DEVOPS LAB MANUAL

CAMBRIDGE INSTITUTE OF TECHNOLOGY

AN AUTONOMOUS INSTITUTION AFFILIATED TO VTU

K.R.Puram, Bangalore-560 036

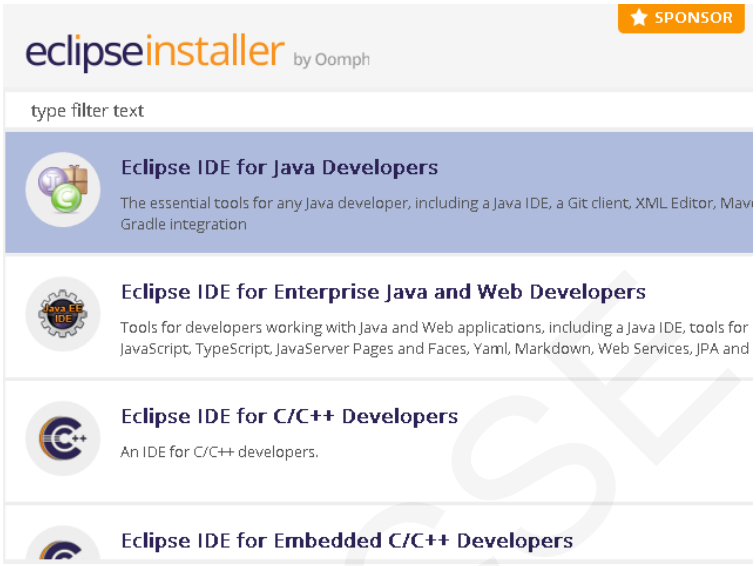
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Program 1 : Overview of Build Automation Tools, Key Differences Between Maven and Gradle, Installation and Setup

STEP 1 : Download Eclipse from the official website:

https://www.eclipse.org/downloads/

Choose "Eclipse IDE for Java Developers" or "Eclipse IDE for Enterprise Java and Web Developers".



STEP 2: Eclipse Installation Setup

Select the appropriate installation folder and Java version during setup.

Complete the installation process.

STEP 3: Launch Eclipse

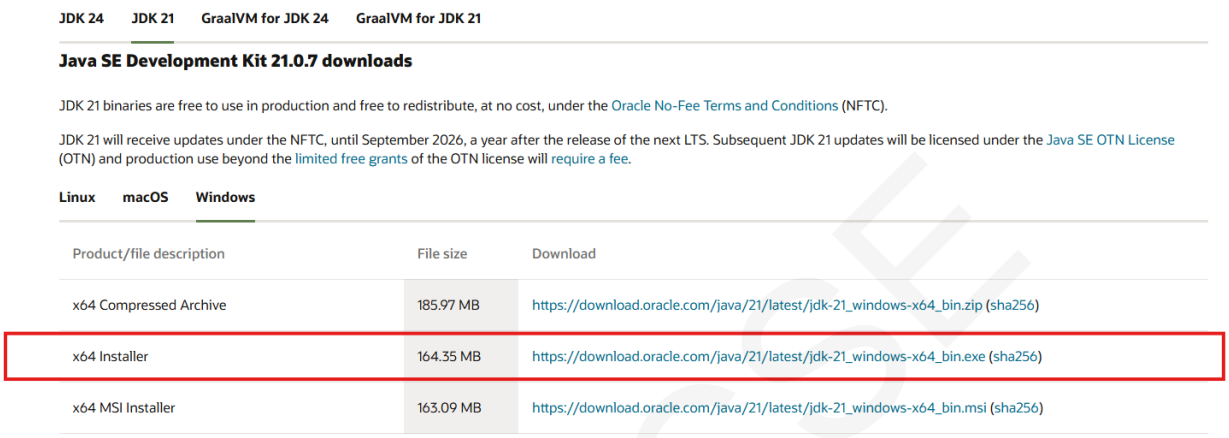
Once installed, open Eclipse. The welcome screen confirms successful installation.

STEP 4: Installing Maven and Gradle — Preparation
(a) Ensure JDK is Installed:

Visit: https://www.oracle.com/java/technologies/downloads

Install the latest JDK and configure the environment variable:

Set JAVA_HOME to the JDK path.
Ensure the java command is accessible from the terminal/command prompt



JDK 24 | JDK 21 | GraalVM for JDK 24 | GraalVM for JDK 21

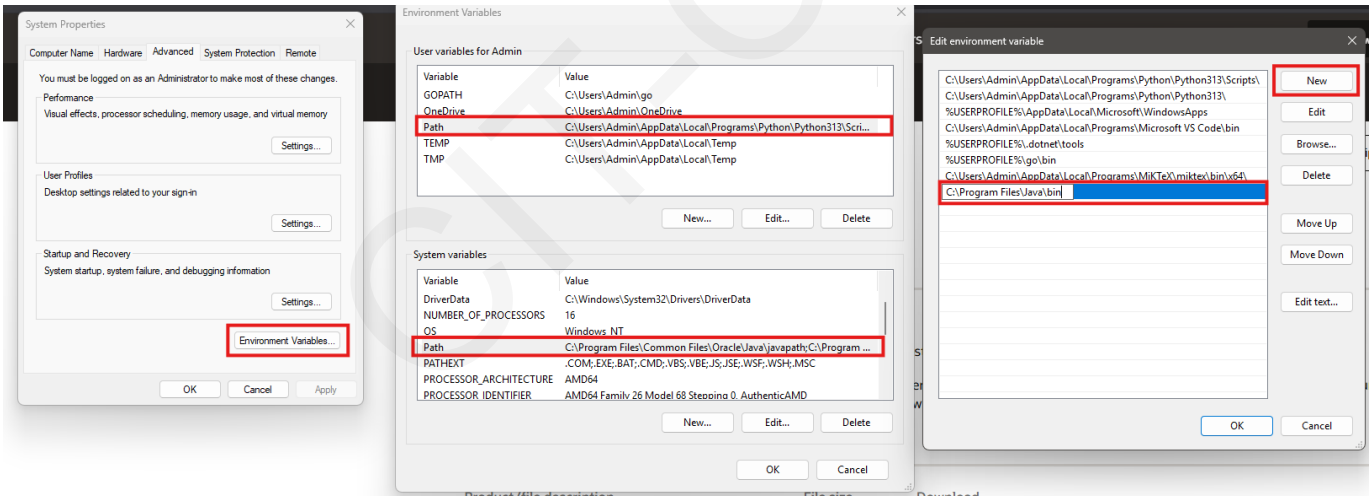
Java SE Development Kit 21.0.7 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the Java SE OTN License (OTN) and production use beyond the limited free grants of the OTN license will require a fee.

Linux | macOS | Windows

Product/file description	File size	Download
x64 Compressed Archive	185.97 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer	164.35 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	163.09 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)



STEP 5: Installing Apache Maven
(b) Download Maven:

Go to: https://maven.apache.org/download.cgi
Download the binary zip file.

(c) Unzip Maven into " C:\Program Files "

(d) Configure Environment Variable: (same as of JAVA)

Add bin directory path of Maven (e.g., C:\Program Files\apache-maven-3.9.9\bin) to your System PATH.

(e) Verify Installation:

Open CMD and run:

mvn --version

You should see Maven's version info and Java environment confirmation.

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> mvn --version
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\apache-maven-3.9.9\apache-maven-3.9.9
Java version: 21.0.7, vendor: Oracle Corporation, runtime: C:\Program Files\Java
Default locale: en_US, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
PS C:\Users\Admin>
```

STEP 6: Installing Gradle

(f) Install Gradle on Windows:

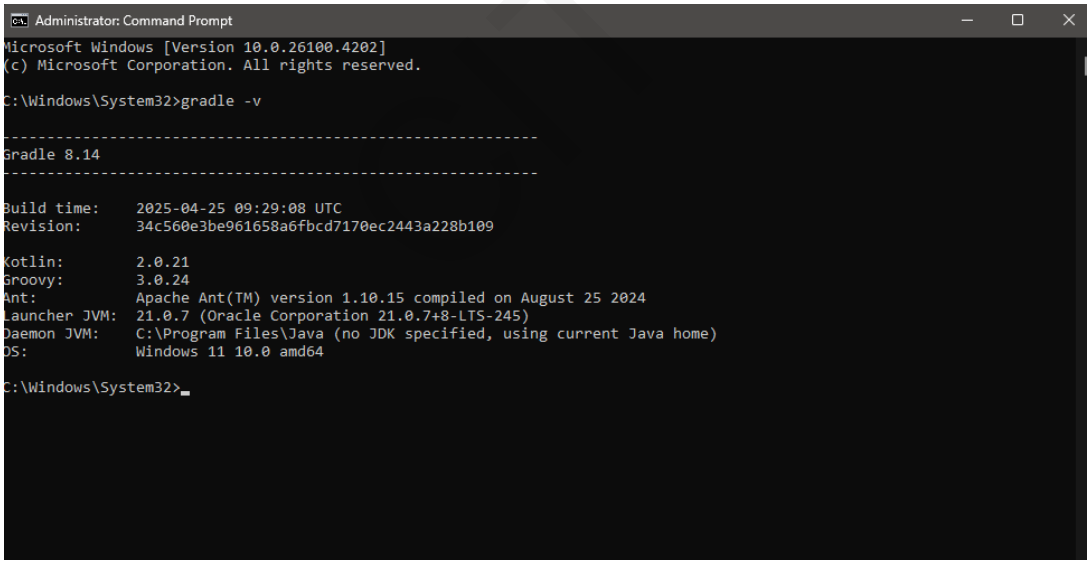
Create a directory C:\Gradle.

Extract the Gradle distribution (gradle-8.12.1) zip into C:\Gradle.

Add the bin path of Gradle (e.g., C:\Gradle\gradle-8.12.1\bin) to your System PATH.

Verify installation by running:

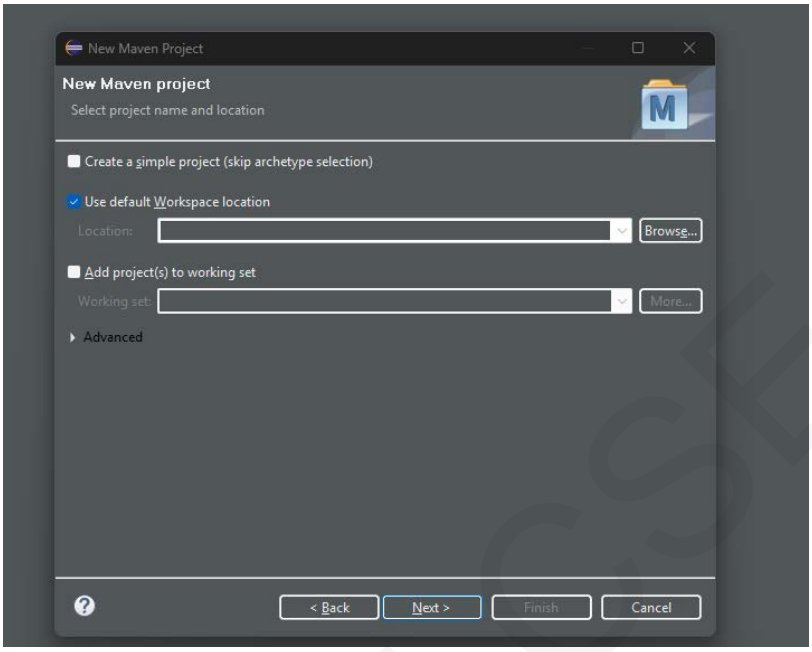
gradle -v



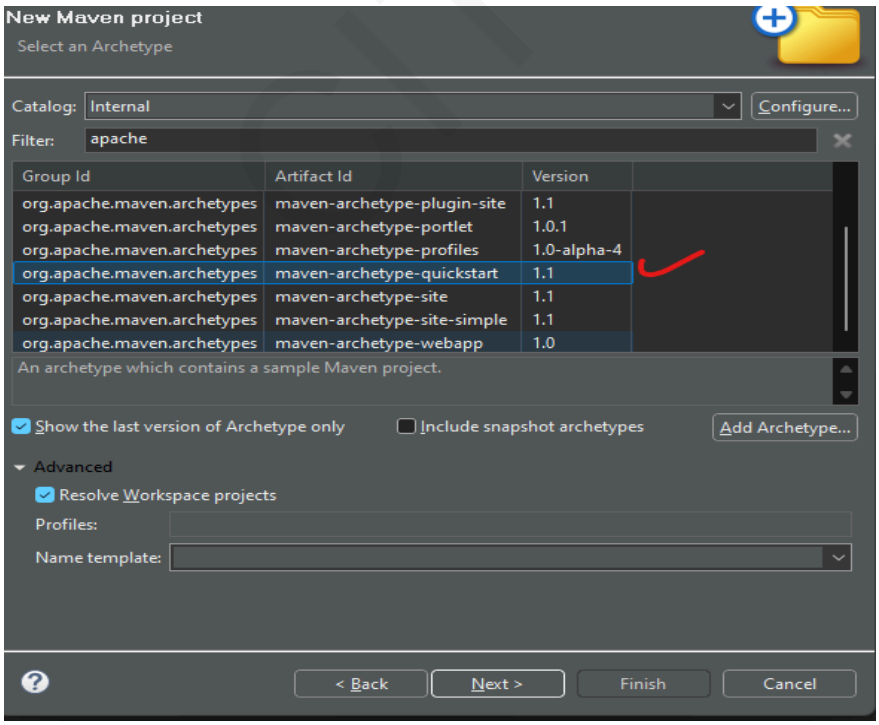
Program 2 : Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins

Creating a Maven Project in Eclipse Open Eclipse IDE.

Navigate to : File → New → Maven Project



Ensure the checkbox "Use default Workspace Location" is selected. Click Next.



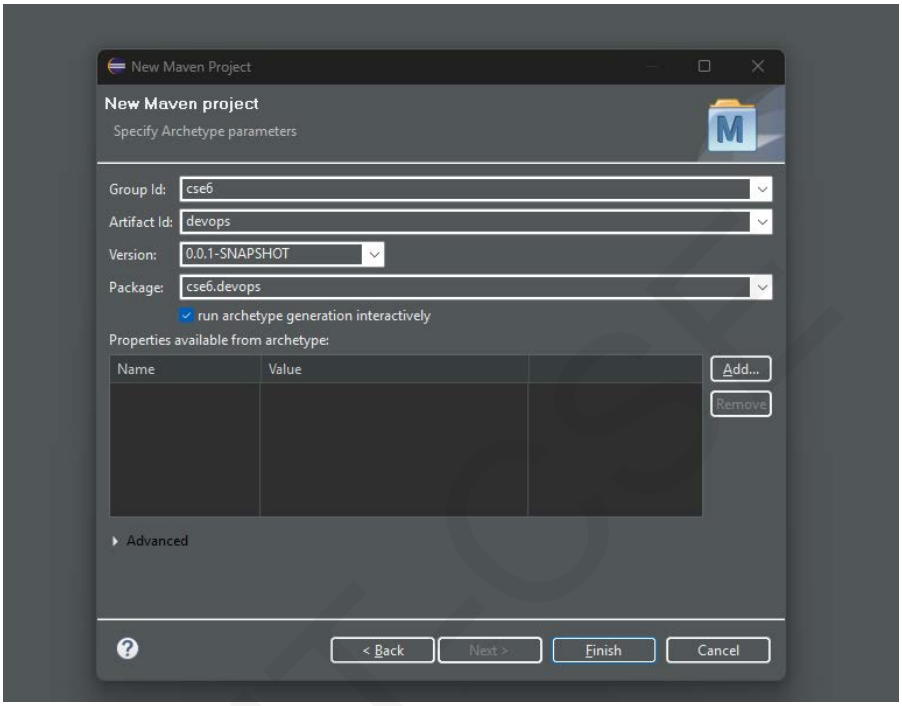
In the Filter box, type "apache" or select Catalog as "Internal".

Choose the archetype: maven-archetype-quickstart

Fill in the required details:
(Group ID and Artifact ID can be of your choice)
Group ID: cse6
Artifact ID: devops

Package: cse6.devops
Keep the default version 0.0.1-SNAPSHOT.

Click Finish.
Maven will automatically generate the project and its directory structure.



Maven will automatically generate the project and its directory structure.

When prompted for confirmation, press Y (Yes).

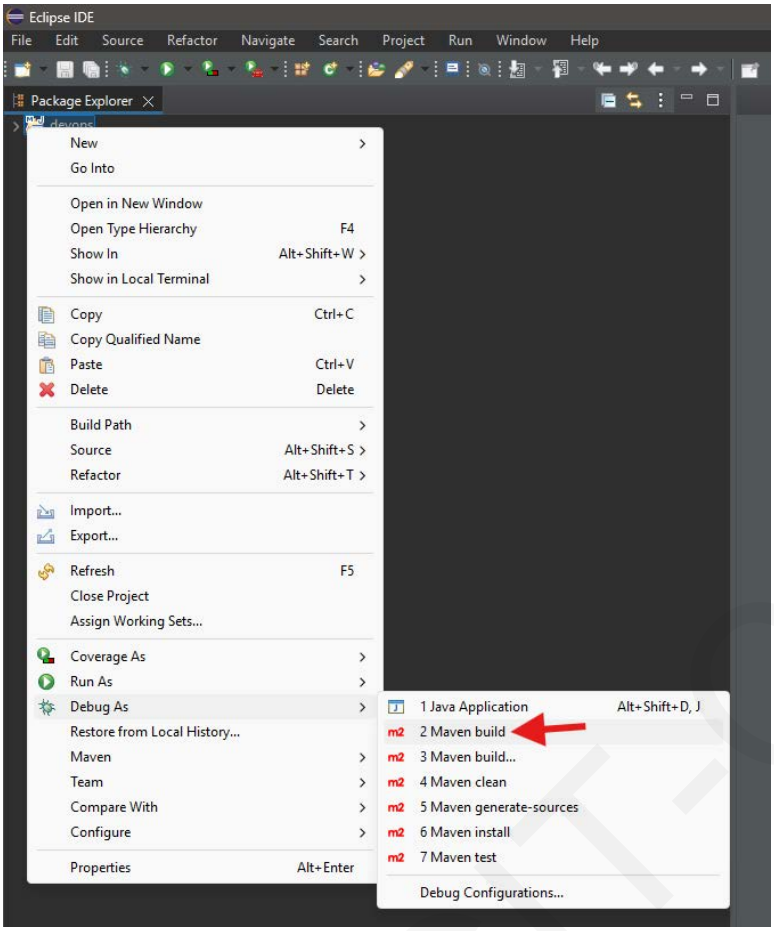
```
C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\javaw.exe (Jun 2, 2025, 5:59:51 PM) [pid: 31000]
Progress (1): 0/17 MB
Progress (1): 0/17 MB

Downloaded from : https://repo.maven.apache.org/maven2/archetype-catalog.xml (17 MB at 10 MB/s)
[INFO] Archetype repository not defined. Using the one from [org.apache.maven.archetypes:maven-archetype-plugin-site:1.5] found in catalog remote
[INFO] Using property: groupId = cse6
[INFO] Using property: artifactId = devops
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = cse6.devops
Confirm properties configuration:
groupId: cse6
artifactId: devops
version: 0.0.1-SNAPSHOT
package: cse6.devops
 Y: y
```

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.124 s
[INFO] Finished at: 2025-06-02T18:05:21+05:30
[INFO] ------------------------------------------------------------------------
```

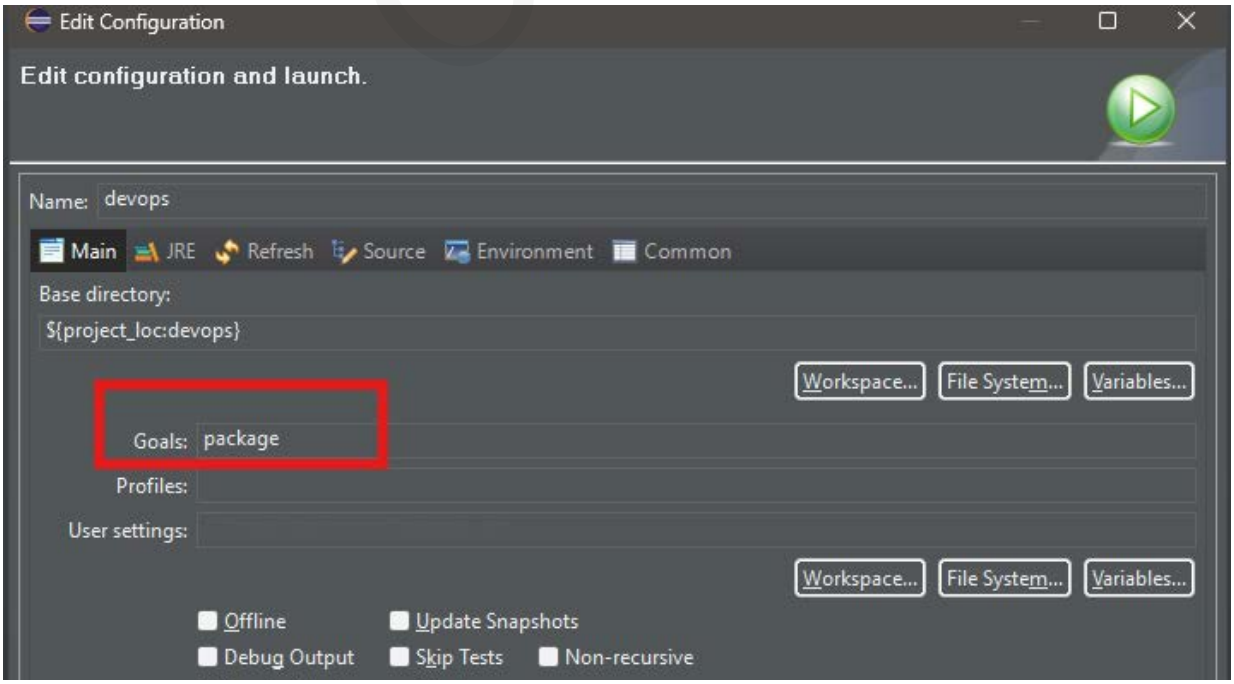
Building the Maven Project

In the Project Explorer, right-click the Maven project.

Select : Run As → Maven Build



In the Goals field, enter : package

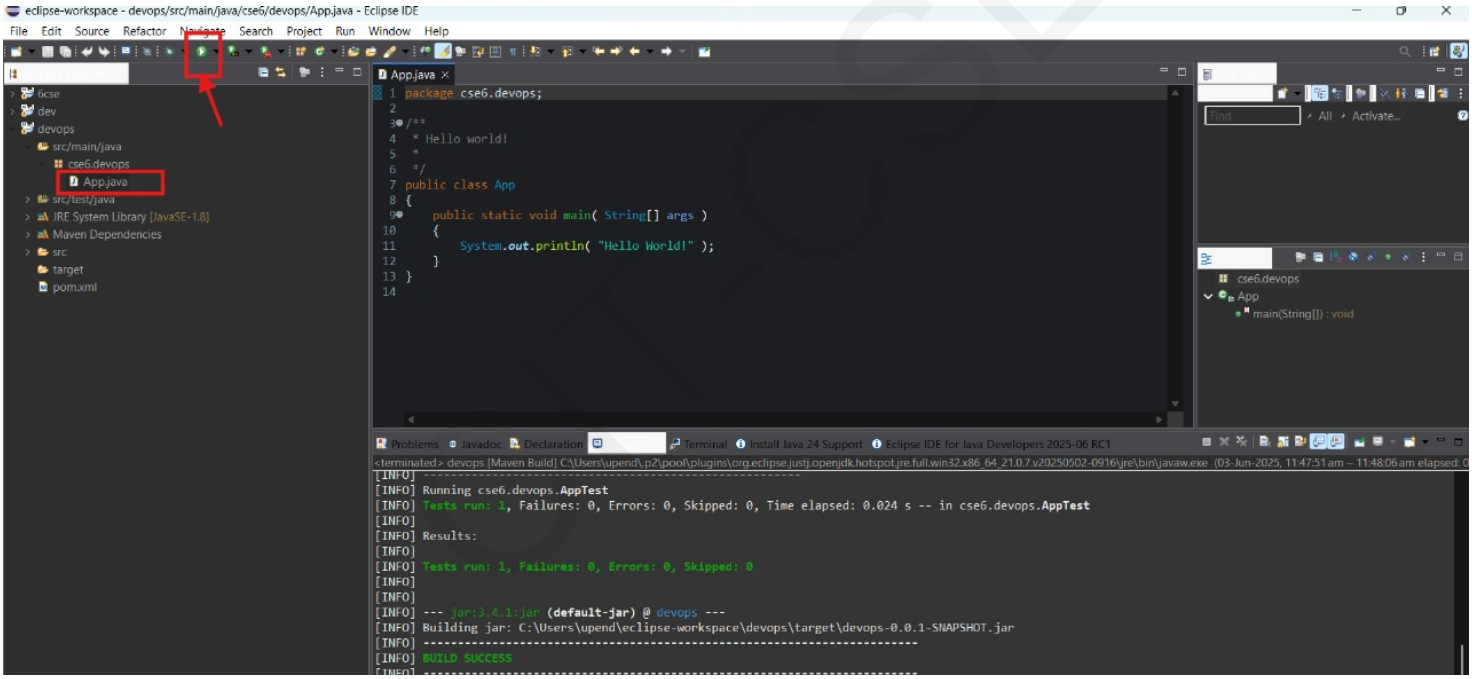


Click Run.

