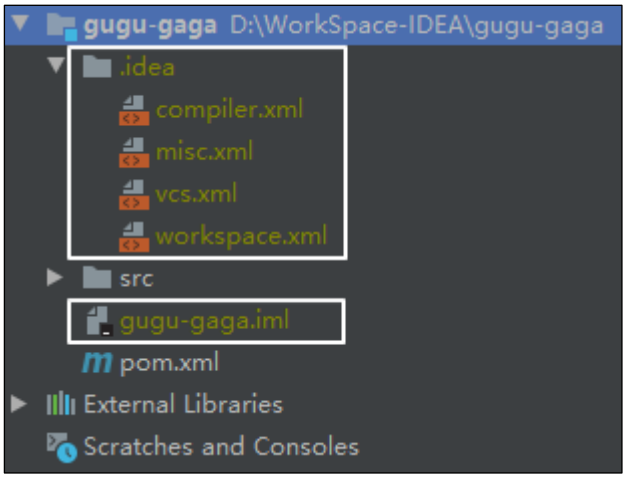
第 7 章 IDEA 集成 Git

7.1 配置 Git 忽略文件

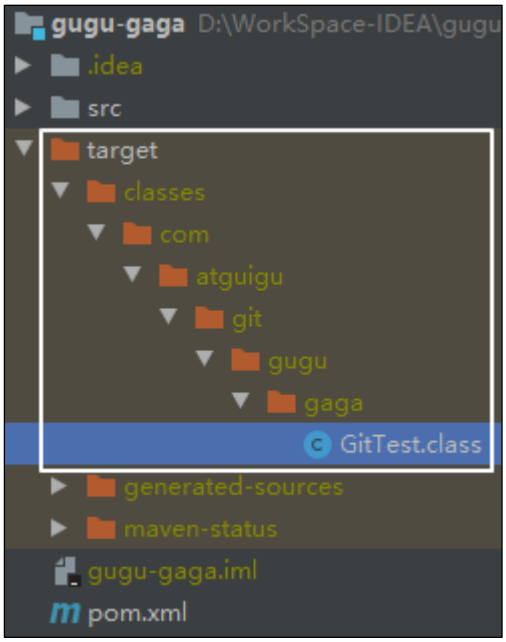
1）Eclipse 特定文件



2）IDEA 特定文件



3）Maven 工程的 target 目录



问题 1:为什么要忽略他们?

答：与项目的实际功能无关，不参与服务器上部署运行。把它们忽略掉能够屏蔽 IDE 工具之间的差异。

问题 2：怎么忽略?

1）创建忽略规则文件 xxxx.ignore（前缀名随便起，建议是 git.ignore）

这个文件的存放位置原则上在哪里都可以，为了便于让~/.gitconfig 文件引用，建议也放在用户家目录下

git.ignore 文件模版内容如下：

```
# Compiled class file
*.class
```

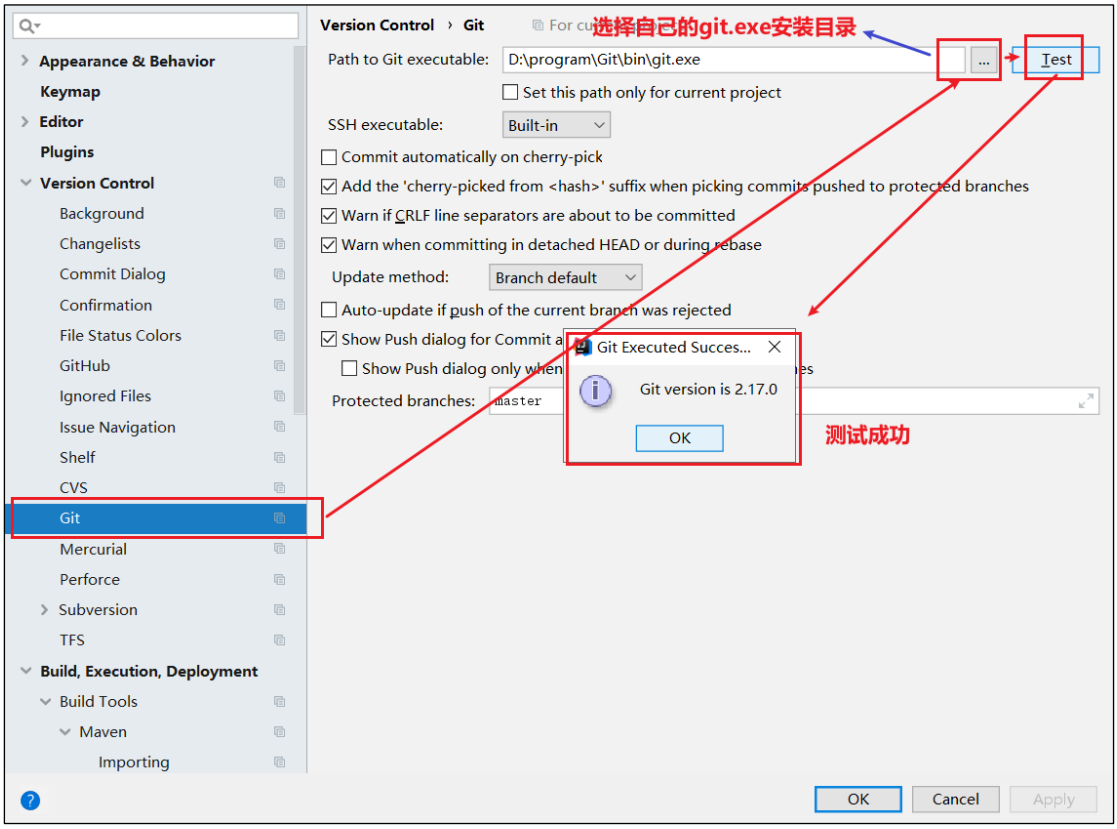
```
# Log file
*.log

# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

#         virtual         machine         crash         logs,         see
http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*

.classpath
.project
.settings
target
.idea
*.iml
```

2）在.gitconfig 文件中引用忽略配置文件（此文件在 Windows 的家目录中）

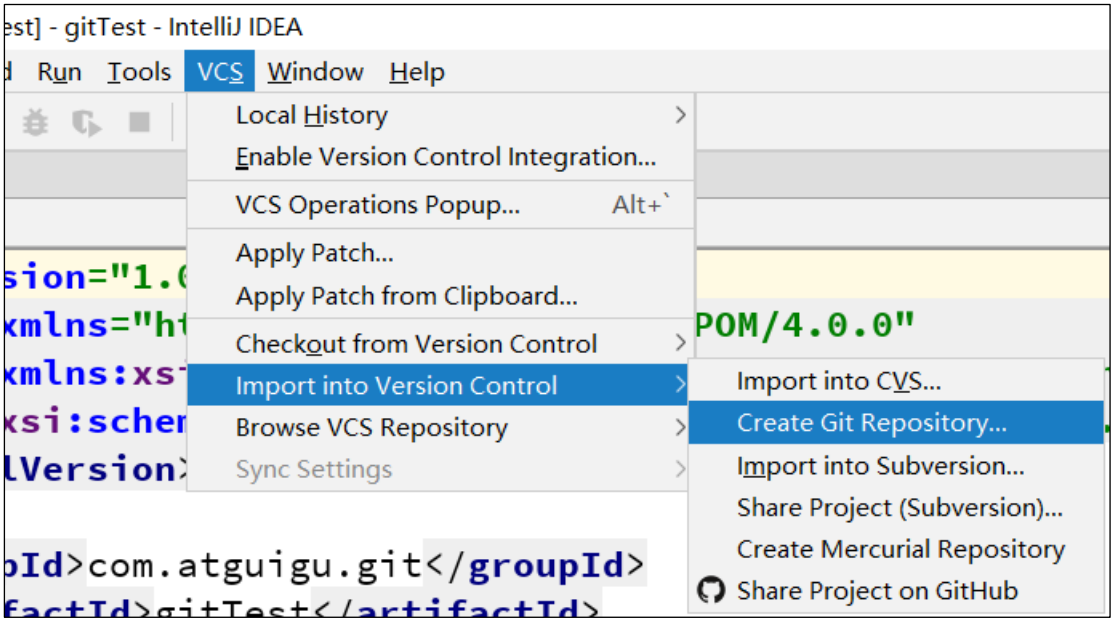
```
[user]
    name = Layne
    email = Layne@atguigu.com
[core]
    excludesfile = C:/Users/asus/git.ignore
```

注意：这里要使用“正斜线（/）”，不要使用“反斜线（\）”

