

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
2023120261	李璇	L1xxx59@163.com/15193215452

## 目 录

1 实验要求 .....	1
2 实验内容 1 Github Actions DevOps 实践 .....	2
3 实验内容 2 Jenkins DevOps 实践 .....	6
4 小结 .....	13

# 1 实验要求

(1) 本次实验旨在训练学生掌握开源软件开发中的基本 DevOps 操作，具体包括：

理解和熟悉开源软件开发中的基本 DevOps 流程。

学习并实践使用 GitHub Actions 和 Jenkins 进行 DevOps 操作。

(2) 实验过程

访问 GitHub Classroom：通过提供的 URL

(<https://classroom.github.com/a/T75S58J->)，按照提示建立自己的 Lab4 仓库，并将仓库命名规则设置为 Lab4-学号。

(3) 使用 GitHub Actions 实践 DevOps

创建 Maven 项目：在本地新建一个 Maven 项目，并引入 JUnit 测试依赖。确保项目包含调试后的程序代码以及编写的单元测试文件。

提交项目到 GitHub：将本地 Maven 项目提交至步骤一中创建的 GitHub 仓库。

编写 DevOps 工作流文件：在项目的根目录下创建 .github/workflows/ 文件夹，并在其中添加一个新的 YAML 文件来定义自动化测试的工作流。

