



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:

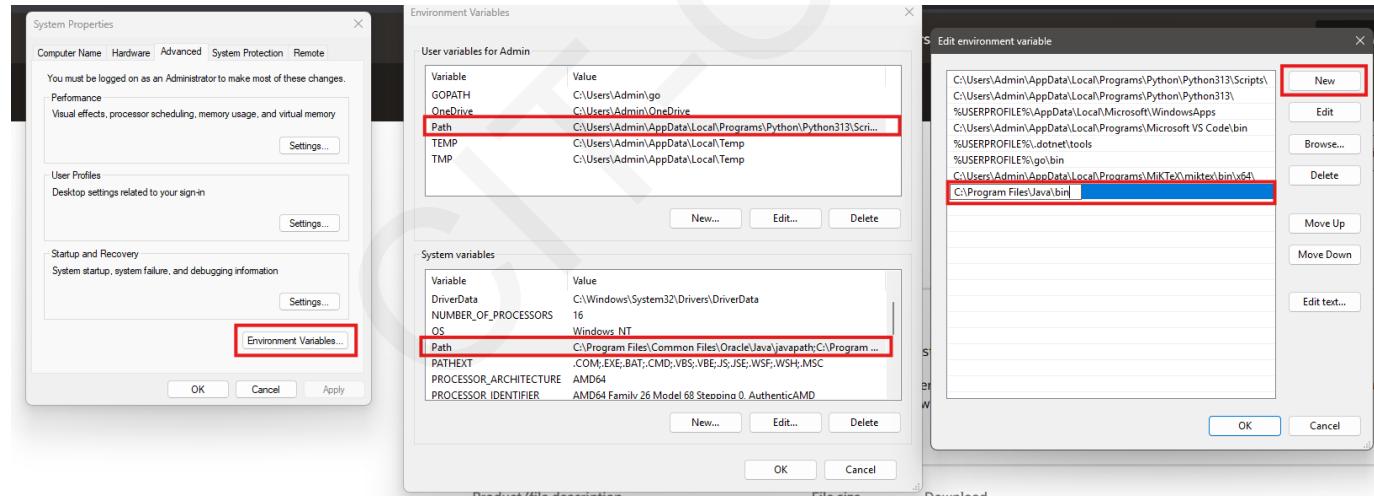
Visit: <https://www.oracle.com/java/technologies/downloads>

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			
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

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.26100.4202]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>gradle -v

Gradle 8.14
-----
Build time: 2025-04-25 09:29:08 UTC
Revision: 34c560e3be961658a6fbcd7170ec2443a228b109

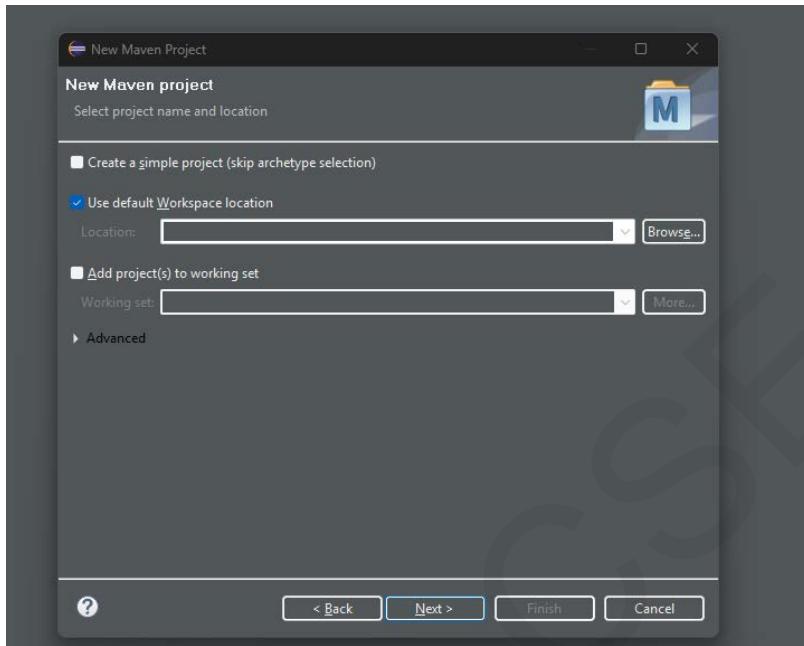
Kotlin: 2.0.21
Groovy: 3.0.24
Ant: Apache Ant(TM) version 1.10.15 compiled on August 25 2024
Launcher JVM: 21.0.7 (Oracle Corporation 21.0.7+8-LTS-245)
Daemon JVM: C:\Program Files\Java (no JDK specified, using current Java home)
OS: Windows 11 10.0 amd64

C:\Windows\System32>
```

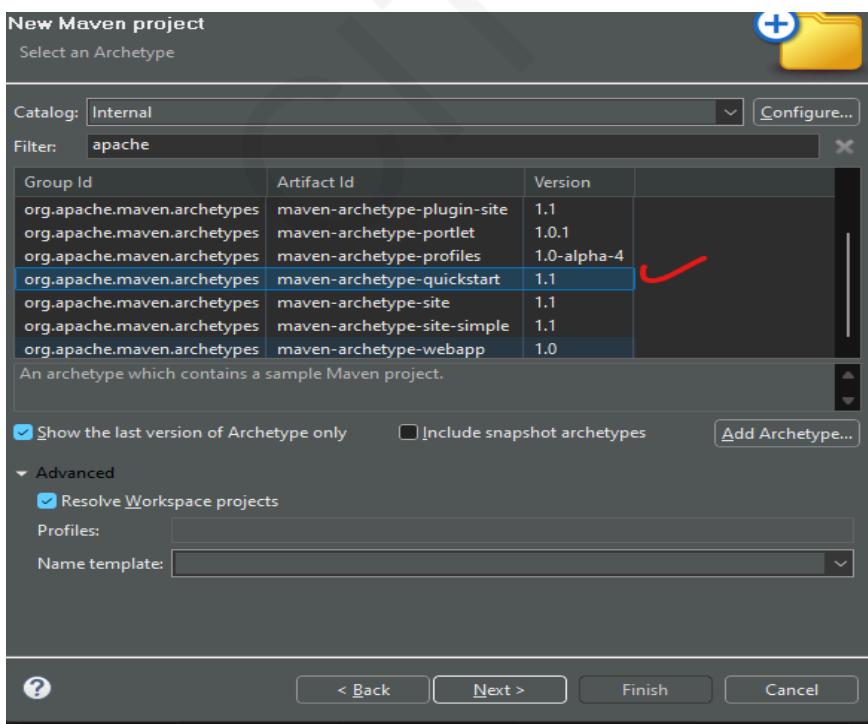
Program 2 : Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins

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.**



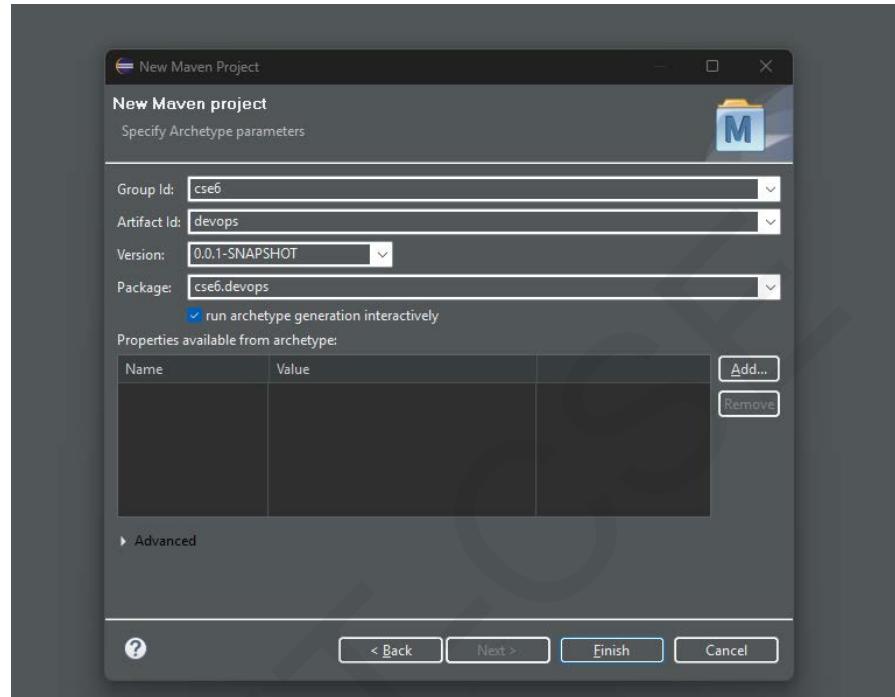
In the Filter box, type "apache" or select Catalog as "Internal".

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).

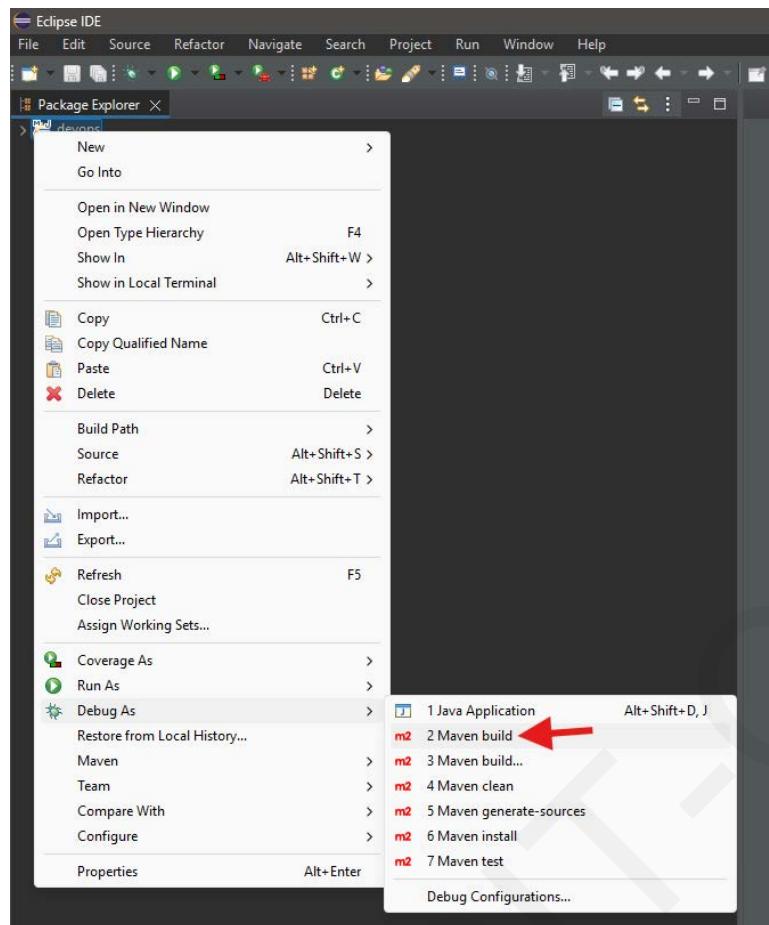
```
C:\eclipse\plugins\org.eclipse.jdt.core\1.1.0.v20220903-1038\jre\bin\javaw.exe (Jun 2, 2025, 5:59:51 PM) [pid: 31000]
Progress (1): 0/17 MB
Progress (1): 0/17 MB

