

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
2022112842	周滨旭	2281632374@qq.com/18655357985

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	7
4 小结	12

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

本次实验在 GitHub Classroom 中的 URL 地址为：
<https://github.com/OSSDP/Lab4-2022112842>。

2 实验内容 1 Github Actions DevOps 实践

在本地新建一个 maven 项目，并引入 junit 依赖。下面是该项目的 pom.xml 中的全部内容。

```
<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>_</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>_</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

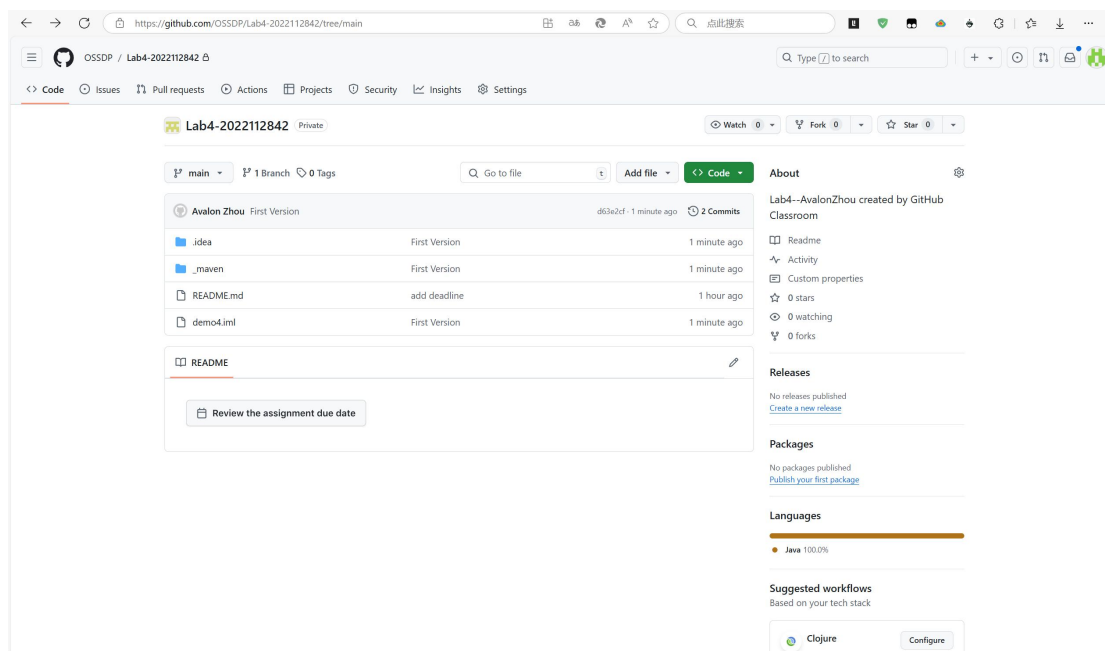
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter</artifactId>
      <version>RELEASE</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

将 Lab2 中的 Debug 的程序（修改正确后的）和编写的单元测试文件，导入到项目中。

```
public class Solution18 {  
    5 个用法  
    public static int[] productExceptSelf(int[] nums) {  
        int length = nums.length;  
  
        // L 和 R 分别表示左右两侧的乘积列表  
        int[] L = new int[length];  
        int[] R = new int[length];  
  
        int[] answer = new int[length];  
  
        // L[i] 为索引 i 左侧所有元素的乘积  
        // 对于索引为 '0' 的元素，因为左侧没有元素，所以 L[0] = 1  
        L[0] = 1;  
        for (int i = 1; i < length; i++) {  
            L[i] = nums[i - 1] * L[i - 1];  
        }  
  
        // R[i] 为索引 i 右侧所有元素的乘积  
        // 对于索引为 'length-1' 的元素，因为右侧没有元素，所以 R[length-1] = 1  
        R[length - 1] = 1;  
        for (int i = length - 2; i >= 0; i--) {  
            R[i] = nums[i + 1] * R[i + 1];  
        }  
  
