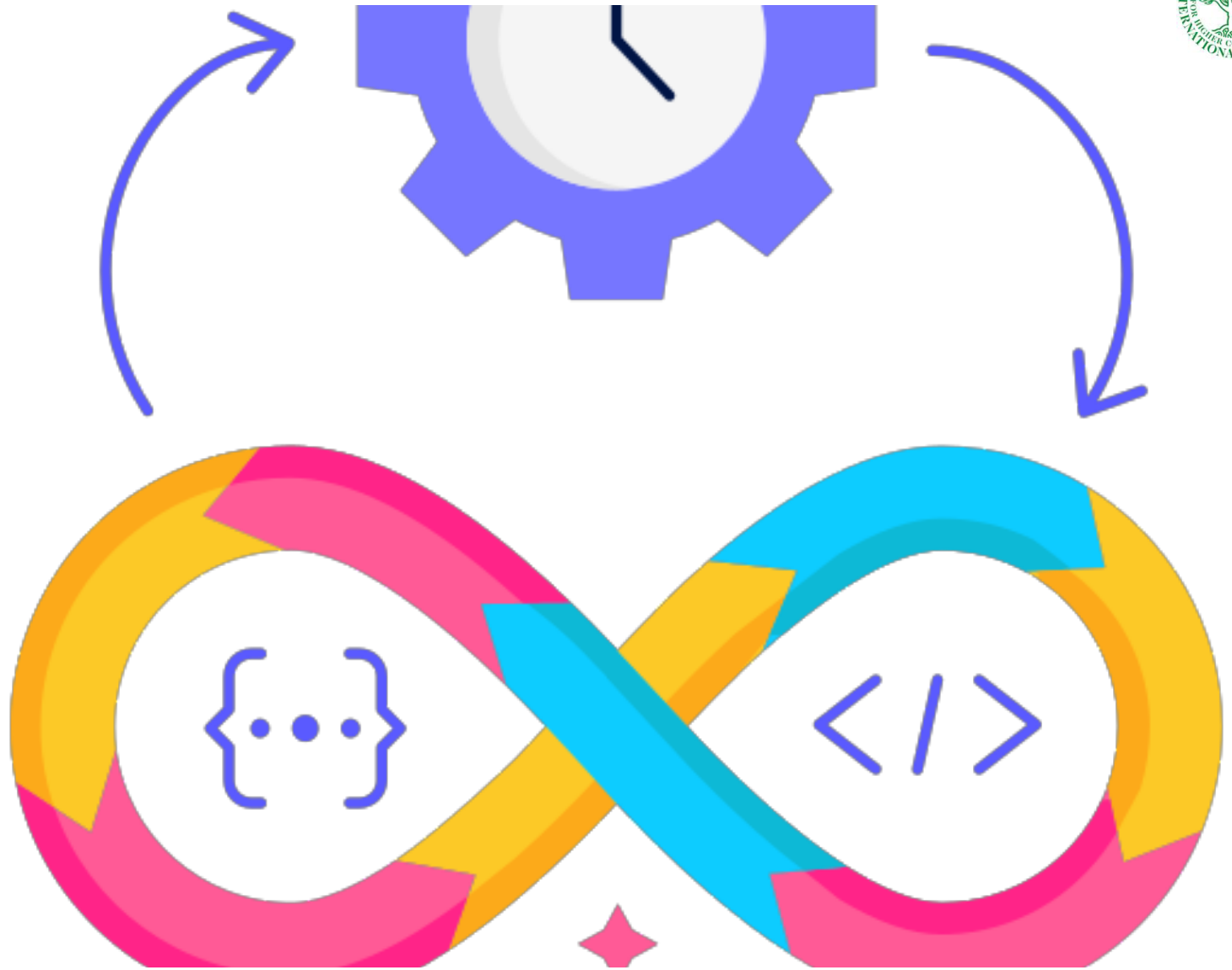
Day #2

Software Build Automation



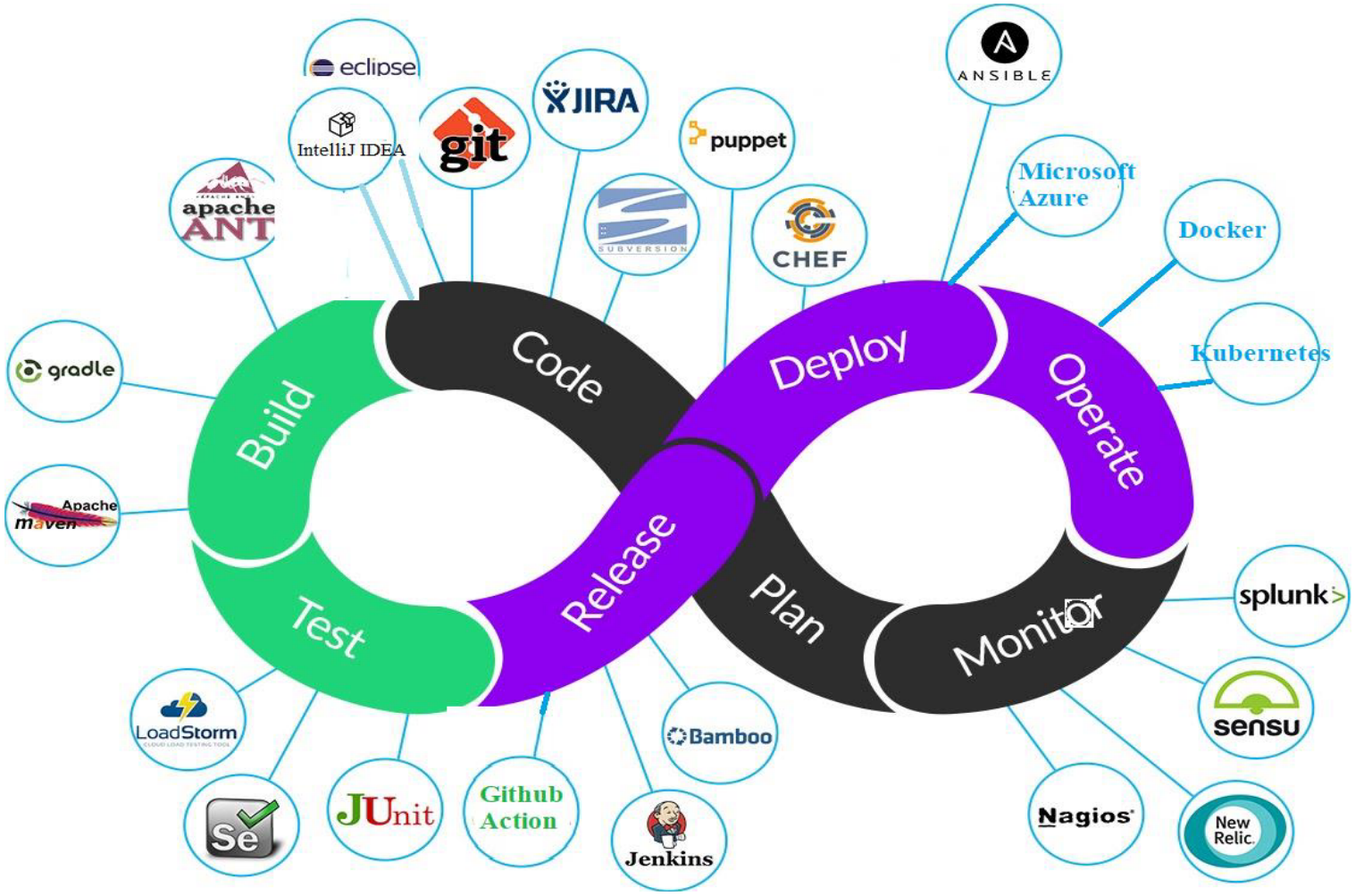
Wholeness

- Automation is the use of technology to perform tasks with reduced or no human assistance/intervention.
- Software Build constitutes the process of compiling source code, executing tests and creating a deployable unit or package/artifact.
- Science of Consciousness: Thought leads to Action. Action leads to Achievement. *Achievement leads to Fulfillment.*

Software Build Automation

- Automation of the Software build process is an essential requirement for a quick and efficient release of software product and subsequent updates.
- For modern business software products the Build -> Test -> Release cycle needs to be fast, frequent and error-free.