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
```

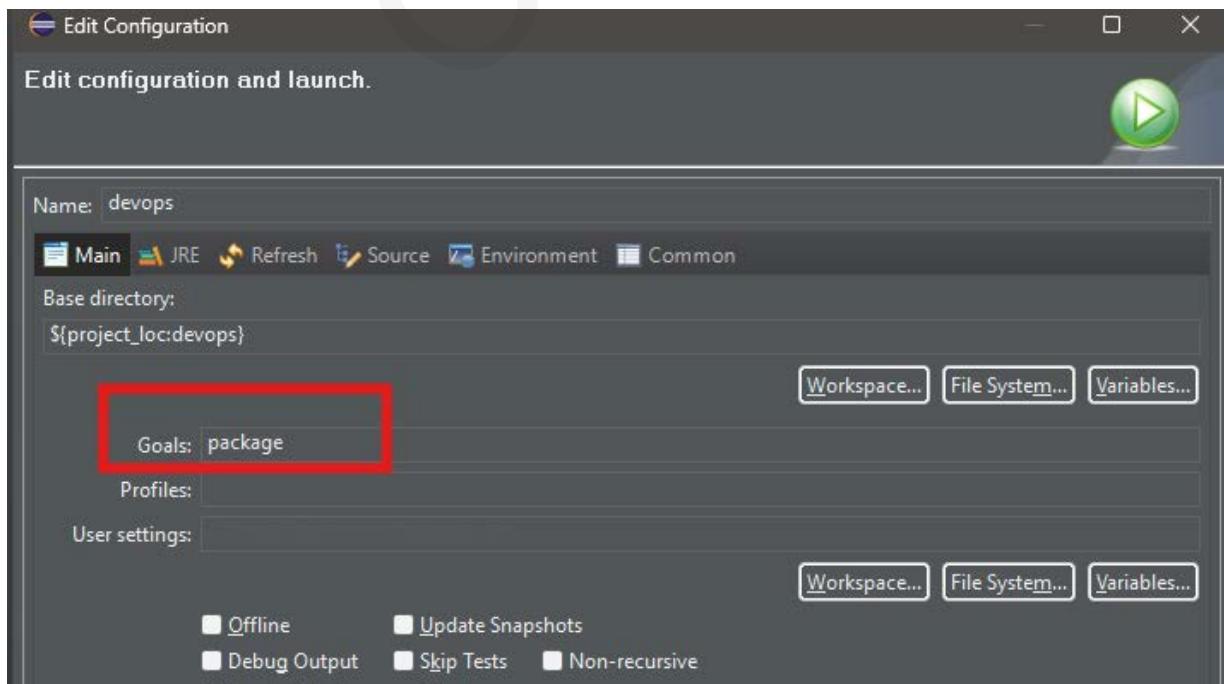
```
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.124 s
[INFO] Finished at: 2025-06-02T18:05:21+05:30
[INFO] -----
```

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

Select : Run As → Maven Build

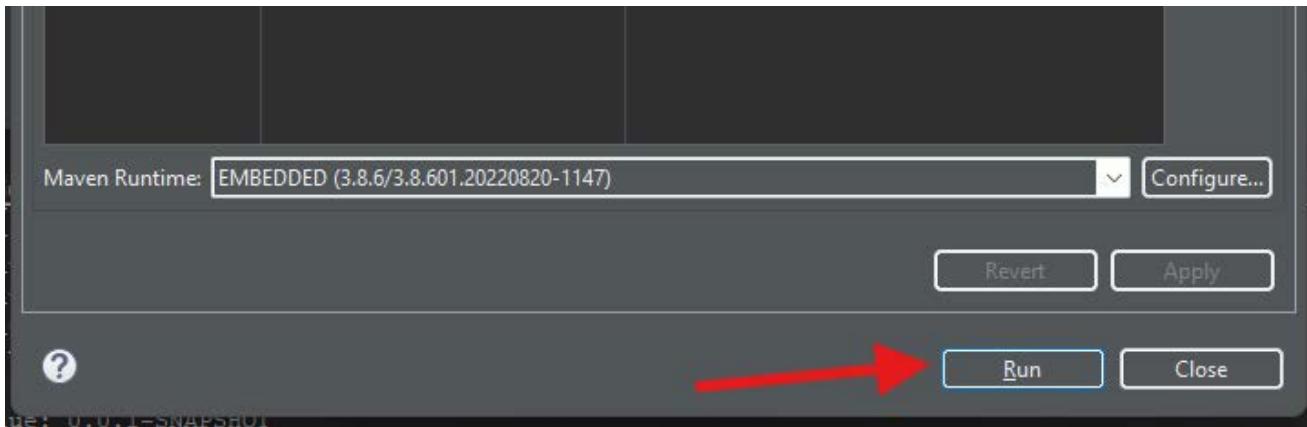


In the Goals field, enter : **package**



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/

The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer view, which contains a project named 'devops' with a package 'cse6.devops' containing an 'App.java' file. The 'App.java' file is open in the center editor, displaying the following code:

```
1 package cse6.devops;
2
3 /**
4  * Hello world!
5  */
6 public class App
7 {
8     public static void main( String[] args )
9     {
10         System.out.println( "Hello World!" );
11     }
12 }
```

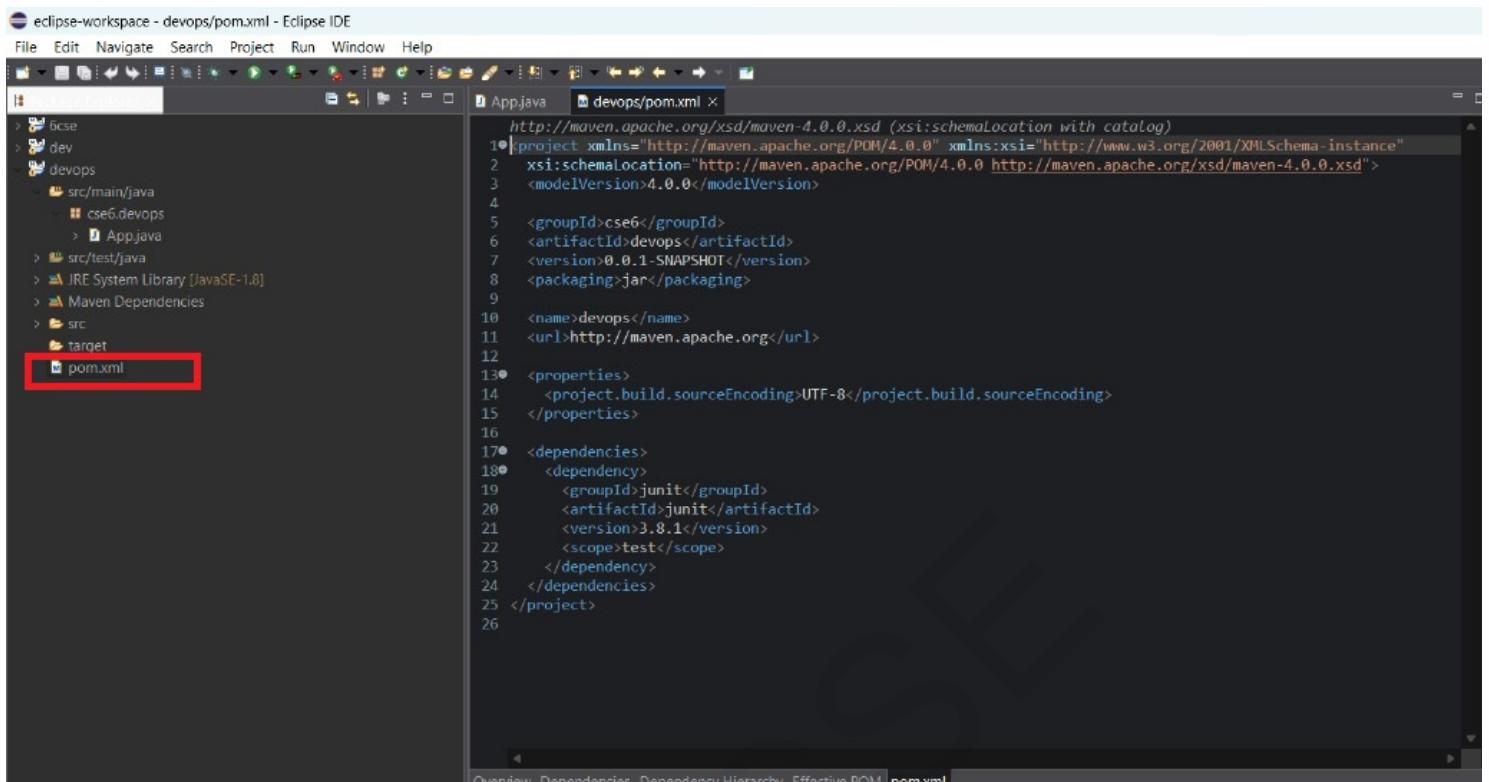
On the right is the Package Explorer view, showing the same project structure. Below the editors is the Terminal view, which displays the Maven build output:

```
<terminated> devops [Maven Build] C:\Users\upendr\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (03-Jun-2025, 11:47:51 am - 11:48:06 am elapsed: 55s)
[INFO] Running cse6.devops.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.024 s -- in cse6.devops.AppTest
[INFO]
[INFO] Results:
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ devops ---
[INFO] Building jar: C:\Users\upendr\eclipse-workspace\devops\target\devops-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
```

Below is the expected Output

The screenshot shows the Eclipse IDE's Terminal view. It displays the Maven build output, which includes the execution of unit tests and the creation of a JAR file. The final line of output is 'Hello World!', indicating the successful execution of the application.

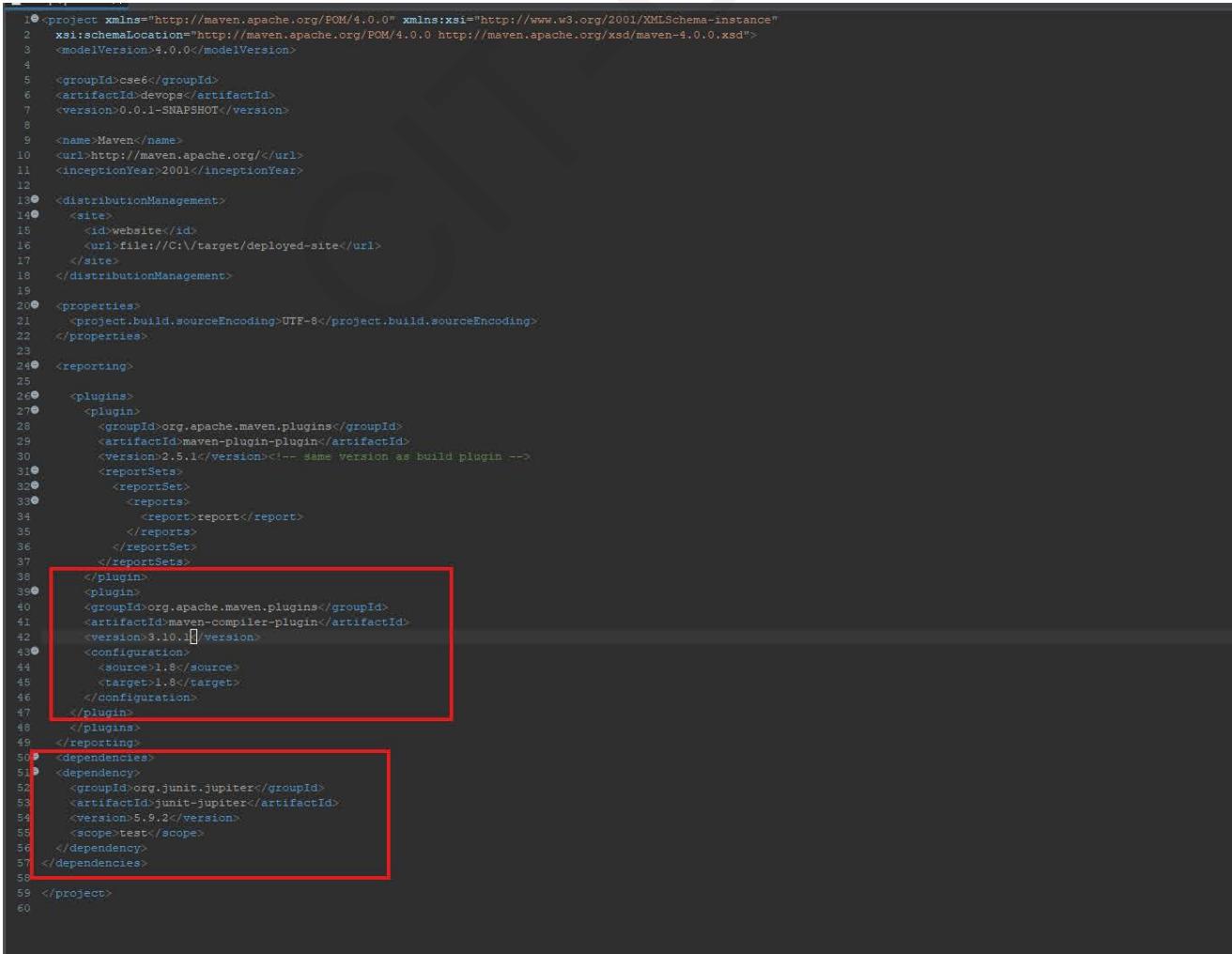
This is the pom.xml file



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - devops/pom.xml - Eclipse IDE". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, Help. The left sidebar shows a project tree with nodes: 6cse, dev, devops (selected), src/main/java (with sub-node cse6.devops), src/test/java, JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, and pom.xml (highlighted with a red box). The main editor area displays the content of the pom.xml file:

