哈尔滨工业大学 计算学部

2024 年秋季学期《开源软件开发实践》

Lab4：开源软件开发中的 DevOps

学号	姓名	联系方式
2022211874	任铄同	18505461579

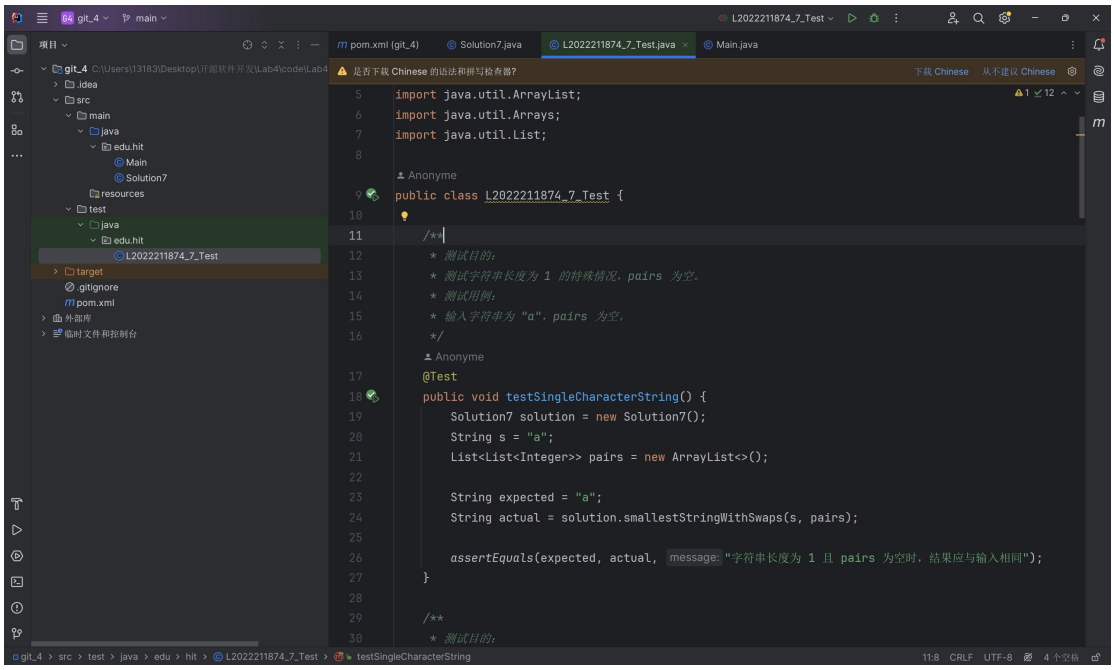
目 录

1 实验要求

本次实验训练开源软件开发中的基本 DevOps 操作，具体来说：
掌握开源软件开发中的基本 DevOps 流程和工具的使用。
熟悉利用 Github Actions 进行 DevOps。
熟悉利用 Jenkins 进行 DevOps。

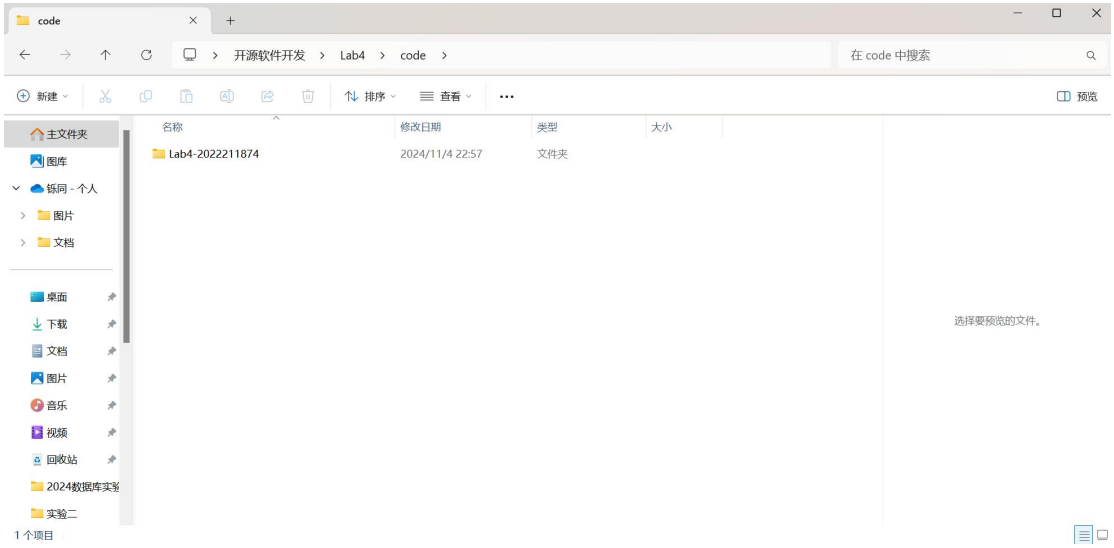
2 实验内容 1 Github Actions DevOps 实践

在本地新建一个项目：git_4，将实验二中修改正确的代码和测试类导入项目中，然后在 pom.xml 中导入 junit 依赖。



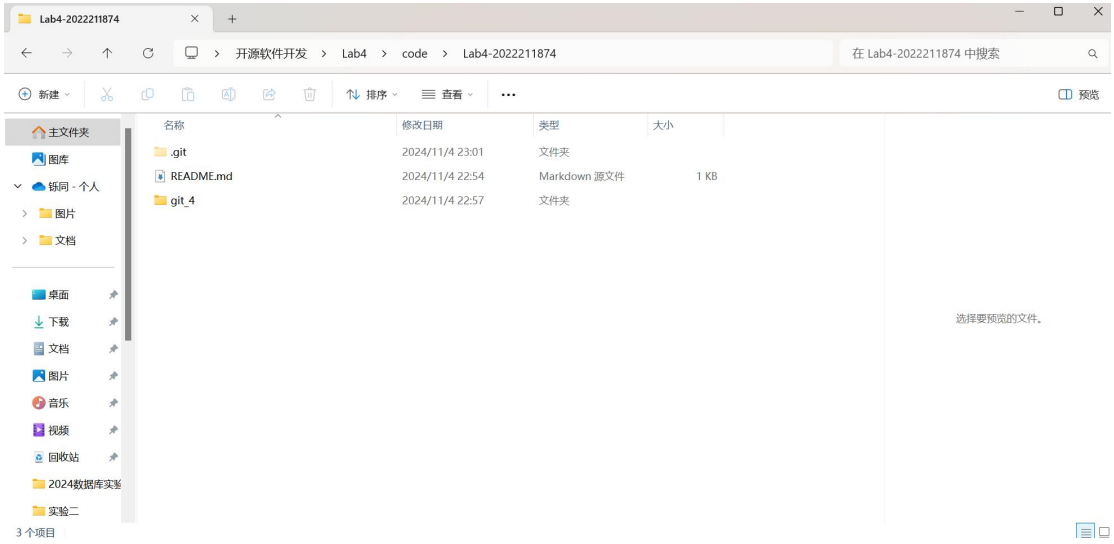

```
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.1</version>
            <scope>test</scope>
        </dependency>

    </dependencies>

</project>
```

