

# [CS304] Lab14. Use CI/CD tools

---

## Using Jenkins

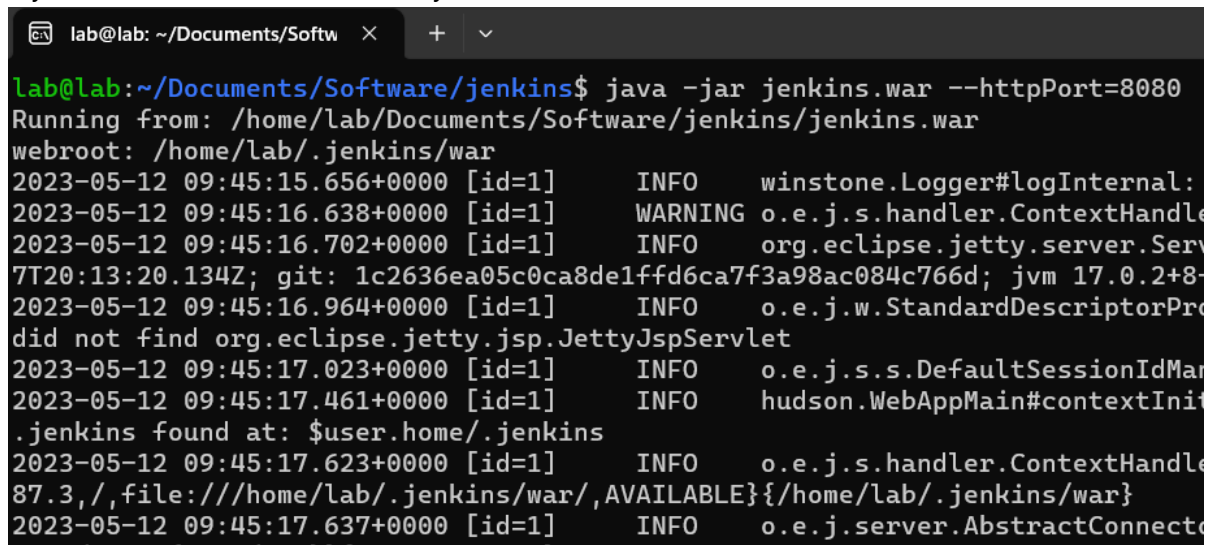
In order to use Jenkins, first download Jenkins Generic Java package (.war)

<https://www.jenkins.io/download/>, put this war file wherever you want.

Open a terminal, "cd" to the directory contains the war file and run:

```
java -jar jenkins.war --httpPort=8080
```

If you installed Maven in wsl, then you should execute the above command in wsl:

A screenshot of a terminal window with a dark background. The title bar shows 'lab@lab: ~/Documents/Softw' and window controls. The terminal text shows the command 'java -jar jenkins.war --httpPort=8080' being executed. The output includes the path to the war file, a webroot path, and several log messages from the Jenkins startup process, including warnings and information about the server configuration and session management. The terminal text is as follows:

```
lab@lab:~/Documents/Softw x + v
lab@lab:~/Documents/Software/jenkins$ java -jar jenkins.war --httpPort=8080
Running from: /home/lab/Documents/Software/jenkins/jenkins.war
webroot: /home/lab/.jenkins/war
2023-05-12 09:45:15.656+0000 [id=1] INFO winstone.Logger#logInternal:
2023-05-12 09:45:16.638+0000 [id=1] WARNING o.e.j.s.handler.ContextHandle
2023-05-12 09:45:16.702+0000 [id=1] INFO org.eclipse.jetty.server.Serv
7T20:13:20.134Z; git: 1c2636ea05c0ca8de1ffd6ca7f3a98ac084c766d; jvm 17.0.2+8-
2023-05-12 09:45:16.964+0000 [id=1] INFO o.e.j.w.StandardDescriptorPro
did not find org.eclipse.jetty.jsp.JettyJspServlet
2023-05-12 09:45:17.023+0000 [id=1] INFO o.e.j.s.s.DefaultSessionIdMan
2023-05-12 09:45:17.461+0000 [id=1] INFO hudson.WebAppMain#contextInit
.jenkins found at: $user.home/.jenkins
2023-05-12 09:45:17.623+0000 [id=1] INFO o.e.j.s.handler.ContextHandle
87.3,,file:///home/lab/.jenkins/war/,AVAILABLE}{/home/lab/.jenkins/war}
2023-05-12 09:45:17.637+0000 [id=1] INFO o.e.j.server.AbstractConnecto
```

The output in the window will tell you an initial password. Copy that password.