```
<?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>cse6</groupId>
  <artifactId>devops</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>devops</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

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



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - devops/pom.xml - Eclipse IDE". The main editor area displays the content of the pom.xml file, with several sections highlighted by red boxes:

- A red box highlights the section from line 13 to line 18, which defines the distribution management site configuration.
- A red box highlights the section from line 26 to line 47, which defines the reporting plugin configuration, specifically the maven-plugin-plugin.
- A red box highlights the section from line 50 to line 57, which defines the dependency section for the junit-jupiter plugin.

```
<?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>cse6</groupId>
  <artifactId>devops</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Maven</name>
  <url>http://maven.apache.org</url>
  <inceptionYear>2001</inceptionYear>
  <distributionManagement>
    <site>
      <id>website</id>
      <url>file:///C:/target/deployed-site</url>
    </site>
  </distributionManagement>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-plugin-plugin</artifactId>
        <version>2.5.1</version><!-- Same version as build plugin --&gt;
        &lt;reportSets&gt;
          &lt;reportSet&gt;
            &lt;reports&gt;
              &lt;report&gt;report&lt;/report&gt;
            &lt;/reports&gt;
          &lt;/reportSet&gt;
        &lt;/reportSets&gt;
      &lt;/plugin&gt;
      &lt;plugin&gt;
        &lt;groupId&gt;org.apache.maven.plugins&lt;/groupId&gt;
        &lt;artifactId&gt;maven-compiler-plugin&lt;/artifactId&gt;
        &lt;version&gt;3.10.1&lt;/version&gt;
        &lt;configuration&gt;
          &lt;source&gt;1.8&lt;/source&gt;
          &lt;target&gt;1.8&lt;/target&gt;
        &lt;/configuration&gt;
      &lt;/plugin&gt;
    &lt;/plugins&gt;
  &lt;/reporting&gt;
  &lt;dependencies&gt;
    &lt;dependency&gt;
      &lt;groupId&gt;org.junit.jupiter&lt;/groupId&gt;
      &lt;artifactId&gt;junit-jupiter&lt;/artifactId&gt;
      &lt;version&gt;5.9.2&lt;/version&gt;
      &lt;scope&gt;test&lt;/scope&gt;
    &lt;/dependency&gt;
  &lt;/dependencies&gt;
&lt;/project&gt;</pre>
```

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. Just 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**

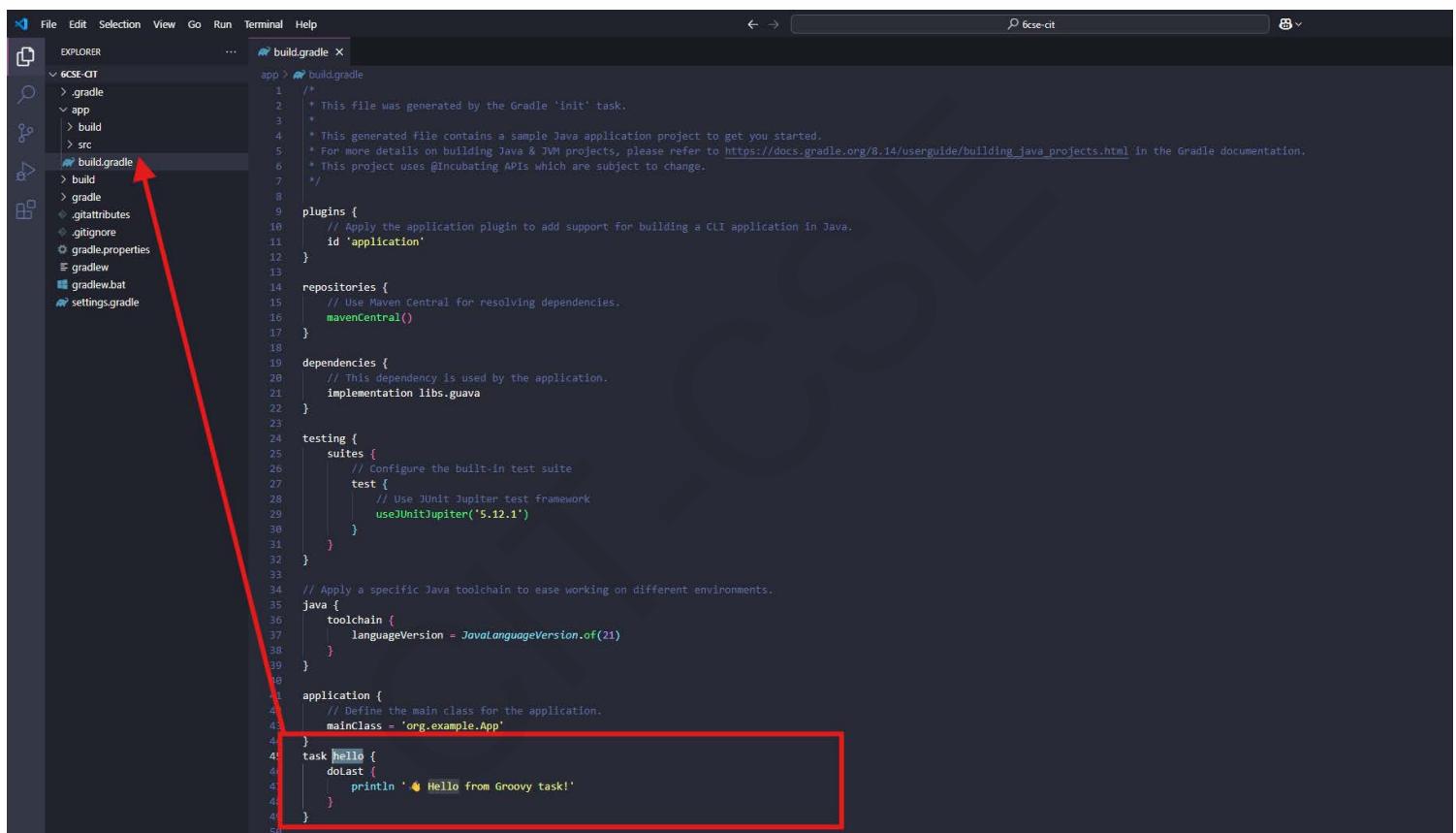
```
C:\Users\Admin\cse-cit>tree  
Folder PATH listing  
Volume serial number is 8CBD-7197  
C:..  
    +-- .gradle  
        +-- 8.14  
            +-- checksums  
            +-- executionHistory  
            +-- expanded  
            +-- fileChanges  
            +-- fileHashes  
            +-- vcsMetadata  
            +-- buildOutputCleanup  
            +-- configuration-cache  
                +-- 3cdc48a-1ec7-413e-a473-51cdalb701d4  
                    +-- 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  
                        +-- af03lt9ur5zaad3twjenxil  
    +-- gradle  
        +-- wrapper
```

Step 3: Add Task in build.gradle

Open build.gradle and at the bottom add:

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

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



A screenshot of the Visual Studio Code interface. The left sidebar shows a project structure with files like .gradle, app, build, src, and build.gradle. A red arrow points from the top-left towards the build.gradle file. The main editor area displays the build.gradle script. A red rectangular box highlights the new task definition at the bottom of the file.

```
/*  
 * This file was generated by the Gradle 'init' task.  
 *  
 * This generated file contains a sample Java application project to get you started.  
 * For more details on building Java & JVM projects, please refer to https://docs.gradle.org/8.14/userguide/building\_java\_projects.html in the Gradle documentation.  
 * This project uses @Incubating APIs which are subject to change.  
 */  
  
plugins {  
    // Apply the application plugin to add support for building a CLI application in Java.  
    id 'application'  
}  
  
repositories {  
    // Use Maven Central for resolving dependencies.  
    mavenCentral()  
}  
  
dependencies {  
    // This dependency is used by the application.  
    implementation libs.guava  
}  
  
testing {  
    suites {  
        // Configure the built-in test suite  
        test {  
            // Use JUnit Jupiter test framework  
            useJUnitJupiter('5.12.1')  
        }  
    }  
}  
  
// Apply a specific Java toolchain to ease working on different environments.  
java {  
    toolchain {  
        languageVersion = JavaLanguageVersion.of(21)  
    }  
}  
  
application {  
    // Define the main class for the application.  
    mainClass = 'org.example.App'  
}  
task hello {  
    doLast {  
        println 'Hello from Groovy task!'  
    }  
}
```