新建 Lab4/code 文件夹，将实验仓库 clone 下来。



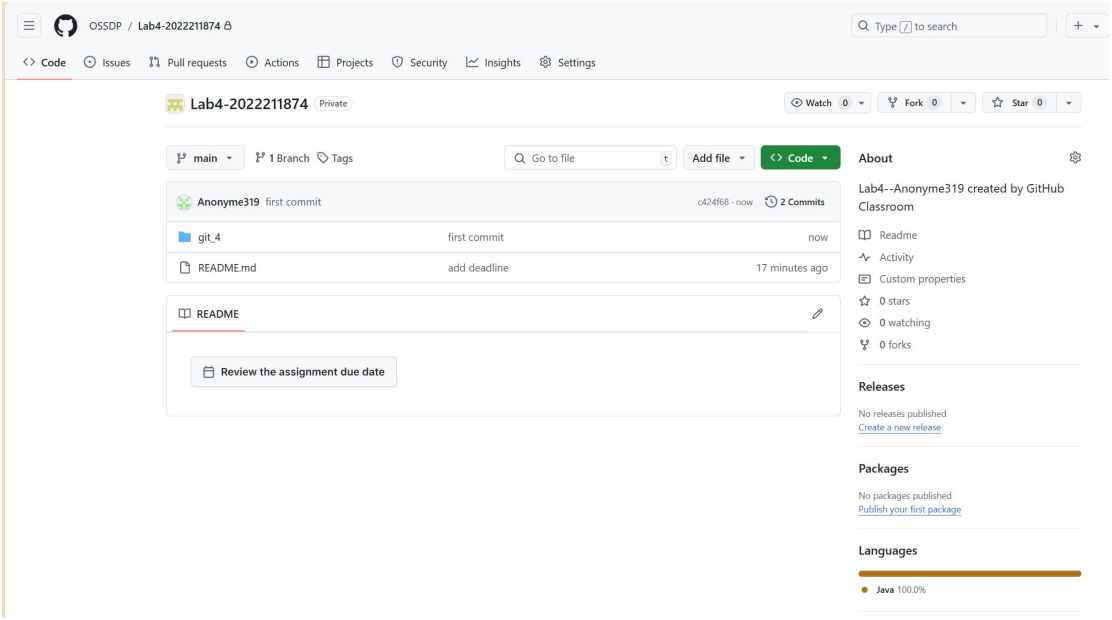

```
13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code (fix)
$ git clone https://github.com/OSSDP/Lab4-2022211874
Cloning into 'Lab4-2022211874'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

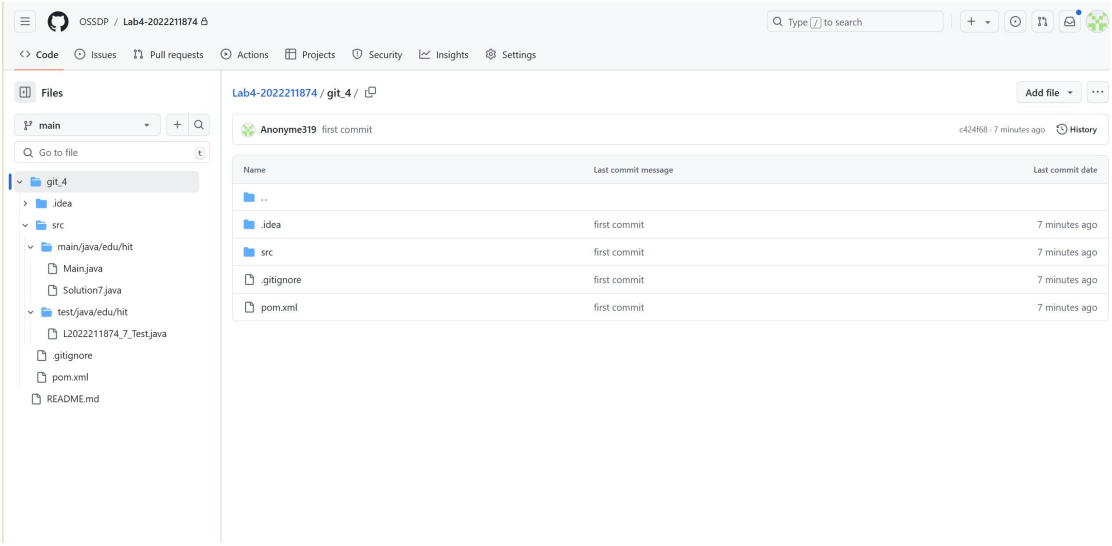
随后将刚刚新建的 maven 项目粘贴到文件夹中，先用 git add 和 git commit 提交到本地仓库，然后 push 到实验仓库中。



```
13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code (fix)
$ cd Lab4-2022211874

13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (main)
$ git add .
warning: in the working copy of 'git_4/.gitignore', LF will be replaced by CRLF
the next time Git touches it
warning: in the working copy of 'git_4/pom.xml', LF will be replaced by CRLF the
 next time Git touches it
warning: in the working copy of 'git_4/src/main/java/edu/hit/Main.java', LF will
 be replaced by CRLF the next time Git touches it

13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (main)
$ git commit -m "first commit"
[main c424f68] first commit
 9 files changed, 372 insertions(+)
 create mode 100644 git_4/.gitignore
 create mode 100644 git_4/.idea/.gitignore
 create mode 100644 git_4/.idea/encodings.xml
 create mode 100644 git_4/.idea/misc.xml
 create mode 100644 git_4/.idea/vcs.xml
 create mode 100644 git_4/pom.xml
 create mode 100644 git_4/src/main/java/edu/hit/Main.java
 create mode 100644 git_4/src/main/java/edu/hit/Solution7.java
 create mode 100644 git_4/src/test/java/edu/hit/L2022211874_7_Test.java

