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
2022211908	谢亚东	2278732301@qq.com/18338468543

目 录

1 实验要求

本次实验的目标是训练学生掌握在开源软件开发中常用的 DevOps 操作，特别是使用 GitHub Actions 和 Jenkins 两种工具进行自动化构建、测试和部署。具体要求包括：

掌握 DevOps 流程和工具的基本操作，熟悉如何利用 GitHub Actions 和 Jenkins 进行 DevOps 实践。

熟悉使用 GitHub Actions 进行自动化测试和构建，通过编写工作流（YML 文件）实现持续集成（CI）过程。

熟悉使用 Jenkins 进行 DevOps 实践，包括配置 Jenkins 环境、创建构建任务、与 GitHub 集成等。

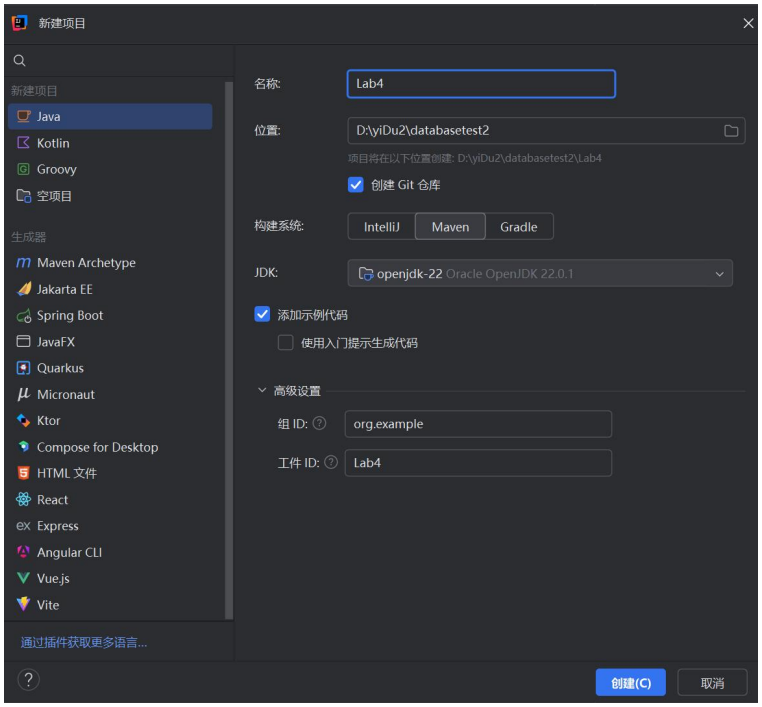
实验过程分为两个主要部分：

GitHub Actions DevOps 实践，在 GitHub Classroom 创建个人仓库，并通过 Git 提交代码。本地创建 Maven 项目，并引入 JUnit 测试依赖，编写单元测试。配置 GitHub Actions 自动化测试流程，通过编写 YML 工作流文件，实现自动化构建和单元测试的运行。在 GitHub 仓库中提交工作流文件，观察自动化测试的执行情况，并修改测试用例，验证测试失败和工作流的执行。

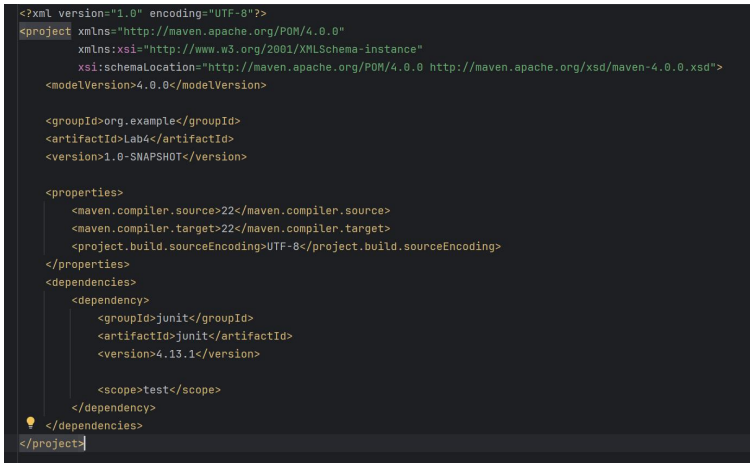
Jenkins DevOps 实践，安装并配置 Jenkins，包括配置 GitHub 仓库访问权限和设置构建触发器。在 Jenkins 中创建 Freestyle 项目，设置 GitHub 仓库与 Jenkins 的连接。配置定时构建或 GitHub hook 触发器，以实现自动化构建和 PR 提交。使用 GitHub CLI 自动创建 PR，并验证 Jenkins 构建是否成功。