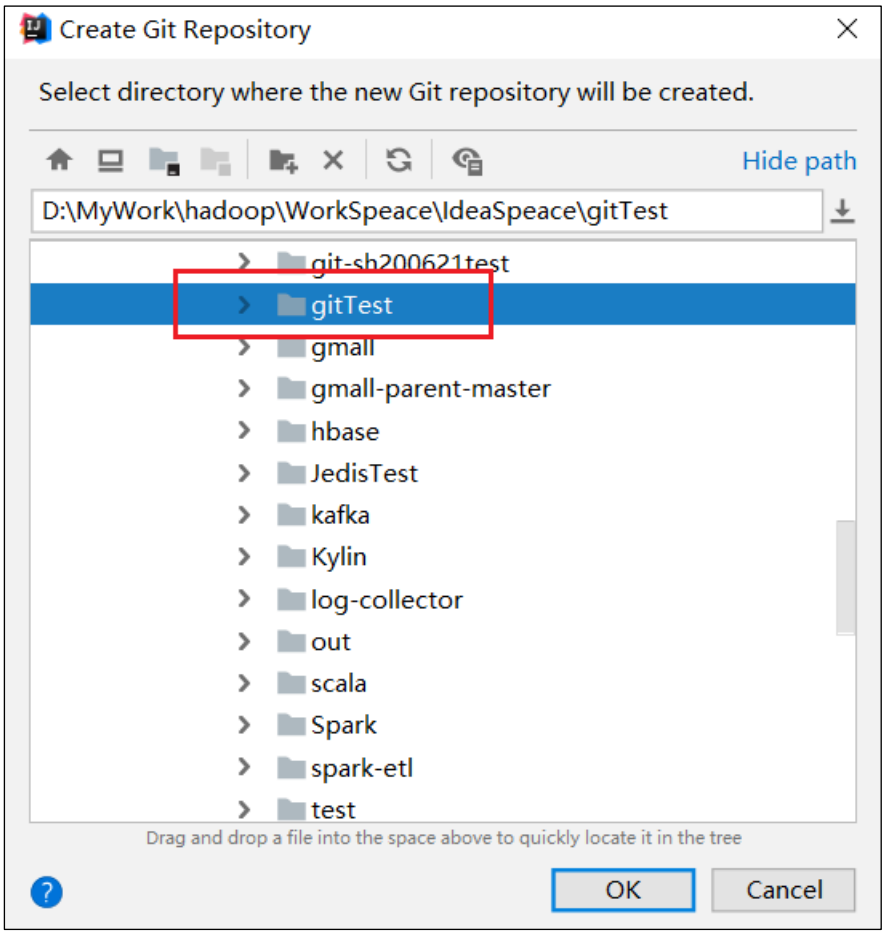
7.2 定位 Git 程序



7.3 初始化本地库

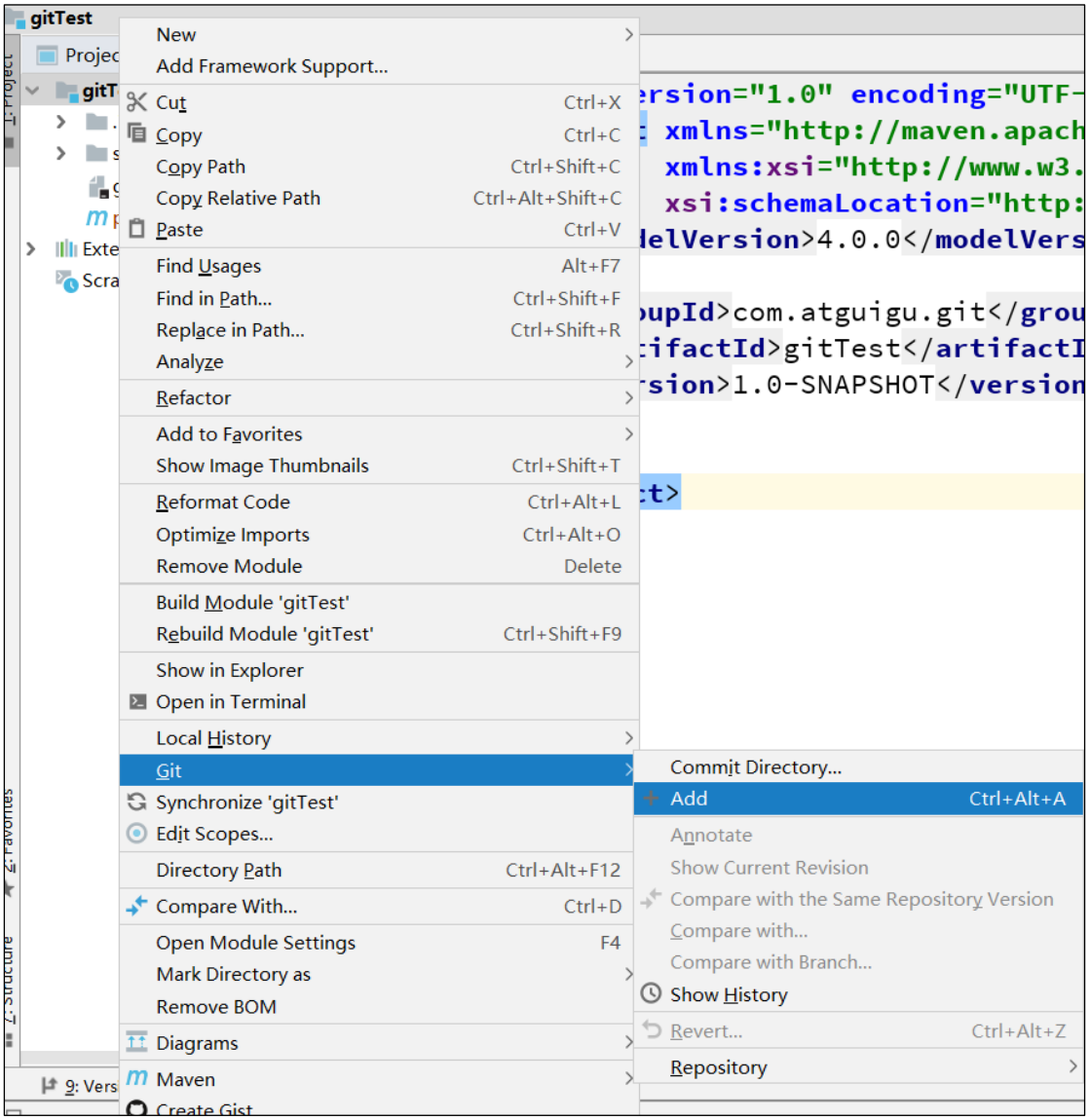


选择要创建 Git 本地仓库的工程。

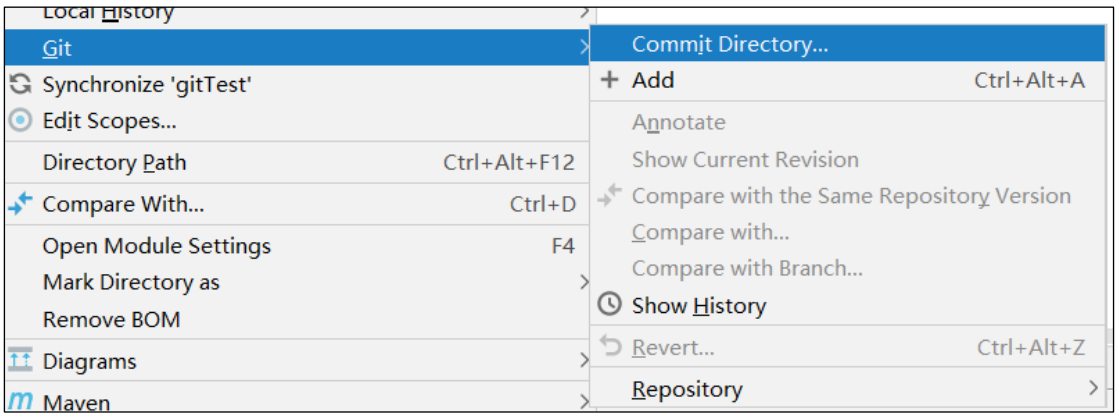


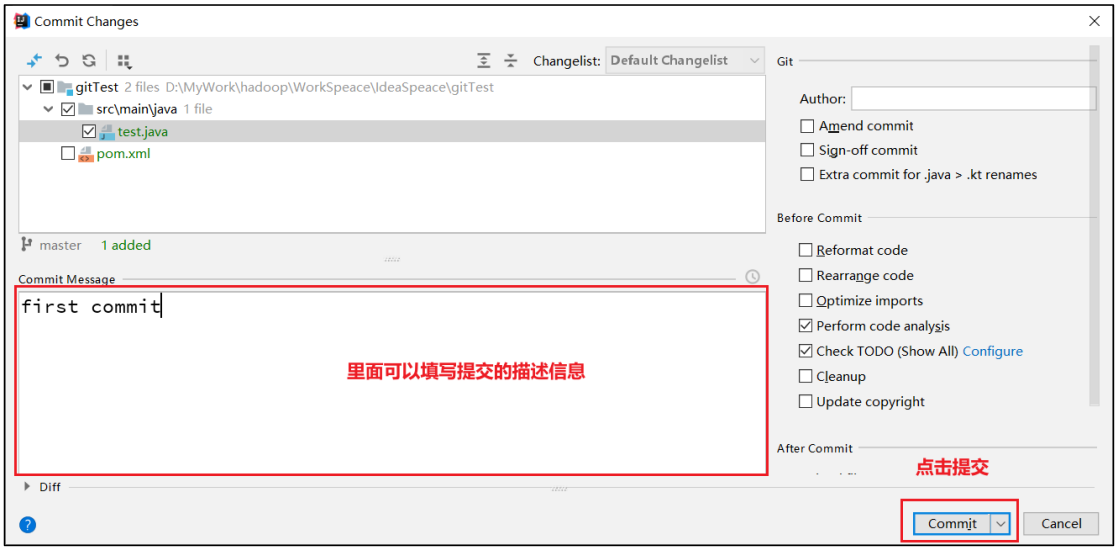
7.4 添加到暂存区

右键点击项目选择 Git -> Add 将项目添加到暂存区。



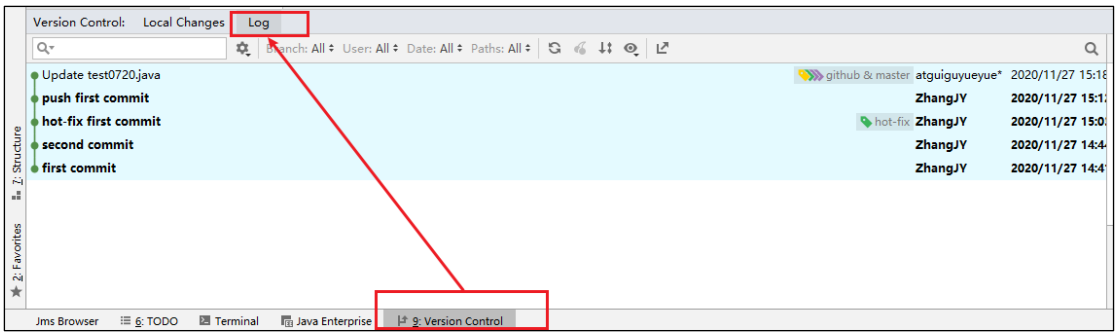
7.5 提交到本地库



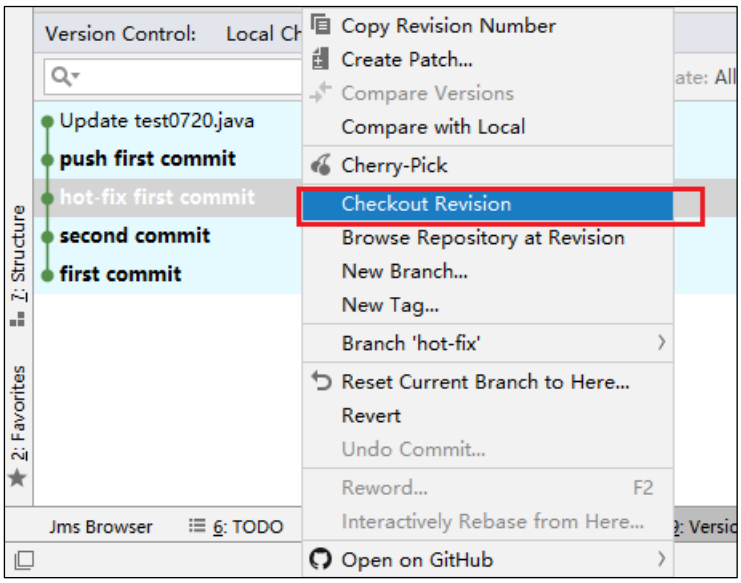


7.6 切换版本

在 IDEA 的左下角，点击 Version Control，然后点击 Log 查看版本

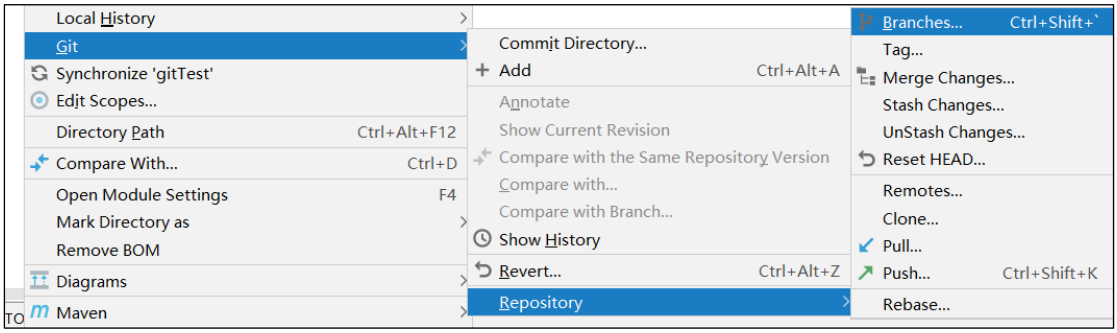


右键选择要切换的版本，然后在菜单里点击 Checkout Revision。

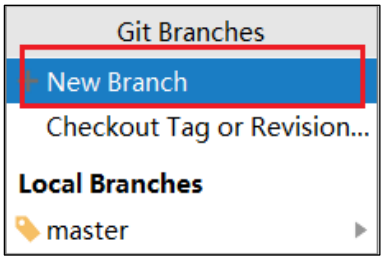


7.7 创建分支

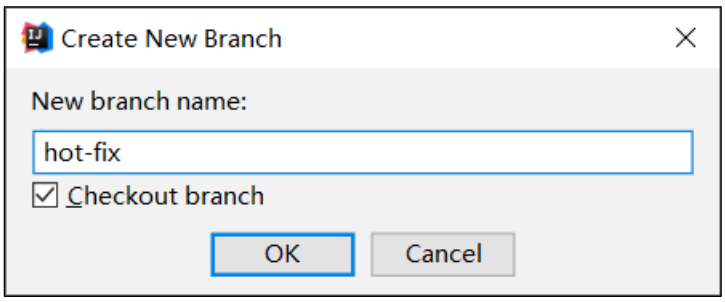
选择 Git，在 Repository 里面，点击 Branches 按钮。



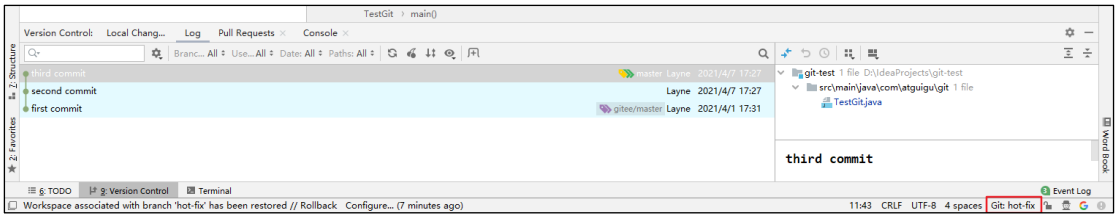
在弹出的 Git Branches 框里，点击 New Branch 按钮。



填写分支名称，创建 hot-fix 分支。

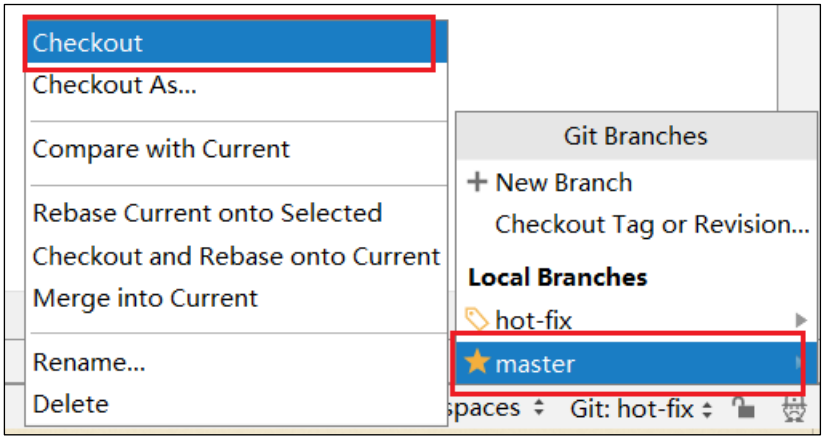


然后再 IDEA 的右下角看到 hot-fix，说明分支创建成功，并且当前已经切换成 hot-fix 分支

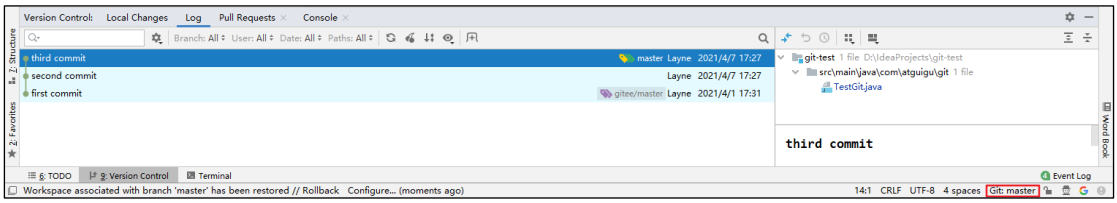


7.8 切换分支

在 IDEA 窗口的右下角，切换到 master 分支。

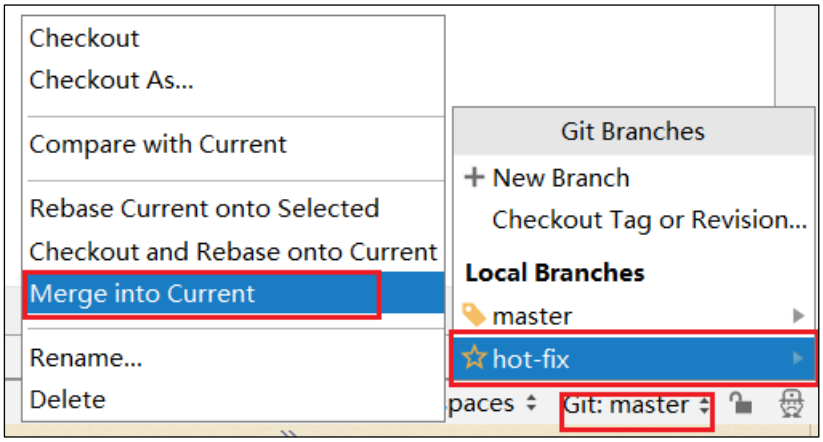


然后在 IDEA 窗口的右下角看到了 master，说明 master 分支切换成功。

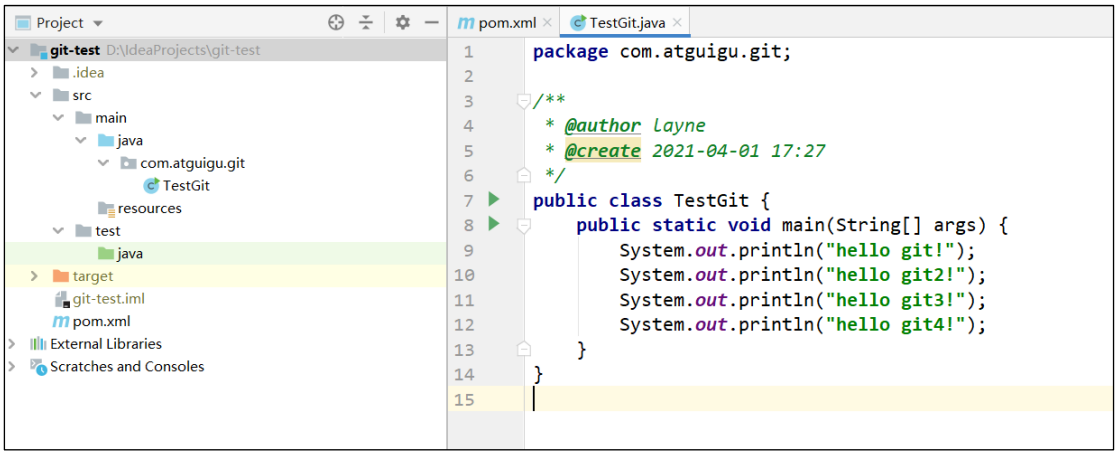


7.9 合并分支

在 IDEA 窗口的右下角，将 hot-fix 分支合并到当前 master 分支。

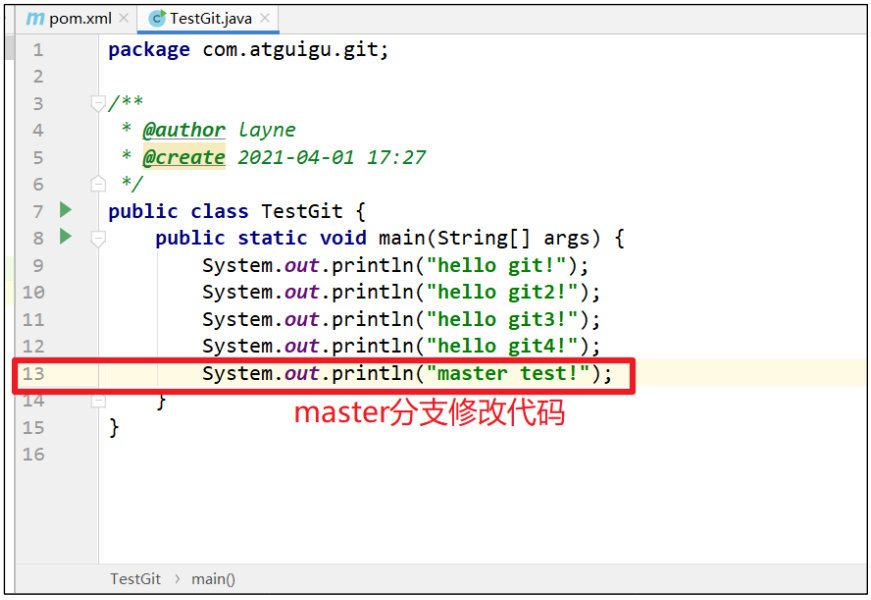


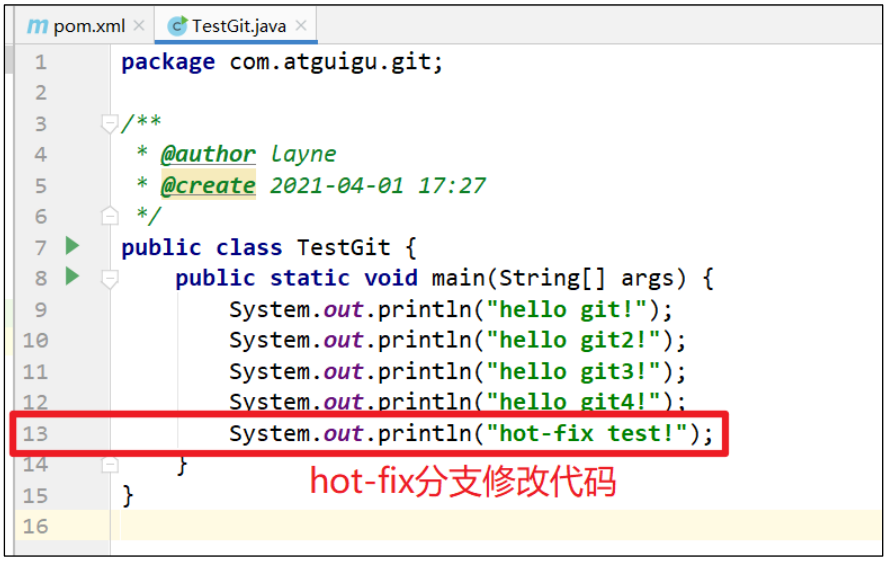
如果代码没有冲突，分支直接合并成功，分支合并成功以后，代码自动提交，无需手动提交本地库。