13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (main)
$ git push origin main
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 16 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (22/22), 4.78 KiB | 1.20 MiB/s, done.
Total 22 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/OSSDP/Lab4-2022211874
   e3e8921..c424f68  main -> main

13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (main)
$
```

可以看到实验仓库已经有了我刚刚在本地新建的 maven 项目。

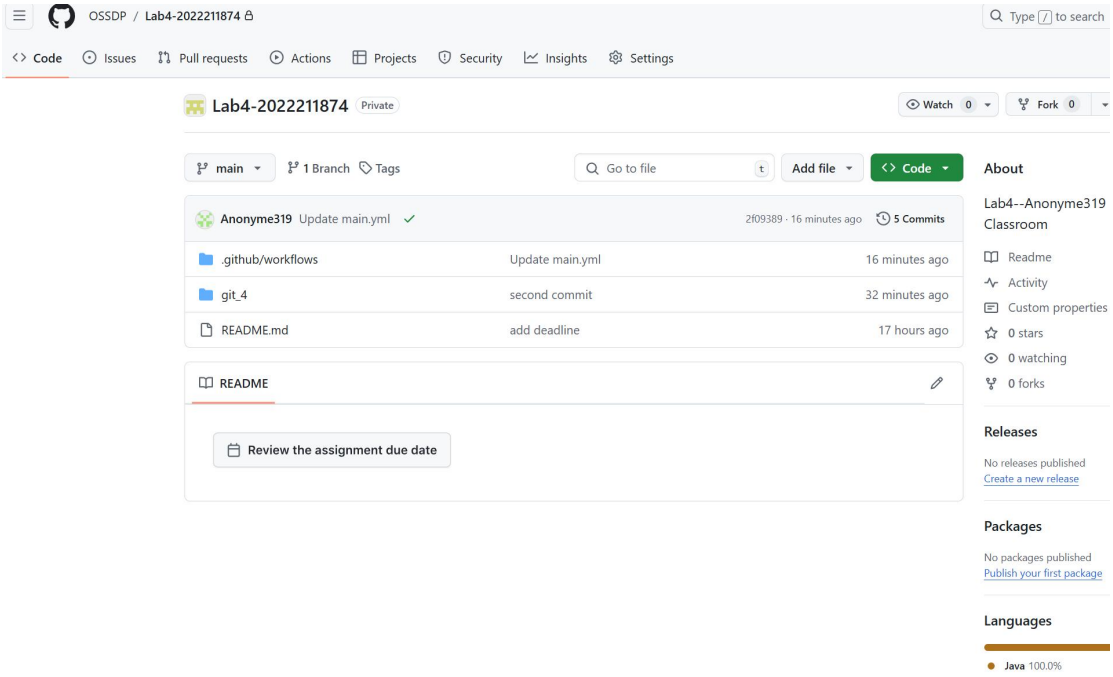




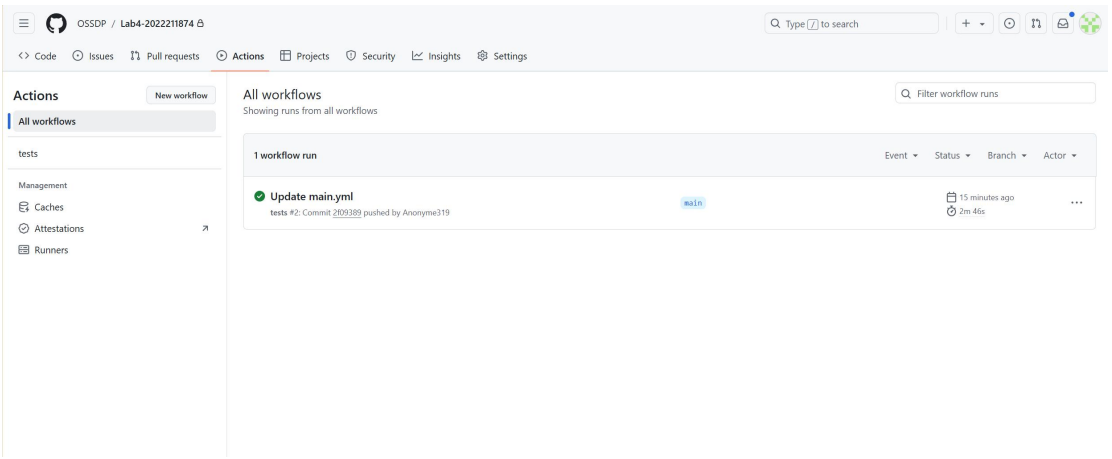
在 github 中的 Actions 下，点击新建 workflows，将实验手册中的代码粘贴进去，这里由于我的项目是以一个文件夹的形式放在仓库中的，所以需要改动一下最后一行：使用相对路径 git_4/pom.xml 来标识 pom 文件的位置。

