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
2022111744	刘译阳	18646011272

目 录

[文档全部完成之后，请更新上述区域]

1 实验要求

1. 掌握开源软件开发中的基本 DevOps 流程和工具的使用

学习并掌握开源软件开发中的 DevOps 基本流程，包括代码版本控制、持续集成（CI）、持续交付（CD）以及自动化部署等关键内容。通过实际操作，熟悉 DevOps 的完整生命周期，从代码开发到部署上线的每一步都实现自动化，减少手动干预带来的错误。熟练使用开源工具，完成代码的版本管理、自动化测试、构建和部署等任务，切身体会 DevOps 如何提高开发效率和代码质量。

2. 熟悉利用 GitHub Actions 进行 DevOps

GitHub Actions 是一种广泛应用于持续集成和持续交付的自动化工作流工具。通过实验，学习如何配置并使用 GitHub Actions 来实现 CI/CD 流程，包括代码测试、构建以及部署等操作。编写 YAML 文件定义工作流，探索触发条件（如代码提交或 Pull Request），实现自动化测试和构建流程。分析工作流运行日志，解决可能出现的错误，确保工作流正常运行。

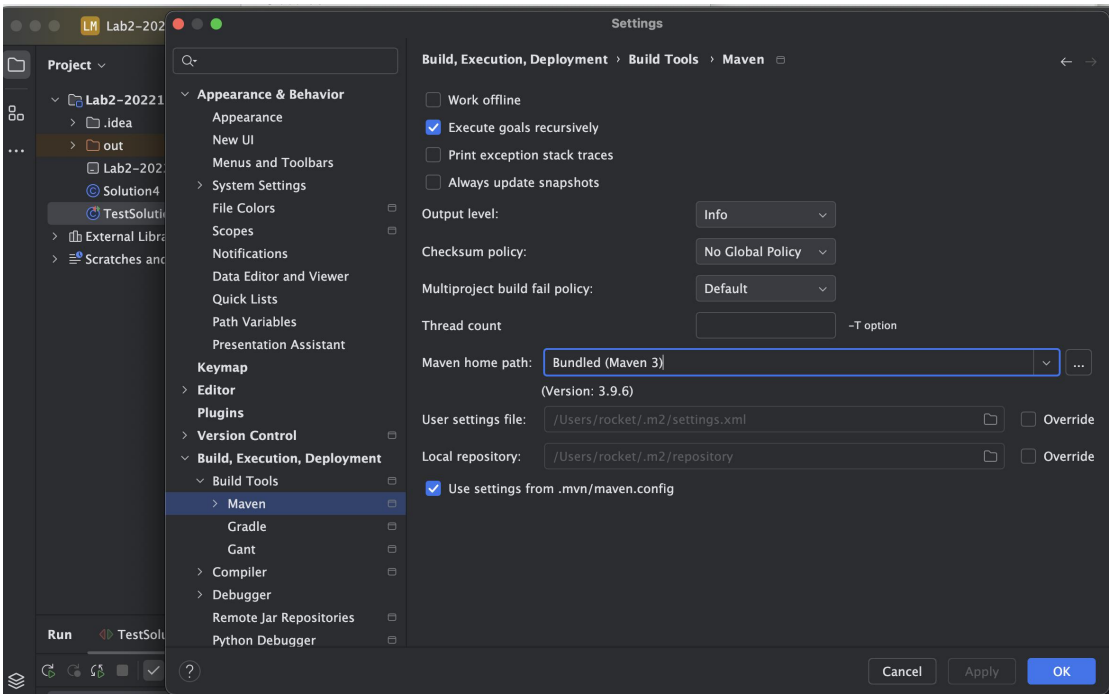
3. 熟悉利用 Jenkins 进行 DevOps

Jenkins 是一款功能强大的开源自动化服务器，是 DevOps 实践的常用工具之一。实验中需安装、配置并使用 Jenkins 来实现类似 GitHub Actions 的 CI/CD 流程。创建 Jenkins Pipeline，配置相关插件，通过流水线实现项目的构建、测试和部署等任务。熟悉 Jenkins 的界面操作，理解与 Git、Maven 等工具的集成方式，并对运行中出现的问题进行分析和调试。