Step 4: Run Custom Task

Execute the task using:

```
gradle hello
```

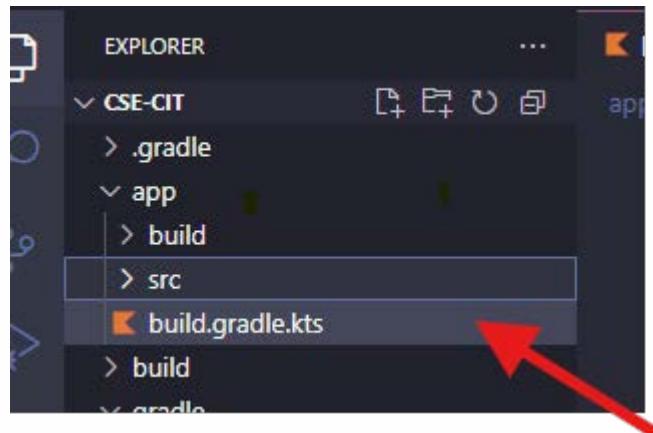
Expected Output:

```
> 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!")  
    }  
}
```



```
/* This file was generated by the Gradle 'init' task.  
 *  
 * This generated file contains a sample Groovy application project to get you started.  
 * For more details on building Java & JVM projects, please refer to https://docs.gradle.org/7.0/userguide/getting\_started.html.  
 * This project uses @Incubating APIs which are subject to change.  
 */  
  
plugins {  
    // Apply the groovy Plugin to add support for Groovy.  
    groovy  
  
    // Apply the application plugin to add support for building a CLI application  
    application  
}  
  
repositories {  
    // Use Maven Central for resolving dependencies.  
    mavenCentral()  
}  
  
dependencies {  
    // Use the latest Groovy version for building this library  
    implementation(libs.groovy.all)  
  
    // This dependency is used by the application.  
    implementation(libs.guava)  
}  
  
testing {  
    suites {  
        // Configure the built-in test suite  
        val test by getting(JvmTestSuite::class)  
        // Use Spock test framework  
        useSpock("2.2-groovy-3.0")  
    }  
}  
  
// Apply a specific Java toolchain to ease working on different environments.  
java {  
    toolchain {  
        languageVersion = JavaLanguageVersion.of(21)  
    }  
}  
  
application {  
    // Define the main class for the application.  
    mainClass = "org.example.App"  
}  
  
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

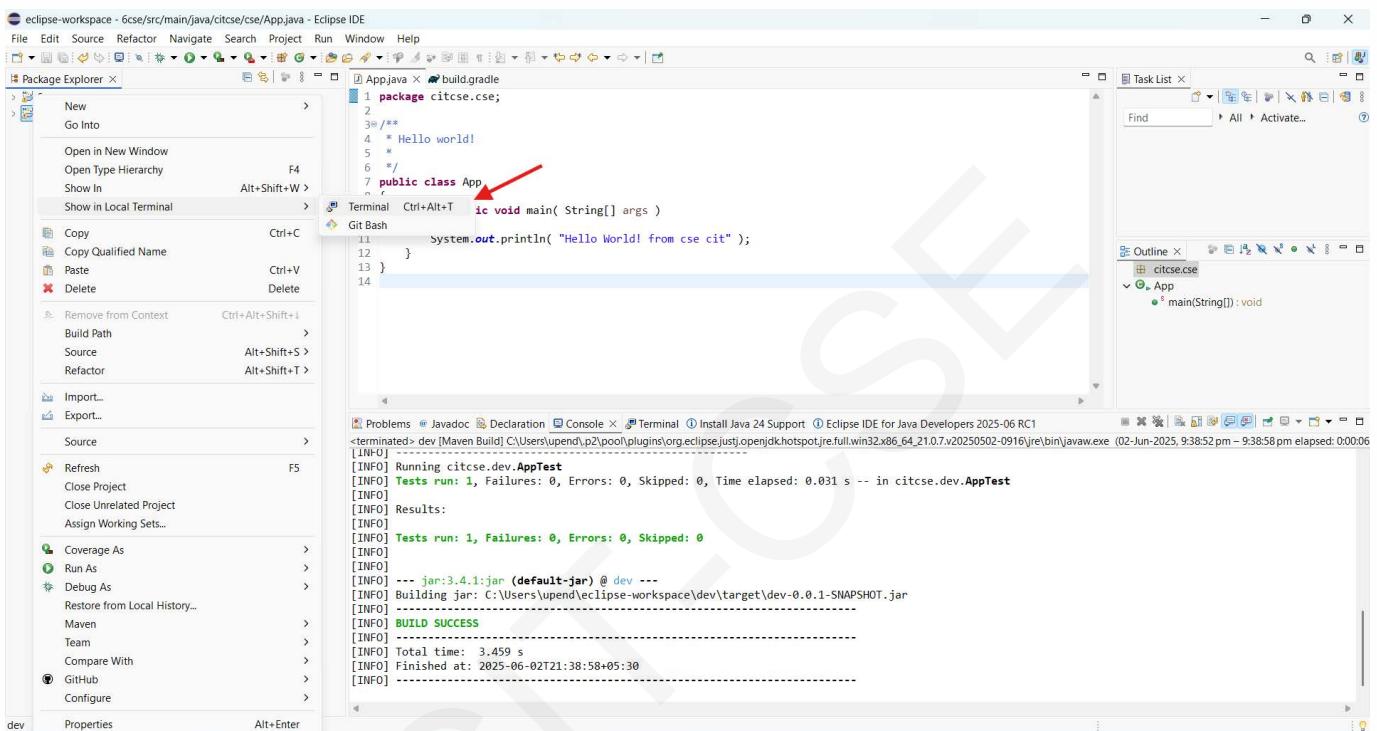
STEP1: First create a Maven Project as in PROGRAM2 then build the project and run java application you will get Hello World Message

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

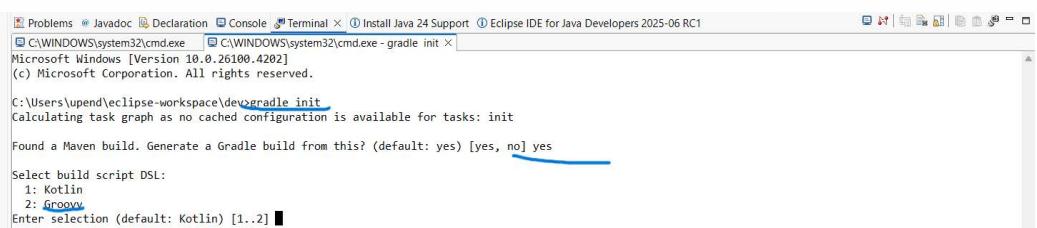
- In the terminal, type: **gradle init**

- 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).

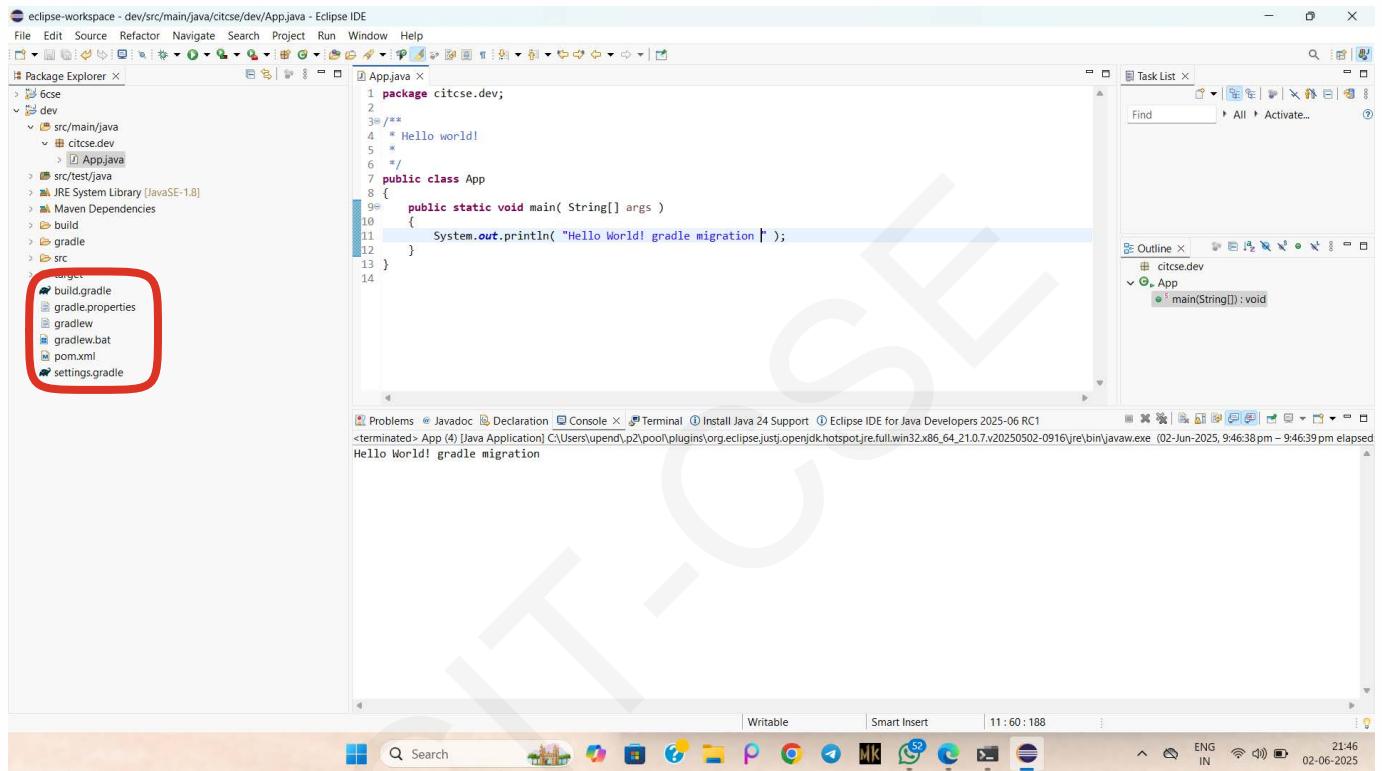


STEP 5: Run Gradle Build

- Run the command: **gradle build**
- Confirm 'BUILD SUCCESSFUL' in the console.

STEP 6: Modify App.java

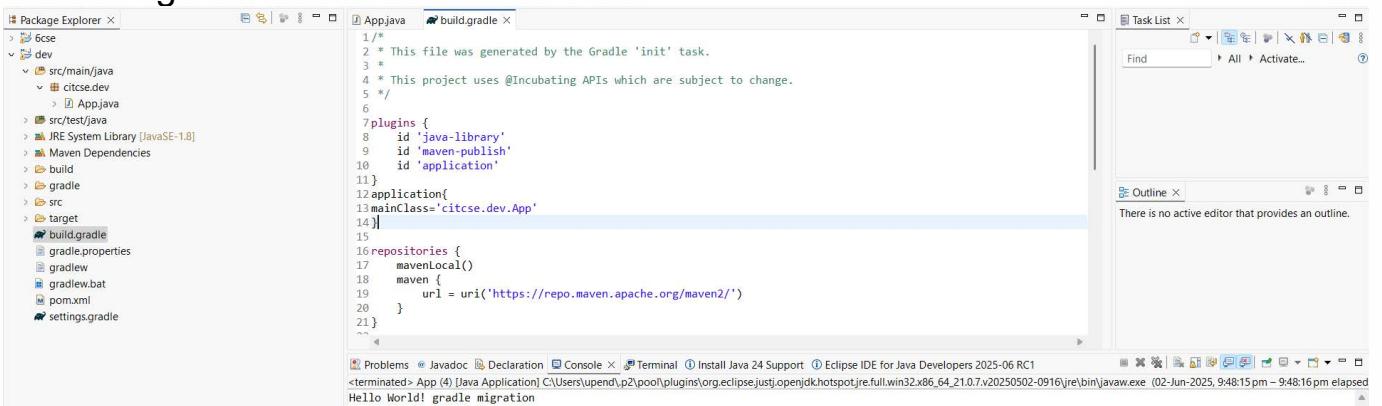
- Change the print statement in App.java to: "Hello World! gradle migration"
- Run the java file
- Gradle files like build.gradle and settings.gradle will be created on the left pane.



STEP 7: Update build.gradle

- Add main class under application block
application {
mainClass = '**Groupid.ArtifacId.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"

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "dev". It contains a package "citicse.dev" with a class "App.java". Other files visible include "build.gradle", "gradle.properties", "gradlew", "gradlew.bat", "pom.xml", and "settings.gradle".
- Build Output:** The bottom pane displays the terminal output of the Gradle build process:

```
[Incubating] Problems report is available at: file:///C:/Users/upend/eclipse-workspace/dev/build/reports/problems/problems-report.html
BUILD SUCCESSFUL in 2s
5 actionable tasks: 5 executed
Configuration cache entry stored.
C:\Users\upend\eclipse-workspace\dev>gradle run
Calculating task graph as no cached configuration is available for tasks: run