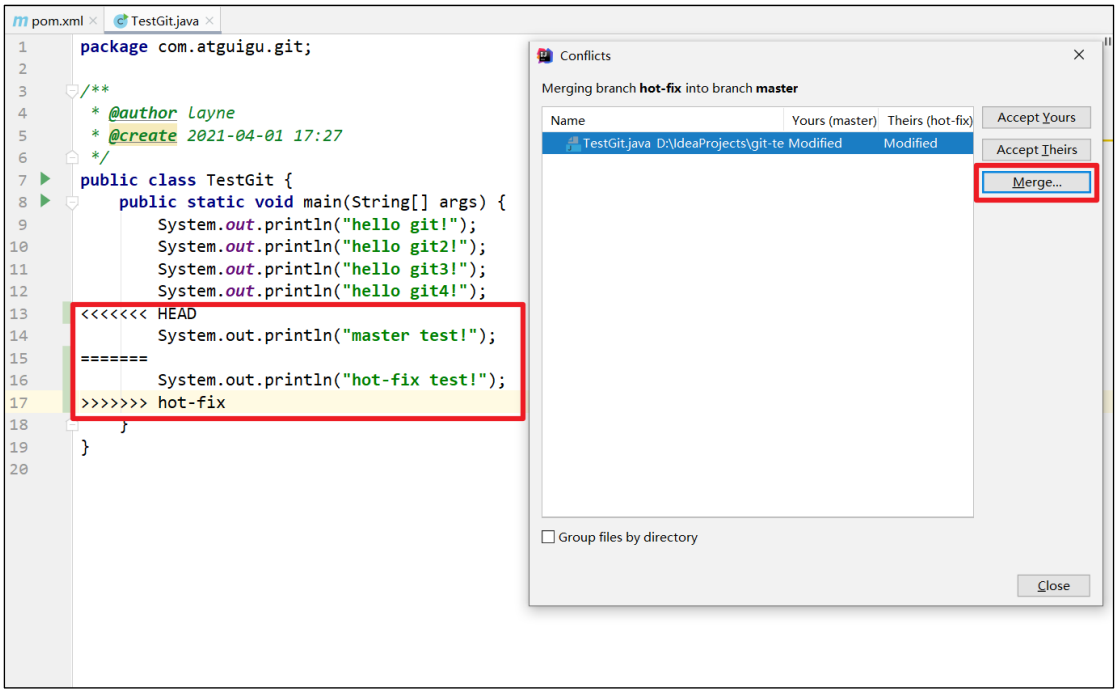
7.10 解决冲突

如图所示，如果 master 分支和 hot-fix 分支都修改了代码，在合并分支的时候就会发生冲突。



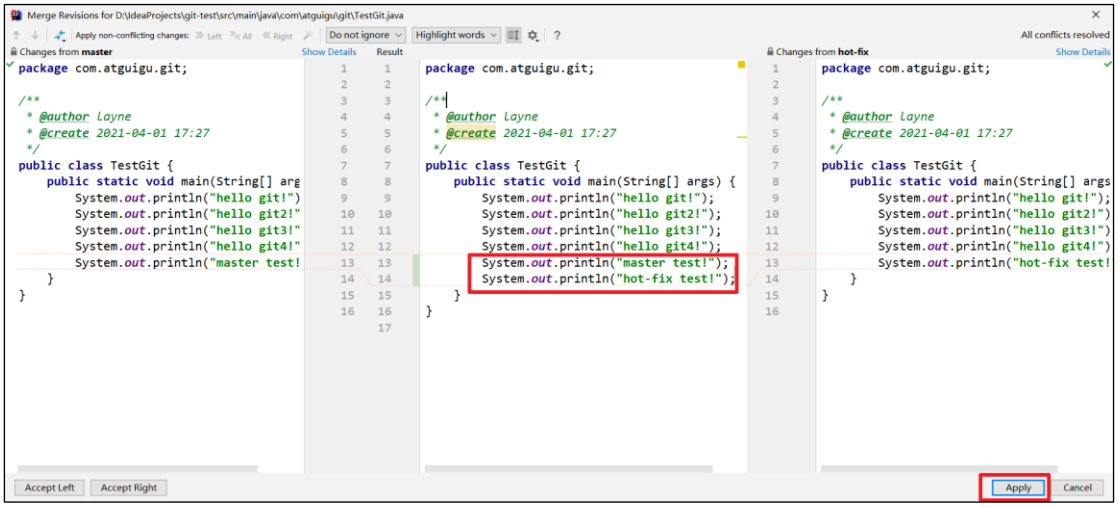


我们现在站在 master 分支上合并 hot-fix 分支，就会发生代码冲突。

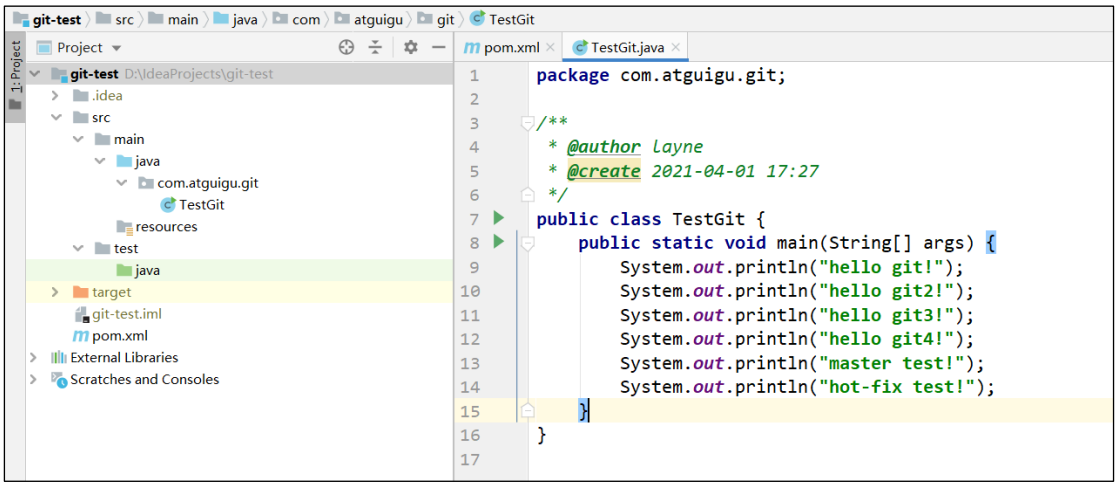


点击 Conflicts 框里的 Merge 按钮，进行手动合并代码。

手动合并完代码以后，点击右下角的 Apply 按钮。

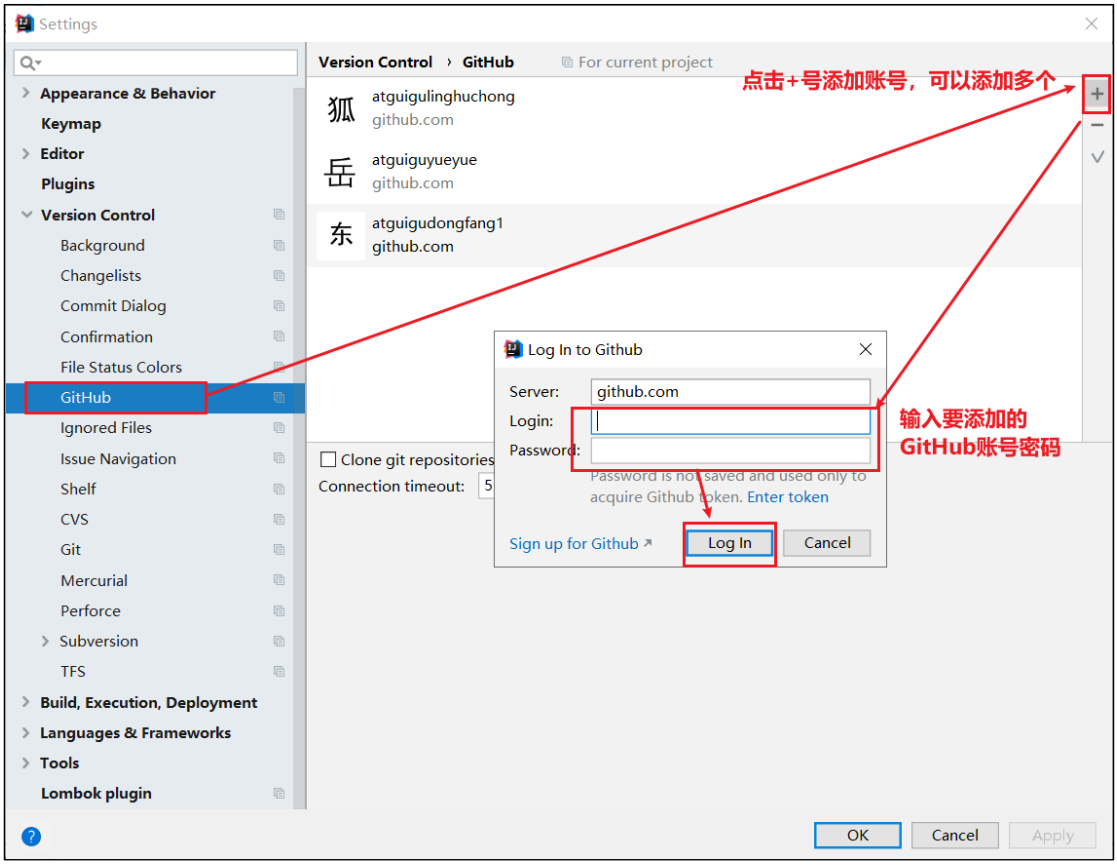


代码冲突解决，自动提交本地库。

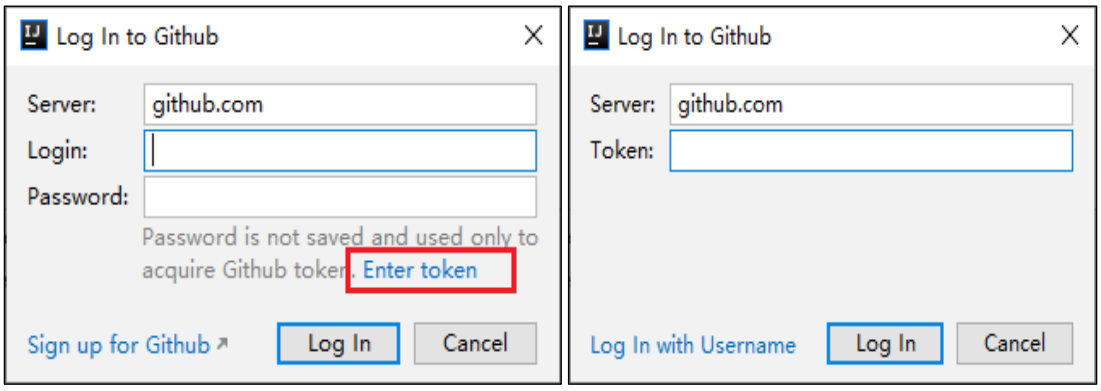


第 8 章 IDEA 集成 GitHub

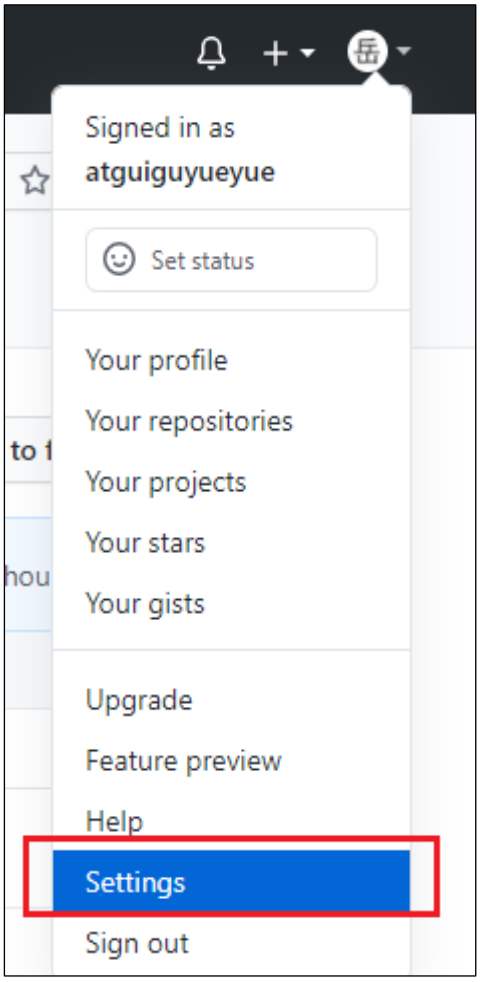
8.1 设置 GitHub 账号

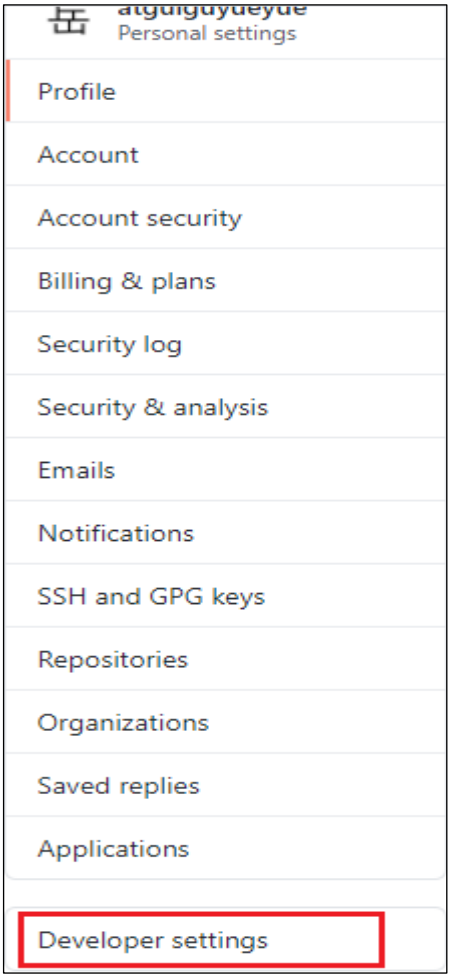


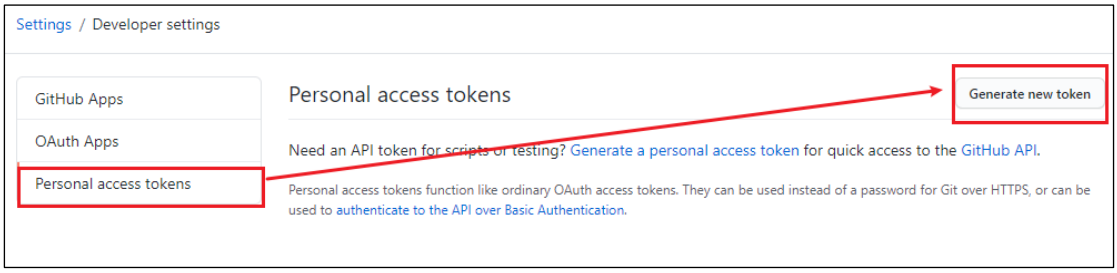
如果出现 401 等情况连接不上的，是因为网络原因，可以使用以下方式连接：

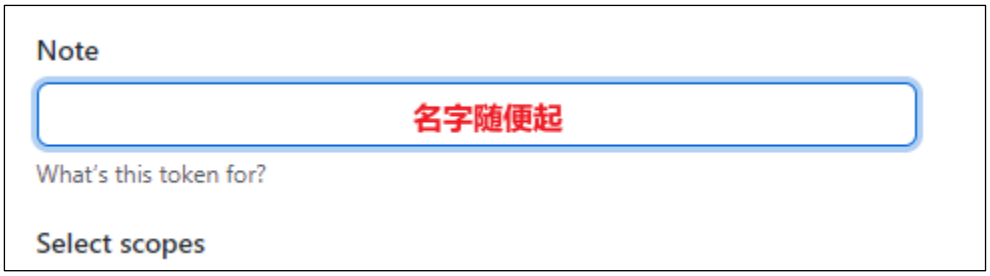


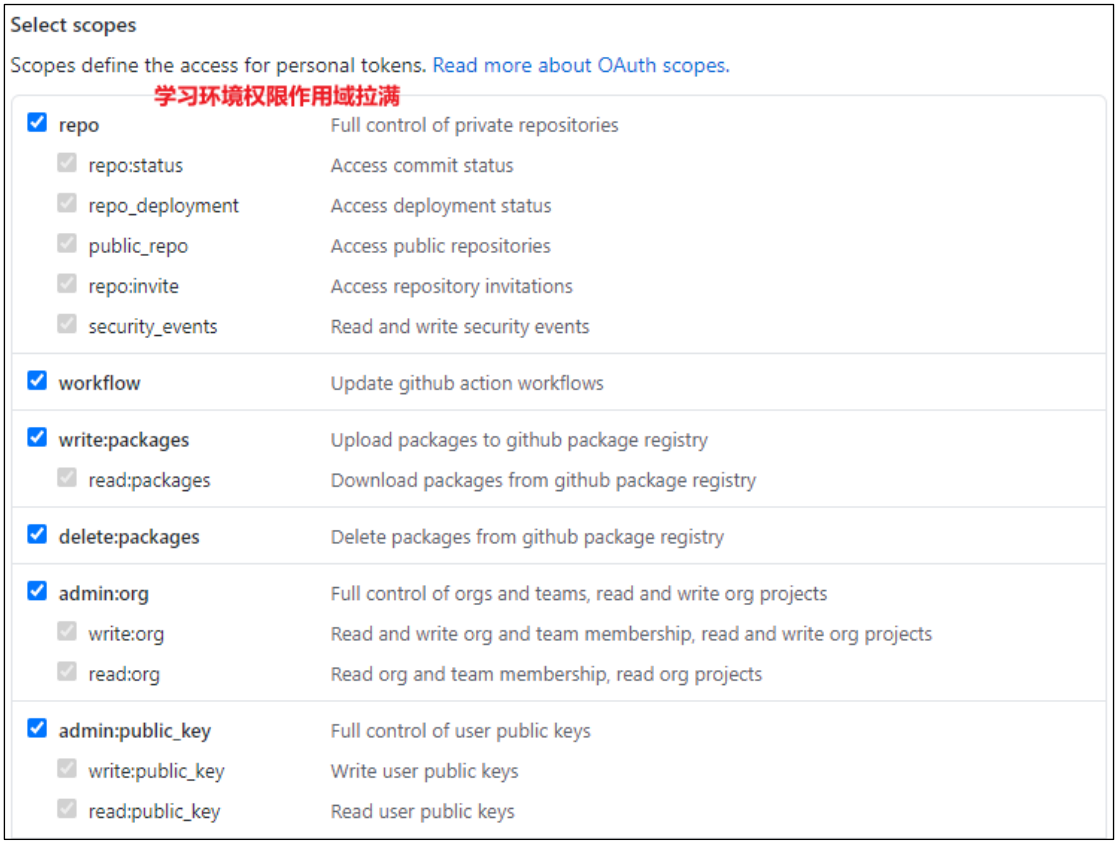
然后去 GitHub 账户上设置 token。



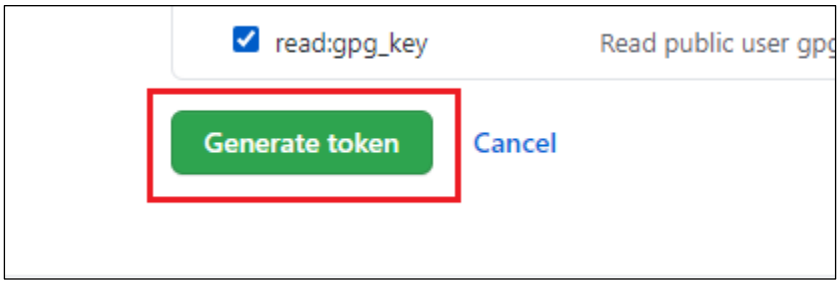




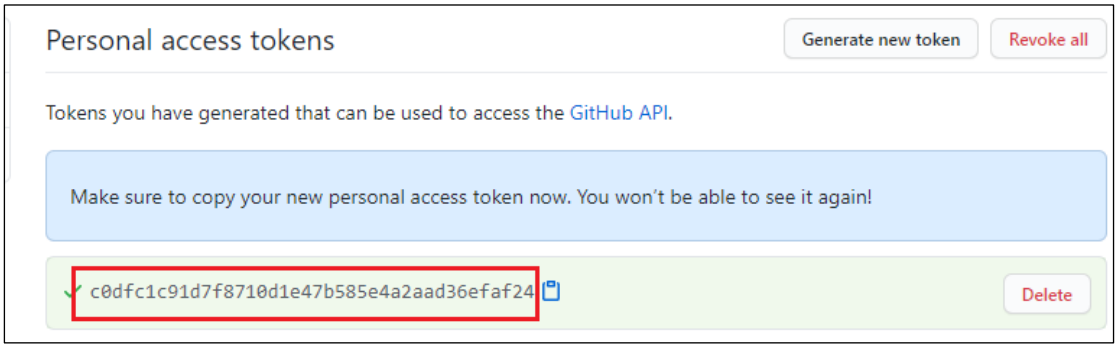




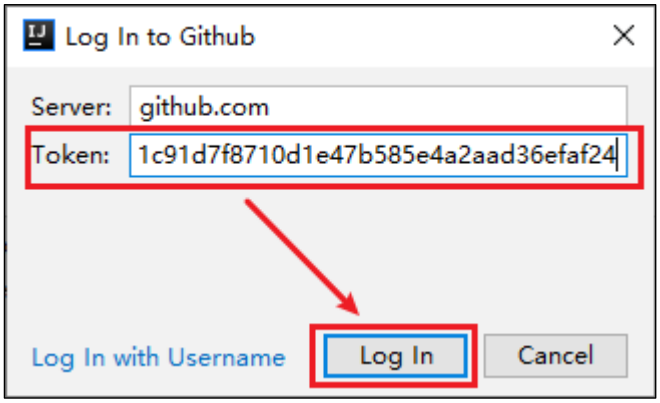
点击生成 token。



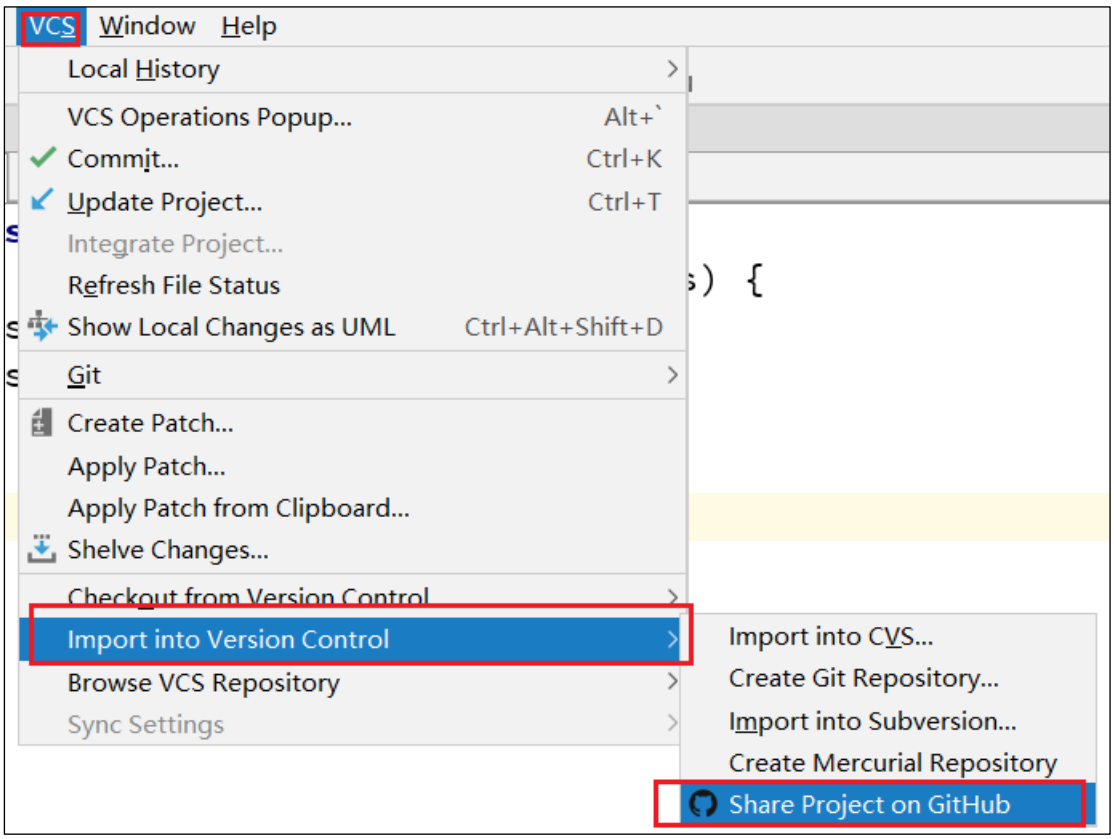
复制红框中的字符串到 idea 中。