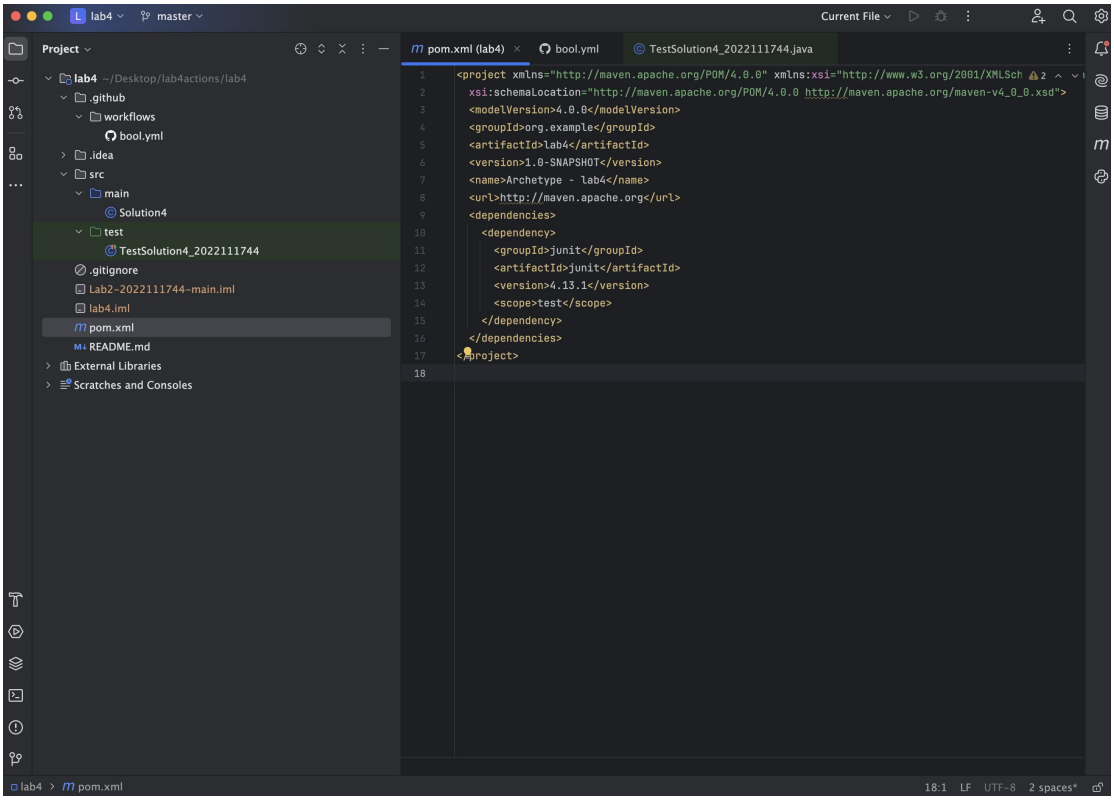
通过本次实验，掌握 GitHub Actions 和 Jenkins 两种工具在 DevOps 实践中的具体应用，全面提升在开源项目开发中的自动化能力和实际操作经验，为未来参与开源社区开发及企业级 DevOps 项目奠定坚实基础。