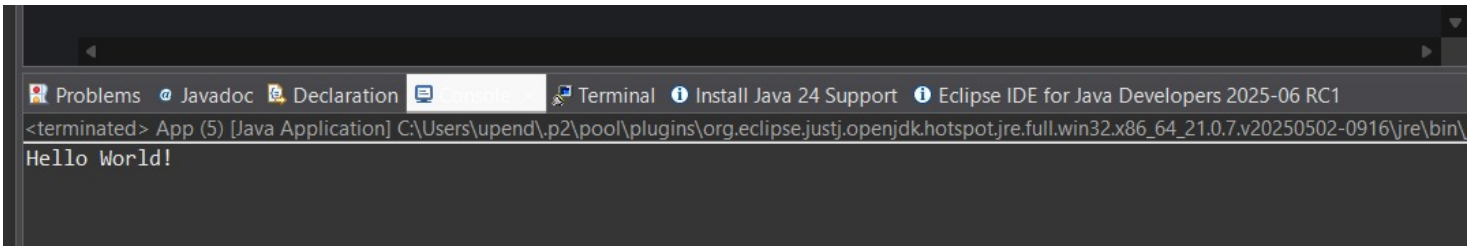
You should see Maven compile the source code, run tests, and package the app into a .jar file.

Running the Application

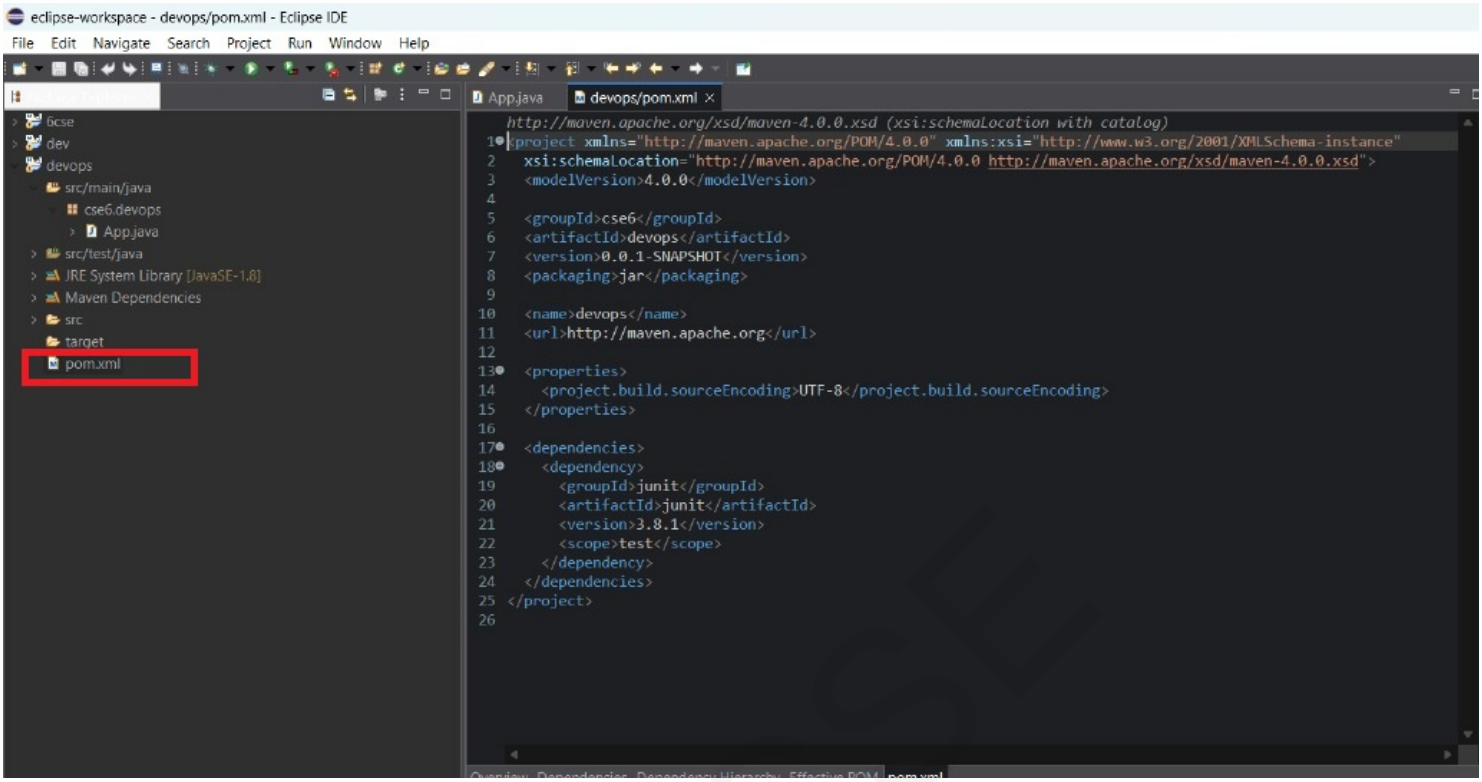
Go to the generated file App.java located in : devops/src/main/java/cse6.devops/



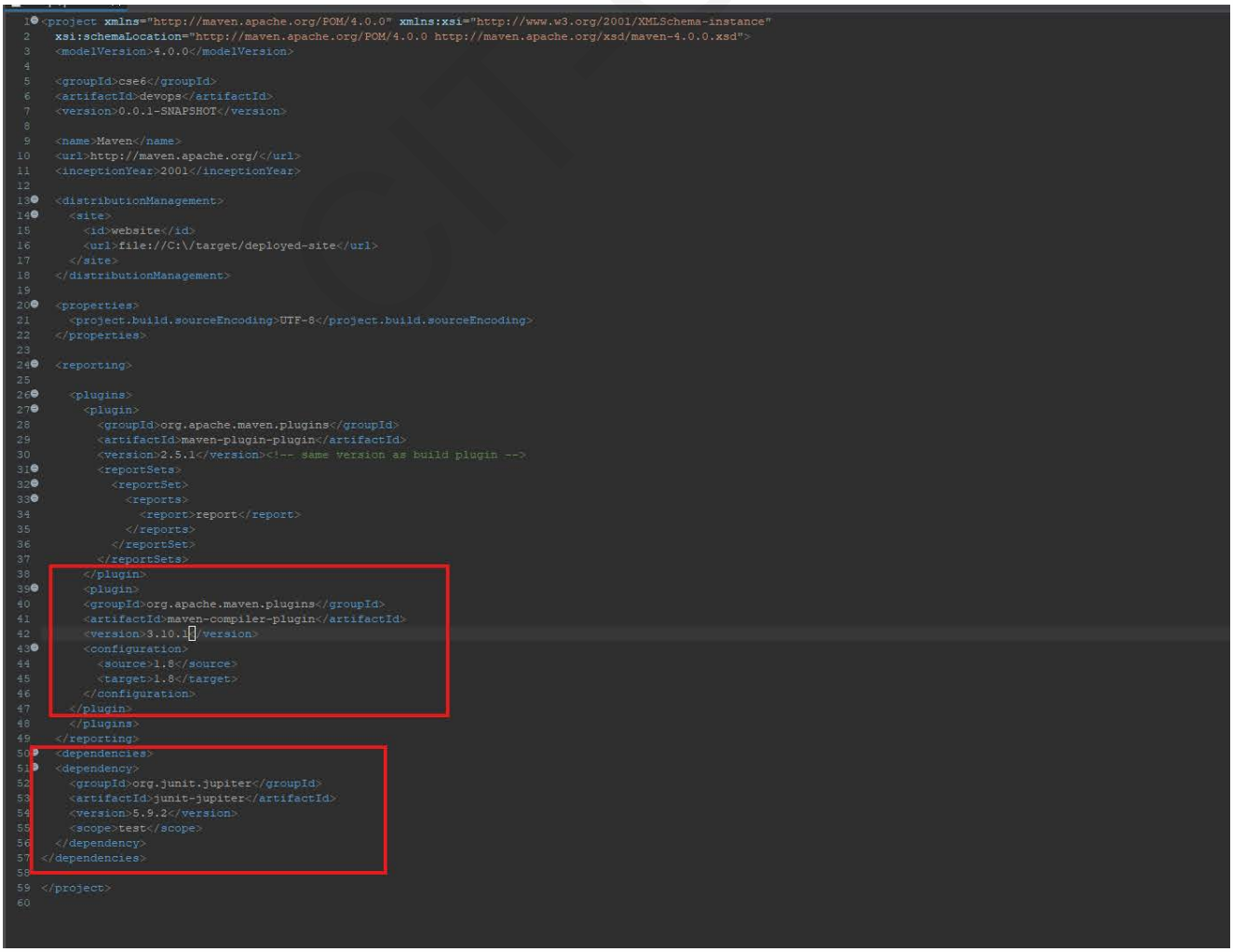
Below is the expected Output



This is the pom.xml file



You can add your desired dependencies and plugins, in the pom.xml file



PROGRAM3:Working with Gradle: Setting Up a Gradle Project, Understanding Build Scripts (Groovy and Kotlin DSL), Dependency Management and Task Automation

Step 1: Create Project Directory
Open Command Prompt and create a new folder:
mkdir cse-cit
cd cse-cit

Step 2: Initialize Gradle Project
Run the command:
gradle init

Choose:
1. Type: Application
2. Language: Groovy (option 3)
3. DSL: Groovy (option 2)
4. Structure: Single application
5. Java Version: 21
6. Project Name: cse6

```
Microsoft Windows [Version 10.0.26100.4202]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>mkdir cse-cit

C:\Users\Admin>cd cse-cit

C:\Users\Admin\cse-cit>gradle init
Starting a Gradle Daemon (subsequent builds will be faster)
Calculating task graph as no cached configuration is available for tasks: init

Select type of build to generate:
  1: Application
  2: Library
  3: Gradle plugin
  4: Basic (build structure only)
Enter selection (default: Application) [1..4] 1
```

```
Select implementation language:
  1: Java
  2: Kotlin
  3: Groovy
  4: Scala
  5: C++
  6: Swift
Enter selection (default: Java) [1..6] 3

Enter target Java version (min: 7, default: 21): 21

Project name (default: cse-cit): cse6
```

```
Select application structure:
  1: Single application project
  2: Application and library project
Enter selection (default: Single application project) [1..2] 1
```

Now its time to Build the script. Jusrt type the command as
gradle run

```
Select build script DSL:
  1: Kotlin
  2: Groovy
Enter selection (default: Groovy) [1..2] 1

Generate build using new APIs and behavior (some features may change in the next minor release)? (default: no) [yes, no] yes


> Task :init
Learn more about Gradle by exploring our Samples at https://docs.gradle.org/8.14/samples/sample_building_groovy_applications.html

BUILD SUCCESSFUL in 1m 7s
1 actionable task: 1 executed
Configuration cache entry stored.
C:\Users\Admin\cse-cit>gradle run
Calculating task graph as no cached configuration is available for tasks: run

> Task :app:run
Hello World!

BUILD SUCCESSFUL in 7s
2 actionable tasks: 2 executed
Configuration cache entry stored.
```

Run the command as **tree**

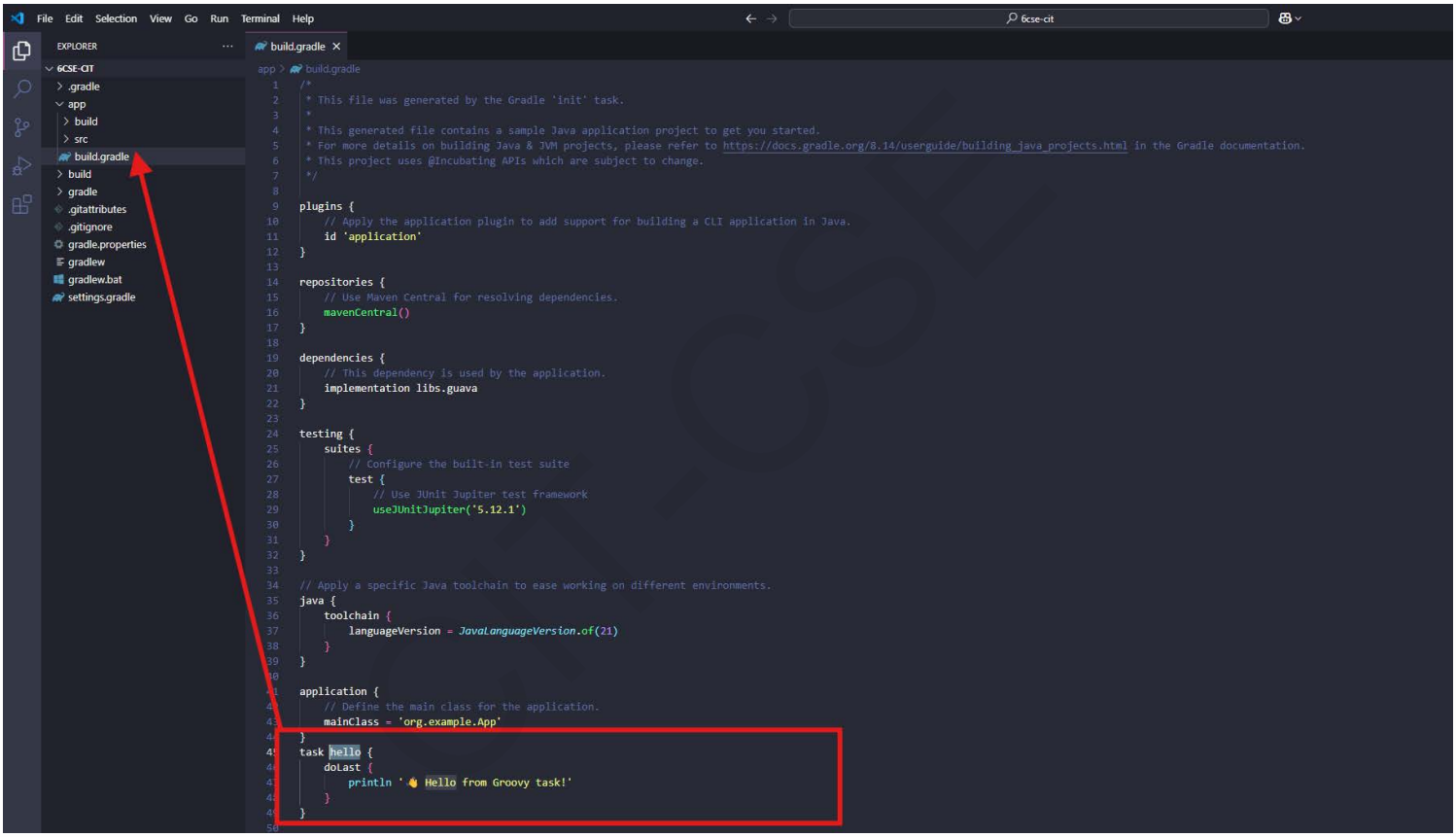
```
C:\Users\Admin\cse-cit>tree
Folder PATH listing
Volume serial number is 8CBD-7197
C:.
├───.gradle
│   ├───8.14
│   │   ├───checksums
│   │   ├───executionHistory
│   │   ├───expanded
│   │   ├───fileChanges
│   │   ├───fileHashes
│   │   └───vcsMetadata
│   ├───buildOutputCleanup
│   ├───configuration-cache
│   │   ├───3cdc4a8a-1ec7-413e-a473-51cda1b701d4
│   │   └───7pnpx4w8dwwmt72kdkd5e9myb
│   └───vcs-1
├───app
│   ├───build
│   │   ├───classes
│   │   │   └───groovy
│   │   │       └───main
│   │   │           └───org
│   │   │               └───example
│   │   ├───generated
│   │   │   └───sources
│   │   │       └───annotationProcessor
│   │   │           └───groovy
│   │   │               └───main
│   │   └───tmp
│   │       └───compileGroovy
│   │           └───groovy-java-stubs
│   └───src
│       ├───main
│       │   ├───groovy
│       │   │   └───org
│       │   │       └───example
│       │   └───resources
│       └───test
│           ├───groovy
│           │   └───org
│           │       └───example
│           └───resources
├───build
│   └───reports
│       └───configuration-cache
│           └───7pnpx4w8dwwmt72kdkd5e9myb
│               └───afo3tlt9ur5zaad3twjjenxil
└───gradle
    └───wrapper

C:\Users\Admin\cse-cit>
```

Step 3: Add Task in build.gradle

NOTE:OPEN THE PROJECT FOLDER IN VISUAL STUDIO CODE EDITOR AS SHOWN BELOW

Open build.gradle and at the bottom add:

```
task hello {
        doLast {
                println 'Hello from Groovy task!'
        }
}
```



Step 4: Run Custom Task

Execute the task using:

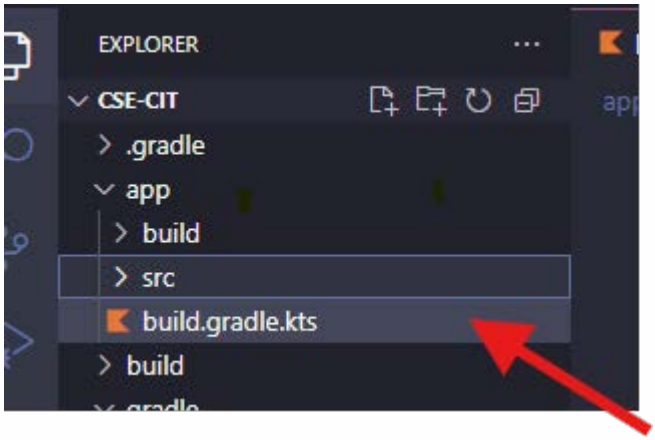
gradle hello

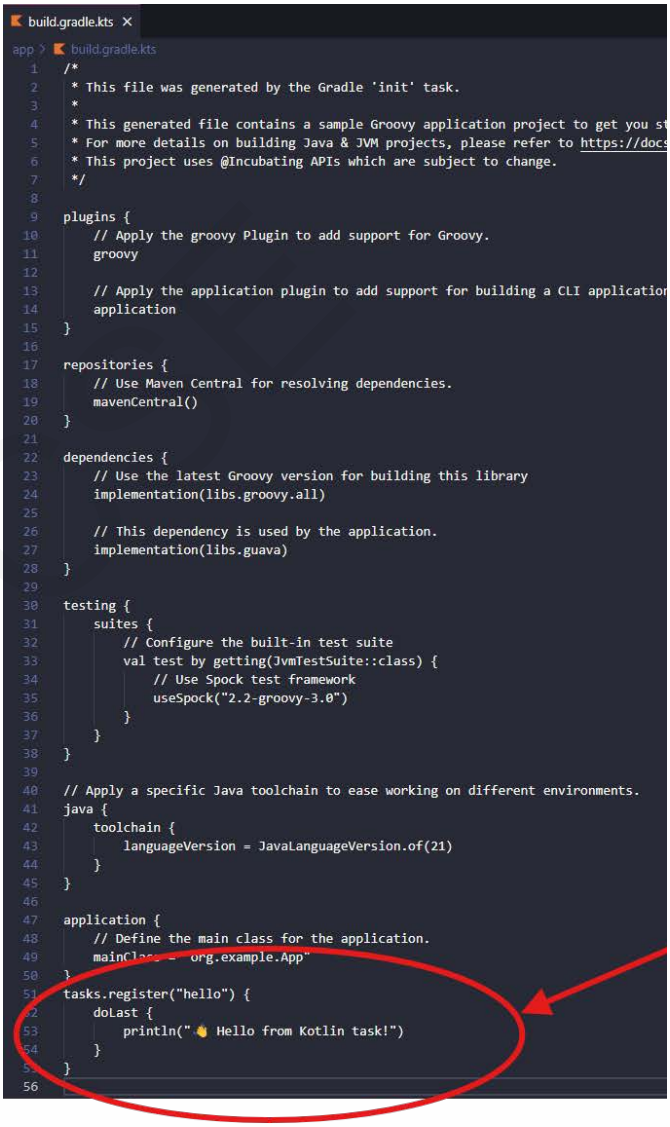
Expected Output:

> Task :hello
Hello from Groovy task!

Step 5: Kotlin DSL Alternative (build.gradle.kts)
If using Kotlin DSL, write the task as:

```
tasks.register("hello") {
doLast {
        println("Hello from Kotlin task!")
    }
}
```





Run Custom Task

Execute the task using:

gradle hello

Expected Output:

> Task :hello
Hello from Kotlin task!

To run the project:

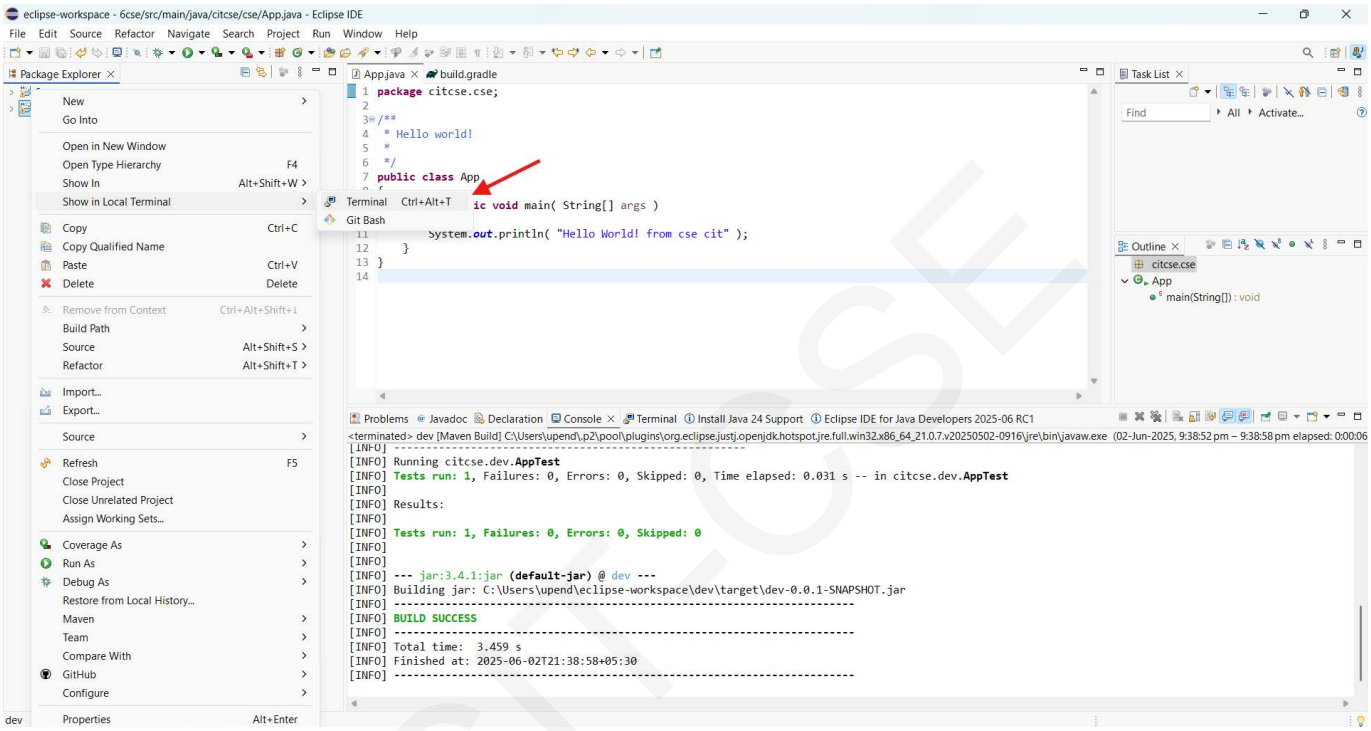
gradle run

Expected Output:
Hello World!

