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
2023120240	王雅雯	17838829131@163.com

目 录

1 实验要求

1.1 实验目标

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

1.2 实验内容

- GitHub Actions DevOps 实践

配置实验开发环境和测试用的 GitHub 仓库，本地创建 Maven 项目并提交至仓库，编写 DevOps workflow 文件，提交工作流文件到仓库，并执行自动化测试，在工作流中增加 PR 自动合并功能（选做已完成）。

- Jenkins DevOps 实践

配置实验开发环境和测试用的 GitHub 仓库，本地创建 Maven 项目并提交至仓库，下载 GitHub CLI，完成 Jenkins 安装与配置，然后在 Jenkins 中进行 DevOps 操作。

2 实验内容 1 Github Actions DevOps 实践

2.1 实验过程

（一）实验开发环境

- Java JDK 21
- Git
- Maven

（二）测试用 Github 仓库

Github 仓库 url：https://github.com/OSSDP/Lab4-2023120240

（三）本地创建 maven 项目

在本地新建一个 Maven 项目，并引入 JUnit 依赖。

将 Lab2 中调试好的程序及单元测试文件导入到项目中。

（四）提交 maven 项目到步骤二中仓库

```
PS E:\A开源软件开发实践\Lab4-2023120240> git add .
PS E:\A开源软件开发实践\Lab4-2023120240> git commit -m"add maven project"
[main 00e5445] add maven project
 14 files changed, 337 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/encodings.xml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 lib/hamcrest-core-1.3.jar
 create mode 100644 lib/junit-4.12.jar
 create mode 100644 pom.xml
 create mode 100644 src/main/java/Solution2.java
 create mode 100644 src/main/resources/META-INF/maven/archetype.xml
 create mode 100644 src/main/resources/archetype-resources/pom.xml
 create mode 100644 src/main/resources/archetype-resources/src/main/java/App.java
 create mode 100644 src/main/resources/archetype-resources/src/test/java/AppTest.java
 create mode 100644 src/test/java/L2023120240_2_Test.java
```

```
PS E:\A开源软件开发实践\Lab4-2023120240> git push origin
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 8 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (32/32), 315.47 KiB | 15.77 MiB/s, done.
Total 32 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/OSSDP/Lab4-2023120240.git
   439cea3..00e5445  main -> main
```

（五）编写 DevOps workflow 文件

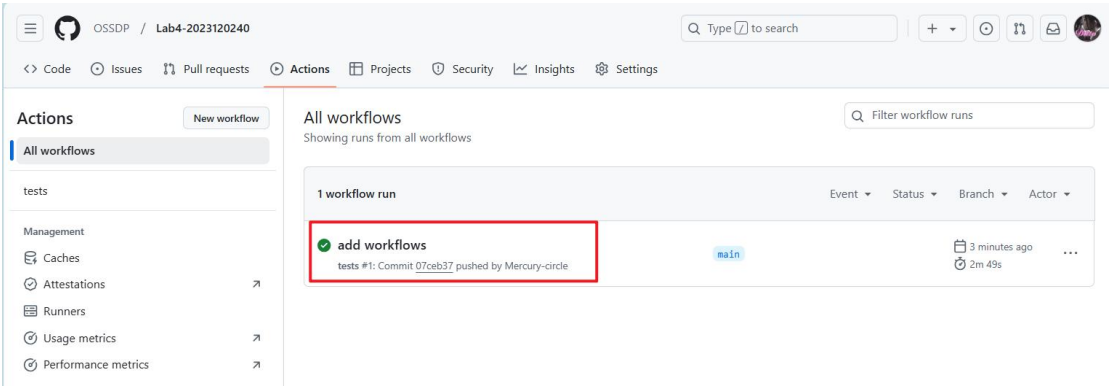
在项目根目录下新建文件夹.github/workflows/，新建 github-actions-demo.yml 文件如下：

```
name: tests
on:
  push:
    branches:
      - main
  pull_request:
    types: [opened, synchronize, reopened]

jobs:
  run_tests:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout the repository
        uses: actions/checkout@v2
      - name: Set up JDK 21
        uses: actions/setup-java@v1
        with:
          java-version: 21
      - 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 pom.xml
```

（六）提交工作流文件到仓库，并执行自动化测试

```
PS E:\A开源软件开发实践\Lab4-2023120240> git add .
PS E:\A开源软件开发实践\Lab4-2023120240> git commit -m"add workflows
[main 07ceb37] add workflows
 1 file changed, 26 insertions(+)
 create mode 100644 .github/workflows/github-actions-demo.yml
PS E:\A开源软件开发实践\Lab4-2023120240> git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 689 bytes | 689.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/OSSDP/Lab4-2023120240.git
   00e5445..07ceb37  main -> main
```