2. 实验内容 1 Github Actions DevOps 实践

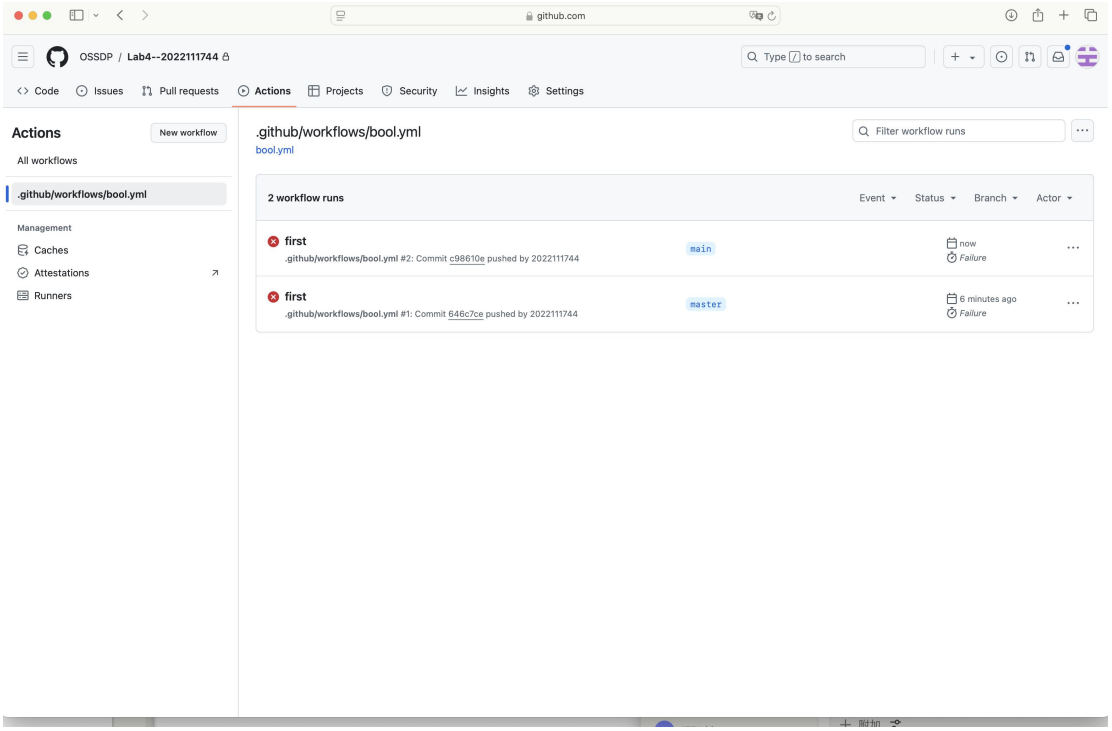
提前配置 maven



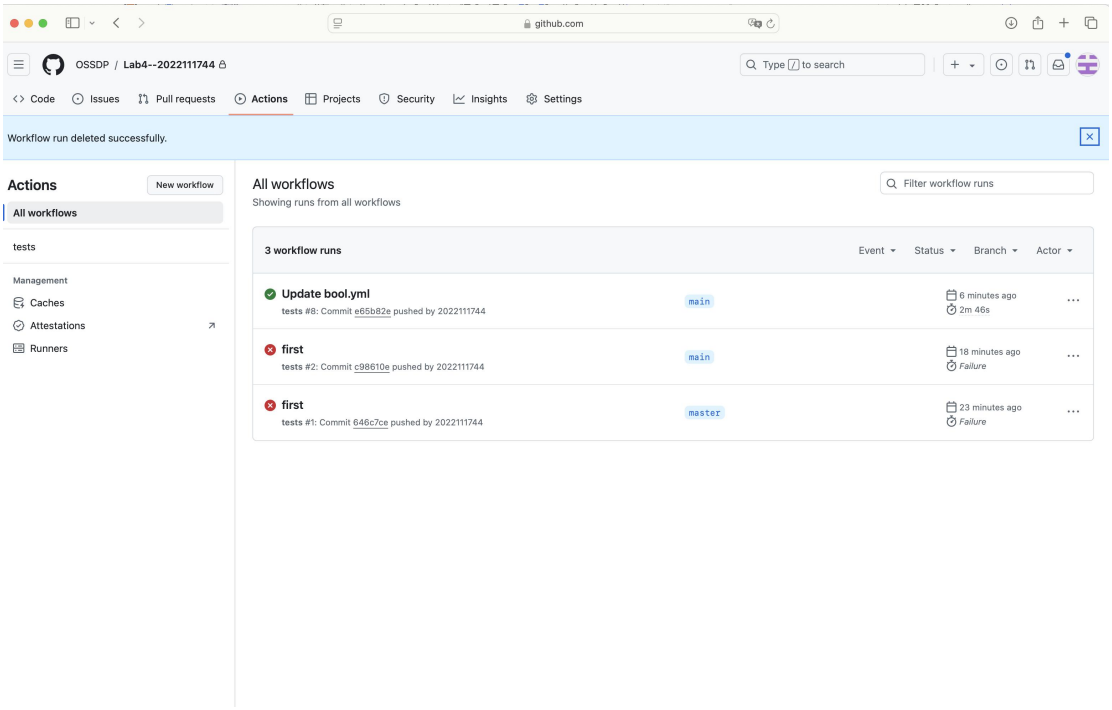
给出项目的目录结构截图，文件管理器或编程 IDE 的界面均可。



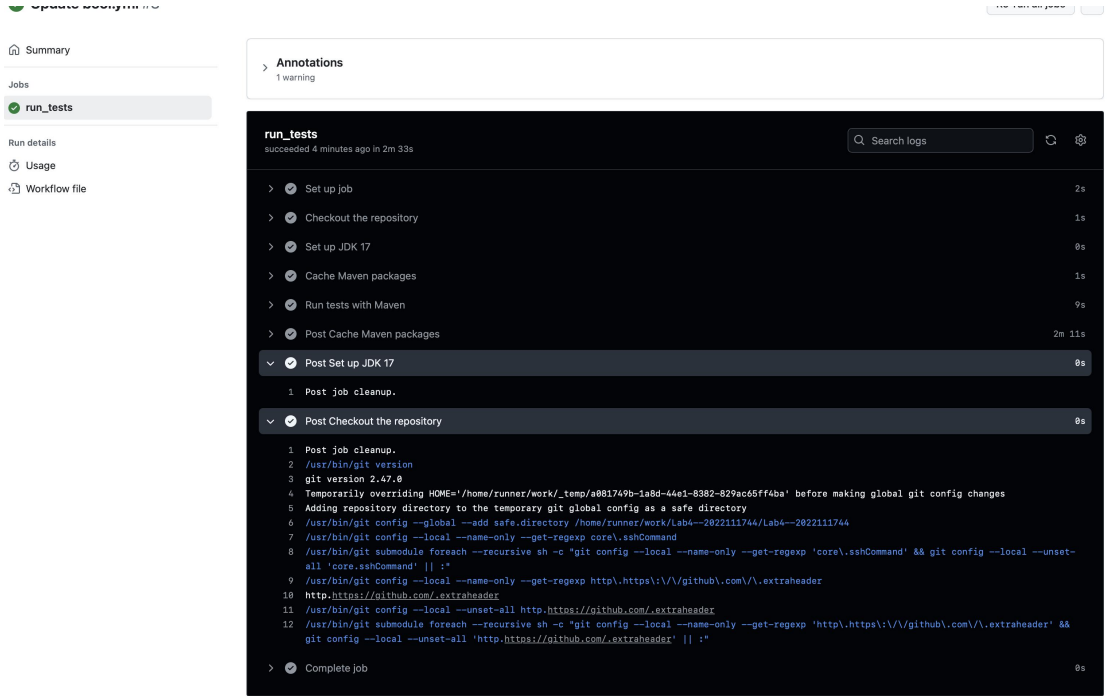
- Github 中的 Actions 的若干界面，包括：执行自动化测试成功界面、执行自动化测试失败界面、执行具体信息等，类似实验指导书中的示例界面，以证明完成了实验