```yaml
name: tests
on: push
jobs:
  run_tests:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout the repository
        uses: actions/checkout@v2
      - name: Set up JDK 17
        uses: actions/setup-java@v1
        with:
          java-version: 17
      - name: Cache Maven packages
        uses: actions/cache@v2
        with:
          path: ~/.m2
          key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
          restore-keys: ${{ runner.os }}-m2
      - name: Run tests with Maven
        run: mvn -B test --file git_4/pom.xml
```

保存后，可以在 github 中看到刚刚创建的 yml 文件（在.github/workflows 中）。



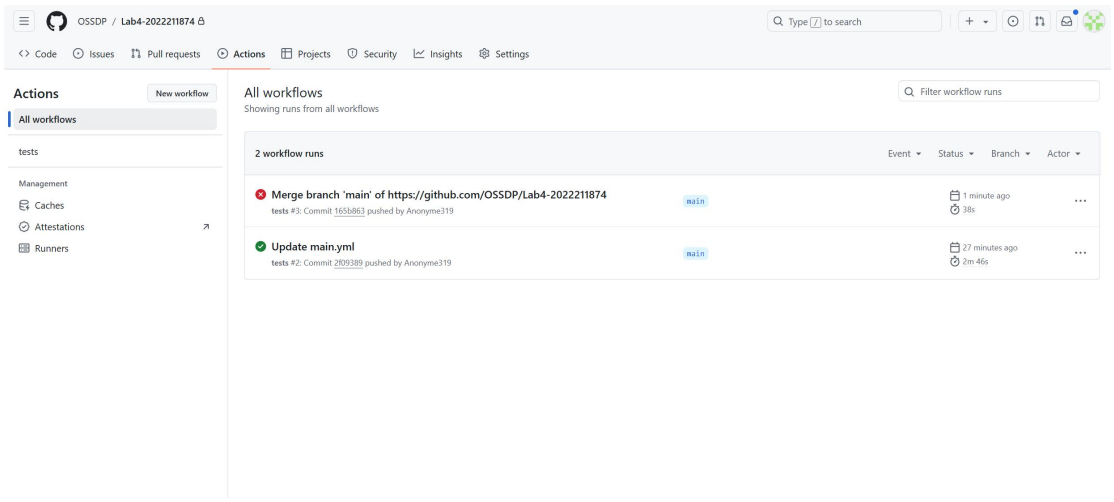
可以看到编写完工作流后，github 会根据刚刚编号的 yml 文件自动化测试，从下图中可以看出测试通过。



然后在本地对测试类进行一下修改，具体是 41 行，我把“abc”改为“abc_wrong”.

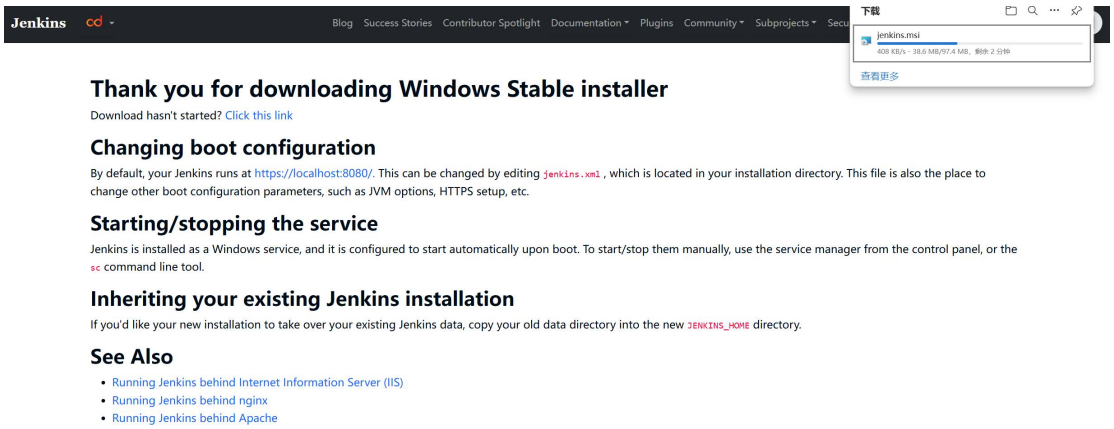