2 实验内容 1 Github Actions DevOps 实践

首先创建一个 maven 项目：



引入 junit 依赖：



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>Lab4</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>22</maven.compiler.source>
        <maven.compiler.target>22</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.1</version>

            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

建立 git 仓库：

```
22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git init
Reinitialized existing Git repository in D:/yiDu2/databasetest2/Lab4/.git/

22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git add .
warning: in the working copy of 'pom.xml', LF will be replaced by CRLF the next
time Git touches it

22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git commit -m "Initial commit with Maven project"
[master (root-commit) 27efa58] Initial commit with Maven project
 9 files changed, 273 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 pom.xml
 create mode 100644 src/main/java/Solution10/Solution10.java
 create mode 100644 src/main/java/org/example/Main.java
 create mode 100644 src/test/java/L2022211908_10_Test/L2022211908_10_Test.java
```

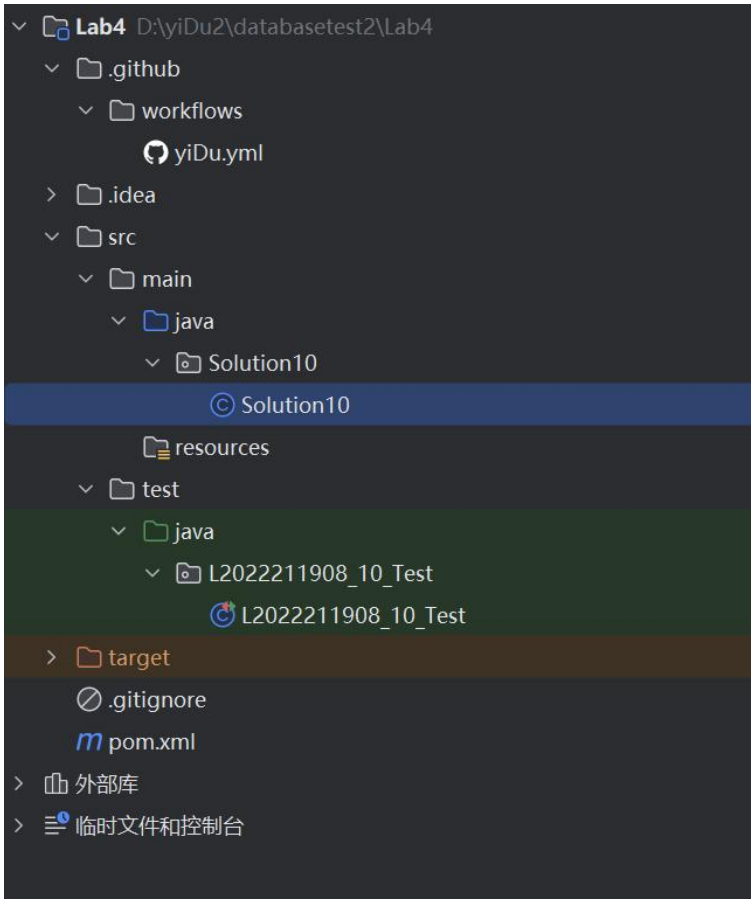
通过 git 上传到远程仓库：https://github.com/OSSDP/Lab4-2022211908：

```
22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git remote add origin https://github.com/OSSDP/Lab4-2022211908

22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git remote -v
origin  https://github.com/OSSDP/Lab4-2022211908 (fetch)
origin  https://github.com/OSSDP/Lab4-2022211908 (push)

22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git push -u origin master
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 24 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (21/21), 4.37 KiB | 4.37 MiB/s, done.
Total 21 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/OSSDP/Lab4-2022211908/pull/new/master
remote:
To https://github.com/OSSDP/Lab4-2022211908
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

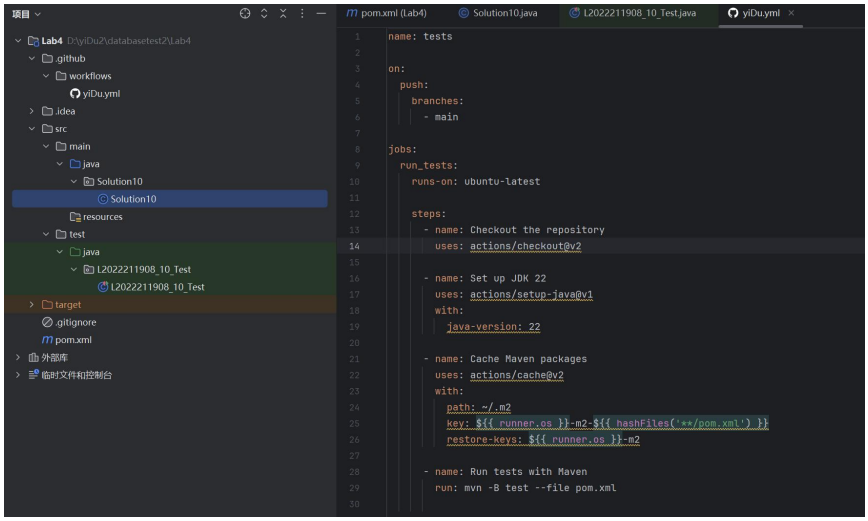
在项目根目录下新建文件夹.github/workflows/，然后新建 yiDu.yml 文件：