***PROGRAM 4:* Practical Exercise: Build and Run a Java Application with Maven, Migrate the Same Application to Gradle**

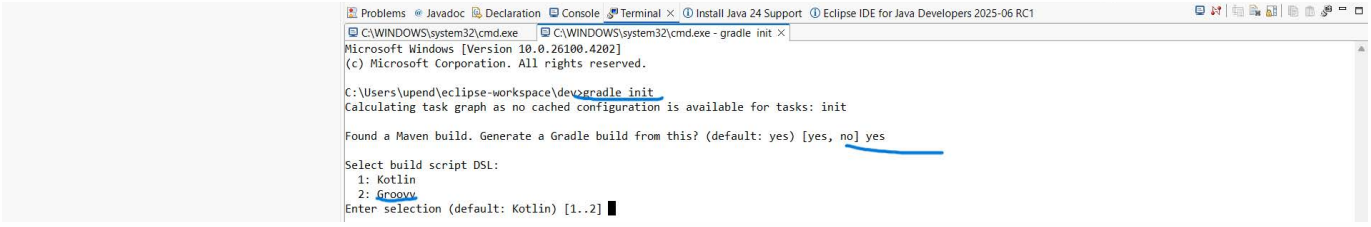
STEP1: First create a Maven Project as in PROGRAM2 then build the project and run java application you will get Hello World Message

STEP 2: Open Terminal in Eclipse
i.Right-click the project folder.
ii. Select: Show in Local Terminal → Terminal or use shortcut:
Ctrl + Alt + Shift + T



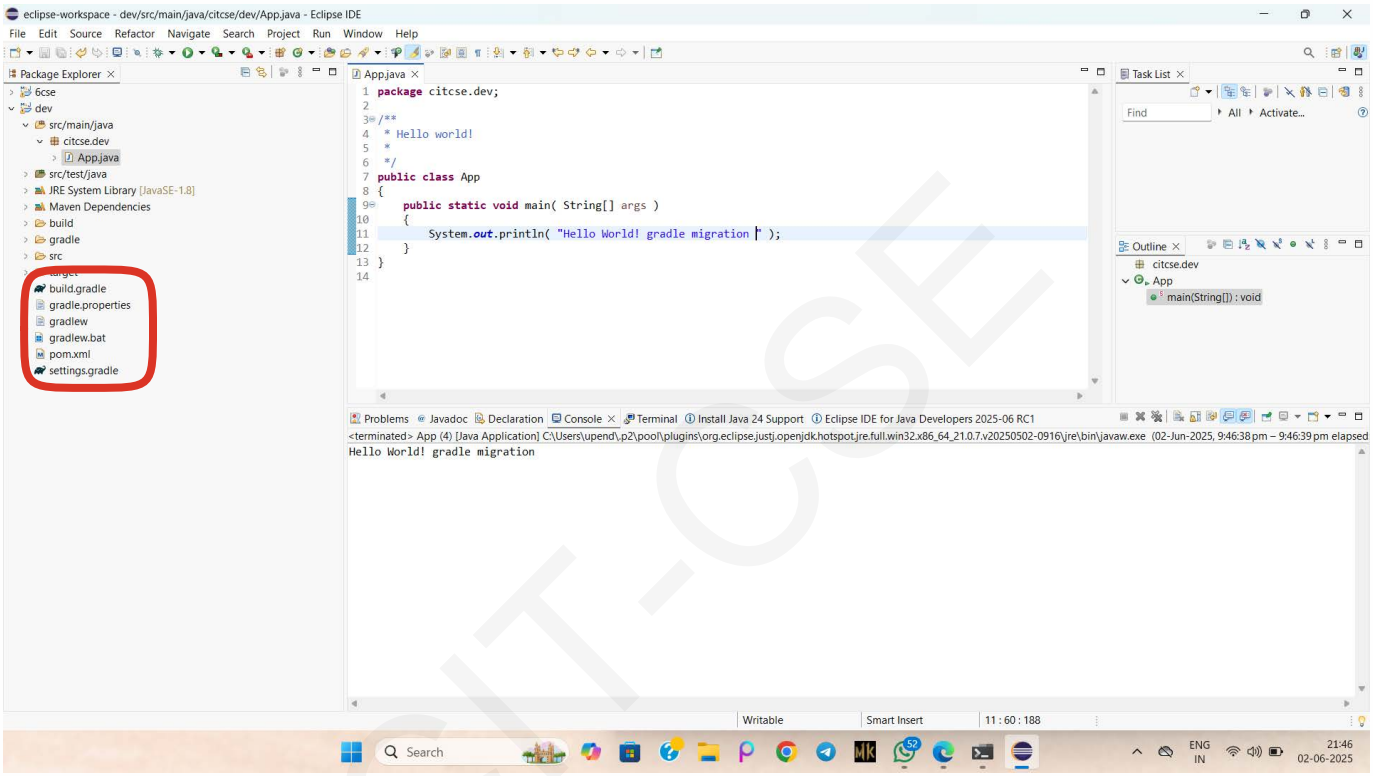
STEP 3: Start Gradle Migration
- In the terminal, type: **gradle init**
- When prompted to migrate from Maven to Gradle, type: yes
- When asked for DSL, choose: 2 (Groovy)

STEP 4: Validate Gradle Initialization
- Confirm the API generator prompt (select default).



STEP 5: Run Gradle Build
- Run the command: **gradle build**
- Confirm 'BUILD SUCCESSFUL' in the console.

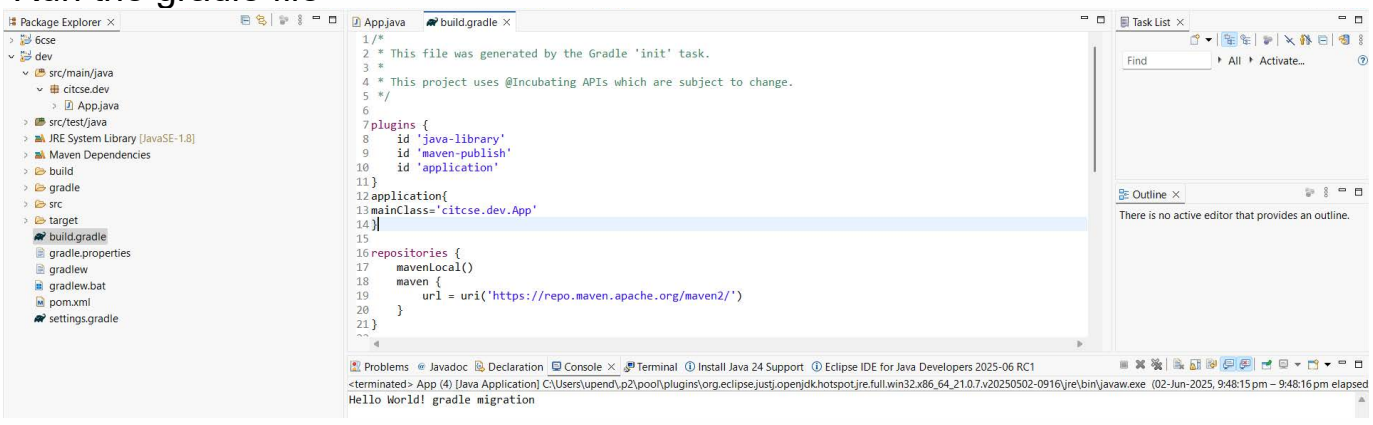
STEP 6: Modify App.java
- Change the print statement in App.java to: "Hello World! gradle migration"
- Run the java file
- Gradle files like build.gradle and settings.gradle will be created on the left pane.



STEP 7: Update build.gradle
- Add main class under application block

```
application {
mainClass = 'Groupid.Artifactid.App'
}
```

-Run the gradle file



AFTER DOING ALL CHANGES FINAL STEP

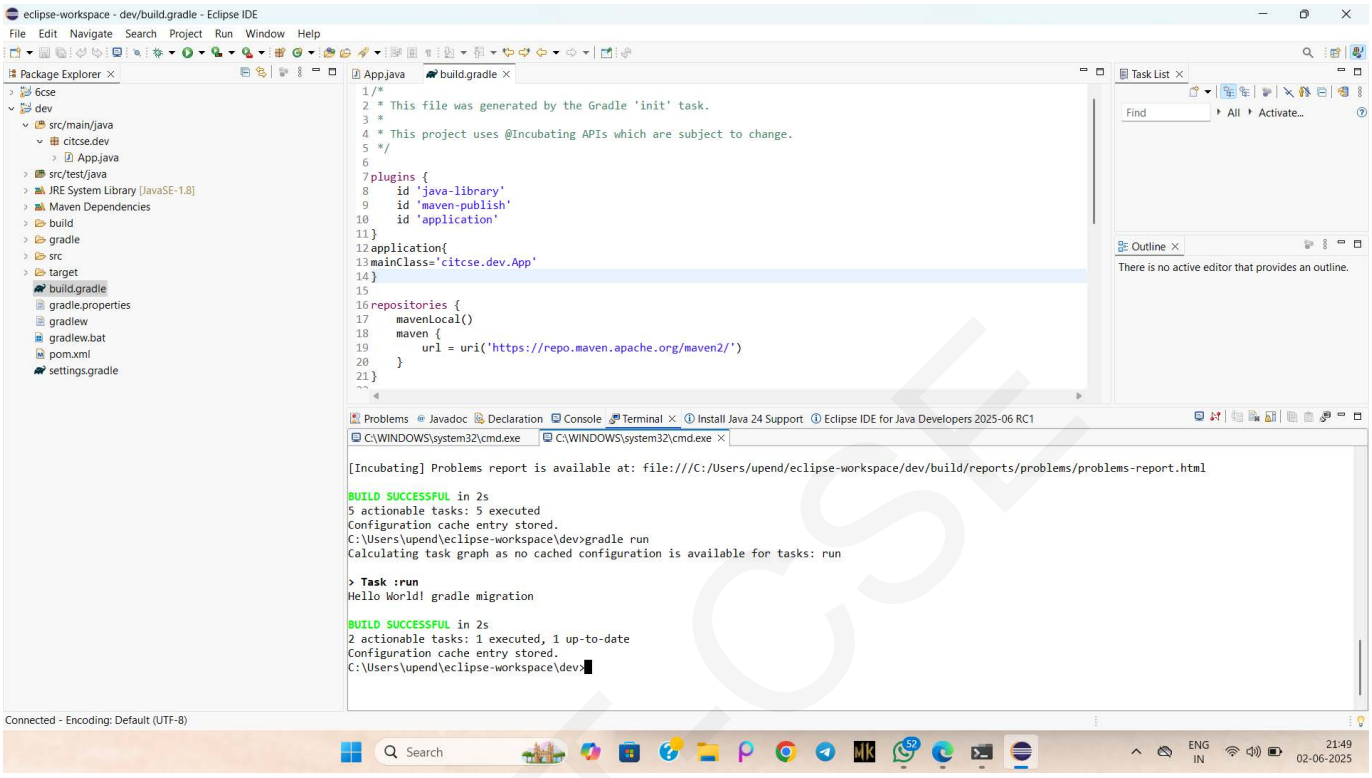
To run commands

gradle clean build

gradle run

You will get Output as

"Hello World! gradle migration"



You have successfully:

- **Created a Java Maven project**
- **Migrated it to Gradle using Groovy DSL**
- **Verified build and run using Gradle**

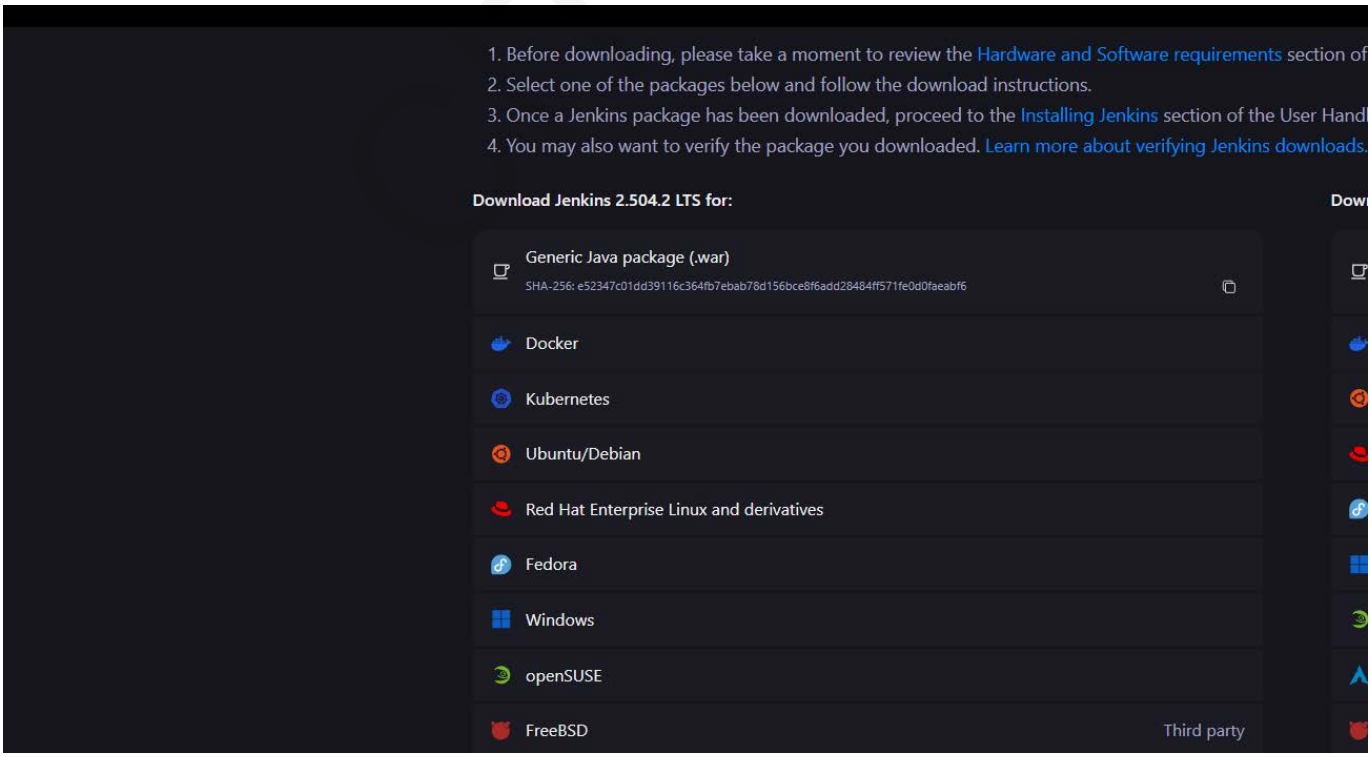
PROGRAM5 : Introduction to Jenkins: What is Jenkins?, Installing Jenkins on Local or Cloud Environment, Configuring Jenkins for First Use

Step 1: What is Jenkins?

Jenkins is an open-source automation server used to automate building, testing, and deploying software. It
supports continuous integration and delivery (CI/CD) through plugins and pipelines.

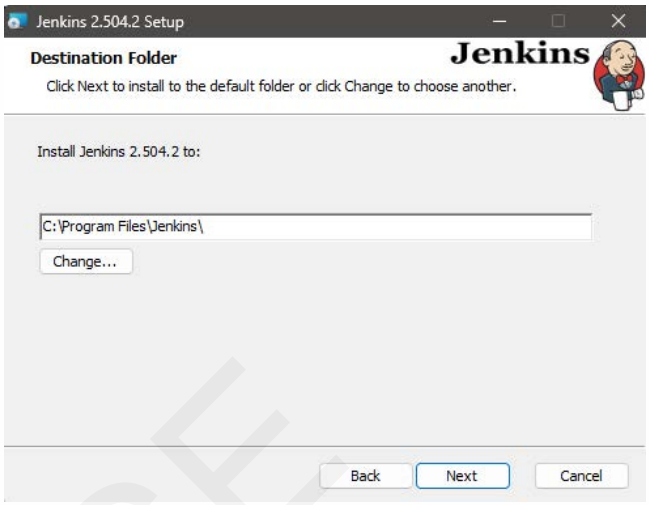
Step 2: Download Jenkins

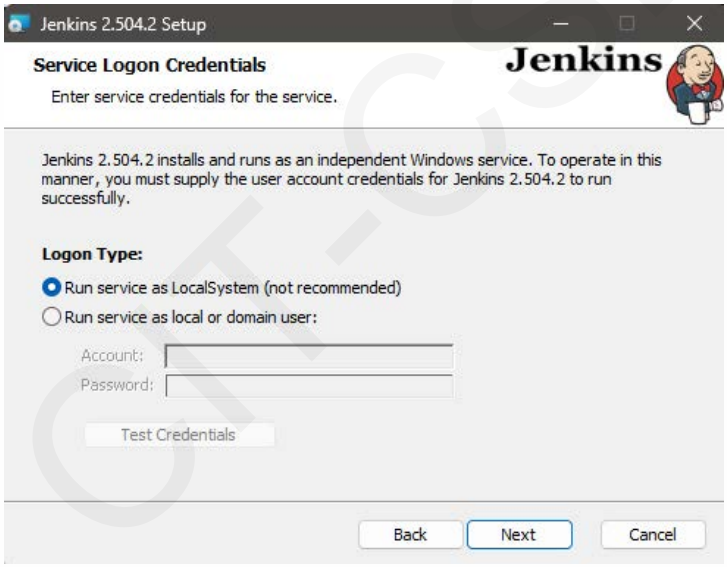
Visit https://www.jenkins.io/download/ and choose the appropriate platform.
For Windows, download the installer or WAR file.
You can also install it on Linux, Docker, or a cloud VM.



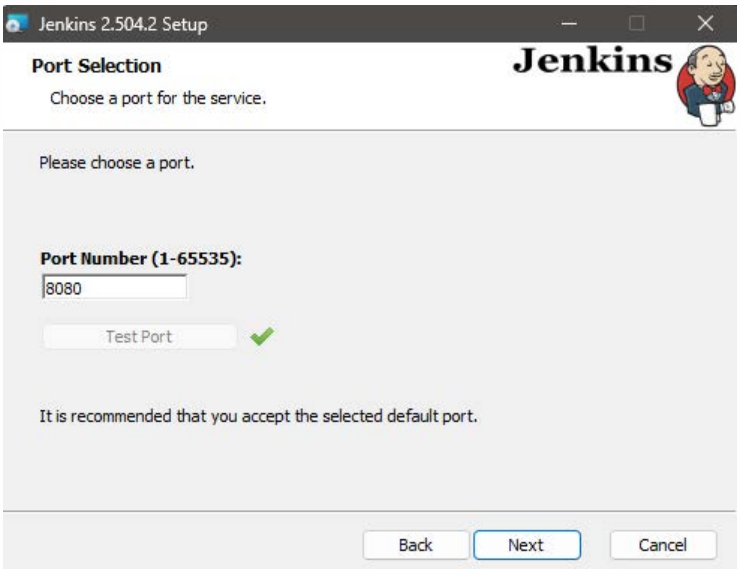
Step 3: Jenkins Installation on Windows (as shown in screenshots)

1. Launch Jenkins installer.

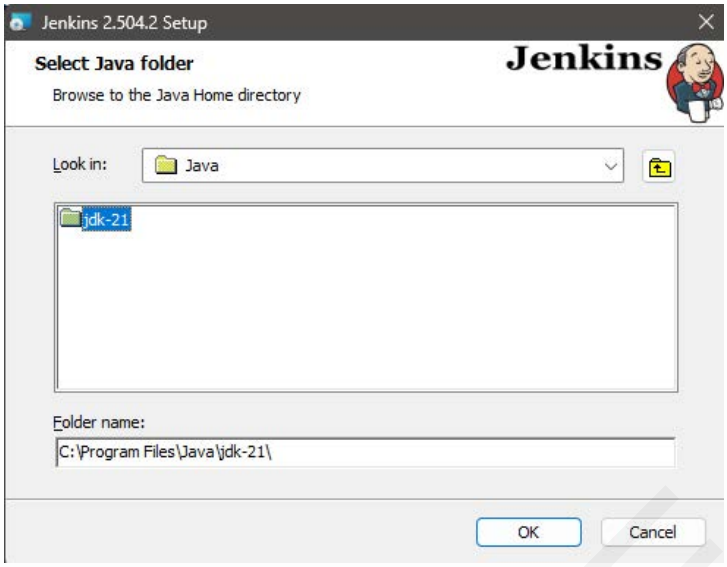




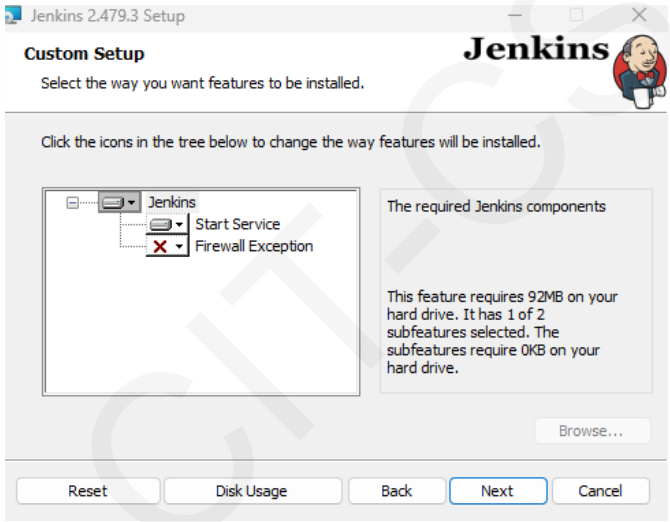
2. Accept default port (e.g., 8080).