执行自动化测试：提交工作流文件后，GitHub 将根据 YAML 文件自动运行测试。可以通过点击仓库中的 "Actions" 图标查看测试结果。

验证失败情况：修改测试用例以使其无法通过，再次提交以验证错误提交的结果。

(4) 使用 Jenkins 实践 DevOps

环境准备：安装 Java JDK、Git、Maven 和 Jenkins，同时安装 GitHub CLI 以便从命令行推送 PR。

Jenkins 配置：下载并安装 Jenkins，进行必要的初始配置，如选择端口、设置 JAVA\_HOME 等。

认证设置：生成 GitHub 访问令牌用于 Jenkins 提交 PR。

构建 DevOps 流程：创建新的 Jenkins 构建项，配置源码管理、构建触发器、构建环境及构建步骤，例如通过定时检查仓库更新来进行自动化构建。

验证效果：提交代码到指定分支，等待 Jenkins 定时任务触发构建流程，检查构建结果是否成功，并确认 PR 是否正确推送。

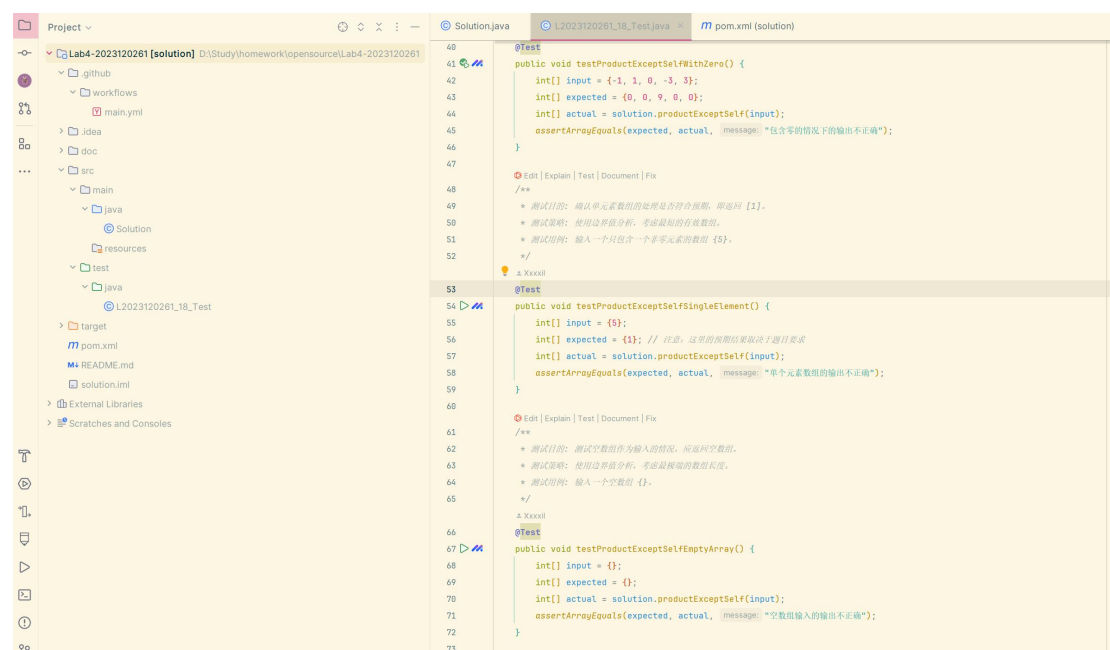
## 2 实验内容 1 Github Actions DevOps 实践

### (1) 项目目录结构

在本地环境中新建了一个 Maven 项目，并确保项目结构正确无误。

引入了 JUnit 作为测试框架依赖，用于编写单元测试案例。

编写了一些简单的 Java 程序代码以及对应的单元测试文件，保证所有测试都能顺利通过。

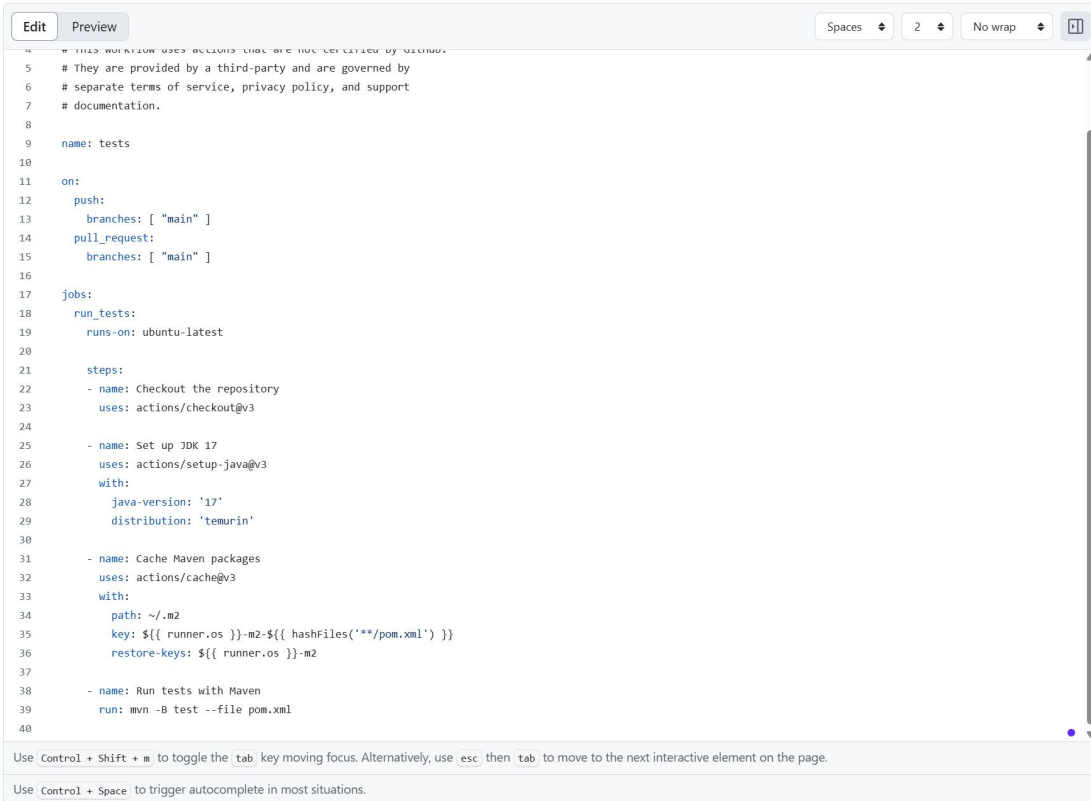


### (2) Actions 的若干界面：

#### ① 在项目的根目录下创建了 .github/workflows/ 文件夹。

在此文件夹内添加了一个名为 main.yml 的 YAML 格式文件，用来定义 CI 流水线的工作流规则。

配置了触发条件、构建环境（例如 Java 版本）、构建命令（如 mvn clean test），以及如何处理构建结果。

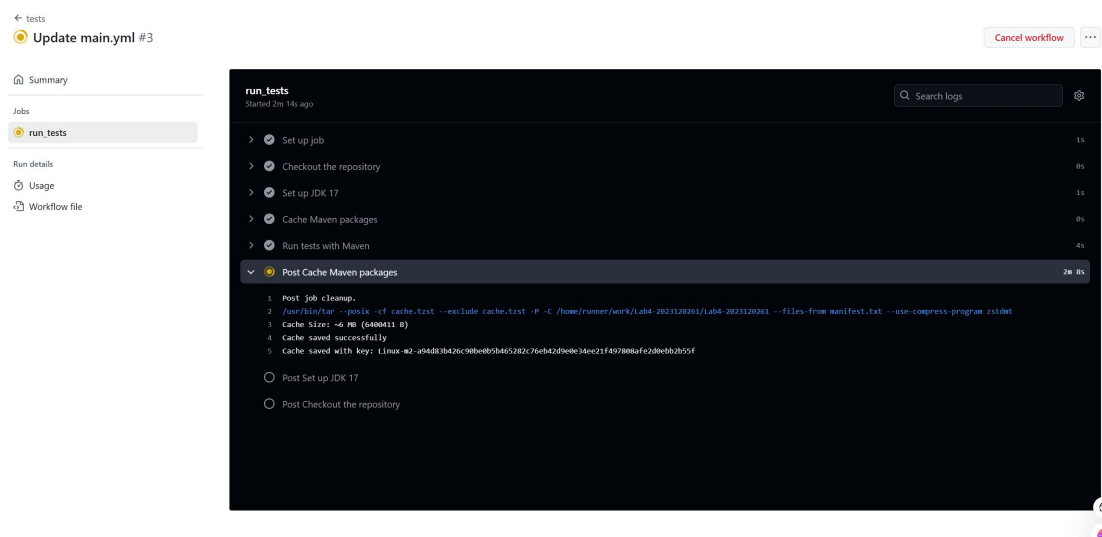


```
1 # This workflow uses actions that are not certified by GitHub.
2 # They are provided by a third-party and are governed by
3 # separate terms of service, privacy policy, and support
4 # documentation.
5
6 name: tests
7
8 on:
9   push:
10     branches: [ "main" ]
11   pull_request:
12     branches: [ "main" ]
13
14 jobs:
15   run_tests:
16     runs-on: ubuntu-latest
17
18     steps:
19       - name: Checkout the repository
20         uses: actions/checkout@v3
21
22       - name: Set up JDK 17
23         uses: actions/setup-java@v3
24         with:
25           java-version: '17'
26           distribution: 'temurin'
27
28       - name: Cache Maven packages
29         uses: actions/cache@v3
30         with:
31           path: ~/.m2
32           key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
33           restore-keys: ${{ runner.os }}-m2
34
35       - name: Run tests with Maven
36         run: mvn -B test --file pom.xml
```

