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
2022211709	曾湘临	18588602616

目 录

1 实验要求.....1
2 实验内容 1 GitHub Actions DevOps 实践.....1
3 实验内容 2 Jenkins DevOps 实践.....3
4 小结.....11

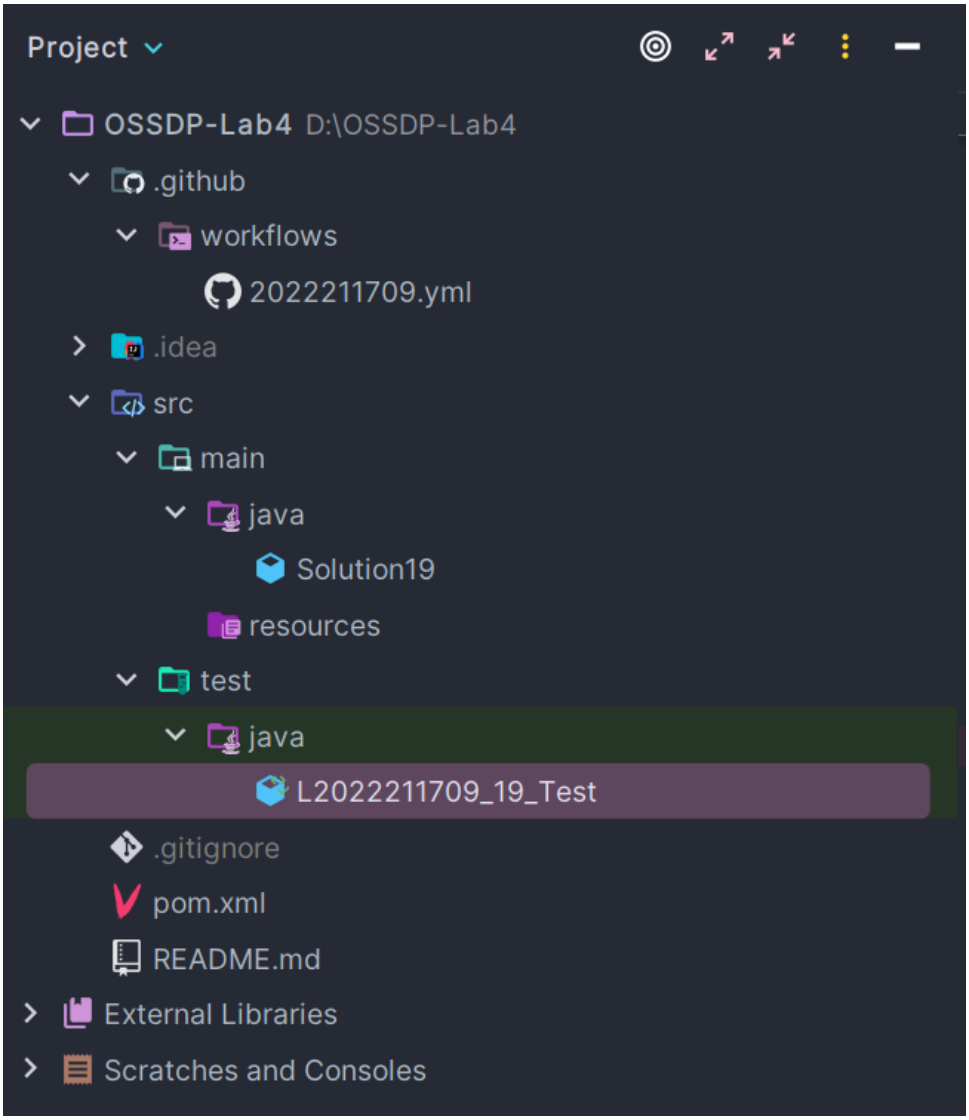
1 实验要求

本次实验训练开源软件开发中的基本 DevOps 操作，具体来说：

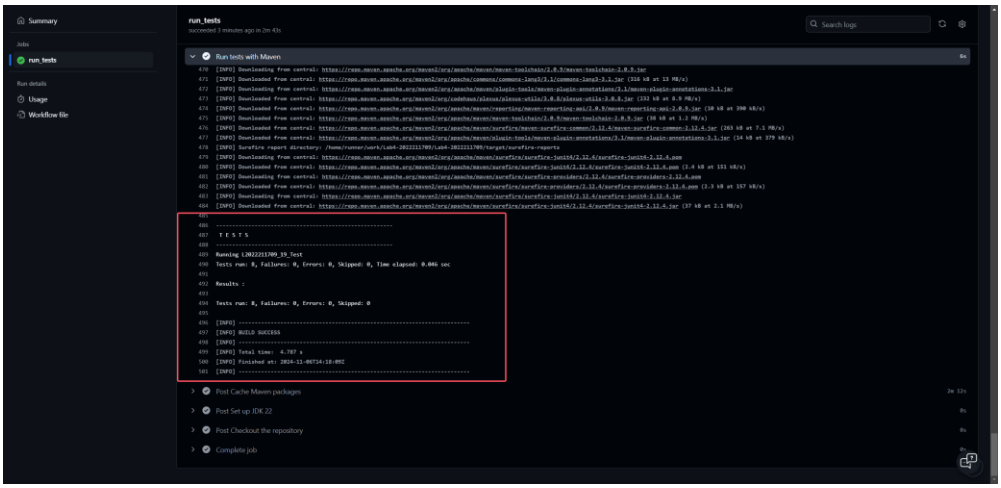
- 掌握开源软件开发中的基本 DevOps 流程和工具的使用
- 熟悉利用 GitHub Actions 进行 DevOps
- 熟悉利用 Jenkins 进行 DevOps

