

Dr. T.THIMMIAH INSTITUTE OF TECHNOLOGY
(Estd.1986)Oorgaum,KolarGoldFields,Karnataka–563120
(Affiliated to Visvesvaraya Technological University)
NAAC Accredited 'A' Grade, NBA Accredited for CSE, ECE & Mining Engg Programs



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VI Semester

DevOps (BCSL657D)

LAB MANUAL

Name of the Student _____

RegNo. Of the Student _____



Dr. T. THIMMIAH INSTITUTE OF TECHNOLOGY

(Estd. 1986) Oorgaum, Kolar Gold Fields, Karnataka – 563120

(Affiliated to VTU, Belgaum, Approved by AICTE - New Delhi)

NAAC Accredited 'A' Grade, NBA Accredited for CSE, ECE & Mining Engg Programs

Department of Computer Science and Engineering

DEVOPS (BCSL657D)

Lab Manual

Prepared by:

1. Prof. Mercy Flora
2. Prof. Sophia S

Name of the Student: _____

Reg No of the Student: _____



Dr. T. THIMMIAH INSTITUTE OF TECHNOLOGY

(Estd. 1986) Oorgaum, Kolar Gold Fields, Karnataka – 563120

(Affiliated to VTU, Belgaum, Approved by AICTE – New Delhi)

NAAC Accredited 'A' Grade, NBA Accredited for CSE, ECE & Mining Engg programs.

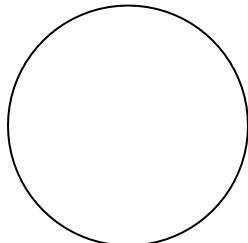
Laboratory Certificate

This is to certify thatbearing USN Has satisfactorily completed the experiments in Practical **DevOps** prescribed by the Visvesvaraya Technological University for the course with course code **BCSL657D** in the Laboratory of this college in the year 2024-25

Signature of the faculty in-charge

Head of the Department

Final Marks obtained.



College Vision and Mission

College-Vision:

To produce technically competent engineers having innovative skills, positive attitude, leadership and professional ethics, with a focus on developing sustainable and new technology

College- Mission:

- Create a conducive environment for teaching, learning and innovation by providing state of the art infrastructure.
- Learn sustainable and new technologies through industry institute collaborations.
- Produce technically competent engineers with ethics, discipline and social consciousness through holistic education.

Department of Computer Science and Engineering

Vision and Mission

Dept.-Vision

To produce highly competent and innovative Computer Science professionals through excellence in teaching, training and research.

Dept.-Mission

- To provide appropriate infrastructure to impart need-based technical education through effective teaching and research.
- To involve the students in innovative projects on emerging technologies to fulfill the industrial requirements.
- To render leadership skills and ethical responsibilities in students that leads them to become globally competent professionals.

BE:Program Outcomes (POs)

Attend of the B.E program, students are expected to have developed the following outcomes.

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis andinterpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

COURSE OBJECTIVES AND OUTCOMES

Course Objectives:

This course will enable students to:

CLO1	To introduce DevOps terminology, definition & concepts
CLO2	To understand the different Version control tools like Git, Mercurial
CLO3	To understand the concepts of Continuous Integration/ Continuous Testing/ Continuous Deployment)
CLO4	To understand Configuration management using Ansible
CLO5	Illustrate the benefits and drive the adoption of cloud-based Devops tools to solve real world problems

Course Outcomes:

At the end of the course the student will be able to:

CO1	Demonstrate Build Automation and Continuous Integration
CO2	Demonstrate Configuration Management and Deployment
CO3	Design and Demonstrate DevOps Pipelines

Conduction of Practical Examination:

Assessment Details (both CIE and SEE)

- The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%.
- The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50).
- The minimum passing mark for the SEE is 35% of the maximum marks (18 marks out of 50).
- A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination (SEE), and a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are 50 Marks. The split-up of CIE marks for record/ journal and test are in the ratio 60:40.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up..
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The marks scored shall be scaled down to 20 marks (40% of the maximum marks). The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

Rubrics for Lab Performance Assessment

In Each Lab, students will be assessed on their participation and performance in the lab. The marks obtained in each lab will be totaled and combined with Final Lab test or Viva-voce. The students need to attend 85% of the labs to qualify to be eligible to attend the university examination.