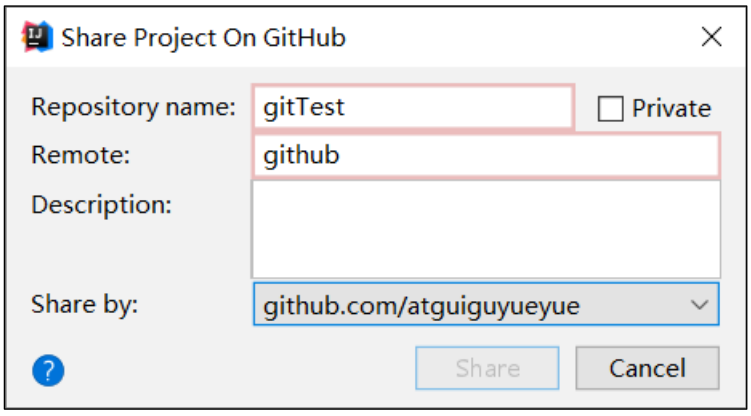


点击登录。

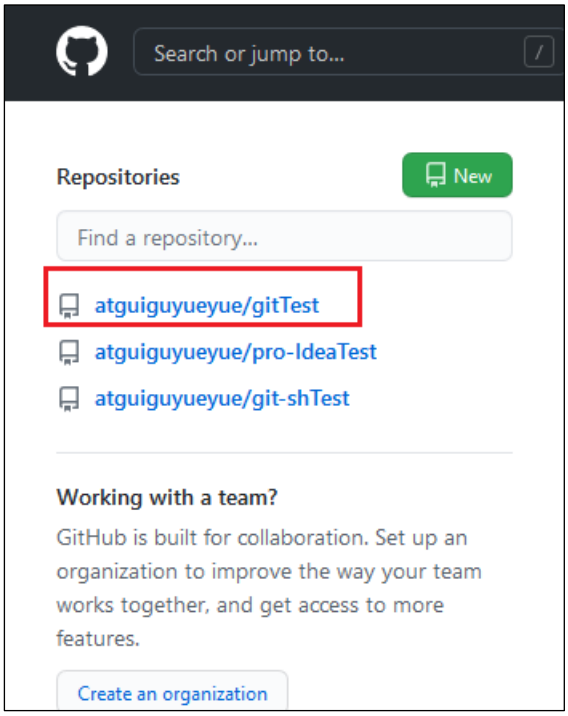


8.2 分享工程到 GitHub



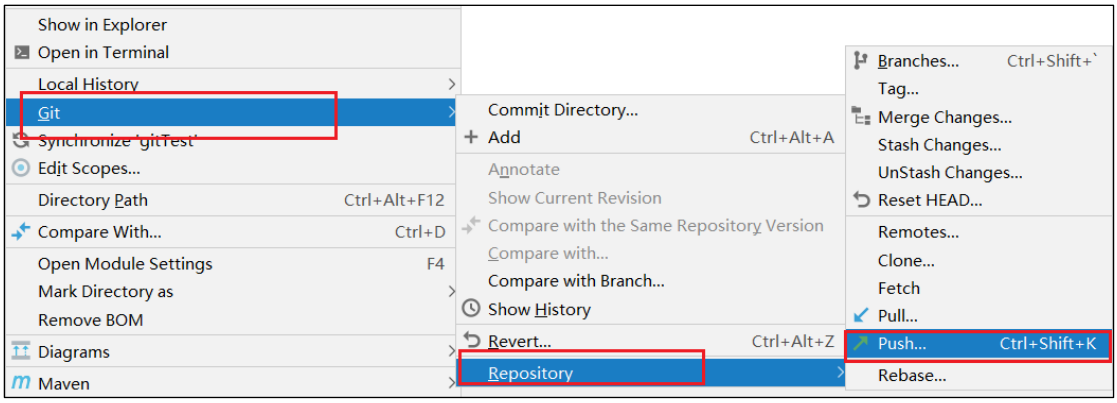


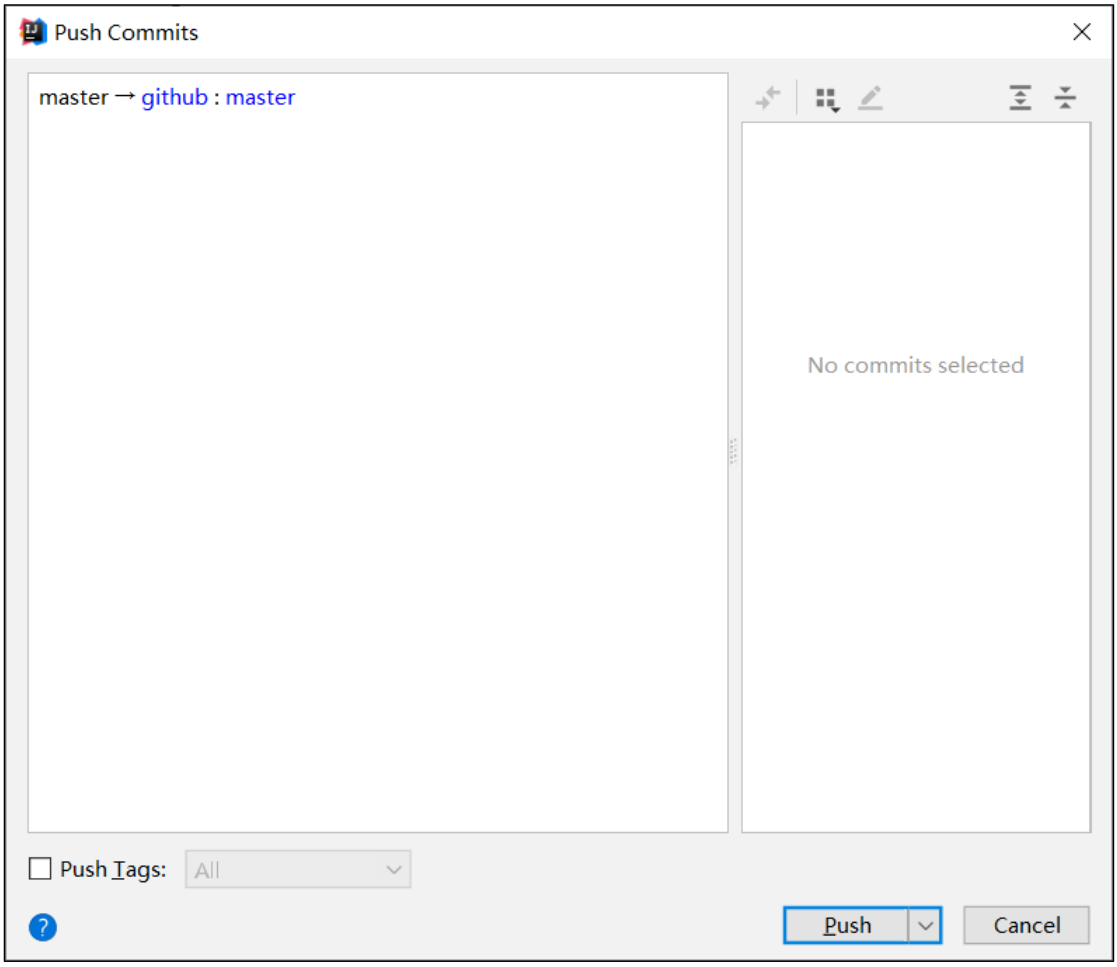
来到 GitHub 中发现已经帮我们创建好了 gitTest 的远程仓库。

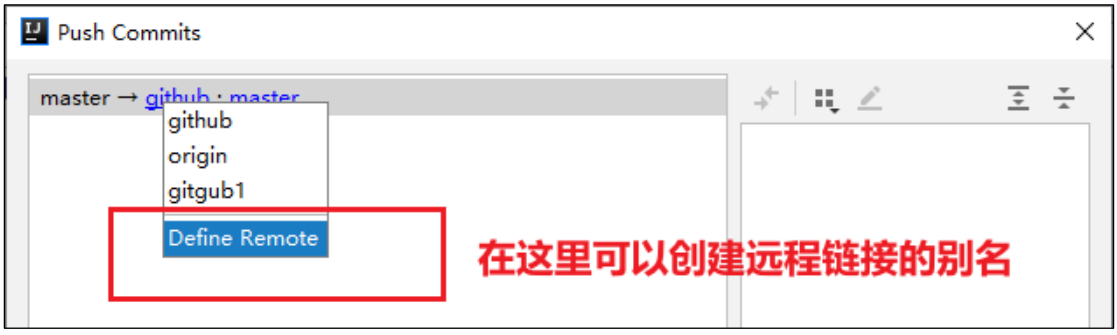


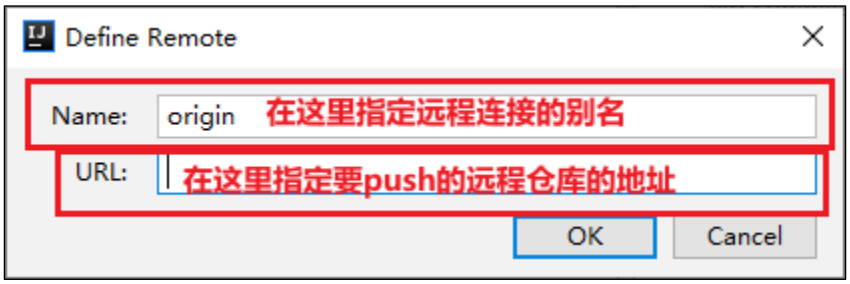
8.3 push 推送本地库到远程库

右键点击项目，可以将当前分支的内容 push 到 GitHub 的远程仓库中。





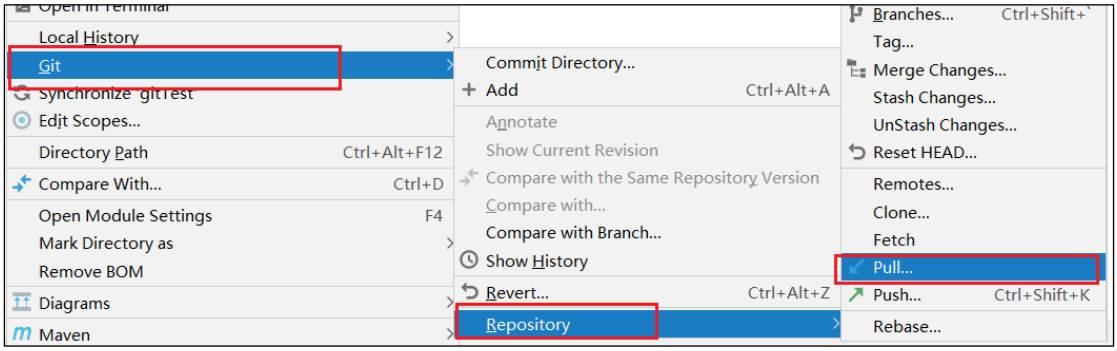


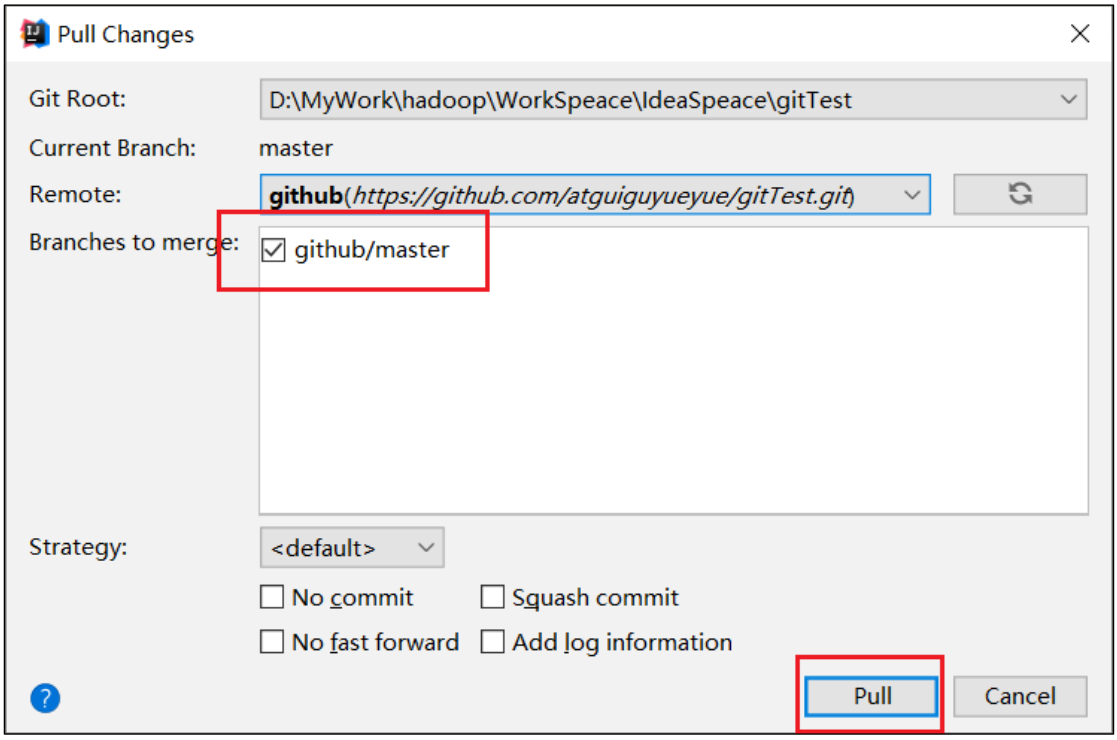


注意：push 是将本地库代码推送到远程库，如果本地库代码跟远程库代码版本不一致，push 的操作是会被拒绝的。也就是说，要想 push 成功，一定要保证本地库的版本要比远程库的版本高！因此一个成熟的程序员在动手改本地代码之前，一定会先检查下远程库跟本地代码的区别！如果本地的代码版本已经落后，切记要先 pull 拉取一下远程库的代码，将本地代码更新到最新以后，然后再修改，提交，推送！

8.4 pull 拉取远程库到本地库

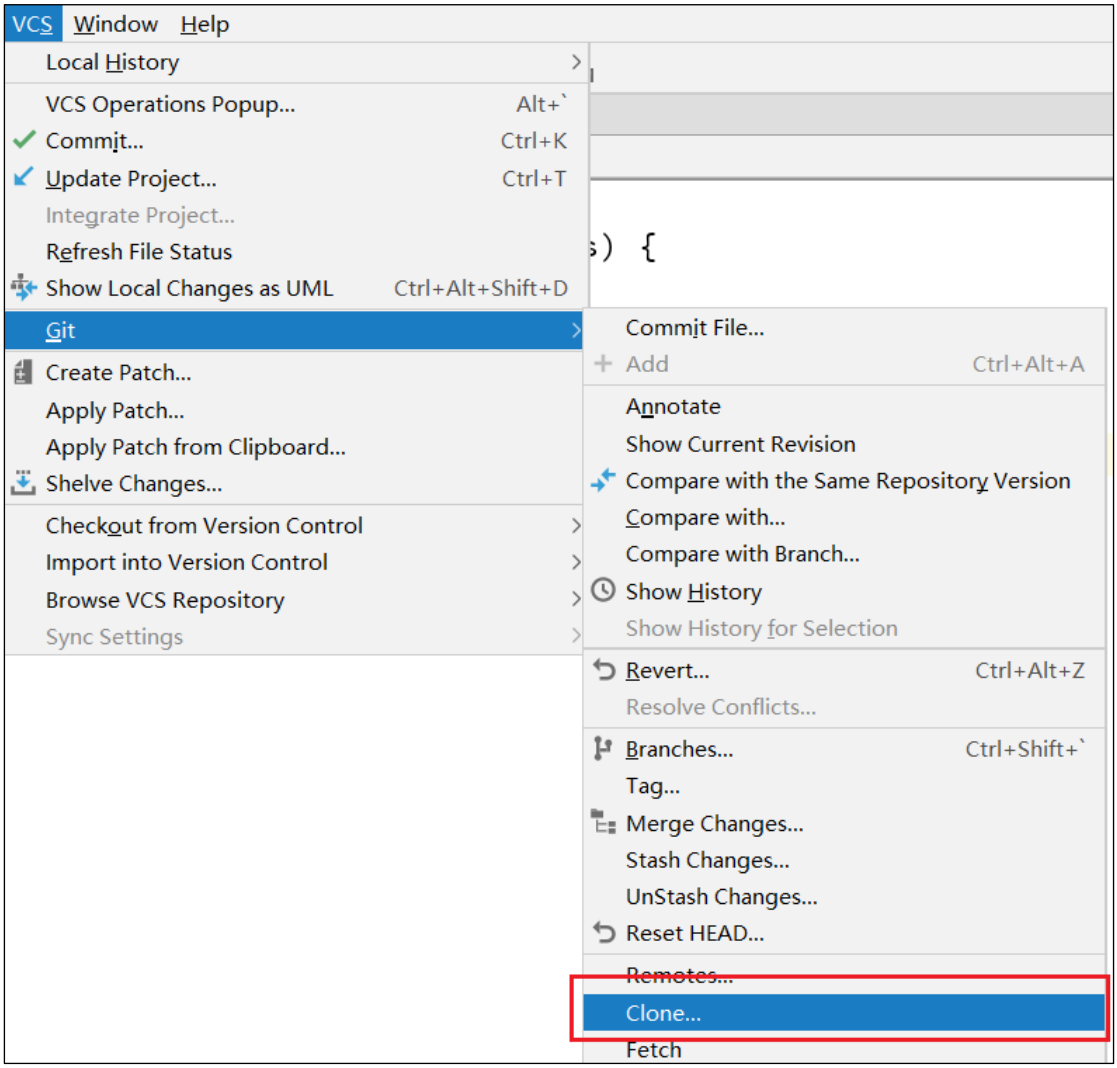
右键点击项目，可以将远程仓库的内容 pull 到本地仓库。



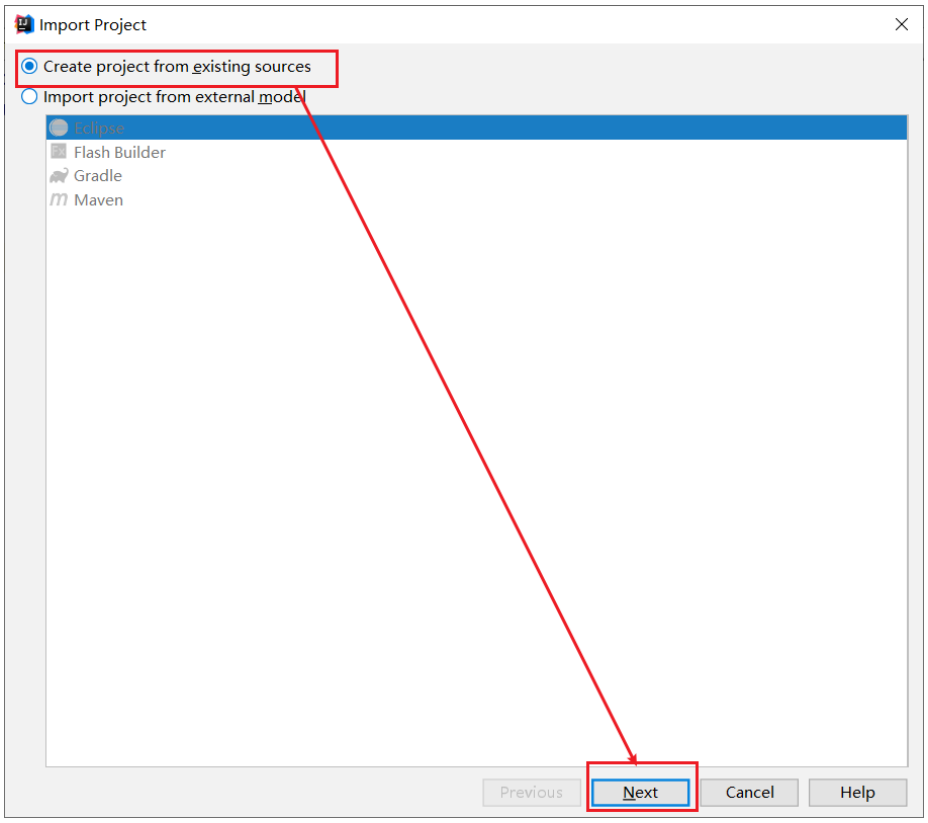


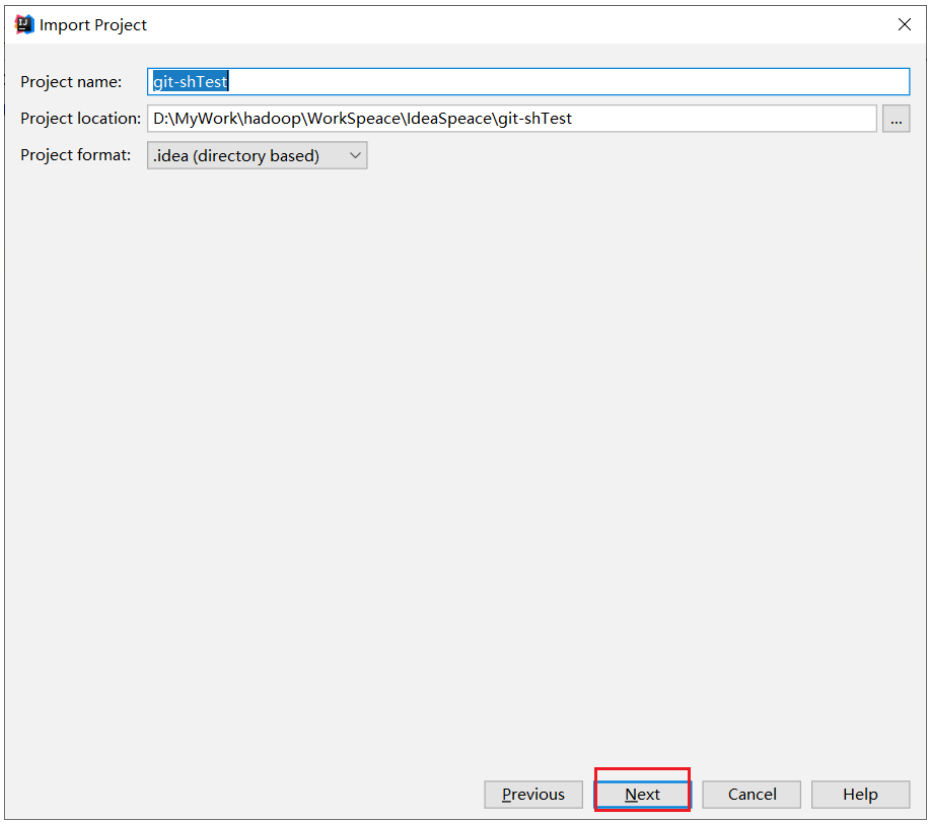
注意：pull 是拉取远端仓库代码到本地，如果远程库代码和本地库代码不一致，会自动合并，如果自动合并失败，还会涉及到手动解决冲突的问题。