然后 git pull 拉取远程仓库的更改，然后使用 git add 和 git commit 将刚刚在本地的修改提交到本地仓库中，最后 push 上远程仓库。

在 push 到 github 上后，刚刚编写的.yml 文件对刚刚提交的文件进行了自动化测试，由于我刚刚故意改错了测试类，所以显然测试结果是错的。

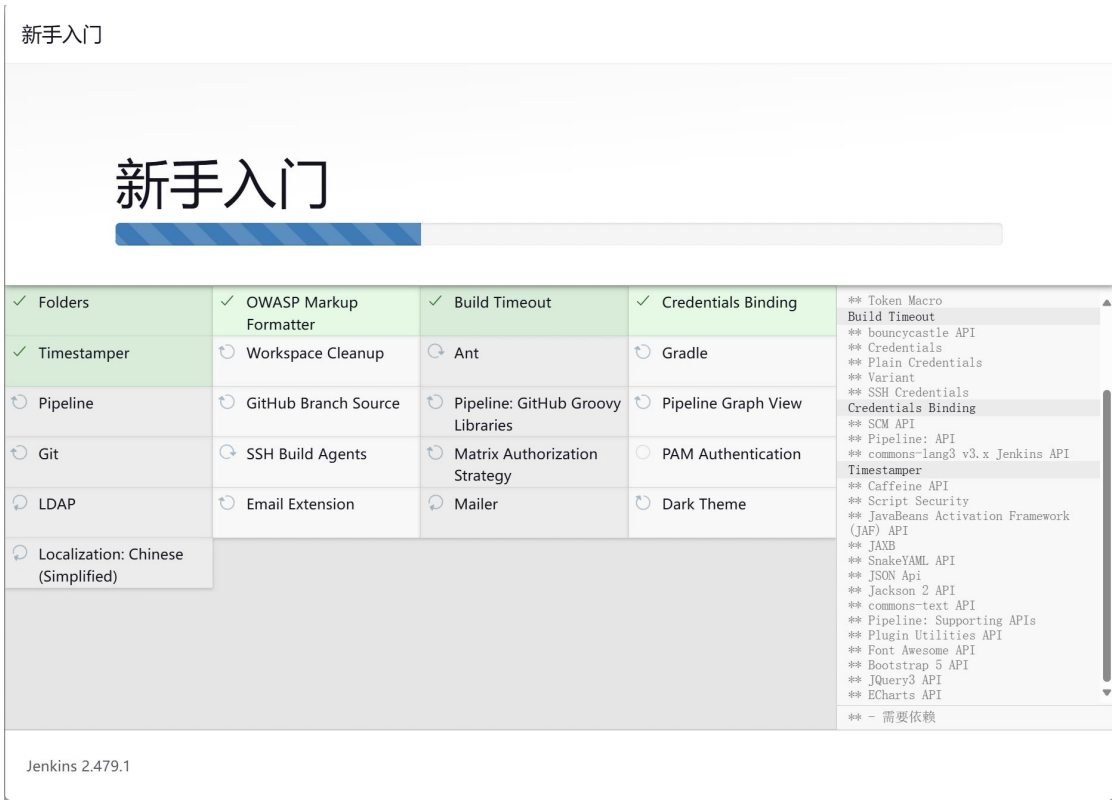


3 实验内容 2 Jenkins DevOps 实践

打开 Jenkins 官网，下载 window 版本。

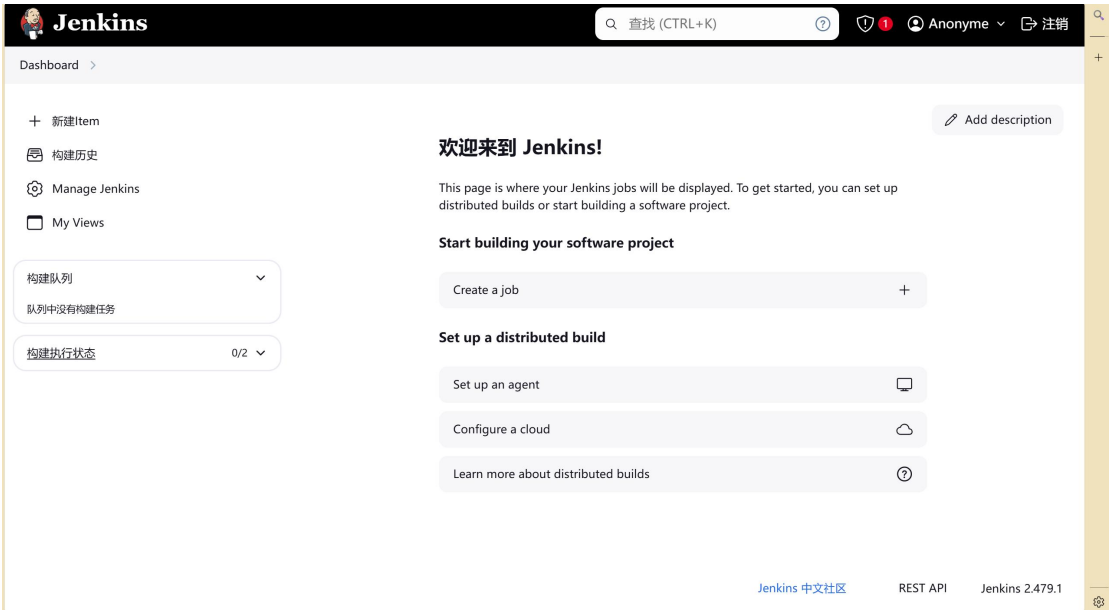


下载完毕后跟着实验手册进行安装以及完成初始化操作。

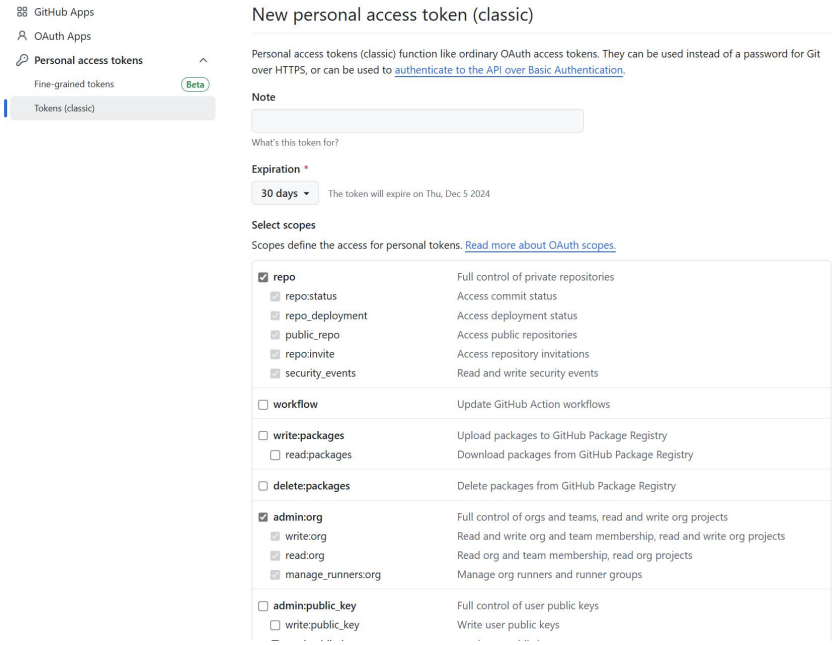

新手入门
新手入门
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline Graph View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Dark Theme
Localization: Chinese (Simplified)
** Token Macro
Build Timeout
** bouncycastle API
** Credentials
** Plain Credentials
** Variant
** SSH Credentials
Credentials Binding
** SCM API
** Pipeline: API
** commons-lang3 v3.x Jenkins API
Timestamper