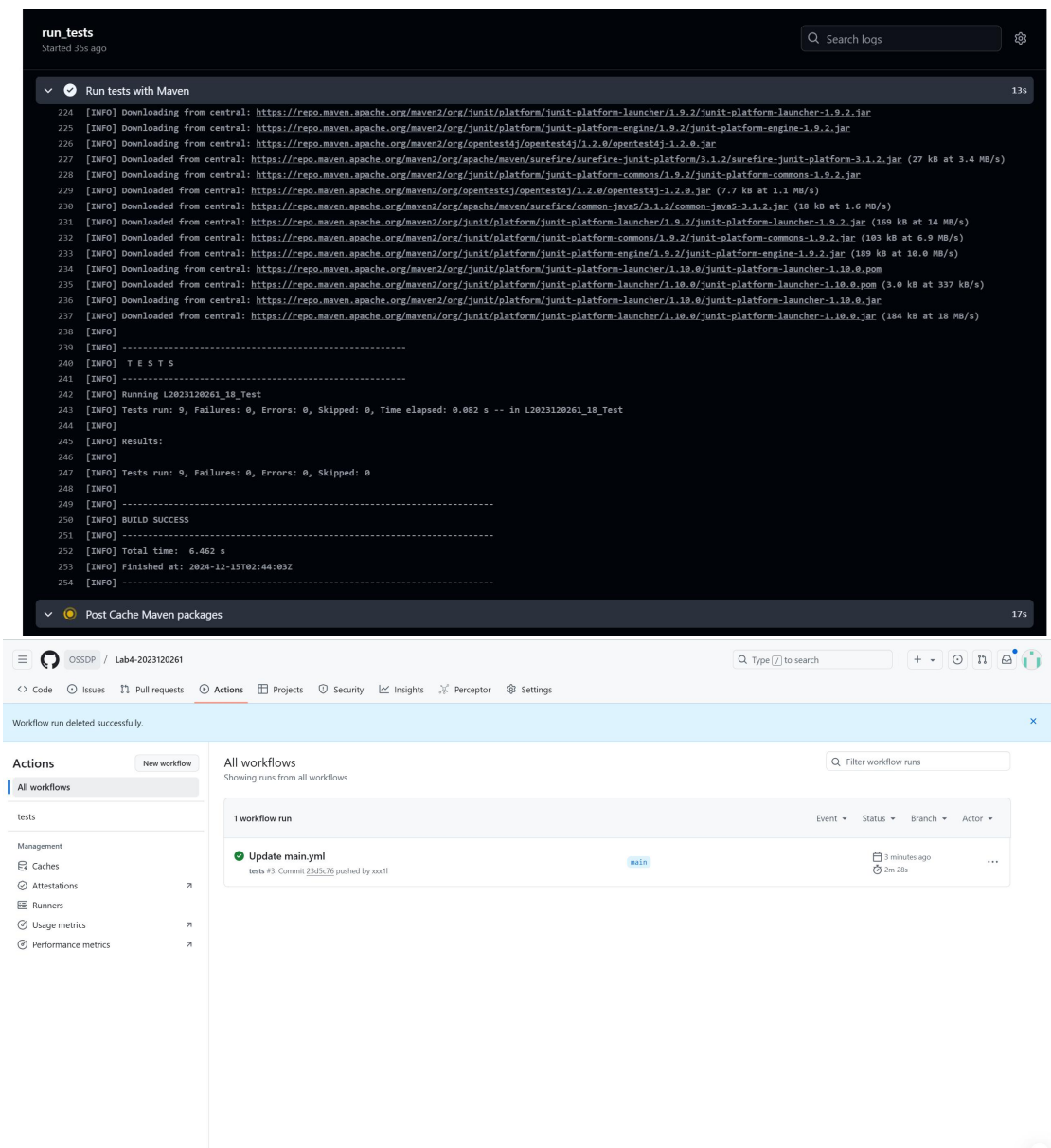
## ② 执行自动测试界面

将上述配置好的工作流文件推送到 GitHub 仓库后，GitHub 会根据设定自动触发构建过程。

通过点击仓库页面上的"Actions"标签来查看每次构建的状态和详细的日志信息。



## ③ 自动化测试执行成功



### (3) 修改测试用例为错误的结果

```
/**
 * 测试目的：检查当输入数组包含零时，函数是否能正确处理并返回预期结果。
 * 测试策略：使用边界值分析，考虑至少存在一个0的情况。
 * 测试用例：输入一个包含至少一个0的数组 {-1, 1, 0, -3, 3}。
 */

@Test
public void testProductExceptSelfWithZero() {
    int[] input = {-1, 1, 0, -3, 3};
    int[] expected = {0, 0, 9, 0, 1};
    int[] actual = solution.productExceptSelf(input);
    assertEquals(expected, actual, "包含零的情况下的输出不正确");
}
```