> Task :run
Hello World! gradle migration

BUILD SUCCESSFUL in 2s
2 actionable tasks: 1 executed, 1 up-to-date
Configuration cache entry stored.
C:\Users\upend\eclipse-workspace\dev>
```
- System Tray:** At the bottom right, the system tray shows icons for battery, signal strength, and date/time (21:49, 02-06-2025).

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.



A screenshot of the Jenkins download page. It shows a list of download options for Jenkins 2.504.2 LTS. The options include: Generic Java package (.war), Docker, Kubernetes, Ubuntu/Debian, Red Hat Enterprise Linux and derivatives, Fedora, Windows, openSUSE, and FreeBSD. Each option has a download link and a "SHA-256" hash value. A "Third party" link is visible at the bottom right.

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

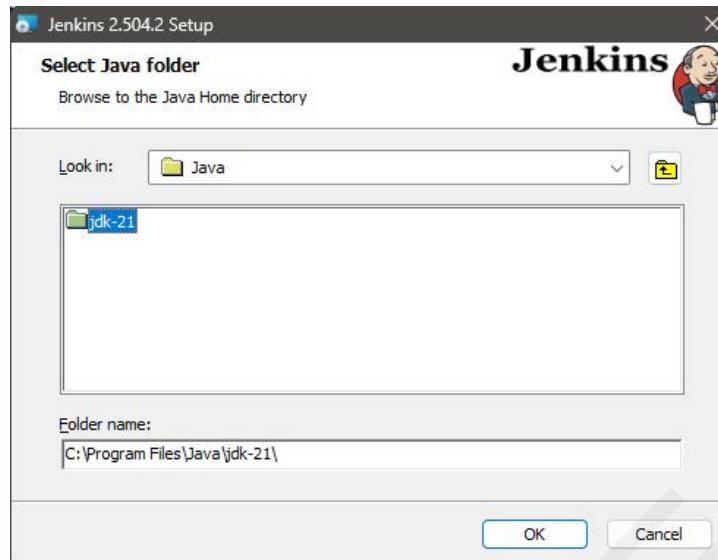
1. Launch Jenkins installer.

The screenshots show the Jenkins 2.504.2 Setup Wizard interface on a Windows system. The first screen is the 'Welcome to the Jenkins 2.504.2 Setup Wizard' with a cartoon Jenkins head icon. The second screen is 'Destination Folder' where the default path 'C:\Program Files\Jenkins\' is selected. The third screen is 'Service Logon Credentials' where the 'Run service as LocalSystem' option is chosen, and the account and password fields are empty. All screens include 'Back', 'Next', and 'Cancel' buttons.

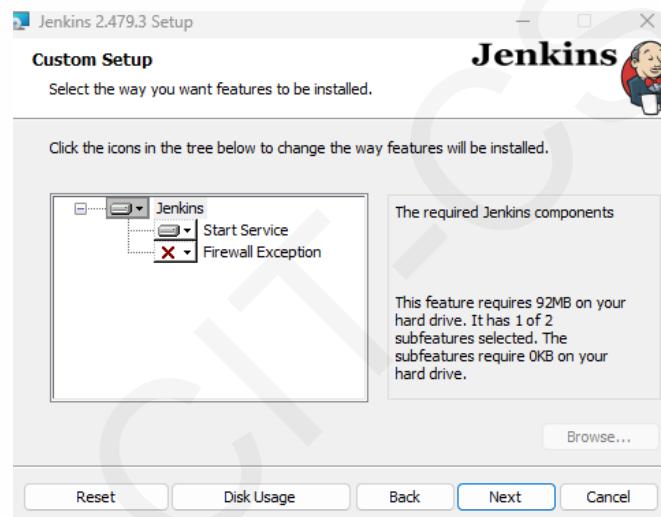
2. Accept default port (e.g., 8080).

The screenshot shows the 'Port Selection' step of the setup wizard. It prompts the user to choose a port for the Jenkins service. A note says 'Please choose a port.' Below is a 'Port Number (1-65535)' field containing '8080', a 'Test Port' button with a green checkmark, and a note at the bottom stating 'It is recommended that you accept the selected default port.' Navigation buttons 'Back', 'Next', and 'Cancel' are at the bottom.

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

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password



Let the installation proceed

Getting Started

The screenshot shows the Jenkins 'Getting Started' page with a sidebar on the left containing a list of installed and available plugins. The available plugins listed are:

- Folders
- Timestamper
- Pipeline
- Git
- LDAP
- OWASP Markup Formatter
- Workspace Cleanup
- Github Branch Source
- SSH Build Agents
- Email Extension
- Build Timeout
- Ant
- Pipeline: GitHub Groovy Libraries
- Matrix Authorization Strategy
- Mailer
- Credentials Binding
- Gradle
- Pipeline Graph View
- PAM Authentication
- Dark Theme

A dropdown menu is open over the 'Build Timeout' plugin, showing its API documentation. The menu items include:

- OWASP Markup Formatter
- ** SCM API
- ** JSON Path API
- ** Structure
- ** Pipeline: Step API
- ** Token Macro
- ** bouncycastle API
- ** Credentials
- ** Plain Credentials
- ** Variables
- ** File Credentials
- ** Credentials Binding
- ** SCM API
- ** Pipeline: API
- ** commons-lang3 v3.x Jenkins API

At the bottom of the page, there is a link to 'Skip and continue as admin'.

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

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.479.3

Skip and continue as admin (circled) Save and Continue

Jenkins is installed successfully

Getting Started

Jenkins is ready!

You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete.

[Start using Jenkins](#)

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.



Sign in to Jenkins

Username

Password

Keep me signed in

Step 2: Access Manage Jenkins

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

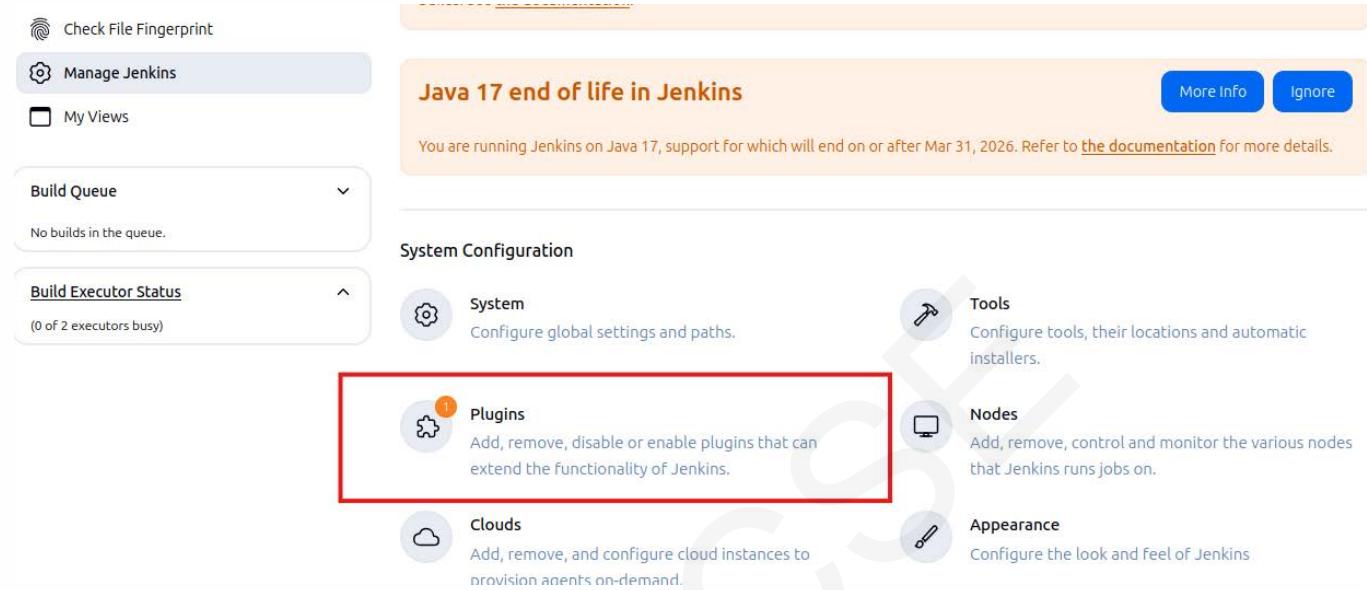
A screenshot of the Jenkins dashboard in a Mozilla Firefox browser. The title bar says "Dashboard - Jenkins — Mozilla Firefox". The address bar shows "localhost:8080". The main header has the Jenkins logo and the word "Jenkins". On the right, there are user icons for "admin" and "log out". Below the header, there's a "Dashboard >" link. A sidebar on the left includes links for "New Item", "Build History", "Project Relationship", "Check File Fingerprint", and "Manage Jenkins". The "Manage Jenkins" link is highlighted with a red box and a red arrow points to it from the bottom. The main content area shows a table with columns: S, W, Name (csecit), Last Success (1 day 21 hr #1), Last Failure (N/A), and Last Duration (13 sec). At the bottom, there are sections for "Build Queue" (No builds in the queue) and "Build Executor Status" (0 of 2 executors busy). The footer includes links for "REST API" and "Jenkins 2.504.2", along with system status icons.

Step 3: Install Maven Integration Plugin

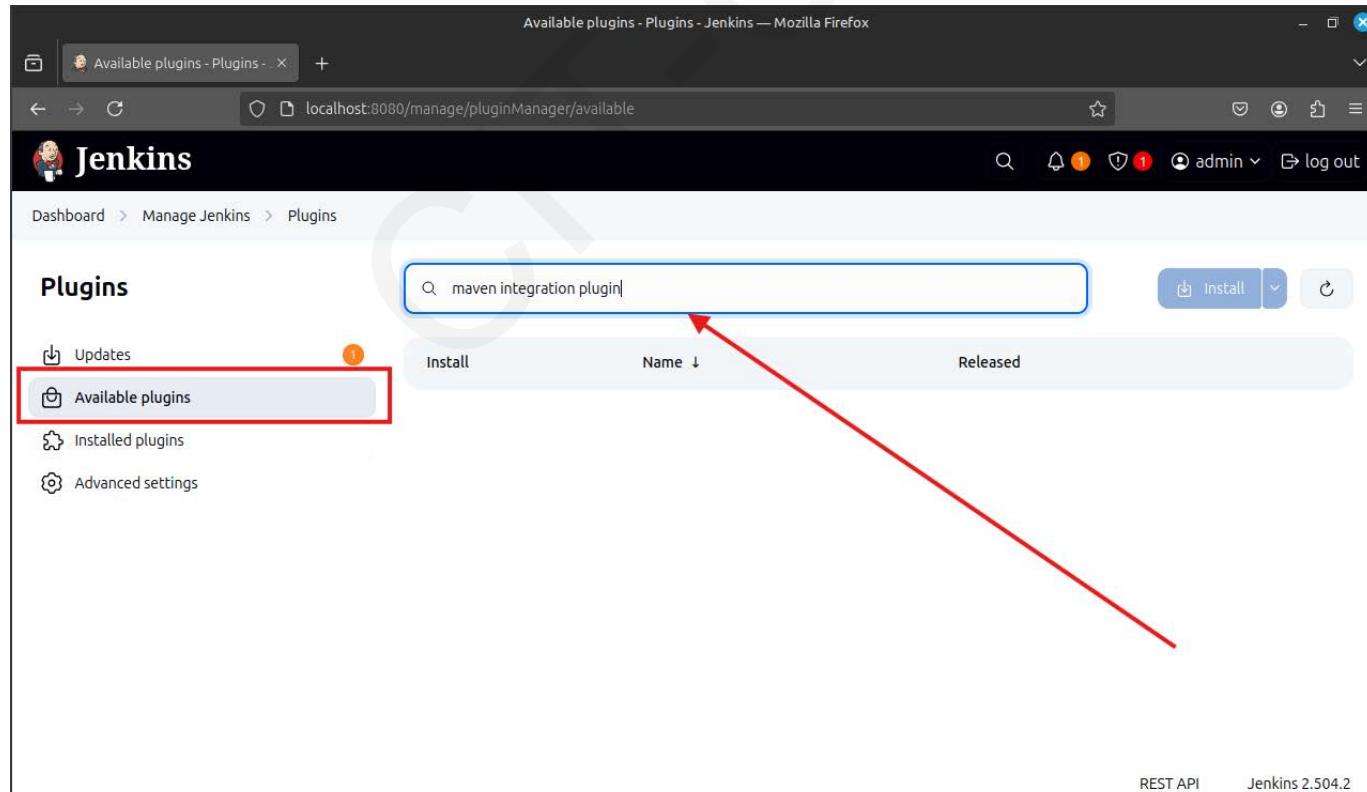
Navigate to:

Manage Jenkins > Plugins > Available Plugins

Search for 'Maven Integration Plugin', select it, and install it without restarting Jenkins.



The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'Check File Fingerprint', 'Manage Jenkins' (which is selected and highlighted in grey), and 'My Views'. Below that are 'Build Queue' (no builds in the queue) and 'Build Executor Status' (0 of 2 executors busy). The main content area has a header 'Java 17 end of life in Jenkins' with 'More info' and 'Ignore' buttons. A large orange banner below it states: 'You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details.' Under 'System Configuration', there are several sections: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Appearance' (Configure the look and feel of Jenkins), and 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins). The 'Plugins' section is highlighted with a red box. Below it are 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand) and 'Advanced Settings' (Configure advanced Jenkins settings).



The screenshot shows the Jenkins Available plugins page. The URL is 'localhost:8080/manage/pluginManager/available'. The top navigation bar includes 'Dashboard', 'Manage Jenkins', and 'Plugins'. The main content area has a title 'Plugins' and a sidebar with 'Updates' (1 update available), 'Available plugins' (selected and highlighted with a red box), 'Installed plugins', and 'Advanced settings'. A search bar contains the text 'maven integration plugin'. An 'Install' button is visible. A red arrow points from the text 'Tick on the Maven Integration Plugin and click on "Install", and let the installation process continue.' to the 'Install' button.

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

The screenshot shows the Jenkins 'New Item' creation interface. At the top, the title bar says 'New Item - Jenkins — Mozilla Firefox'. Below it, the URL is 'localhost:8080/view/all/newJob'. The main area is titled 'New Item' and has a sub-section 'Enter an item name' with the value 'CSECIT_prog6'. Under 'Select an item type', there are three options: 'Freestyle project' (selected), 'Maven project' (highlighted with a red border), and 'Pipeline'. A blue 'OK' button is at the bottom.

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

This screenshot shows the same Jenkins 'New Item' creation interface as the previous one, but with a different selection. The 'Freestyle project' option is now highlighted with a red border. The other two options ('Maven project' and 'Pipeline') are visible below it. The 'OK' button is at the bottom.

Click on " Ok "

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

The screenshot shows the Jenkins 'Configure' screen for a project named 'CITCSE'. In the left sidebar, 'Build Steps' is selected. Under the 'Environment' heading, there is a list of build steps: 'Delete workspace before build starts', 'Use secret text(s) or file(s)', 'Execute Windows batch command', 'Execute shell', 'Invoke Ant', 'Invoke Gradle script', and 'Invoke top-level Maven targets'. The 'Invoke top-level Maven targets' option is highlighted with a red box. Below this list is a note: 'like code compilation, testing, and deployment.' At the bottom of the environment section are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins 'Configure' screen for the 'CITCSE' project. In the left sidebar, 'Build Steps' is selected. Under the 'Build Steps' heading, there is a section titled 'Invoke top-level Maven targets' with a dashed border. Inside this section, the 'Maven Version' dropdown is set to '(Default)' and the 'Goals' dropdown contains the value 'clean install'. At the bottom of this section is an 'Advanced' button. Below the main configuration area is an 'Add build step' button.

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**

The screenshot shows the Jenkins configuration interface for a job named 'CITCSE'. On the left, there's a sidebar with options: General, Source Code Management, Triggers, Environment, Build Steps (which is selected), and Post-build Actions. The main panel has sections for Goals (set to 'clean install'), Advanced (status: Edited), POM (path: C:\Users\CSE_CIT\Desktop\Prog6\pom.xml), Properties (empty), and JVM Options (empty). At the bottom are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins item dashboard for 'csecit'. The left sidebar includes links for Status, Changes, Console Output (which is selected), Edit Build Information, Delete build '#2', Timings, Git Build Data, and Previous Build. The right panel displays the 'Console Output' tab, which shows the build log for build #2. The log output is as follows:

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/csecit
The recommended git tool is: NONE
using credential dc095bed-68ab-4836-ab7b-9bd34fda91b2
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/csecit/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/rohittsinghh/jenkinsdemo.git # timeout=10
Fetching upstream changes from https://github.com/rohittsinghh/jenkinsdemo.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/rohittsinghh/jenkinsdemo.git +refs/
heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision e10b984284f915feb073134ece5c4488628fc009 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
```