** Caffeine API
** Script Security
** JavaBeans Activation Framework (JAF) API
** JAXB
** SnakeYAML API
** JSON Api
** Jackson 2 API
** commons-text API
** Pipeline: Supporting APIs
** Plugin Utilities API
** Font Awesome API
** Bootstrap 5 API
** JQuery3 API
** ECharts API
** - 需要依赖
Jenkins 2.479.1


新手入门
Jenkins已就绪!
Jenkins安装已完成。
开始使用Jenkins
Jenkins 2.479.1

创建账号并登录。



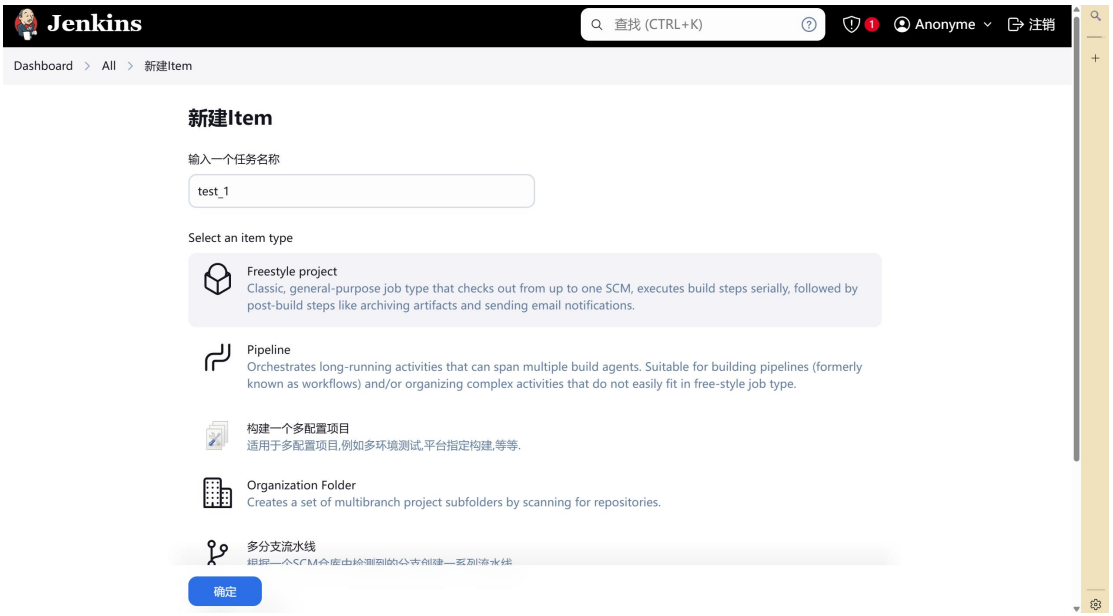
在 github 上申请 Access Token，并记下申请结果。

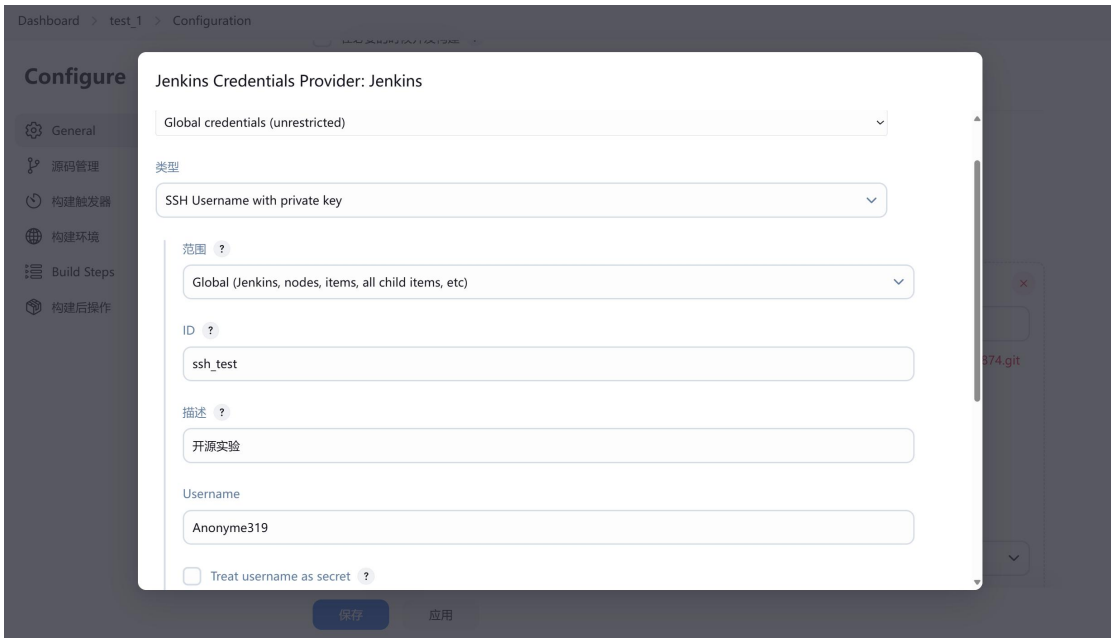


在本地新建 dev 分支，然后 push 到远程仓库，为后续实验做好准备。

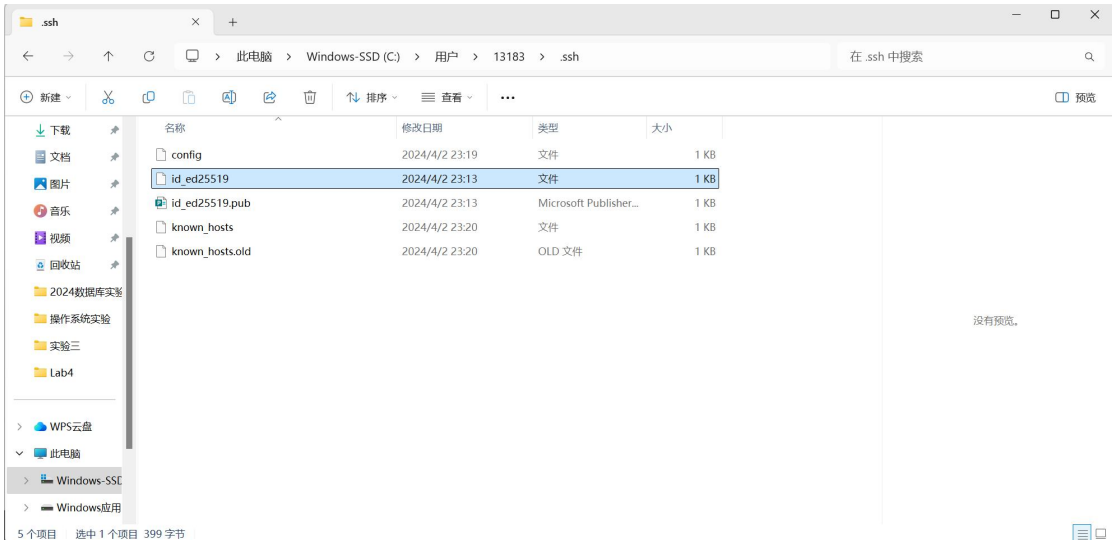
```
13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (main)
$ git pull origin main
fatal: unable to access 'https://github.com/OSSDP/Lab4-2022211874/': Failed to c
onnect to github.com port 443 after 21154 ms: Couldn't connect to server