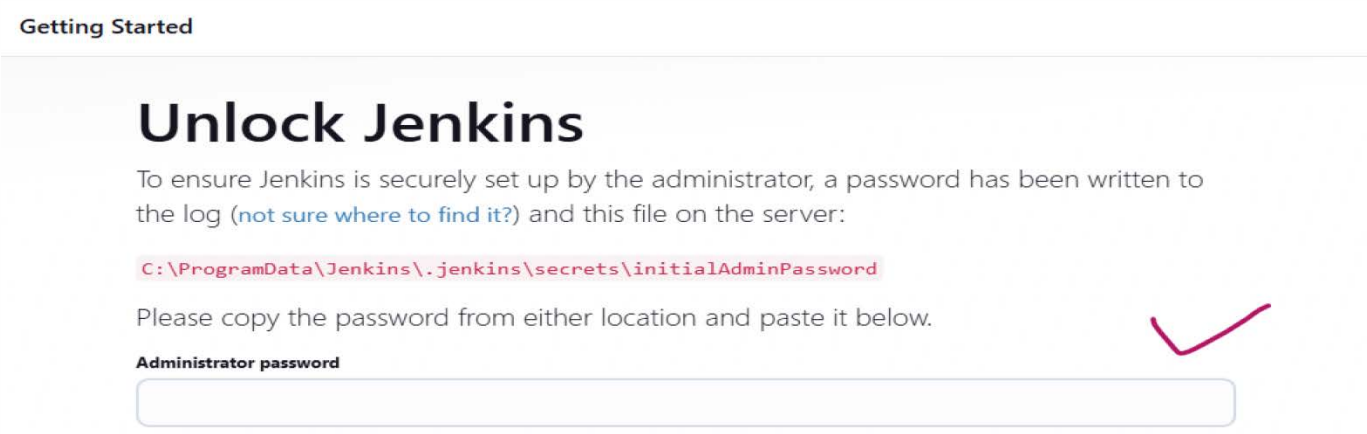
3. *Select the Java JDK directory (e.g., C:\Program Files\Java\jdk-21\).*



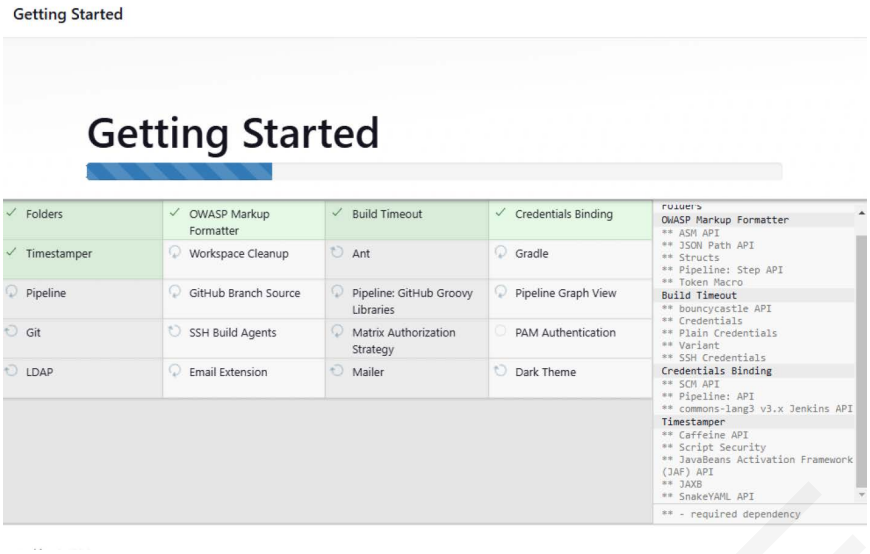
4. *Later Click on "Next" and proceed with finishing the installation*



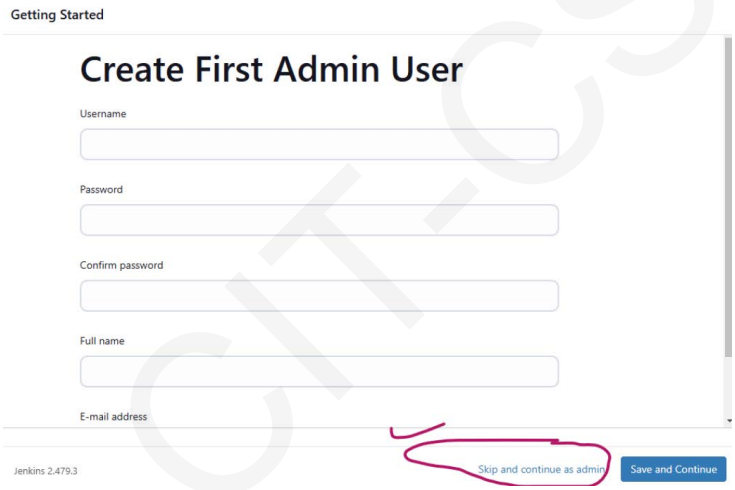
Final step very important Once It will ask for Administrator Password so u should locate as in directory mention copy paste as in mention in file location: ***C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword*** *and paste password as in*



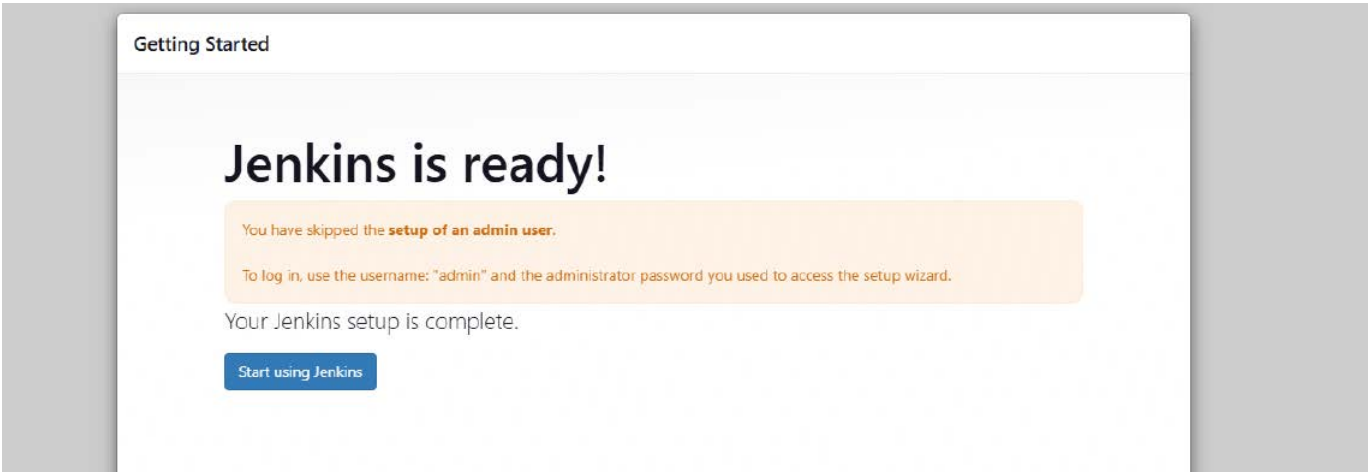
Let the installation proceed



Here just click on " Skip and continue as admin" , thus your default username and password will be **admin**



Jenkins is installed successfully



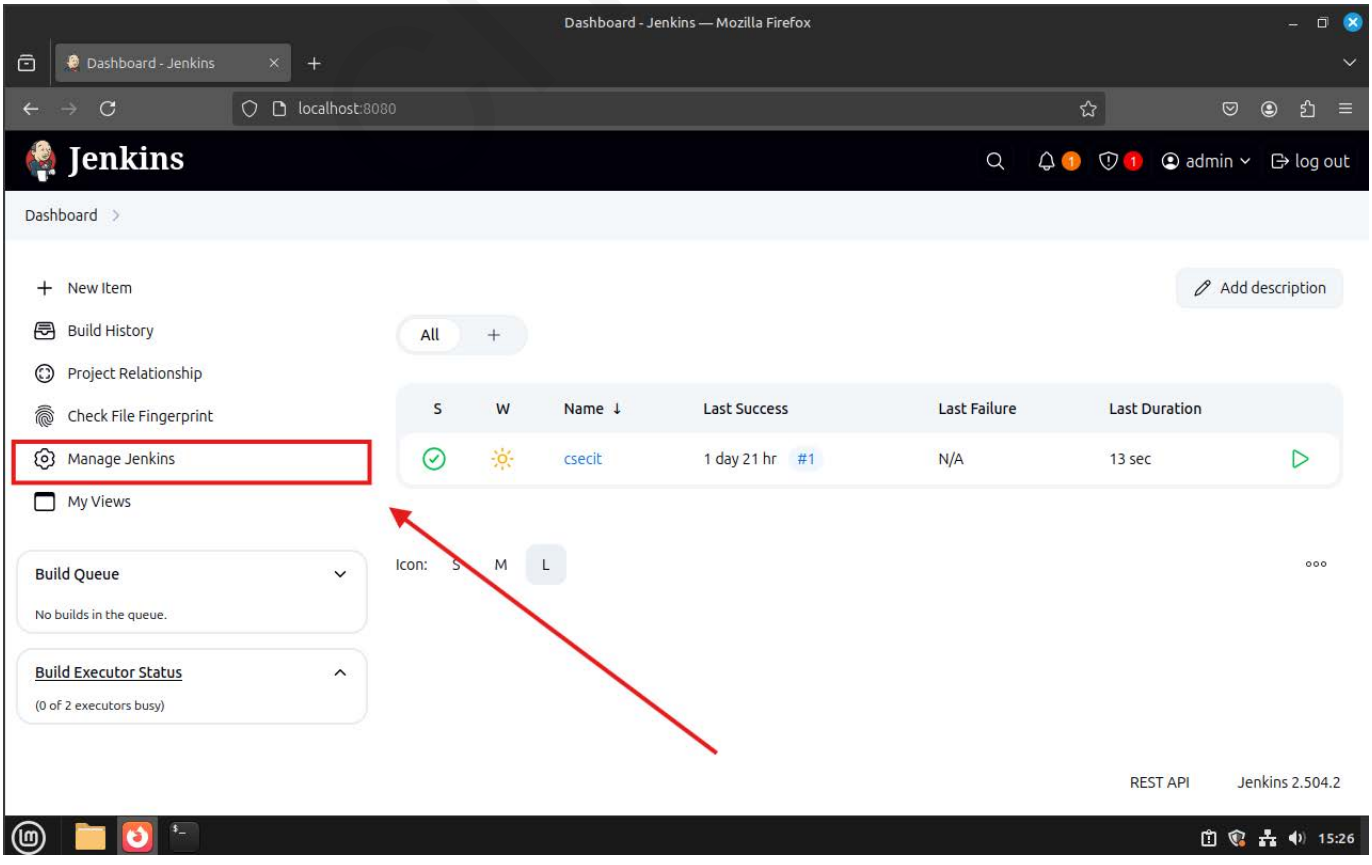
PROGRAM6 :Continuous Integration with Jenkins: Setting Up a CI Pipeline, Integrating Jenkins with Maven/Gradle, Running Automated Builds and Tests

Step 1: Login to Jenkins

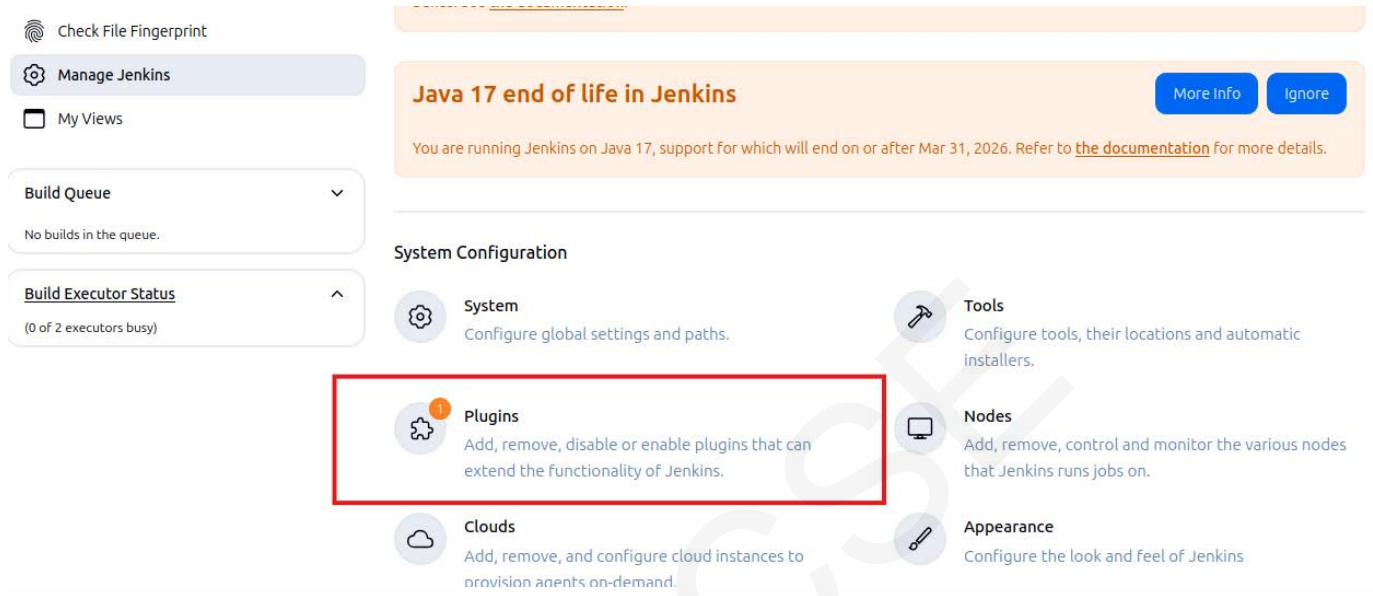
Open Jenkins at http://localhost:8080 and sign in using your admin credentials.

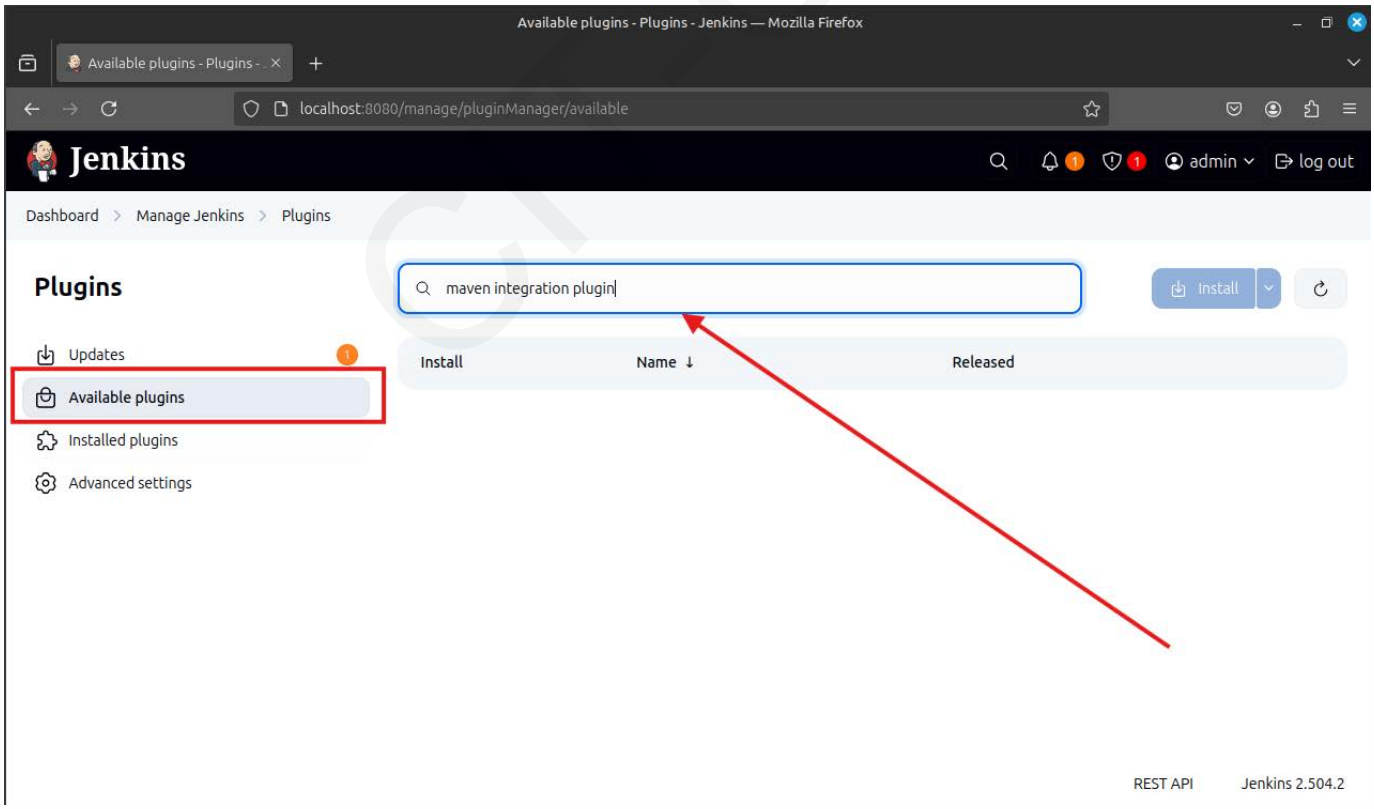
Step 2: Access Manage Jenkins

From the Jenkins dashboard, click on 'Manage Jenkins' to open global configuration settings.



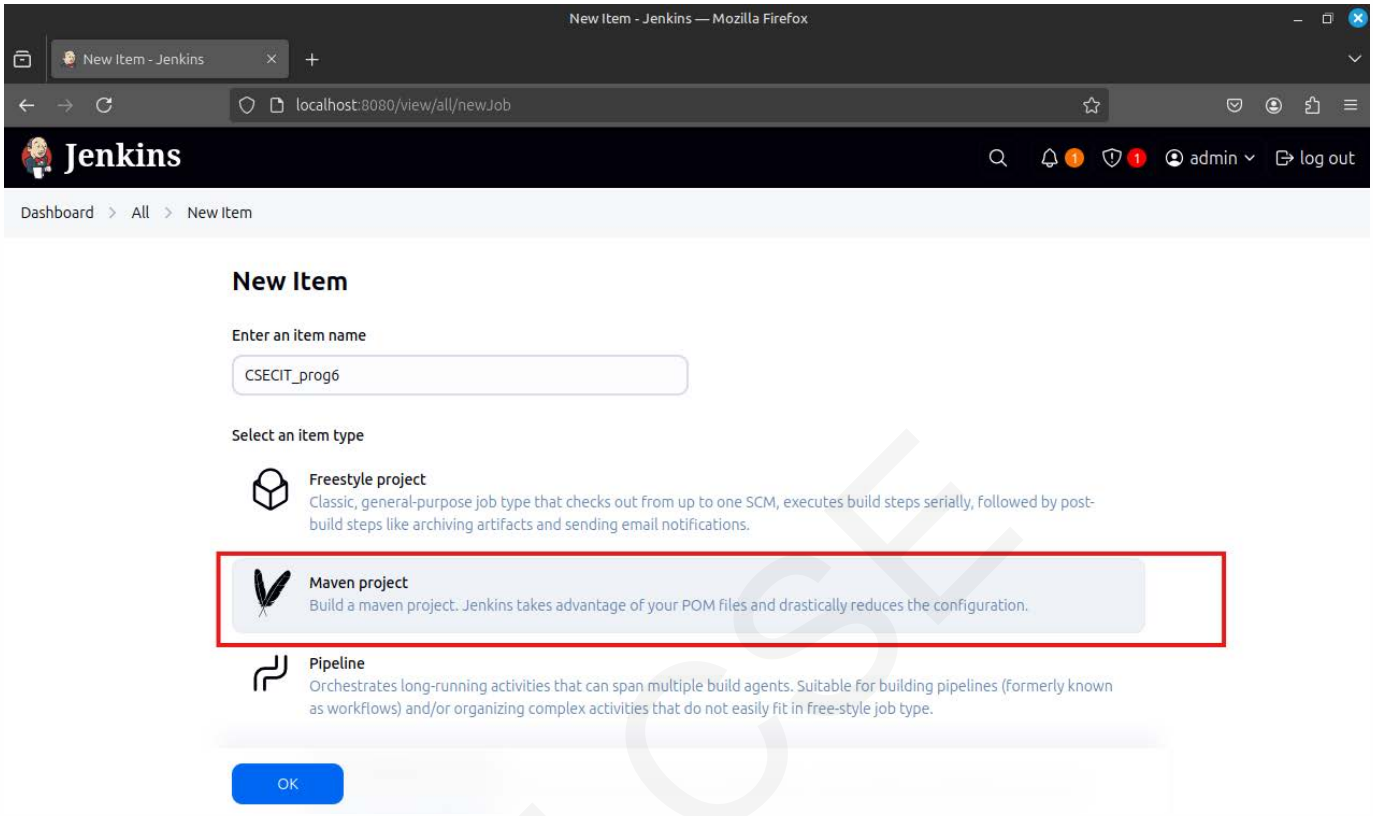
Step 3: Install Maven Integration Plugin
Navigate to:
Manage Jenkins > Plugins > Available Plugins
Search for 'Maven Integration Plugin', select it, and install it without restarting Jenkins.



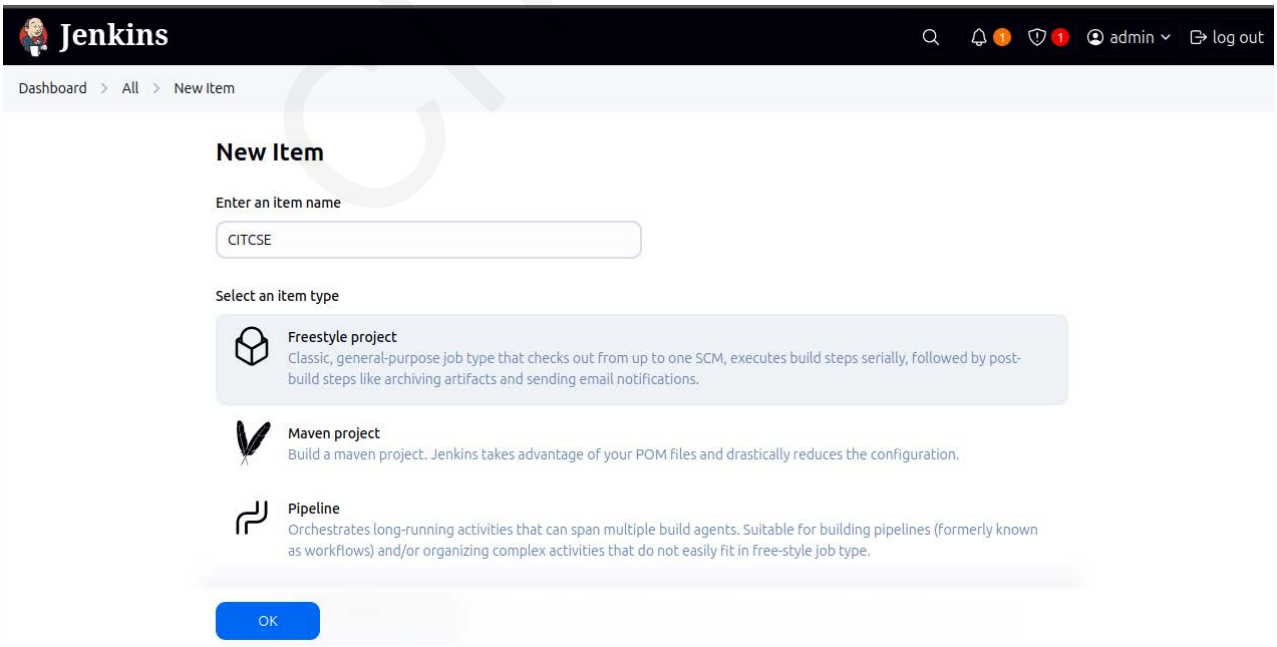


Tick on the Maven Integration Plugin and click on "Install" , and let the installation process continue.