Lab Participation Rubric(Conduction)

Proficient(15-13)	Good(12-10)	Adequate(9-7)	Substandard(6-4)	Unacceptable(3-0)
Lab performance is excellent. The student has attended all labs. Student demonstrates an accurate understanding of the lab objectives and concepts. The student can correctly answer questions and if appropriate, can explain concepts to fellow classmates. Student is eager to participate and assists when needed.	Lab performance is good. At most, the student has one lab absence. Student demonstrates a fair understanding of the lab objectives and concepts. The student can answer questions.	Lab performance is adequate. The student is absent more than 2 classes. Student arrives on time to lab, but may be unprepared. Answers to questions are basic and superficial suggesting that concepts are not fully grasped.	Lab performance is fair. The student may have been absent 3-4 times. Student unpreparedness makes it impossible to fully participate. If able to participate, student has difficulty explaining key lab concepts.	Lab performance is barely adequate. The student is irregular. Student was late to lab or did not participate. There was no attempt to make prior arrangements to make up the lab.

Dr. T.THIMMIAHINSTITUTEOFTECHNOLOGY

Department of Computer Science and Engineering.

DEVOPS - BCSL657D Contents

Exp. No	Title of the Experiment	Page No
1.	Introduction to Maven and Gradle: Overview of Build Automation Tools, Key Differences Between Maven and Gradle, Installation and Setup	1-4
2.	Working with Maven: Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins	5-10
3.	Working with Gradle: Setting Up a Gradle Project, Understanding Build Scripts (Groovy and Kotlin DSL), Dependency Management and Task Automation	11-13
4.	Practical Exercise: Build and Run a Java Application with Maven, Migrate the Same Application to Gradle	14-17
5.	Introduction to Jenkins: What is Jenkins?, Installing Jenkins on Local or Cloud Environment, Configuring Jenkins for First Use	18-24
6.	Continuous Integration with Jenkins: Setting Up a CI Pipeline, Integrating Jenkins with Maven/Gradle, Running Automated Builds and Tests	25-32
7.	Configuration Management with Ansible: Basics of Ansible: Inventory, Playbooks, and Modules, Automating Server Configurations with Playbooks, Hands-On: Writing and Running a Basic Playbook	33-39
8.	Practical Exercise: Set Up a Jenkins CI Pipeline for a Maven Project, Use Ansible to Deploy Artifacts Generated by Jenkins	
9.	Introduction to Azure DevOps: Overview of Azure DevOps Services, Setting Up an Azure DevOps Account and Project	40-51
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	52-60
11.	Creating Release Pipelines: Deploying Applications to Azure App Services, Managing Secrets and Configuration with Azure Key Vault, Hands-On: Continuous Deployment with Azure Pipelines	61-63
12.	Practical Exercise and Wrap-Up: Build and Deploy a Complete DevOps Pipeline, Discussion on Best Practices and Q&A	
	VivaQuestions	64-68