```
22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git add .github/workflows/yiDu.yml

22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git commit -m "Add GitHub Actions workflow for automated tests"
[master c533f15] Add GitHub Actions workflow for automated tests
 1 file changed, 29 insertions(+)
 create mode 100644 .github/workflows/yiDu.yml

22787@yiDu26 MINGW64 /d/yiDu2/databasetest2/Lab4 (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 24 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 665 bytes | 665.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/OSSDP/Lab4-2022211908
   27efa58..c533f15  master -> master
```

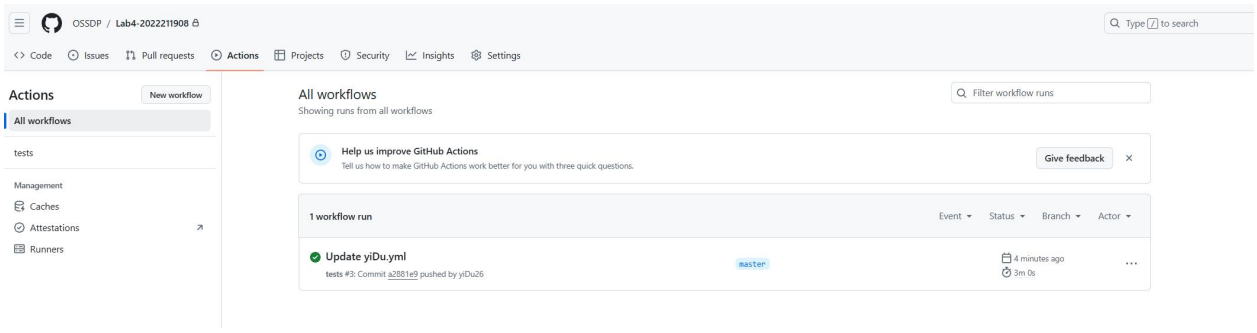
项目的目录结构截图:



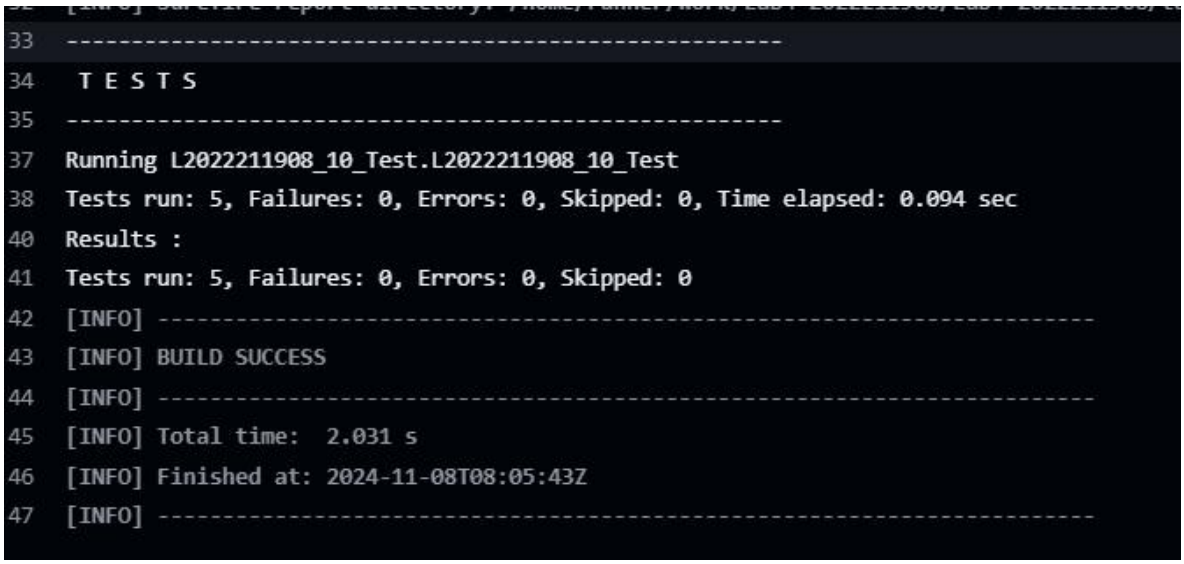
yiDu.yml 文件：



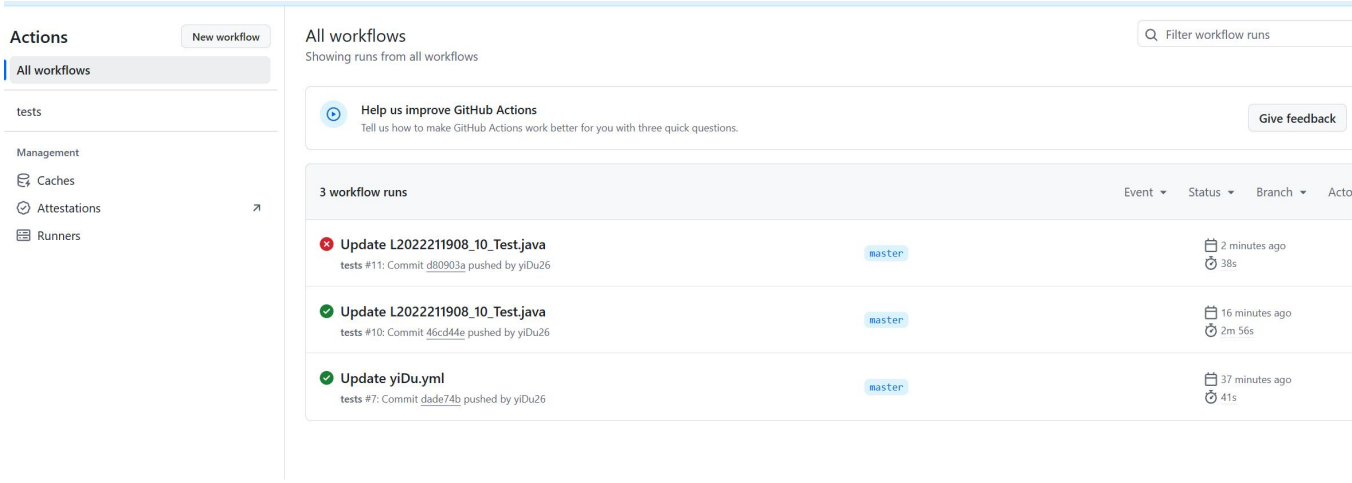