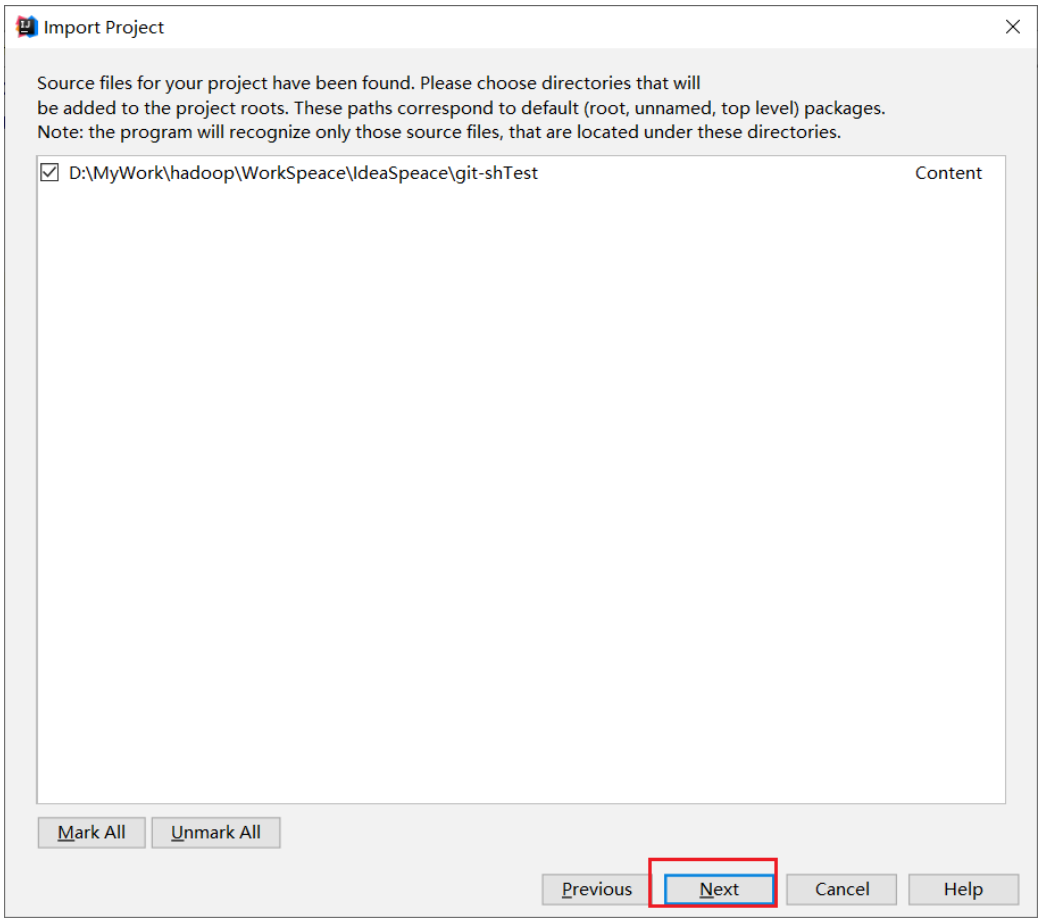
8.5 clone 克隆远程库到本地

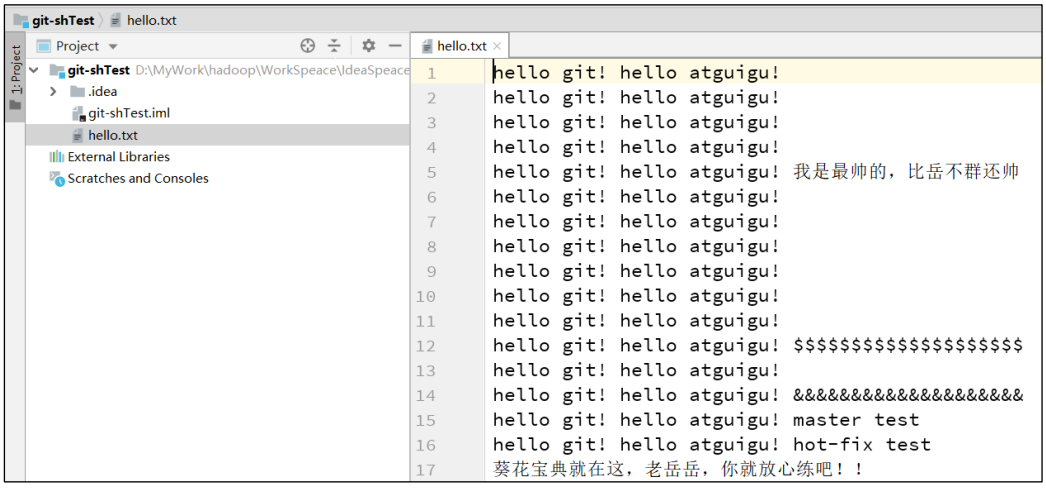


为 clone 下来的项目创建一个工程，然后点击 Next。









第 9 章 国内代码托管中心-码云

9.1 简介

众所周知，GitHub 服务器在国外，使用 GitHub 作为项目托管网站，如果网速不好的话，严重影响使用体验，甚至会出现登录不上的情况。针对这个情况，大家也可以使用国内的项目托管网站-码云。

码云是开源中国推出的基于 Git 的代码托管服务中心，网址是 https://gitee.com/ ，使用方式跟 GitHub 一样，而且它还是一个中文网站，如果你英文不是很好它是最好的选择。

9.2 码云帐号注册和登录

进入码云官网地址：https://gitee.com/，点击注册 Gitee

输入个人信息，进行注册即可。

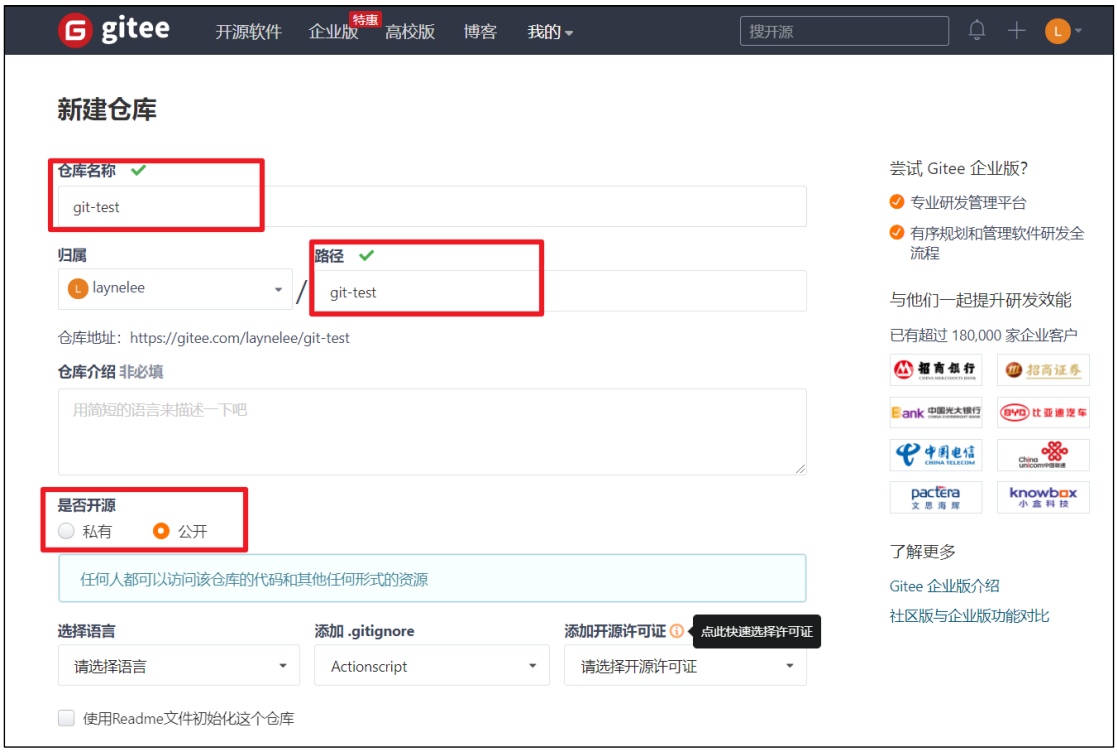
帐号注册成功以后，直接登录。

登录以后，就可以看到码云官网首页了。

9.3 码云创建远程库

点击首页右上角的加号，选择下面的新建仓库

填写仓库名称，路径和选择是否开源（共开库或私有库）



最后根据需求选择分支模型，然后点击创建按钮。

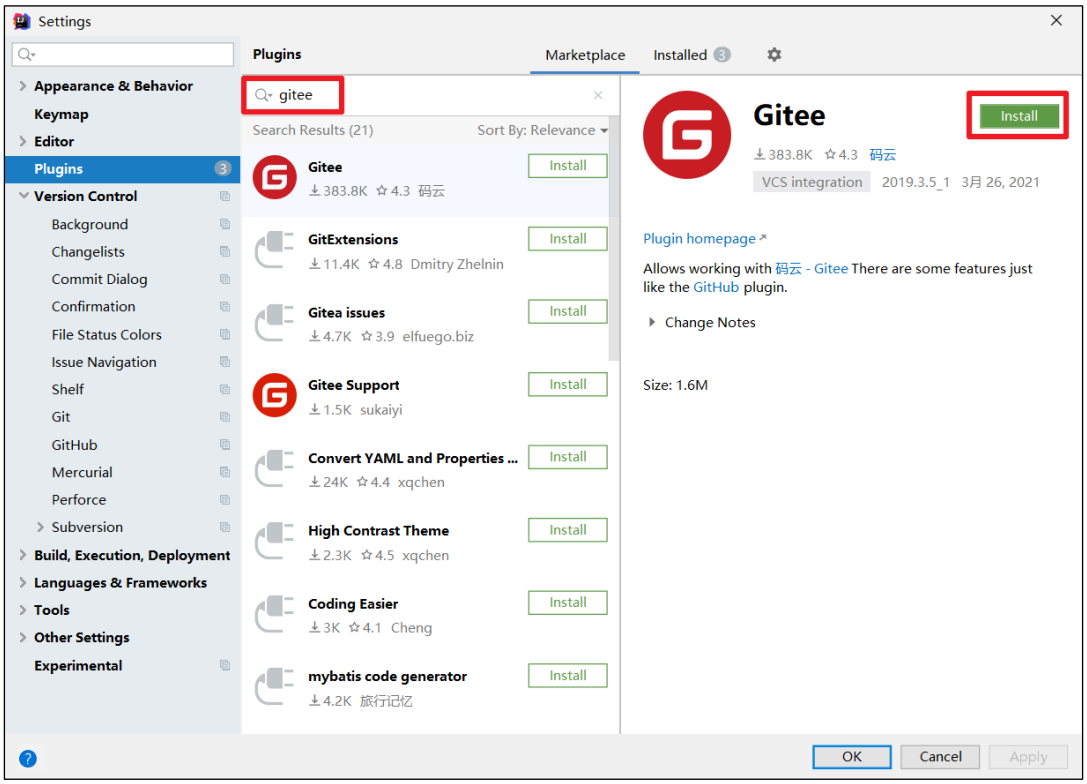
远程库创建好以后，就可以看到 HTTPS 和 SSH 的链接。

9.4 IDEA 集成码云

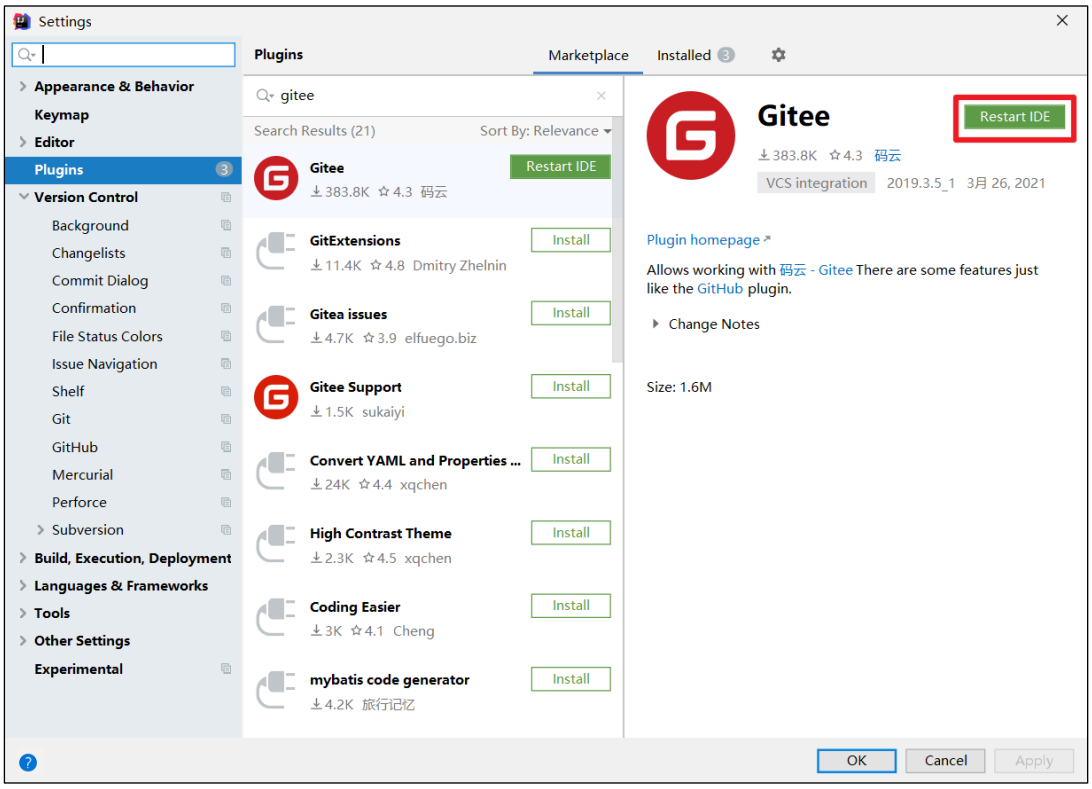
9.4.1 IDEA 安装码云插件

Idea 默认不带码云插件，我们第一步要安装 Gitee 插件。

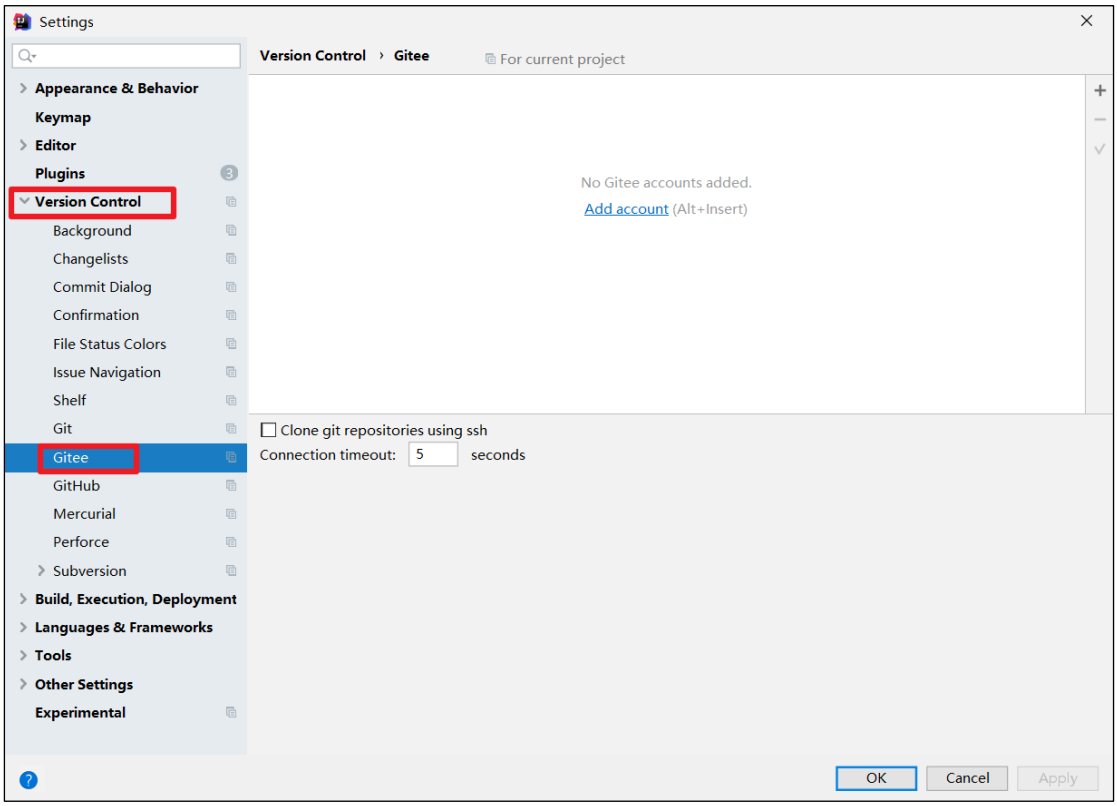
如图所示，在 Idea 插件商店搜索 Gitee，然后点击右侧的 Install 按钮。