Now you can see " Maven Project" option in the New Item Section

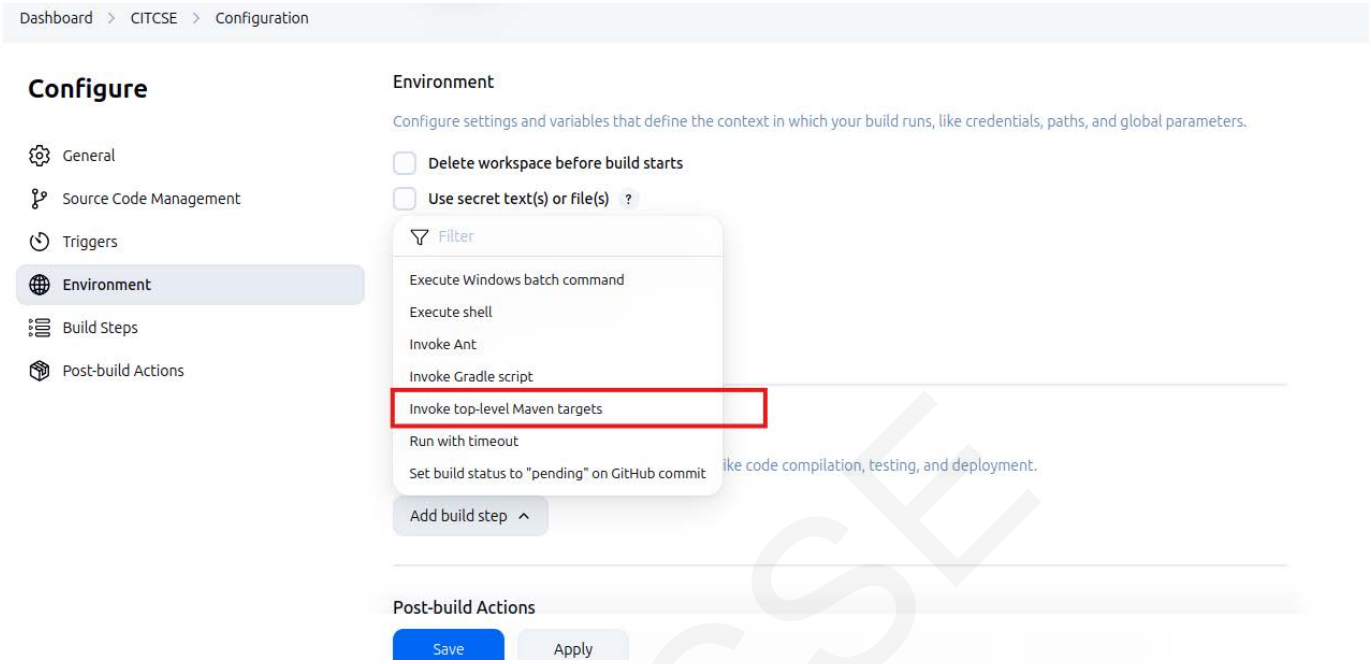


Now before this , Go to " New Item" and and choose Freestyle Project

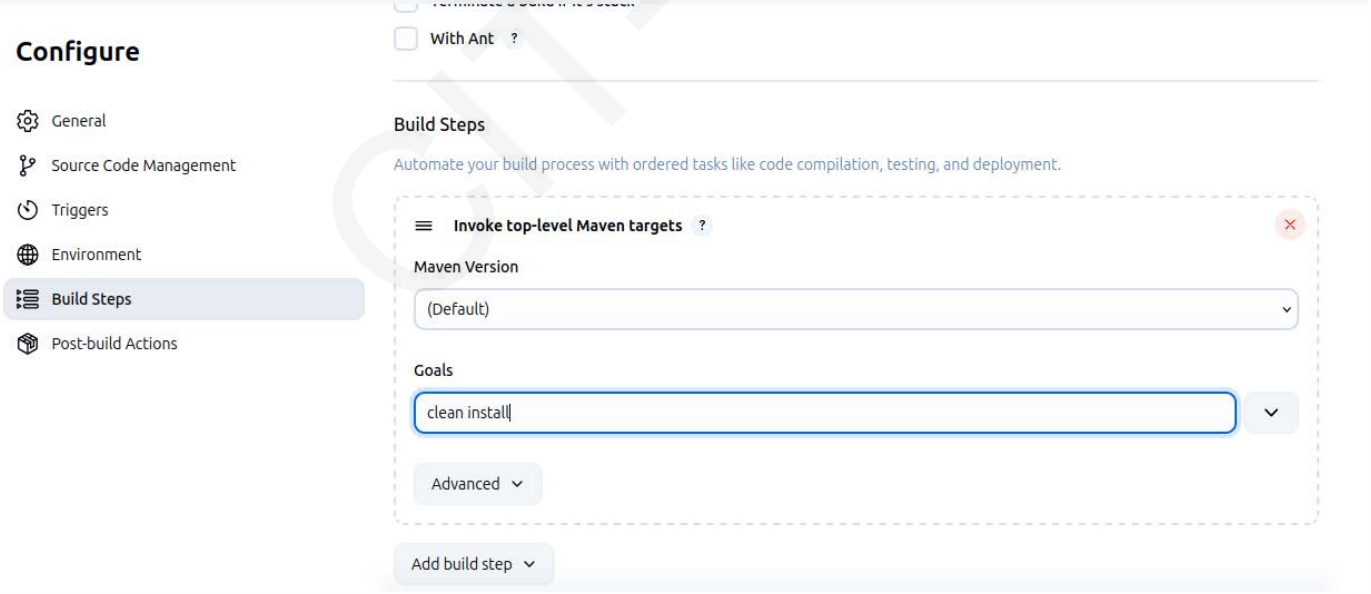


Click on " Ok "

Now to Configure , Go to Environement ---> Click on " Add build step " in Build Section , and here add the **Invoke top-level Maven targets**

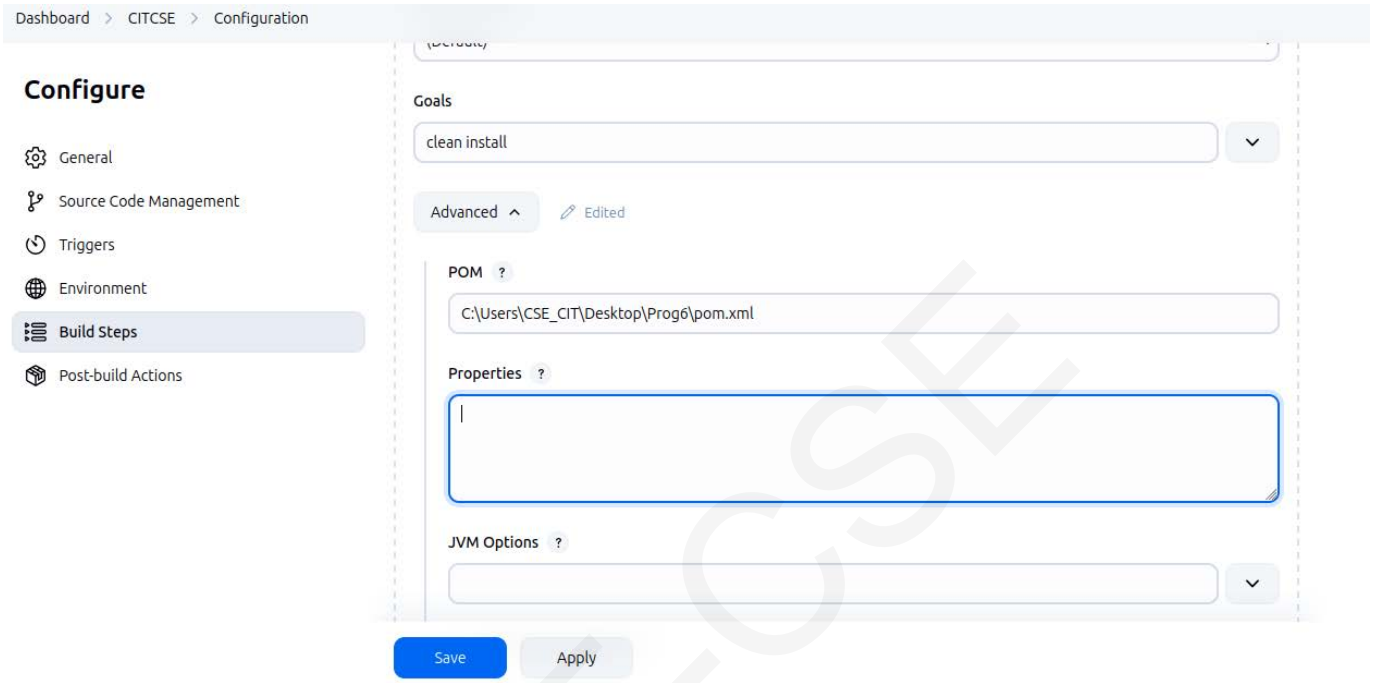


For the Maven_Version leave it as default , and in Goals section add - **clean install**

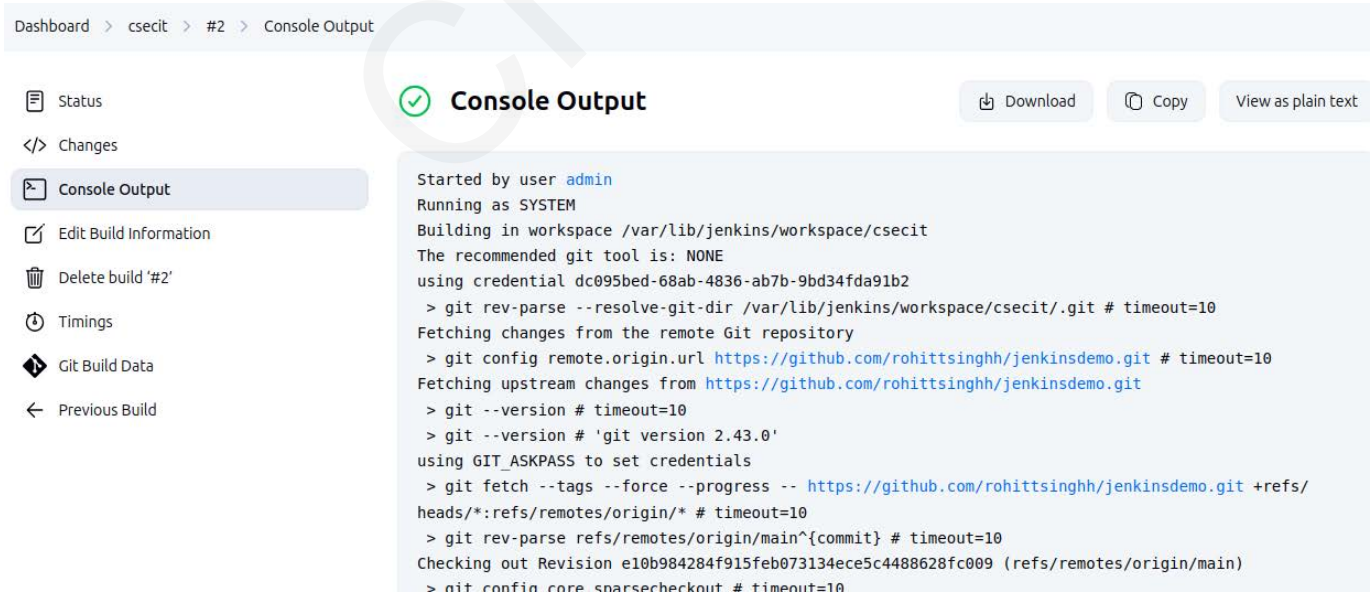


Now click on the drop down menu named **Advance ,** and here in POM section add your *pom.xml* file path

(**Note :** Your *pom.xml* file may be saved in different directory , copy the correct path and paste it here) and hit **Save and Apply**



Now here go back to the Item Dashboard and click on **Build Now**



PROGRAM7: Configuration Management with Ansible: Basics of Ansible:Inventory,Playbooks, and Modules, Automating Server Configurations withPlaybooks, Hands-On: Writing and Running a Basic Playbook.

Step 1 : On your VM Machine , in Ubuntu (or any Linux VM) open the terminal

Run the following command in terminal:

sudo apt install ansible -y

```
                                                    cse-cit@csecit-VirtualBox: ~
cse-cit@csecit-VirtualBox:~$ sudo apt install ansible -y
```

Step 2: Verify Ansible Installation

Check the installed version with:

ansible --version

```
cse-cit@csecit-VirtualBox:~$ ansible --version
ansible [core 2.16.3]
  config file = None
  configured module search path = ['/home/cse-cit/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/cse-cit/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Nov  6 2024, 18:32:19) [GCC 13.2.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
cse-cit@csecit-VirtualBox:~$
```

Step 3: Create Hosts Inventory File
Create the directory and open the hosts file:

sudo mkdir -p /etc/ansible
sudo nano /etc/ansible/hosts

Add the following content:
[local]
localhost ansible_connection=local

```
cse-cit@csecit-VirtualBox:~$ sudo mkdir -p /etc/ansible
[sudo] password for cse-cit:
cse-cit@csecit-VirtualBox:~$ sudo nano /etc/ansible/hosts
cse-cit@csecit-VirtualBox:~$
```

After typing these info Click **Ctrl + Q --------> Enter ---------> Ctrl + O**

Step 4: Test Connection Using Ping Module

Use the following command:

ansible all -m ping

Expected output:

SUCCESS with msg: pong

```
cse-cit@csecit-VirtualBox:~$ ansible all -m ping
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python
interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.16/reference_appendices/interpreter_discovery.html for more
information.
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.12"
    },
    "changed": false,
    "ping": "pong"
}
cse-cit@csecit-VirtualBox:~$
```

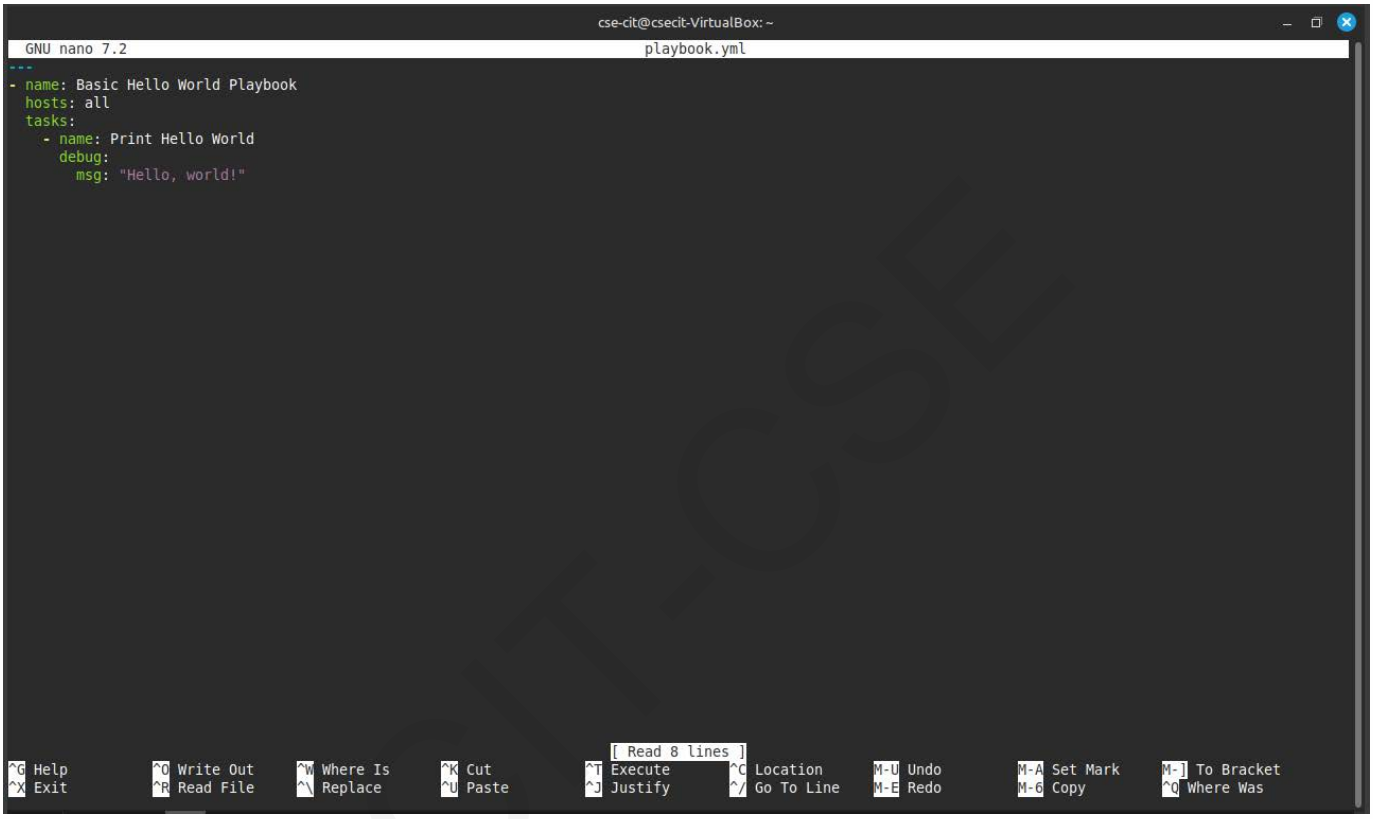
Create a file named playbook.yml using:

nano playbook.yml

```
cse-cit@csecit-VirtualBox:~$ nano playbook.yml
```

Add the following content:

```
- name: Basic Hello World Playbook
  hosts: all
  tasks:
    - name: Print Hello World
      debug:
        msg: "Hello, world!"
```



Step 6: Run the Playbook

Execute the playbook using:

```
ansible-playbook playbook.yml
```

```
cse-cit@csecit-VirtualBox:~$ ansible-playbook  playbook.yml

PLAY [Basic Hello World Playbook] ***********************************************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************************************************************
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python
interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.16/reference_appendices/interpreter_discovery.html for more
information.
ok: [localhost]

TASK [Print Hello World] *********************************************************************************************************************************************
ok: [localhost] => {
    "msg": "Hello, world!"
}

PLAY RECAP ***********************************************************************************************************************************************************
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

cse-cit@csecit-VirtualBox:~$
```

PROGRAM 8: Set Up a Jenkins CI Pipeline for a Maven Project, Use Ansible to Deploy Artifacts Generated by Jenkins

Step 1: Install Java
Ensure Java is installed and check using:

java --version

```
cse-cit@csecit-VirtualBox:~$ java --version
openjdk 17.0.15 2025-04-15
OpenJDK Runtime Environment (build 17.0.15+6-Ubuntu-0ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.15+6-Ubuntu-0ubuntu124.04, mixed mode, sharing)
```

Step 2: Install Jenkins
Update and install Jenkins:

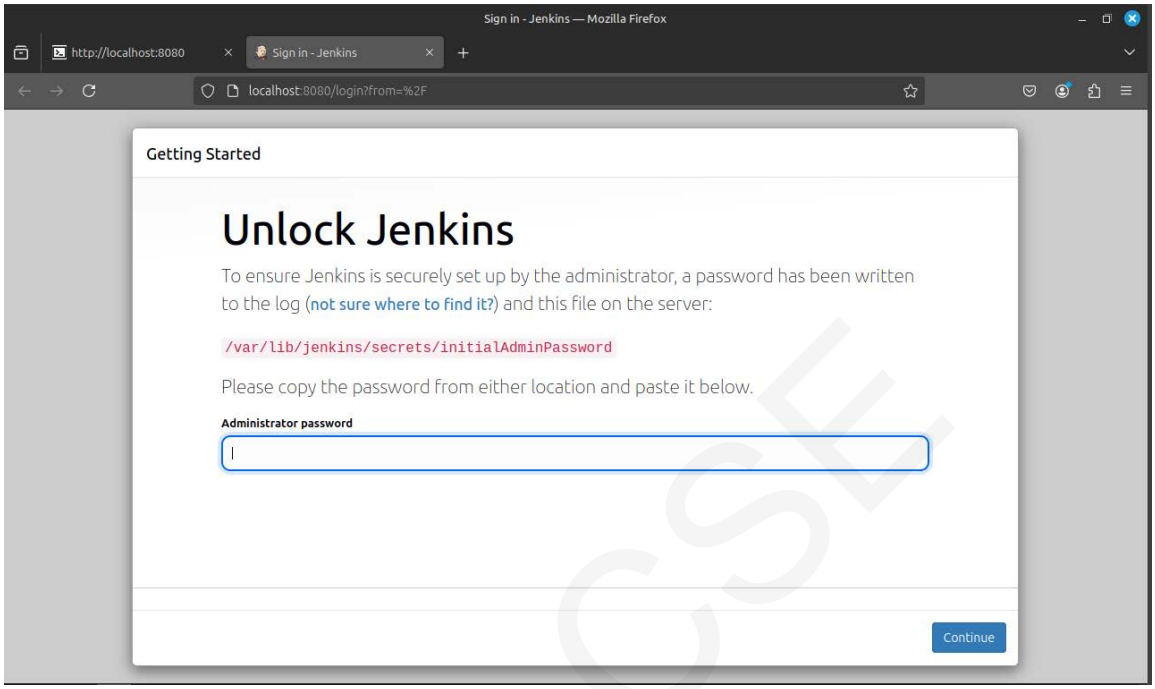
sudo apt update
sudo apt install jenkins -y
sudo systemctl start jenkins

```
cse-cit@csecit-VirtualBox:~$ sudo apt update
sudo apt install jenkins -y
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:2 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]
Get:3 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Get:4 https://pkg.jenkins.io/debian-stable binary/ Packages [29.1 kB]
Hit:5 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:7 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Ign:8 http://packages.linuxmint.com xia InRelease
Hit:9 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:10 http://packages.linuxmint.com xia Release
Fetched 32.0 kB in 1s (33.0 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
278 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  jenkins
0 upgraded, 1 newly installed, 0 to remove and 278 not upgraded.
Need to get 92.2 MB of archives.
After this operation, 94.4 MB of additional disk space will be used.
Get:1 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.504.2 [92.2 MB]
Fetched 92.2 MB in 8s (12.0 MB/s)
Selecting previously unselected package jenkins.
(Reading database ... 478026 files and directories currently installed.)
Preparing to unpack .../jenkins_2.504.2_all.deb ...
Unpacking jenkins (2.504.2) ...
Setting up jenkins (2.504.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
cse-cit@csecit-VirtualBox:~$ sudo systemctl start jenkins
sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
cse-cit@csecit-VirtualBox:~$
```

Step 3: Get Admin Password
Run the following to get Jenkins admin password:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Now here enter the Jenkins Password




```
cse-cit@csecit-VirtualBox:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Step 4: Install Maven

Install and verify Maven:

sudo apt install maven

mvn --version

```
cse-cit@csecit-VirtualBox:~$ mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 17.0.15, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-51-generic", arch: "amd64", family: "unix"
cse-cit@csecit-VirtualBox:~$
```

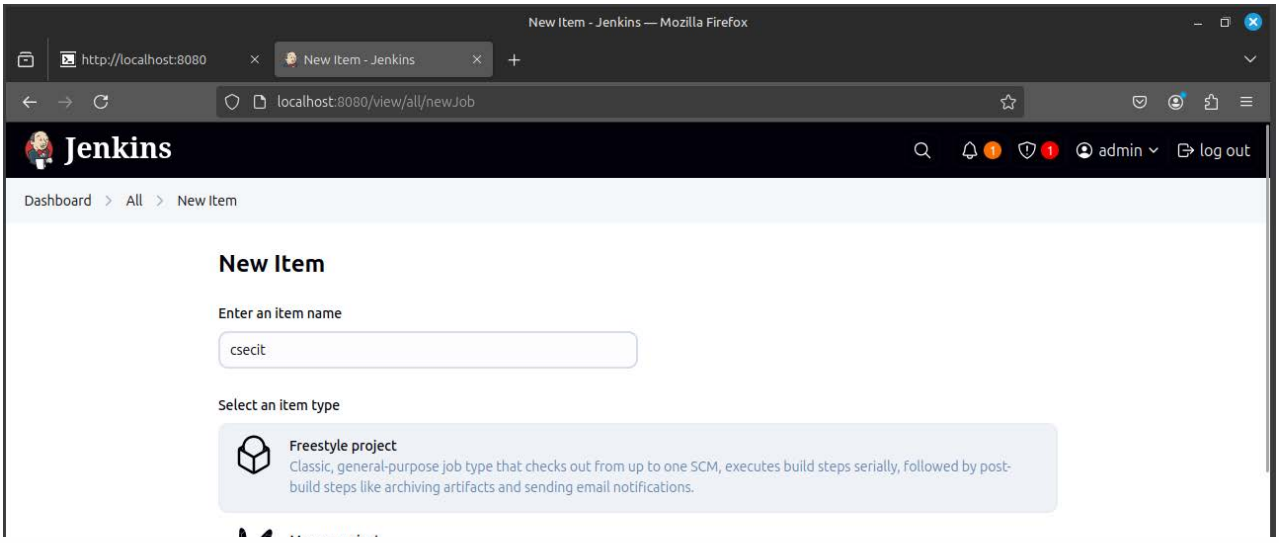
Step 5: Generate a Maven Project
Generate a sample Maven project:

mvn archetype:generate -DgroupId=com.example -DartifactId=demo \
-DarchetypeGroupId=org.apache.maven.archetypes \
-DarchetypeArtifactId=maven-archetype-quickstart \
-DarchetypeVersion=1.4 -DinteractiveMode=false