DevOps Life-cycle:
Plan -> Code -> Build -> Test -> Release



Build Automation Tools

Commonly used Software build automation tools:

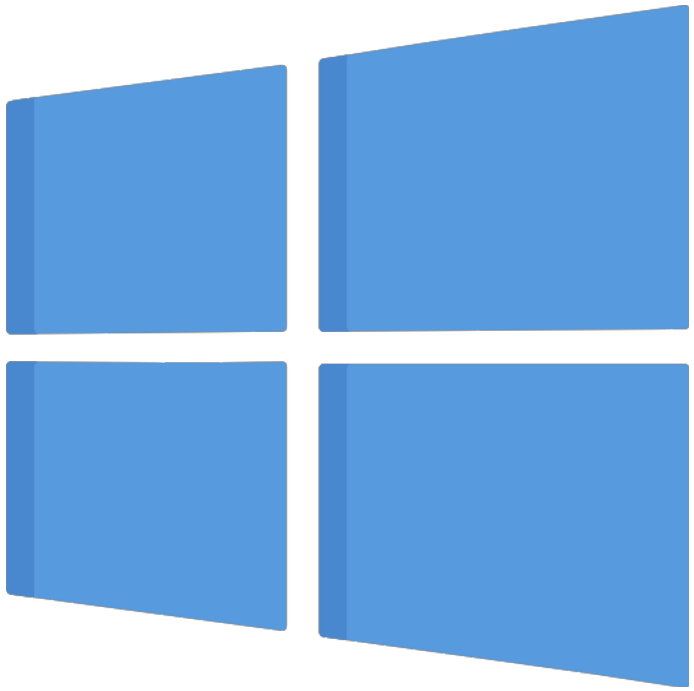
- Apache Maven
- Gradle
- Github Action
- Jenkins

Apache Maven

- Apache Maven is an open-source software build automation and dependency management tool, preeminent within the Java application development space.
- Maven uses the concept of a Project Object Model (POM) to manage the software build operation
- Every Maven project uses a pom.xml file which contains build configuration data

Download and Install Maven (Windows)

https://maven.apache.org/download.cgi



maven.apache.org/download.cgi

Apache Maven Project
http://maven.apache.org/

Apache / Maven / Download Apache Maven

- Welcome
- License

ABOUT MAVEN

- What is Maven?
- Features
- Download
- Use
- Release Notes

DOCUMENTATION

- Maven Plugins
- Maven Extensions
- Index (category)
- User Centre
- Plugin Developer Centre
- Maven Repository Centre
- Maven Developer Centre
- Books and Resources
- Security

COMMUNITY

- Community Overview
- Project Roles
- How to Contribute
- Getting Help
- Issue Management

Downloading Apache Maven 3.9.9

Apache Maven 3.9.9 is the latest release: it is the recommended version for all users.

System Requirements

Java Development Kit (JDK)	Maven 3.9+ requires JDK 8 or above to execute. It still allows you to
Memory	No minimum requirement
Disk	Approximately 10MB is required for the Maven installation itself. In a depending on usage but expect at least 500MB.
Operating System	No minimum requirement. Start up scripts are included as shell scrip

Files

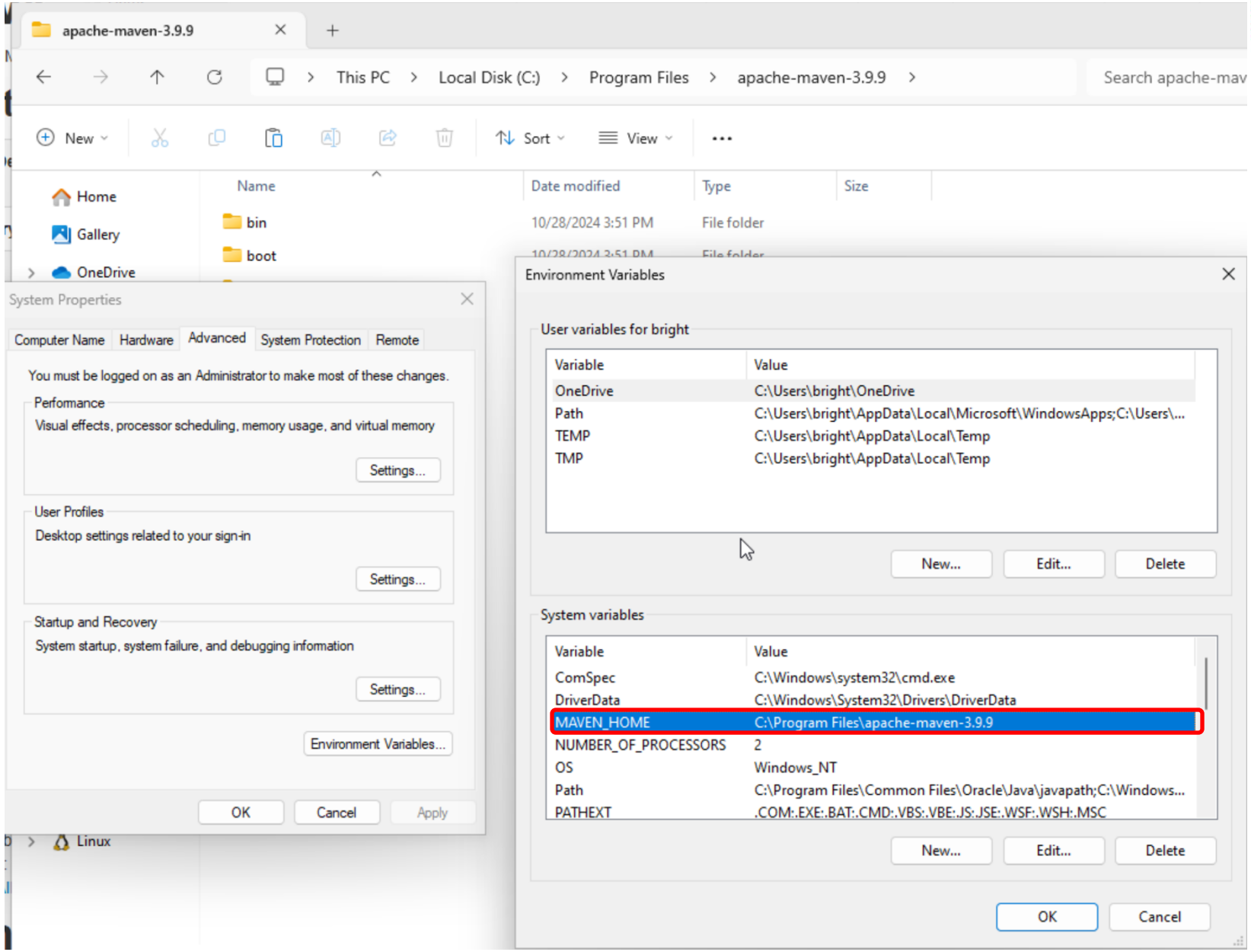
Maven is distributed in several formats for your convenience. Simply pick a ready-made binary dist

In order to guard against corrupted downloads/installations, it is highly recommended to verify the

	Link	Che
Binary tar.gz archive	apache-maven-3.9.9-bin.tar.gz	apa
Binary zip archive	apache-maven-3.9.9-bin.zip	apa
Source tar.gz archive	apache-maven-3.9.9-src.tar.gz	apa
Source zip archive	apache-maven-3.9.9-src.zip	apa