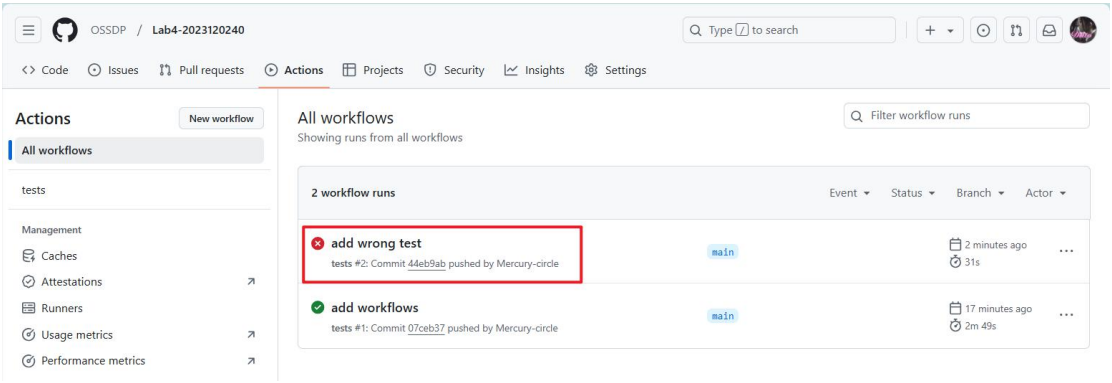


（七）再次执行自动化测试

修改测试用例为错误的结果，使其无法通过测试。修改后再次提交至仓库。

```java
// 等价类划分原则：所有字符都是唯一的字符串是一个独立的等价类
@Test
public void testAllUniqueCharacters() {
    String input = "abc";
    String expected = "acb"; 正确答案为"abc"
    assertEquals(expected, solution2.removeDuplicateLetters(input));
}
```

```
PS E:\A开源软件开发实践\Lab4-2023120240> git add .
PS E:\A开源软件开发实践\Lab4-2023120240> git commit -m"add wrong test"
[main 44eb9ab] add wrong test
 1 file changed, 1 insertion(+), 1 deletion(-)
PS E:\A开源软件开发实践\Lab4-2023120240> git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 461 bytes | 461.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/OSSDP/Lab4-2023120240.git
   07ceb37..44eb9ab  main -> main
```



（八）选作任务

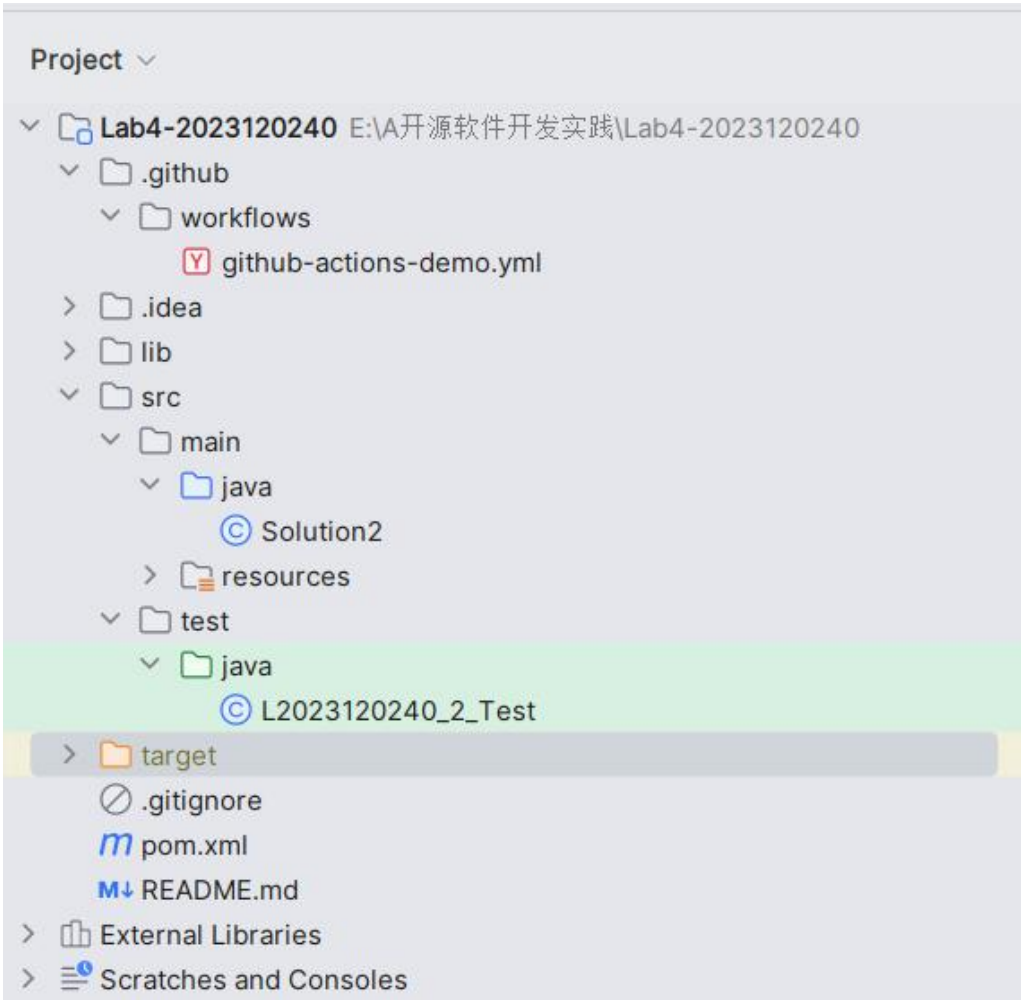
自行查询资料和文档，在上述工作流方法基础上，增加 PR 的自动 merge 功能，当某个 PR 通过了自动测试后，接受并 merge 到项目中。