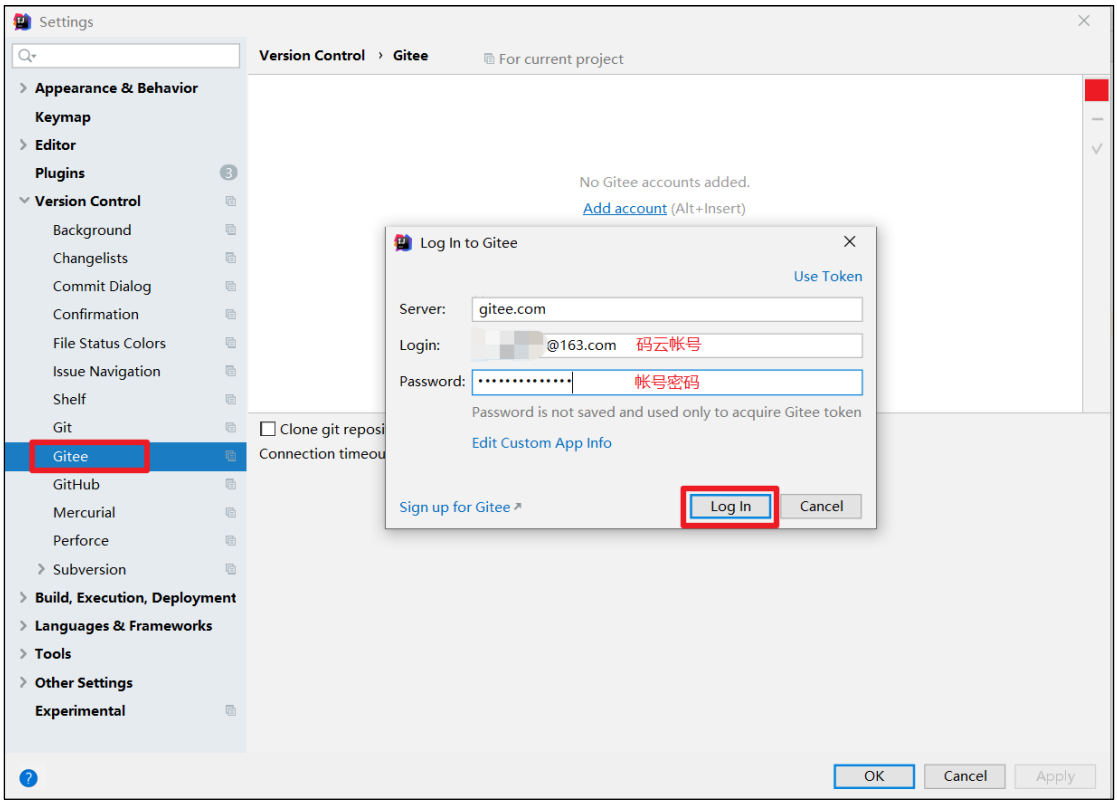
Idea 链接码云和链接 GitHub 几乎一样，安装成功后，重启 Idea。



Idea 重启以后在 Version Control 设置里面看到 Gitee，说明码云插件安装成功。



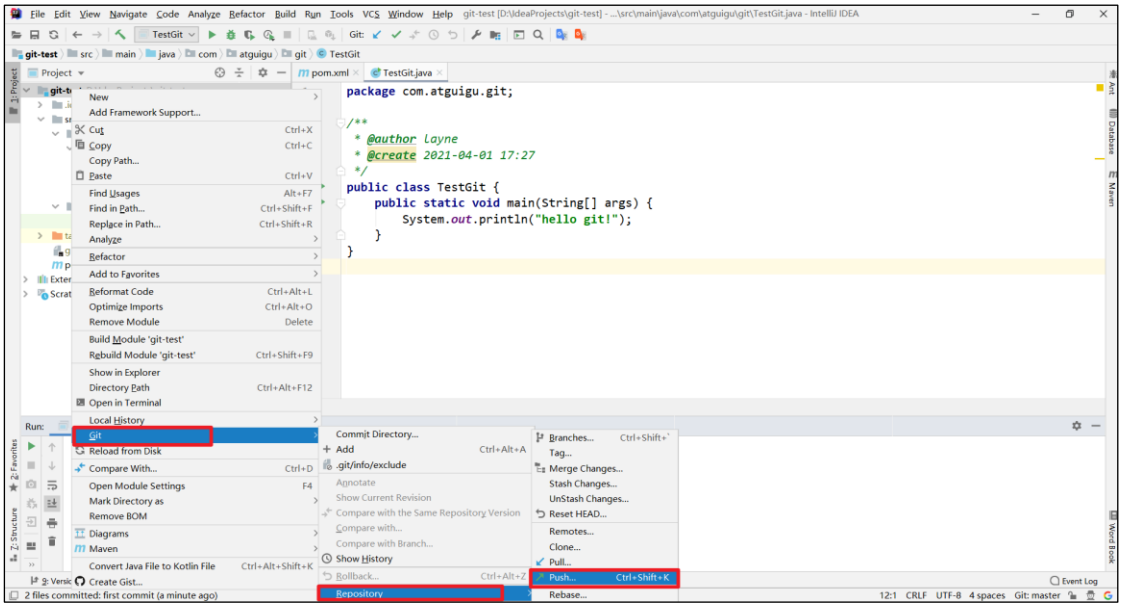
然后在码云插件里面添加码云帐号，我们就可以用 Idea 连接码云了。



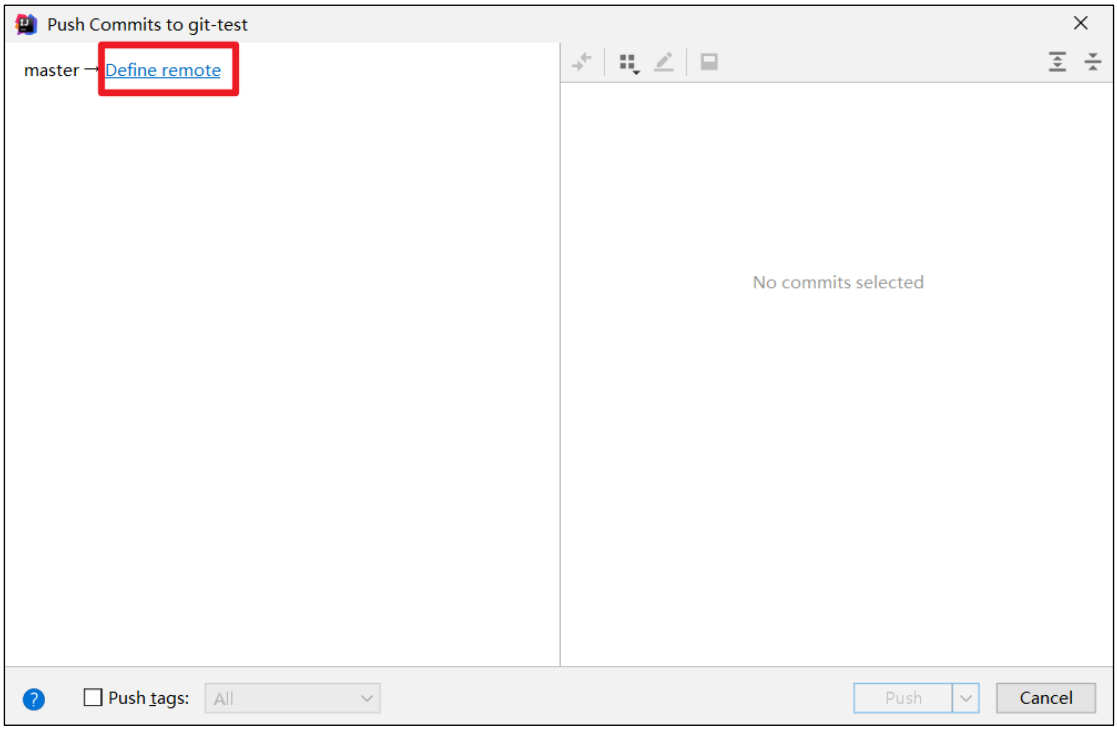
9.4.2 IDEA 连接码云

Idea 连接码云和连接 GitHub 几乎一样，首先在 Idea 里面创建一个工程，初始化 git 工程，然后将代码添加到暂存区，提交到本地库，这些步骤上面已经讲过，此处不再赘述。

➢ **将本地代码 push 到码云远程库**

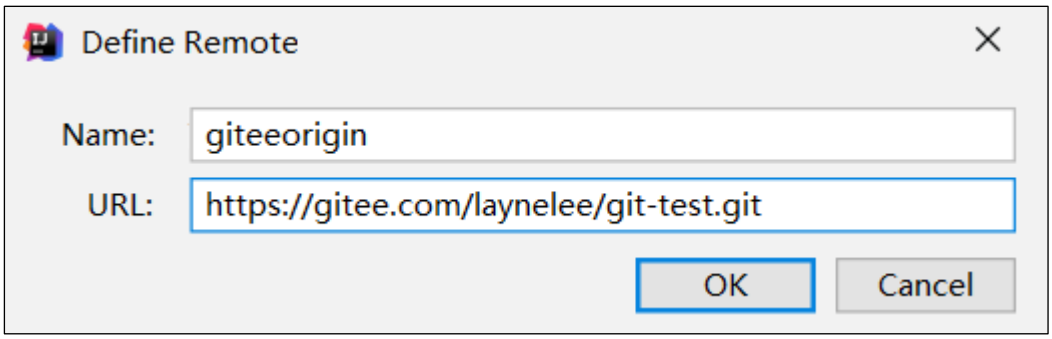


自定义远程库链接。

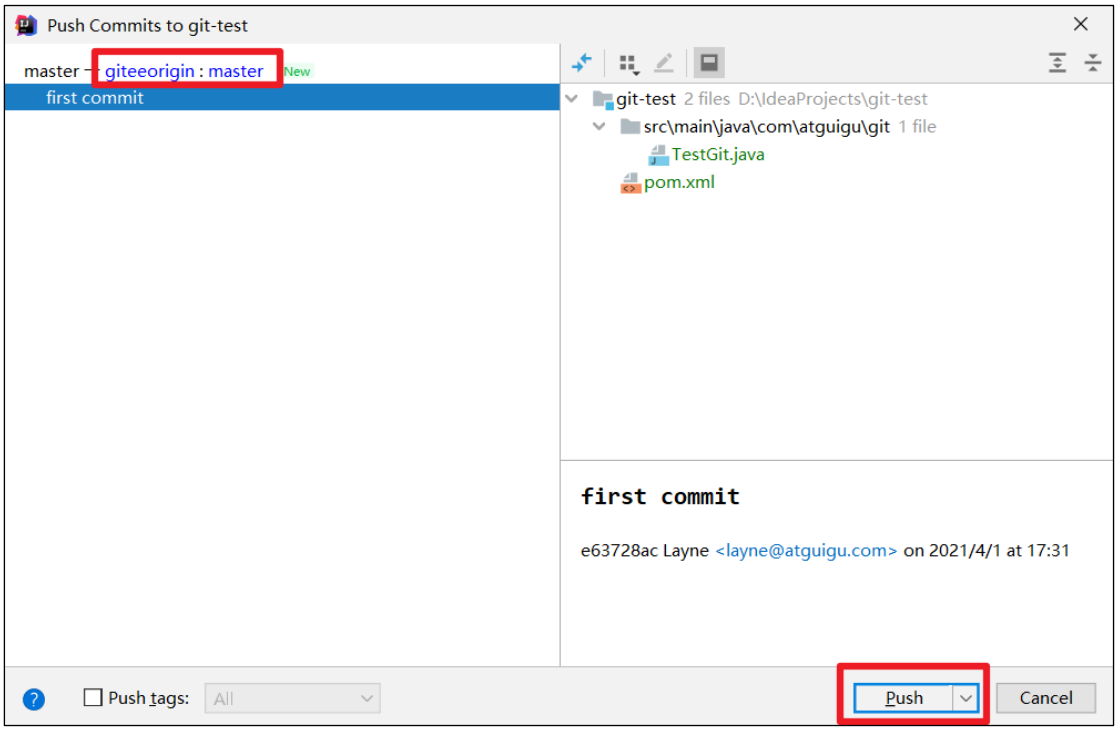


给远程库链接定义个 name，然后再 URL 里面填入码云远程库的 HTTPS 链接即可。码

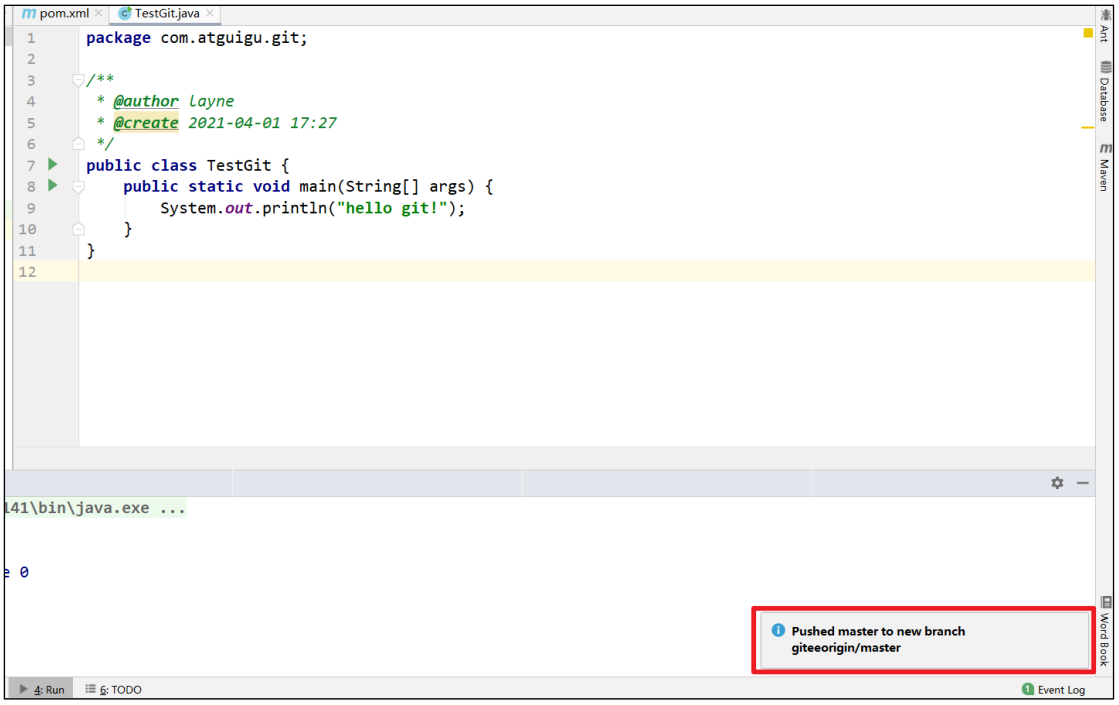
云服务器在国内，用 HTTPS 链接即可，没必要用 SSH 免密链接。



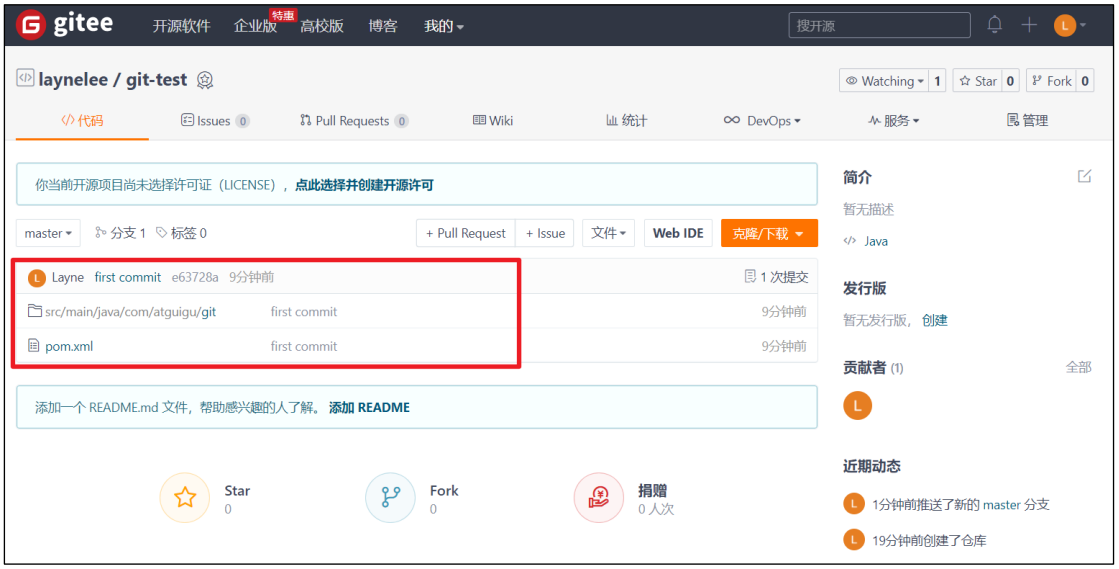
然后选择定义好的远程链接，点击 Push 即可。



看到提示就说明 Push 远程库成功。



去码云远程库查看代码。



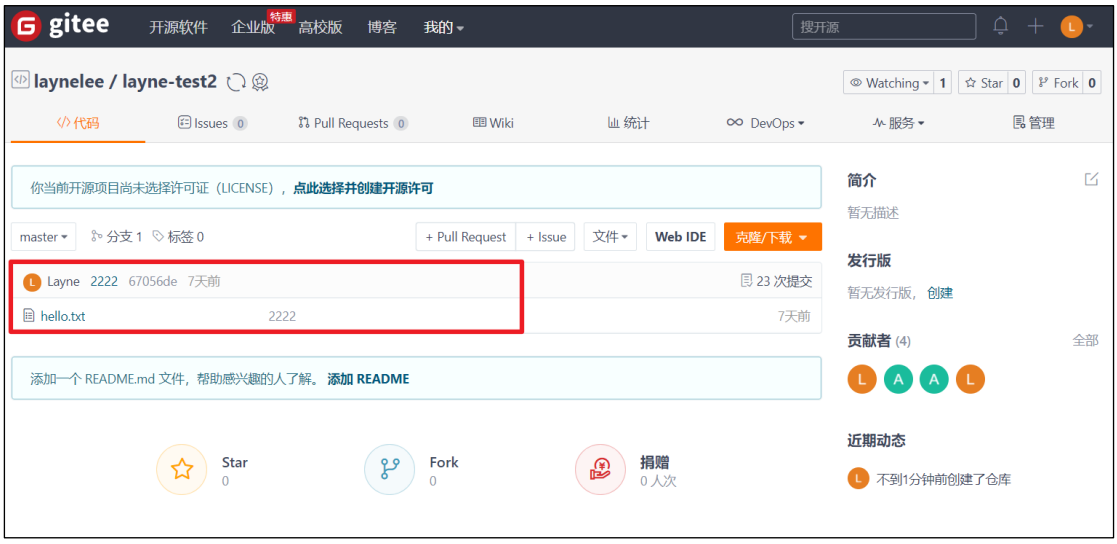
只要码云远程库链接定义好以后，对码云远程库进行 pull 和 clone 的操作和 Github 一致，此处不再赘述。

9.5 码云复制 GitHub 项目

码云提供了直接复制 GitHub 项目的功能，方便我们做项目的迁移和下载。

具体操作如下：

将 GitHub 的远程库 HTTPS 链接复制过来，点击创建按钮即可。



如果 GitHub 项目更新了以后，在码云项目端可以手动重新同步，进行更新！

第 10 章 自建代码托管平台-GitLab

10.1 GitLab 简介

GitLab 是由 GitLabInc.开发，使用 MIT 许可证的基于网络的 Git 仓库管理工具，且具有 wiki 和 issue 跟踪功能。使用 Git 作为代码管理工具，并在此基础上搭建起来的 web 服务。

GitLab 由乌克兰程序员 DmitriyZaporozhets 和 ValerySizov 开发，它使用 Ruby 语言写成。后来，一些部分用 Go 语言重写。截止 2018 年 5 月，该公司约有 290 名团队成员，以及 2000 多名开源贡献者。GitLab 被 IBM，Sony，JülichResearchCenter，NASA，Alibaba，Invincea，O'ReillyMedia，Leibniz-Rechenzentrum(LRZ)，CERN，SpaceX 等组织使用。

10.2 GitLab 官网地址

官网地址：https://about.gitlab.com/

安装说明：https://about.gitlab.com/installation/

10.3 GitLab 安装

10.3.1 服务器准备

准备一个系统为 CentOS7 以上版本的服务器，要求内存 4G，磁盘 50G。

关闭防火墙，并且配置好主机名和 IP，保证服务器可以上网。

此教程使用虚拟机：主机名：gitlab-server IP 地址：192.168.6.200

10.3.2 安装包准备

Yum 在线安装 gitlab- ce 时，需要下载几百 M 的安装文件，非常耗时，所以最好提前把所需 RPM 包下载到本地，然后使用离线 rpm 的方式安装。

下载地址：

```
https://packages.gitlab.com/gitlab/gitlab-
ce/packages/el/7/gitlab-ce-13.10.2-ce.0.el7.x86_64.rpm
```