PROGRAM7: Configuration Management with Ansible: Basics of Ansible:Inventory,Playbooks, and Modules, Automating Server Configurations with Playbooks, 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:~$ 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:~$
```

```
cse-cit@csecit-VirtualBox:~  
GNU nano 7.2  
[local]  
localhost ansible_connection=local  
  
[ Read 2 lines ]  
^G Help ^O Write Out ^W Where Is ^K Cut ^U Paste ^T Execute ^C Location M-U Undo  
^X Exit ^R Read File ^A Replace ^J Justify ^V Go To Line M-E Redo M-A Set Mark M-6 Copy M-] To Bracket  
^Q Where Was
```

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!"
```

The screenshot shows a terminal window titled 'cse-cit@csecit-VirtualBox: ~'. Inside the terminal, the file 'playbook.yml' is being edited in the 'GNU nano 7.2' editor. The content of the file is the YAML code provided in the previous text block. The terminal window has a dark background and light-colored text. At the bottom, there is a menu bar with various keyboard shortcuts for file operations like Help, Exit, Write Out, Read File, etc.

```
GNU nano 7.2          playbook.yml
...
- name: Basic Hello World Playbook
  hosts: all
  tasks:
    - name: Print Hello World
      debug:
        msg: "Hello, world!"
```

[Read 8 lines] [Execute] [Location] [Undo] [Set Mark] [To Bracket]
[Read 8 lines] [Execute] [Location] [Undo] [Set Mark] [To Bracket]
[Read 8 lines] [Execute] [Location] [Undo] [Set Mark] [To Bracket]
[Read 8 lines] [Execute] [Location] [Undo] [Set Mark] [To Bracket]
[Read 8 lines] [Execute] [Location] [Undo] [Set Mark] [To Bracket]
[Read 8 lines] [Execute] [Location] [Undo] [Set Mark] [To Bracket]
[Read 8 lines] [Execute] [Location] [Undo] [Set Mark] [To Bracket]
[Read 8 lines] [Execute] [Location] [Undo] [Set Mark] [To Bracket]

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
```

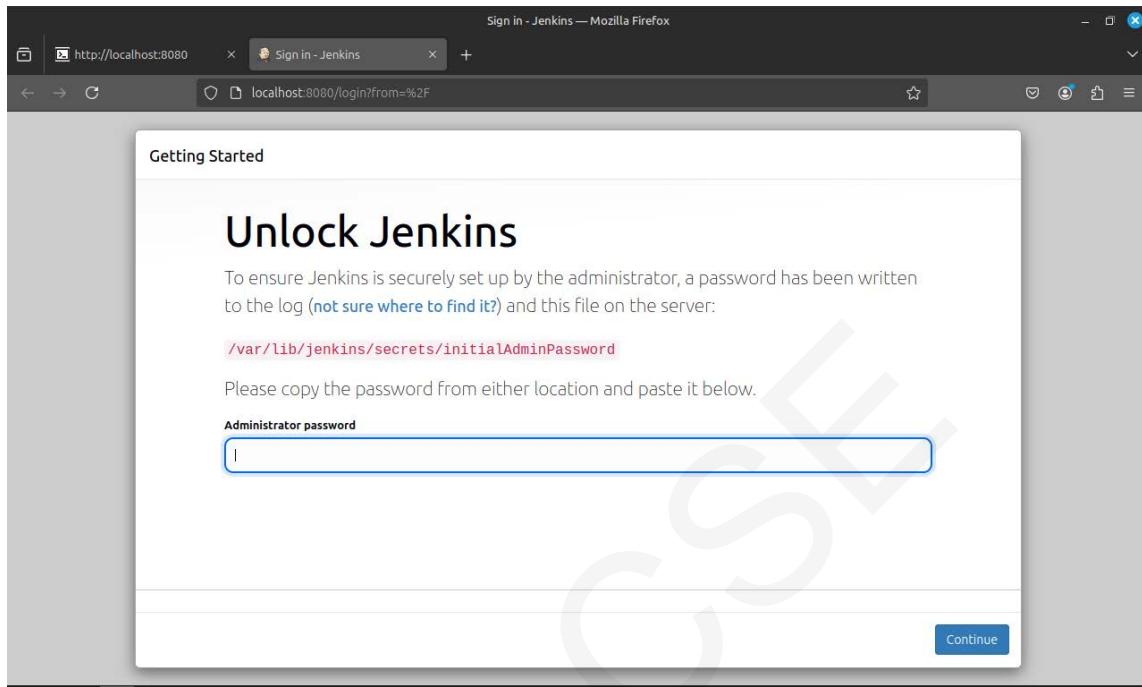
```
cse-cit@csecit-VirtualBox:~$ sudo apt update
sudo apt install jenkins -y
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:2 https://pkg.jenkins.io/debian-stable binary/ Release [2,044 B]
Get:3 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Get:4 https://pkg.jenkins.io/debian-stable 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

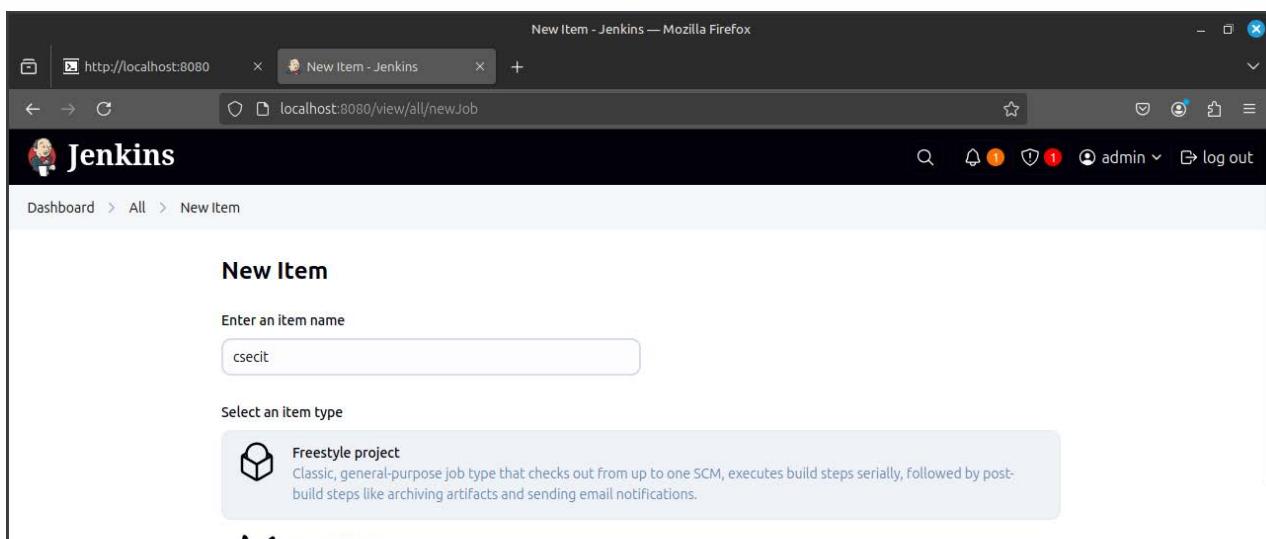
Generate a sample Maven project:

```
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] -----
[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] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 1.158 s
[INFO] Finished at: 2025-06-04T17:59:46+05:30
[INFO]
[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

The screenshot shows the Jenkins job configuration page for 'csecit'. The 'Source Code Management' section is selected. It displays the 'Repository URL' as 'https://github.com/rohitsinghh/jenkinsdemo.git' and the 'Credentials' as 'rohitsinghh/*****'. The 'Branches to build' section shows the 'Branch Specifier' as '*/main'. The 'Save' and 'Apply' buttons are visible at the bottom.

Step 8: Add Maven Build Step

In 'Build Steps', select 'Invoke top-level Maven targets'.

Set Goals to:

clean package

The screenshot shows the Jenkins job configuration page for 'csecit'. The 'Build Steps' section is selected. A red box highlights the 'Goals' input field, which contains 'clean package'. Other options shown include 'Terminate a build if it's stuck' and 'With Ant'.

Step 9: Build the Project

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

The screenshot shows the Jenkins project 'csecit' dashboard. The left sidebar contains links for Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main area shows a green checkmark icon next to 'csecit'. Below it is a 'Permalinks' section with a bulleted list of build history: Last build (#1), 2 min 45 sec ago; Last stable build (#1), 2 min 45 sec ago; Last successful build (#1), 2 min 45 sec ago; and Last completed build (#1), 2 min 45 sec ago. A 'Builds' section displays a table with one row, showing a green checkmark next to '#1 6:18 PM'. The bottom status bar shows icons for network, file, and system status, along with the time 18:21.

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

```
cse-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

```
cse-cit@csecit-VirtualBox: ~
GNU nano 7.2
- 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
```

The terminal window shows the Ansible playbook file 'deploy-artifact.yml' in the nano editor. The file contains the YAML code for deploying a JAR file from Jenkins. The terminal window also displays various keyboard shortcuts for the nano editor.

Playbook Execution Output

Command Executed:

```
ansible-playbook deploy-artifact.yml
```

```
cse-cit@csecit-VirtualBox:~$ ansible-playbook deploy-artifact.yml
PLAY [Deploy artifact built by Jenkins] ****
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 [Copy JAR to deployment folder] ****
ok: [localhost]

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

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
```