2 实验内容 1 GitHub Actions DevOps 实践

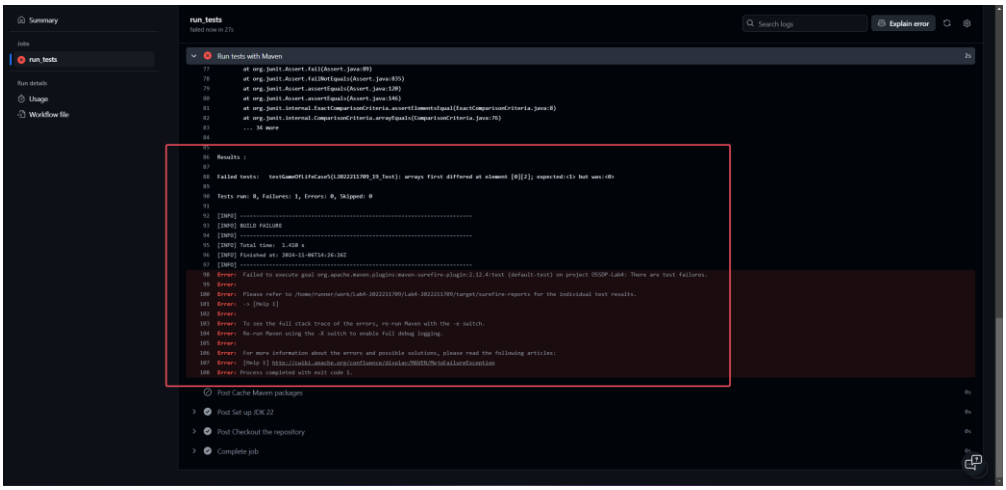
- 项目的目录结构截图



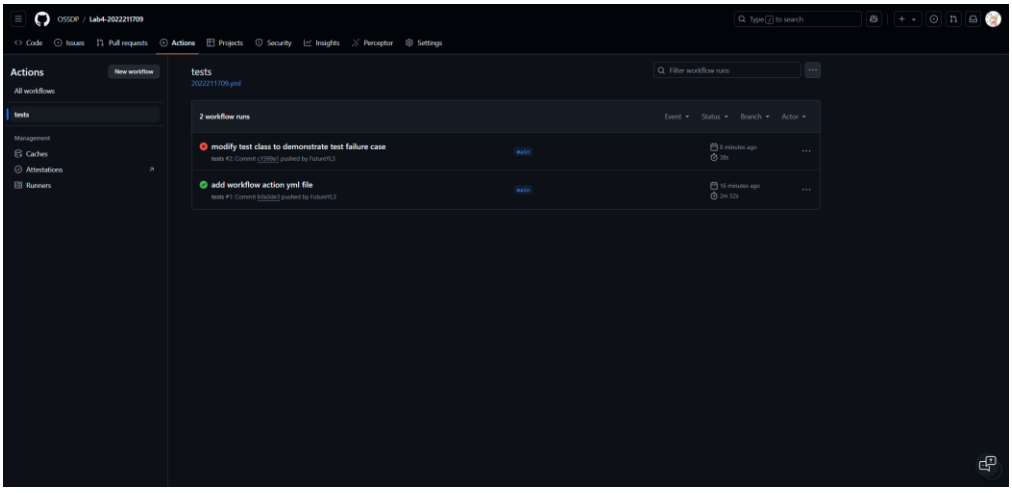
- GitHub 中的 Actions 的若干界面，包括：执行自动化测试成功界面、执行自动化测试失败界面、执行具体信息等
  执行自动化测试成功界面及执行具体信息：



