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
2022112831	蔡志宇	phantasia_march@outlook.com

目 录

[文档全部完成之后，请更新上述区域]

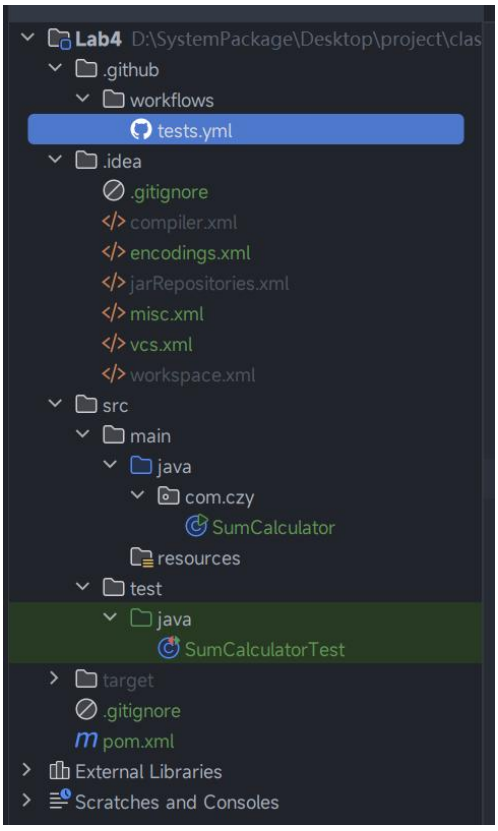
1 实验要求

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

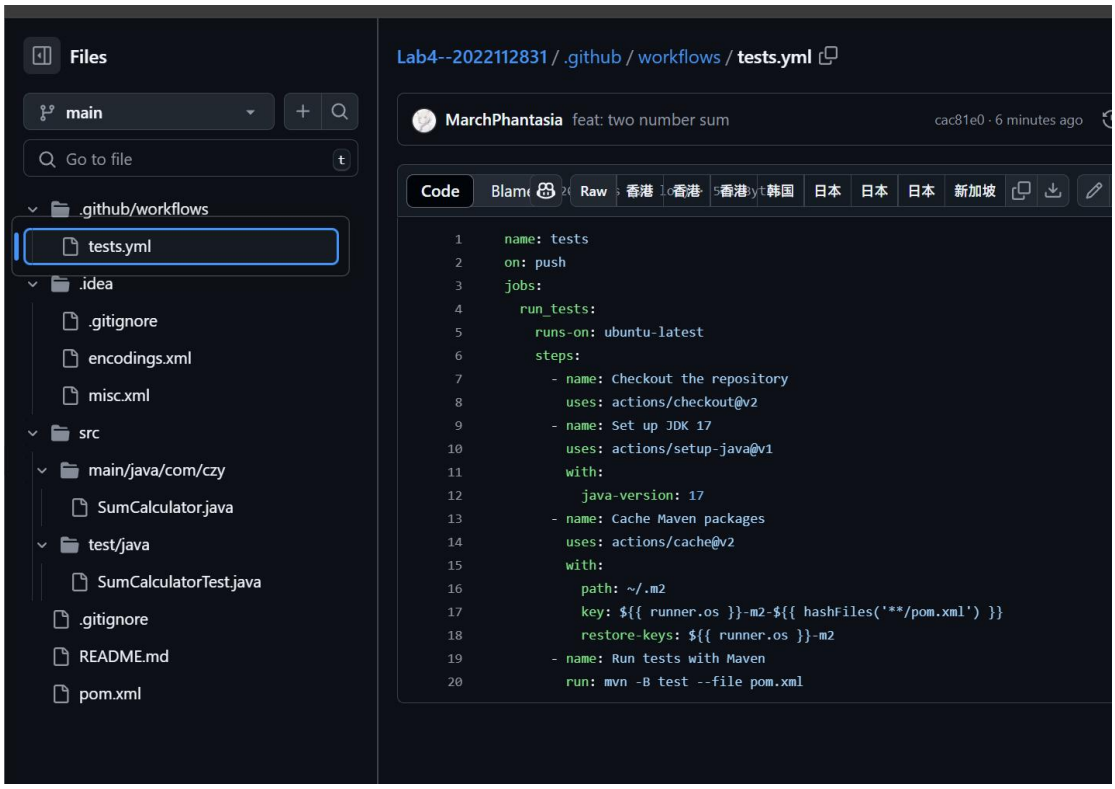
2 实验内容 1 Github Actions DevOps 实践

- **给出项目的目录结构截图，文件管理器或编程 IDE 的界面均可。**

下图是当前项目的目录结构，实现了一个实现任意两数相加求和的简单程序。

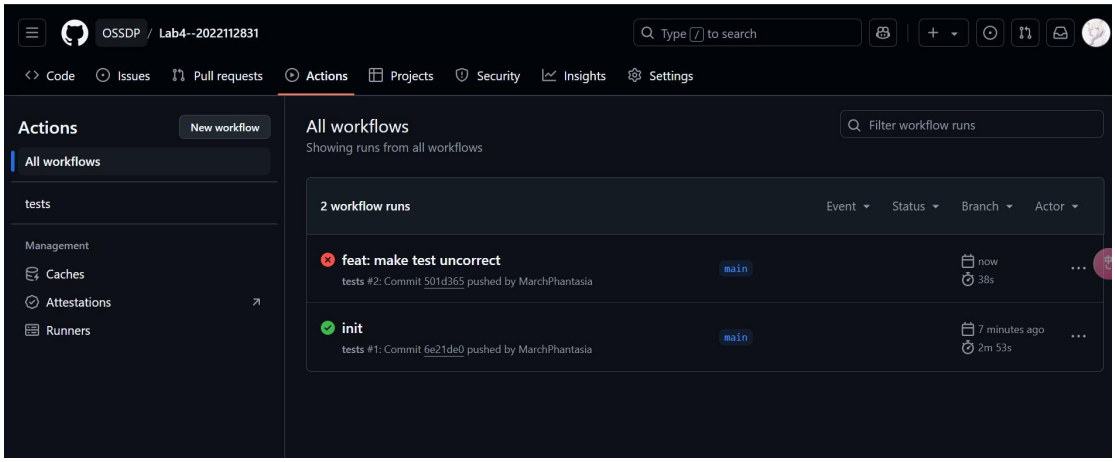


稍后将仓库代码 push 到 github 中，查看当前的仓库，以及设置的 workflow 的 yml 文件（如图所示）。

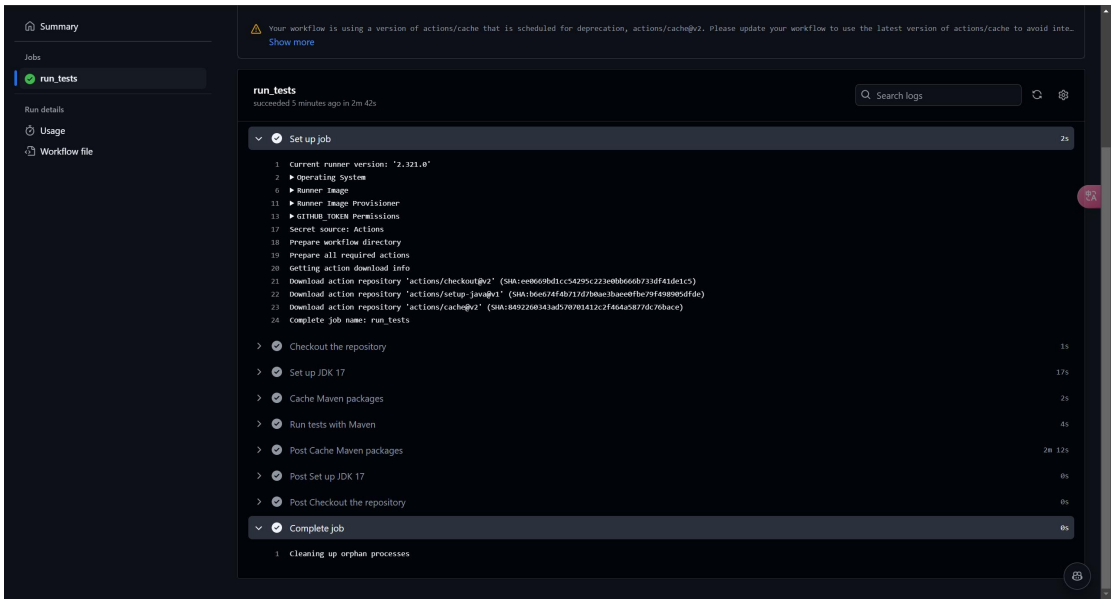