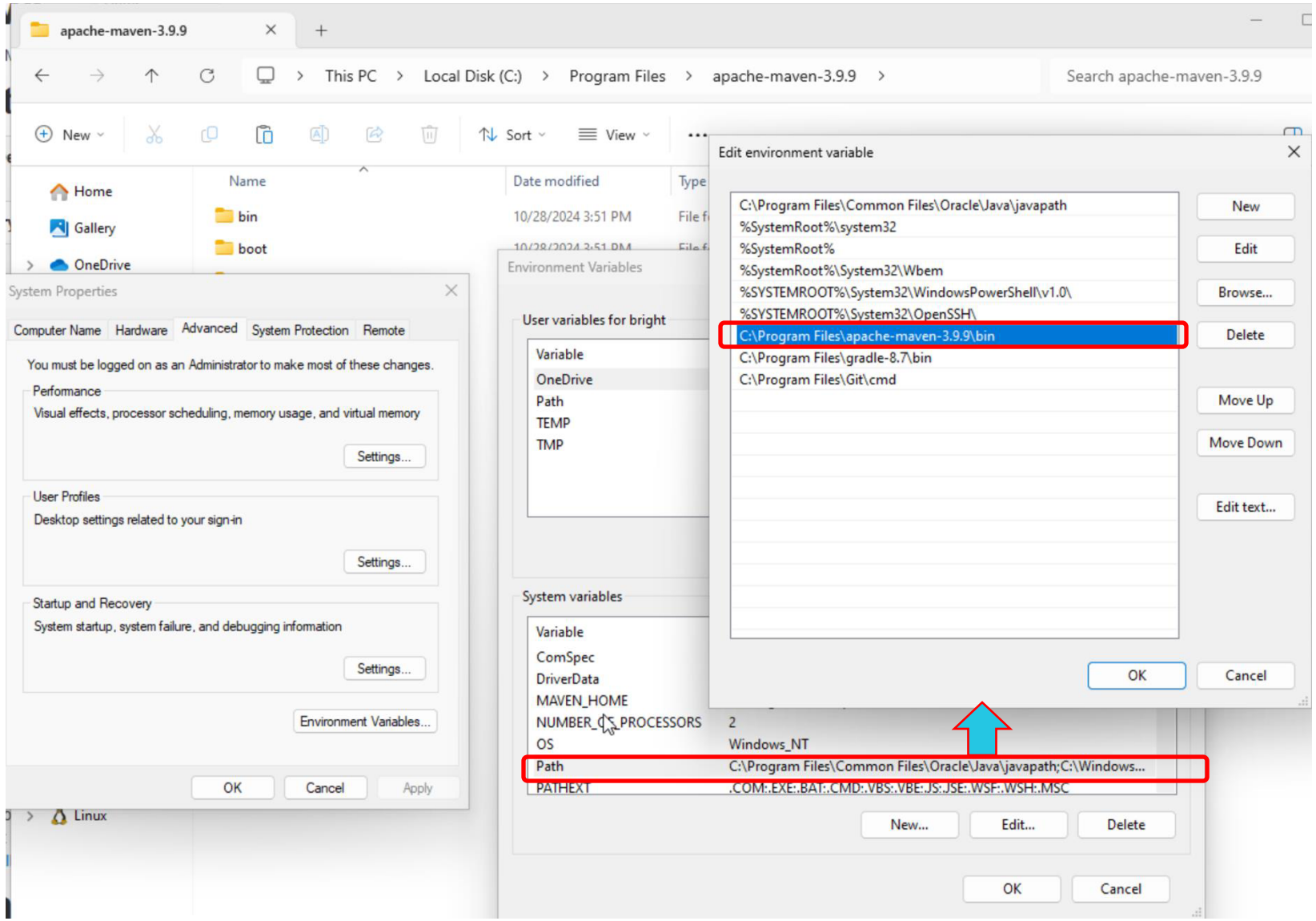
Set MAVEN_HOME

Environment Variable



Set Path

Environment Variable



Check mvn

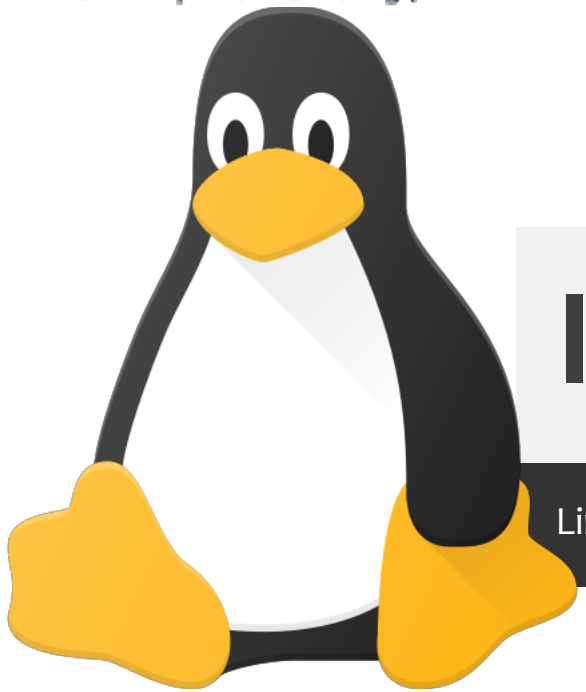
mvn -v

Install Maven

Mac OS

```
bright~$brew upgrade maven
==> Downloading https://formulae.brew.sh/api/formula.jws.json
######################################################################## 100.0%
==> Downloading https://formulae.brew.sh/api/cask.jws.json
######################################################################## 100.0%
Warning: maven 3.9.9 already installed
bright~$mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: /usr/local/Cellar/maven/3.9.9/libexec
Java version: 22, vendor: Oracle Corporation, runtime: /Library/Java/JavaVirtual
Machines/jdk-22.jdk/Contents/Home
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "15.0.1", arch: "x86_64", family: "mac"
bright~$
```

- Install
  - brew install maven
- Upgrade
  - brew upgrade maven



Install Maven

Linux

- sudo apt-get install maven

Maven Commands

mvn compile

mvn clean

mvn package

mvn install


```
bright~$pwd
/Users/bright/.m2/repository/edu/miu/cs489/exdemo/0.0.1-SNAPSHOT
bright~$ls
_remote.repositories                exdemo-0.0.1-SNAPSHOT.pom
exdemo-0.0.1-SNAPSHOT.jar           maven-metadata-local.xml
bright~$
```

Running all tests

mvn test

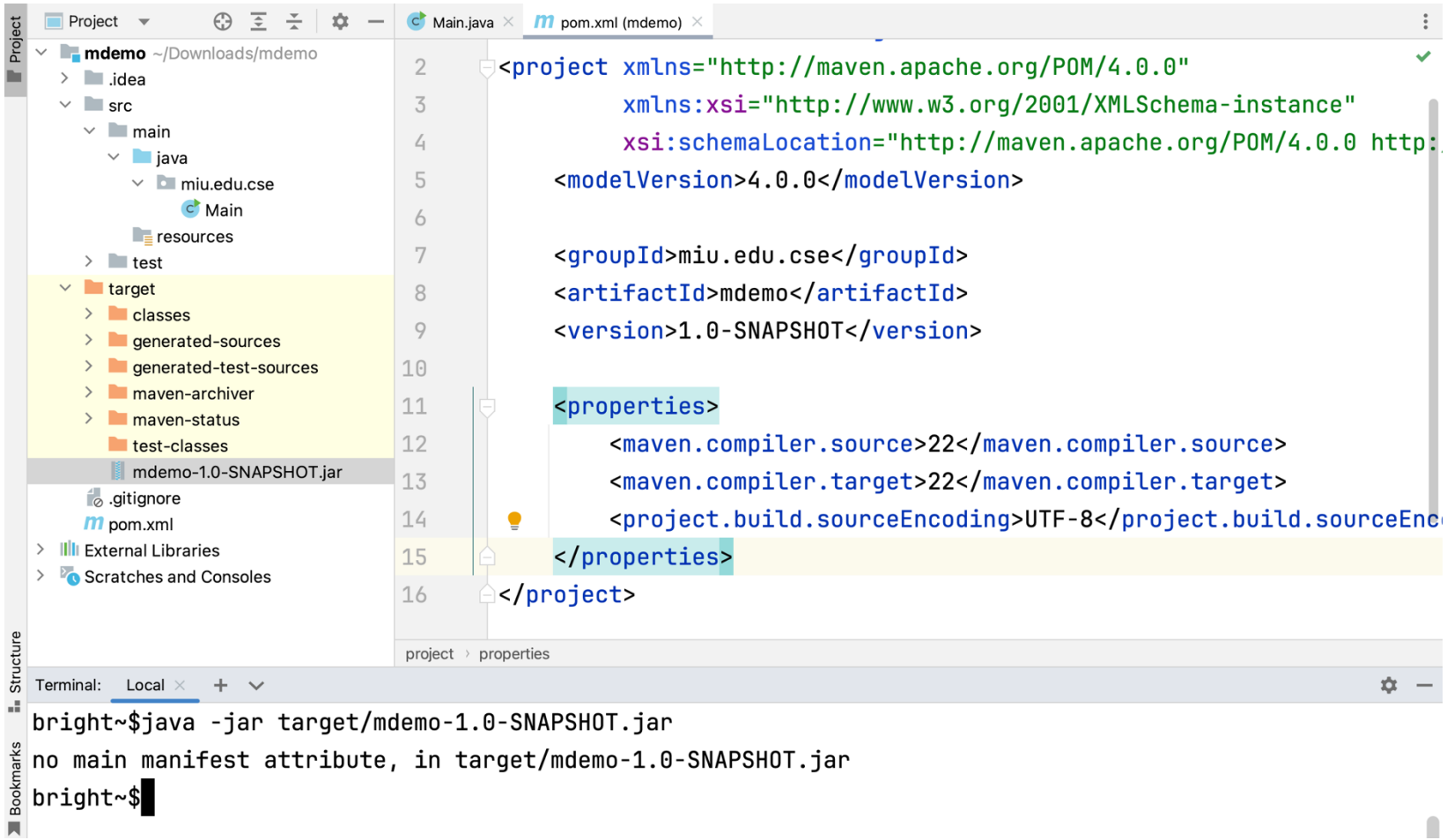
Running all tests in a class

mvn test -Dtest=MyClassName

Running single test method

mvn test -Dtest=MyClassName#myTestMethod

Fix this error



Add the following configuration in pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>edu.miu.cse</groupId>
  <artifactId>demo2</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>22</maven.compiler.source>
    <maven.compiler.target>22</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.4.2</version>
        <configuration>
          <archive>
            <manifest>
              <mainClass>
                edu.miu.cse.Main
              </mainClass>
            </manifest>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.4.2</version>
      <configuration>
        <archive>
          <manifest>
            <mainClass>
              edu.miu.cse.Main
            </mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

```
bright~$mvn clean package
…
bright~$java -jar target/mdemo-1.0-SNAPSHOT.jar
Hello world!
bright~$
```

https://gradle.org/releases/

DPE University is live! Learn more about Gradle Build Tool, Gradle Build Scan, Develocity and the ecosy

About Docs Learn Get He

Releases

Here you can find binaries and reference documentation for current and past versions of Gradle. You can find the next release candidate or a bleeding edge nightly build fo and master branches on their respective pages.