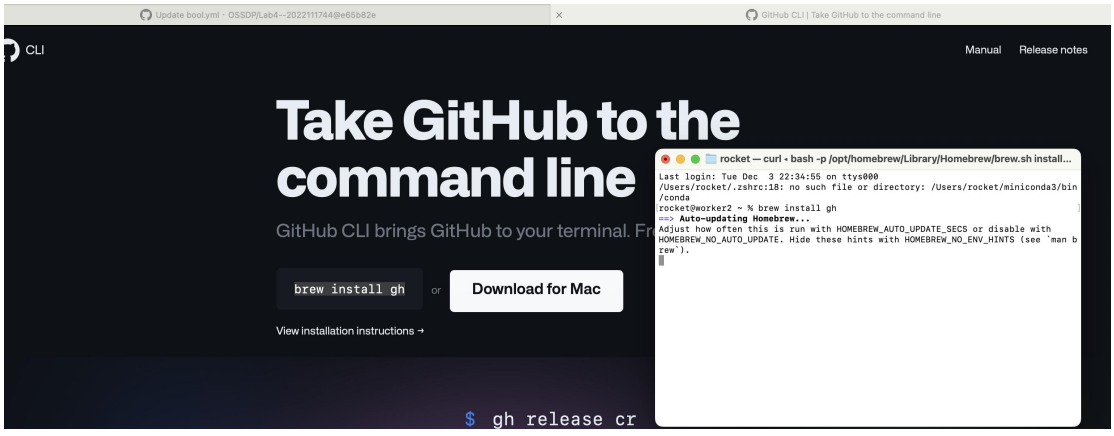
上图因为第一次没有修改好配置



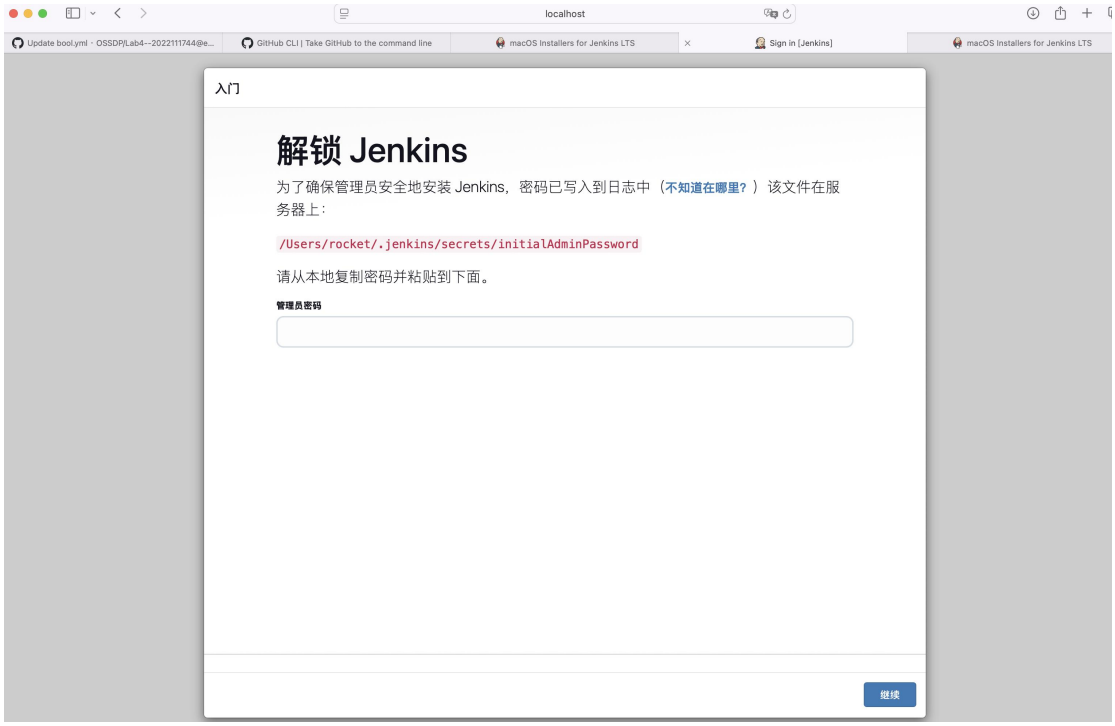
修改 main 分支配置后成功（下面为具体信息）



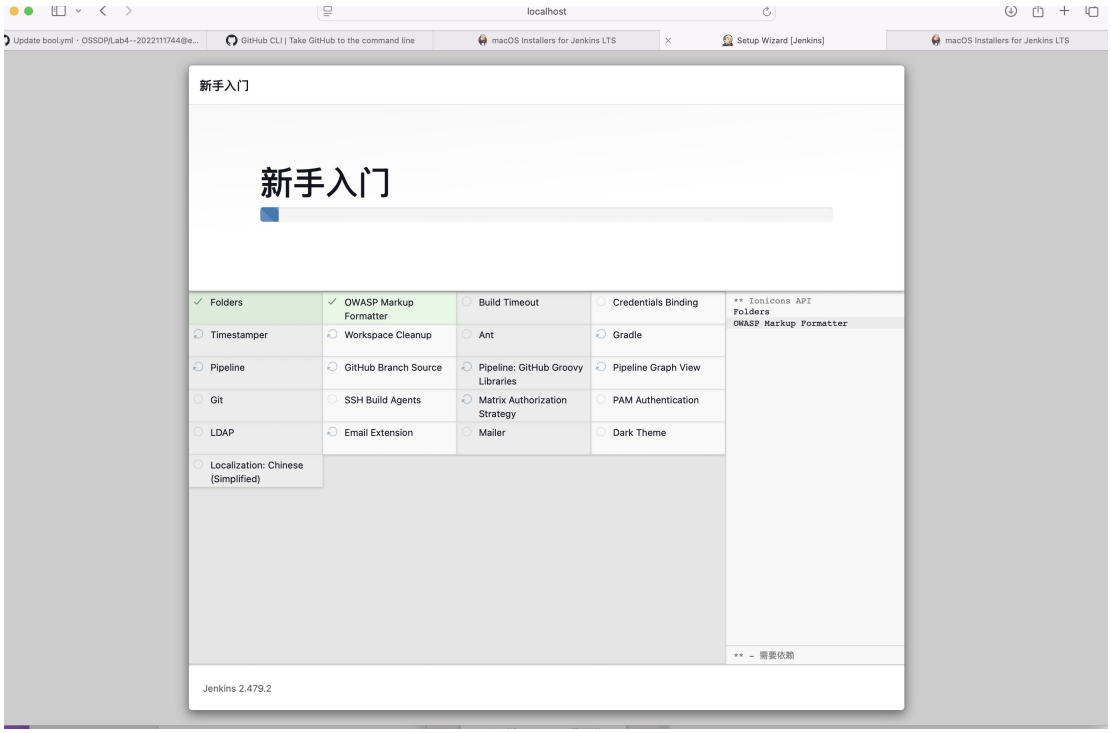
2 实验内容 2 Jenkins DevOps 实践



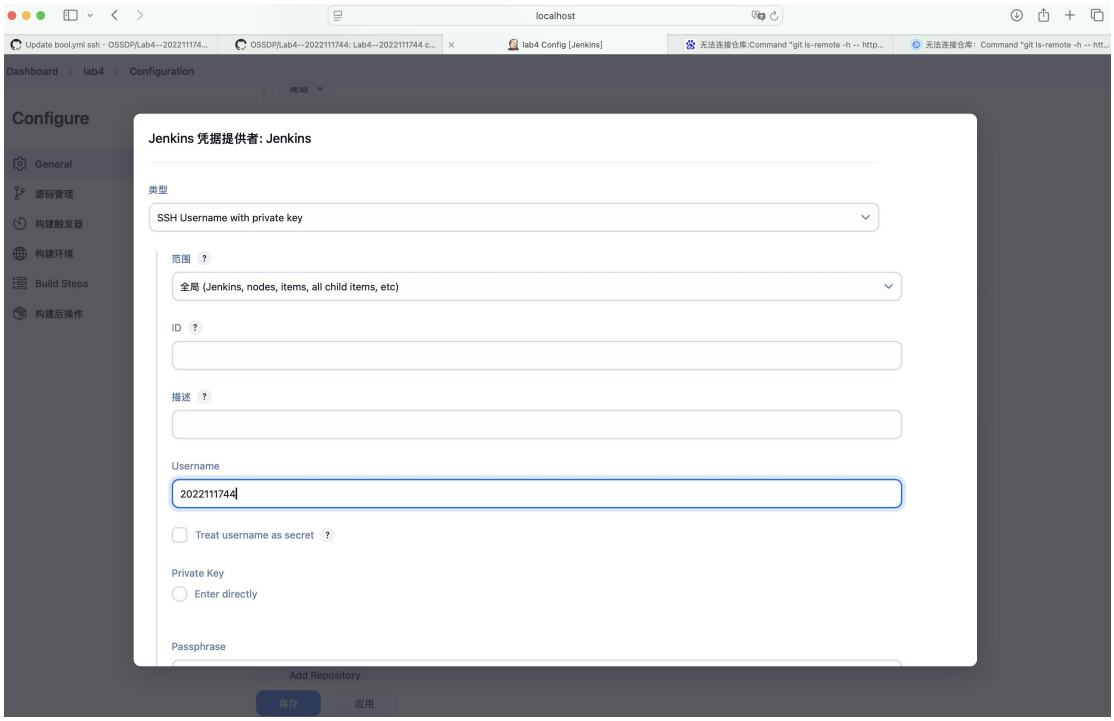
在本地启动

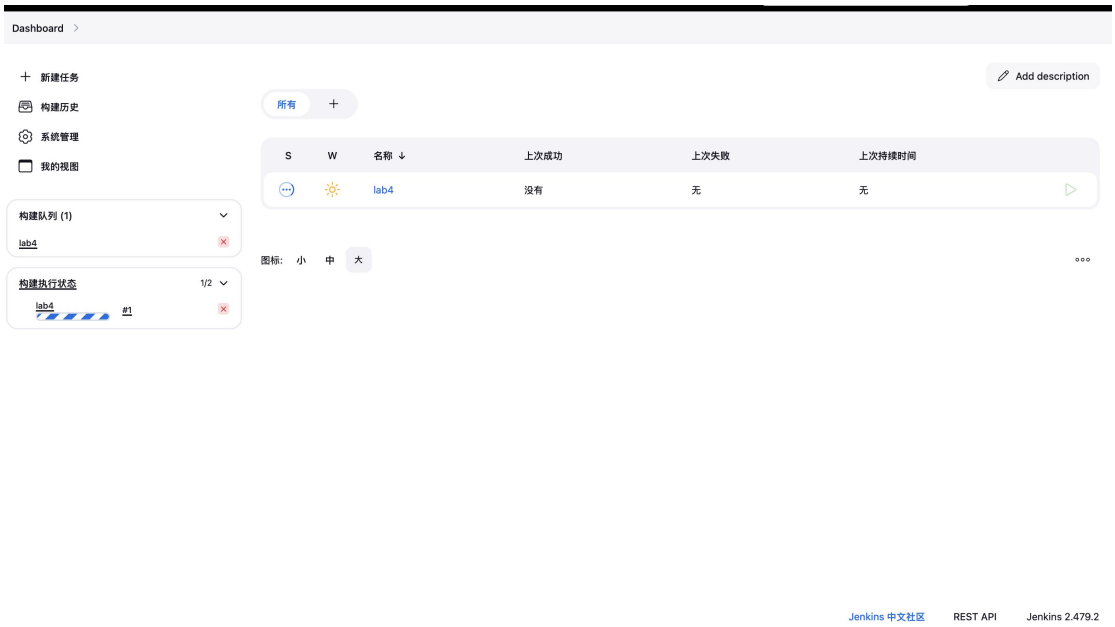


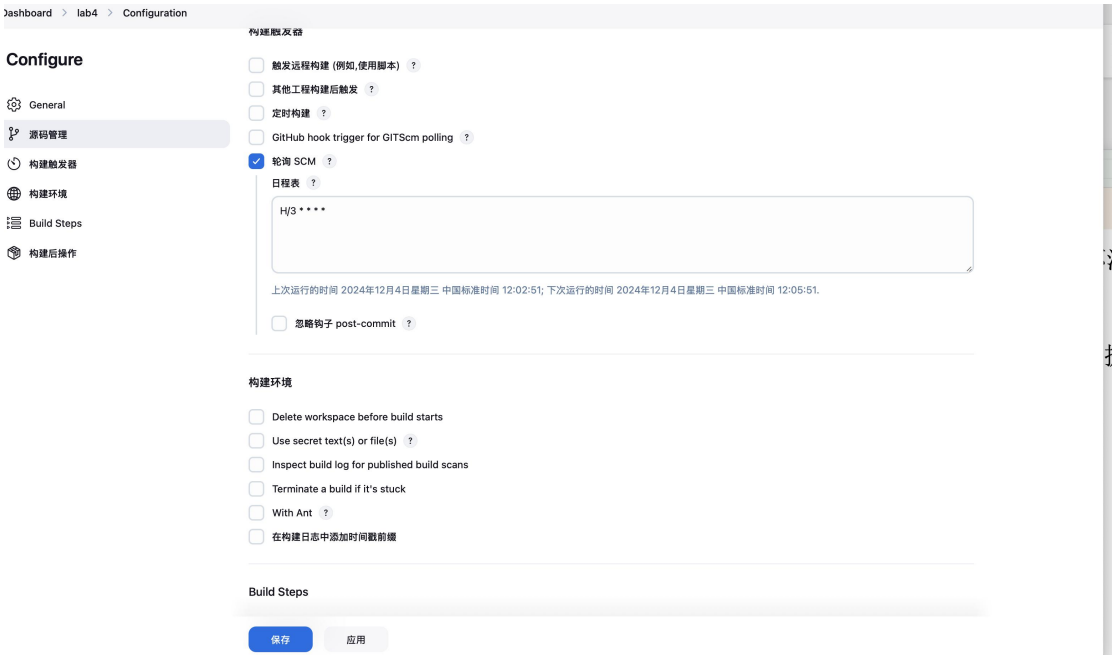
开始安装：