- **Github 中的 Actions 的若干界面，包括：执行自动化测试成功界面、执行自动化测试失败界面、执行具体信息等，类似实验指导书中的示例界面，以证明完成了实验。**

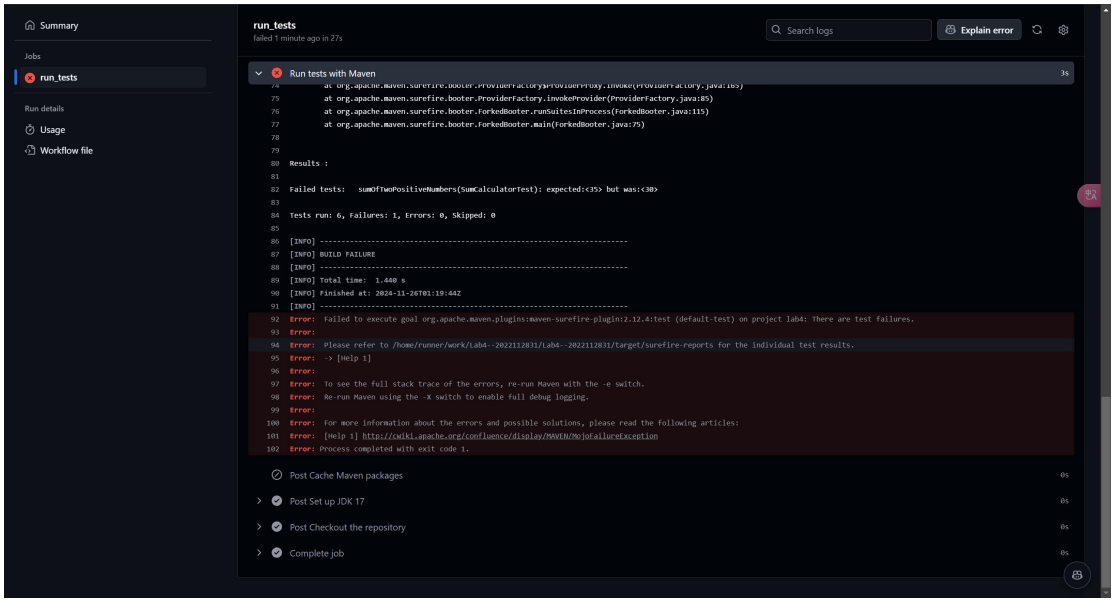
这是自动化测试执行成功、以及失败的页面



这是成功的 action 中的具体信息



这是失败的 action 中的具体信息



- **给出针对步骤八的 YML 代码（如果完成了）。**

这个工作流在 PR 打开、同步或重新打开时触发，运行测试后，如果测试通过，将自动合并 PR。请确保在仓库的 Settings > Secrets 中添加 GITHUB_TOKEN。

```
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 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: Validate Maven installation
        run: mvn -v

      - name: Run tests with Maven
        run: mvn -B clean test --file pom.xml

  auto_merge:
    needs: run_tests
    runs-on: ubuntu-latest
    if: github.event_name == 'pull_request' && github.event.action == 'opened'
    steps:
      - name: Checkout the repository
        uses: actions/checkout@v2

      - name: Merge pull request
        uses: peter-evans/merge@v2
        with:
          repo-token: ${{ secrets.GITHUB_TOKEN }}
          merge-method: squash
          commit-message: merge
```