注：资料里提供了此 rpm 包，直接将此包上传到服务器/opt/module 目录下即可。

10.3.3 编写安装脚本

安装 gitlab 步骤比较繁琐，因此我们可以参考官网编写 gitlab 的安装脚本。

```
[root@gitlab-server module]# vim gitlab-install.sh
sudo rpm -ivh /opt/module/gitlab-ce-13.10.2-ce.0.el7.x86_64.rpm

sudo yum install -y curl policycoreutils-python openssh-server cronie
```

```
sudo lokkit -s http -s ssh

sudo yum install -y postfix

sudo service postfix start

sudo chkconfig postfix on

curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash

sudo EXTERNAL_URL="http://gitlab.example.com" yum -y install gitlab-ce
```

给脚本增加执行权限

```
[root@gitlab-server module]# chmod +x gitlab-install.sh
[root@gitlab-server module]# ll
总用量 403104
-rw-r--r--. 1 root root 412774002 4 月   7 15:47 gitlab-ce-13.10.2-ce.0.el7.x86_64.rpm
-rwxr-xr-x. 1 root root       416 4 月   7 15:49 gitlab-install.sh
```

然后执行该脚本，开始安装 gitlab-ce。注意一定要保证服务器可以上网。

```
[root@gitlab-server module]# ./gitlab-install.sh
警告：/opt/module/gitlab-ce-13.10.2-ce.0.el7.x86_64.rpm: 头 V4 RSA/SHA1 Signature, 密钥 ID f27eab47: NOKEY
准备中...                          ################################# [100%]
正在升级/安装...
   1:gitlab-ce-13.10.2-ce.0.el7     ################################# [100%]
。 。 。 。 。 。
```

10.3.4 初始化 GitLab 服务

执行以下命令初始化 GitLab 服务，过程大概需要几分钟，耐心等待…

```
[root@gitlab-server module]# gitlab-ctl reconfigure


。 。 。 。 。 。
Running handlers:
Running handlers complete
Chef Client finished, 425/608 resources updated in 03 minutes 08 seconds
gitlab Reconfigured!
```

10.3.5 启动 GitLab 服务

执行以下命令启动 GitLab 服务，如需停止，执行 gitlab-ctl stop

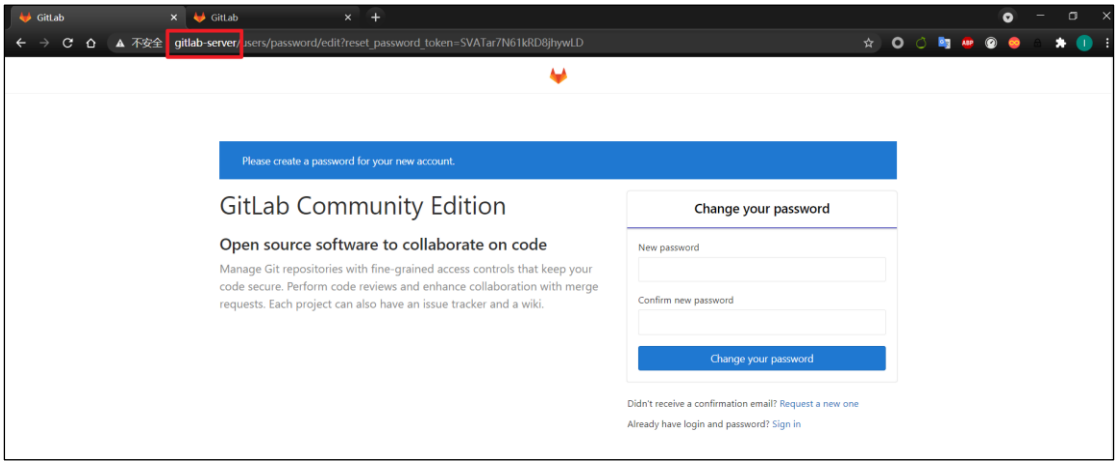
```
[root@gitlab-server module]# gitlab-ctl start
ok: run: alertmanager: (pid 6812) 134s
ok: run: gitaly: (pid 6740) 135s
ok: run: gitlab-monitor: (pid 6765) 135s
```

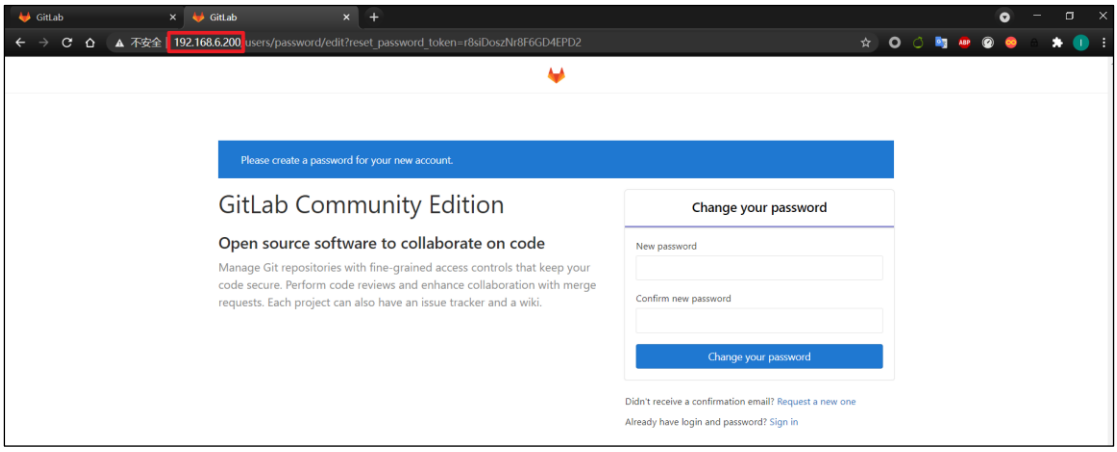
```
ok: run: gitlab-workhorse: (pid 6722) 136s
ok: run: logrotate: (pid 5994) 197s
ok: run: nginx: (pid 5930) 203s
ok: run: node-exporter: (pid 6234) 185s
ok: run: postgres-exporter: (pid 6834) 133s
ok: run: postgresql: (pid 5456) 257s
ok: run: prometheus: (pid 6777) 134s
ok: run: redis: (pid 5327) 263s
ok: run: redis-exporter: (pid 6391) 173s
ok: run: sidekiq: (pid 5797) 215s
ok: run: unicorn: (pid 5728) 221s
```

10.3.6 使用浏览器访问 GitLab

使用主机名或者 IP 地址即可访问 GitLab 服务。需要提前配一下 windows 的 hosts 文件。

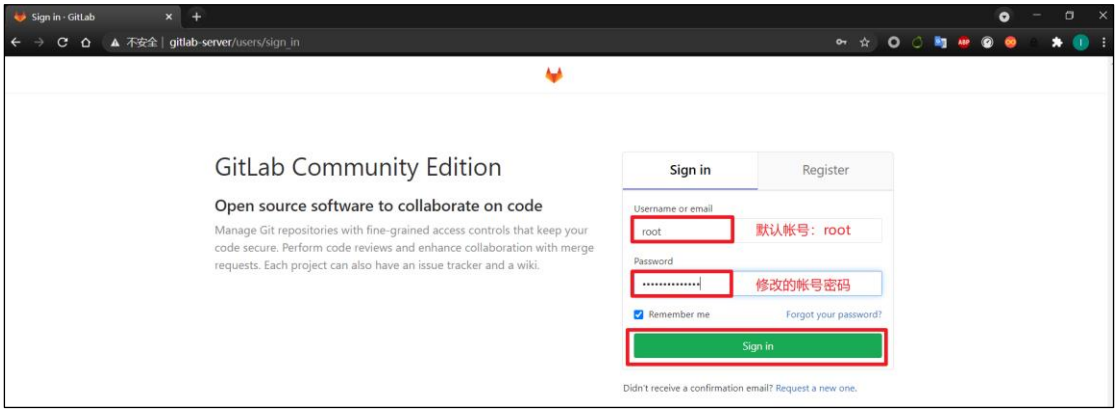
```
192.168.6.100   hadoop100
192.168.6.101   hadoop101
192.168.6.102   hadoop102
192.168.6.103   hadoop103
192.168.6.104   hadoop104
192.168.6.105   hadoop105
192.168.6.106   hadoop106
192.168.6.107   hadoop107
192.168.6.108   hadoop108
192.168.6.109   hadoop109
192.168.6.200   gitlab-server
```



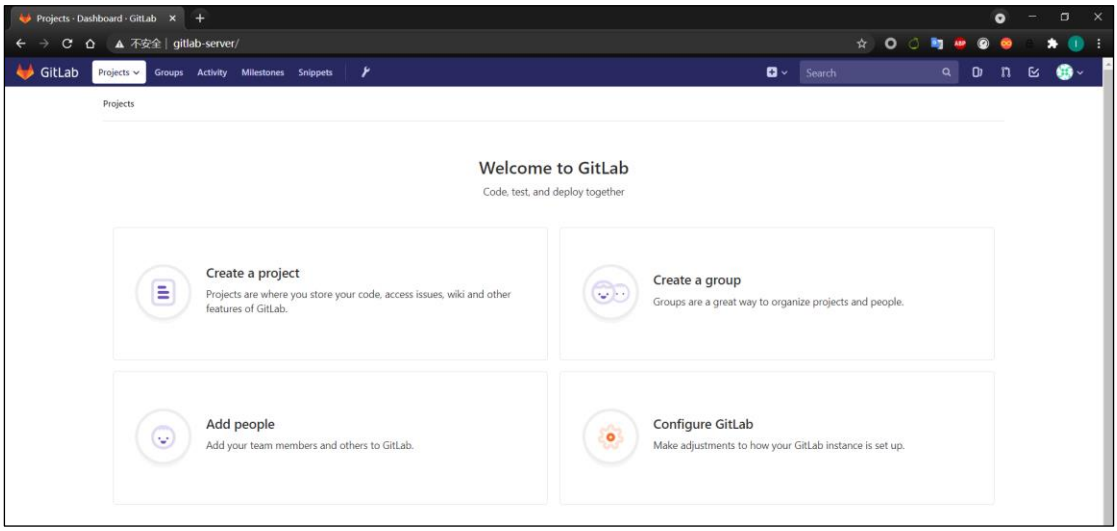


首次登陆之前，需要修改下 GitLab 提供的 root 账户的密码，要求 8 位以上，包含大小写子母和特殊符号。因此我们修改密码为 Atguigu.123456