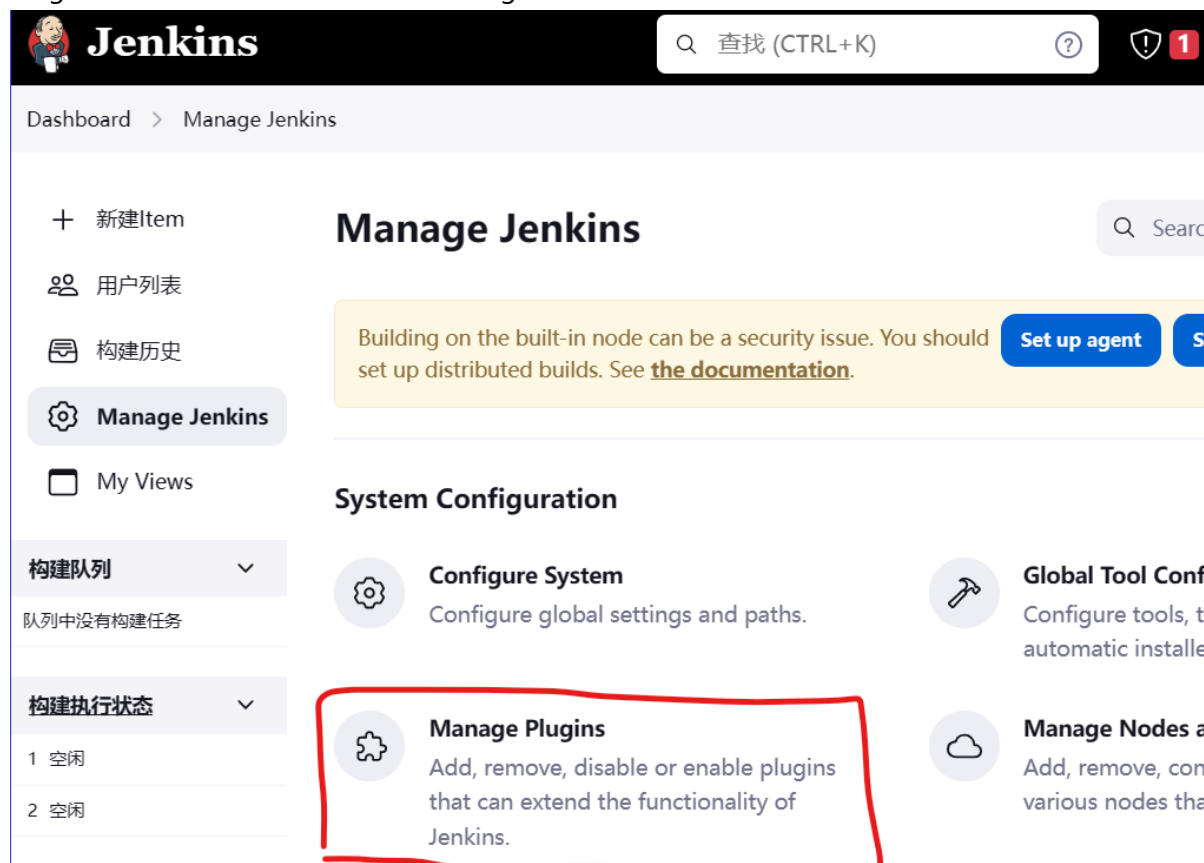
Open you browser and go to: <http://localhost:8080>

After you paste the initial password showing that you have access to your computer, Jenkins will ask you to finish some initial steps. Follow the steps to finish installation. Choose "Install

Recommended Plugins" if you are new to Jenkins.



After that you may want to install "Maven Intergration Plugin" in your Jenkins. Open "Manage Plugins" and find it under "Available Plugins" and install it:



Restart Jenkins. The plugins needs to restart.