编写完工作流文件后，再次提交到仓库，Github 根据 yml 文件进行自动化测试：



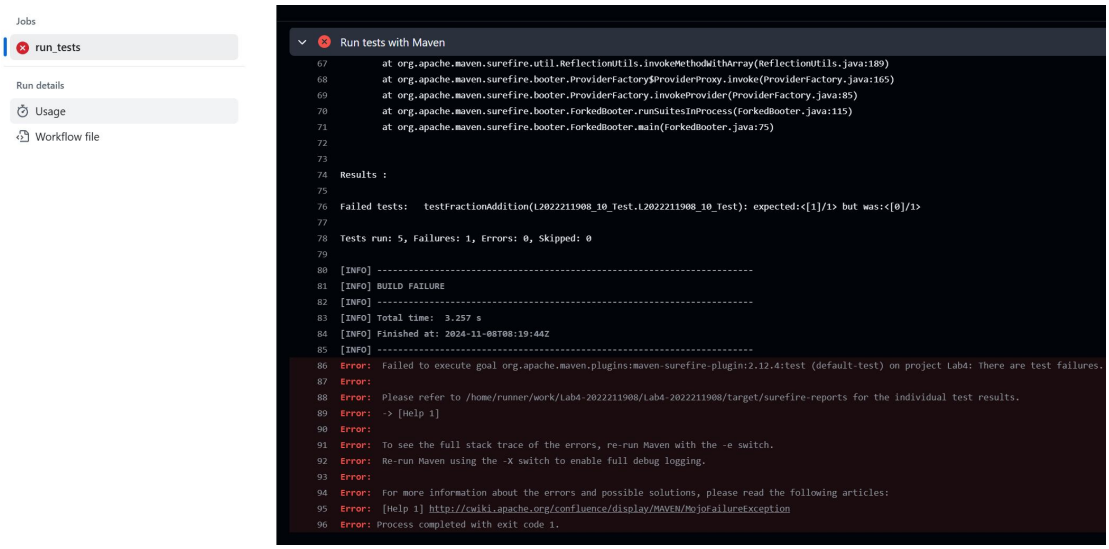
在仓库中点击 Actions 图标，在 All workflows 中可以查看到已提交的工作流。点击进去可以查看到具体的执行流程，可以看到下图中测试成功。



修改测试用例为错误的结果，使其无法通过测试：

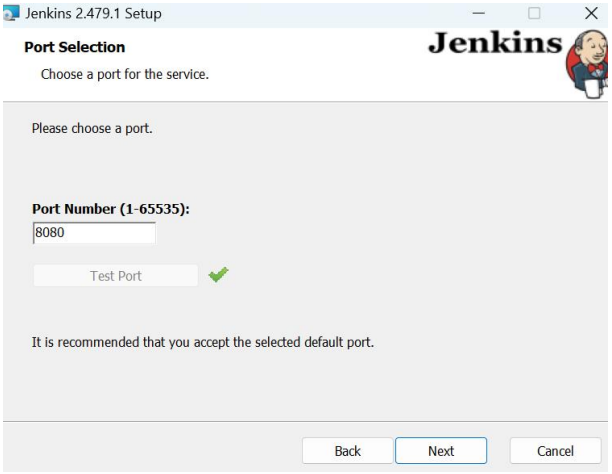


详细信息：

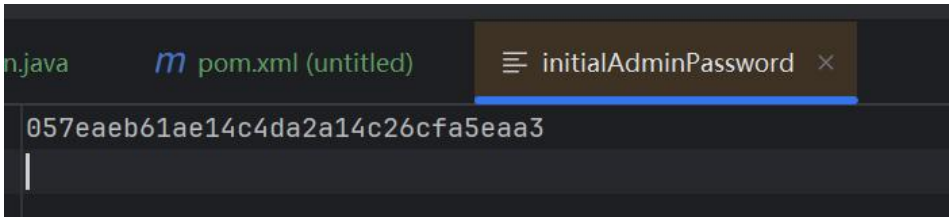


3 实验内容 2 Jenkins DevOps 实践

在 https://www.jenkins.io/zh/download/网址下载并安装 jenkins：

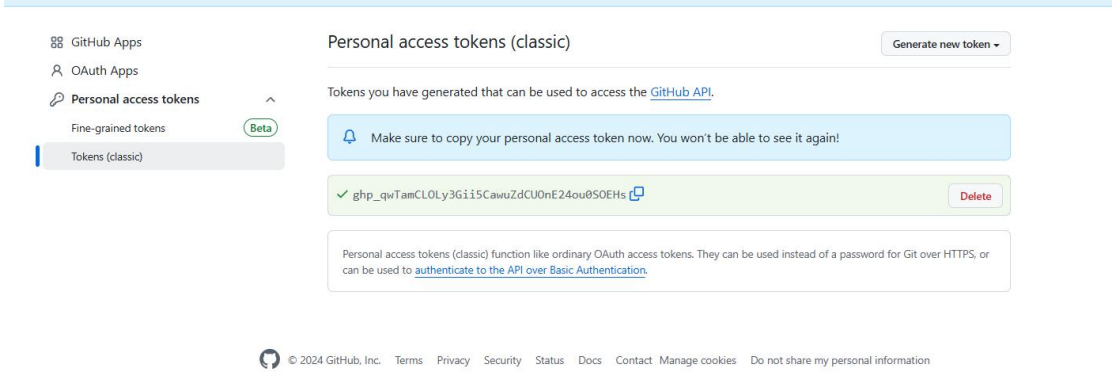