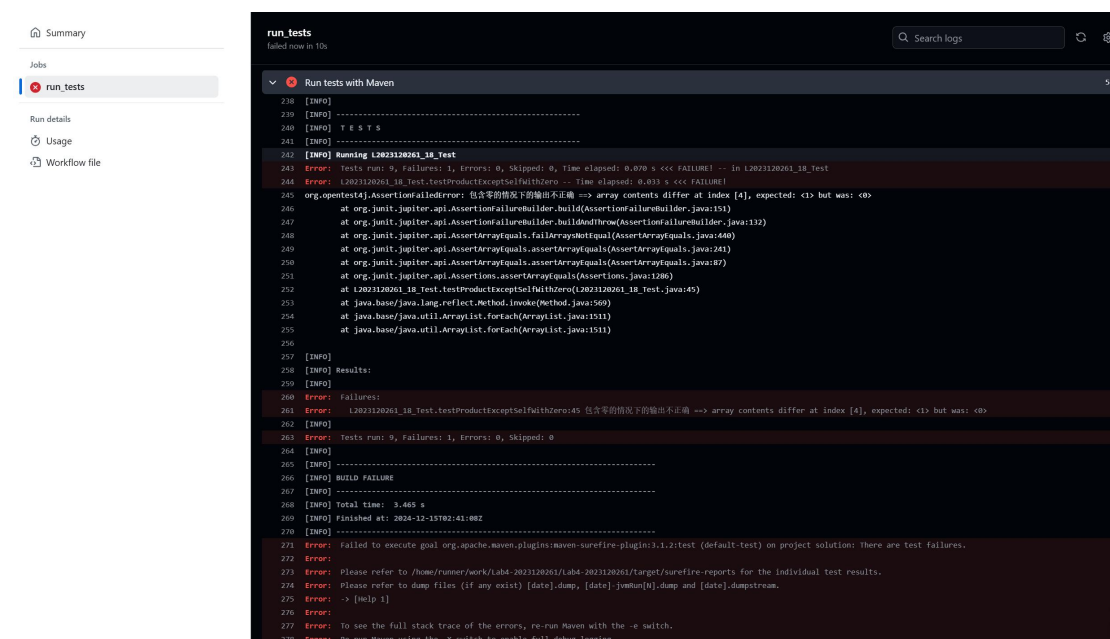
使其无法通过测试

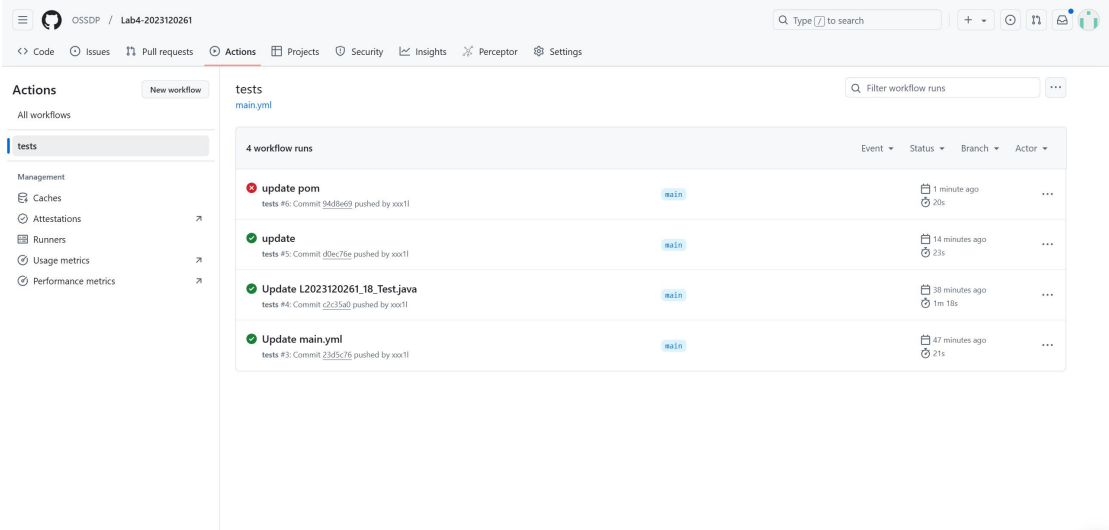


(4) 修改后再次提交至仓库，下面展示了错误提交的结果

故意修改部分测试用例以使其无法通过，再次提交更改后的代码到仓库。

观察到 GitHub Actions 成功检测到了错误，并报告了构建失败的信息，证明了 CI 设置是有效的。

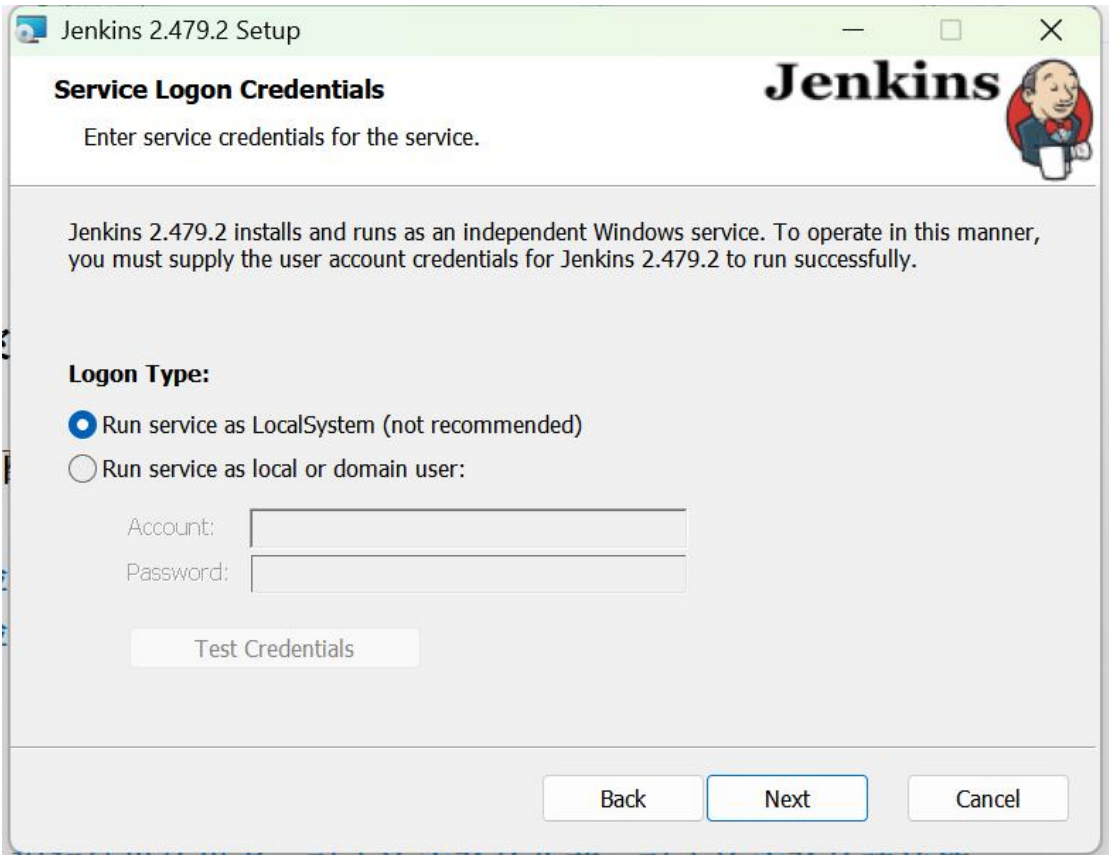




3 实验内容 2 Jenkins DevOps 实践

(1) Jenkins 安装与配置

下载并安装 Jenkins，完成初次启动时的基本设置，比如选择监听端口、指定 JAVA\_HOME 等环境变量。







The image shows two screenshots from the Jenkins installation process. The top screenshot is the 'Jenkins 2.479.2 Setup' window. It has a title bar with the Jenkins logo and the text 'Jenkins 2.479.2 Setup'. Below the title bar, it says 'Select Java home directory (JDK or JRE)'. There is a text box containing 'C:\Program Files\Java\jdk-17\' and a 'Change...' button below it. At the bottom, there are 'Back', 'Next', and 'Cancel' buttons. The bottom screenshot is the 'Jenkins unlock' screen. It has a title bar with the text '入门' (Getting Started). The main heading is '解锁 Jenkins' (Unlock Jenkins). Below it, there is a paragraph of text in Chinese: '为了确保管理员安全地安装 Jenkins, 密码已写入到日志中 (不知道在哪里?) 该文件在服务器上:'. This is followed by a red code block containing the path 'C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword'. Below this, it says '请从本地复制密码并粘贴到下面。' (Please copy the password from the local machine and paste it below). There is a label '管理员密码' (Admin Password) and a password input field with dots. At the bottom right, there is a blue button labeled '继续' (Continue).