修改测试类文件，使其无法通过测试后提交代码，执行自动化测试失败界面及执行信息：



总的 **Actions** 界面：



- 针对步骤八的 **YML** 代码：

```
name: tests
on:
  push:
  pull_request:
```

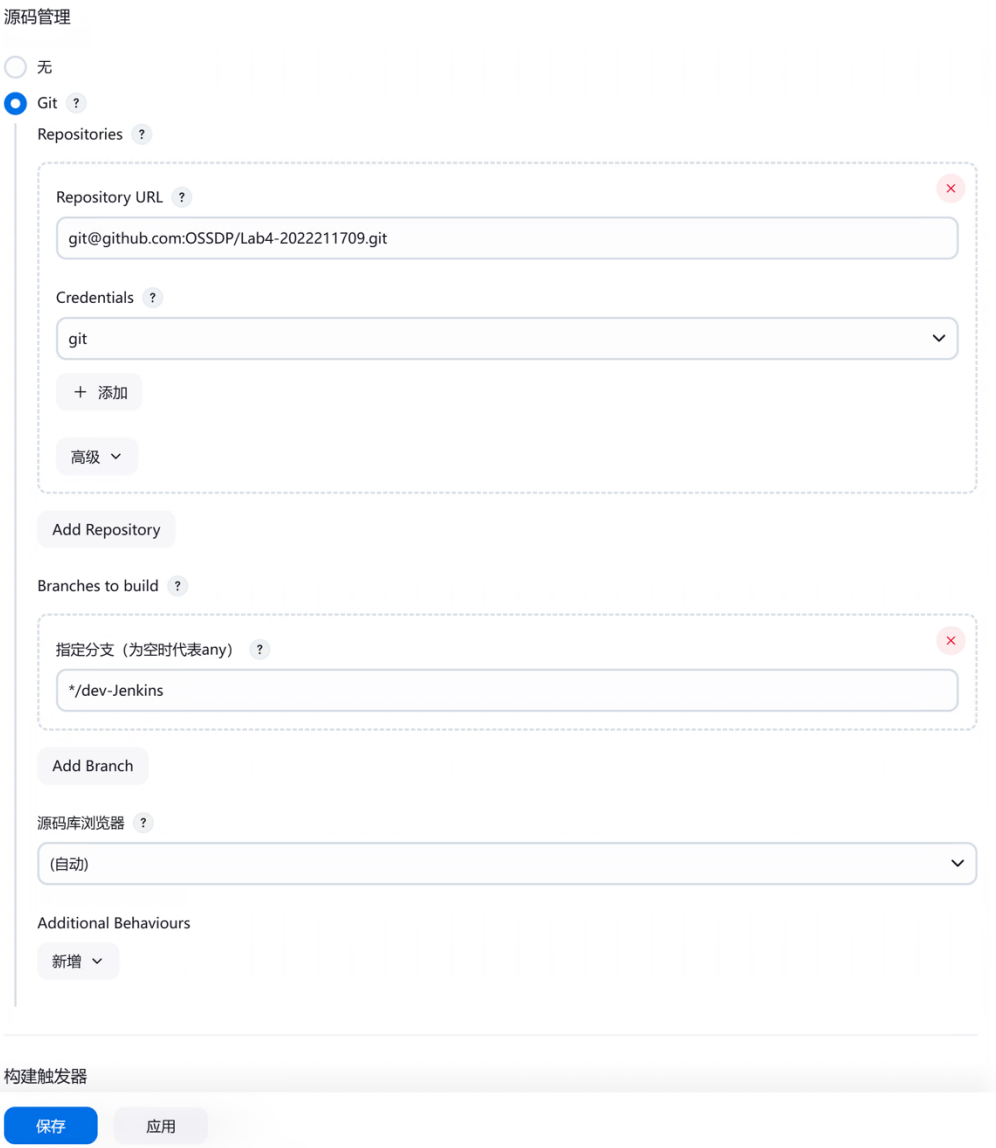
```
permissions:
  pull-requests: write
  contents: write
jobs:
  run_tests:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout the repository
        uses: actions/checkout@v2
      - name: Set up JDK 17
        uses: actions/setup-java@v1
        with:
          java-version: 17
      - name: 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 pom.xml

  merge_pr:
    needs: run_tests
    if: ${{ needs.run_tests.result == 'success' && github.event_name
== 'pull_request' }}
    runs-on: ubuntu-latest
    steps:
      - name: Merge pull request
        uses: actions/github-script@v6
        with:
          github-token: ${{ secrets.GITHUB_TOKEN }}
          script: |
            await github.rest.pulls.merge({
              owner: context.repo.owner,
              repo: context.repo.repo,
              pull_number: context.payload.pull_request.number,
            })
```