You can install Gradle through various other tools, or download a ZIP using the lir page.

Command-line completion scripts for bash and zsh can be downloaded from the gradle-completion project page.

Getting Started Resources

There are many Gradle Build Tool trainings available to help you get started quickly. The new free courses are a part of the DPE University by Gradle Inc.

The Gradle team offers free training

v8.10.2

Sep 23, 2024

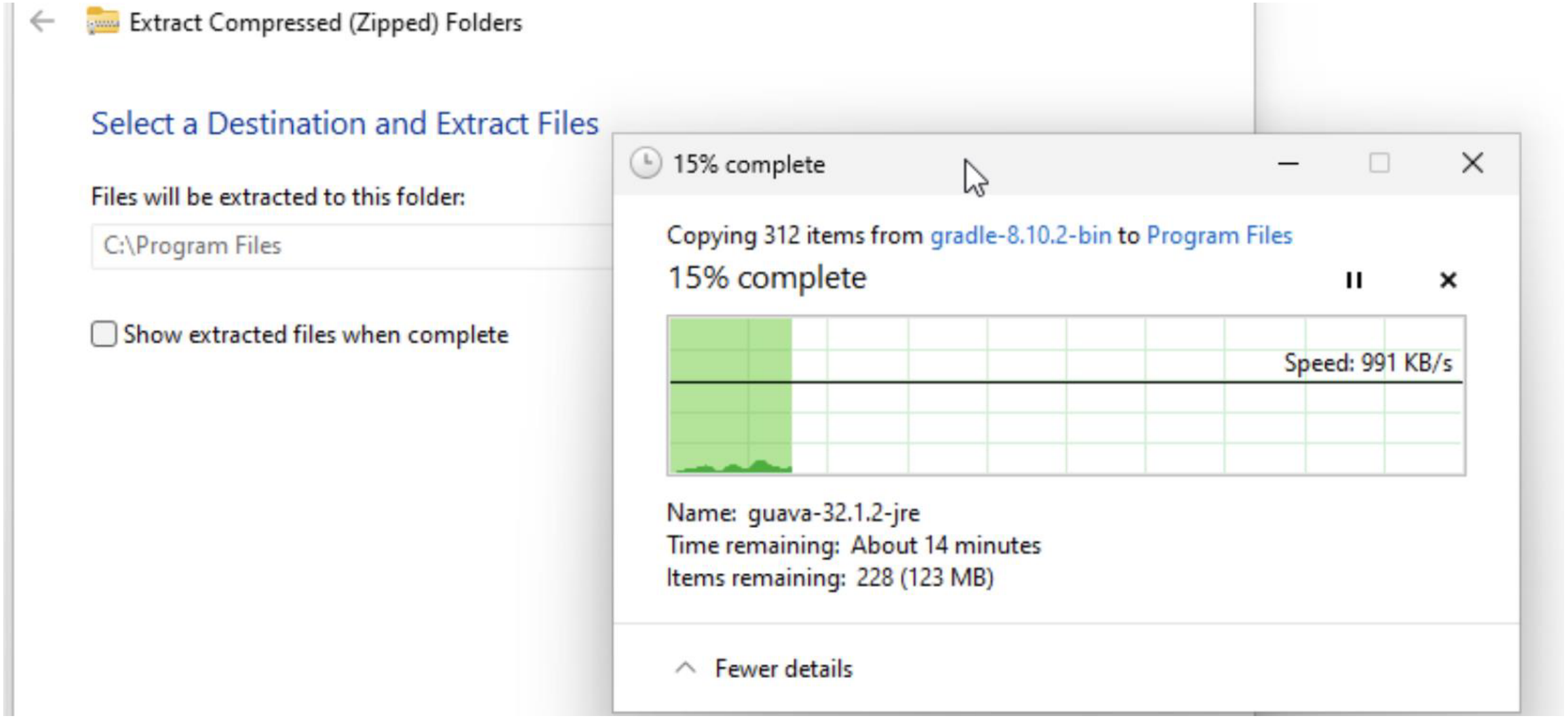
- Download: binary-only or complete (checksums)
- User Manual
- API Javadoc
- Groovy DSL Reference
- Release Notes