        // 对于索引 i，除 nums[i] 之外其余各元素的乘积就是左侧所有元素的乘积乘以右侧所有元素的乘积  
        for (int i = 0; i < length; i++) {  
            answer[i] = L[i] * R[i];  
        }  
  
        return answer;  
    }  
}
```

```
class Solution18Test {
    //测试数组中只含有一个数的情况
    @Test
    void test_single() {
        int[] nums1 = {9};
        int[] ret1 = {1};
        assertEquals(ret1, Solution18.productExceptSelf(nums1));
    }
    //测试数组中含有超过10个数的情况
    @Test
    void test_long() {
        int[] nums2 = {2,5,7,9,8,4,6,7,2,3,4,5};
        int[] ret2 = {50803200,20321280,14515200,11289600,12700800,25401600,
            16934400,14515200,50803200,33868800,25401600,20321280};
        assertEquals(ret2, Solution18.productExceptSelf(nums2));
    }
    //测试数组中含0的情况
    @Test
    void test_zero() {
        int[] nums3 = {1,0,2,4};
        int[] ret3 = {0,8,0,0};
        assertEquals(ret3, Solution18.productExceptSelf(nums3));
    }
    //测试数组中含负数的情况
    @Test
    void test_negative() {
        int[] num4 = {4,-6,-2,-1,3};
        int[] ret4 = {-36,24,72,144,-48};
        assertEquals(ret4, Solution18.productExceptSelf(num4));
    }
    //测试最普通的情况
    @Test
    void test_normal() {
        int[] nums5 = {4,2,3,7,6};
        int[] ret5 = {252,504,336,144,168};
        assertEquals(ret5, Solution18.productExceptSelf(nums5));
    }
}
```

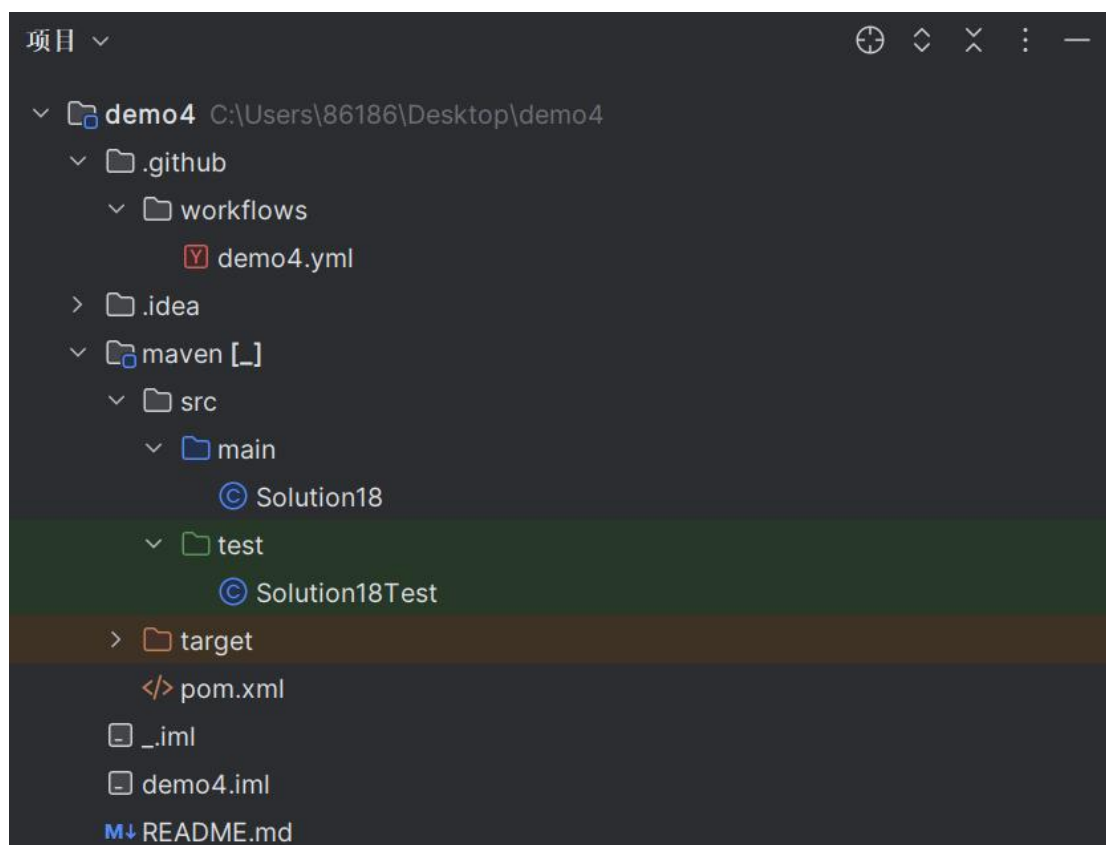
提交这个 maven 项目到实验给出的仓库中去。



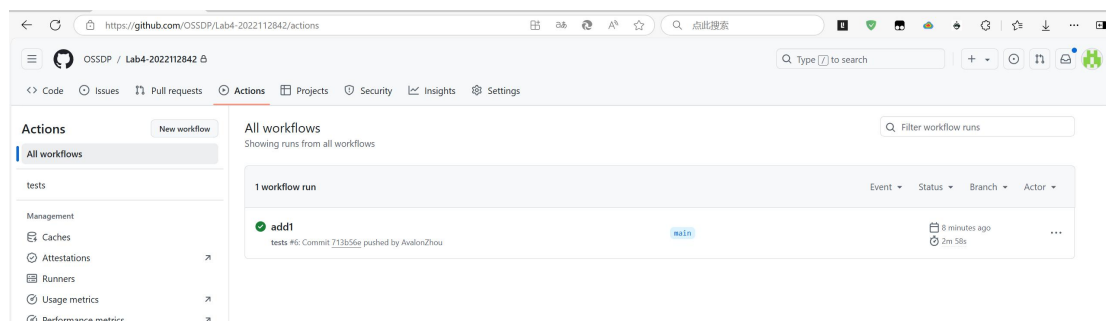
在项目根目录下新建文件夹.github/workflows/, 然后新建 demo4.yml 文件, y 内容如下。