3 实验内容 2 Jenkins DevOps 实践

实验步骤六和步骤七的操作截图

步骤六：

Github 仓库设置：

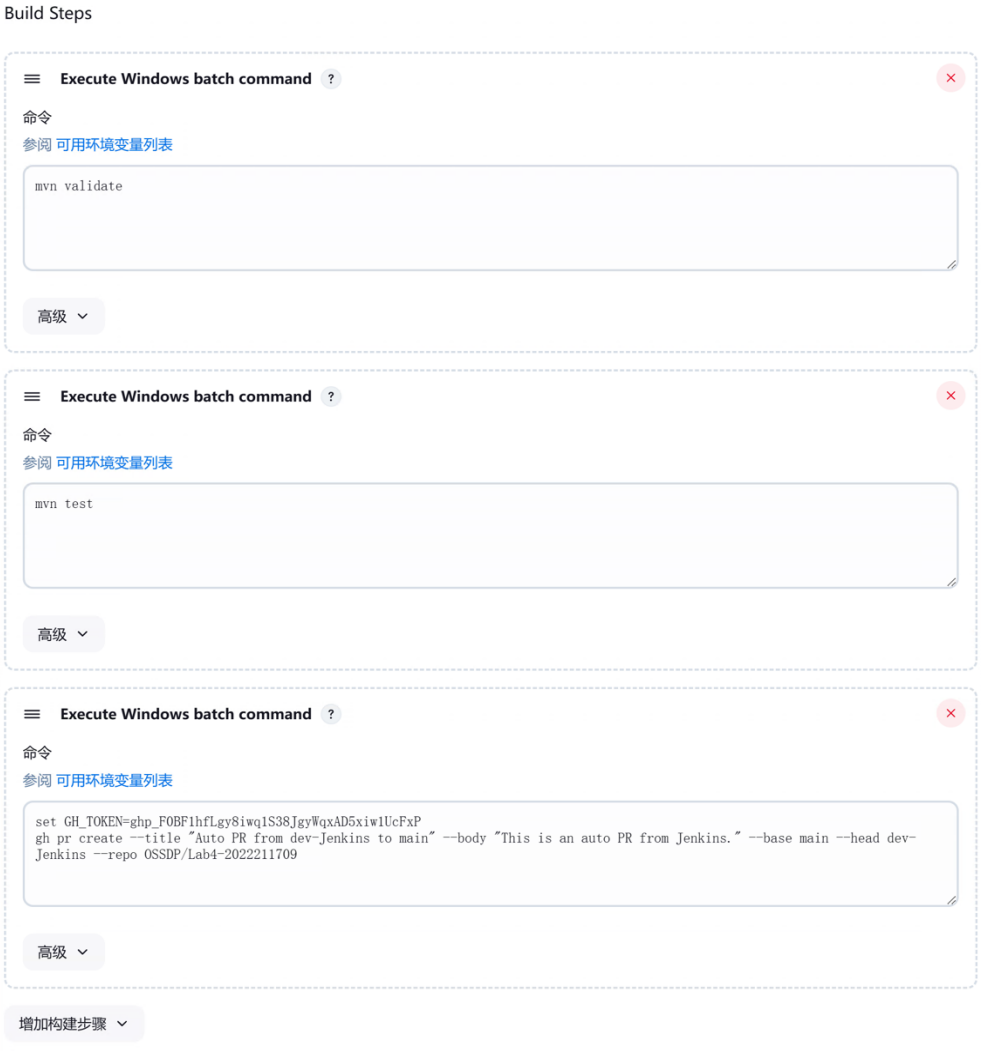


其中的凭证 Credentials 采用实验指导书中的方法配置不成功，所以采用其他办法。Credentials 中的私钥是通过本地执行命令 ssh-keygen 来生成的 ssh 私钥，公钥已配置到 GitHub 个人账户的 SSH keys 中，根据规定，使用该方法配置的 Credentials 的用户名必须为 git。这种配置方法可以通过验证访问到 GitHub 仓库。