Install Gradle

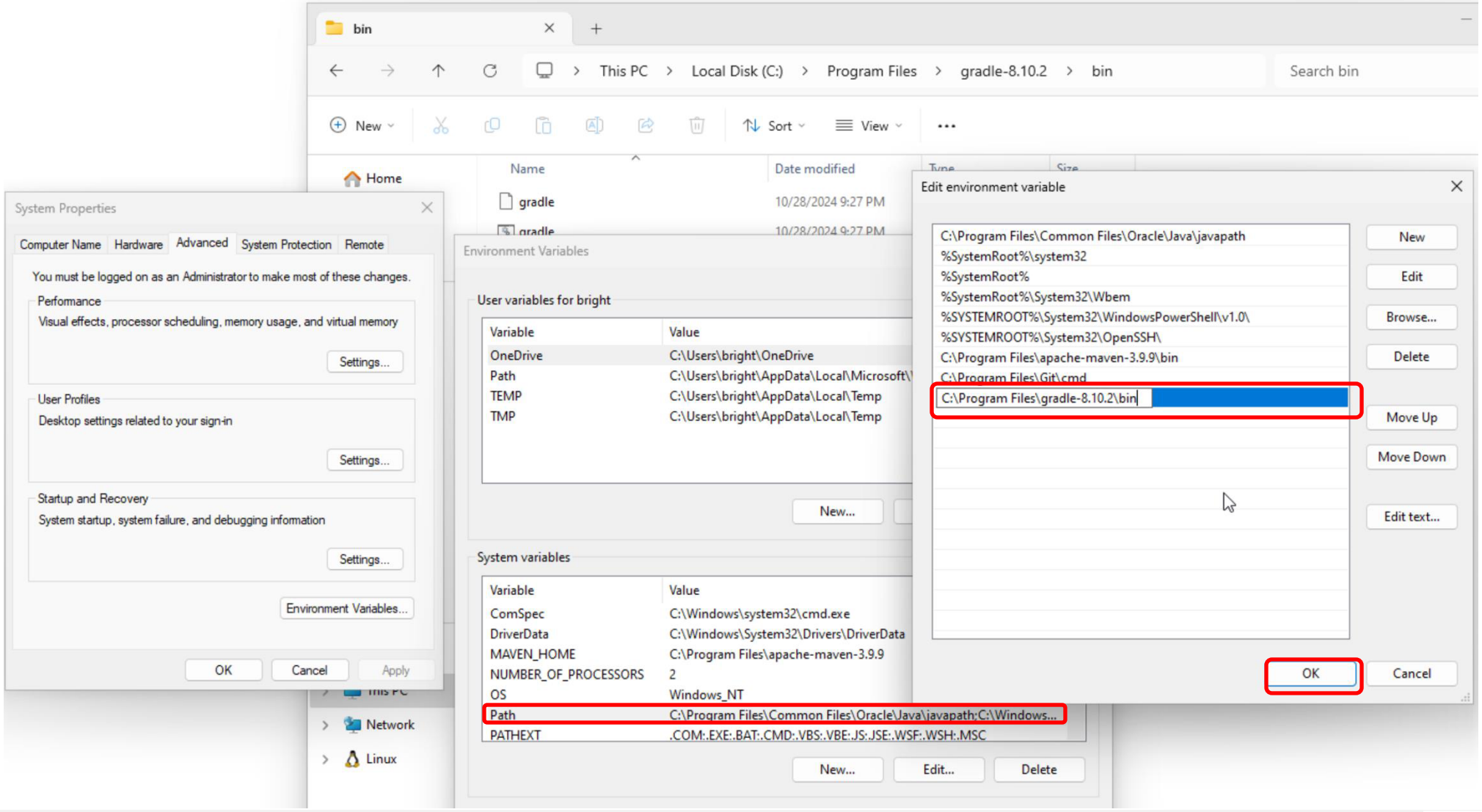
Windows

- https://gradle.org/releases/

Download & Extract it to Program Files

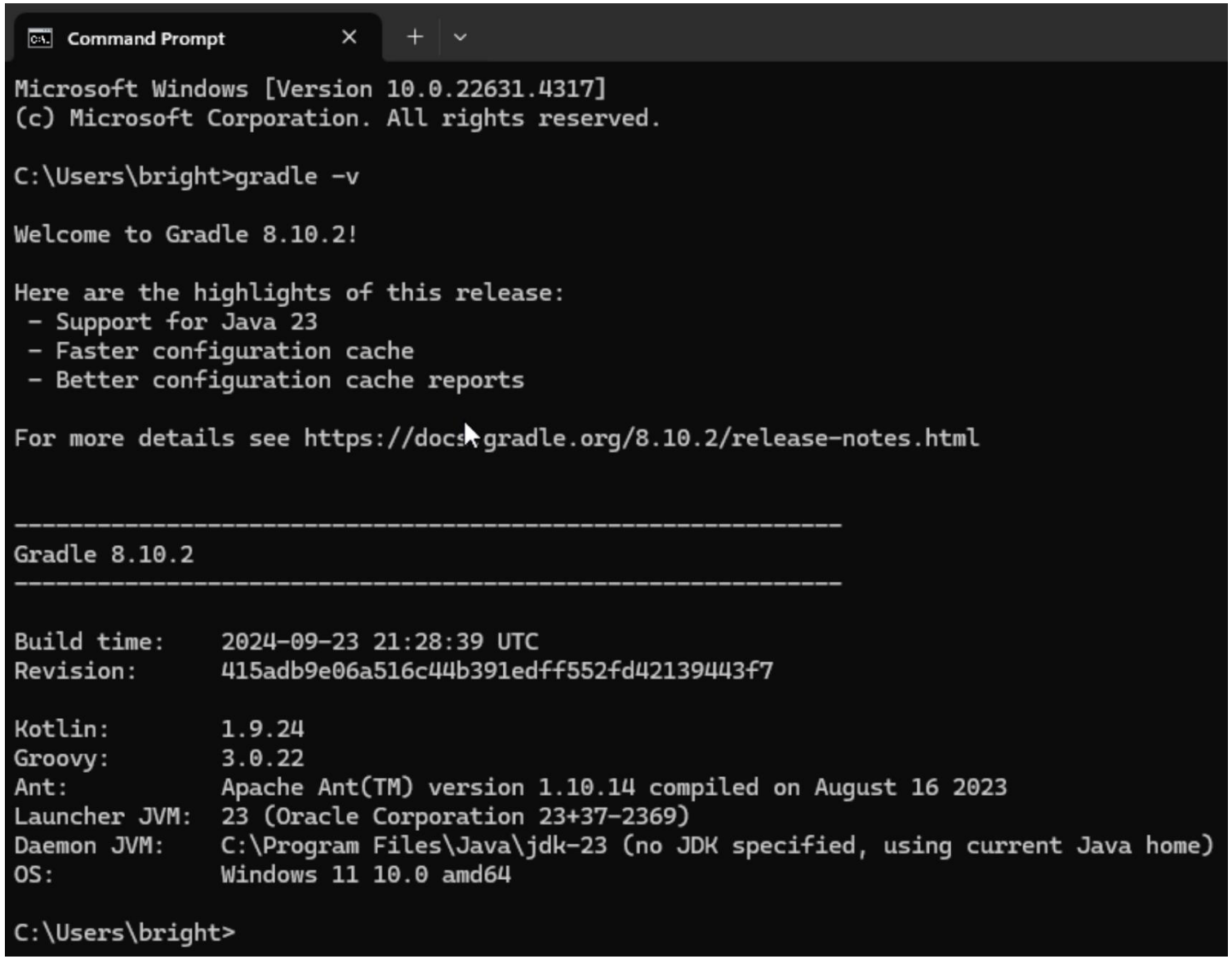


Set Environment Variables



Check the Gradle tool

gradle -v



Install Gradle on Mac

brew install gradle

If you have already installed it, upgrade it

- brew upgrade gradle

```
bright~$brew upgrade gradle
==> Downloading https://formulae.brew.sh/api/formula.jws.json
######################################################################## 100.0%
==> Downloading https://formulae.brew.sh/api/cask.jws.json
######################################################################## 100.0%
Warning: gradle 8.10.2 already installed
bright~$gradle -v

------------------------------------------------------------

Gradle 8.10.2

------------------------------------------------------------

Build time:    2024-09-23 21:28:39 UTC
Revision:      415adb9e06a516c44b391edff552fd42139443f7

Kotlin:        1.9.24
Groovy:        3.0.22
Ant:           Apache Ant(TM) version 1.10.14 compiled on August 16 2023
Launcher JVM:  22 (Oracle Corporation 22+36-2370)
Daemon JVM:    /Library/Java/JavaVirtualMachines/jdk-22.jdk/Contents/Home (no JD
K specified, using current Java home)
OS:            Mac OS X 15.0.1 x86_64

bright~$
```

Install Gradle on Linux

Manually

Step 1 - Download the latest Gradle distribution

The distribution ZIP file comes in two flavors:

•Binary-only (bin)

•Complete (all) with docs and sources

Downloading the bin file.

Step 2 - Unpack the distribution

Unzip the distribution zip file in the directory of your choosing:

```
❯ mkdir /opt/gradle
❯ unzip -d /opt/gradle gradle-8.10.2-bin.zip
❯ ls /opt/gradle/gradle-8.10.2
```

Step 3 - Configure your system environment

To install Gradle, the path to the unpacked files needs to be in your Path.

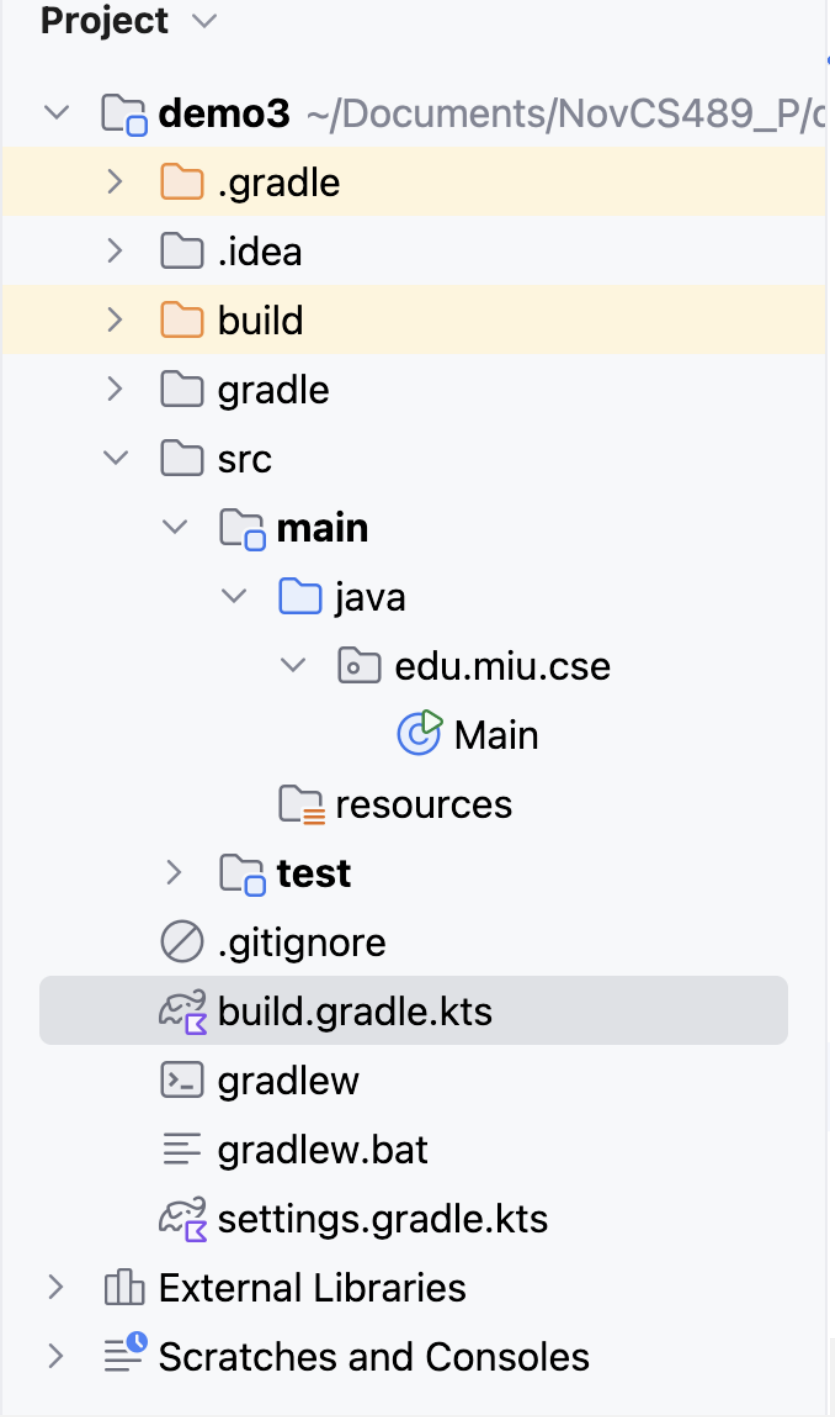
Configure your PATH environment variable to include the bin directory of the unzipped distribution,

```
❯ export PATH=$PATH:/opt/gradle/gradle-8.10.2/bin
```