在 yml 中加入下面的内容：

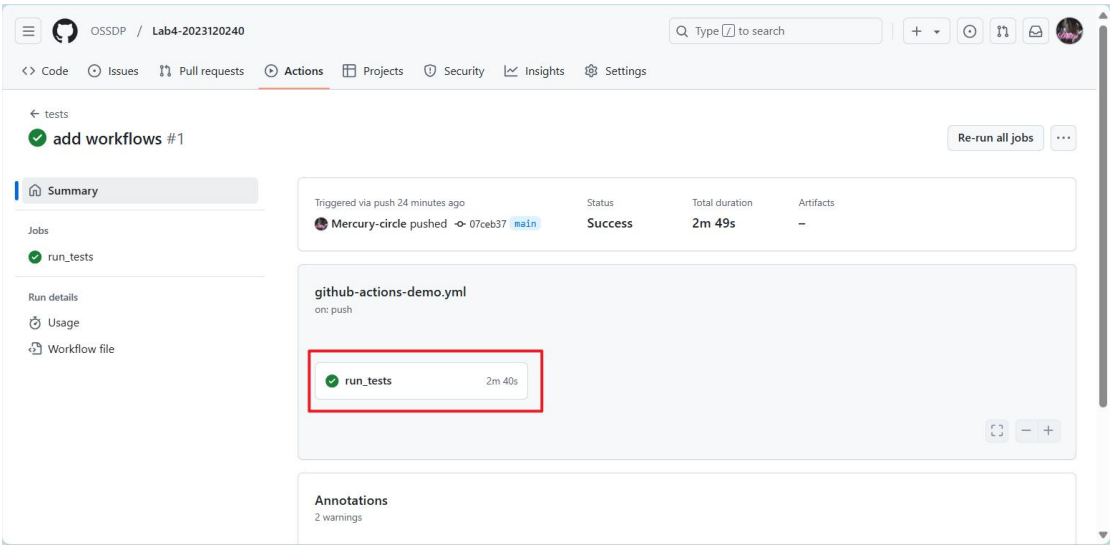
```yaml
auto-merge:
  needs: run_tests
  runs-on: ubuntu-latest
  if: success()
  steps:
    - name: Auto-merge PR
      uses: pascalgn/automerge-action@v0.16.3
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        MERGE_METHOD: "merge"
        MERGE_COMMIT_MESSAGE: "automatic"
        MERGE_FORKS: "true"
        MERGE_RETRIES: "50"
        MERGE_RETRY_SLEEP: "10000"
        UPDATE_METHOD: "rebase"
```