或者单独创建一个 github action，在相关人员进行 review 后自动实现代码的 merge

参考链接 https://github.com/mxsm/rocketmq-rust/blob/main/.github/workflows/automerge.yml

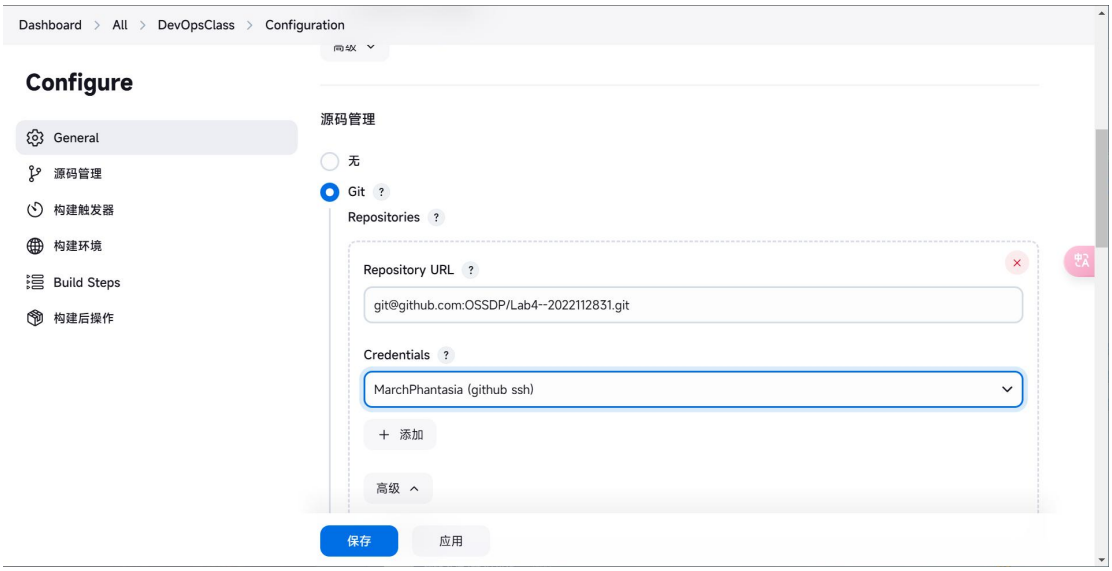
```
name: Auto Merge PR
on:
  pull_request_target:
    types:
```