初始登录密码如下：



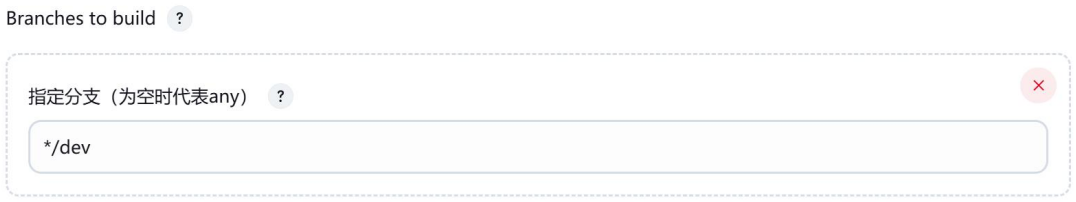
登录成功：

申请一个 Access Token：



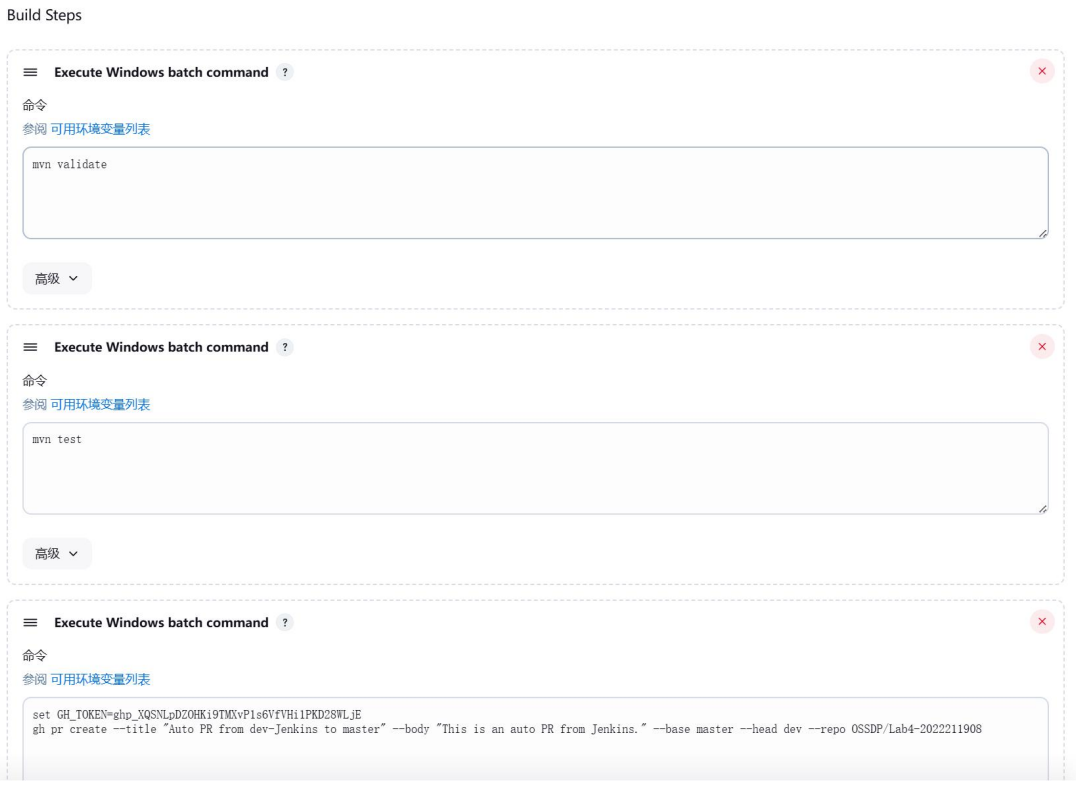
源码管理：

指定 Local 项目代码所在的分支：

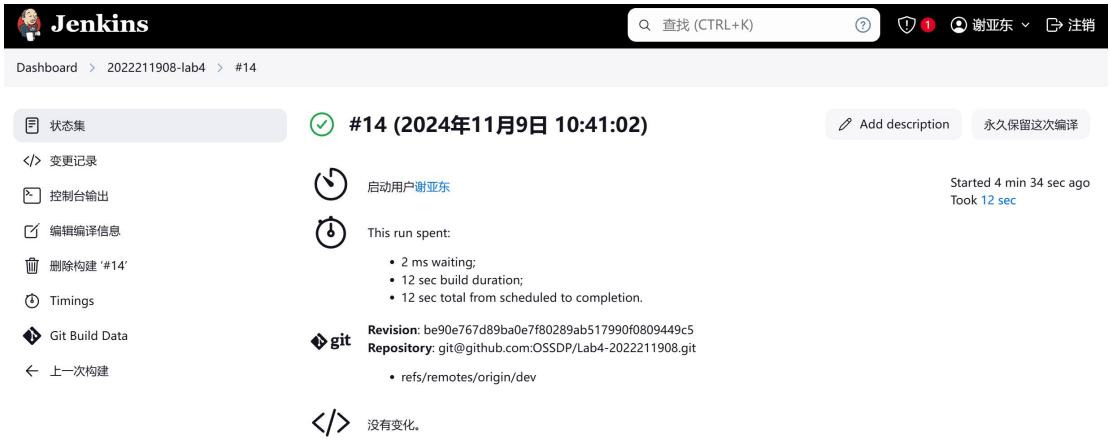


构建触发器：

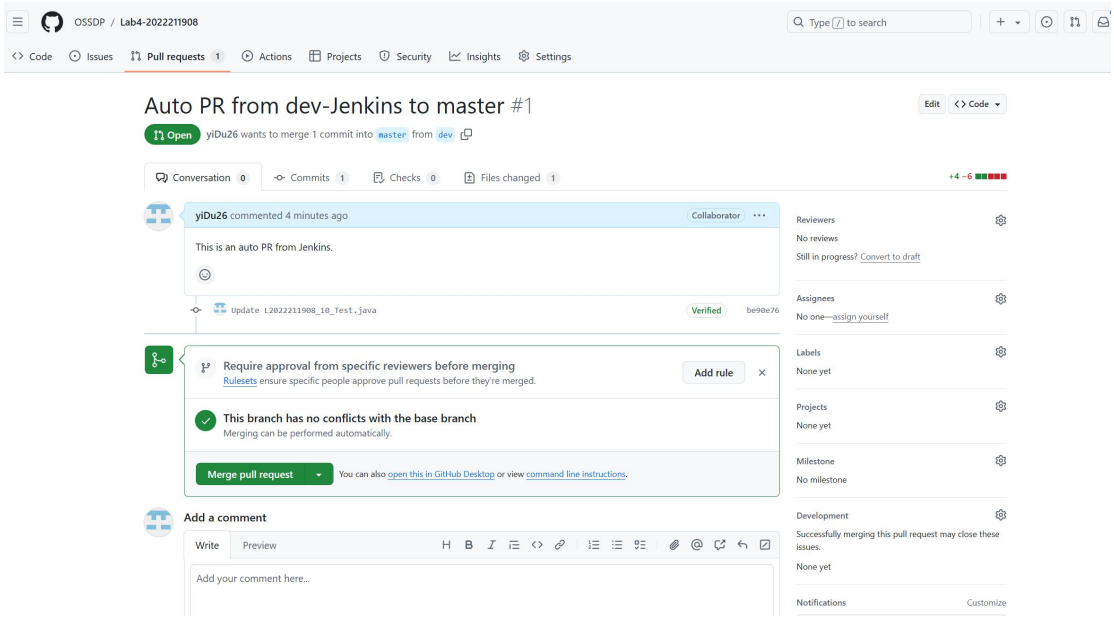
构建步骤：



Jenkins Build 结果成功：



在 Github 上查看仓库的 PR 推送成功



4 小结

本次实验主要围绕 DevOps 工具的应用进行，分别通过 GitHub Actions 和 Jenkins 两个工具来实践自动化构建和持续集成（CI）/持续交付（CD）流程。