Install Gradle on Linux

Installing with a package manager

```
sdk install gradle
```




```
plugins {
    id("java")
    application
}

group = "edu.miu.cse"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
}

application{
    mainClass = "edu.miu.cse.Main"
}

dependencies {
    testImplementation(platform("org.junit:junit-bom:5.10.0"))
    testImplementation("org.junit.jupiter:junit-jupiter")
}

tasks.jar{
    manifest{
        attributes["Main-Class"] = "edu.miu.cse.Main"
    }
}

tasks.test {
    useJUnitPlatform()
}
```

Run Maven

How to clean, build, and run?

./gradlew clean build

BUILD SUCCESSFUL in 1s

6 actionable tasks: 6 executed

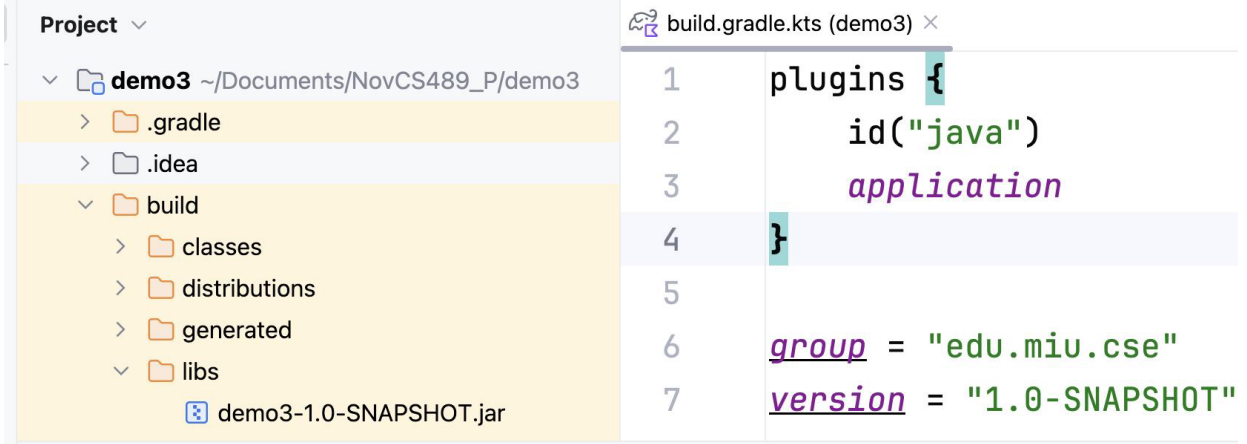
~/Documents/NovCS489_P/demo3

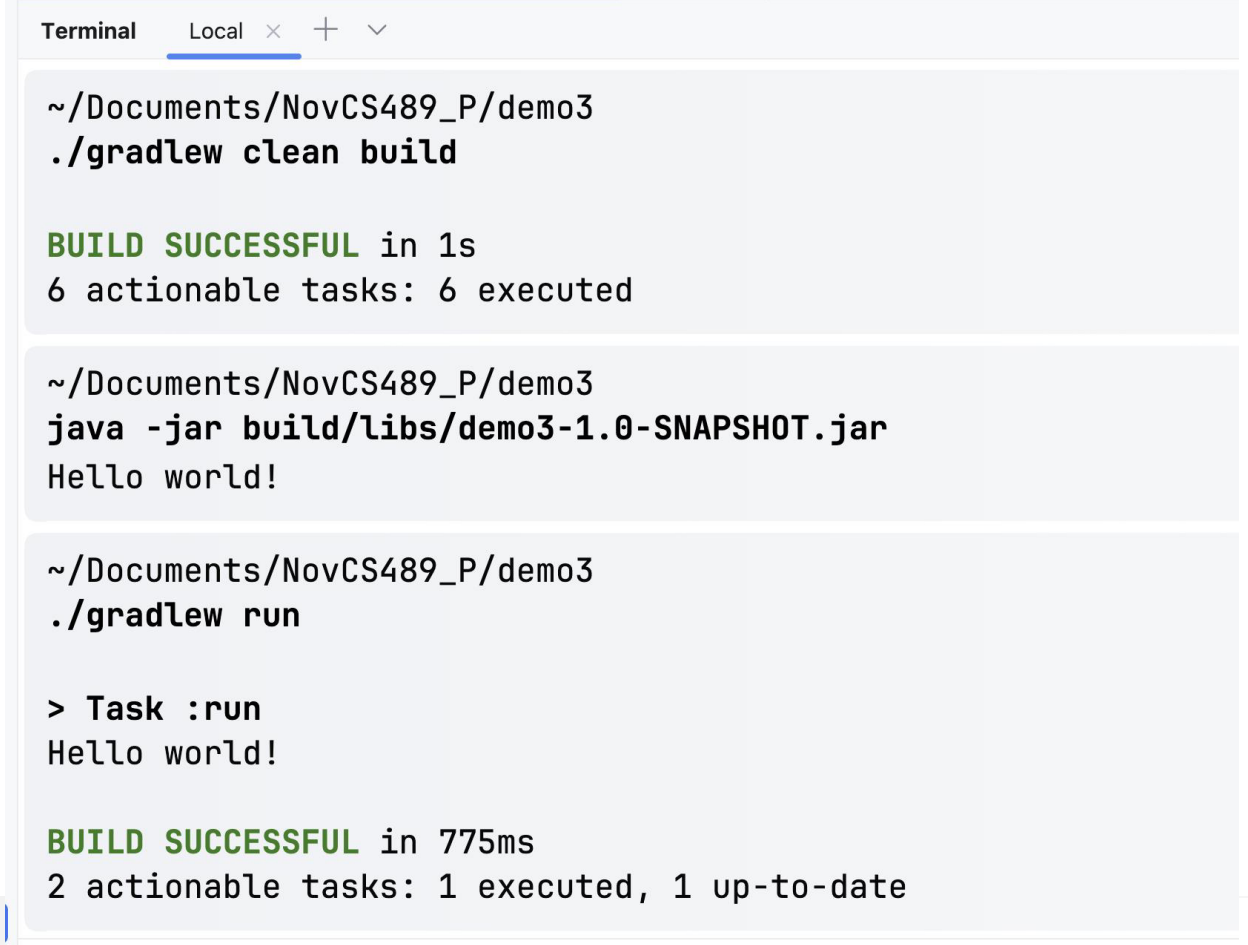
java -jar build/libs/demo3-1.0-SNAPSHOT.jar