触发器设置：

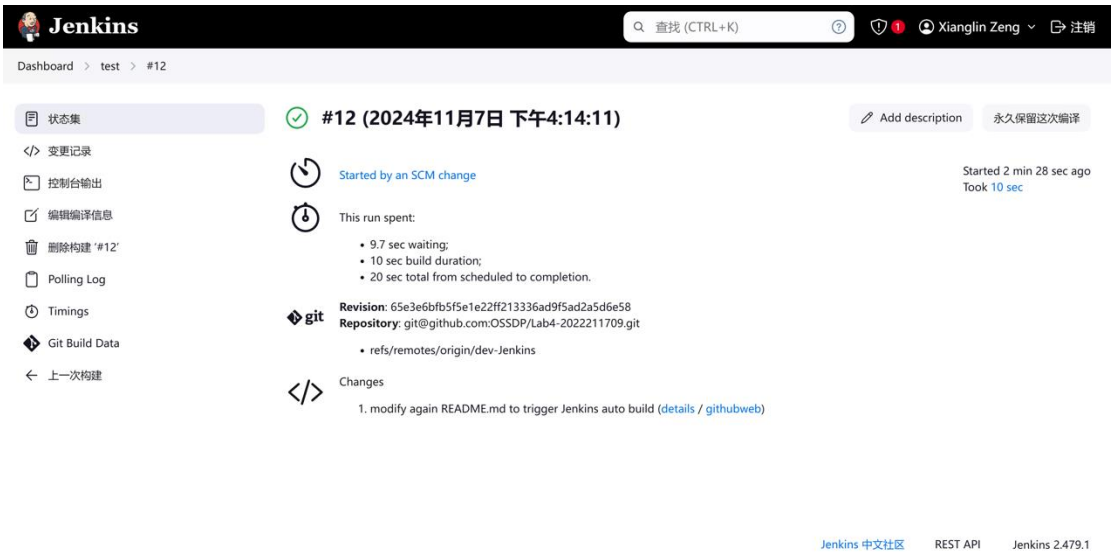
这里设置成每三分钟检查一次。

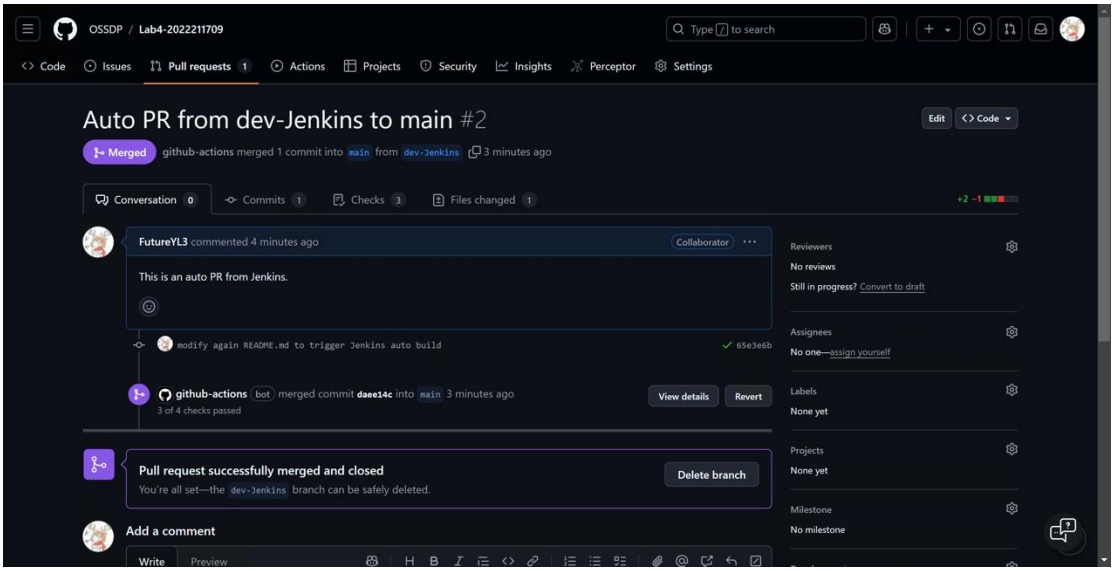
Build steps：



这里要执行的 build steps 配置为先进行 maven 的验证，然后执行测试，测试通过后设置 GitHub 的 Access Token 并向仓库发起申请从分支 dev-Jenkins 合并到分支 main 的 PR。

步骤七：





Jenkins 在检测到仓库变化后，执行了 maven 验证和测试，PR 成功经 Jenkins 提出，并通过 2.1 的步骤八（选作任务）进行 pr 的自动测试和自动 merge，成功通过测试并 merge 到了 main 分支。