实验过程分为两部分：首先是使用 GitHub Actions 设置自动化测试流程，通过编写 YAML 配置文件实现自动化测试和错误检测；其次是在 Jenkins 环境下配置并执行与 GitHub 集成的自动化构建流程，通过 GitHub CLI 创建 PR 来模拟自动化部署的完整流程。

通过本次实验，我对 DevOps 流程有了更深的理解，尤其是如何将自动化测试和构建集成到 GitHub 仓库和 Jenkins 环境中。实践中，GitHub Actions 提供了一个灵活且高效的工具来管理 CI/CD 流程，而 Jenkins 则通过强大的插件和集成功能，能够满足更复杂的构建需求。此外，了解了如何配置和使用 GitHub CLI 来自动化提交 PR，以及如何处理 Jenkins 配置和构建触发器，为将来开发与运维一体化的工作奠定了基础。

在实验过程中，遇到了几个挑战。首先是 GitHub Actions 中 YML 文件的语法问题，尤其是在设置测试环境和缓存时，曾因版本不一致或路径错误导致自动化测试无法正常执行。解决这些问题后，测试流程顺利进行。其次，Jenkins 的配置过程较为复杂，尤其是 SSH 密钥和 GitHub 仓库的权限设置部分。在配置 Jenkins 与 GitHub 的连接时，我遇到了认证问题，通过查阅文档并确保私钥配置正确后，成功解决了问题。

这些挑战让我更加熟悉了 DevOps 工具的配置和调试，也提高了我在自动化工具配置方面的实践能力。

本次实验整体上比较顺利，但对于初学者来说，某些步骤可能稍显繁琐，尤其是 Jenkins 的配置和 GitHub CLI 的使用。为了帮助学生更好地理解和掌握这些工具，可以提供更为详细的教程，尤其是在 Jenkins 配置和 GitHub ActionsYML 文件编写的部分。