Create a Maven project on github, your project may look something like this:

selab722 / hello-worldPublic

PinUnwatch1

<> Code

Issues

Pull requests1

Actions

Projects

Wiki

Security

main3 branches0 tags

Go to file

Add file

<> Code

Your main branch isn't protected

Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#)

Protect this branch

Your Name delete jenkinsfile

dcbbd1e48 minutes ago

🕒 13 commits

src

add maven and jenkins files

18 hours ago

.gitignore

Initial commit

3 months ago

LICENSE

Initial commit

3 months ago

README.md

clear text in readme.md

2 months ago

pom.xml

add maven and jenkins files

18 hours ago

Now Let's use Jenkins to manage this project. Click "New Item" on the Jenkins startup page:

localhost:8080

Jenkins

查找 (CTRL+K)

Dashboard

+ 新建Item

👤 用户列表

📁 构建历史

⚙️ Manage Jenkins

📄 My Views

构建队列

▼

队列中没有构建任务

欢迎来到 Jenkins!

This page is where your Jenkins jobs will be displayed. To get distributed builds or start building a software project.

Start building your software project


Create a job

Type a name here and choose "Build a Maven Project" here:



The screenshot shows the Jenkins Dashboard. At the top, there's a search bar with the text "查找 (CTRL+K)" and a user profile icon labeled "admin". Below the header, the breadcrumb "Dashboard > 所有 >" is visible. The main content area has a section titled "输入一个任务名称" (Enter a task name) with a text input field containing "hello-world" and a note "» 必填项" (» Required). Below this, there are two task cards. The first card is "构建一个自由风格的软件项目" (Build a free-style software project) with a description: "这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统." The second card is "构建一个maven项目" (Build a maven project) with a description: "构建一个maven项目.Jenkins利用你的POM文件,这样可以大大减轻构建配置." This second card is highlighted with a red rectangle.

Then we enter a long list of configuration. First type your github url:



The screenshot shows the Jenkins Configuration page for the "hello-world" project. The breadcrumb is "Dashboard > hello-world > Configuration". The page is titled "Configure" and "General". On the left, there's a sidebar with various configuration options: "General" (selected), "源码管理" (Source Management), "构建触发器" (Build Triggers), "构建环境" (Build Environment), "Pre Steps", "Build", "Post Steps", "构建设置" (Build Settings), and "构建后操作" (Post-build Actions). The main content area is titled "描述" (Description) and contains a text input field with the URL "https://github.com/selab722/hello-world". Below this, there's a link "[纯文本] 预览" ([Plain Text] Preview). Further down, there's a checkbox labeled "GitHub 项目" (GitHub Project) which is checked. Below this, there's a label "项目 URL ?" (Project URL ?) and a text input field with the URL "https://github.com/selab722/hello-world/". At the bottom, there's a button labeled "高级" (Advanced) with a dropdown arrow.

Then fill in the github url again. Add a credential if you need to access any privilege of your github account, but that not necessary for today:

● Git ?

# Configure

⚙️ General

🔑 源码管理

🕒 构建触发器

🌐 构建环境

⚙️ Pre Steps

⚙️ Build

⚙️ Post Steps

⚙️ 构建设置

📦 构建后操作

Repositories ?

Repository URL ?

https://github.com/selab722/hello-world.git

Credentials ?

+ 添加 ▾

高级 ▾

Add Repository

Branches to build ?

指定分支 (为空时代表any) ?

\*/main

Choose how a new build is triggered.

Dashboard > hello-world > Configuration

Configure

General

源码管理

构建触发器

构建环境

Pre Steps

Build

Post Steps

构建设置

构建后操作

构建触发器

☒ Build whenever a SNAPSHOT dependency is built ?

If checked, Jenkins will parse the POMs of this project, and see if any of its snapshot dependencies are built on this Jenkins as well. If so, Jenkins will set up build dependency relationship so that whenever the dependency job is built and a new SNAPSHOT jar is created, Jenkins will schedule a build of this project. This is convenient for automatically performing continuous integration. Jenkins will check the snapshot dependencies from the <dependency> element in the POM, as well as <plugin>s and <extension>s used in POMs.

If this behavior is problematic, uncheck this option.

☐ Schedule build when some upstream has no successful builds ?

☐ 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 ?

You need to configure the location of your Maven (the Maven software, not the project). Then add "clean package" to "Goals and options", this is what we want Jenkins to do for now:

Dashboard > hello-world > Configuration

Configure

General

源码管理

构建触发器

构建环境

Pre Steps

Build

Post Steps

构建设置

Build

Maven Version

! Jenkins needs to know where your Maven is installed. Please do so from the tool configuration.