配置完成 ssh 开始构建：

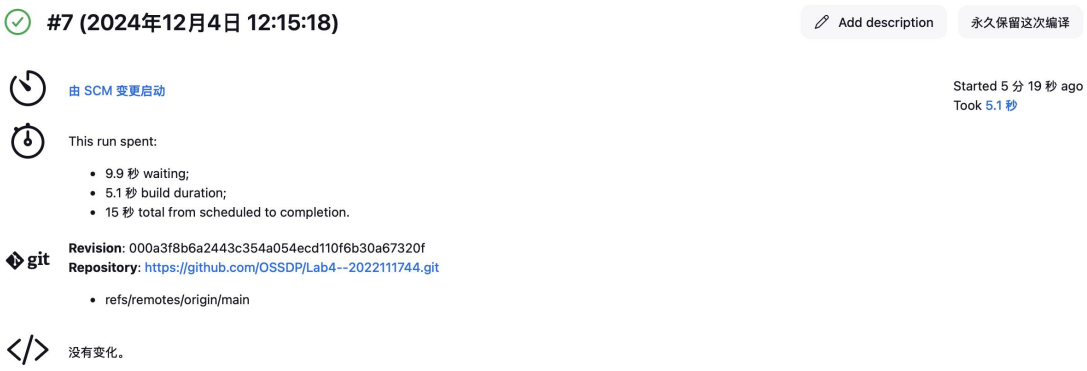

localhost
Dashboard > lab4 > Configuration
Configure
General
源码管理
构建触发器
构建环境
Build Steps
构建后操作
Jenkins 凭据提供者: Jenkins
类型
SSH Username with private key
范围
全局 (Jenkins, nodes, items, all child items, etc)
ID
描述
Username
2022111744
Treat username as secret
Private Key
Enter directly
Passphrase