EVALUATION SHEET

Sl. No	Date of Conduction	TITLE OF THE EXPERIMENT	Date of Submission	Conduction (20)	Faculty sign with Date
1		Introduction to Maven and Gradle: Overview of Build Automation Tools, Key Differences Between Maven and Gradle, Installation and Setup			
2		Working with Maven: Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins			
3		Working with Gradle: Setting Up a Gradle Project, Understanding Build Scripts (Groovy and Kotlin DSL), Dependency Management and Task Automation			
4		Practical Exercise: Build and Run a Java Application with Maven, Migrate the Same Application to Gradle			
5		Introduction to Jenkins: What is Jenkins?, Installing Jenkins on Local or Cloud Environment, Configuring Jenkins for First Use			
6		Continuous Integration with Jenkins: Setting Up a CI Pipeline, Integrating Jenkins with Maven/Gradle, Running Automated Builds and Tests			
7		Configuration Management with Ansible: Basics of Ansible: Inventory, Playbooks, and Modules, Automating Server Configurations with Playbooks, Hands-On: Writing and Running a Basic Playbook			
8		Practical Exercise: Set Up a Jenkins CI Pipeline for a Maven Project, Use Ansible to Deploy Artifacts Generated by Jenkins			
9		Introduction to Azure DevOps: Overview of Azure DevOps Services, Setting Up an Azure DevOps Account and Project			
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			
11		Creating Release Pipelines: Deploying Applications to Azure App Services, Managing Secrets and Configuration with Azure Key Vault, Hands-On: Continuous Deployment with Azure Pipelines			
12		Practical Exercise and Wrap-Up: Build and Deploy a Complete DevOps Pipeline, Discussion on Best Practices and Q&A			
Average of Conduction					

PROGRAM-1

Introduction to Maven and Gradle: Overview of Build Automation Tools, Key Differences Between Maven and Gradle, Installation and Setup

STEP1:Install Eclipse using this link

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

ENTERPRISE JAVA AND WEB DEVELOPERS

The screenshot shows the 'eclipseinstaller' website by Oomph. It features a search bar with 'type filter text' and a 'SPONSOR' button. Below the search bar, there are four main sections: 1) 'Eclipse IDE for Java Developers' with a Java EE IDE icon, describing it as the essential tools for any Java developer. 2) 'Eclipse IDE for Enterprise Java and Web Developers' with a Java EE IDE icon, describing it as tools for Java and Web applications. 3) 'Eclipse IDE for C/C++ Developers' with a C/C++ icon, describing it as an IDE for C/C++ developers. 4) 'Eclipse IDE for Embedded C/C++ Developers' with an Embedded C/C++ icon.

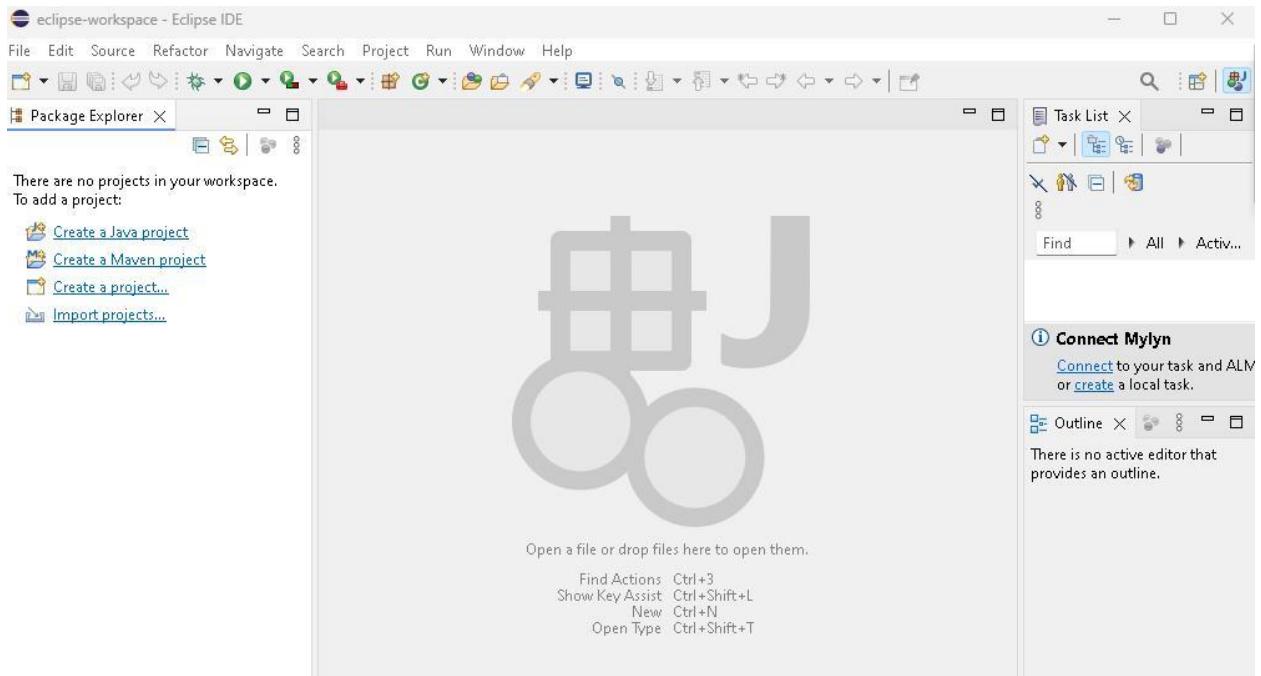
STEP2:Select ECLIPSE IDE FOR JAVA DEVELOPER OR ECLIPSE IDE