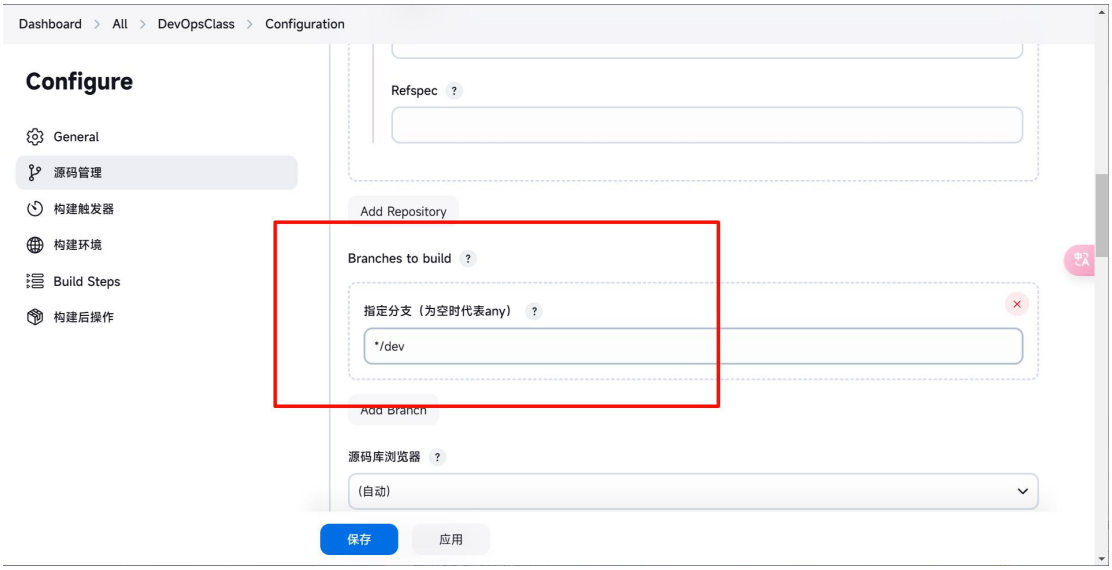
```
      - labeled
      - unlabeled
      - synchronize
      - opened
      - edited
      - ready_for_review
      - reopened
      - unlocked
  pull_request_review:
    types:
      - submitted
  check_suite:
    types:
      - completed
  status: { }
jobs:
  automerge:
    runs-on: ubuntu-latest
    steps:
      - name: automerge
        uses: "pascalgn/automerge-action@v0.16.4"
        env:
          GITHUB_TOKEN: "${{ secrets.BOT_TOKEN }}"
          MERGE_LABELS: "approved,auto merge,!ready to review,!waiting-review"
          MERGE_METHOD: "squash"
          MERGE_COMMIT_MESSAGE: "automatic"
          MERGE_FORKS: "true"
          MERGE_RETRIES: "50"
          MERGE_RETRY_SLEEP: "10000"
          MERGE_REQUIRED_APPROVALS: "1"
          UPDATE_METHOD: "rebase"
```

3 实验内容 2 Jenkins DevOps 实践

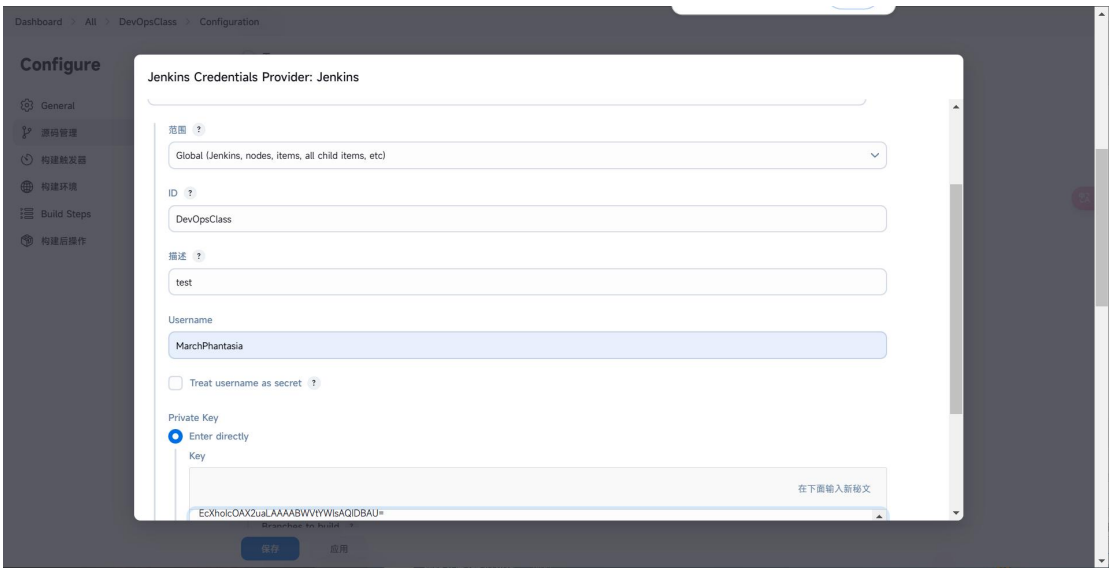
（一）构建 DevOps workflow

1) 点击新建 Item 以创建新的构建流程，任务类型选择 Freestyle Project。
2) 源码管理：在该部分我们需要对前文设置的 Github 仓库进行访问配置。