```
name: tests
on: push
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 maven/pom.xml
```

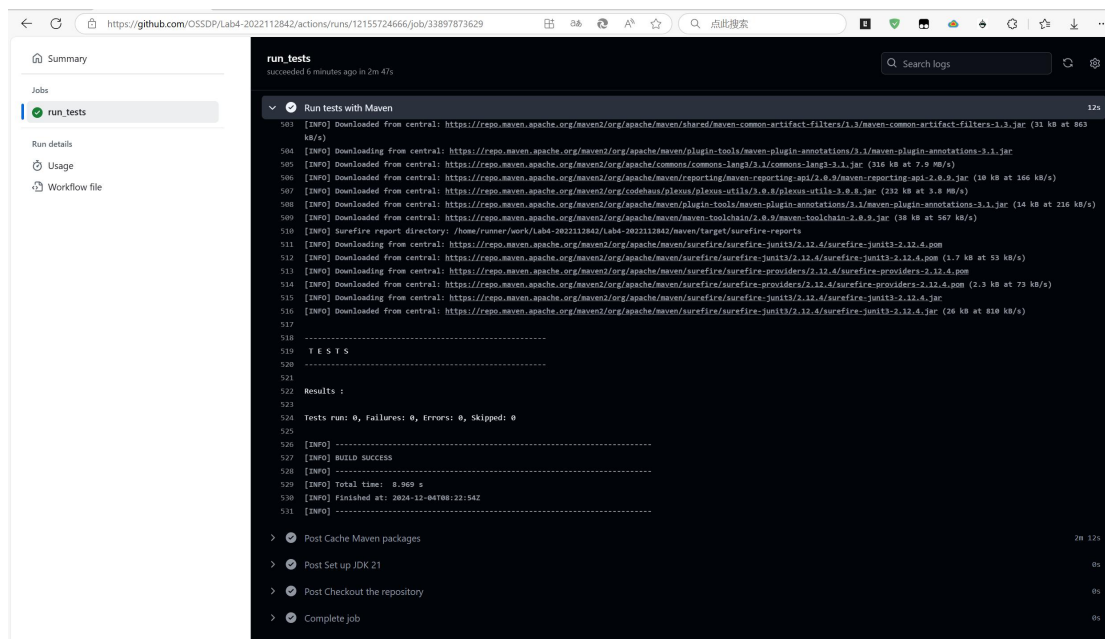
此时项目的目录结构截图如下。



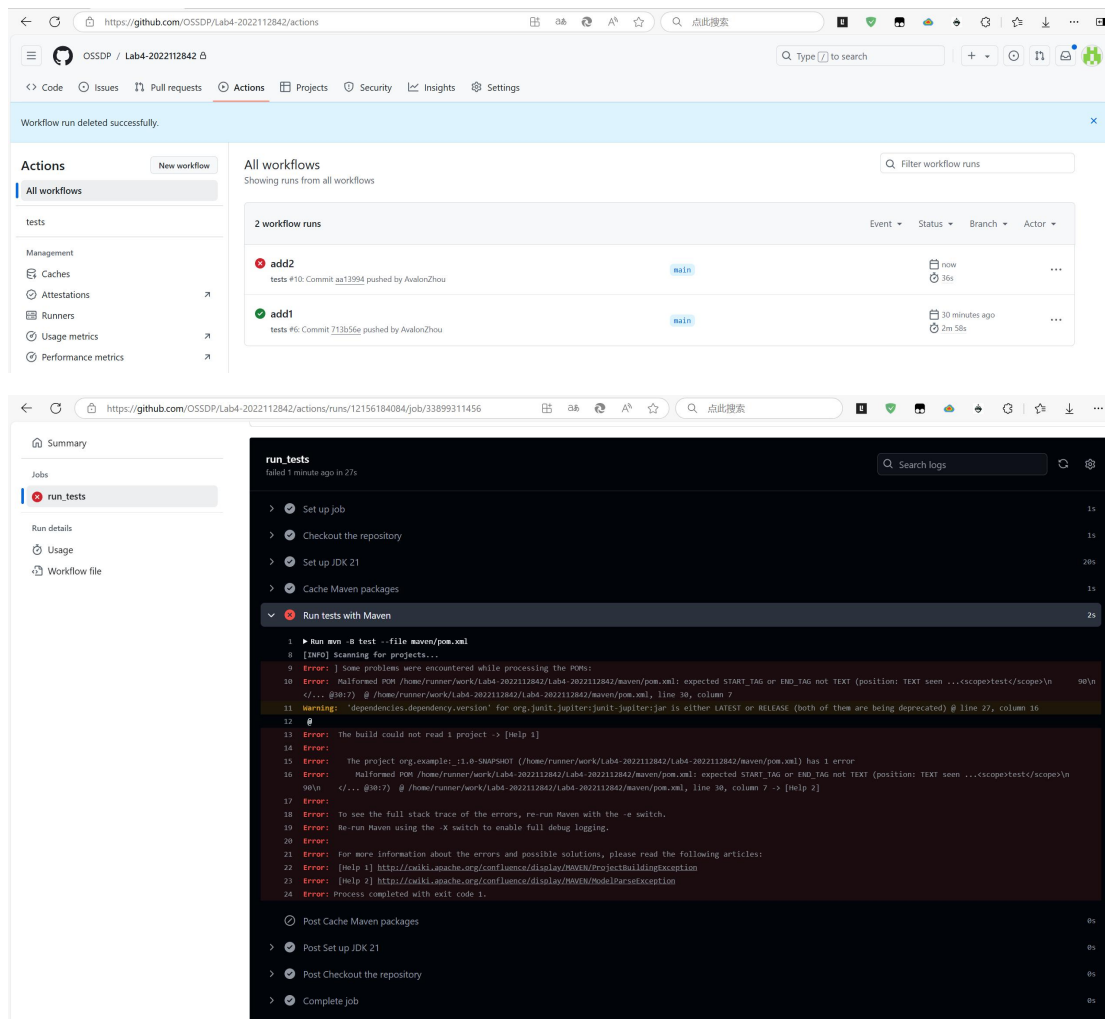
编写完 workflow 文件后，再次提交到仓库，Github 会根据 yml 文件进行自动化测试。



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，可以看到下图中我们定义的测试用例已完成了测试。



修改测试用例为错误的结果，使其无法通过测试。修改后我们再次提交至仓库，下面展示了错误提交的结果。



3 实验内容 2 Jenkins DevOps 实践

在 Jenkins 的官网(<https://www.jenkins.io/zh/download/>), 根据自己计算机操作系统的类型下载安装文件。

安装时, Logon Type 选择 Run Service as Local System 以避免繁琐的额外账户配置(生产环境中应当额外配置 Jenkins 的权限水平, 此实验略), Port Number 默认为 8080, 输入 JAVA_HOME 地址。

安装完成后即可通过所选端口访问 Jenkins 服务, 默认为 <http://localhost:8080/>。

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:

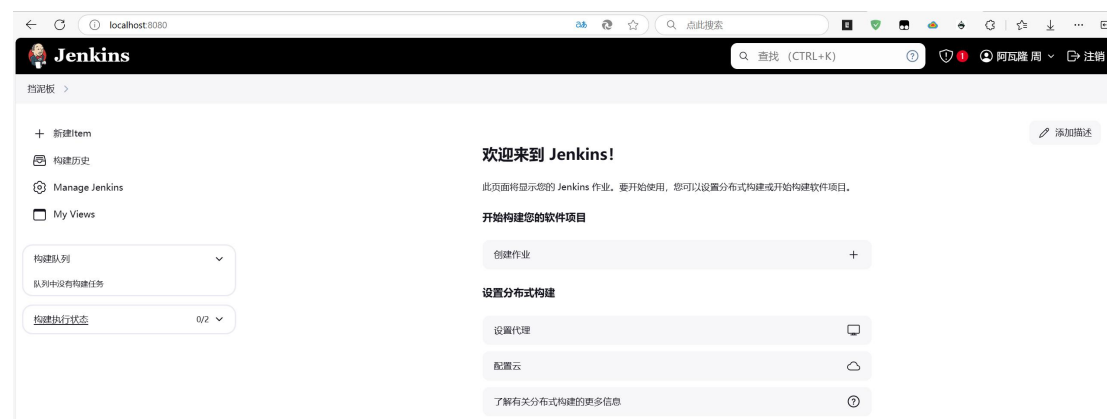
```