FOR STEP3:SELECT INSTALLATION FOLDER JAVA VERSION

The screenshot shows the 'Eclipse IDE for Java Developers' setup screen. It includes a 'details' link, a description of the essential tools for Java developers, and dropdown menus for 'Java 21+ VM' (set to 'IE 23.0.1') and 'Installation Folder' (set to 'C:\Users\abhijith.k_cmrit\eclipse\java-2024-12'). At the bottom, there are two checked checkboxes: 'create start menu entry' and 'create desktop shortcut'. A large orange 'INSTALL' button is at the bottom.

◀ BACK

STEP4:ONCE ECLIPSE IS INSTALLED THE SCREEN LOOKS AS IN BELOW



STEP5:Lets see Procedure to install MAVEN & GRADLE

a) First make sure JDK current version is installed

<https://www.oracle.com/java/technologies/downloads/?er=221886#jdk23-windows>

Then set environment variable path both user and system

- Have a JDK installation on your system. Either set the **JAVA_HOME** environment variable pointing to your JDK installation or have the **java executable** on your PATH.

b) To install apache maven pls go to link as in below and download zip file of bin

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

The screenshot shows the Apache Maven download page. The top navigation bar includes "Welcome", "License", "ABOUT MAVEN", "What is Maven?", "Apache", "Download", "Use", "Release Notes", "DOCUMENTATION", "Maven Plugins", "Maven Repository", "Maven Tools", "Index (category)", "User Guide", "Plugin Developer Guide", "Maven Repository Guide", "Maven Developer Guide", "Books and Resources", "Security", "COMMUNITY", "Community Overview", "Project Details", "How to Contribute", "Getting Help", "Issue Management", "Getting Maven Source", "The Maven Team", and "PROJECT DOCUMENTATION". The main content area is titled "DOWNLOADING Apache Maven 3.9.0". It states "Apache Maven 3.9.0 is the latest release. It is the recommended version for all users." Below this is a "System Requirements" section. Under "Java Development Kit (JDK)", it says "Maven 3.9+ requires JDK 8 or above to execute. It still allows you to build against 1.3 and other JDK versions by using toolchains." Under "Memory", it says "No minimum requirement". Under "Disk", it says "Approximately 10MB is required for the Maven installation itself. In addition to that, disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB". Under "Operating System", it says "No minimum requirement. Start up scripts are included as shell scripts (readme on many Unix flavors) and Windows batch files." A "Files" section follows, with a table showing download links for different formats:

	LINK	CHECKSUMS	SIGNATURE
Binary tar.gz archive	apache-maven-3.9.0-bin.tar.gz	apache-maven-3.9.0-bin.tar.gz.sha12	apache-maven-3.9.0-bin.tar.gz.asc
Binary zip archive	apache-maven-3.9.0-bin.zip	apache-maven-3.9.0-bin.zip.sha12	apache-maven-3.9.0-bin.zip.asc
Source tar.gz archive	apache-maven-3.9.0-src.tar.gz	apache-maven-3.9.0-src.tar.gz.sha12	apache-maven-3.9.0-src.tar.gz.asc
Source zip archive	apache-maven-3.9.0-src.zip	apache-maven-3.9.0-src.zip.sha12	apache-maven-3.9.0-src.zip.asc