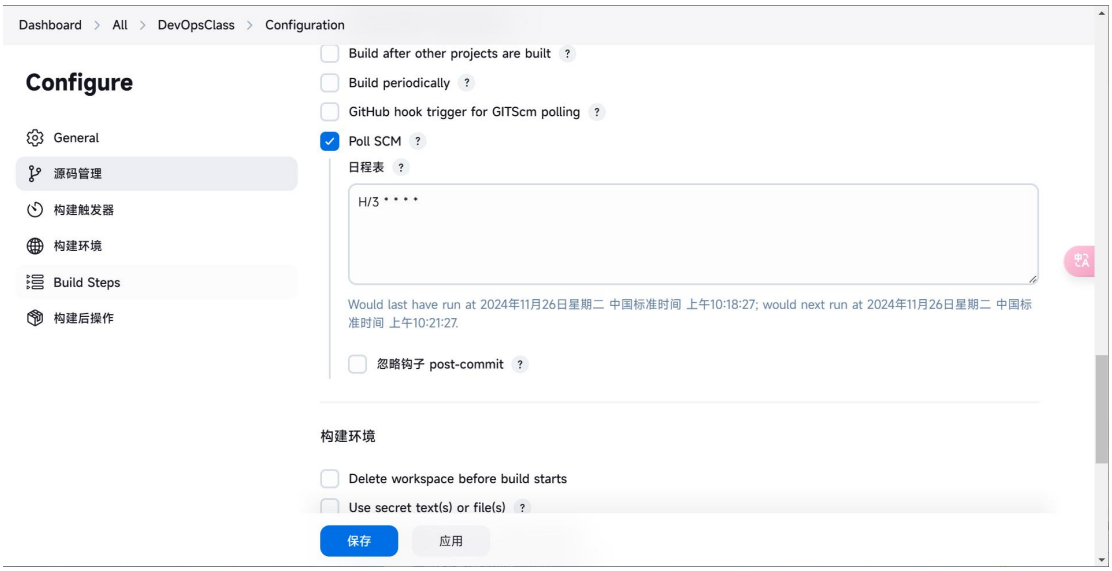




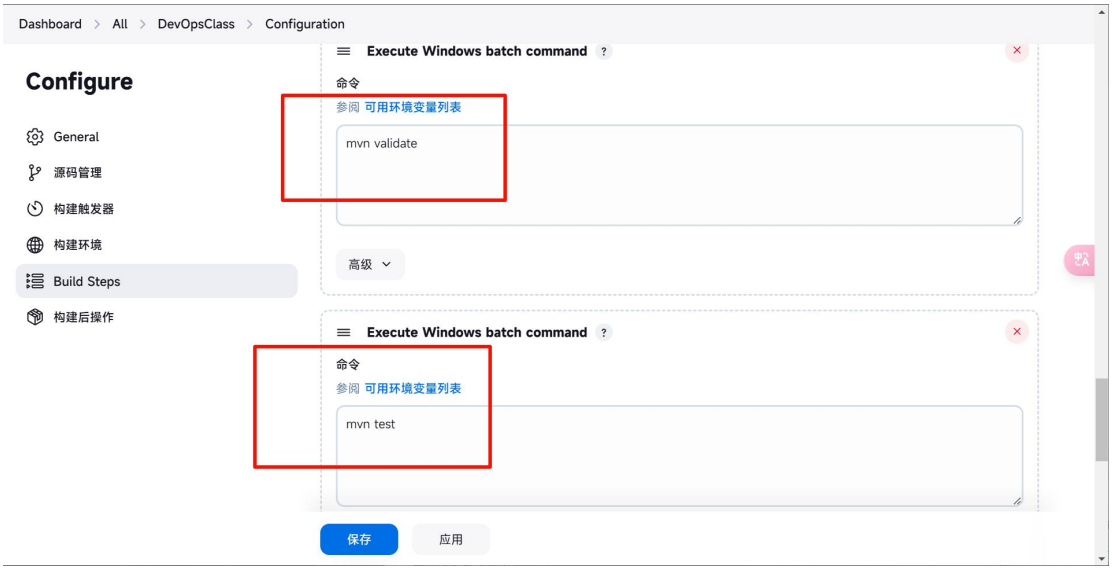
3） 下面是其中关于 SSH 的认证凭证的设置

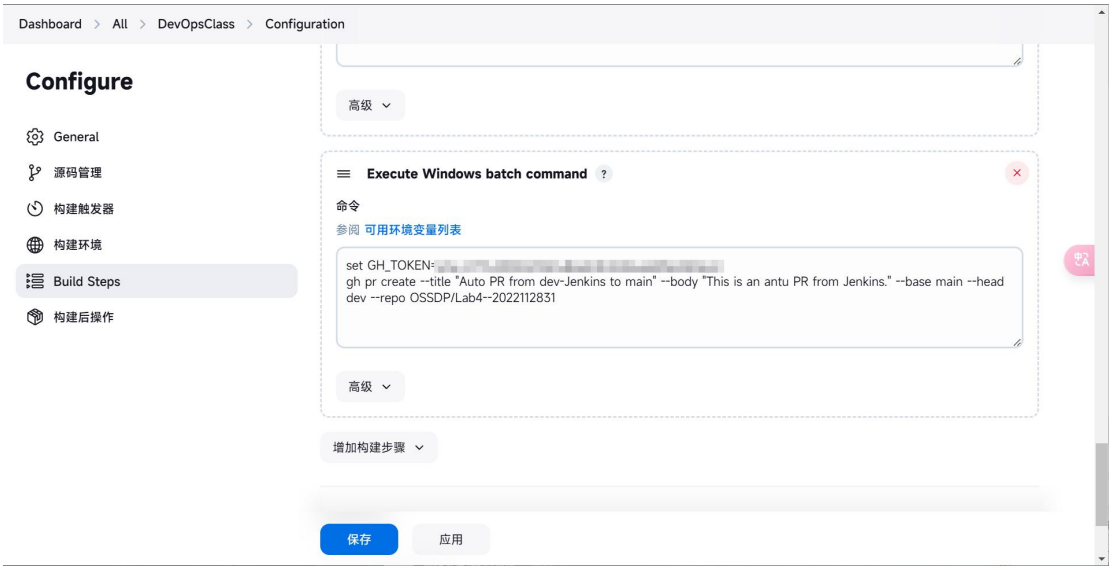


4） 构建触发器



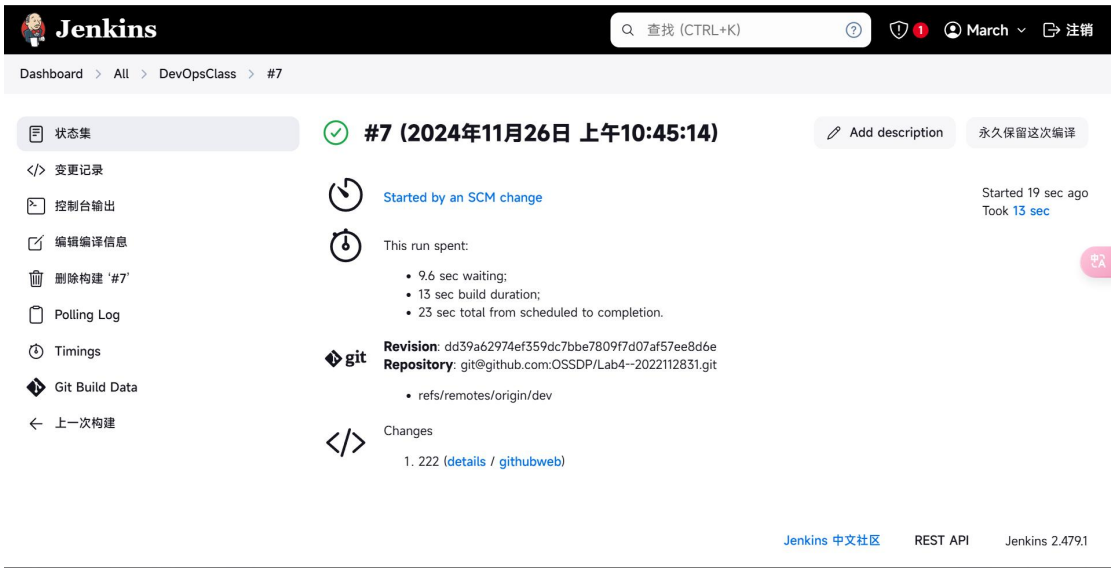
构建步骤的设置

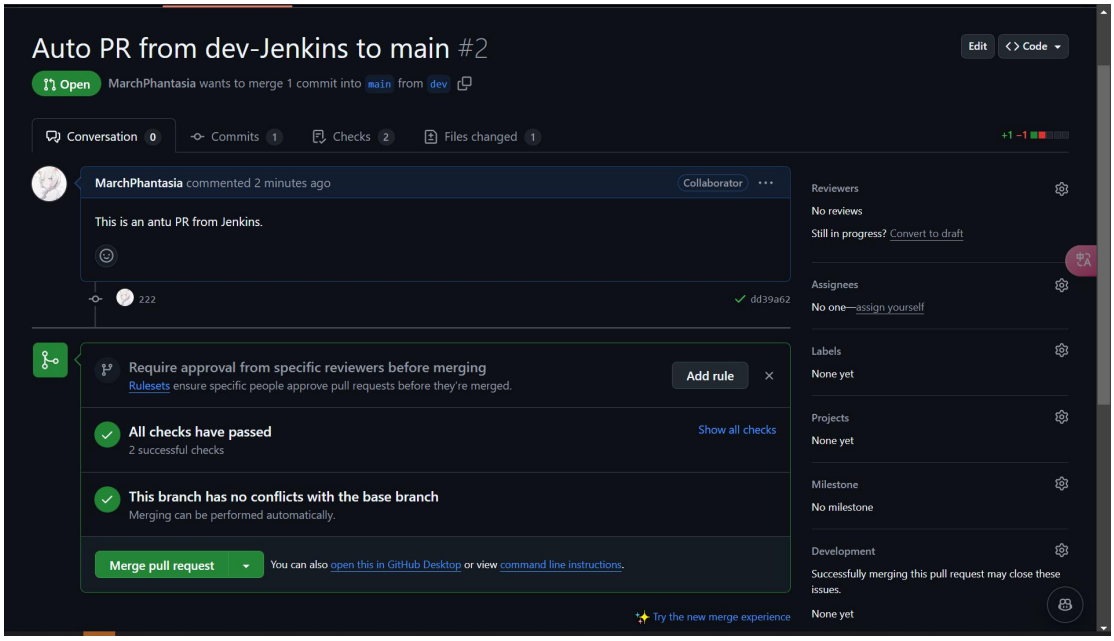




（二）验证实验效果

在本地修改 dev 分支的代码，并提交到远程 dev 分支后，通过 jenkins 的定时任务，可以看见，代码正确的被构建，并且 Jenkins 自动提交了一个合并到主分支的 PR。





4 小结

本次实验通过实际操作，深入了解了开源软件开发中的 DevOps 流程和工具的使用。在 Github Actions 的实践中，我成功配置了自动化测试工作流，并通过修改测试用例验证了自动化测试的成功和失败情况。这一过程让我深刻体会到自动化测试在软件开发中的重要性，它不仅提高了测试效率，还能及时发现代码中的问题。

在 Jenkins 的实践中，我学会了如何配置 Jenkins 进行自动化构建和 PR 提交。通过定时任务的设置，Jenkins 能够自动检查仓库更新并执行构建流程，这大大简化了开发流程，减少了人工干预的需要。实验结果表明，Jenkins 能够成功构建项目并自动提交 PR，验证了自动化 CI/CD 流程的可行性和有效性。

实验手册很详细，感谢老师和助教老师，在软件学院的课程乃至计算学部的所有课程里堪称是实验手册的典范。