C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
```

请从本地复制密码并粘贴到下面。

管理员密码

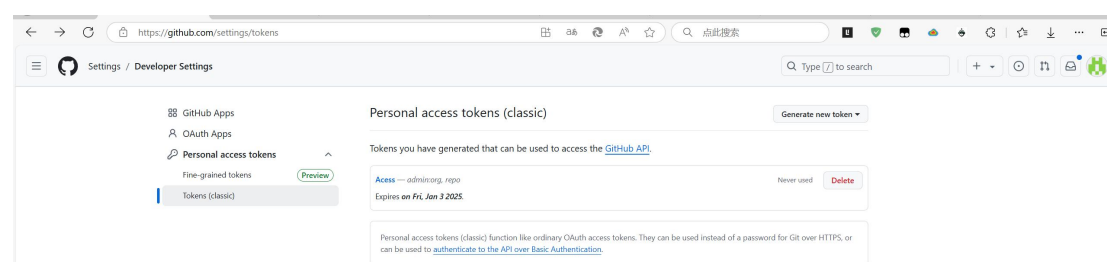
继续

跟随系统提示, 复制初始密码, 解锁 Jenkins; 选择安装推荐的插件(其中包括 Github plugin 等对实验内容有帮助的插件); 创建非管理员账号(之后的操作将在非管理员账号上执行)。



为了能够提交 PR，需要向 Github 认证自己，需要申请一个 Access Token。

- a) 点击 Github 头像
- b) 点击侧边栏的 settings
- c) 点击左侧最下边的 Developer settings
- d) 点击 Personal access token，选择 Token(classic)
- e) Generate new token，勾选 repo、勾选 admin:org，Generate Token 并记下 token。



点击新建 Item 以创建新的构建流程，任务类型选择 Freestyle Project。

1. **General 部分：**该部分是对任务的描述以及一些涉及并发构建、构建结果处理、参数配置的内容，本次实验不涉及该部分内容。

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。

源码管理

☐ 无

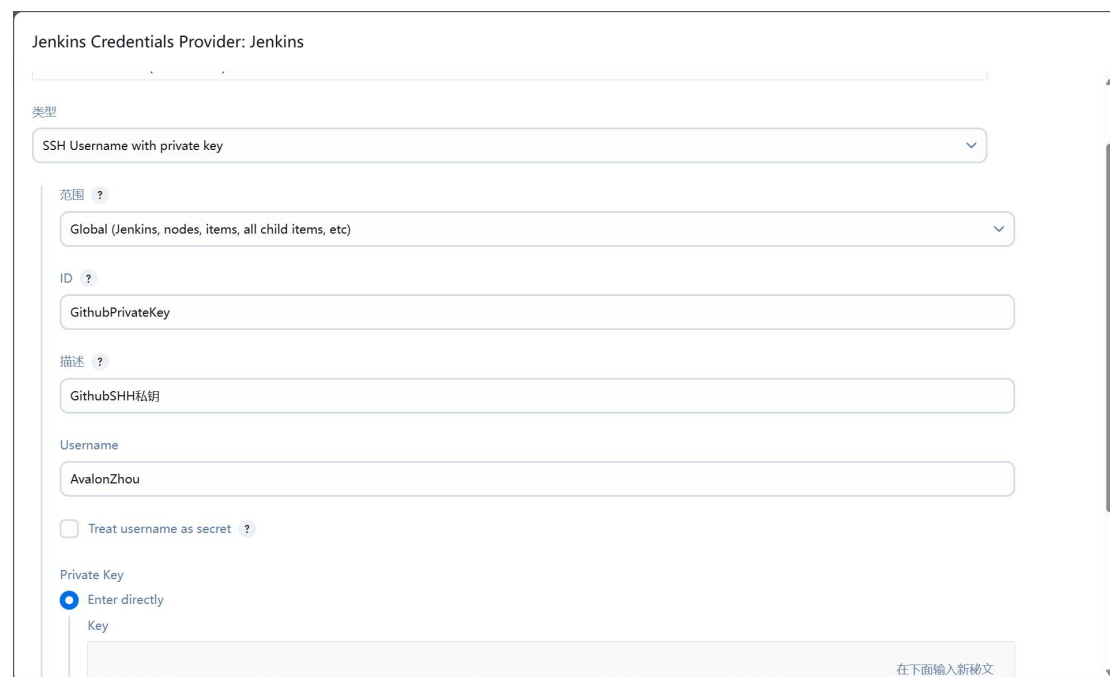
☒ Git ?

Repositories ?

Repository URL ?

git@github.com:OSSDP/Lab4-2022112842.git