At the bottom, there are links for "3.9.0 Release Notes and Release Reference Documentation", "Source code from source repository", "Download under the Apache License, version 2.0", "Other", and "All current release sources (page 1 shared licenses, available at <http://downloads.apache.org/maven/>)".

c) To unzip the Source zip archive

Run in Windows cmd prompt

unzip apache-maven-3.9.9-bin.zip

If don't want to run directly extract the file to Program Files

d) Setup a PATH in environmental settings

**"Add the bin directory of the created directory apache-maven-3.9.9 to the
PATH environment variable"**

e) After environment variable is set

Run this command in CMD prompt

mvn --v(2 hyphen)

After running, you see the text screen as in below

```
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfc97d260186937)
Maven home: /opt/apache-maven-3.9.9
Java version: 1.8.0_45, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os X", version: "10.8.5", arch: "x86_64", family: "mac"
```

f) TO INSTALL GRADLE FOR WINDOWS follow procedure as in below

1. Create a new directory C:\Gradle with File Explorer.

2. Open a second File Explorer window and go to the directory where the Gradle distribution was downloaded. Double-click the ZIP archive to expose the content. **Drag the content folder gradle-8.12.1 to your newly created C:\Gradle folder.**

Alternatively you can unpack the Gradle distribution ZIP into C:\Gradle using an archiver tool of your choice or run command with path folder where the folder is created.

unzip apache-maven-3.9.9-bin.zip

Or can directly extract the zip file.

3. Configure your system environment**4. Finally type the command `gradle -v` to check if the gradle is installed.**

```
Gradle 8.12.1
-----
Build time: 2025-01-24 12:55:12 UTC
Revision: 0blee1ff81d1f4a26574ff4a362ac9180852b140

Kotlin: 2.0.21
Groovy: 3.0.22
Ant: Apache Ant(TM) version 1.10.15 compiled on August 25 2024
Launcher JVM: 21.0.5 (Oracle Corporation 21.0.5+9-LTS-239)
Daemon JVM: C:\Program Files\Java\jdk-21 (no JDK specified, using current Java home)
OS: Windows 11 10.0 amd64
```