Add Repository
保存
应用


Dashboard
新建任务
构建历史
系统管理
我的视图
构建队列 (1)
lab4
构建执行状态
1/2
lab4
#1
Add description
所有
S
W
名称
上次成功
上次失败
上次持续时间
lab4
没有
无
无
图标: 小 中 大
Jenkins 中文社区
REST API
Jenkins 2.479.2

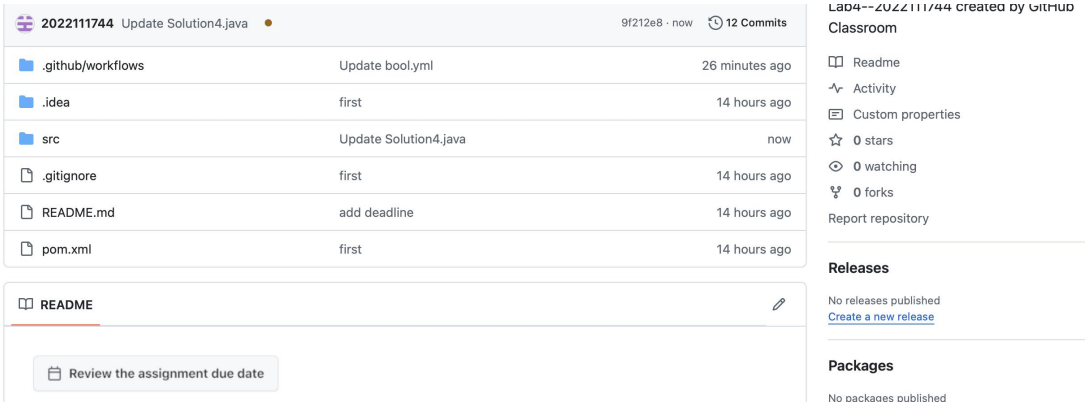

Dashboard > lab4 > Configuration
Configure
General
源码管理
构建触发器
构建环境
Build Steps
构建后操作
触发远程构建 (例如,使用脚本)
其他工程构建后触发
定时构建
GitHub hook trigger for GITScm polling
轮询 SCM
日程表
H/3 * * * *
上次运行的时间 2024年12月4日星期三 中国标准时间 12:02:51; 下次运行的时间 2024年12月4日星期三 中国标准时间 12:05:51.
忽略钩子 post-commit
构建环境
Delete workspace before build starts
Use secret text(s) or file(s)
Inspect build log for published build scans
Terminate a build if it's stuck
With Ant
在构建日志中添加时间戳前缀
Build Steps
保存
应用