此处使用 git@开头的仓库地址，以避免 HTTPS 产生的 443 超时错误。使用 SSH 访问 Github 仓库需要额外的认证凭证，点击“添加”按钮进行配置。



Jenkins Credentials Provider: Jenkins

类型
SSH Username with private key

范围 ?
Global (Jenkins, nodes, items, all child items, etc)

ID ?
GithubPrivateKey

描述 ?
GithubSSH私钥

Username
AvalonZhou

☐ Treat username as secret ?

Private Key
☒ Enter directly
☐ Key

在下面输入新秘文

一般使用 SSH Username 与私钥作为凭证，在 Username 处填入 Github 账户名，Private Key 处粘贴私钥内容（注意：粘贴时要保留第一行中的 BEGIN 和最后一行中的 END 的相关内容）。

注意：如果遇到 Unknown host 错误，这是由于 Github 服务器对于本机来说是不可信任引起的，最简单的方法即：Manage Jenkins=>Security，然后关闭 Host 检查即可。

Git Host Key Verification Configuration

Host Key Verification Strategy ?

No verification

指定 Local 项目代码所在的分支，本例中为 dev。

Branches to build ?

指定分支（为空时代表any） ?

*/dev

3. 构建触发器

该部分配置如何触发构建流程。

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4日星期三 中国标准时间 18:54:31; would next run at 2024年12月4日星期三 中国标准时间 18:54:31.

☐ 忽略钩子 post-commit ?

可以通过点击 Jenkins 页面中的 Build Now 进行构建，但为实现自动化的 CI/CD，可选择如下方案。

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（H/3 * * * *， 具体频率可自行设置，注意“*”号之间用空格分隔）。

第一个*表示分钟，取值0~59

第二个*表示小时，取值0~23

第三个*表示一个月的第几天，取值1~31

第四个*表示第几月，取值1~12

第五个*表示一周中的第几天，取值0~7，其中0和7代表的都是周日

	*	*	*	*	*
含义	分钟	小时	日期	月份	星期
取值范围	0-59	0-23	1月31日	1月12日	0-7
示例					
每隔15分钟执行一次	H/15	*	*	*	*
每隔2个小时执行一次	H	H/2	*	*	*
每隔3天执行一次	H	H	H/3	*	*
每隔3天执行一次(每月的1-15号)	H	H	1-15/3	*	*
每周1,3,5执行一次	H	H	*	*	1,3,5
规则					
指定时间范围	a-b				
指定时间间隔	/				
指定变量取值	a,b,c				

4. 构建环境：该部分用于设置构建流程的环境变量，本实验不涉及这部分内容。
5. 构建步骤：这部分将使用 cmd 脚本进行代码的测试与 PR 的提交。

Build Steps

Execute Windows batch command ?

命令

参阅 可用环境变量列表