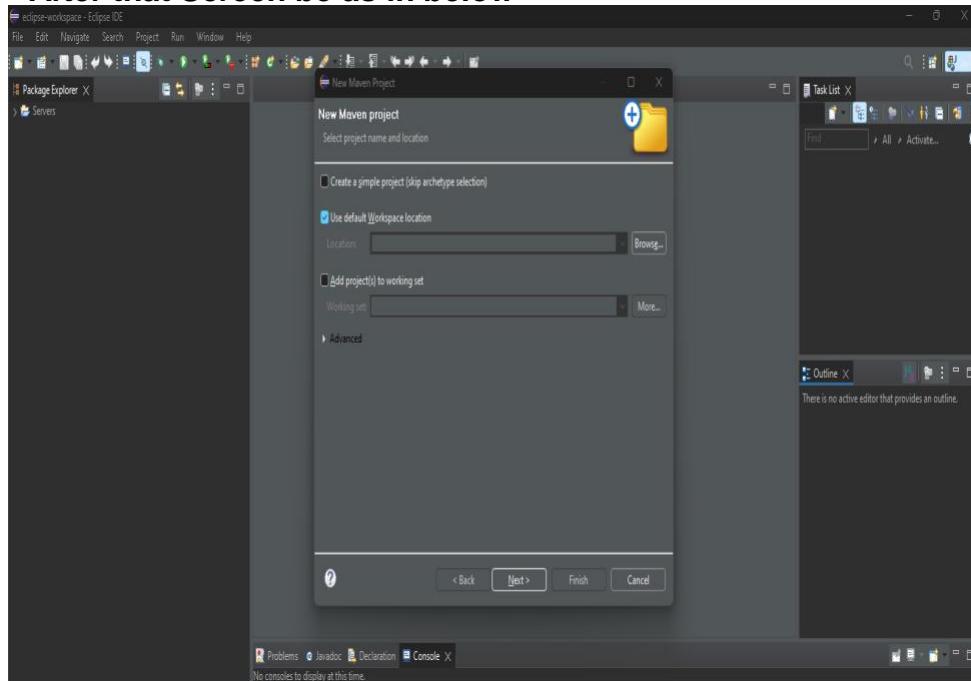
I PROGRAM-2

Working with Maven: Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins

STEP1:OPEN ECLIPSE THEN follow this navigation

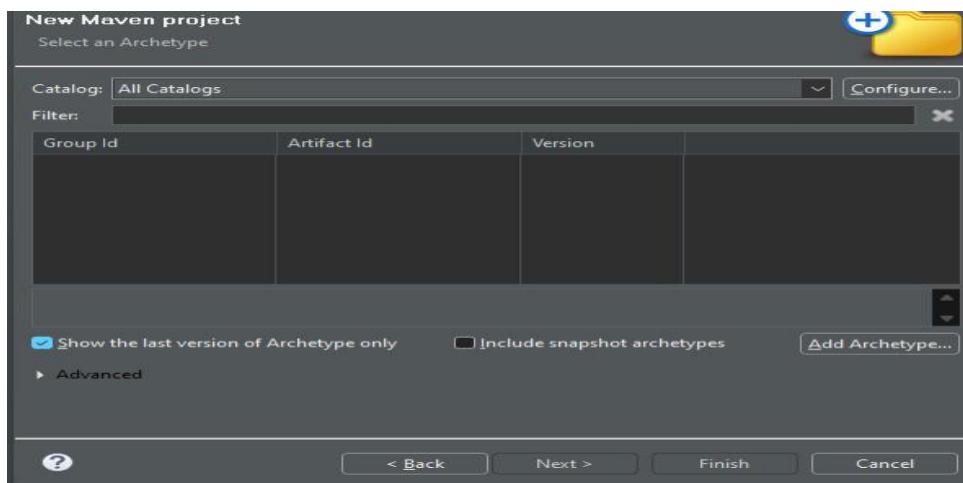
File -----> New -----> Maven Project

After that Screen be as in below



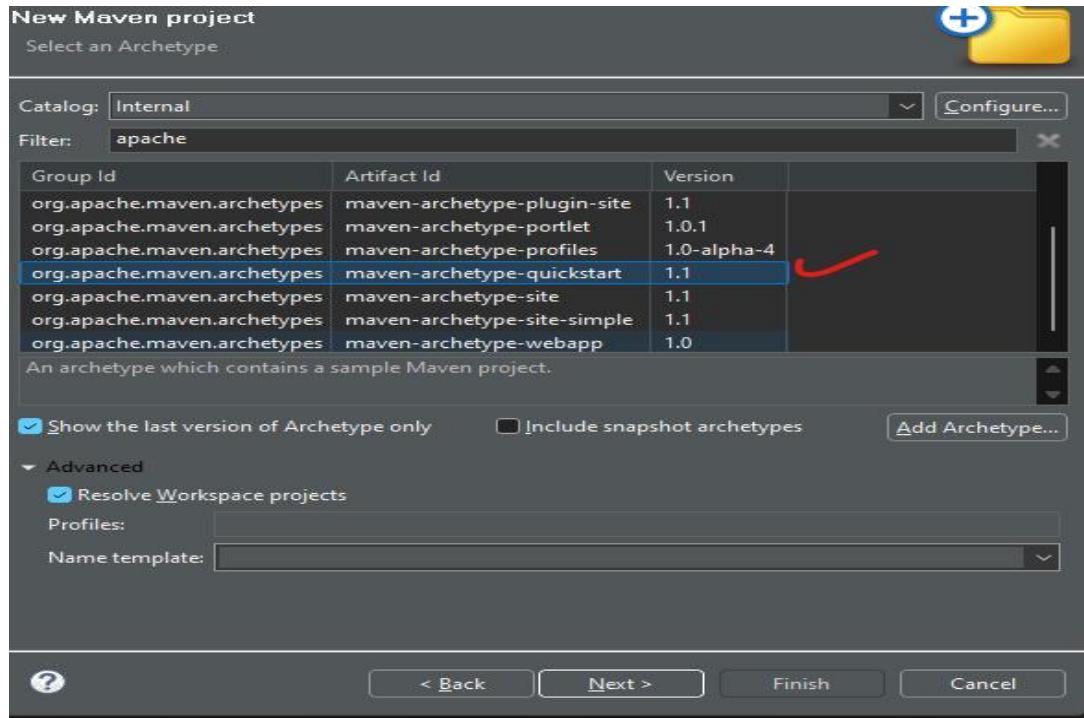
STEP2: Make sure Use default Workspace Location is selected, then click Next