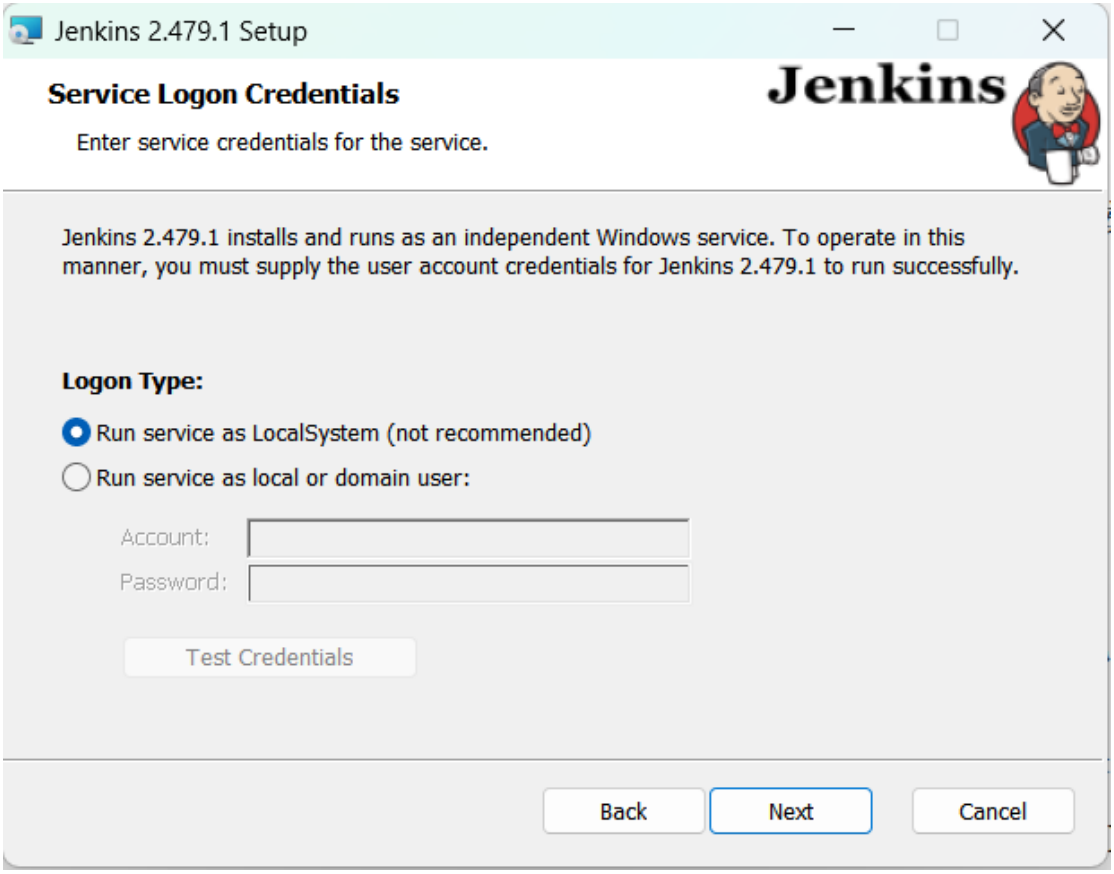
实验中各步骤结果的截图

步骤一（开发环境）：

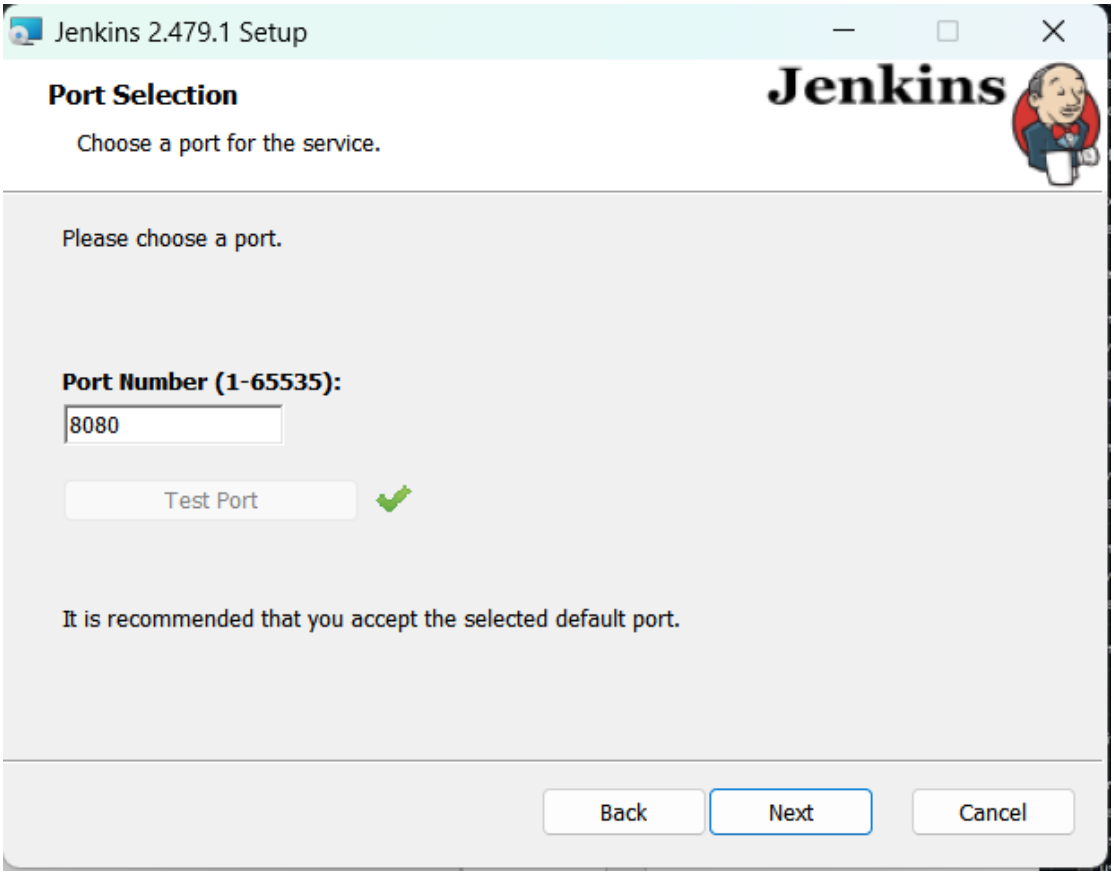
```
PS C:\Users\yl> java -version
openjdk version "17.0.12" 2024-07-16 LTS
OpenJDK Runtime Environment Corretto-17.0.12.7.1 (build 17.0.12+7-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.12.7.1 (build 17.0.12+7-LTS, mixed mode, sharing)
PS C:\Users\yl> git --version
git version 2.45.2.windows.1
PS C:\Users\yl> mvn --version
Apache Maven 3.9.8 (36645f6c9b5079805ea5009217e36f2cffd34256)
Maven home: D:\Apps\Idea\IntelliJ IDEA 2024.1.4\plugins\maven\lib\maven3
Java version: 17.0.12, vendor: Amazon.com Inc., runtime: C:\Users\yl\.jdks\corretto-17.0.12
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
PS C:\Users\yl> gh --version
gh version 2.60.1 (2024-10-25)
https://github.com/cli/cli/releases/tag/v2.60.1
PS C:\Users\yl>
```