```
validate
```

高级 ▾

Execute Windows batch command ?

命令

参阅 可用环境变量列表

```
test
```

高级 ▾

Execute Windows batch command ?

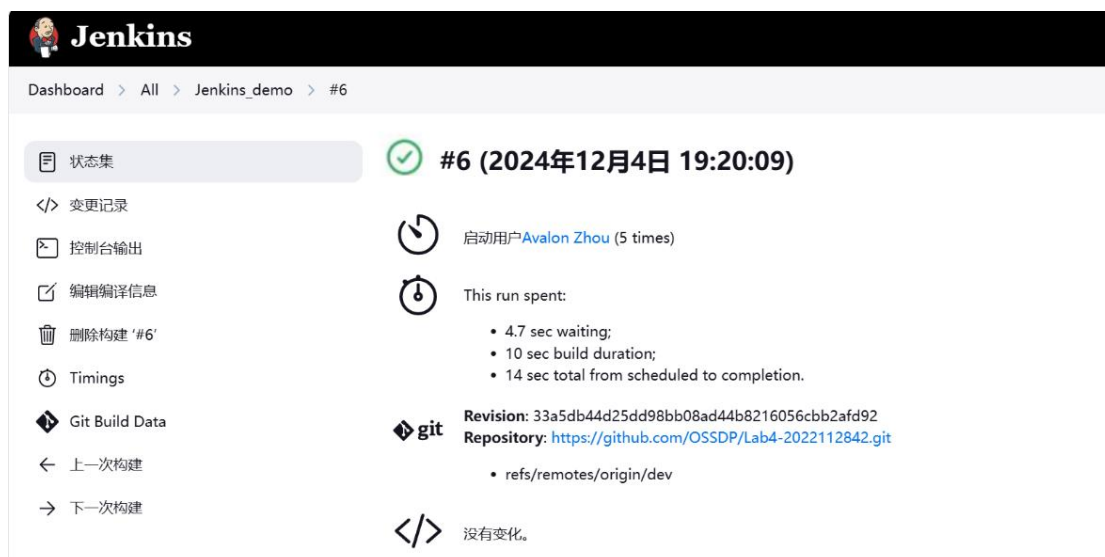
命令

参阅 可用环境变量列表