The screen be as in below



STEP3: In Screen shown above, click near the entry place of Filter and type “apache” or select catalog as Internal

We want a simple maven JAR based application. So, we will choose the “maven-archetype-quickstart” artifact to create the project.



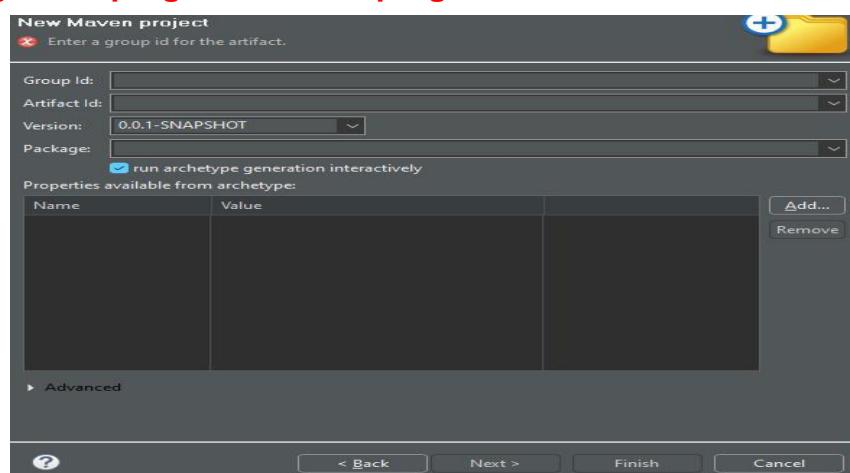
STEP4: Enter

Group Id: com.program2.maven

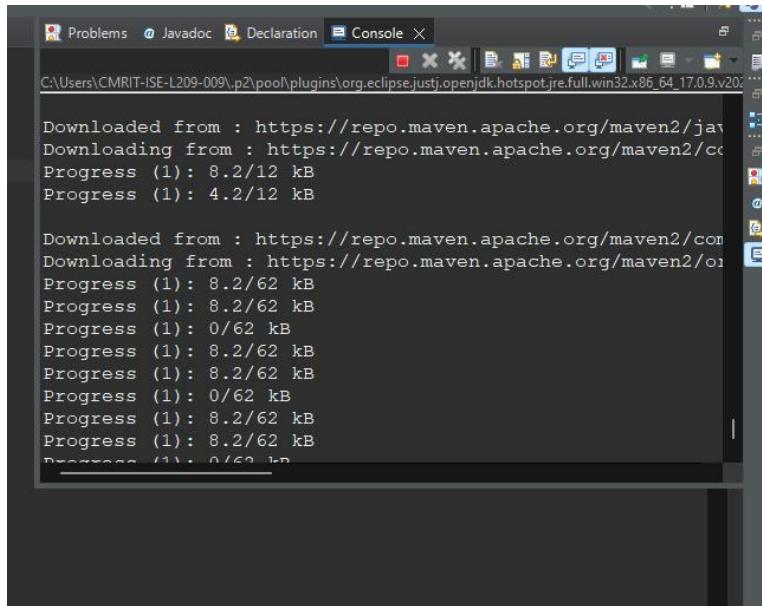
Artifact Id: program2-example-jar

Keep snapshot as it is

Package: com.program2.maven.program2



**After entering above mentioned details click on Finish
You be able to see the automation build happening for Maven Jar Pro**