2.2 实验结果

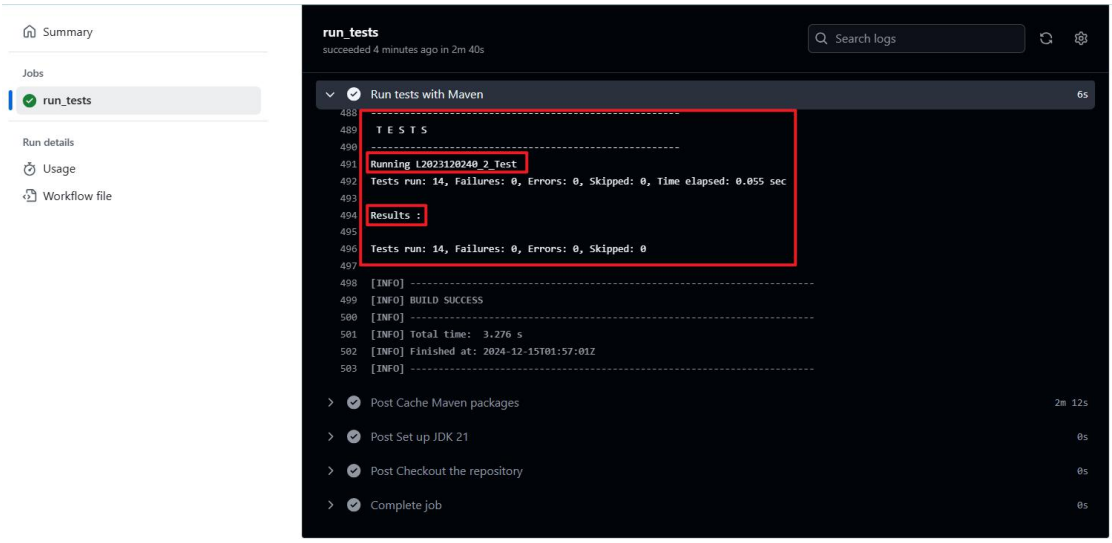
- 项目的目录结构截图如下：



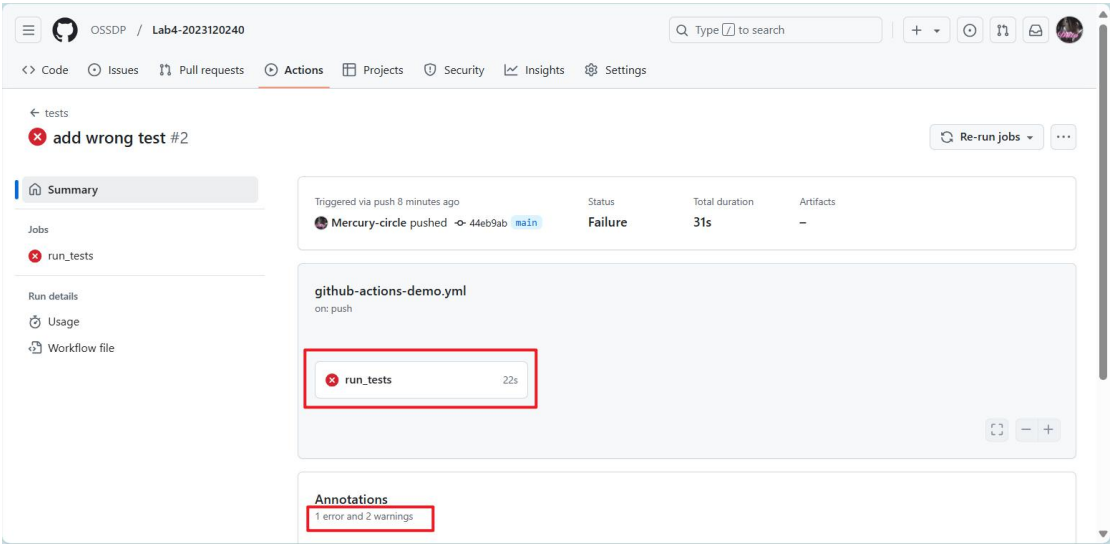
- 执行自动化测试成功界面如下：



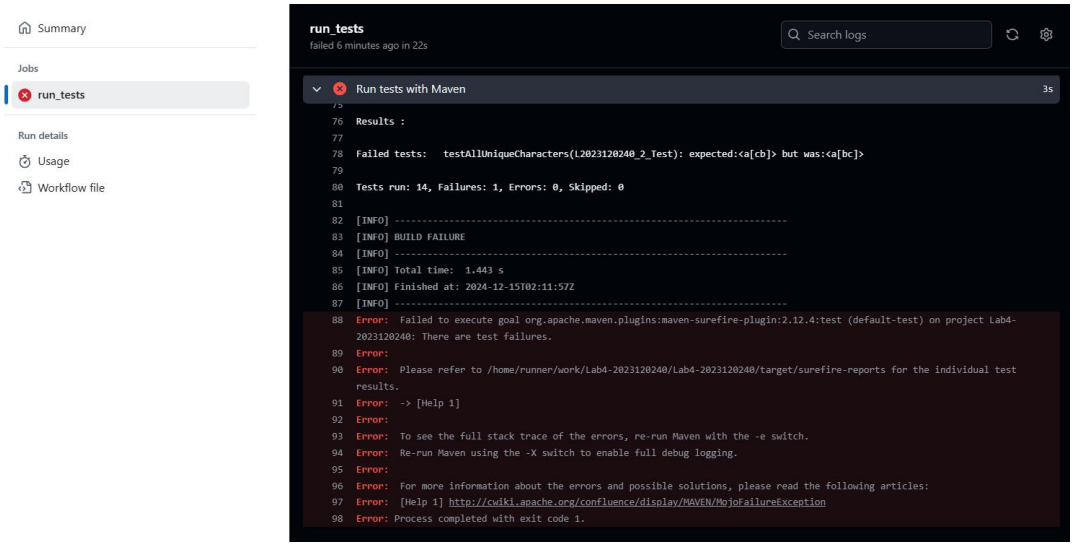
- 具体执行信息如下：



- 执行自动化测试失败界面如下：



- 具体错误信息如下：



- 针对步骤八的 YML 代码

```
name: tests
on:
  push:
    branches:
      - main
  pull_request:
    types: [opened, synchronize, reopened]

jobs:
  run_tests:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout the repository
        uses: actions/checkout@v2
      - name: Set up JDK 21
        uses: actions/setup-java@v1
        with:
          java-version: 21
      - 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 pom.xml

  auto-merge:
  needs: run_tests
  runs-on: ubuntu-latest
  if: success()
  steps:
    - name: Auto-merge PR
      uses: pascalgn/automerge-action@v0.16.3
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        MERGE_METHOD: "merge"
        MERGE_COMMIT_MESSAGE: "automatic"
        MERGE_FORKS: "true"
        MERGE_RETRIES: "50"
        MERGE_RETRY_SLEEP: "10000"
        UPDATE_METHOD: "rebase"
```

3 实验内容 2 Jenkins DevOps 实践

3.1 实验过程

（一）实验开发环境

- Java JDK 21
- Git
- Maven
- GitHub CLI
- Jenkins

（二）Jenkins 安装与配置

根据操作系统类型下载安装 Jenkins，配置服务运行方式（选择 Run Service as Local System 以避免额外账户配置）、端口号（默认为 8080）和 JAVA_HOME 路径。