13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (main)
$ git pull origin main
From https://github.com/OSSDP/Lab4-2022211874
 * branch            main       -> FETCH_HEAD
Already up to date.

13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (main)
$ git checkout -b dev
Switched to a new branch 'dev'

13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (dev)
$ git push origin dev
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:      https://github.com/OSSDP/Lab4-2022211874/pull/new/dev
remote:
To https://github.com/OSSDP/Lab4-2022211874
 * [new branch]      dev -> dev

13183@Anonyme MINGW64 ~/Desktop/开源软件开发/Lab4/code/Lab4-2022211874 (dev)
$
```

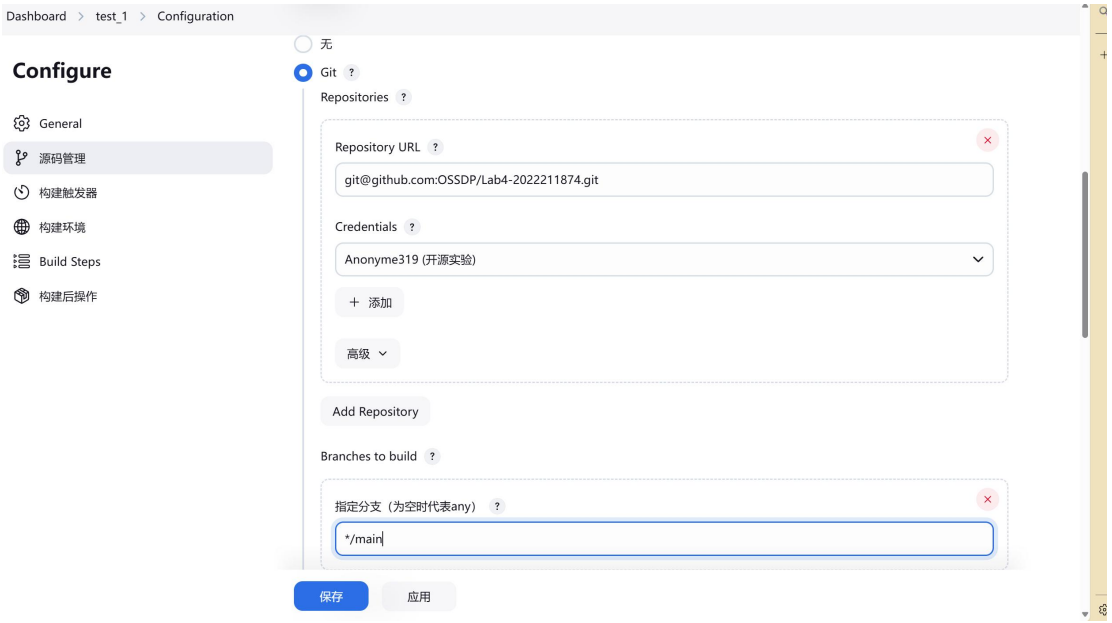
构建 DevOps workflow，新建 Item 以创建新的构建流程，随后按照实验手册进行配置。

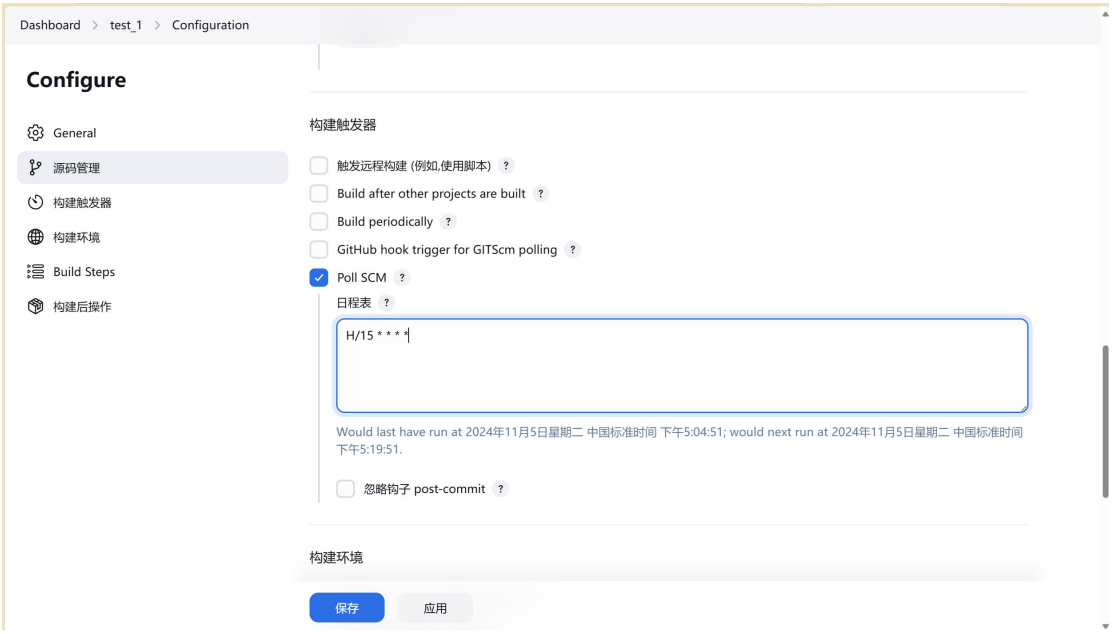




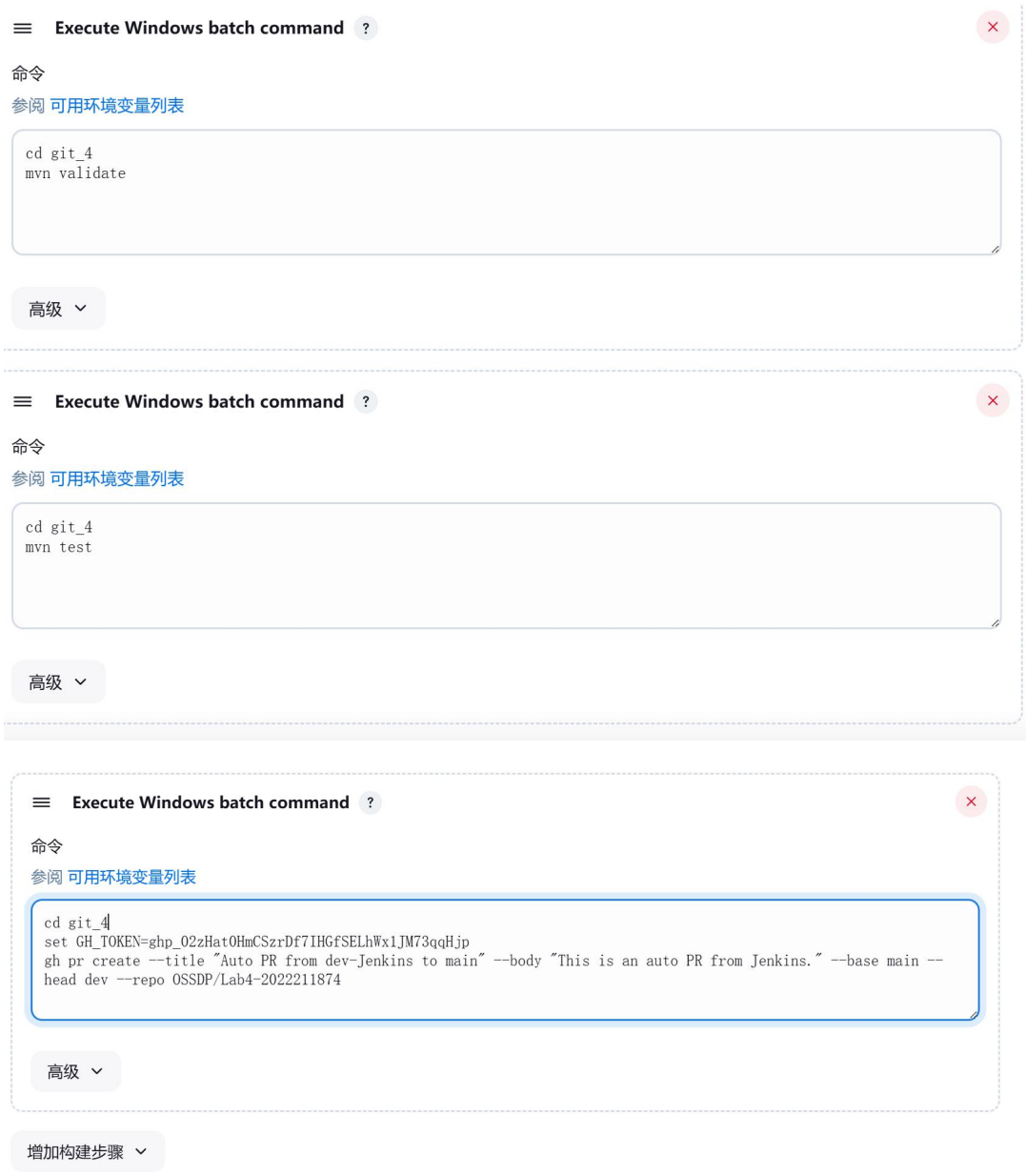
在.ssh 文件夹中找到 ssh 密钥。





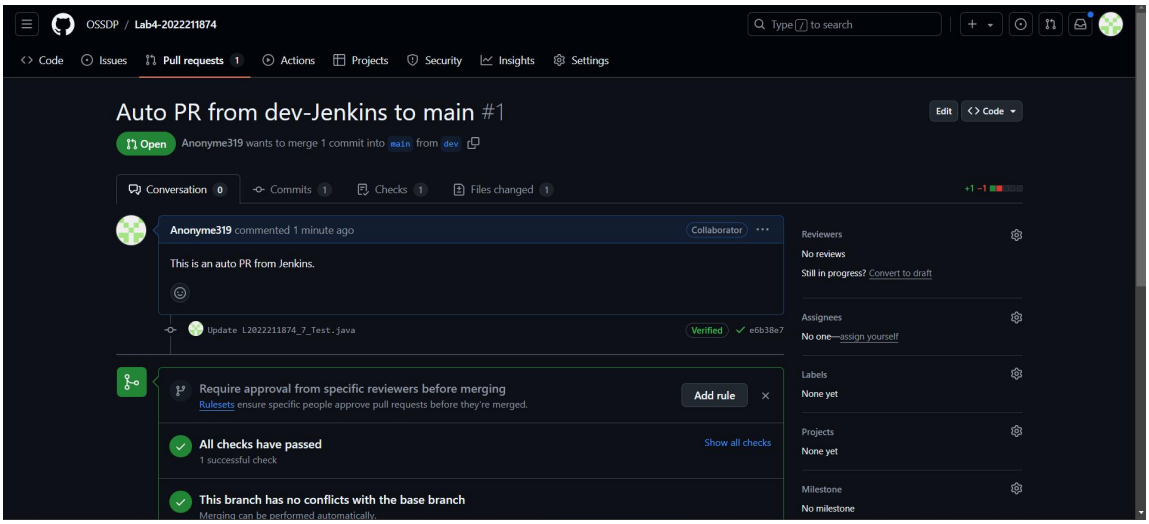


这里由于我的 pom 文件不在根目录，而是在 git_4 文件夹中，于是在命令前要添一行 cd git_4。



配置完毕，进行连接，可以看到连接成功。（由于我的电脑在连接这一步骤上出现了很奇怪的问题，不管怎么连都连不上，尝试了各种解决方式都没有效果，询问了老师、同学也无法解决问题，于是我怀疑是在我在下载 Jenkins 或是之前的某个步骤中因大意点错了某个键而导致不行，所以我向同学借了一个电脑，重新认真做了一遍这个实验，最后终于连接成功，下面是在另一台电脑上完成后续实验后的截图）

打开 github，发现 Jenkins 自动实现了 PR 提交。



4 小结