Root POM ?

pom.xml

Goals and options ?

高级 ▾

Click "the tool configuration" in the above graph, and configure your maven location:

### Maven

Maven 安装 ^    Edited

Maven 安装

系统下Maven 安装列表

[新增 Maven](#)

Maven Name

maven3.9

MAVEN\_HOME

/home/lab/Documents/Software/apache-maven-3.9.0

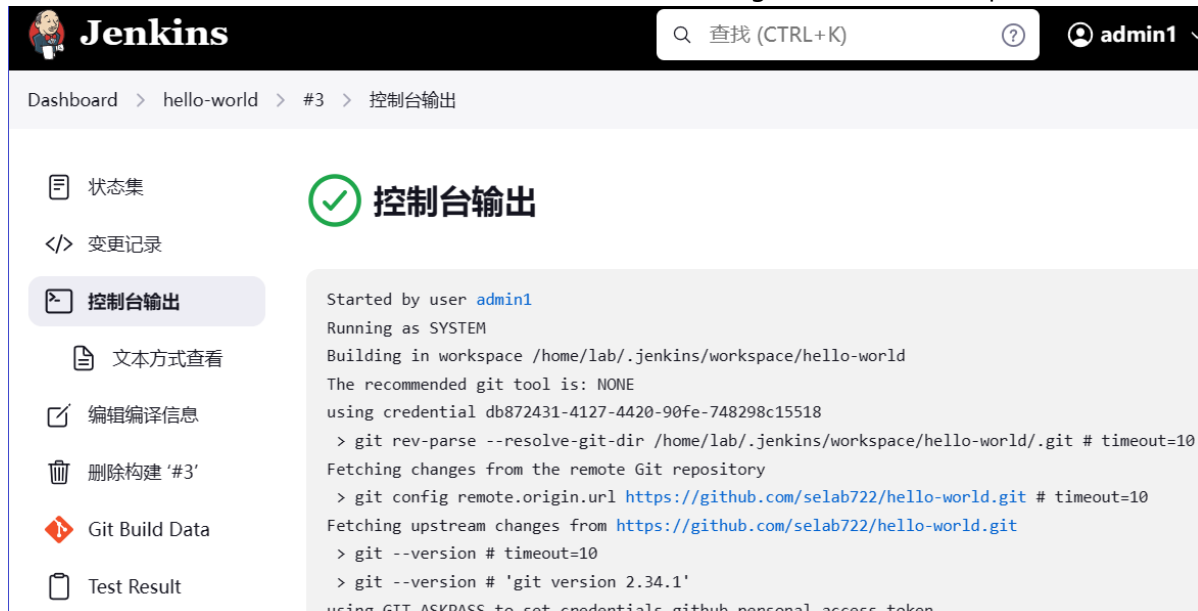
☐ 自动安装 ?

This location should contains "bin/mvn" file that runs Maven. You could run "which mvn" to have a clue where it is.

After you finished all the config, click "Save". Now you can click "Build Instantly" to build this project.

The screenshot shows the Jenkins web interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and user information. Below this is a breadcrumb trail: Dashboard > hello-world >. The main content area is titled "Maven project hello-world" and shows the repository URL: https://github.com/selab722/hello-world. On the left sidebar, the "立即构建" (Build Now) button is highlighted with a red box. The main area displays a "测试结果趋势" (Test Results Trend) chart showing a single green dot for "Passed". Below the chart, there's a "相关链接" (Related Links) section listing recent build results, including "最近一次构建(#2), 52 秒之前" and "最近失败的构建(#1), 20 分之前". At the bottom, there's a "Build History" section with a search bar and a dropdown menu.

After the build is successful or failed we can see the result together with the output.



The screenshot shows the Jenkins web interface. At the top, there's a search bar and a user profile for 'admin1'. The breadcrumb navigation shows 'Dashboard > hello-world > #3 > 控制台输出'. On the left sidebar, there are links for '状态集', '变更记录', '控制台输出' (which is selected), '文本方式查看', '编辑编译信息', '删除构建 '#3'', 'Git Build Data', and 'Test Result'. The main area displays the console output for the selected build, which is a successful build. The output text is as follows:

```
Started by user admin1
Running as SYSTEM
Building in workspace /home/lab/.jenkins/workspace/hello-world
The recommended git tool is: NONE
using credential db872431-4127-4420-90fe-748298c15518
> git rev-parse --resolve-git-dir /home/lab/.jenkins/workspace/hello-world/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/selab722/hello-world.git # timeout=10
Fetching upstream changes from https://github.com/selab722/hello-world.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
using GIT_ASKPASS to set credentials github personal access token
```