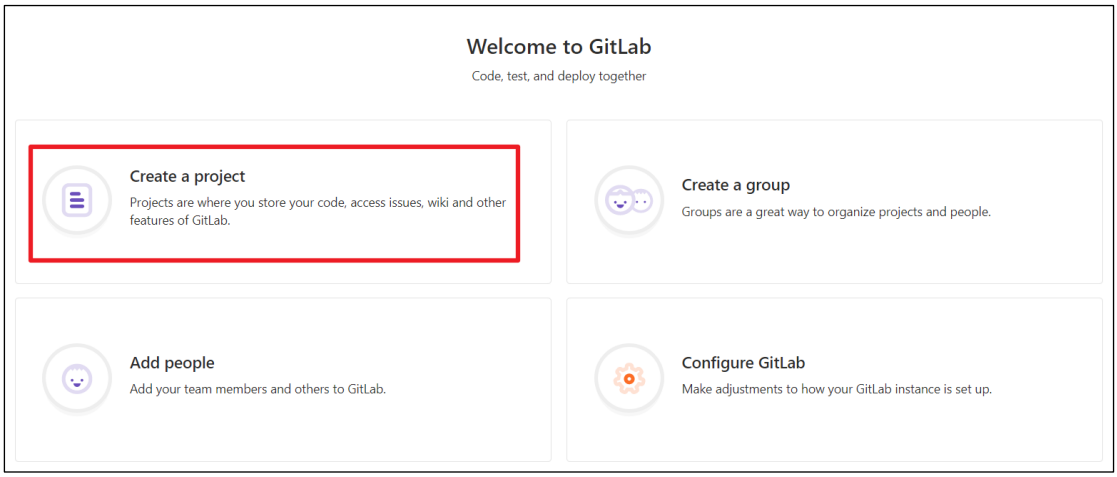
然后使用修改后的密码登录 GitLab。

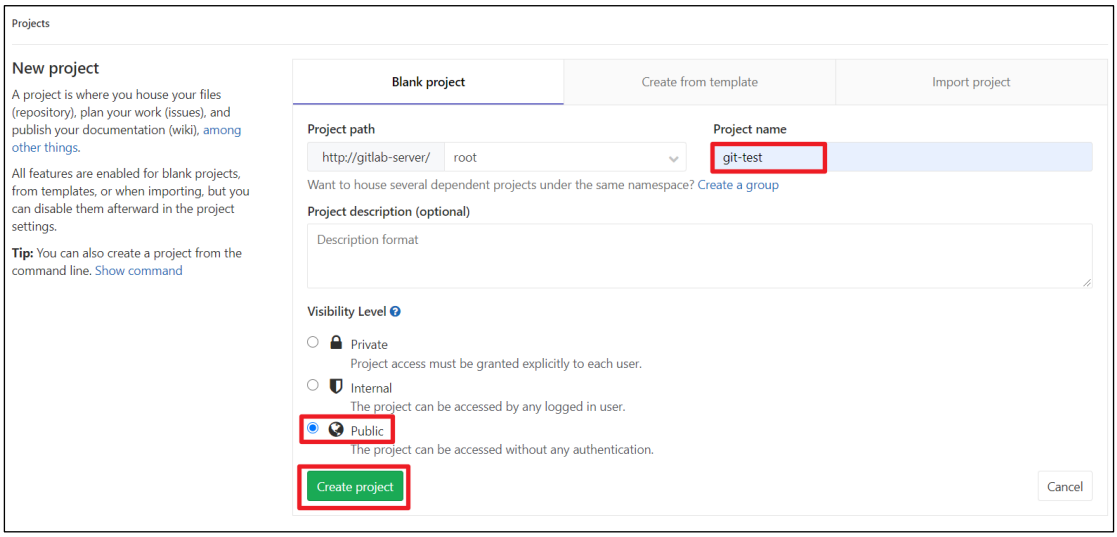


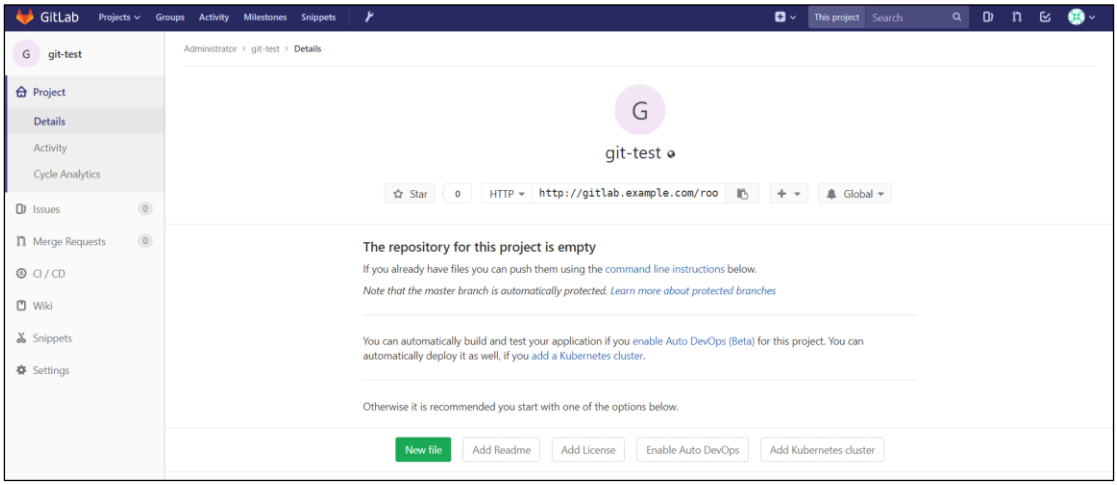
GitLab 登录成功。



10.3.7 GitLab 创建远程库

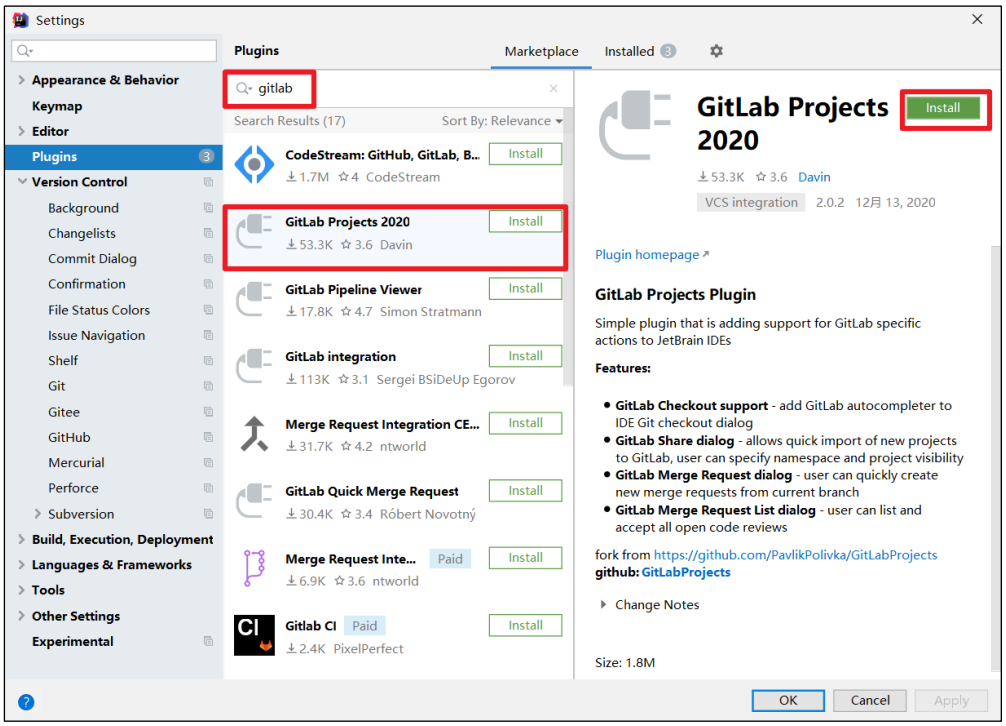




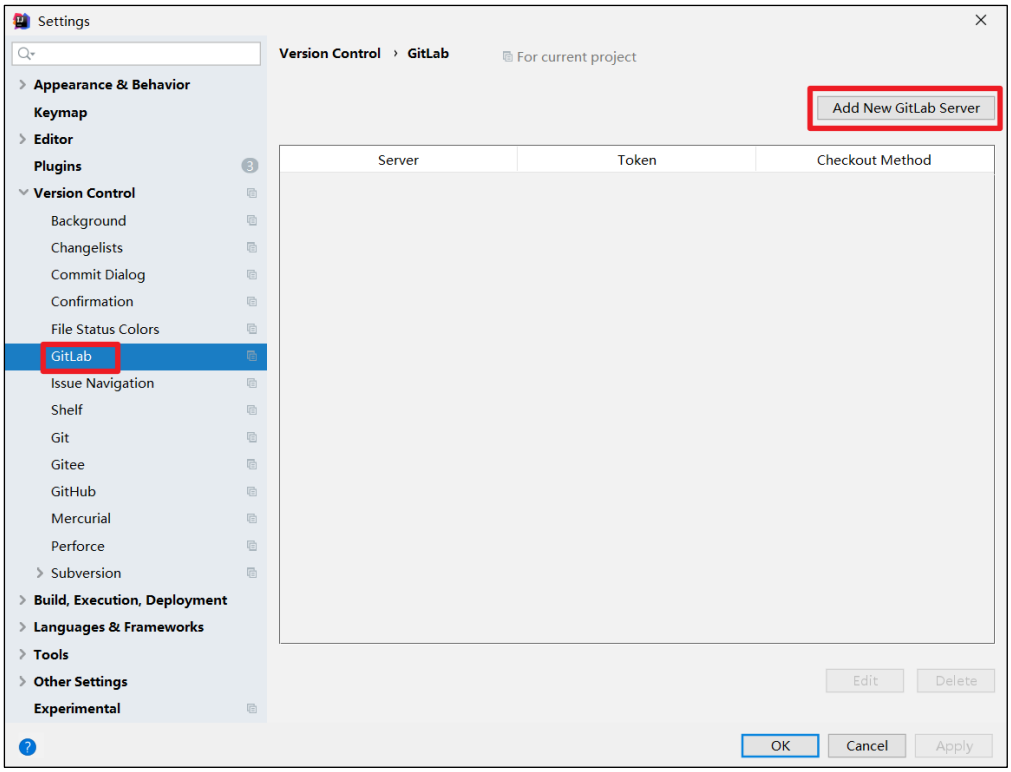


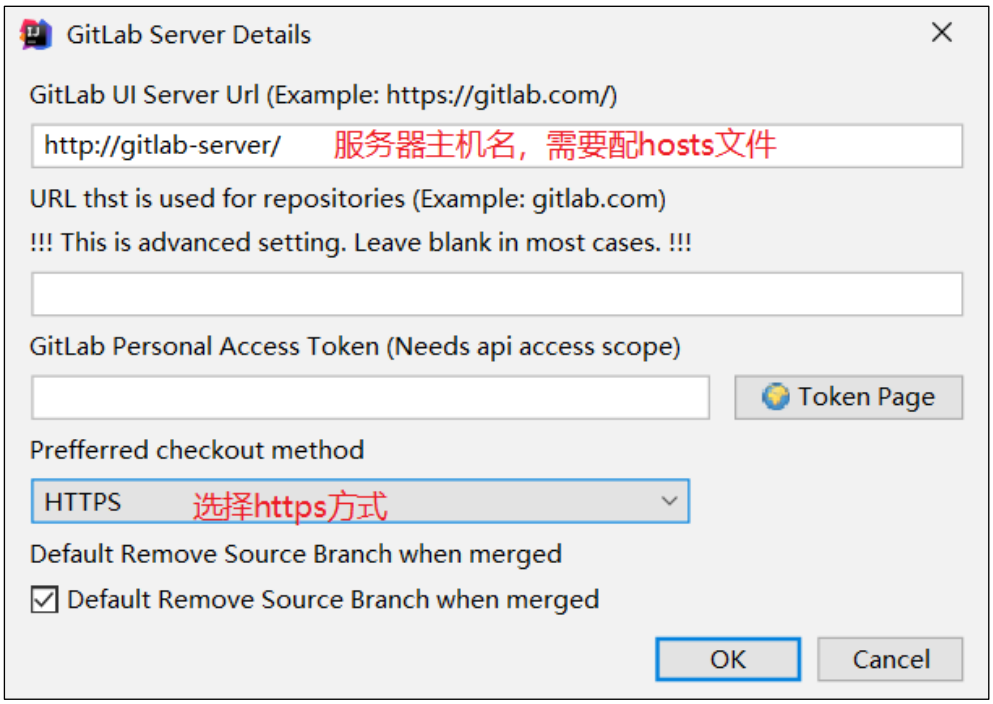
10.3.8 IDEA 集成 GitLab

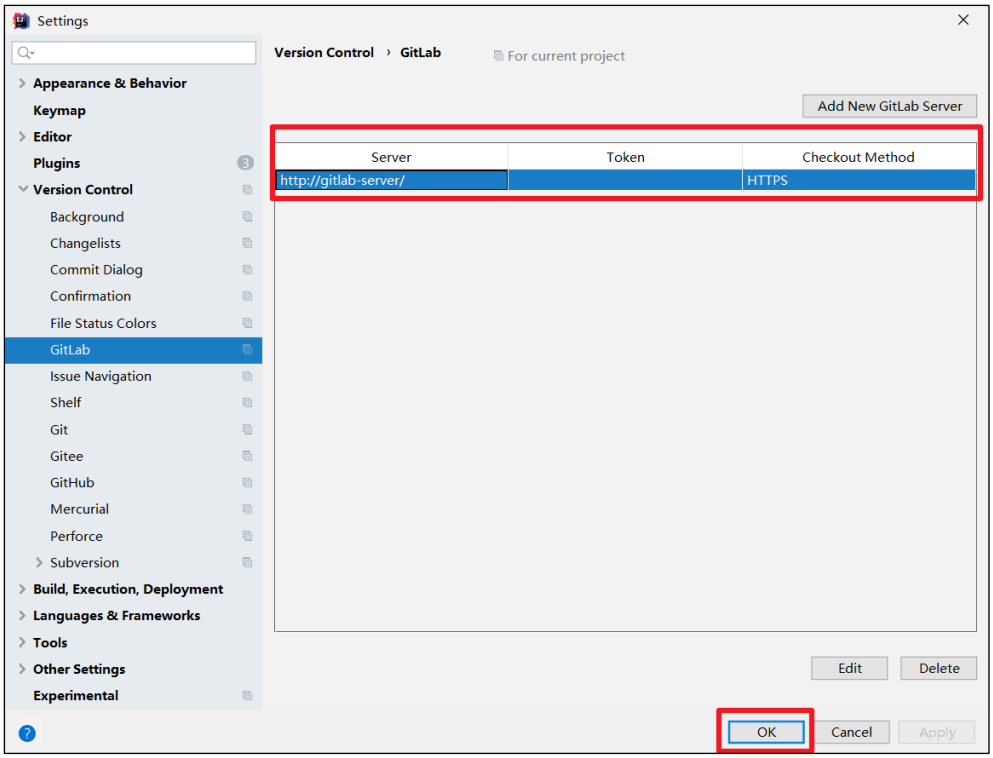
- 1）安装 GitLab 插件



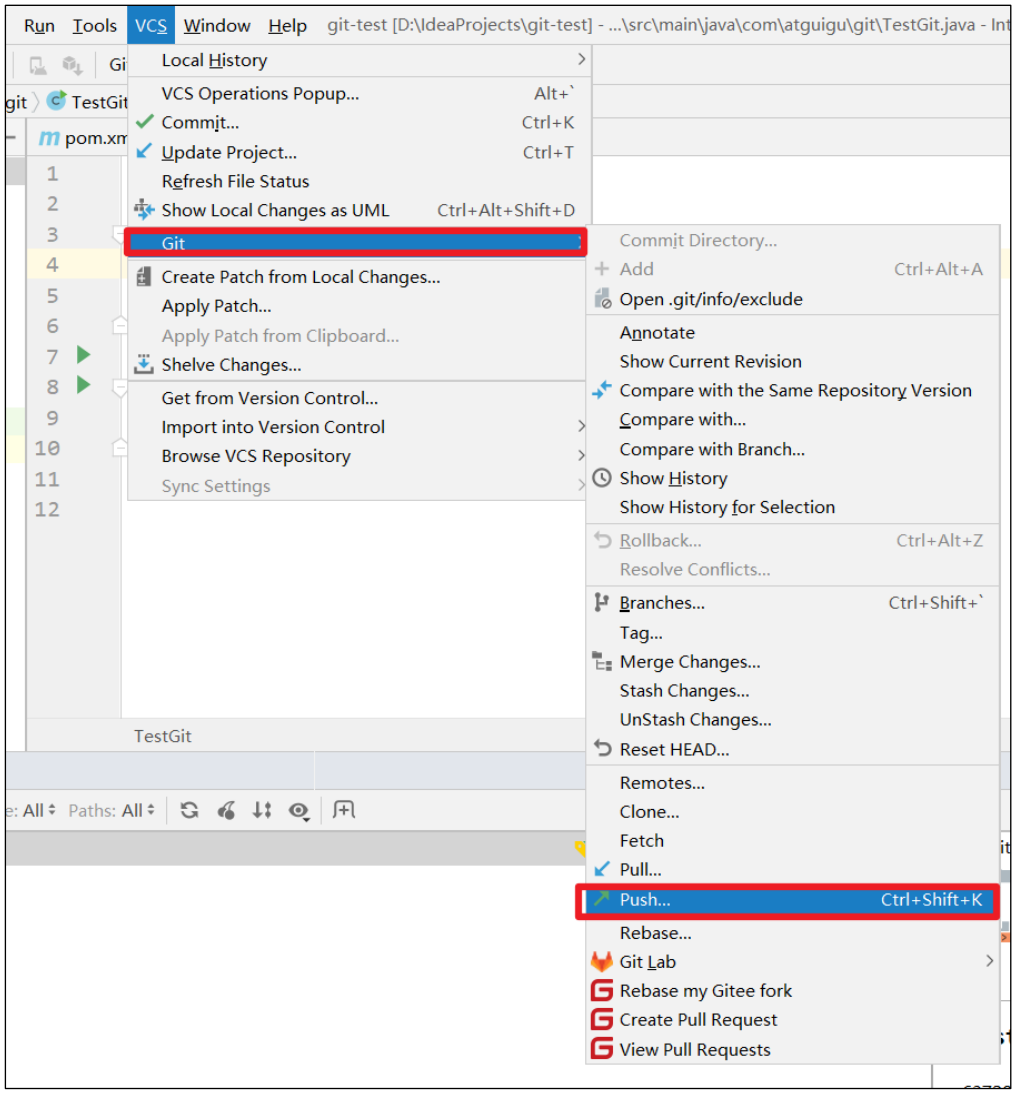
➢ 2）设置 GitLab 插件



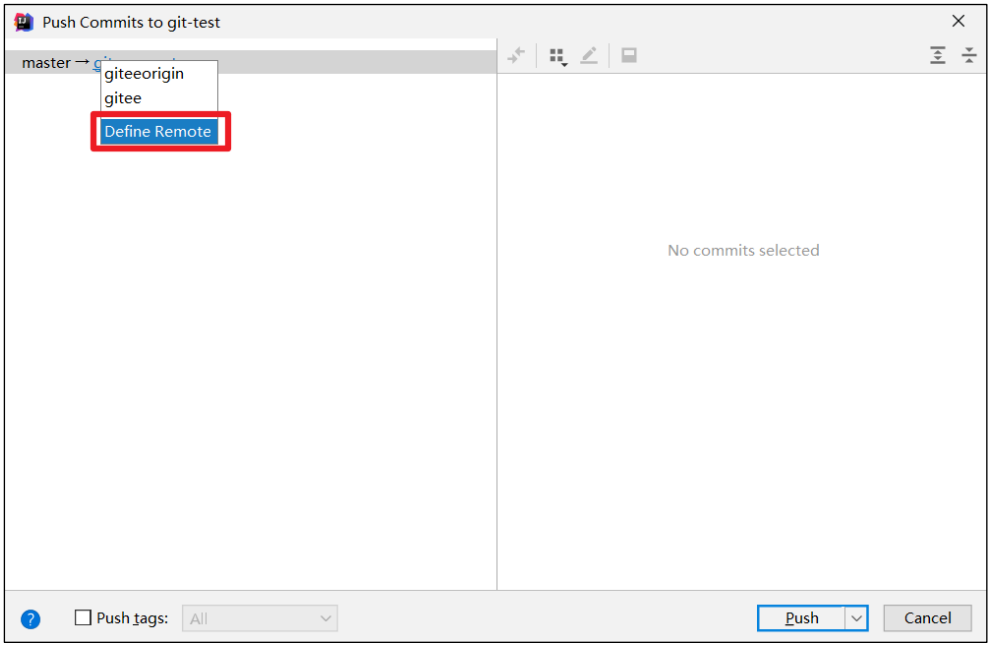


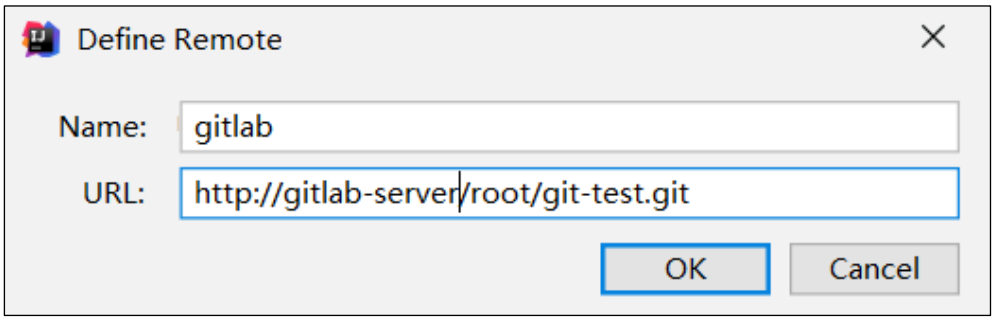


➢ 3）push 本地代码到 GitLab 远程库



自定义远程连接

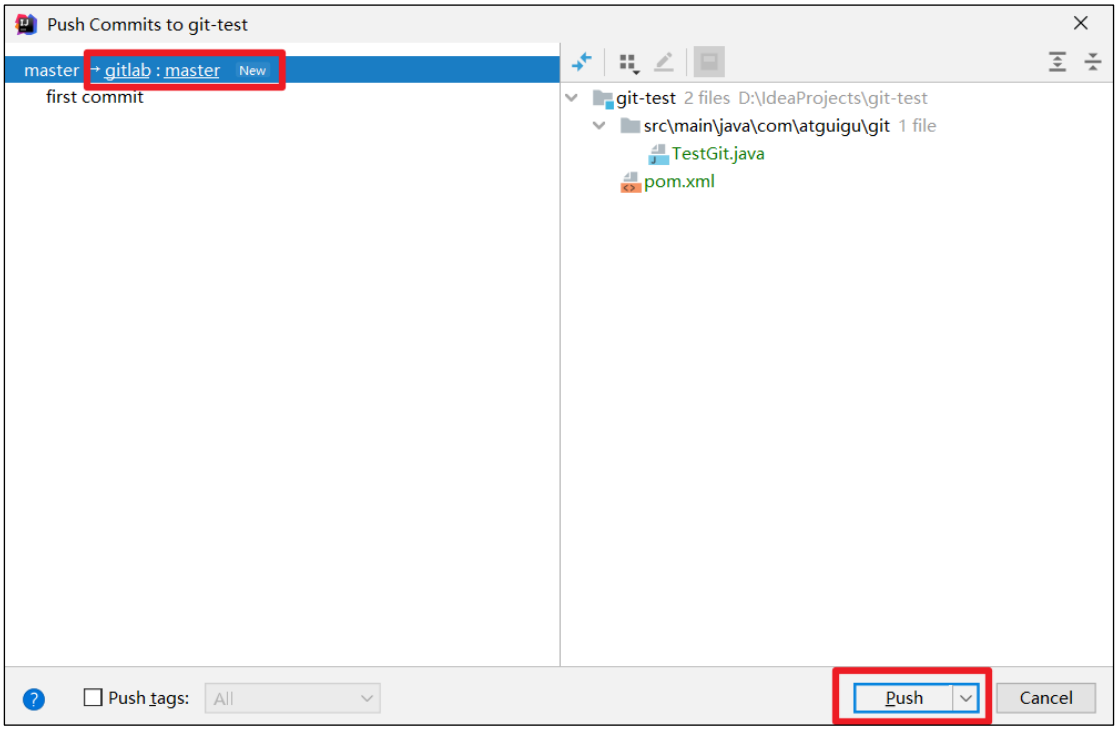




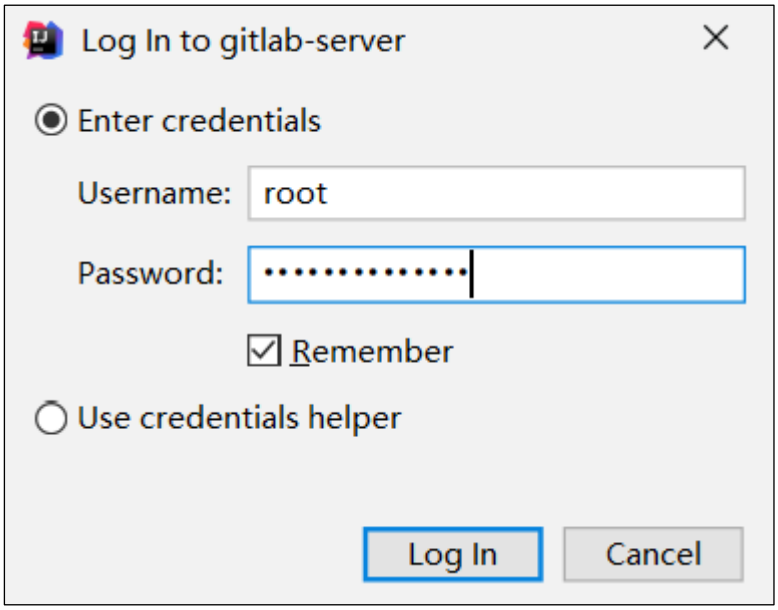
注意：gitlab 网页上复制过来的连接是：http://gitlab.example.com/root/git-test.git，

需要手动修改为：http://gitlab-server/root/git-test.git

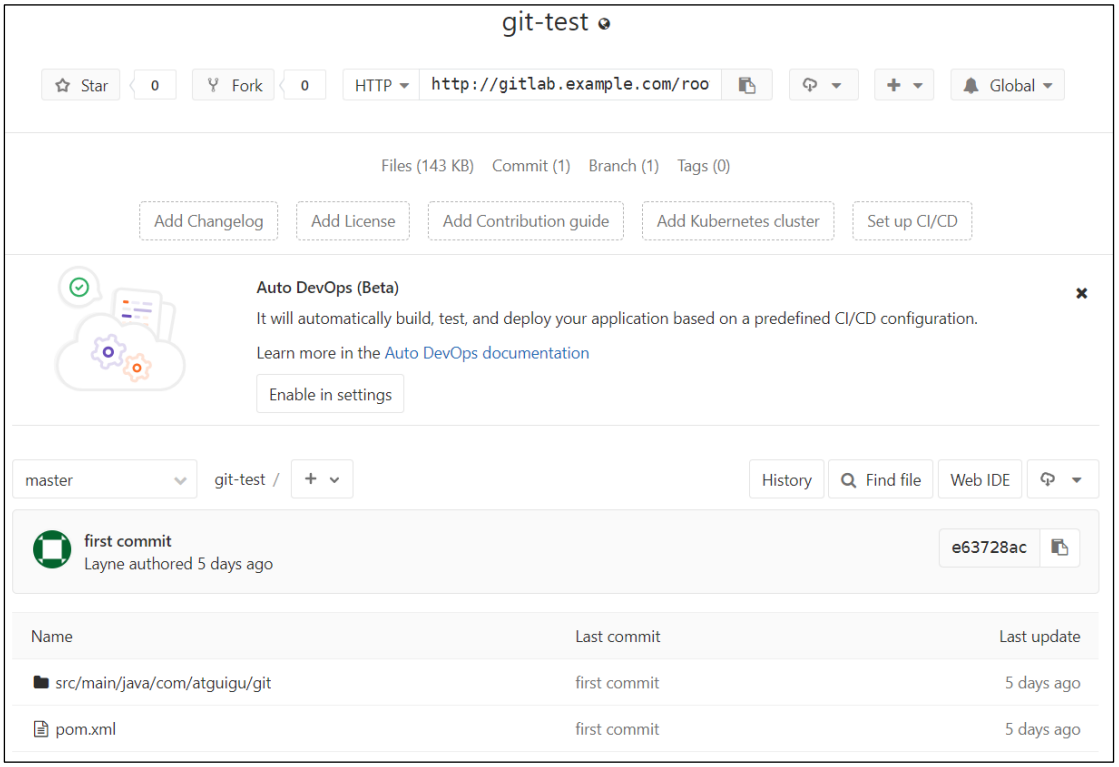
选择 gitlab 远程连接，进行 push。



首次向连接 gitlab，需要登录帐号和密码，用 root 帐号和我们修改的密码登录即可。



代码 Push 成功。



只要 GitLab 的远程库连接定义好以后，对 GitLab 远程库进行 pull 和 clone 的操作和 Github 和码云一致，此处不再赘述。