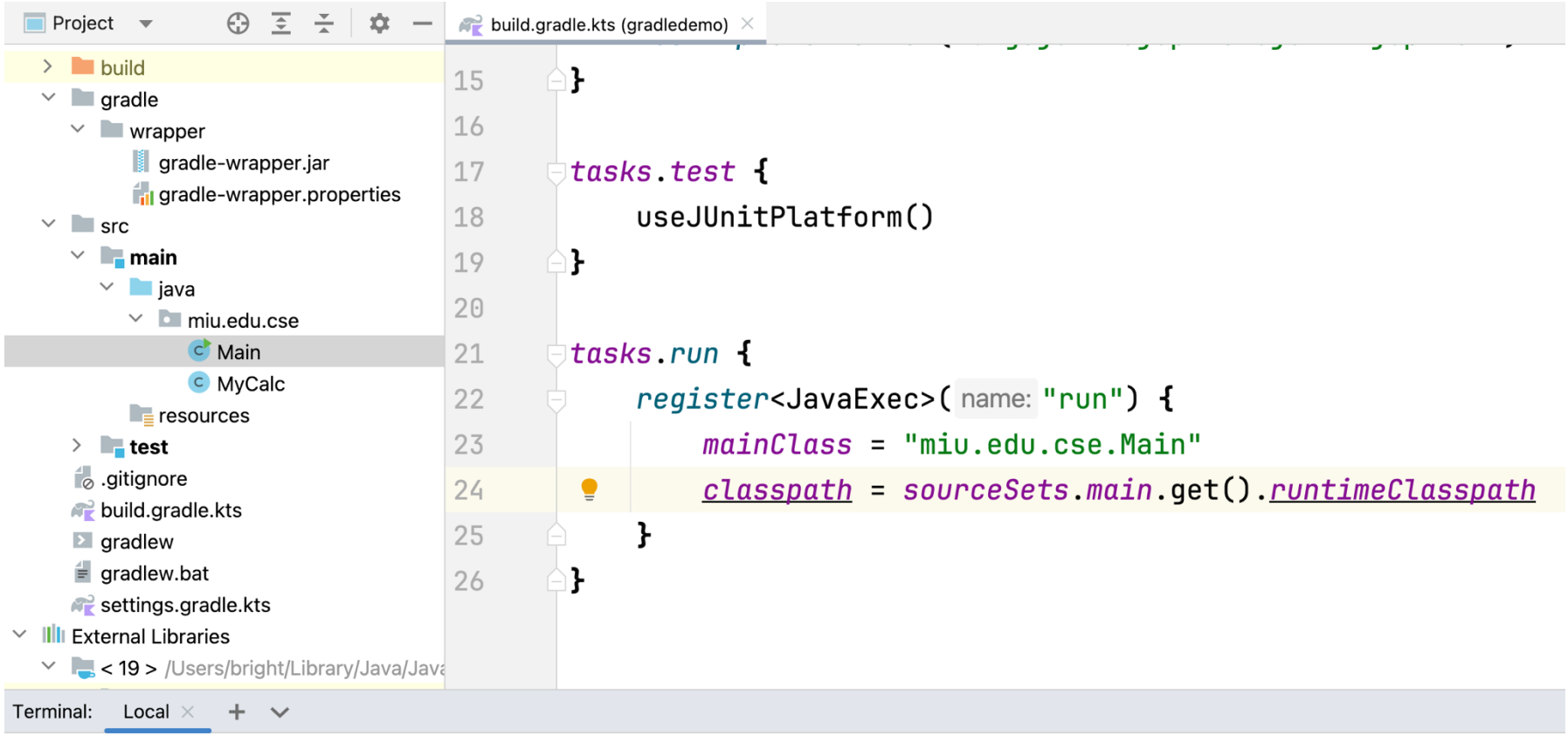
Hello world!

~/Documents/NovCS489_P/demo3

./gradlew run

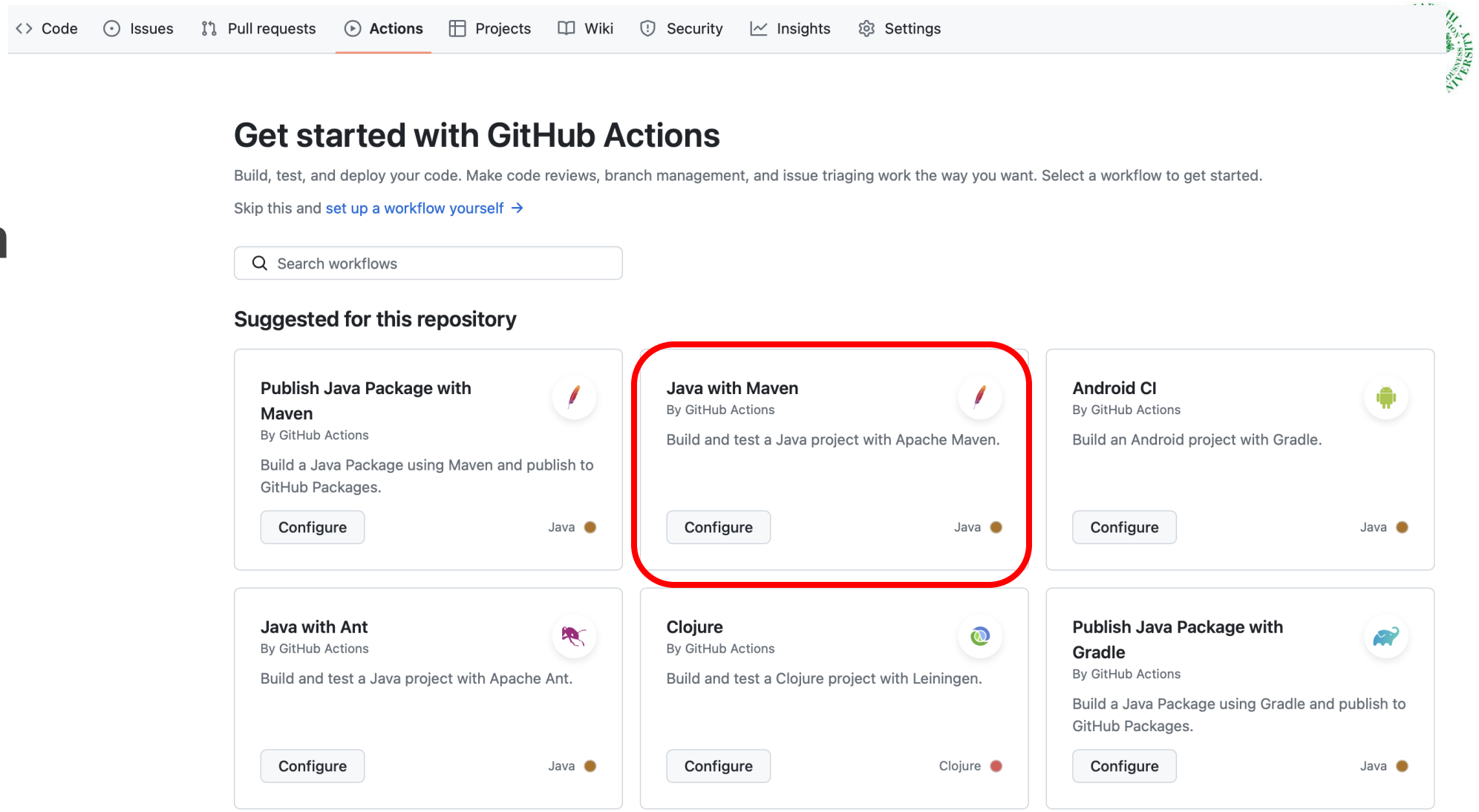





Project
build
gradle
wrapper
gradle-wrapper.jar
gradle-wrapper.properties
src
main
java
miu.edu.cse
Main
MyCalc
resources
test
.gitignore
build.gradle.kts
gradlew
gradlew.bat
settings.gradle.kts
External Libraries
< 19 > /Users/bright/Library/Java/Java
build.gradle.kts (gradledemo)
15 }
16
17 tasks.test {
18 useJUnitPlatform()
19 }
20
21 tasks.run {
22 register<JavaExec>(name: "run") {
23 mainClass = "miu.edu.cse.Main"
24 classpath = sourceSets.main.get().runtimeClasspath
25 }
26 }
Terminal: Local
bright~$./gradlew run
> Task :run
Hello and welcome!i = 1
i = 2
i = 3
i = 4
i = 5