By adding a "Jenkinsfile" file you can ask Jenkins to do more, such as using a docker. Refer to the Jenkins document for a more detailed description:

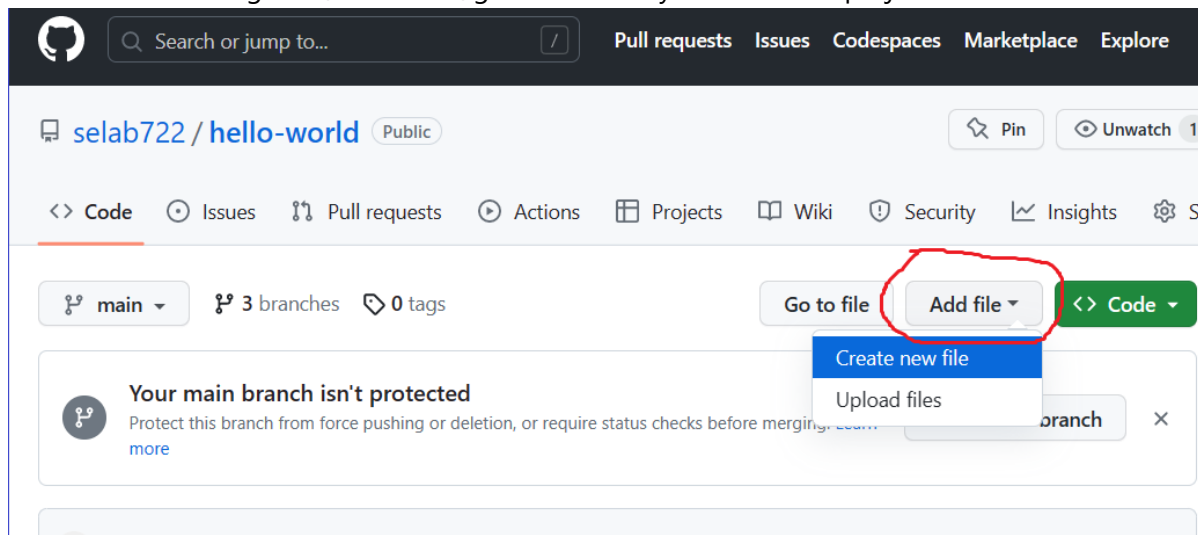
<https://www.jenkins.io/doc/pipeline/tour/hello-world/>.

## Github Actions

Github also has its own CI/CD tools. It is called "Github Actions".

Since we already created hello-world project, we add actions to this project. Follow the instructions in: <https://docs.github.com/en/actions/quickstart>.

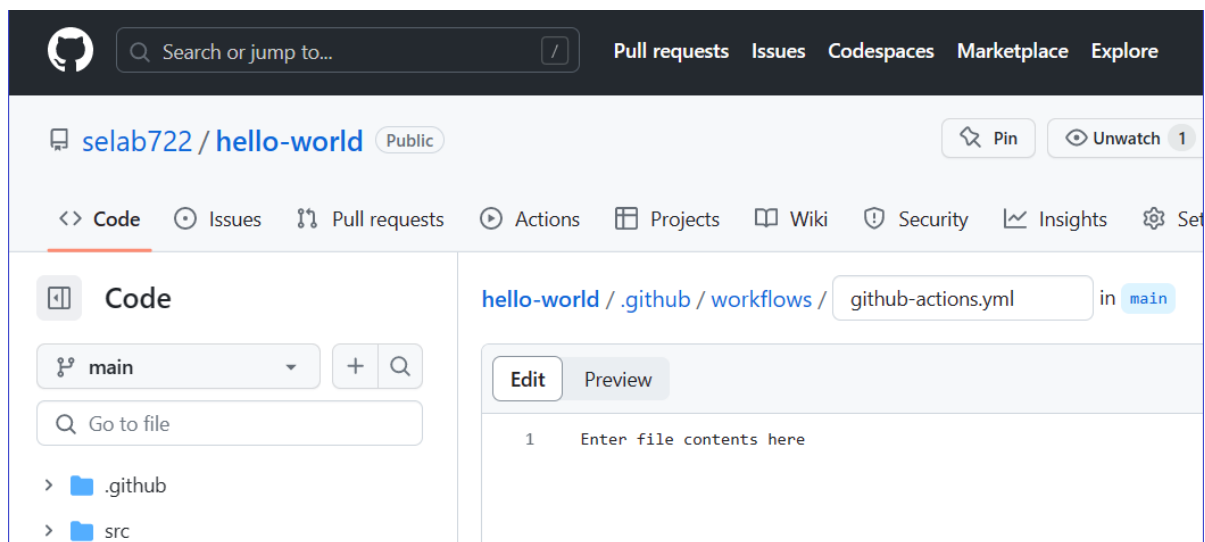
First let's create a ".github/workflows/github-actions.yml" file in the project:



The screenshot shows the GitHub web interface for the repository 'selab722 / hello-world'. The 'Code' tab is selected. In the top right area, there are buttons for 'Go to file', 'Add file', and 'Code'. The 'Add file' button is circled in red, and a dropdown menu is visible below it with options 'Create new file' and 'Upload files'. Below this, there is a notification that says 'Your main branch isn't protected'.

The file name can be different, but make sure it's under ".github/workflows" directory and it ends with ".yml":





Next, what should we put in this file? We can find a simple example of yml for Maven project here: <https://github.com/actions/starter-workflows/blob/main/ci/maven.yml>. Let's copy and paste the contents of the file to our yml file:

```
name: Java CI with Maven

on:
  push:
    branches: [ $default-branch ]
  pull_request:
    branches: [ $default-branch ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 11
        uses: actions/setup-java@v3
        with:
          java-version: '11'
          distribution: 'temurin'
          cache: maven
      - name: Build with Maven
        run: mvn -B package --file pom.xml
```

After that, commit this file to a new branch and start a pull request. This will trigger this workflow and the Github Actions will run it.

However, if you commit this file change to the default branch, you may find out that it didn't run later. You can still run it manually according to:

<https://docs.github.com/en/actions/managing-workflow-runs/manually-running-a-workflow>.

Let's modify our github-actions.yml file to:

```
name: Java CI with Maven

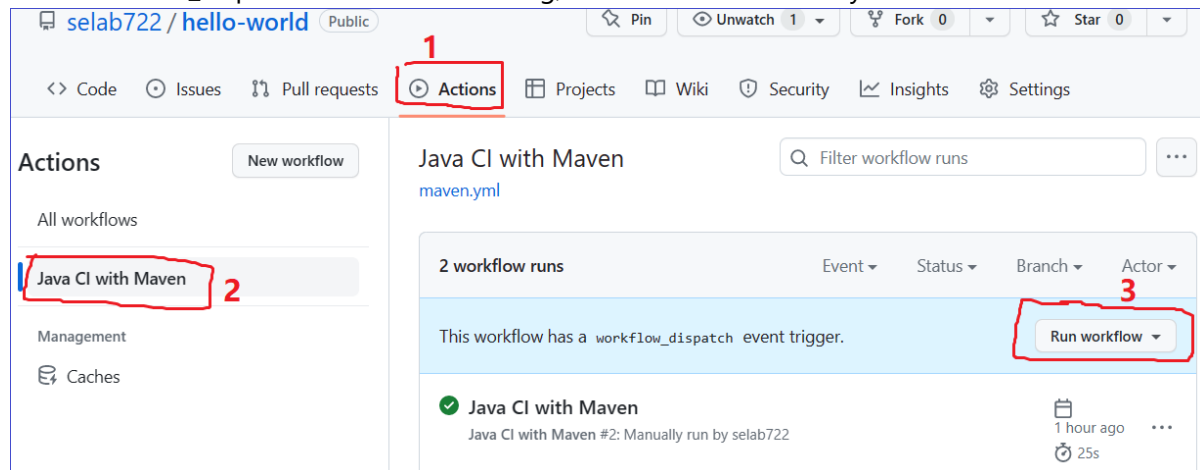
on:
  [ push, pull_request, workflow_dispatch ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 11
        uses: actions/setup-java@v3
        with:
          java-version: '11'
          distribution: 'temurin'
          cache: maven
      - name: Build with Maven
        run: mvn -B package --file pom.xml
```

With workflow\_dispatch added to the config, we are able to manually run the action:



Click and run you will see the result.