Jenkins 2.479.2 Setup

Select Java home directory (JDK or JRE)

Please select the path of a Java Development Kit or Java Runtime Environment. Only Java 17 and 21 are supported by Jenkins.

C:\Program Files\Java\jdk-17\

Change...

Back Next Cancel

入门

## 解锁 Jenkins

为了确保管理员安全地安装 Jenkins, 密码已写入到日志中 (不知道在哪里?) 该文件在服务器上:

```
C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
```

请从本地复制密码并粘贴到下面。

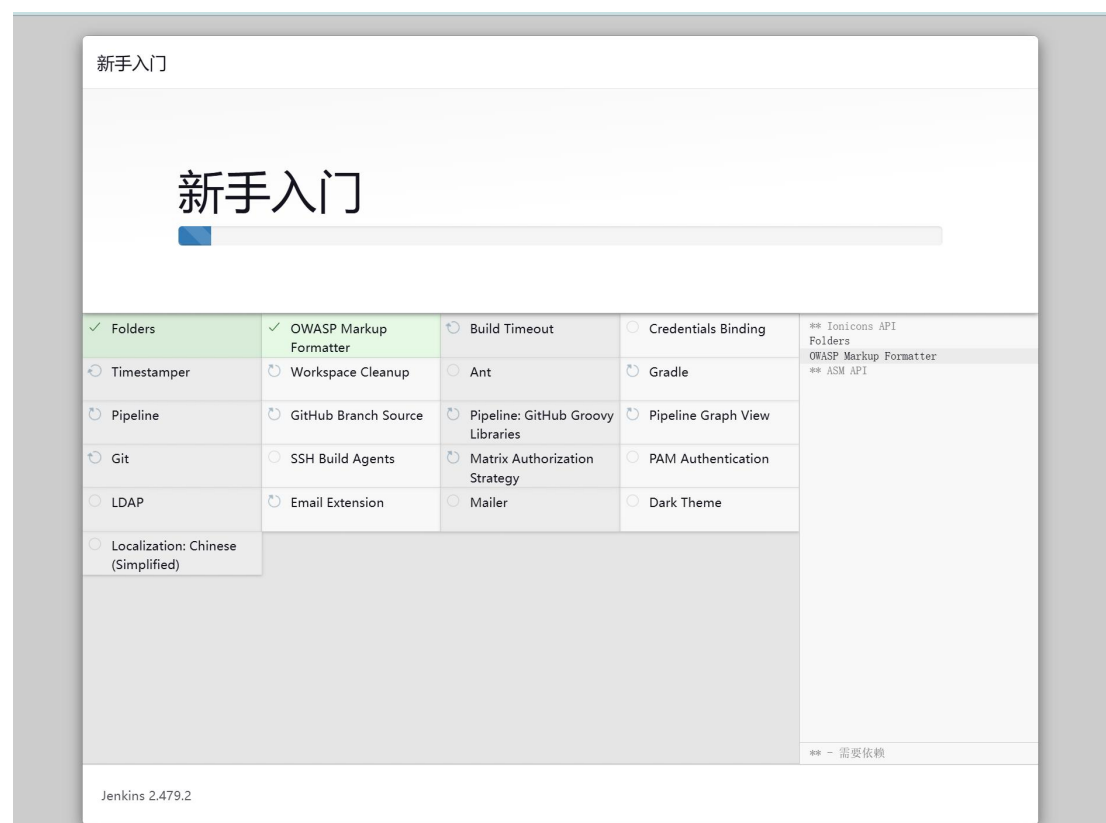
管理员密码

.....