```
set GH_TOKEN=ghp_Qw1lSjER3QdFuLyBD8tb5UuVWu0rmM0w08N1
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/Lab4-2022112842
```

高级 ▾

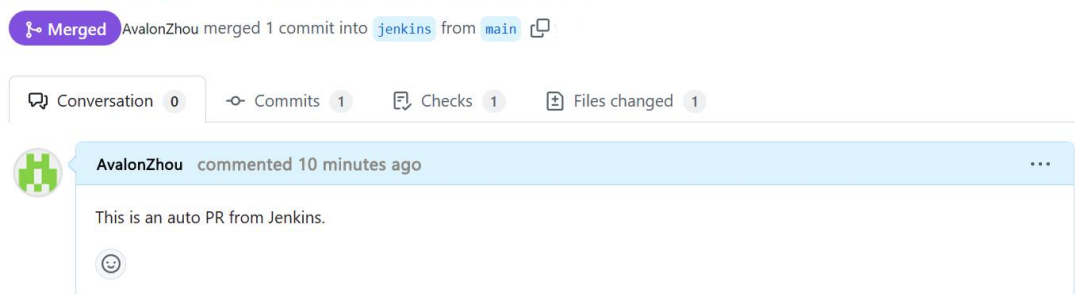
将项目代码提交到选定的仓库的指定分支，本例中为 dev，等待定时任务进行构建流程。查看 Jenkins Build 的结果是否成功。



The screenshot shows the Jenkins dashboard for a job named 'Jenkins_demo' at build #6. The build status is successful, indicated by a green checkmark and the timestamp '2024年12月4日 19:20:09'. The left sidebar contains navigation links: '状态集' (selected), '变更记录', '控制台输出', '编辑编译信息', '删除构建 '#6'', 'Timings', 'Git Build Data', '上一次构建', and '下一次构建'. The main content area displays build details: '启动用户 Avalon Zhou (5 times)', 'This run spent:' (4.7 sec waiting, 10 sec build duration, 14 sec total), 'Revision: 33a5db44d25dd98bb08ad44b8216056cbb2afd92', 'Repository: https://github.com/OSSDP/Lab4-2022112842.git', and 'refs/remotes/origin/dev'. At the bottom, it shows '没有变化。' (No changes).

在 Github 上查看仓库的 PR 是否推送成功。

Auto PR from dev-Jenkins to main #4



4 小结

在本次实验中，我学习了开源软件开发中的基本 DevOps 操作流程，并深入实践了使用 Github Actions 与 Jenkins 进行持续集成与持续部署（CI/CD）的技能。整个过程从理论到实践，让我对 DevOps 理念有了更深刻的理解。通过亲手配置和触发自动化构建、测试、部署流程，我深刻体会到 DevOps 在提高软件开发效率、缩短交付周期以及增强团队协作方面的巨大优势。

使用 Github Actions 时，我感受到了其作为 GitHub 原生 CI/CD 工具的便捷性，尤其是在与代码仓库紧密集成、配置简单直观以及支持丰富的预定义操作和社区动作方面，这使得快速搭建 CI/CD 流水线变得轻松可行。而 Jenkins 则以其高度的可扩展性和丰富的插件生态系统给我留下了深刻印象，尽管初始配置相对复杂，但一旦熟悉其强大的脚本和配置管理能力，就能灵活应对各种复杂的构建和部署需求。

这次实验不仅让我掌握了两种主流 CI/CD 工具的使用，更重要的是培养了我从全局视角审视软件开发流程的能力，学会了如何根据项目特点和团队需求选择合适的工具和技术栈来优化 DevOps 实践。此外，实验过程中遇到的各种问题和挑战，如配置错误、脚本调试等，也极大地锻炼了我的问题解决能力和耐心，让我学会了如何有效地利用文档、社区资源来寻求帮助。

总之，本次实验是一次非常宝贵的学习经历，它不仅丰富了我的技术栈，更重要的是提升了我的综合技能，为我未来在开源软件开发和 DevOps 领域的职业发展奠定了坚实的基础。我深刻认识到，持续学习和实践是成为一名优秀 DevOps 工程师的关键，期待在未来的项目中能够将这些知识和经验应用到实际，不断优化软件开发流程，提升产品质量和交付效率。