安装完成后，通过 http://localhost:8080/ 访问 Jenkins 服务，解锁 Jenkins 并安装推荐的插件（包括 GitHub 插件）。

（三）GitHub CLI 注意事项

为提交 PR，需要在 GitHub 生成一个 Access Token，具体步骤包括：

点击 GitHub 头像，进入 Settings。

选择 Developer settings，点击 Personal access token，选择 Token (classic)。

生成新 token，勾选 repo 和 admin:org 权限，并记下 token。

（四）测试用 Github 仓库

同 2.1 使用同一个即可。

Github 仓库 url：https://github.com/OSSDP/Lab4-2023120240

（五）本地创建 maven 项目

同 2.1 中一致。

（六）构建 DevOps workflow

点击 Jenkins 主页上的新建 Item，创建新的构建流程，任务类型选择 Freestyle Project。

General 部分：对任务进行基本描述。

源码管理：配置对 GitHub 仓库的访问，使用 SSH 密钥认证。

构建触发器：配置如何触发构建流程，如使用定时检查仓库更新的方式（每 3 分钟检查一次）。

构建环境：设置构建流程的环境变量。

构建步骤：编写命令以执行代码测试和 PR 提交，使用 GitHub CLI 提交 PR。

（七）验证实验效果

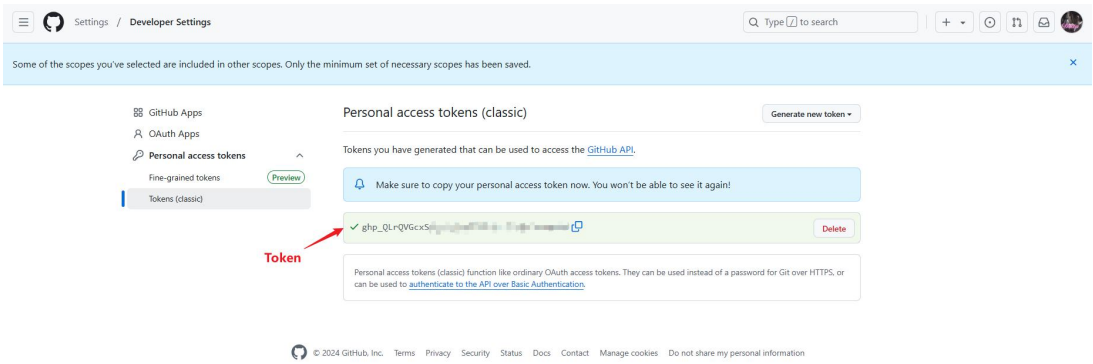
将项目代码提交到选定的仓库的指定分支，等待定时任务进行构建流程。

查看 Jenkins Build 的结果是否成功。

在 GitHub 上查看仓库的 PR 是否推送成功。

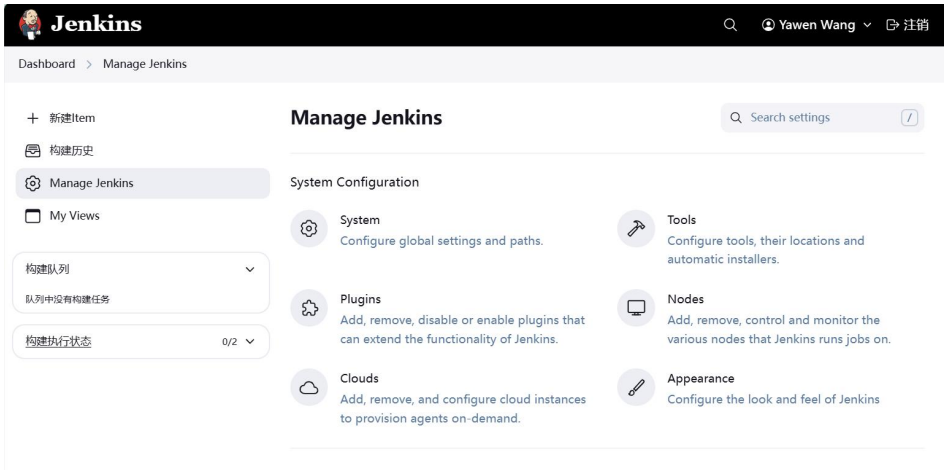
3.2 实验结果

- 在 GitHub 生成一个 Access Token

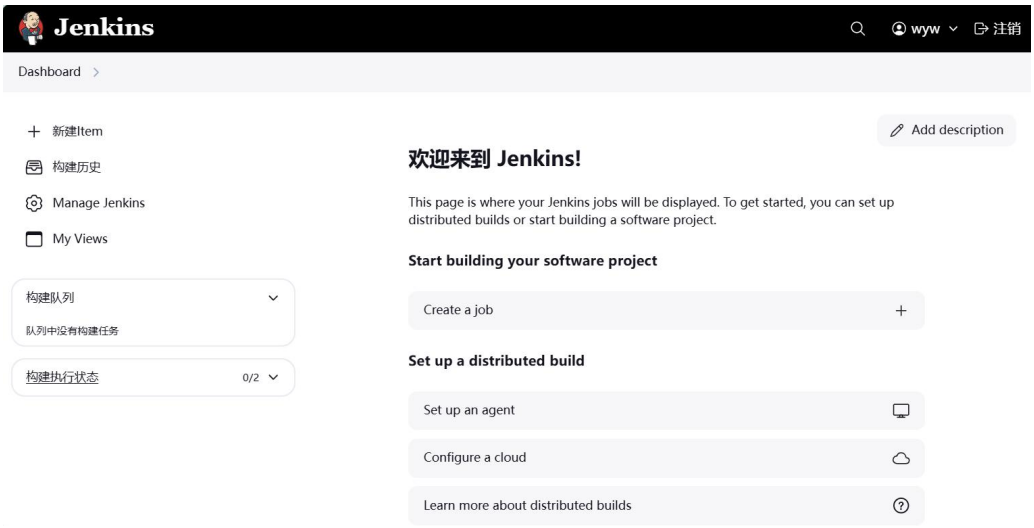


- 登录 Jenkins

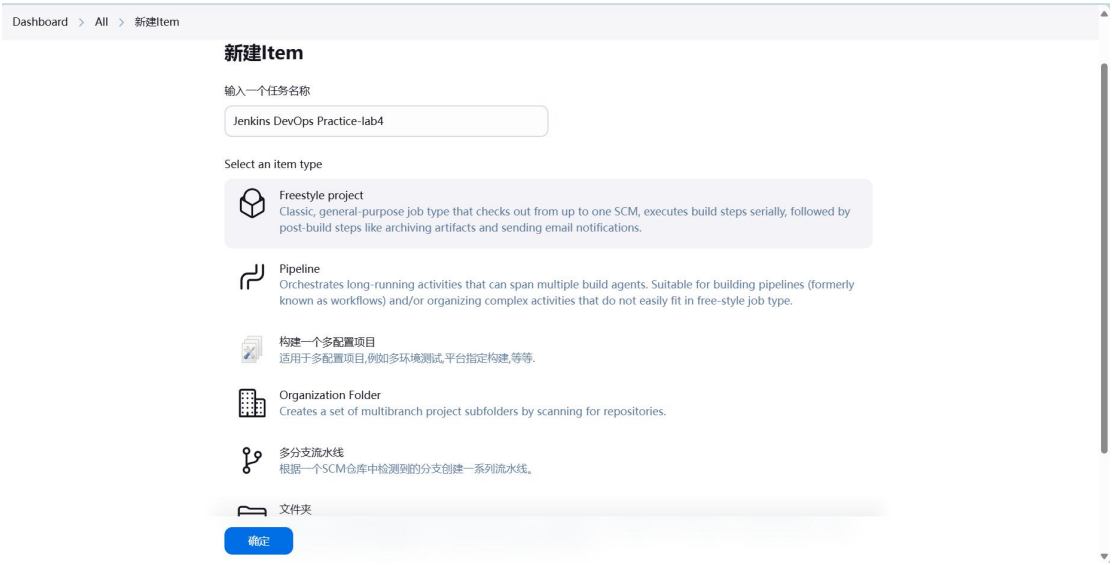
- 管理员账户界面如下



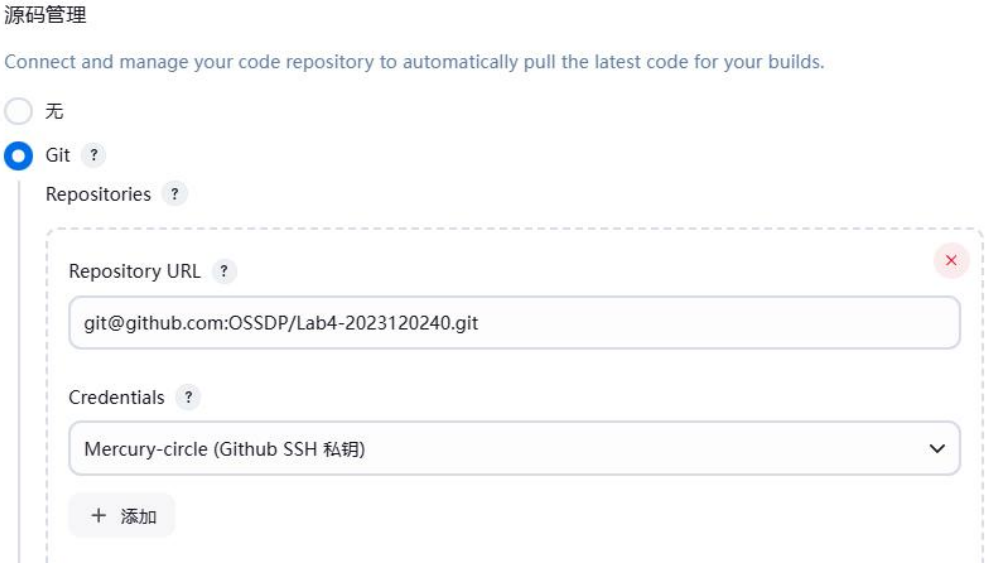
- 创建一个非管理员账户如下：



- 新建 Item



- 配置私钥成功可使用