在本次实验中，我成功地使用 GitHub Actions 和 Jenkins 实现了项目的自动化构建和测试。GitHub Actions 部分，我不仅完成了基本的 CI/CD 流程配置，还尝试了修改测试用例以验证错误提交的情况。Jenkins 部分，我学会了如何配置 Jenkins 以实现自动化构建，并且成功地使用了 GitHub CLI 提交了 PR。

通过实验，我对很多概念和工具有了更深入的理解。DevOps 强调开发和运维之间的紧密合作，通过自动化工具提高软件交付的速度和质量。GitHub Actions 和 Jenkins 都是优秀的 CI/CD 工具，各有特点，GitHub Actions 更加轻量级，集成度高，适合小型项目和个人开发者；而 Jenkins 功能更为强大，配置灵活。自动化构建和测试极大地提高了开发效率，减少了人为错误，使得团队可以更快地响应需求变化，同时，自动化测试也保证了代码的质量，降低了生产环境中的风险。

实验过程中，我遇到了几个挑战，但最终都成功解决，并完成了实验目标。首先就是 GitHub Actions Workflow 配置问题，在 GitHub Actions 下新建 workflows 时，我没有关注我对实际仓库结构，导致错误提示找不到 pom 文件，通过网上搜索和排查，我发现需要使用相对路径 git_4/pom.xml 来正确标识 pom 文件的位置，在这个调整后，workflow 能够正确识

别并执行测试。再就是 Jenkins DevOps Workflow 连接问题，在构建 DevOps workflow 时，我遇到了连接不上 GitHub 仓库的问题。最初，我怀疑是私有仓库权限问题，因此将其改为公有，但问题依旧，根据实验手册的提示，我尝试关闭了 Host 检查，更换公钥私钥，甚至新建了 public 仓库，但都未能解决问题。我怀疑是在我下载 Jenkins 因大意选错了某个选项，于是我更换了一台计算机重新操作了一遍，最终成功连接。连接后，又遇到了找不到 pom 文件和测试类不通过的问题。我意识到 pom 文件不在根目录中，且测试类在之前的实验中被我故意改错，在将这些问题进行了对应的修正后，workflow 终于能够成功执行。

本次实验让我对自动化测试和部署有了深入的认识与理解，以及它们是如何提高软件开发的效率和质量的。还掌握了 GitHub Actions 和 Jenkins 这两个重要的 DevOps 工具的使用，了解了它们在自动化测试和持续集成中的作用。同时本次实验还极大地锻炼了我的问题解决能力。在遇到问题时，我学会了如何通过查阅资料、调整配置和代码来解决问题，在解决问题的过程中也让我对项目结构、配置文件、Github 的 ssh 连接等知识点有了更深入的理解。

总的来说，本次实验不仅让我掌握了 DevOps 的基本概念和工具，还提高了我的问题解决能力和对软件开发流程的理解，令我受益匪浅。