```
cse-cit@csecit-VirtualBox:~$ mvn archetype:generate -DgroupId=com.example -DartifactId=demo \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DarchetypeArtifactId=maven-archetype-quickstart \
  -DarchetypeVersion=1.4 \
  -DinteractiveMode=false
cd demo
```

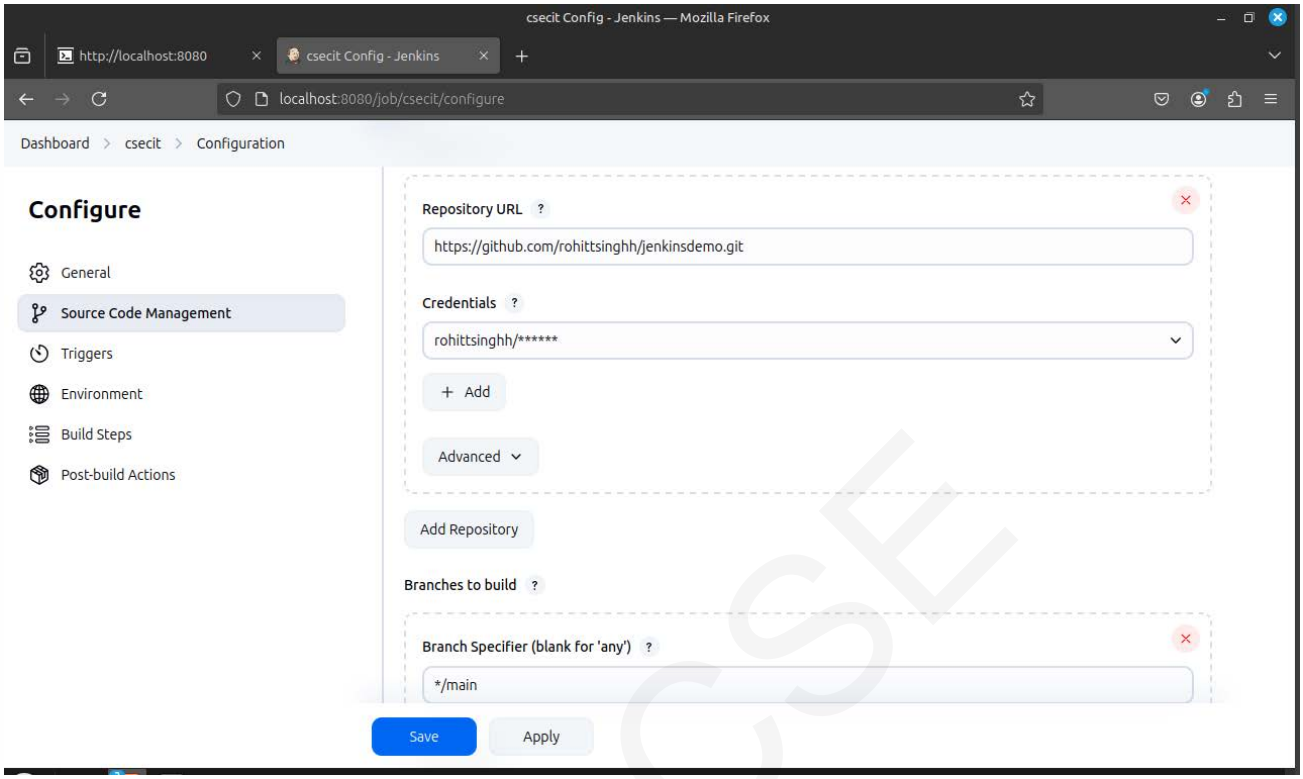
```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.4.0:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.4.0:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.4.0:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] Archetype repository not defined. Using the one from [org.apache.maven.archetypes:maven-archetype-quickstart:1.5] found in catalog remote
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.4/maven-archetype-quickstart-1.4.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.4/maven-archetype-quickstart-1.4.jar (7
.1 kB at 25 kB/s)
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: maven-archetype-quickstart:1.4
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: demo
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: packageInPathFormat, Value: com/example
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: demo
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: /home/cse-cit/demo
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.158 s
[INFO] Finished at: 2025-06-04T17:59:46+05:30
[INFO] ------------------------------------------------------------------------
cse-cit@csecit-VirtualBox:~/demo$
```

Step 6: Open Jenkins and Create Job
Access Jenkins at http://localhost:8080, click 'New Item', name it, and select 'Freestyle project'.



Step 7: Configure SCM

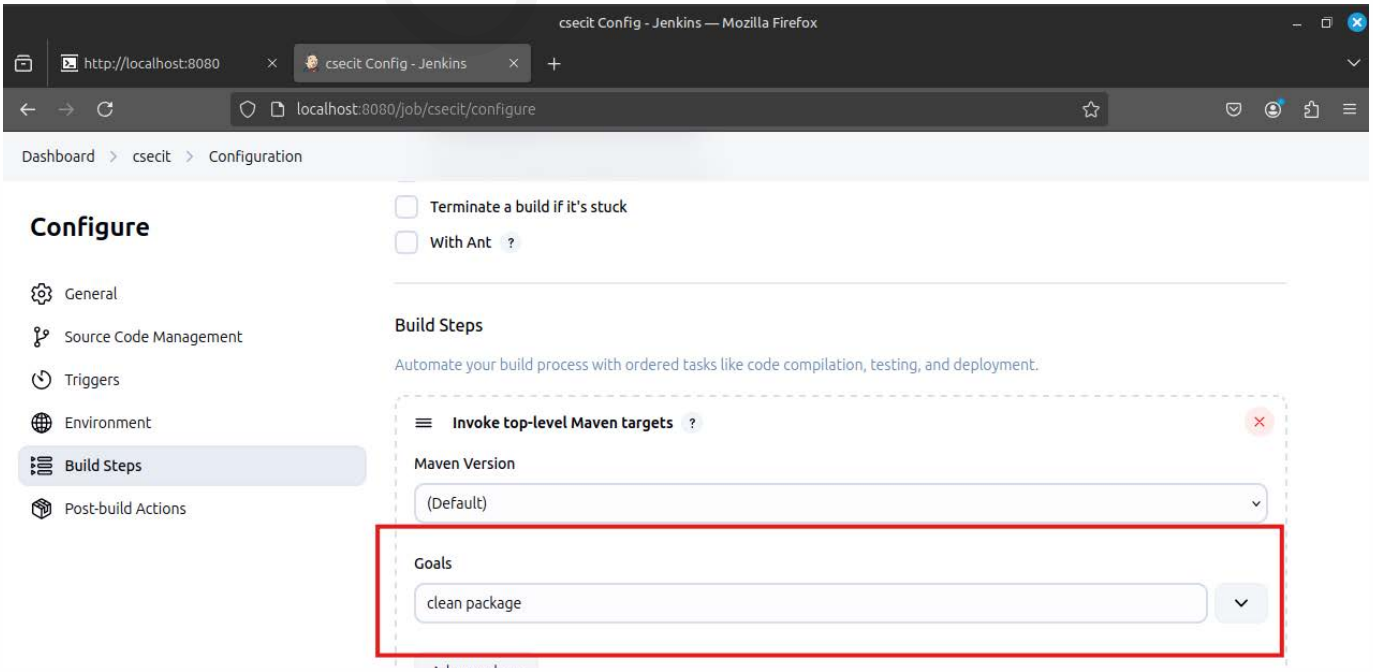
In job configuration > Source Code Management:
Add GitHub Repository URL and credentials.
Set Branch to build: */main



Step 8: Add Maven Build Step

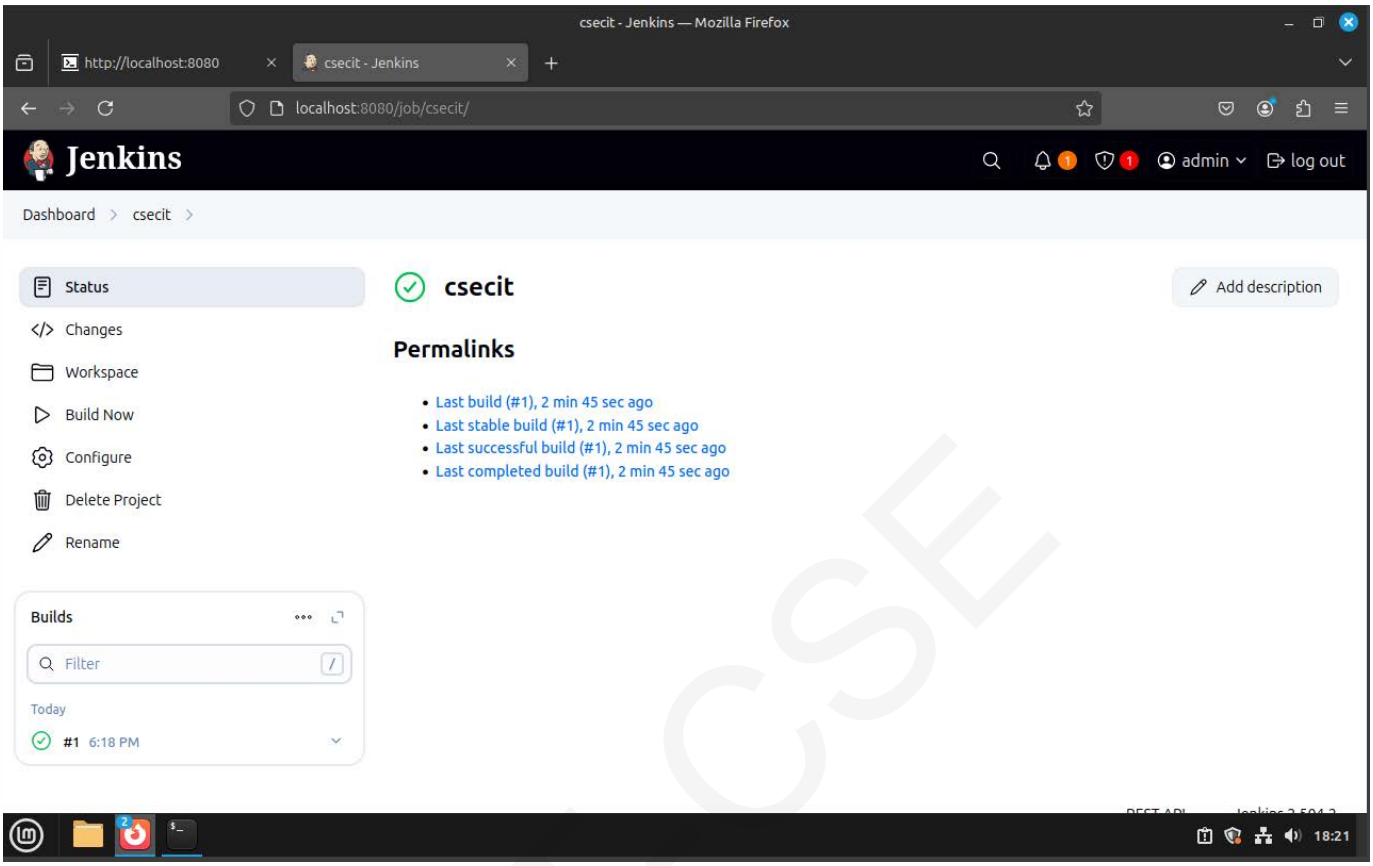
In 'Build Steps', select 'Invoke top-level Maven targets'.
Set Goals to:

clean package



Step 9: Build the Project

Click 'Build Now' from project dashboard.
It fetches code, builds, and packages the Maven project.



Step 10: Use Ansible to Deploy Artifact

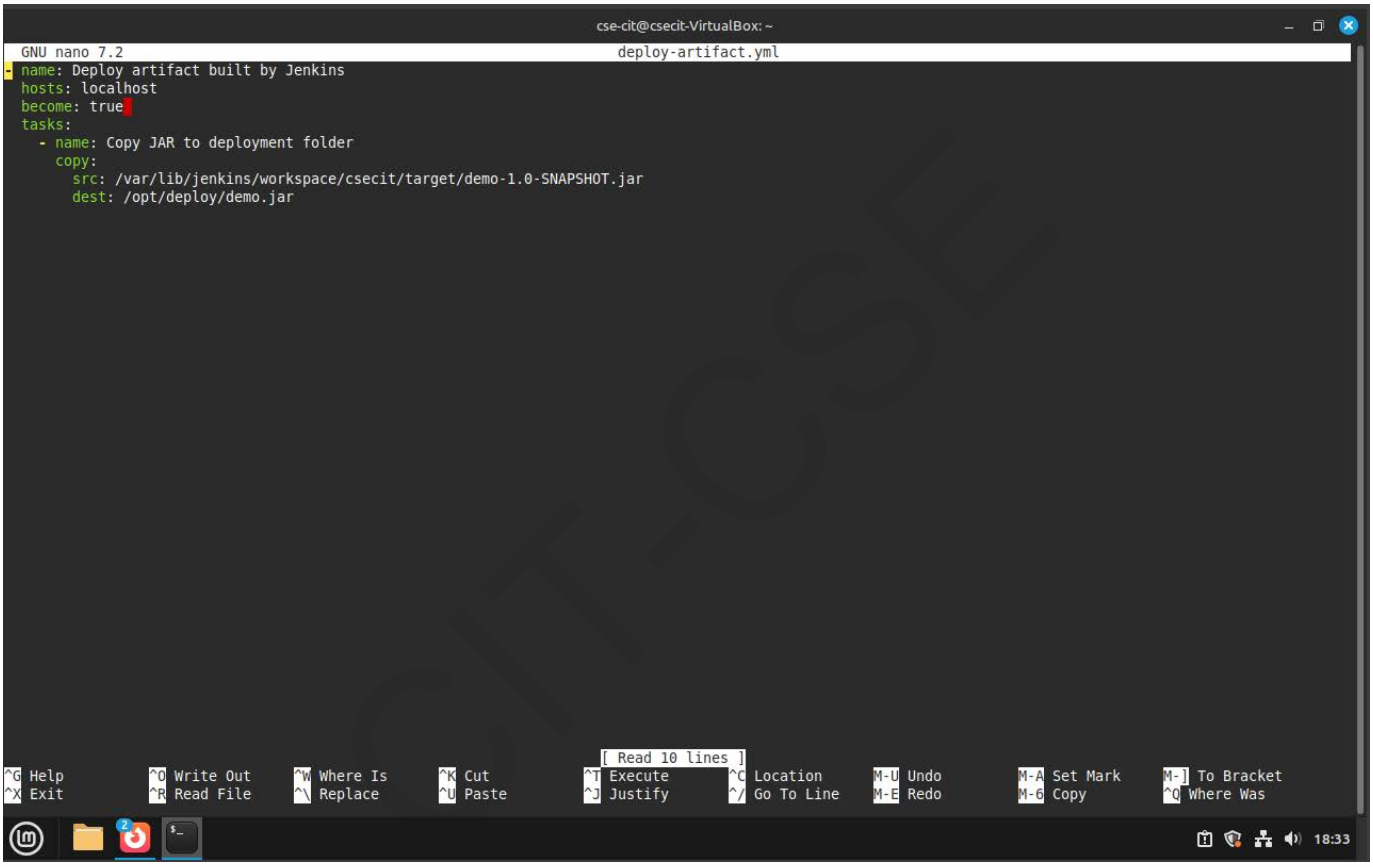
Write a playbook like deploy-artifact.yml to copy and deploy the generated JAR/WAR file.
Run: ansible-playbook deploy-artifact.yml

```
se-cit@csecit-VirtualBox:~$ nano deploy-artifact.yml
```

(Continue on next page)

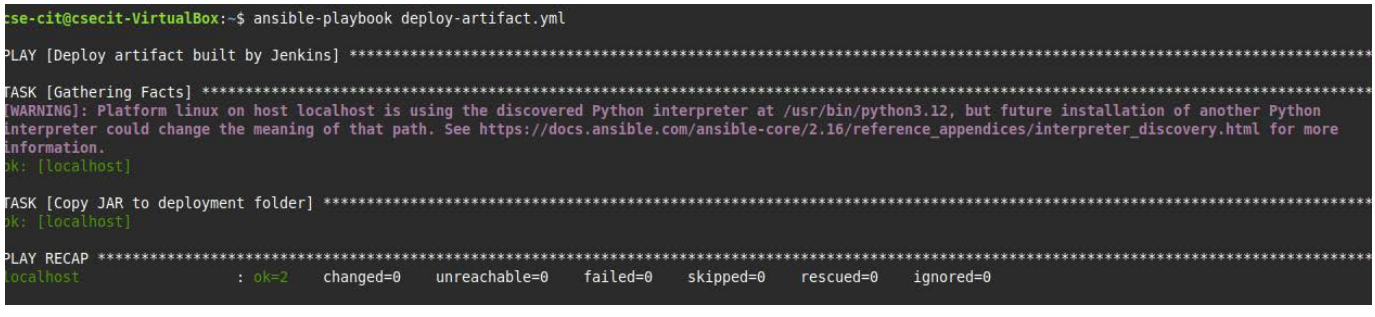
Ansible Playbook (deploy-artifact.yml)

```
---
- name: Deploy artifact built by Jenkins
 hosts: localhost
 become: true
 tasks:
 - name: Copy JAR to deployment folder
 copy:
 src: /var/lib/jenkins/workspace/csecit/target/demo-1.0-SNAPSHOT.jar
 dest: /opt/deploy/dem
```



Playbook Execution Output
Command Executed:

ansible-playbook deploy-artifact.yml



PROGRAM 9: ***Introduction to Azure DevOps: Overview of Azure DevOps Services***

Setting Up an Azure DevOps Account and Project.

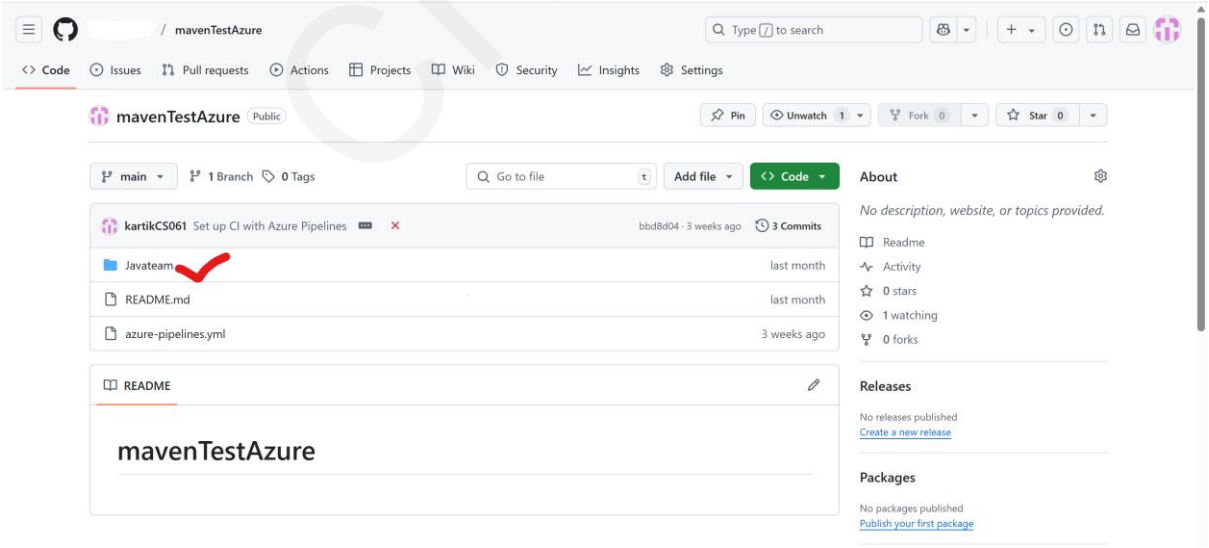
STEPS FOR THE MICROSOFT AZURE

STEP 1: Create a maven project in the local System, command:

```
mvn archetype:generate -DgroupId=com.dineshonjava -DartifactId=Javateam -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

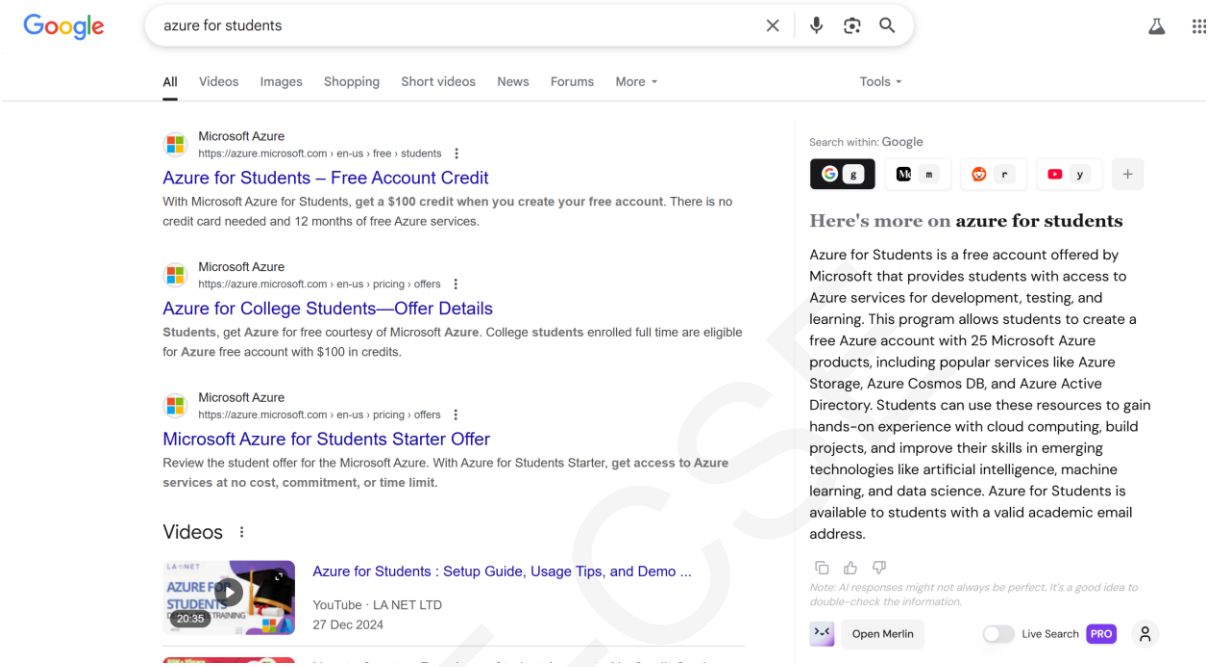
STEP 2: Push it to GitHub, commands:

```
git init
git add .
git commit -m "Your-Message"
git branch -M main
git remote add origin https://github.com/YOUR -URL
git push -u origin main
```

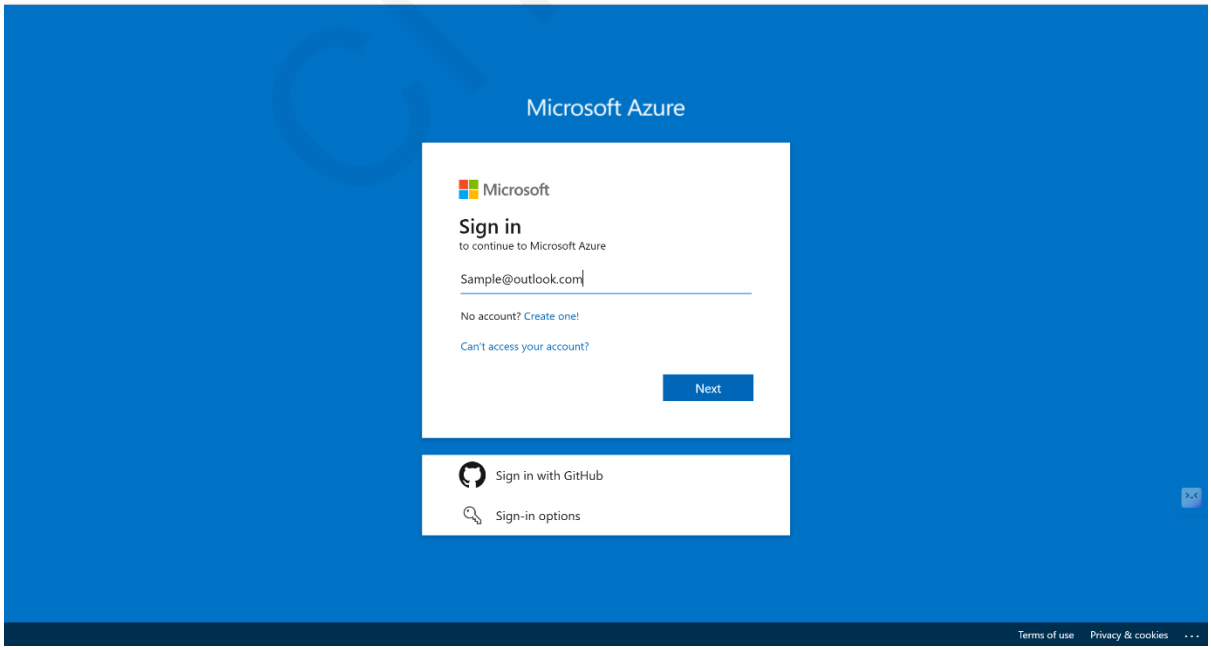


STEPS FOR THE MICROSOFT AZURE

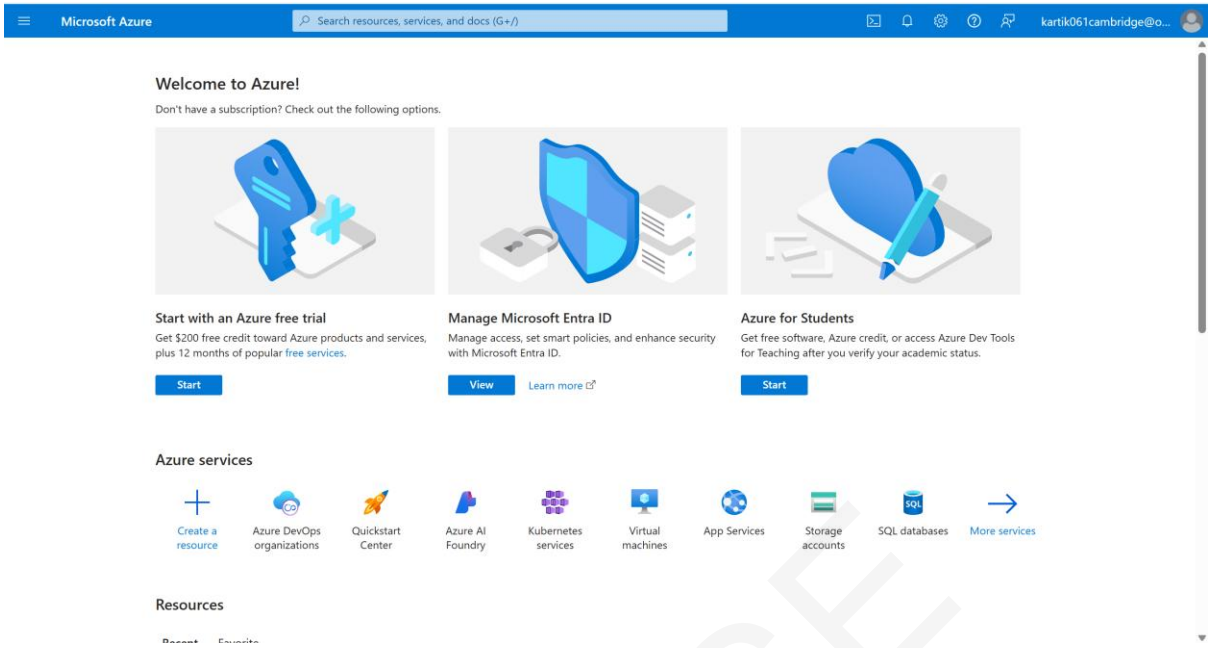
STEP 1: In the browser search **Azure of Students.**