轮询

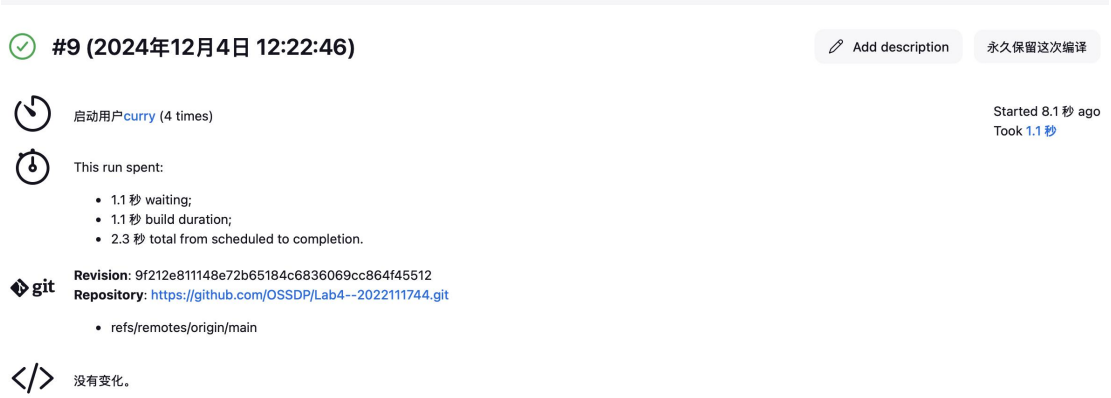
配置成功:


#7 (2024年12月4日 12:15:18)
Add description
永久保留这次编译
由 SCM 变更启动
Started 5 分 19 秒 ago
Took 5.1 秒
This run spent:
9.9 秒 waiting;
5.1 秒 build duration;
15 秒 total from scheduled to completion.
git
Revision: 000a3f8b6a2443c354a054ecd110f6b30a67320f
Repository: https://github.com/OSSDP/Lab4--2022111744.git
refs/remotes/origin/main
没有变化。

修改代码:



再次构建成功:



3 小结

在本次实验中，我们主要通过使用 Jenkins 自动化构建工具，结合 Git 代码管理，完成了对一个 GitHub 仓库的自动拉取、构建和部署过程。实验的目标是加深对 Jenkins 持续集成（CI）流程的理解，并学会如何配置 Git 仓库与 Jenkins 进行有效的集成，以便实现自动化部署和持续集成的目的。

首先，在实验的开始阶段，我们配置了 Jenkins 环境，并确保其能够正常运行。接下来，我们配置了 Jenkins 与 GitHub 仓库的连接，确保 Jenkins 能够通过 Git 拉取代码。由于实验过程中需要使用 GitHub 上的 Lab4 仓库，我们设置了仓库的 URL，并在 Jenkins 配置中指定了需要拉取的分支。通过配置合适的凭证，Jenkins 能够使用 SSH 或 HTTPS 协议成功连接到 GitHub，并获取最新的代码版本。

实验过程中遇到了一些问题，主要表现为 Jenkins 无法从 GitHub 拉取代码。经过排查，我

们发现问题出在网络连接上，Jenkins 服务器由于某些防火墙或代理设置导致无法连接到 GitHub。在解决了网络问题后，实验顺利进行了下去，并成功完成了自动构建和部署的过程。

通过本次实验，我深刻理解了 Jenkins 在持续集成中的重要性。Jenkins 可以自动化处理代码拉取、构建、测试和部署等一系列过程，大大提高了软件开发的效率和质量。此外，我还学会了如何通过配置分支选择器来指定需要构建的分支，并且通过凭证管理确保了与 Git 仓库的安全连接。这些技能对我未来的自动化部署和 DevOps 实践非常有帮助。

总的来说，本次实验不仅让我掌握了 Jenkins 的基本使用方法，还让我深入理解了持续集成的理念。通过自动化构建和部署，开发团队可以更高效地管理代码版本，及时发现和修复问题，从而提高软件开发的质量和速度。这些经验将为我未来在软件工程实践中应用 CI/CD 流程打下坚实的基础。