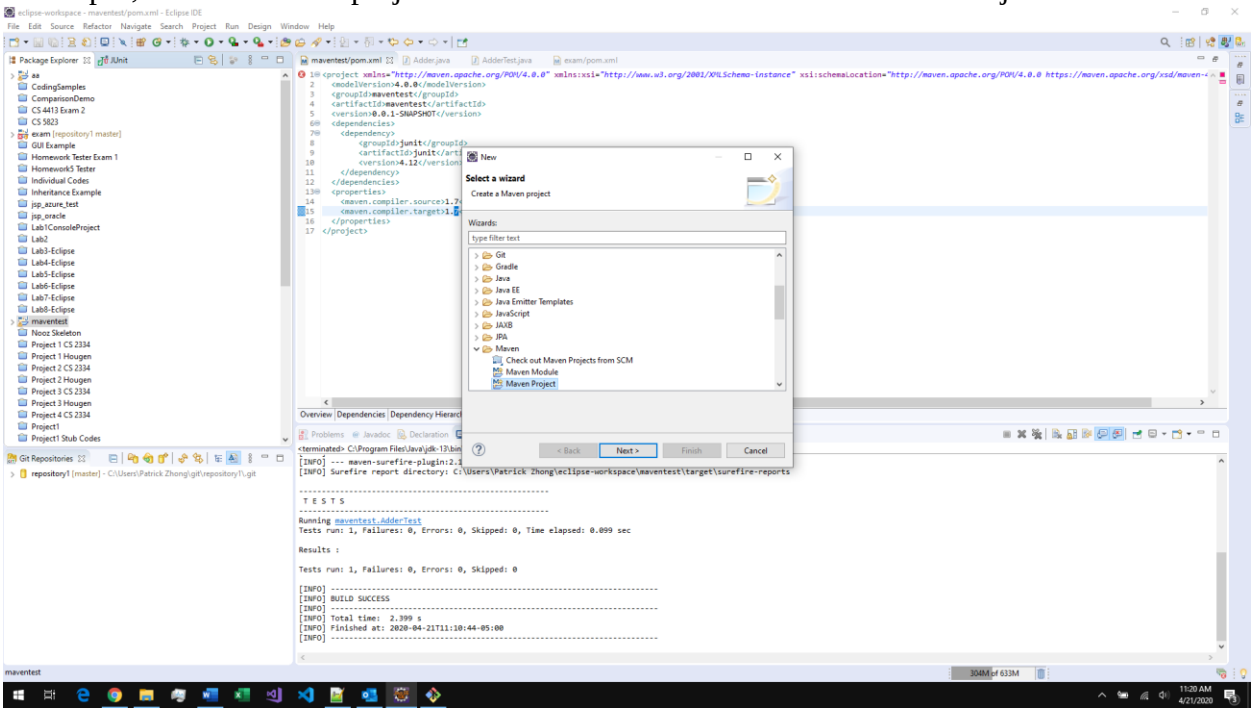
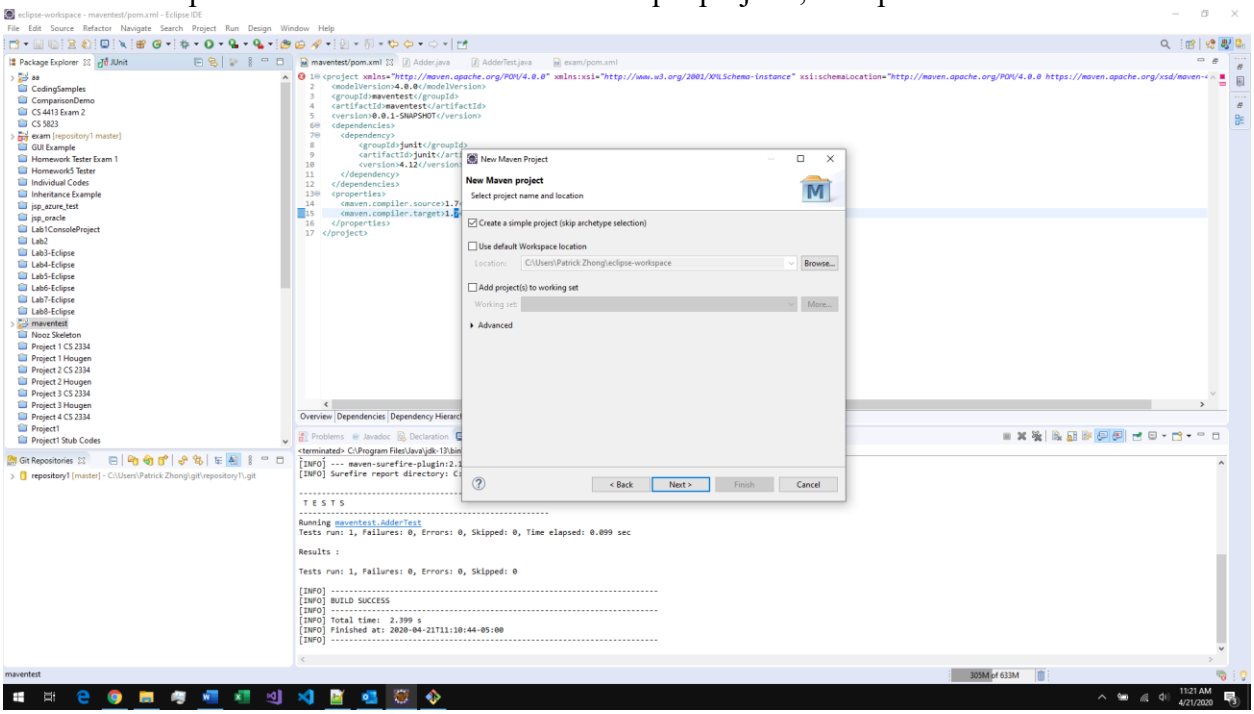
Github Hosted Maven