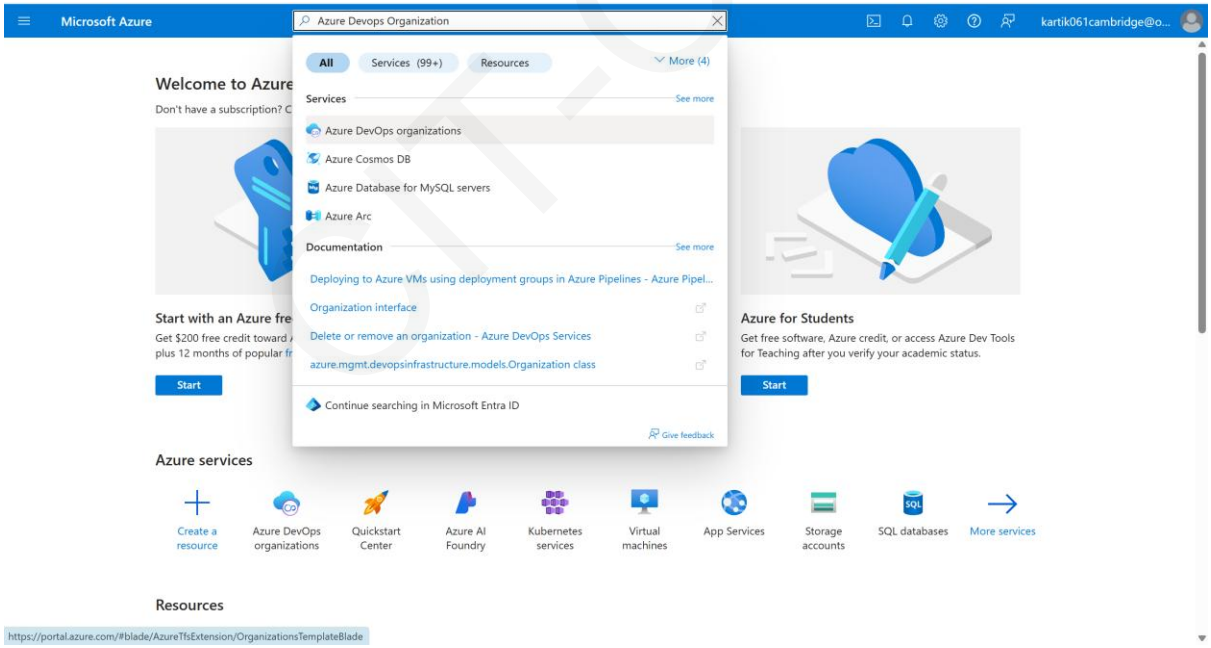


STEP 2: **Click** on the First link and **sing-in**.

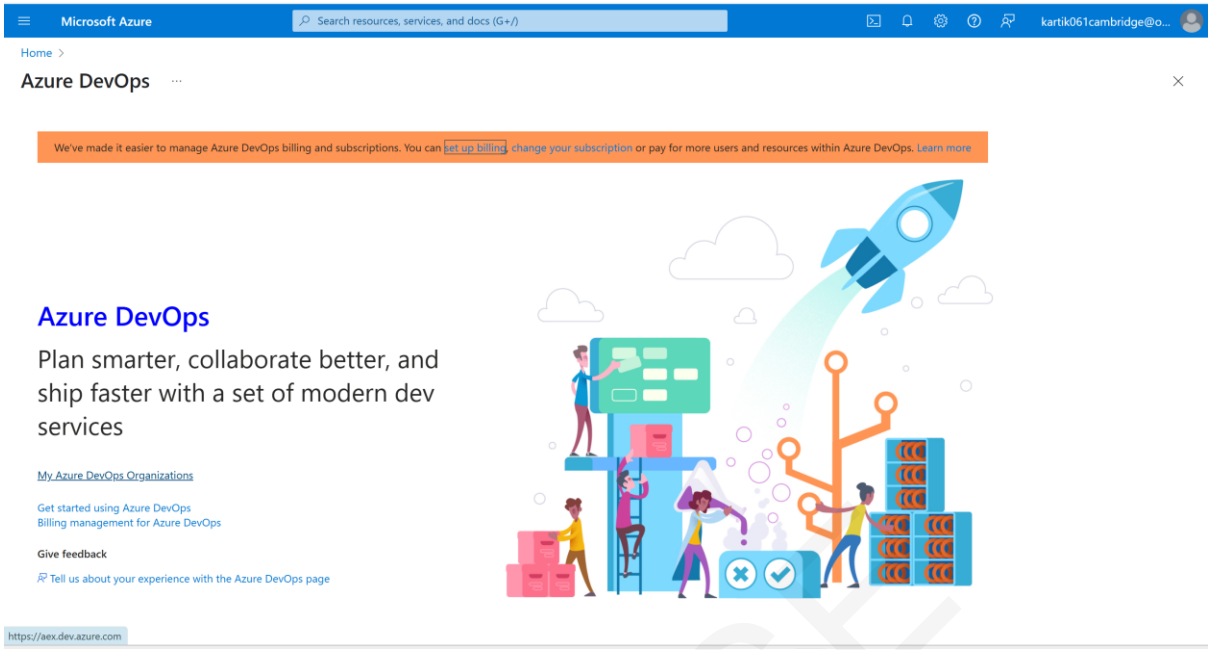


<u>STEP 3</u>: Go to the search bar and type: **Azure Devops organization**



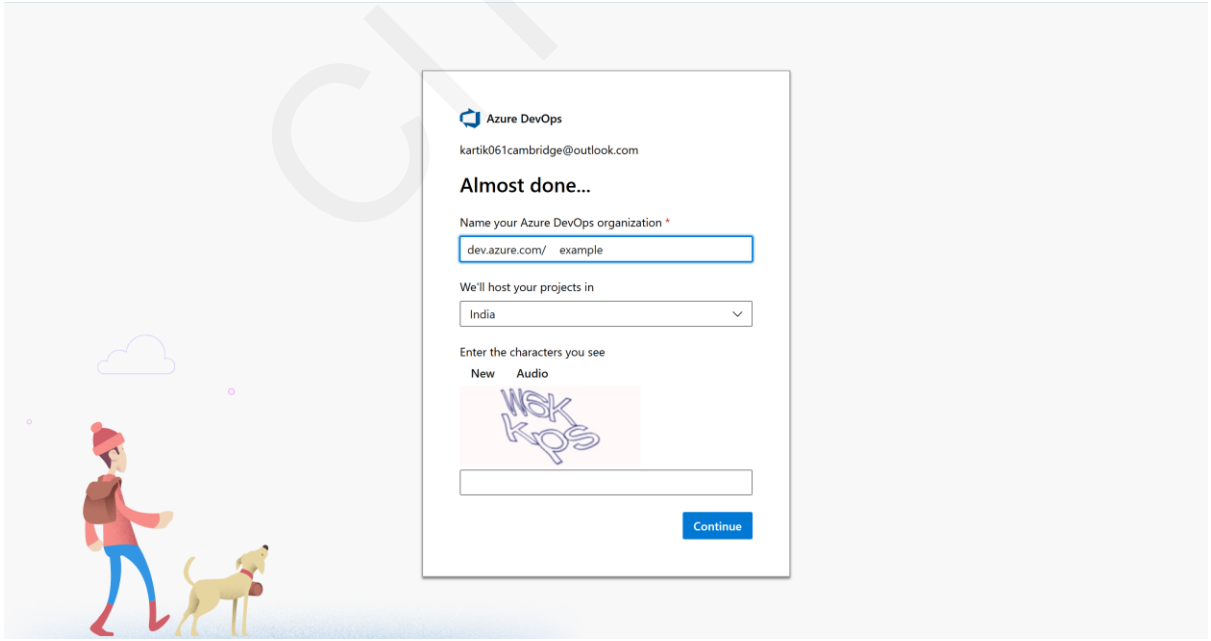


STEP 4: When Page appears, select "**my Azure Devops organization**"



STEP 5: Now Click on **Create New Organization.**

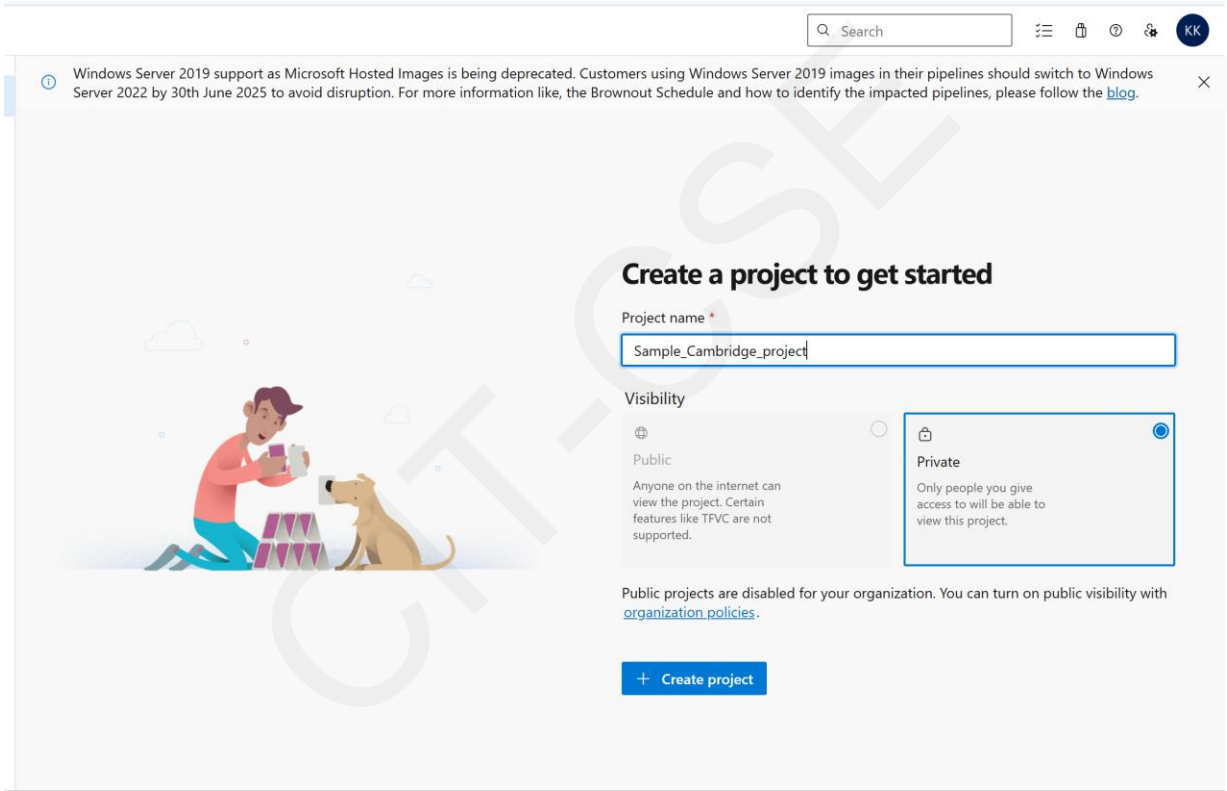
STEP 6: Continue with all the pop-ups and this page will appears.



- Let the Organization name be the default name and select country specific to your location.
- Enter the Captcha

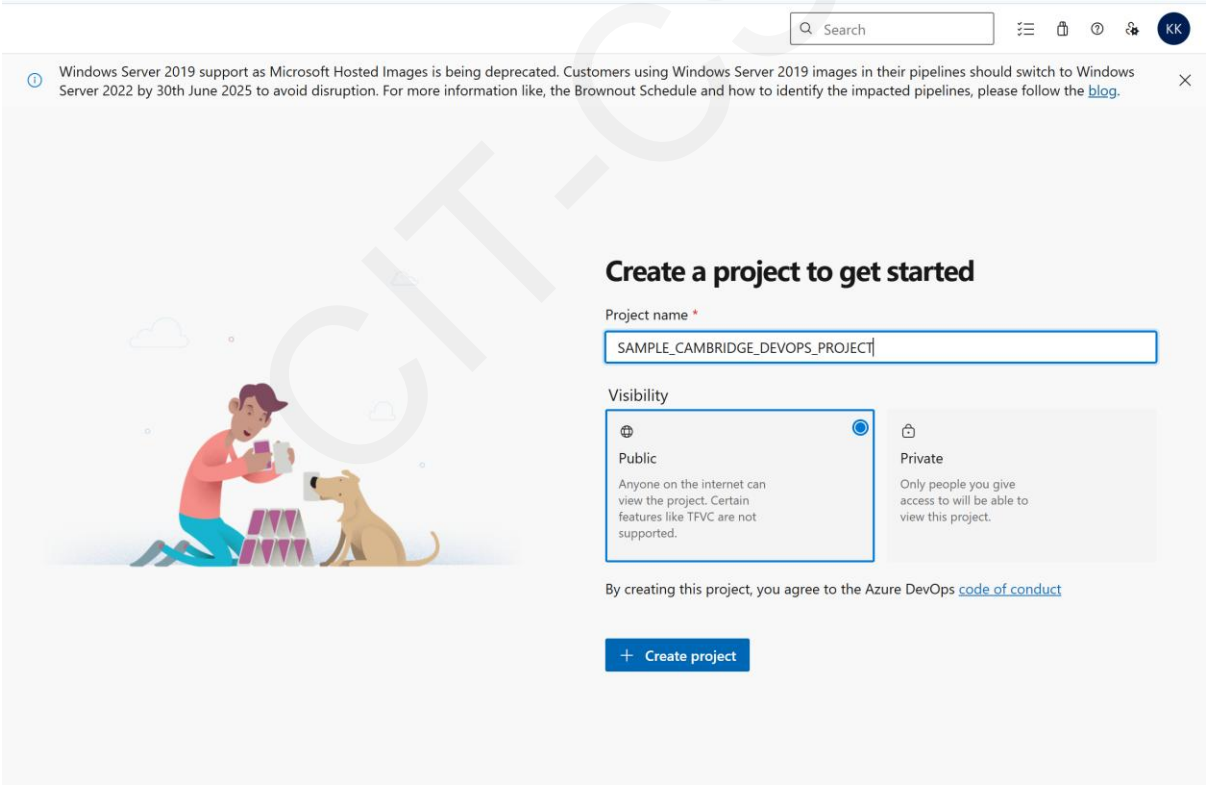
STEP 7: Click **Continue**.

STEP 8: Below page appears, click on the **organization policies**



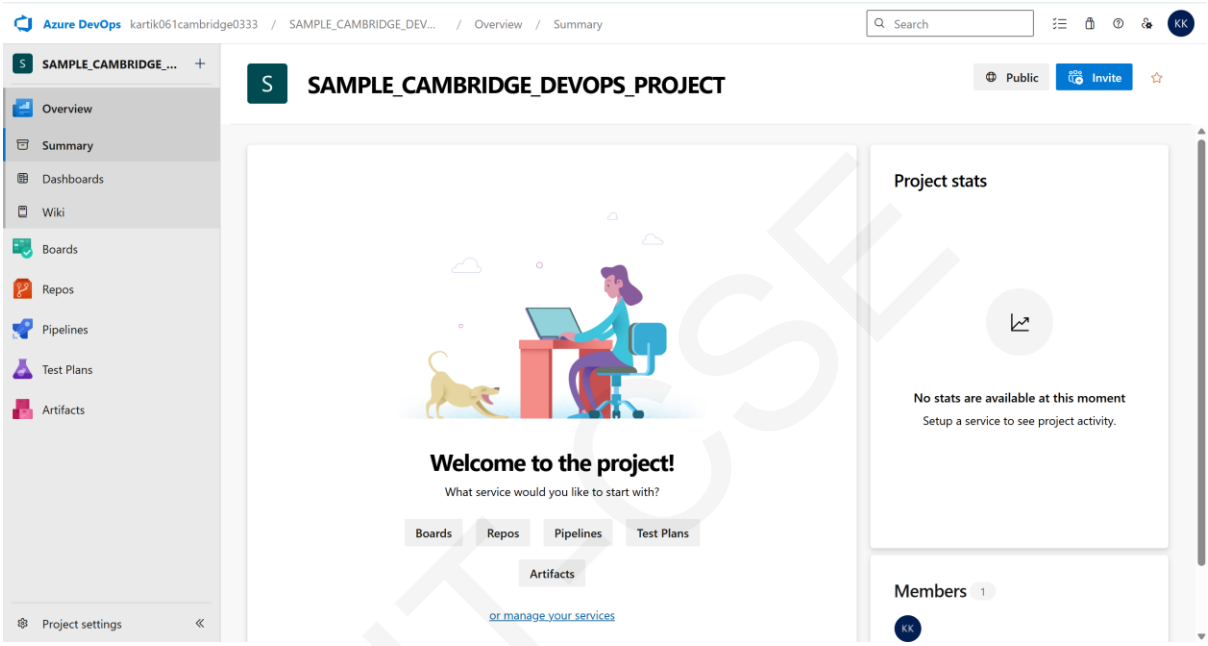
<u>STEP 9</u>: Enable **public project.**

<u>STEP 10</u>: Come back to the project page and give the **project name** and select **Public,** then click on the **"+ Create project".**

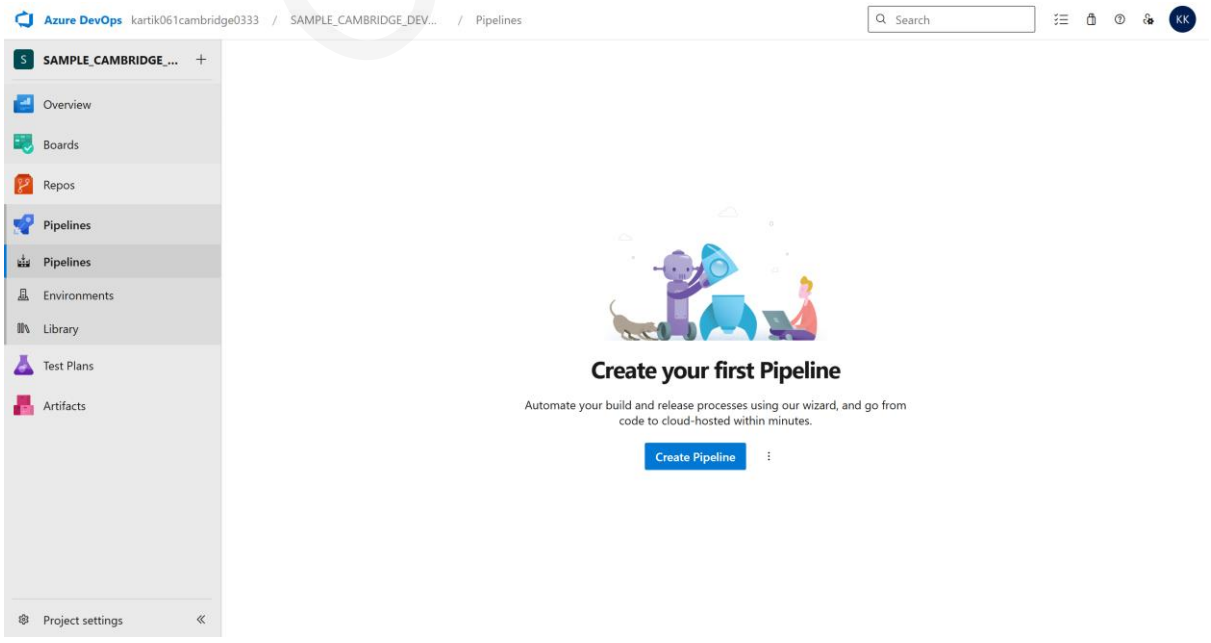


PROGRAM 10 : Creating Build Pipelines: Building a Maven/Gradle Project with Azure Pipelines, Integrating Code Repositories (e.g., GitHub, Azure Repos), Running Unit Tests and Generating Reports.