继续

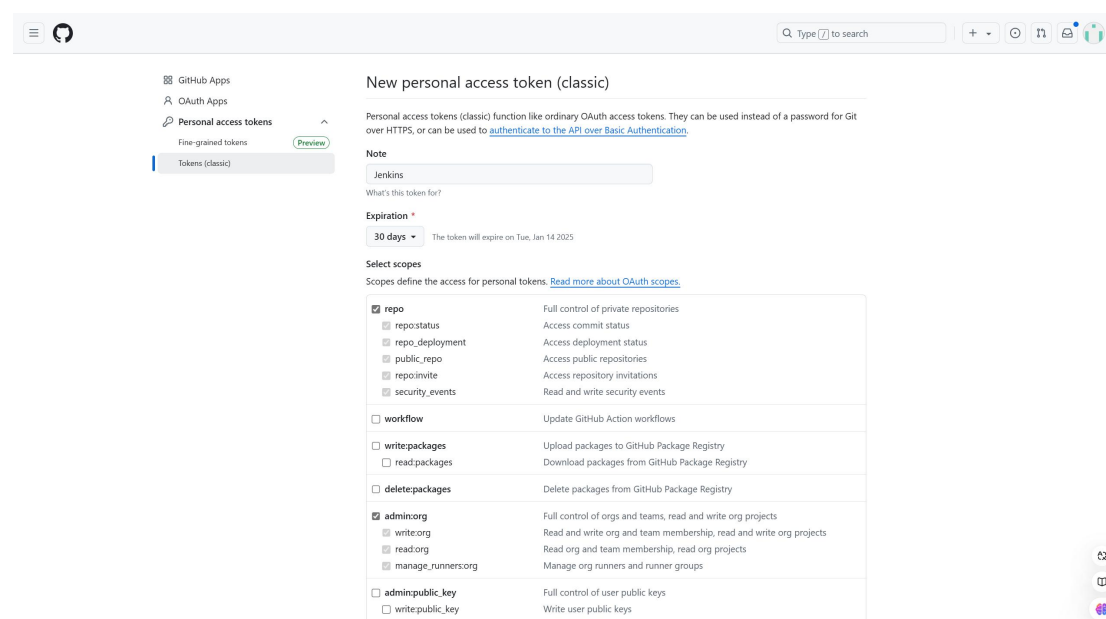
## (2) 安装推荐的插件

安装推荐的插件，特别是那些与 GitHub 集成相关的插件，以简化后续操作。



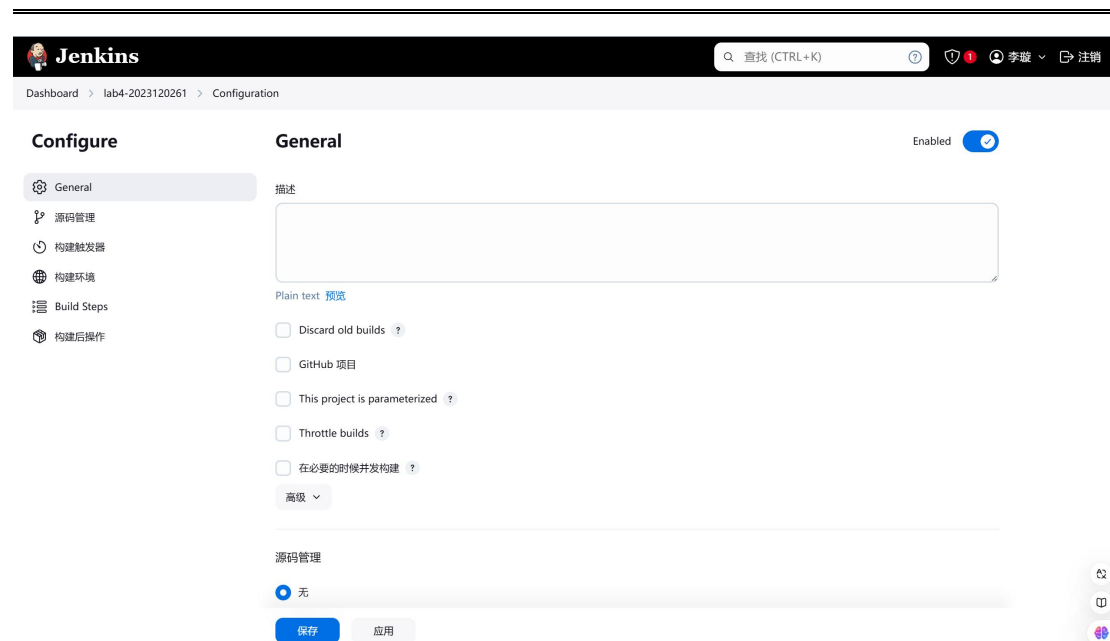
### (3) 向 Github 认证申请 Access Token

在 GitHub 上生成了一个个人访问令牌(PAT)，并在 Jenkins 中配置好，使得 Jenkins 能够访问我的 GitHub 仓库。



### (4) 构建 DevOps workflow

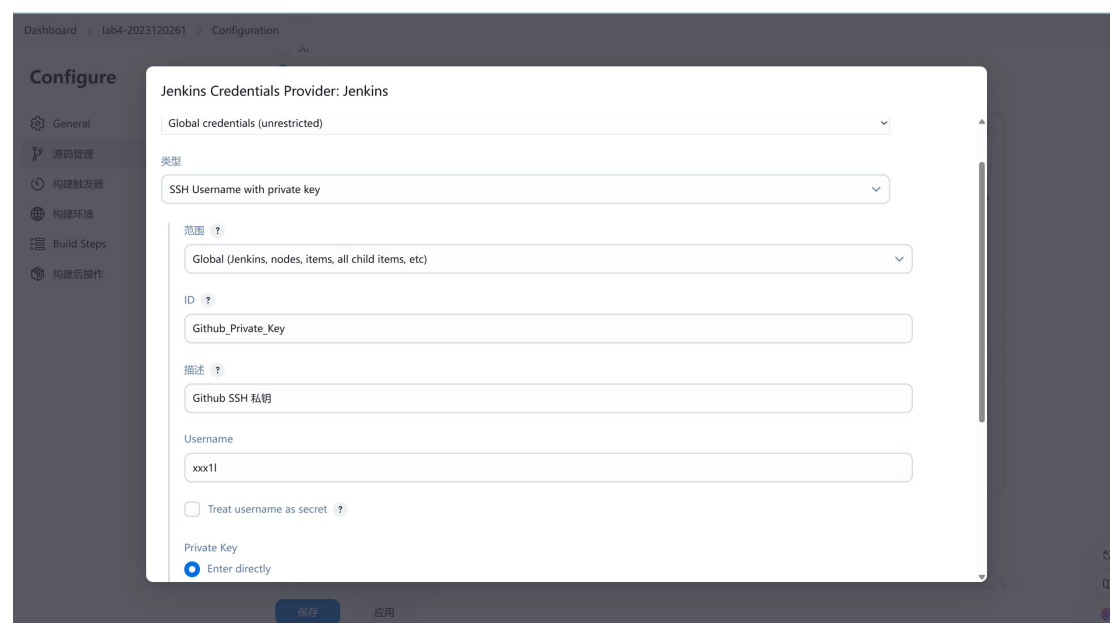
在 Jenkins 中创建一个 Freestyle Project。



### ① 源码管理

配置源码管理部分，选择从 GitHub 拉取代码，填写仓库 URL 及分支信息。

对前文设置的 Github 仓库进行访问配置，使用 SSH Username 与私钥作为凭证



### ② 构建触发器

使用 Poll SCM，定时检查仓库是否有更新，进行自动化构建，按照

图中配置，每 3 分钟检查一次仓库。

## 构建触发器

- ☐ 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15日星期日 中国标准时间 下午5:38:26; would next run at 2024年12月15日星期日 中国标准时间 下午5:41:26.

- ☐ 忽略钩子 post-commit ?

## ③ 指定 Local 项目代码所在的分支 dev

高级 ▾

Add Repository

Branches to build ?

指定分支 (为空时代表any) ?

\*/dev

Add Branch

源代码浏览器 ?