步骤二（Jenkins 的安装和配置）：

选择 Run Service as Local System

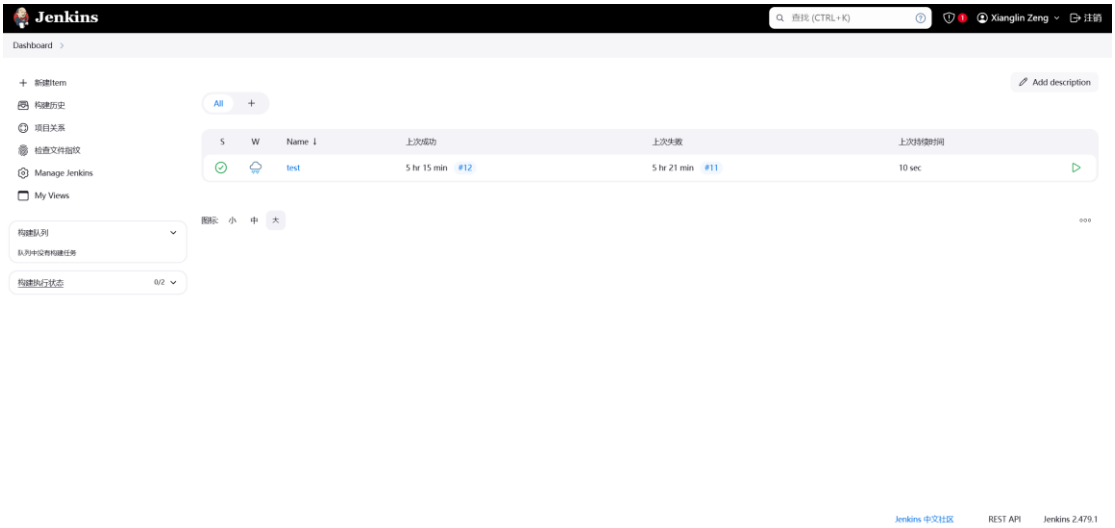


选择 8080 端口：

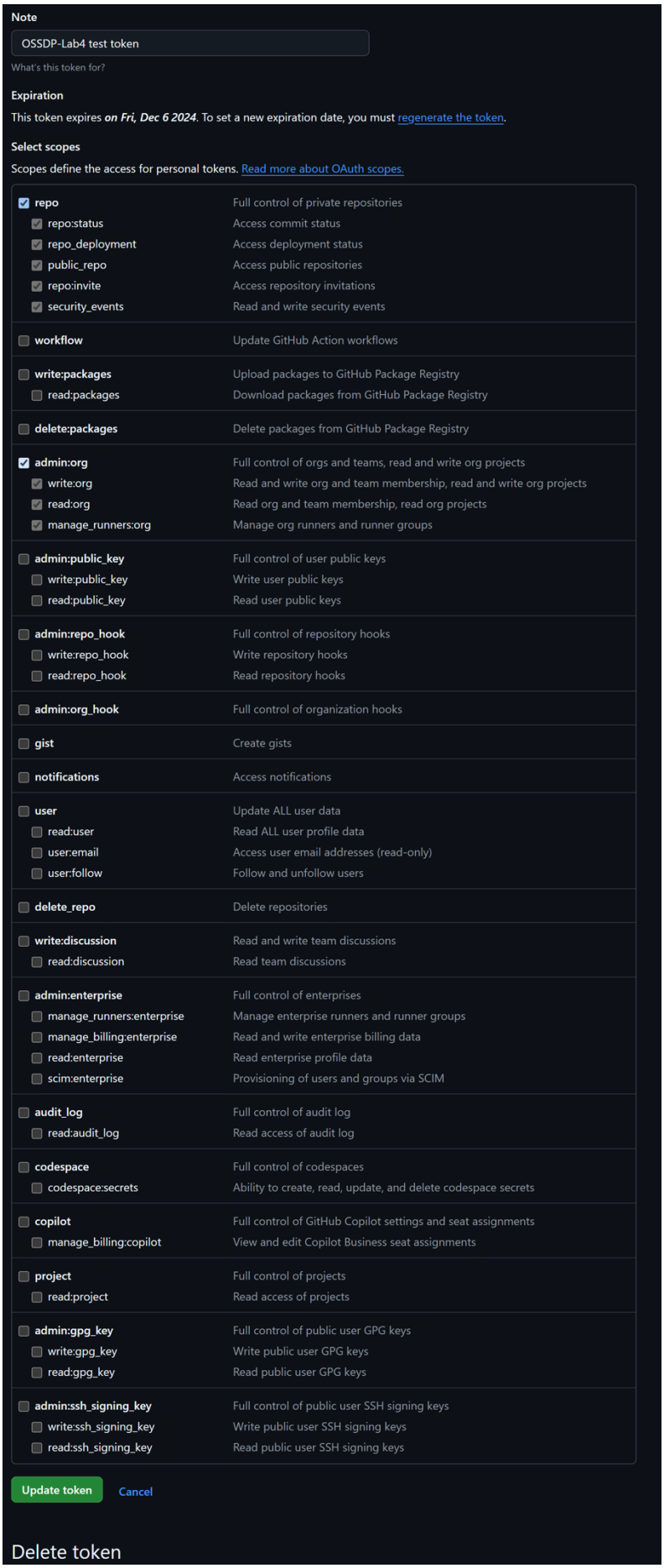


选择 JAVA_HOME 路径：

设置好初始化密码，并安装推荐的插件后，创建非 admin 用户，进入到 Jenkins 主界面：



步骤三：生成 Access Token：


Note
OSSDP-Lab4 test token
What's this token for?
Expiration
This token expires on Fri, Dec 6 2024. To set a new expiration date, you must regenerate the token.
Select scopes
Scopes define the access for personal tokens. Read more about OAuth scopes.
repo Full control of private repositories
repo:status Access commit status
repo_deployment Access deployment status
public_repo Access public repositories
repo:invite Access repository invitations
security_events Read and write security events
workflow Update GitHub Action workflows
write:packages Upload packages to GitHub Package Registry
read:packages Download packages from GitHub Package Registry
delete:packages Delete packages from GitHub Package Registry
admin:org Full control of orgs and teams, read and write org projects
write:org Read and write org and team membership, read and write org projects
read:org Read org and team membership, read org projects
manage_runners:org Manage org runners and runner groups
admin:public_key Full control of user public keys
write:public_key Write user public keys
read:public_key Read user public keys
admin:repo_hook Full control of repository hooks
write:repo_hook Write repository hooks
read:repo_hook Read repository hooks
admin:org_hook Full control of organization hooks
gist Create gists
notifications Access notifications
user Update ALL user data
read:user Read ALL user profile data
user:email Access user email addresses (read-only)
user:follow Follow and unfollow users
delete_repo Delete repositories
write:discussion Read and write team discussions