- 指定 local 代码所在分支

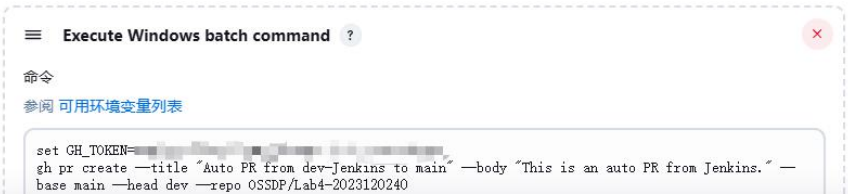

Branches to build
指定分支（为空时代表any）
*/dev
Add Branch
源码库浏览器
(自动)
Additional Behaviours
新增

- 构建触发器

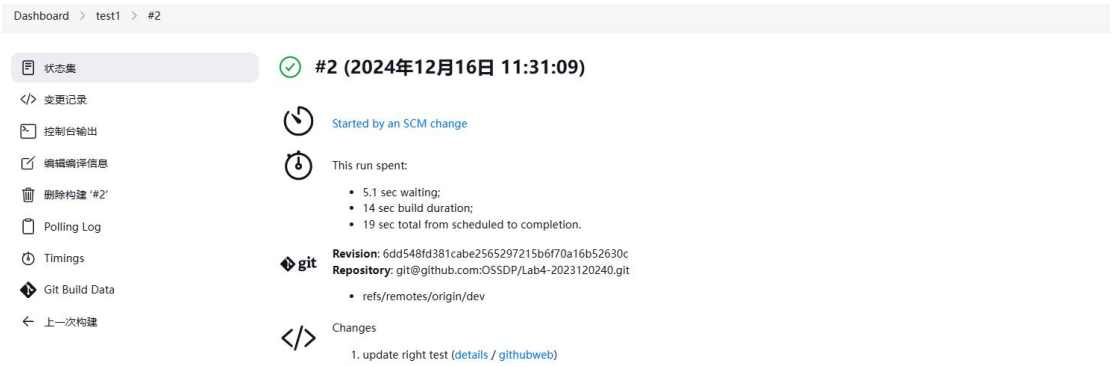

Triggers
Set up automated actions that start your build based on specific events, like code changes or scheduled times.
触发远程构建 (例如,使用脚本)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
日程表
H/3 * * * *
Would last have run at 2024年12月16日星期一 中国标准时间 11:10:29; would next run at 2024年12月16日星期一 中国标准时间 11:13:29.
忽略钩子 post-commit