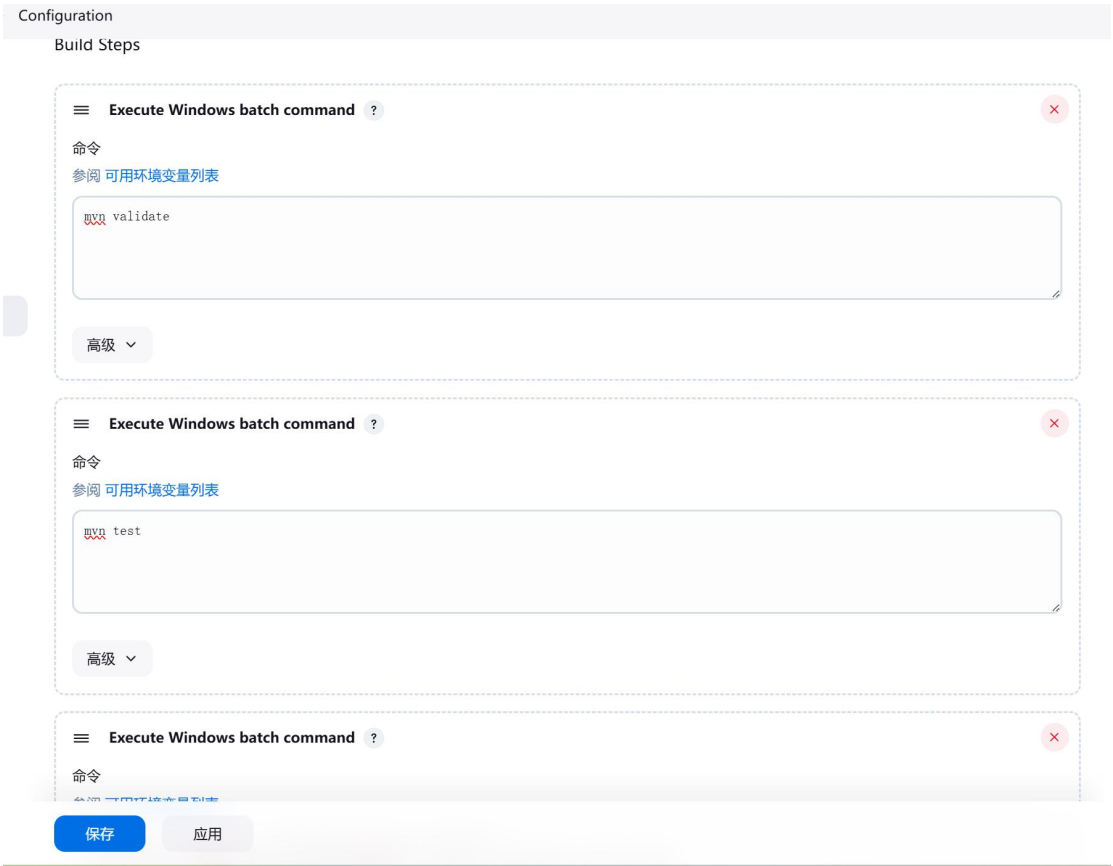
(自动) ▾

Additional Behaviours

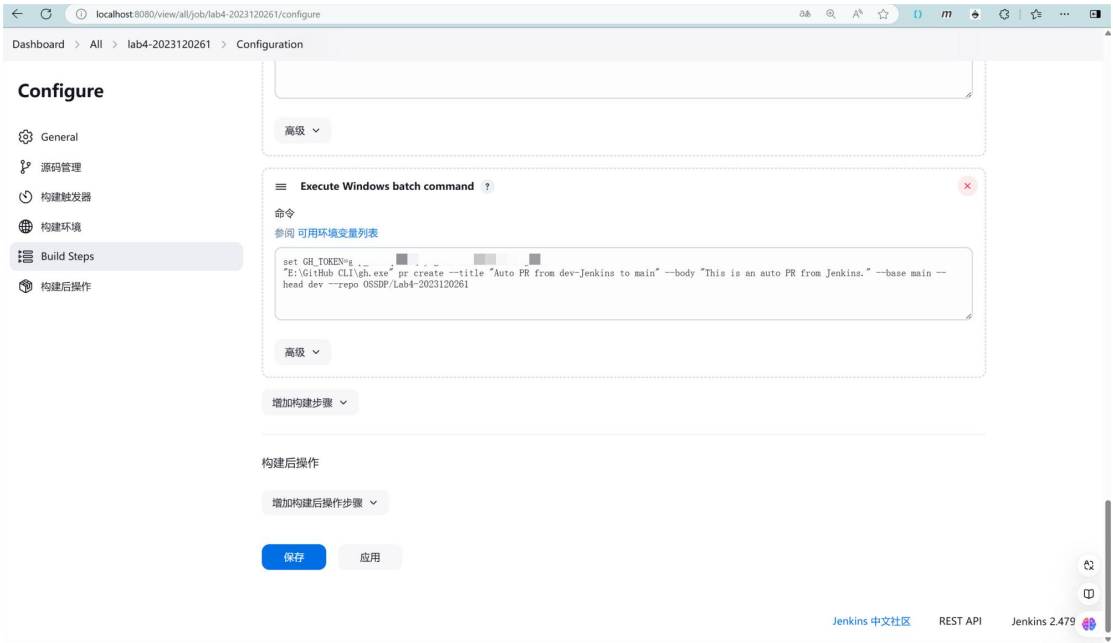
自定义 ..