GitHub Action

Experimenting is fun ☺



GitHub Action

Experimenting is fun ☺

mavendemo / .github / workflows / maven.yml in main

Edit Preview Code 55% faster with GitHub Copilot Spaces 2 No w

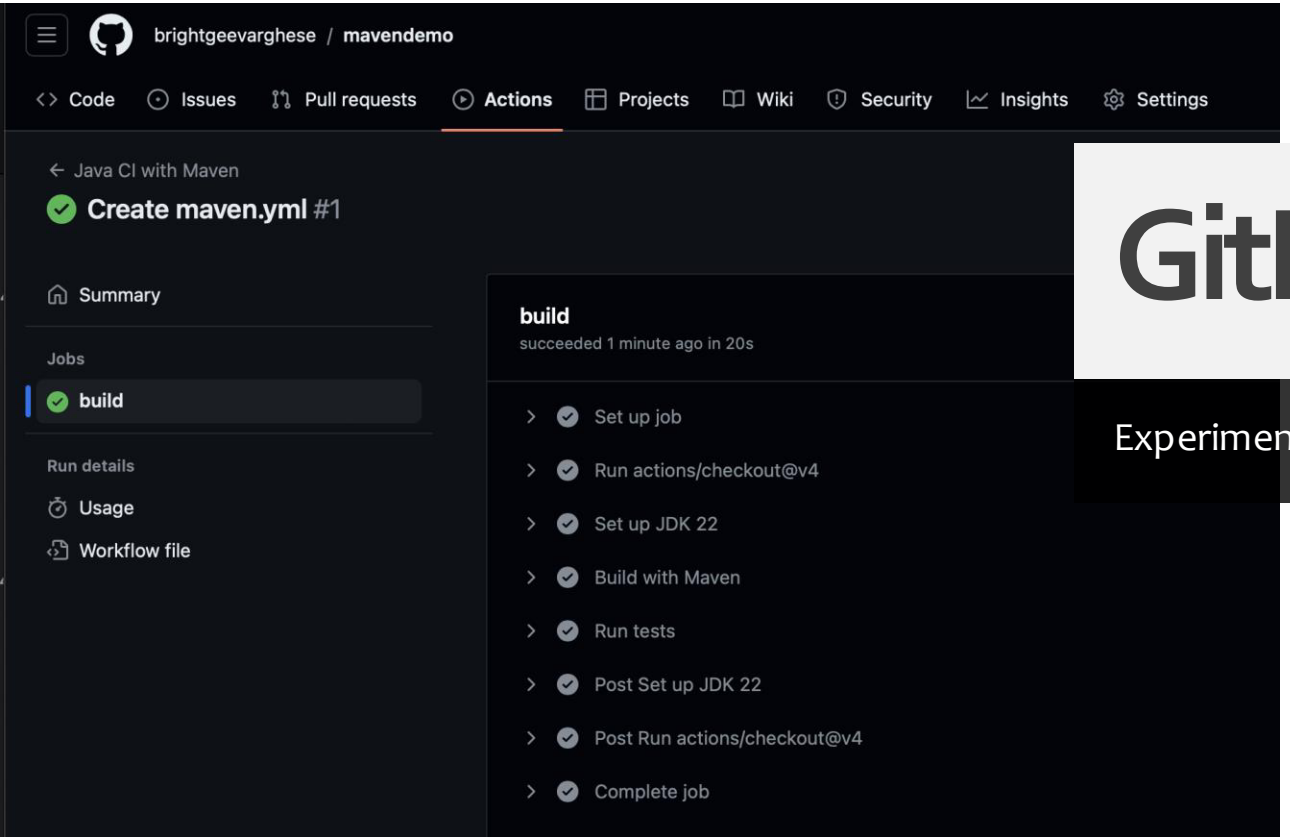
```yaml
# This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the workflow execution time
# For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-java-with-maven

# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.

name: Java CI with Maven

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4
    - name: Set up JDK 22
      uses: actions/setup-java@v3
      with:
        java-version: '22'
        distribution: 'temurin'
        cache: maven
    - name: Build with Maven
      run: mvn -B package --file pom.xml

    - name: Run tests
      run: mvn test
    # Optional: Uploads the full dependency graph to GitHub to improve the quality of Dependabot alerts this repository can receive
    # - name: Update dependency graph
    #   uses: advanced-security/maven-dependency-submission-action@571e99aab1055c2e71a1e2309b9691de18d6b7d6

```

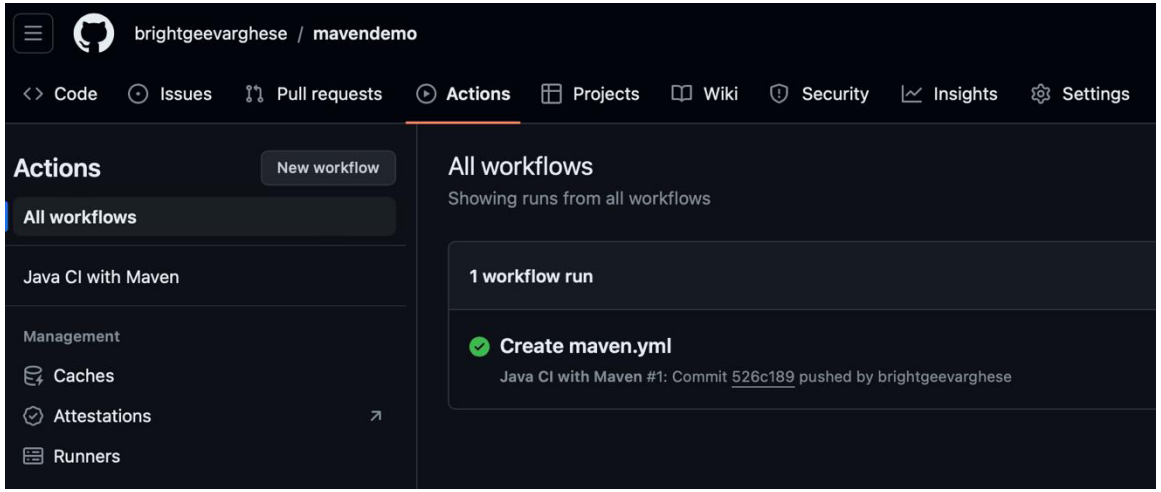
This action retrieves the project's code repository from GitHub and makes it available to the runner machine that executes the workflow.

```
...
 - name: Login to Docker registry (using secrets)
   env:
     DOCKER_USERNAME: ${{ secrets.DOCKER_USERNAME }}
     DOCKER_PASSWORD: ${{ secrets.DOCKER_PASSWORD }}
   run: docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD

 - name: Build Docker image
   run: docker build -t my-java-app:latest .  # Replace "my-java-app" with your desired image name

 - name: Push Docker image to registry (Optional)
   run: docker push my-java-app:latest
```

GitHub Action

Experimenting is fun



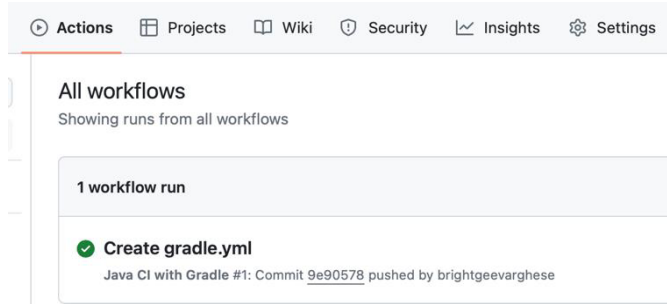


gdemo1 / .github / workflows / gradle.yml

brightgeevarghese Create gradle.yml ✓

Code | Blame | 71 lines (57 loc) · 2.37 KB | Code 55% faster with GitHub Copilot

```yaml
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.
# This workflow will build a Java project with Gradle and cache/restore any dependencies to improve the wor
# For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-tes

name: Java CI with Gradle

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:

    runs-on: ubuntu-latest
    permissions:
      contents: read

    steps:
    - uses: actions/checkout@v4
    - name: Set up JDK 19
      uses: actions/setup-java@v4
      with:
        java-version: '19'
        distribution: 'temurin'

    # Configure Gradle for optimal use in GitHub Actions, including caching of downloaded dependencies.
    # See: https://github.com/gradle/actions/blob/main/setup-gradle/README.md
    - name: Setup Gradle
      uses: gradle/actions/setup-gradle@417ae3ccd767c252f5661f1ace9f835f9654f2b5 # v3.1.0

    - name: Build with Gradle Wrapper
      run: ./gradlew build
    - name: Test with Gradle Wrapper
      run: ./gradlew test
```



Requirements for self-hosted runner machines

https://docs.github.com/en/actions/hosting-your-own-runners/managing-self-hosted-runners/about-self-hosted-runners

```
<?xml version="1.0" encoding="UTF-8"?>
<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>miu.edu.cse</groupId>
   <artifactId>demo</artifactId>
   <version>1.0-SNAPSHOT</version>

   <properties>
      <maven.compiler.source>22</maven.compiler.source>
      <maven.compiler.target>22</maven.compiler.target>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   </properties>
```

```
<dependencies>

    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter</artifactId>
      <version>5.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

```
<build>
    <plugins>
<!--        compiling Java source code within a Maven project-->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.11.0</version>
            <configuration>
              <enablePreview>true</enablePreview>
            </configuration>
        </plugin>
<!--        for executing the JUnit tests during the Maven build process.-->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.2.5</version>
        </plugin>
    </plugins>
  </build>
</project>
```