Execute Windows batch command
命令
参阅 可用环境变量列表
mvn validate
高级

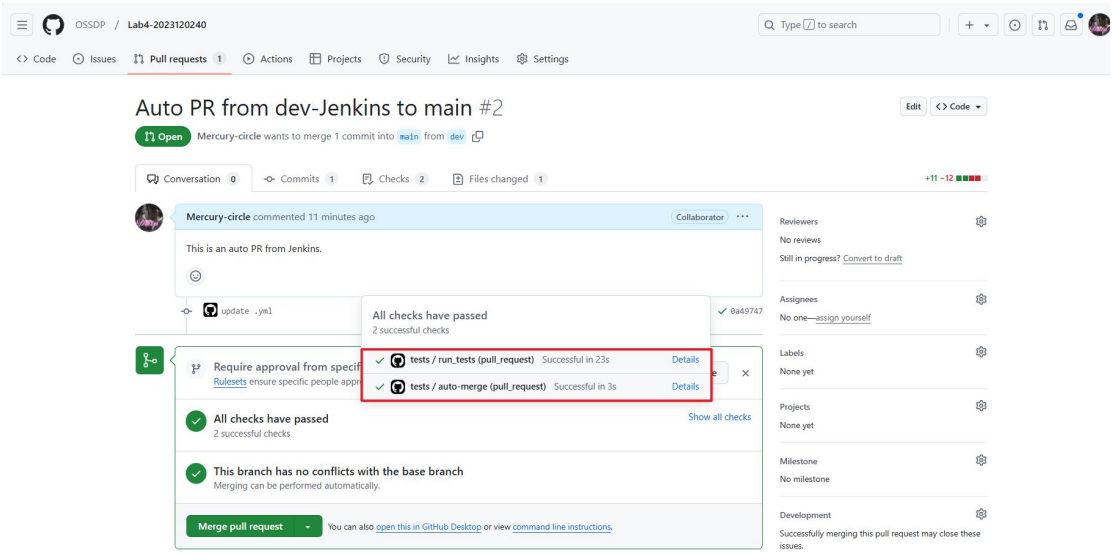

Execute Windows batch command
命令
参阅 可用环境变量列表
mvn test
高级


Execute Windows batch command
命令
参阅 可用环境变量列表
set GH_TOKEN=
gh pr create --title "Auto PR from dev-Jenkins to main" --body "This is an auto PR from Jenkins." --base main --head dev --repo OSSDP/Lab4-2023120240

- 构建成功



- 在Github上查看仓库的PR推送成功，实验1的第8步骤设置的自动merge也成功了。



4 小结

通过这次实验，我对开源软件开发中的基本DevOps操作有了更深入的了解。在实验中，我分别使用了 GitHub Actions 和 Jenkins 来实现自动化测试和 PR 自动合并。两者都有各自的优缺点。

GitHub Actions 与 GitHub 生态系统紧密集成，使用便捷，无需自己管理服务器，由 GitHub 托管。相对于 Jenkins，自定义选项较少，但构建速度有点慢。

Jenkins 可以完全自定义构建流程，有大量插件支持各种需求，可以部署在任何服务器上，适应不同的环境，但是初始配置和维护相对复杂，需要自己管理服务器，增加运维成本。

在实验中，我实现了 PR 的自动合并功能。这种自动化流程可以大大提高开发效率，减少人工操作的错误。不过自动合并的前提是测试通过，但测试可能不完全覆盖所有情况，仍需人工审查。在实验中因为 yml 文件不正确，导致第一次 PR 时出现 merge 失败的情况，后面又改了新的写法，才顺利通过。在这次实验

里我学会了如何编写和配置 workflow 文件，理解了如何利用 GitHub Actions 进行自动化测试和 CI/CD 流程。通过自己一步步配置，掌握了 Jenkins 的基本使用方法，然后对 DevOps 的整体流程也有了更清晰的认识。

对于本次实验的建议是，在实验指导书里可以增添一些本实验中 Github CLI 主要作用的解释。