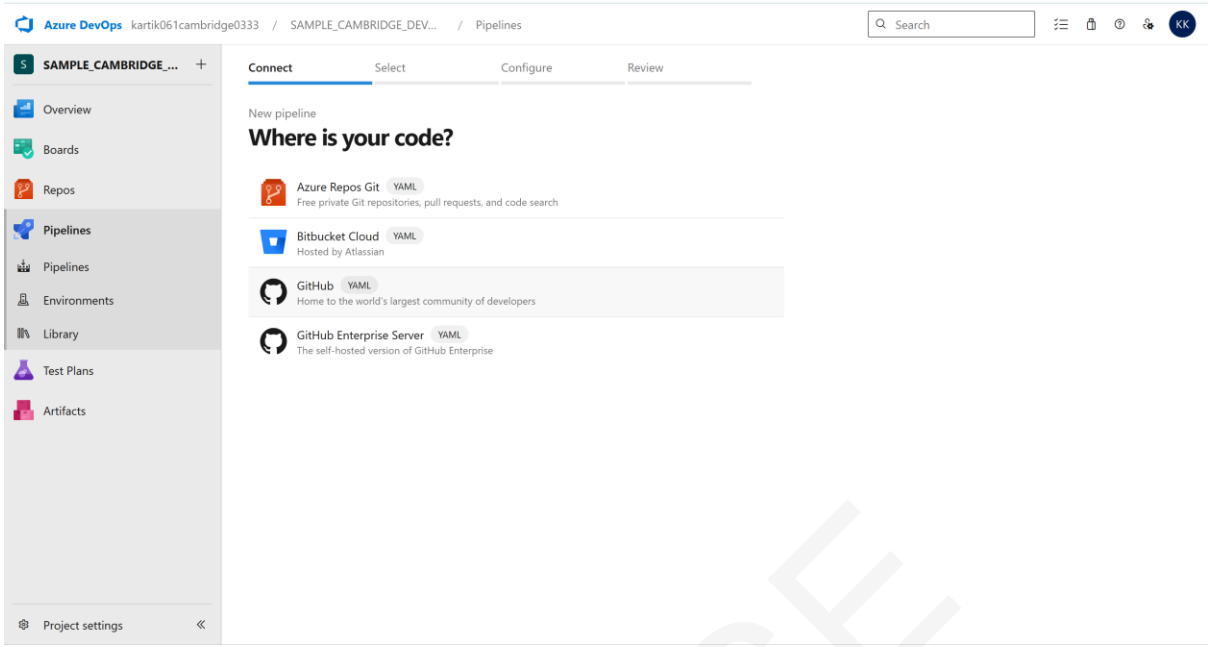
STEP 1: Below page appears, select **Pipeline**



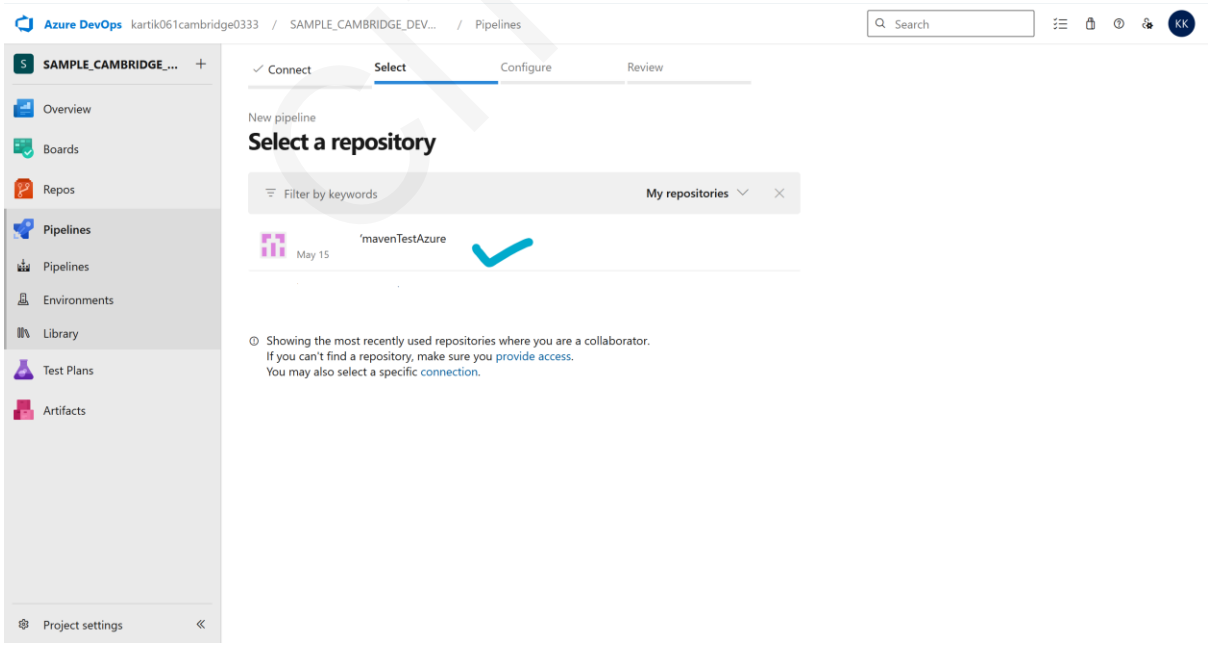
STEP 2: Click **Create Pipeline**



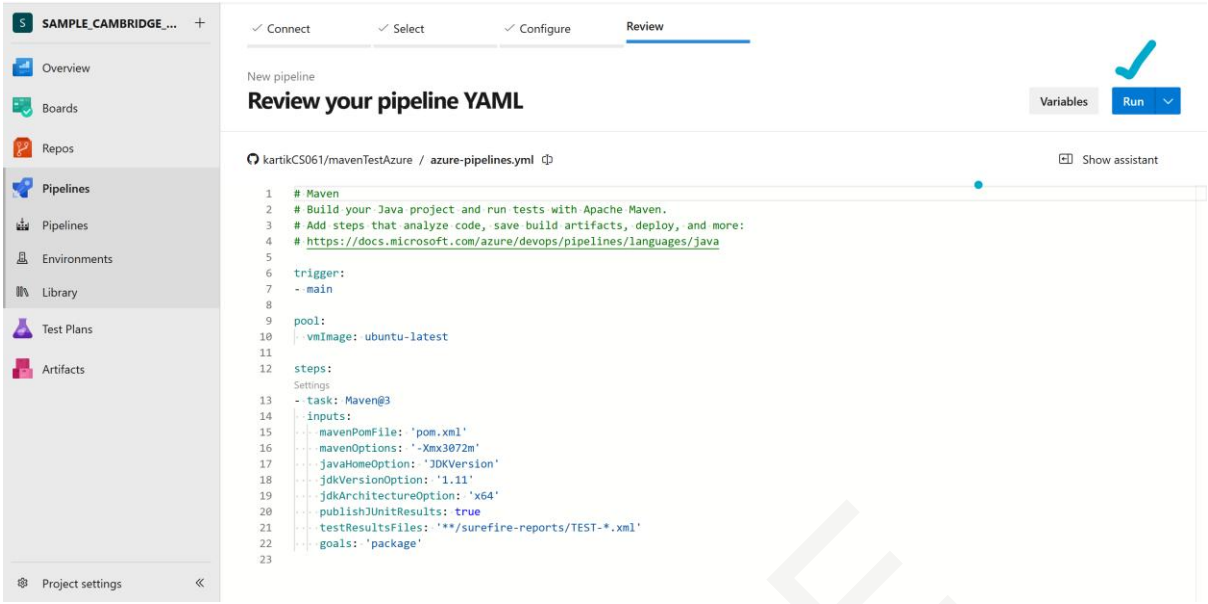
STEP 3: Select **GitHub YAML**



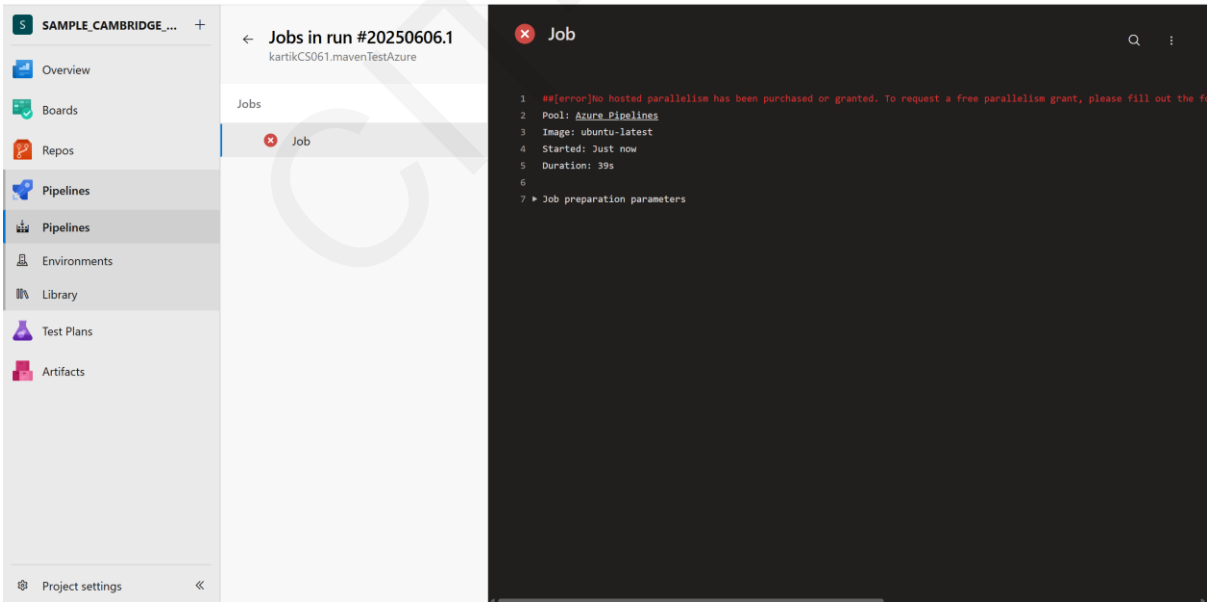
STEP 4: Select the correct repository



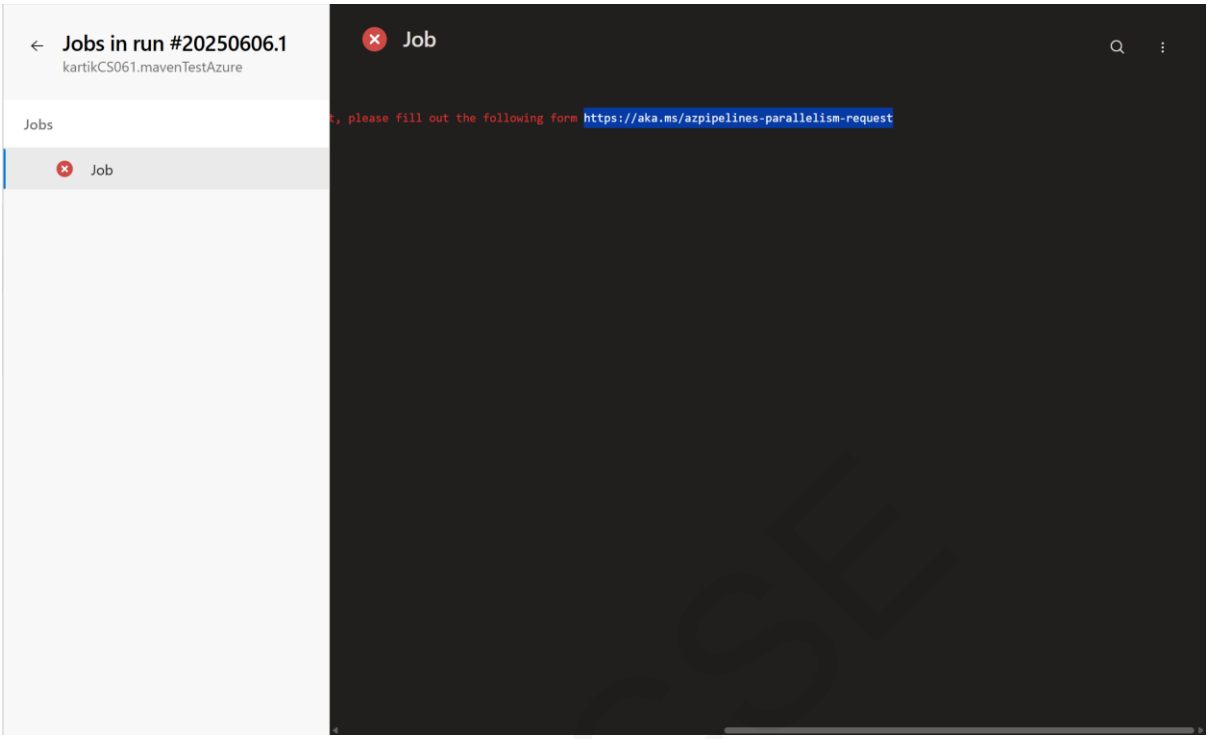
STEP 5: Click on **RUN**



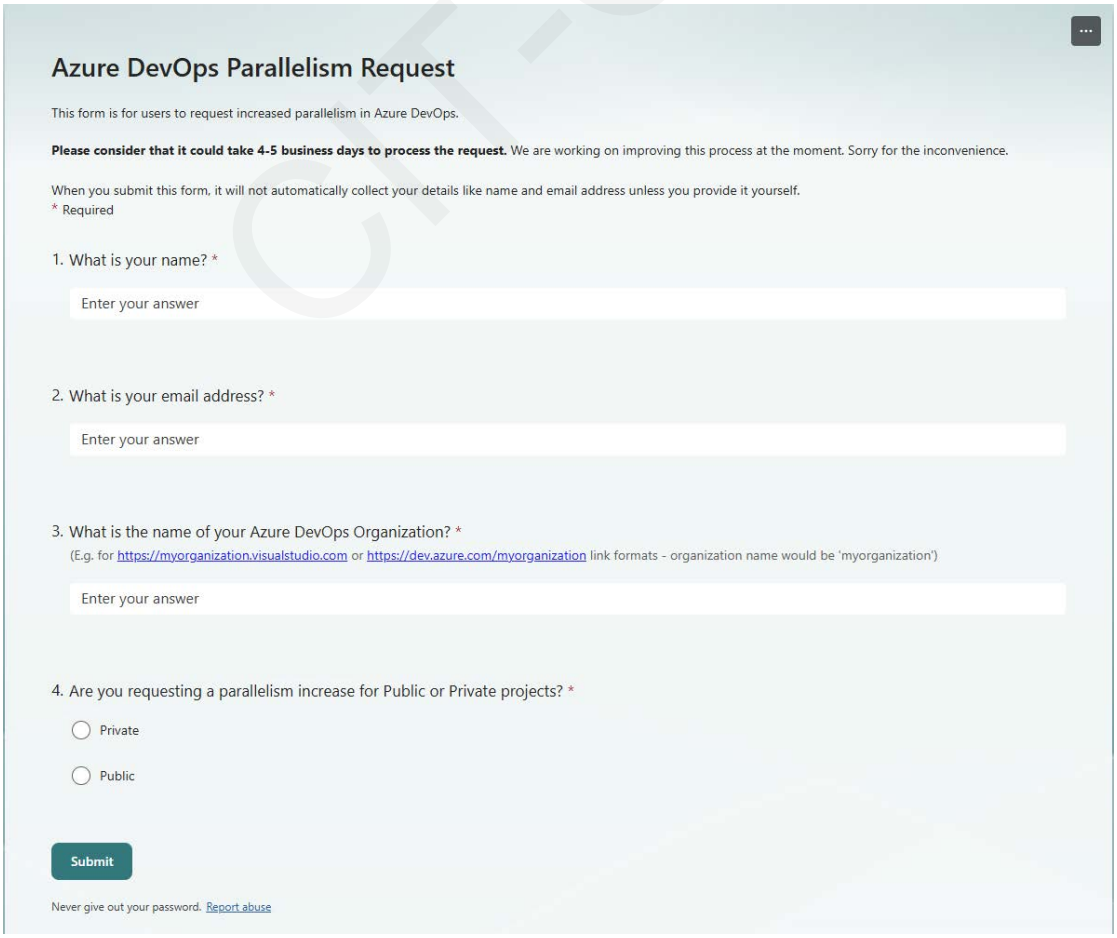
STEP 6: You will get an **ERROR (If you are doing this for the first time).**



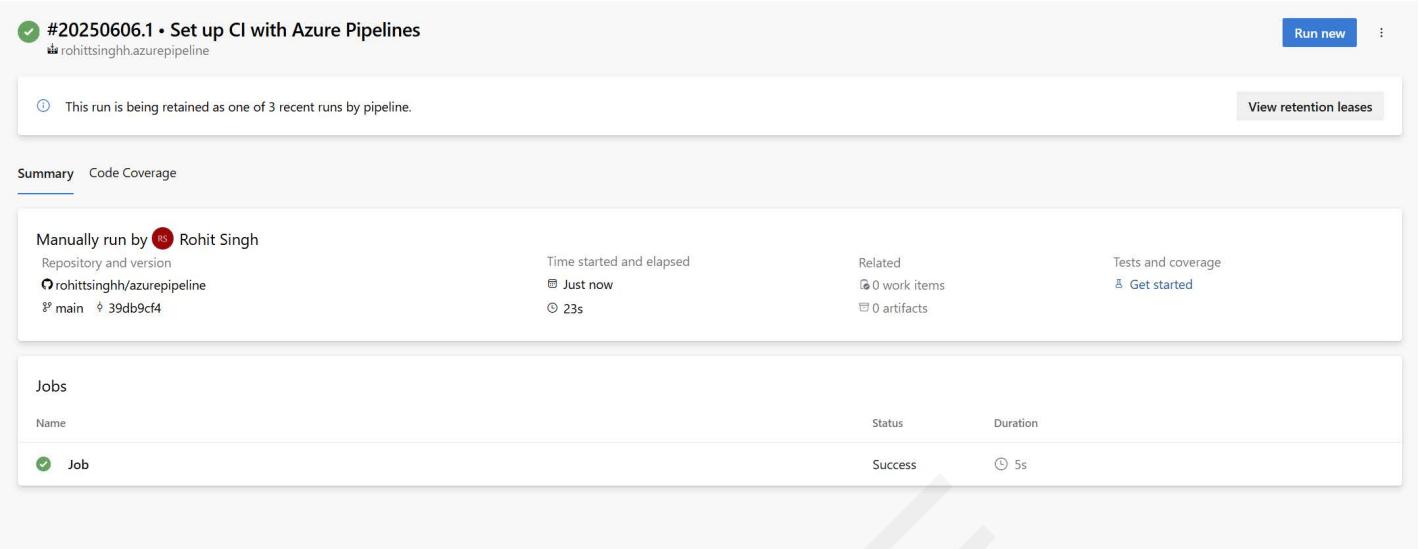
STEP 7: Check out the given link for the form and fill it, you will get the access in ~48 hours.



Fill in the form with registered same email as used on azure , and enter organization name.

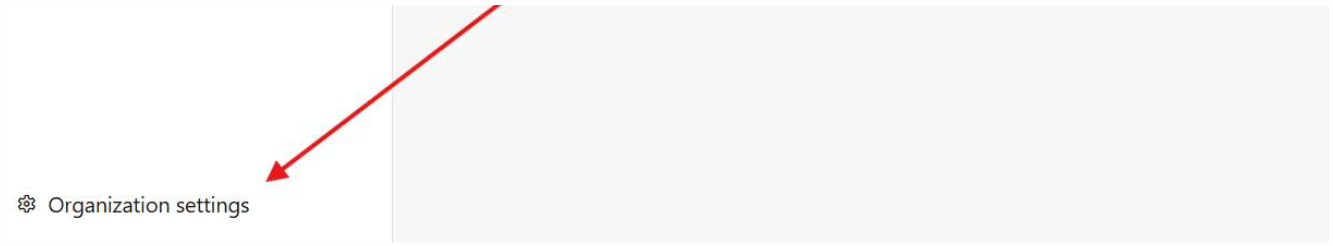


After 48 hours your application will be accepted and now you can Run the Pipeline again , this time it will run succesfully

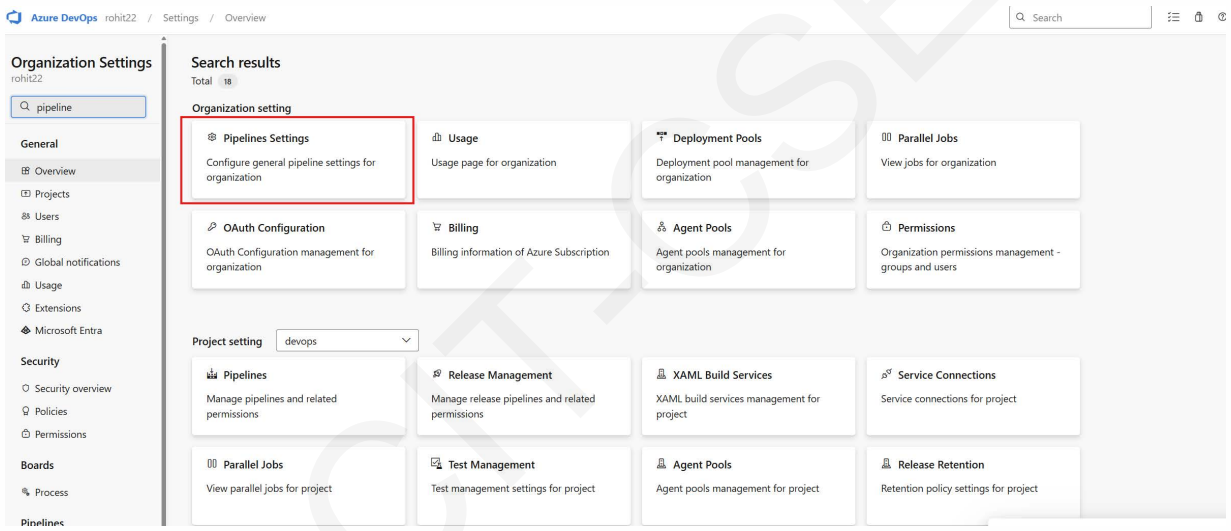


PROGRAM 11 : Creating Release Pipelines: Deploying Applications to Azure App Services,Managing Secrets and Configuration with Azure Key Vault, Hands-On : Continuous Deployment with Azure Pipeline

Go to your Organization and in the bottom left , Click on " Organization Settings "

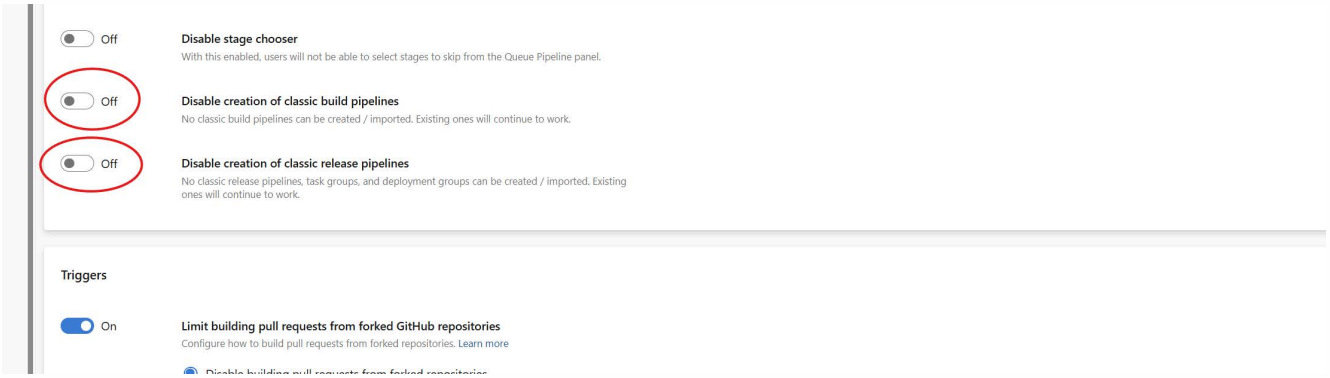


Click on " Pipelines Settings "

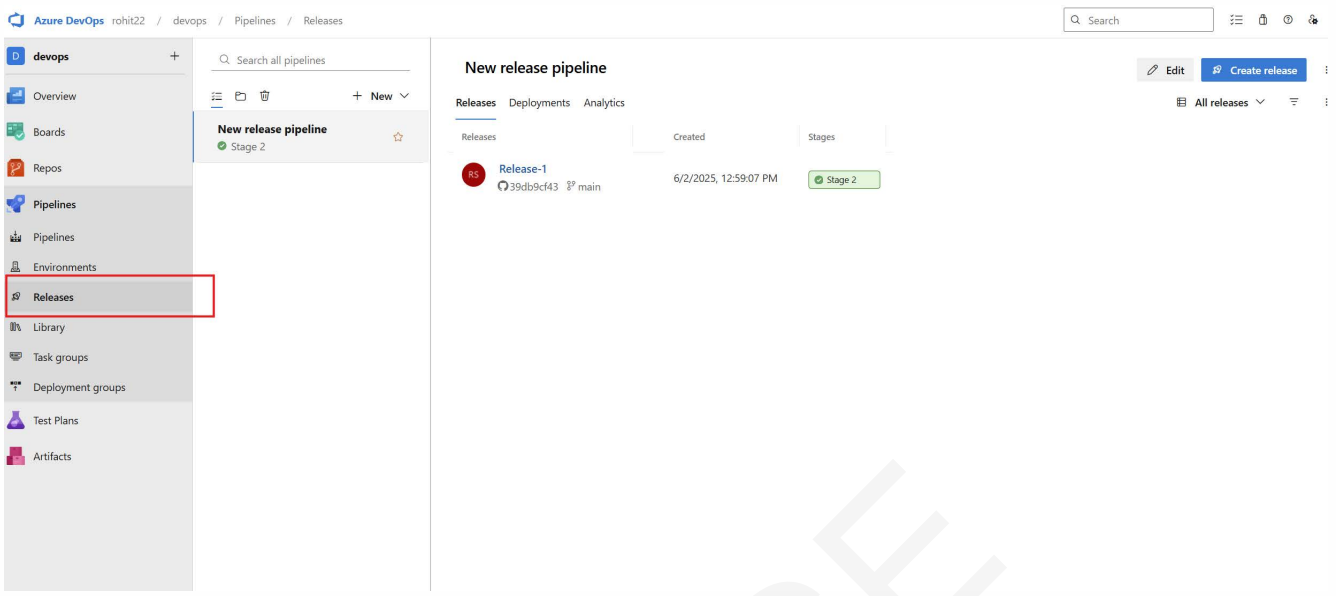


Make sure these are turned off :

1. Disable Creation of Classic build pipelines
2. Disable creation of classic release pipelines



Now go to Pipelines ----> Releases of your Project and click on " Create Release "



Now after this you will get the following output