read:discussion Read team discussions
admin:enterprise Full control of enterprises
manage_runners:enterprise Manage enterprise runners and runner groups
manage_billing:enterprise Read and write enterprise billing data
read:enterprise Read enterprise profile data
scim:enterprise Provisioning of users and groups via SCIM
audit_log Full control of audit log
read:audit_log Read access of audit log
codespace Full control of codespaces
codespace:secrets Ability to create, read, update, and delete codespace secrets
copilot Full control of GitHub Copilot settings and seat assignments
manage_billing:copilot View and edit Copilot Business seat assignments
project Full control of projects
read:project Read access of projects
admin:gpg_key Full control of public user GPG keys
write:gpg_key Write public user GPG keys
read:gpg_key Read public user GPG keys
admin:ssh_signing_key Full control of public user SSH signing keys
write:ssh_signing_key Write public user SSH signing keys
read:ssh_signing_key Read public user SSH signing keys
Update token
Cancel
Delete token

步骤四、步骤五（同上一节实验，使用的测试仓库仍然为 GitHub Classroom 创建的仓库，使用上一节实验创建的 maven 项目，此处略）
步骤六、步骤七（截图如上）

4 小结

本次实验中，我深入实践了开源软件开发中的 DevOps 流程，分别使用 GitHub Actions 和 Jenkins 来完成自动化构建、测试和部署操作。通过实验，我对 DevOps 的整体流程有了更加系统的理解，也掌握了利用不同工具进行持续集成与持续交付的基本技能。实验中，我通过修改 YML 文件，完成了 GitHub Actions 的自动化测试和合并操作，探索了 Jenkins 触发 GitHub 仓库变更的自动测试与合并的过程。这些实践使我更加熟悉了各个工具的具体应用及其在自动化流程中的协作方式。

实验的收获主要体现在以下几个方面：

1. 我对 GitHub Actions 的工作流程和 Jenkins 的配置方法有了更加深刻的认识；
2. 实验中遇到的一些问题，例如 Jenkins 中凭证配置不成功等，让我学会了如何灵活调整方案来解决实际问题，并增强了我对 DevOps 工具配置的熟练度。
3. 通过对 GitHub Actions 和 Jenkins 的操作对比，我对不同工具在实际开发中的适用场景有了更清晰的认知。

对本次实验的建议主要有以下几点：

1. 实验中 Jenkins 的配置相对繁琐，尤其是凭证的配置过程，建议在实验指导中提供更为详尽的配置步骤和可能的排错方法。
2. 在 GitHub Actions 的操作过程中，YML 文件的编写容易出现语法错误，建议在实验指导中给出一些常见的错误及其解决方案。