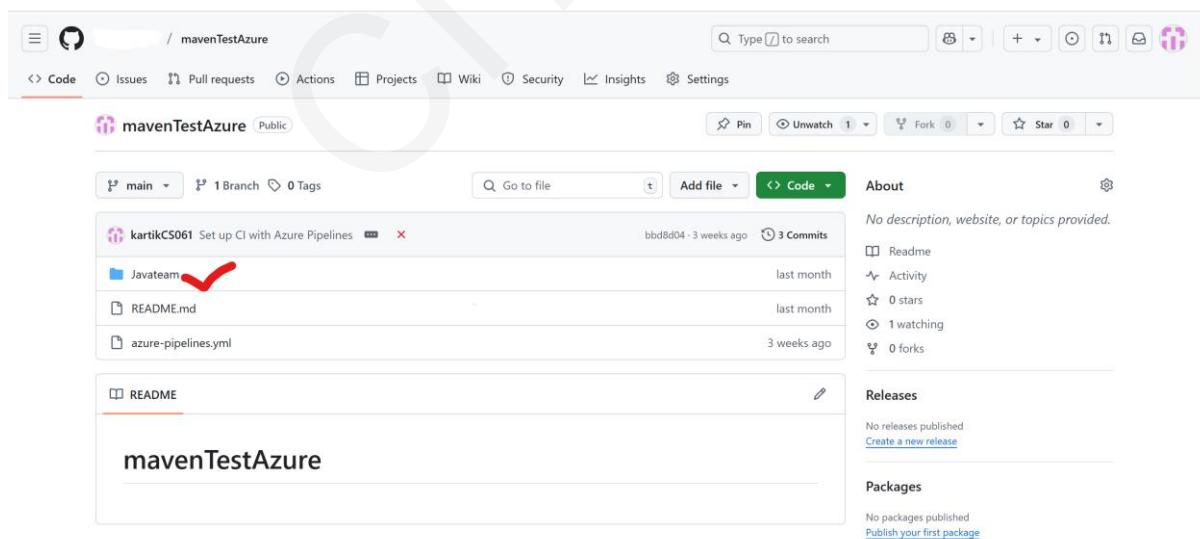
```
git add .
```

```
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.

Google search results for "azure for students":

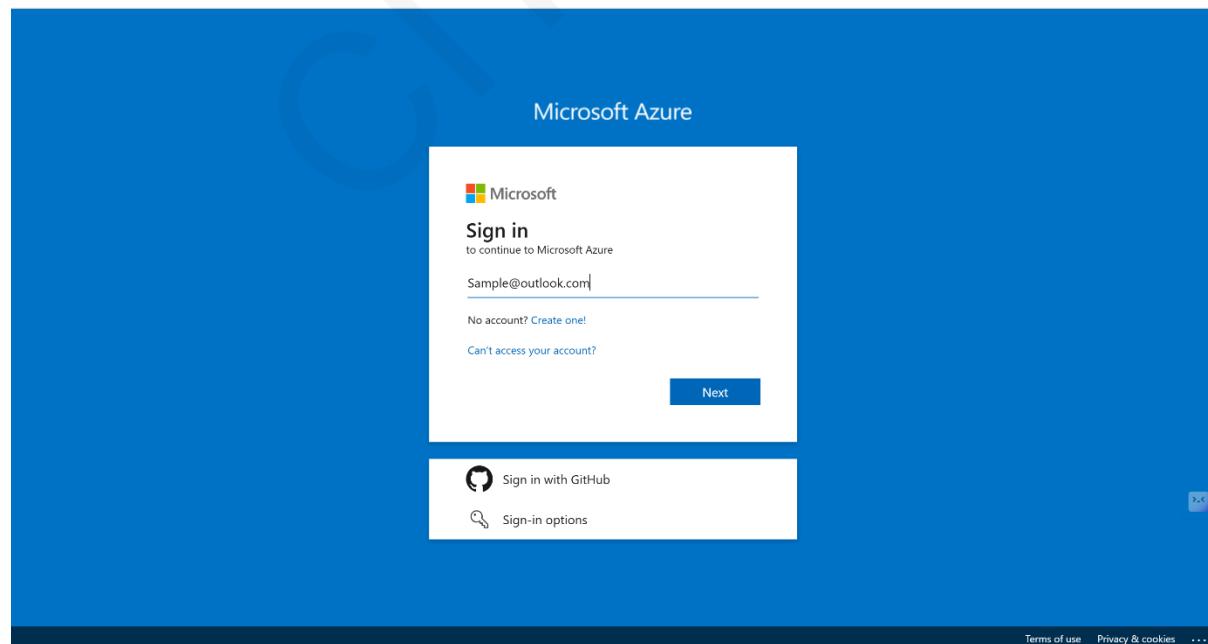
- Azure for Students – Free Account Credit**
With Microsoft Azure for Students, get a \$100 credit when you create your free account. There is no credit card needed and 12 months of free Azure services.
- Azure for College Students—Offer Details**
Students, get Azure for free courtesy of Microsoft Azure. College students enrolled full time are eligible for Azure free account with \$100 in credits.
- Microsoft Azure for Students Starter Offer**
Review the student offer for the Microsoft Azure. With Azure for Students Starter, get access to Azure services at no cost, commitment, or time limit.

Search within: Google

Note: AI responses might not always be perfect. It's a good idea to double-check the information.

Open Merlin Live Search PRO

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

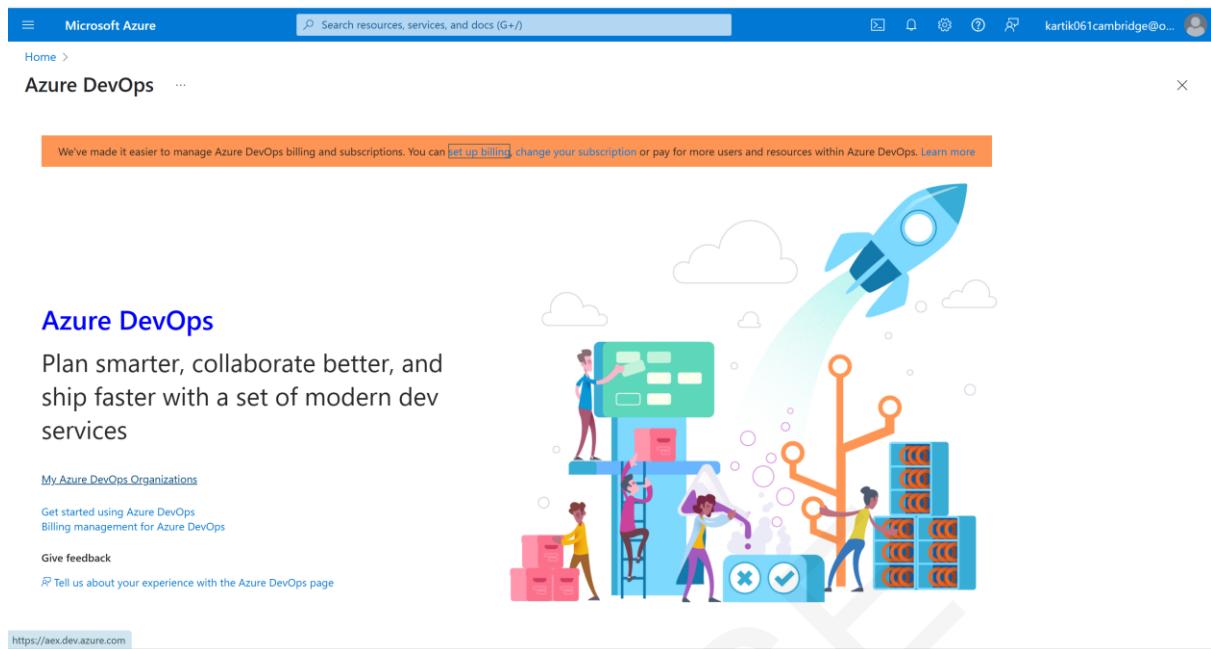


STEP 3: Go to the search bar and type: Azure Devops organization

The screenshot shows the Microsoft Azure homepage. At the top, there is a search bar with the placeholder "Search resources, services, and docs (G+/-)". Below the search bar, the "Welcome to Azure!" section is visible, featuring three cards: "Start with an Azure free trial", "Manage Microsoft Entra ID", and "Azure for Students". Under "Azure services", there are icons for "Create a resource", "Azure DevOps organizations", "Quickstart Center", "Azure AI Foundry", "Kubernetes services", "Virtual machines", "App Services", "Storage accounts", "SQL databases", and "More services". A large "Resources" section follows, containing links for "Recent" and "Enqueue".

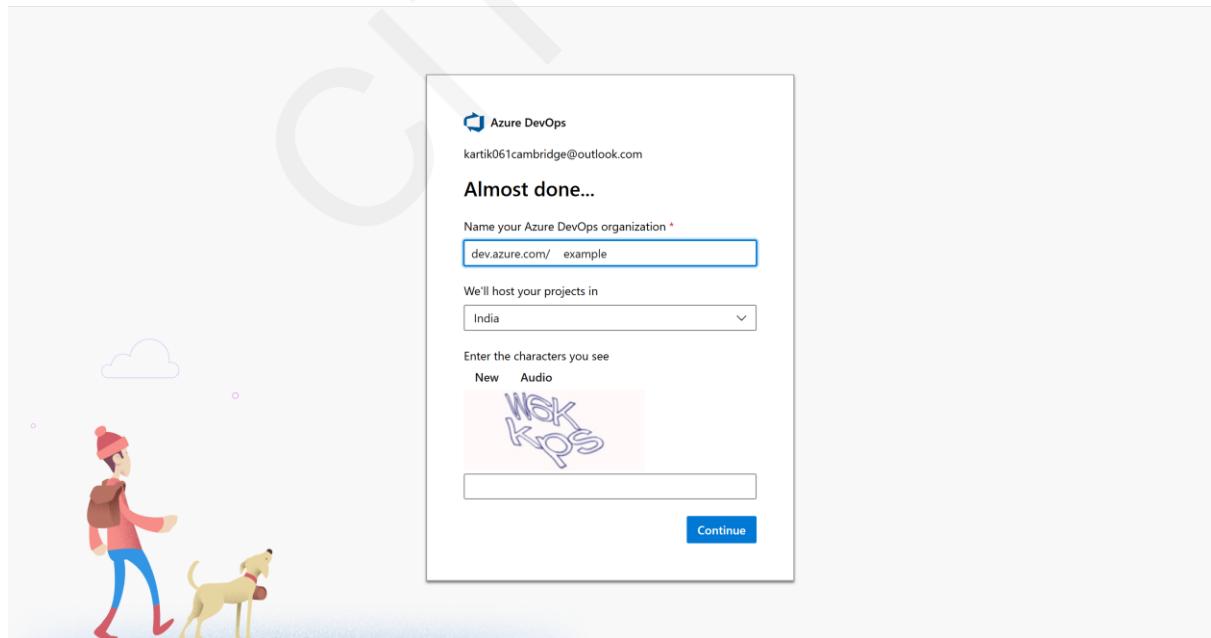
This screenshot is identical to the one above, but the search bar now contains the text "Azure Devops Organization". The search results are displayed below the search bar, showing "All" selected, "Services (99+)" under "Services", and "Resources" under "Documentation". The "Azure for Students" card is also visible.

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 appear.



- 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**

Windows Server 2019 support as Microsoft Hosted Images is being deprecated. Customers using Windows Server 2019 images in their pipelines should switch to Windows Server 2022 by 30th June 2025 to avoid disruption. For more information like, the Brownout Schedule and how to identify the impacted pipelines, please follow the [blog](#).

Create a project to get started

Project name *

Sample_Cambridge_project

Visibility

Public

Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private

Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).

+ Create project

STEP 9: Enable public project.

Security policies



STEP 10: Come back to the project page and give the **project name and select **Public**, then click on the “+ Create project”.**

The screenshot shows the 'Create a project to get started' interface. On the left, there is a cartoon illustration of a person sitting on the floor, working on a laptop, with a dog sitting next to them. The main form area has the following fields:

- Project name ***: A text input field containing "SAMPLE_CAMBRIDGE_DEVOPS_PROJECT".
- Visibility**: A section with two options:
 - Public** (selected): "Anyone on the internet can view the project. Certain features like TFVC are not supported."
 - Private**: "Only people you give access to will be able to view this project."
- By creating this project, you agree to the Azure DevOps [code of conduct](#)**: A note at the bottom of the form.
- + Create project**: A blue button at the bottom right of the form.

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

The screenshot shows the Azure DevOps interface for the project "SAMPLE_CAMBRIDGE_DEVOPS_PROJECT". The left sidebar includes links for Overview, Summary, Dashboards, Wiki, Boards, Repos, Pipelines, Test Plans, and Artifacts. The main area features a central dashboard with a welcome message "Welcome to the project!" and a "Project stats" section indicating no available data. The "Pipelines" tab is highlighted in the navigation bar at the bottom.

STEP 2: Click Create Pipeline

The screenshot shows the Azure DevOps interface for the "Pipelines" section of the project. The left sidebar has the "Pipelines" tab selected. The main area displays a "Create your first Pipeline" message with a sub-instruction about automating build and release processes. A prominent blue "Create Pipeline" button is located at the bottom of the central content area.

STEP 3: Select GitHub YAML

The screenshot shows the 'Select' step of creating a new pipeline in Azure DevOps. The left sidebar lists project navigation options like Overview, Boards, Repos, Pipelines, Environments, Library, Test Plans, and Artifacts. The main area is titled 'Where is your code?' and lists four options: 'Azure Repos Git - YAML' (selected), 'Bitbucket Cloud - YAML' (Hosted by Atlassian), 'GitHub - YAML' (Home to the world's largest community of developers), and 'GitHub Enterprise Server - YAML' (The self-hosted version of GitHub Enterprise). A large watermark 'MICROSOFT' is diagonally across the page.

STEP 4: Select the correct repository

The screenshot shows the 'Select' step of creating a new pipeline in Azure DevOps. The left sidebar lists project navigation options. The main area is titled 'Select a repository' and shows a list of repositories under 'My repositories'. One repository, 'mavenTestAzure', is selected, indicated by a blue checkmark next to its name. A note at the bottom says: 'Showing the most recently used repositories where you are a collaborator. If you can't find a repository, make sure you [provide access](#). You may also select a specific connection.' A large watermark 'MICROSOFT' is diagonally across the page.

STEP 5: Click on RUN

The screenshot shows the 'Review your pipeline YAML' step in the Azure DevOps interface. On the left, there's a sidebar with project navigation: Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Library, Test Plans, and Artifacts. Below the sidebar, 'Project settings' and a back arrow are visible. The main area displays the pipeline YAML code:

```
1 # Maven
2 # Build your Java project and run tests with Apache Maven.
3 # Add steps that analyze code, save build artifacts, deploy, and more:
4 # https://docs.microsoft.com/azure/devops/pipelines/languages/java
5
6 trigger:
7 - main
8
9 pool:
10 - vmImage: ubuntu-latest
11
12 steps:
13 - task: Maven@3
14   inputs:
15     mavenPomFile: 'pom.xml'
16     mavenOptions: '-Xmx3072m'
17     javaHomeOption: 'JDKVersion'
18     jdkVersionOption: '1.11'
19     jdkArchitectureOption: 'x64'
20     publishJUnitResults: true
21     testResultsFiles: '**/surefire-reports/TEST-*.xml'
22     goals: 'package'
23
```

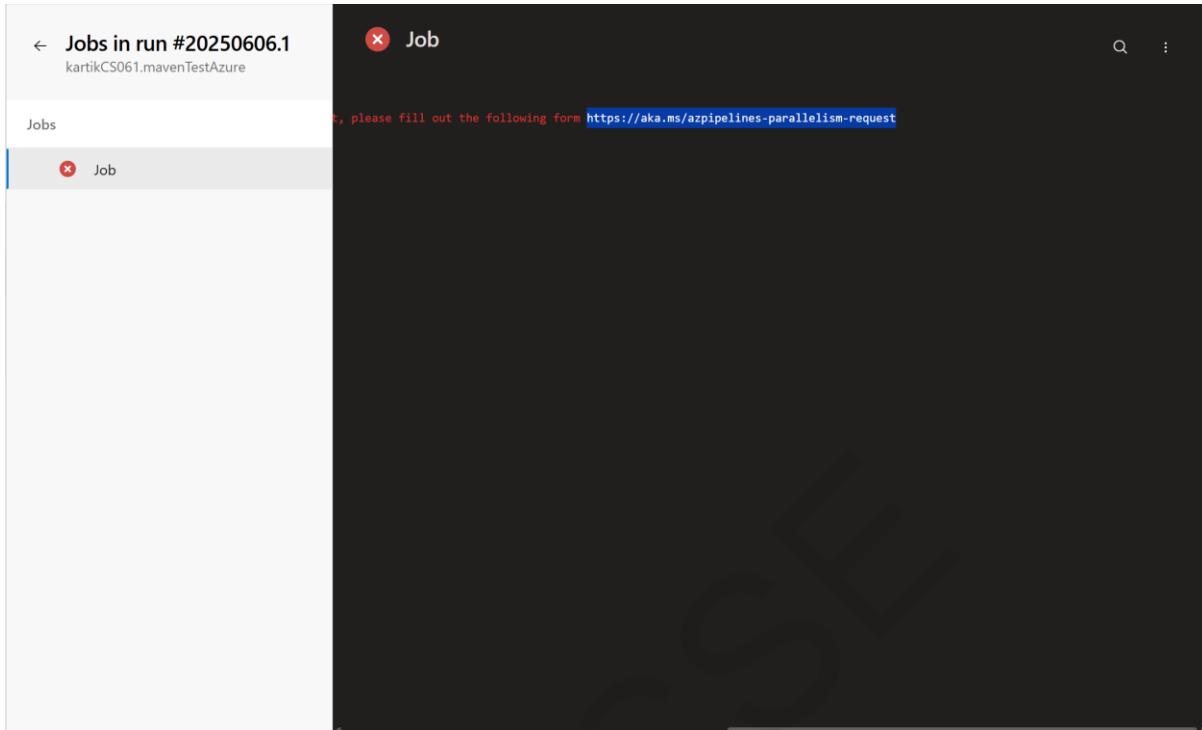
At the top right, there are 'Variables' and a large blue 'Run' button with a checkmark icon. A 'Show assistant' link is also present.

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

The screenshot shows the 'Job' details screen in the Azure DevOps interface. The left sidebar is identical to the previous screenshot. The main area shows a job named 'Job' under 'Jobs'. The job status is 'Running' with a red 'X' icon. The job details pane shows the following log output:

```
1 ##[error]No hosted parallelism has been purchased or granted. To request a free parallelism grant, please fill out the form at https://aka.ms/parallelism
2 Pool: Azure Pipelines
3 Image: ubuntu-latest
4 Started: Just now
5 Duration: 39s
6
7 ► Job preparation parameters
```

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.

Azure DevOps Parallelism Request

This form is for users to request increased parallelism in Azure DevOps.

Please consider that it could take 4-5 business days to process the request. We are working on improving this process at the moment. Sorry for the inconvenience.

When you submit this form, it will not automatically collect your details like name and email address unless you provide it yourself.

* Required

1. What is your name? *

Enter your answer

2. What is your email address? *

Enter your answer

3. What is the name of your Azure DevOps Organization? *

(E.g. for <https://myorganization.visualstudio.com> or <https://dev.azure.com/myorganization> link formats - organization name would be 'myorganization')

Enter your answer

4. Are you requesting a parallelism increase for Public or Private projects? *

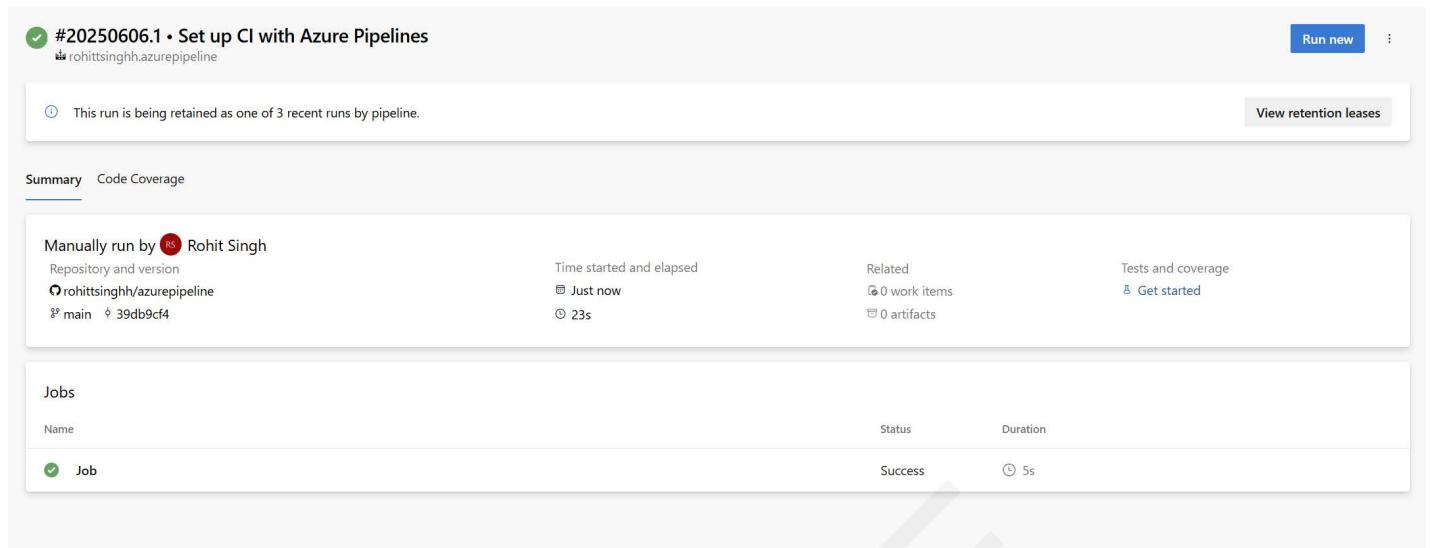
Private

Public

Submit

Never give out your password. [Report abuse](#)

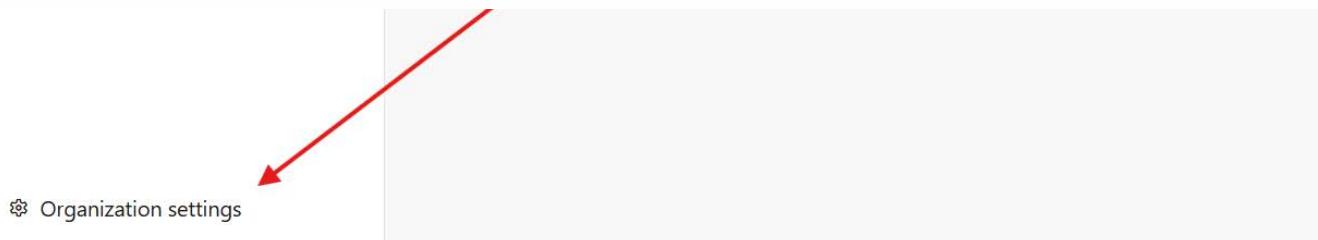
After 48 hours your application will be accepted and now you can Run the Pipeline again , this time it will run successfully



The screenshot shows the Azure Pipelines interface for a pipeline run. At the top, there's a header with a green checkmark icon, the run ID "#20250606.1", the pipeline name "Set up CI with Azure Pipelines", and a "rohittsinghh.azurepipeline" repository icon. To the right are buttons for "Run new" and more options. Below the header, a message says "This run is being retained as one of 3 recent runs by pipeline." with a "View retention leases" link. There are two tabs: "Summary" (which is selected) and "Code Coverage". The "Summary" section contains details about the run: "Manually run by Rohit Singh" (with a profile picture), "Repository and version" (rohittsinghh.azurepipeline, main branch, commit 39db9cf4), "Time started and elapsed" (Just now, 23s), "Related" (0 work items, 0 artifacts), and "Tests and coverage" (Get started). The "Jobs" section lists a single job named "Job" which has completed successfully in 5 seconds. A large watermark "CT-CSE" is diagonally across the page.

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 "

This screenshot shows the 'Pipelines Settings' page. The left sidebar has 'Pipelines' selected. The main area contains several cards: 'Pipelines Settings' (highlighted with a red box), 'Usage', 'Deployment Pools', 'Parallel Jobs', 'OAuth Configuration', 'Billing', 'Agent Pools', 'Permissions', 'Pipelines', 'Release Management', 'XAML Build Services', 'Service Connections', 'Parallel Jobs', 'Test Management', 'Agent Pools', and 'Release Retention'. A 'Project setting' dropdown is set to 'devops'.

Make sure these are turned off :

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

This screenshot shows the 'Disable creation of classic build pipelines' settings. It features three toggle switches: 'Disable stage chooser' (Off), 'Disable creation of classic build pipelines' (Off, circled in red), and 'Disable creation of classic release pipelines' (Off, circled in red). Below these are sections for 'Triggers' (On) and 'Limit building pull requests from forked GitHub repositories' (Configure how to build pull requests from forked repositories, Learn more).

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

The screenshot shows the Azure DevOps interface for a project named 'devops'. On the left sidebar, under the 'Pipelines' category, the 'Releases' option is highlighted with a red box. The main area is titled 'New release pipeline' and shows a single release named 'Release-1' with one stage named 'Stage 2'. Below this, another release is shown: 'Release-2' (39db9cf43) from 6/6/2025 at 10:27:17 PM, also with 'Stage 2'.

Now after this you will get the following output

The screenshot shows the 'Release' details page for 'Release-2'. The top navigation bar includes 'Pipeline', 'Variables', 'History', 'Deploy', 'Cancel', 'Refresh', 'Edit', and more. The 'Pipeline' tab is selected. The 'Release' section shows it was 'Manually triggered by Rohit Singh' on 6/6/2025 at 10:27 PM. The 'Stages' section shows 'Stage 2' has succeeded with 2 warnings. An artifact named '_rohitssinghh_azurepip...' (39db9cf43) is listed under Artifacts.