1. Install Git and make a GitHub account.
2. Install Eclipse, follow instructions here: https://www.eclipse.org/downloads/packages/installer.
3. In Eclipse, create a Maven project. Select File -> New -> Other -> Maven Project



4. Select a workspace location and check ‘Create simple project’, then press next



5. Enter a group id and an artifact Id. Then press Finish

6. Open the pom.xml file. Add JUnit 4 as a dependency for your Maven Project in you pom.xml file and set the compiler. To do this insert this after the </version> tag and before the </project> tag:

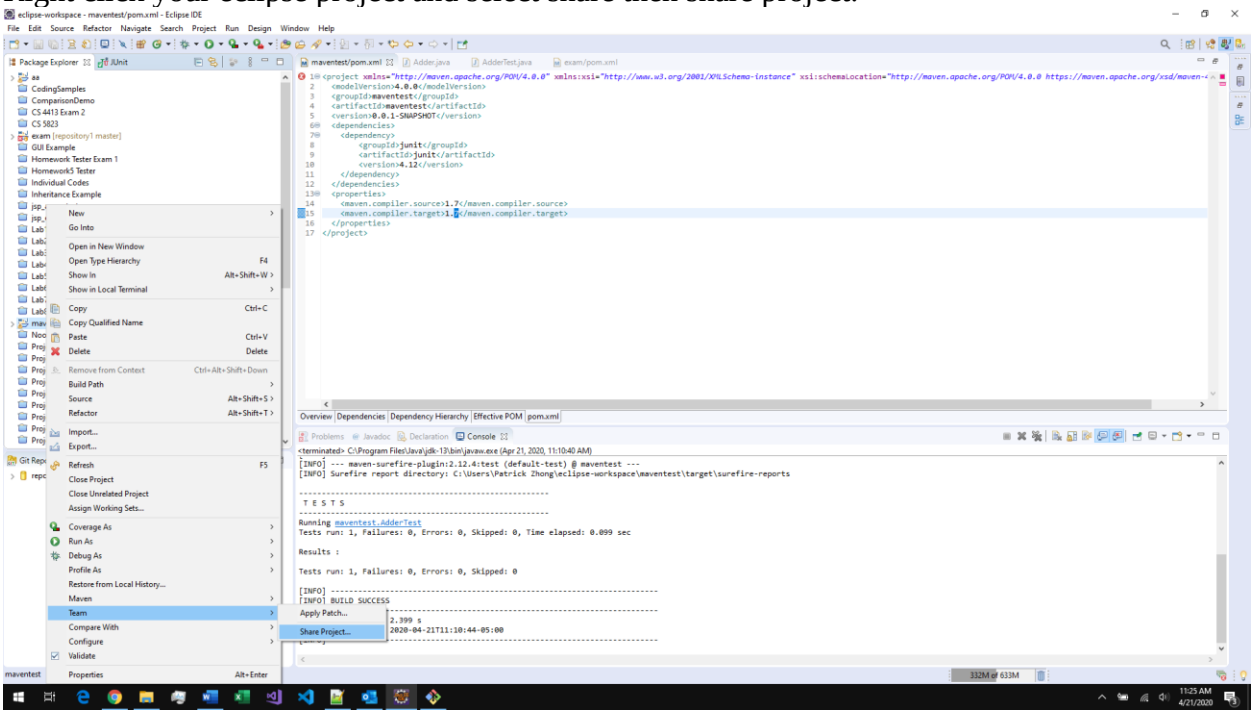
```
  <dependencies>
        <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.12</version>
        </dependency>
  </dependencies>
  <properties>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
```