## ④ 构建步骤：使用 cmd 脚本进行代码的测试与 PR 的提交

编写构建步骤，涉及到运行 **Maven** 命令来进行编译和测试，还可以包含其他任务如打包、部署等。



配置构建后动作，如发送通知、提交 PR 等。

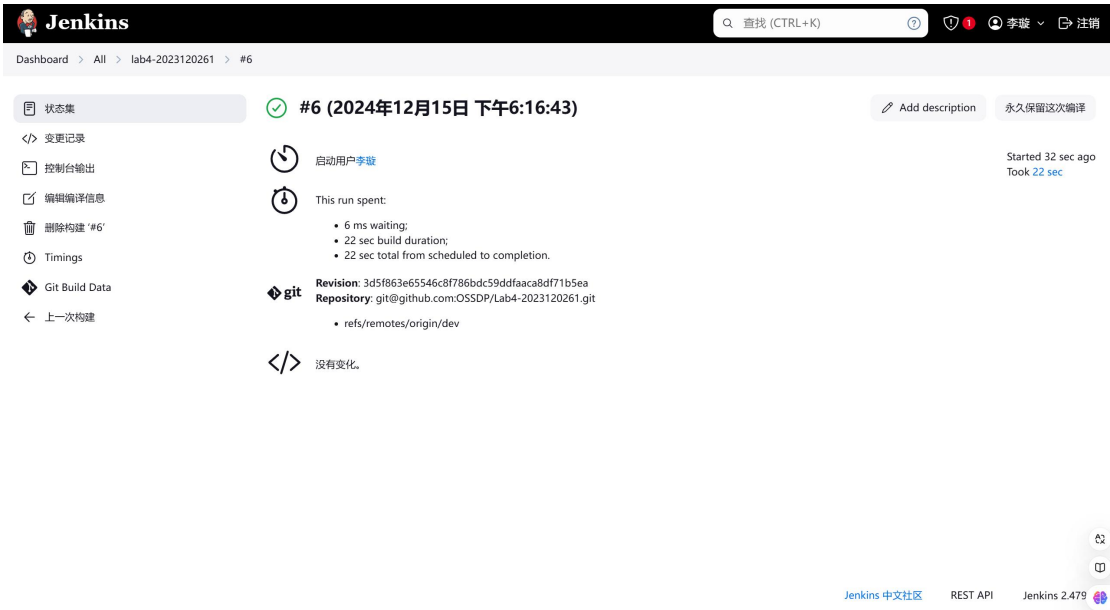


(5) 验证实验效果

- ① 将项目代码提交到选定的仓库的指定分支为 dev

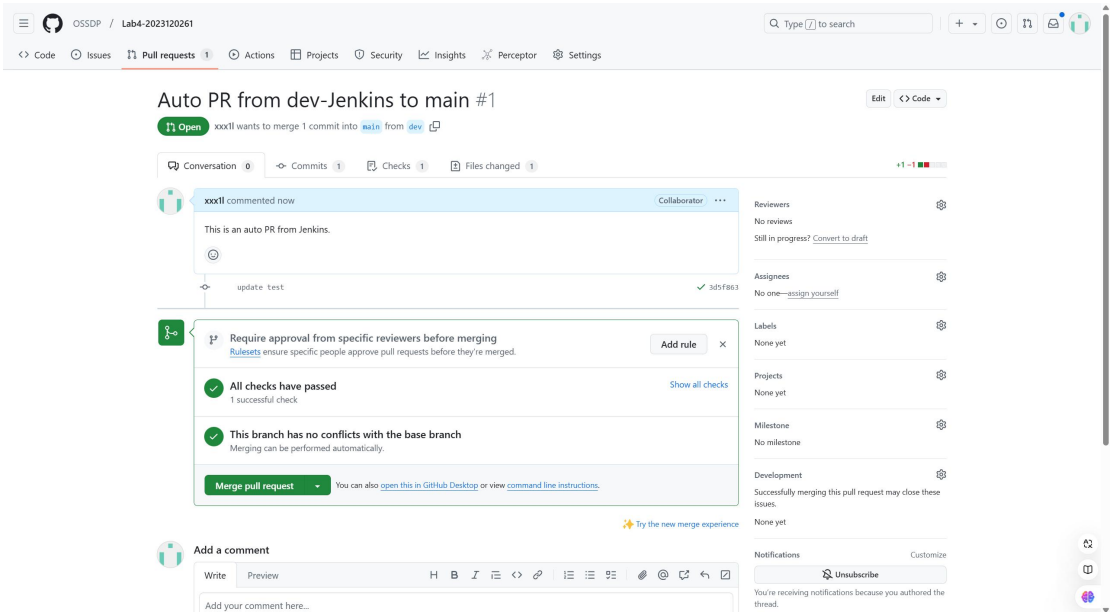
```
PS D:\Study\homework\opensource\Lab4-2023120261> git push -u origin dev
git: 'credential-manager-core' is not a git command. See 'git --help'.
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 452 bytes | 452.00 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.
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:   https://github.com/OSSDP/Lab4-2023120261/pull/new/dev
remote:
To https://github.com/OSSDP/Lab4-2023120261.git
 * [new branch]      dev -> dev
branch 'dev' set up to track 'origin/dev'.
```

② 等待定时任务进行构建流程



③ 在 Github 上查看仓库的 PR 推送成功

检查构建结果成功，PR 被正确推送到 GitHub 仓库中。



## 4 小结

在完成了这次《开源软件开发实践》课程中的 Lab4 实验后，我对 DevOps 有了更深刻的理解。通过实际操作 GitHub Actions 和 Jenkins，我不仅掌握了如何设置自动化测试和构建流程，还体验到了不同工具之间的差异。在这个过程中，我也遇到了一些挑战，比如配置环境时的小问题或是编写工作流文件时遇到的逻辑错误。这些问题让我意识到，在实际项目中，良好的规划和细致的操作是多么重要。在这次实验中，对 DevOps 有了全面的认识：以前我对 DevOps 的概念仅停留在理论层面，但这次实验让我亲身体验了从代码提交到自动构建、测试再到部署的一系列过程，真正理解了它是如何提升开发效率和产品质量的。

熟悉了 CI/CD 工具的实际应用：通过配置 GitHub Actions 和 Jenkins，我学会了使用这两个平台进行自动化流水线的设置。这对我将来从事软件开发工作非常有帮助。

提升了编程技能和解决问题的能力：创建 Maven 项目并编写 JUnit 测试用例，这些活动不仅巩固了我的编程基础，也增强了我的工程实践能力。而解决实验中遇到的问题，则锻炼了我的故障排查技巧。

加强了团队协作意识：虽然实验是个人完成的，但在查阅资料和向同学请教的过程中，我学到了很多与人沟通交流的方法，这对于未来的团队合作至关重要。