   Final result for the pom.xml file should look similar to this, the information above the dependencies tag may be different:

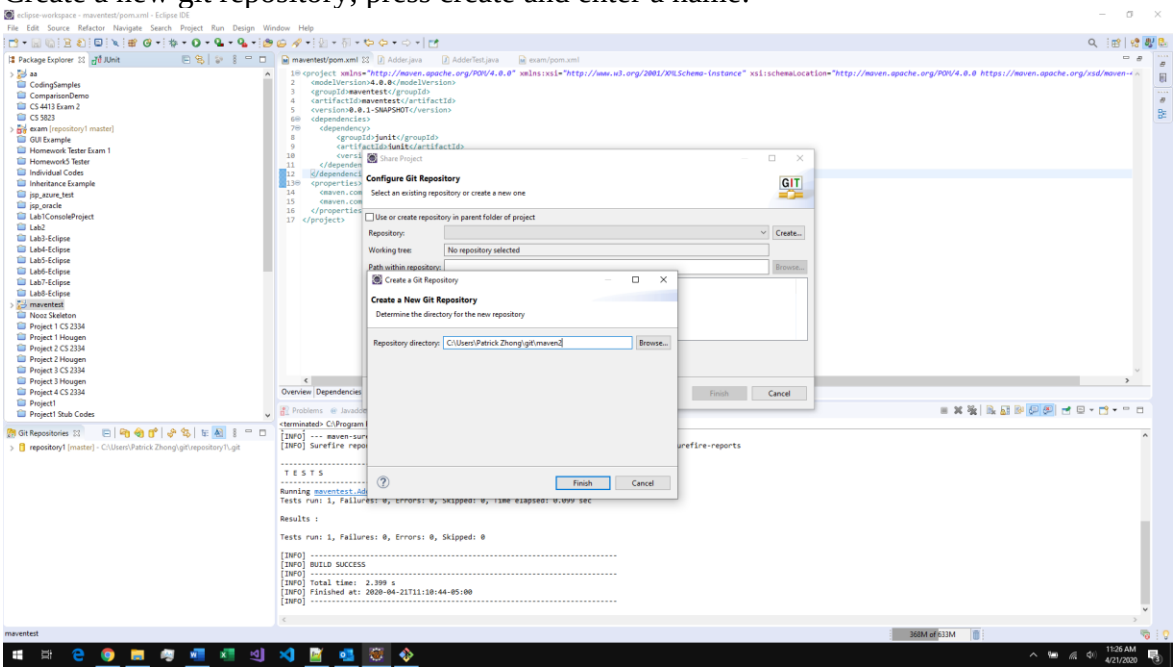
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>maventest</groupId>
  <artifactId>maventest</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
        <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.12</version>
        </dependency>
  </dependencies>
  <properties>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
</project>
```

7. Put your Maven Project into Git.

8. Right click your eclipse project and select share then share project.



9. Create a new git repository, press create and enter a name:



10. Create a new GitHub repository and add it as a remote for your local git repo.
11. Go to Github and select actions.
12. Go to ‘Setup up a workflow yourself’
13. Create maven.yml and add this content:

```
# This workflow will build a Java project with Maven
```

```
# For more information see:
https://help.github.com/actions/language-and-framework-
guides/building-and-testing-java-with-maven

name: Java CI with Maven

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 1.8
      uses: actions/setup-java@v1
      with:
        java-version: 1.8
    - name: Build with Maven
      run: mvn -B package --file "exam/pom.xml"
```

14. Change “exam/pom.xml” to NAME_OF_YOUR_MAVEN_PROEJCT/pom.xml
15. Press Start commit and it should work.

Reference https://help.github.com/en/actions/language-and-framework-guides/building-and-testing-java-with-maven.

GCP

Running on GCP:

1. Create an Ubuntu 18.04 VM on GCP and install Maven using 'sudo apt install maven'.
2. Follow instructions here to add a GCP VM to your repo: https://help.github.com/en/actions/hosting-your-own-runners/adding-self-hosted-runners. Run the application.
3. Copy the .yml file from the section 'Github hosted Maven'. and start creating a new GitHub Action using Actions ->set up a workflow yourself -> and enter the text there.
4. Change the line 'runs-on: ubuntu-latest' to 'runs-on: self-hosted'. This makes your GCP VM run the actions.
5. Your .yml file should similar to .github/workflows/mavenGCP.yml:

```
# This workflow will build a Java project with Maven
# For more information see: https://help.github.com/actions/language-and-framework-guides/building-and-testing-java-with-maven

name: Java CI with Maven - GCP

on:
```

```
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:

    runs-on: self-hosted

    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 1.8
      uses: actions/setup-java@v1
      with:
        java-version: 1.8
    - name: Build with Maven
      run: mvn -B package --file "exam/pom.xml"
```

6. Commit the workflow file.
OPTIONAL. You can configure the runner application as a service, see second reference below.
Reference:
- https://help.github.com/en/actions/hosting-your-own-runners/about-self-hosted-runners
- https://help.github.com/en/actions/hosting-your-own-runners/configuring-the-self-hosted-runner-application-as-a-service

GCP Docker

Docker GCP:
1. Install Docker on your VM. See: https://docs.docker.com/engine/install/ubuntu/
2. Copy the text from the .yml file from the section 'Running on GCP' and start creating a new GitHub Action using Actions ->set up a workflow yourself -> and enter the text there.
3. Change the line '- name: Build with Maven' to '- name: Docker build'
4. Change the line 'run: mvn -B package --file "exam/pom.xml"' to 'run: docker build -t demo .'
5. Your .yml should now look like .github/workflows/mavenDockerGCP.yml:

```
# This workflow will build a Java project with Maven
# For more information see: https://help.github.com/actions/language-and-framework-guides/building-and-testing-java-with-maven

name: Java CI with Maven Docker GCP

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:

    runs-on: self-hosted

    steps:
    - uses: actions/checkout@v2
```

```
    - name: Set up JDK 1.8
      uses: actions/setup-java@v1
      with:
        java-version: 1.8
    - name: Docker build
      run: docker build -t demo .
```

6. Press start commit.
7. Add a Dockerfile to the project with these contents, changing 'exam' to the name of your maven project:

```
FROM maven:3.6.0-jdk-11-slim AS build
COPY exam/src /home/app/src
COPY exam/pom.xml /home/app
RUN mvn -f /home/app/pom.xml test
```

Reference:
- https://hub.docker.com/_/maven
- https://docs.docker.com/engine/reference/builder/
- https://help.github.com/en/actions