It asks for Configuration confirmation just click Y

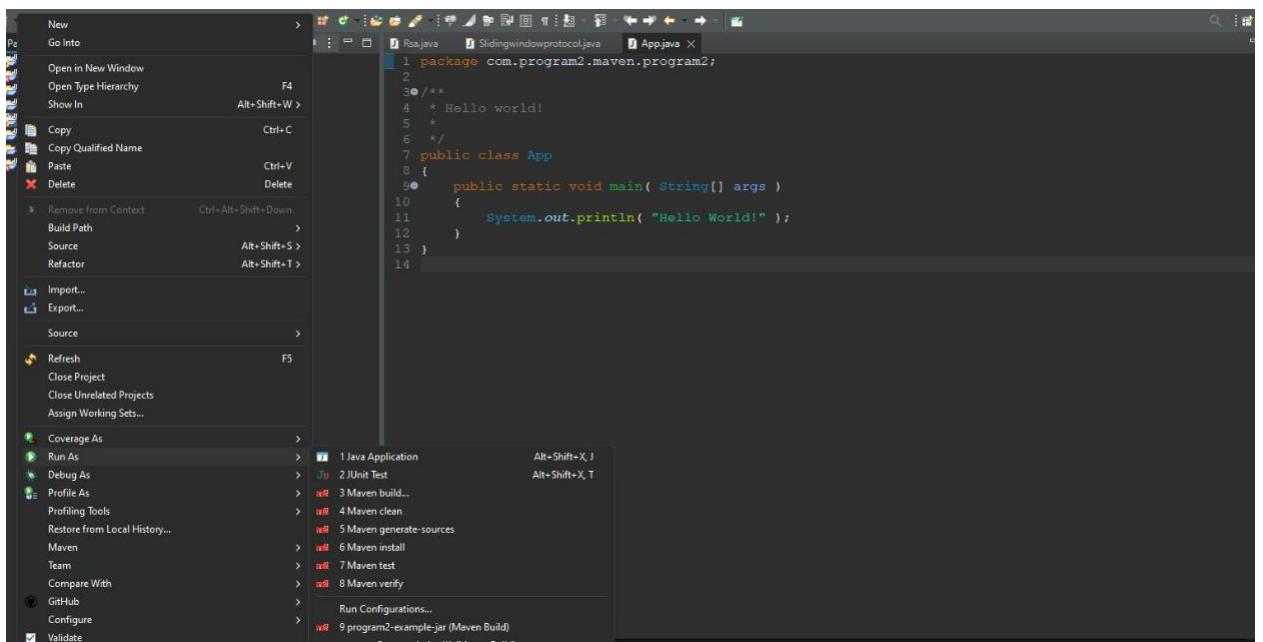
```
Confirm properties configuration:  
groupId: com.program2.maven  
artifactId: program2-example-jar  
version: 0.0.1-SNAPSHOT  
package: com.program2.maven.program2
```

The RESULT be as in below

```
package: com.program2.maven.program2
[Y: y
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.1
[INFO] -----
[INFO] Parameter: basedir, Value: C:\Users\CMRIT-ISE-L209-009\Desktop\IS147
[INFO] Parameter: package, Value: com.program2.maven.program2
[INFO] Parameter: groupId, Value: com.program2.maven
[INFO] Parameter: artifactId, Value: program2-example-jar
[INFO] Parameter: packageName, Value: com.program2.maven.program2
[INFO] Parameter: version, Value: 0.0.1-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\program2-example-jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 03:37 min
[INFO] Finished at: 2025-01-29T10:31:45+05:30
[INFO] -----
```

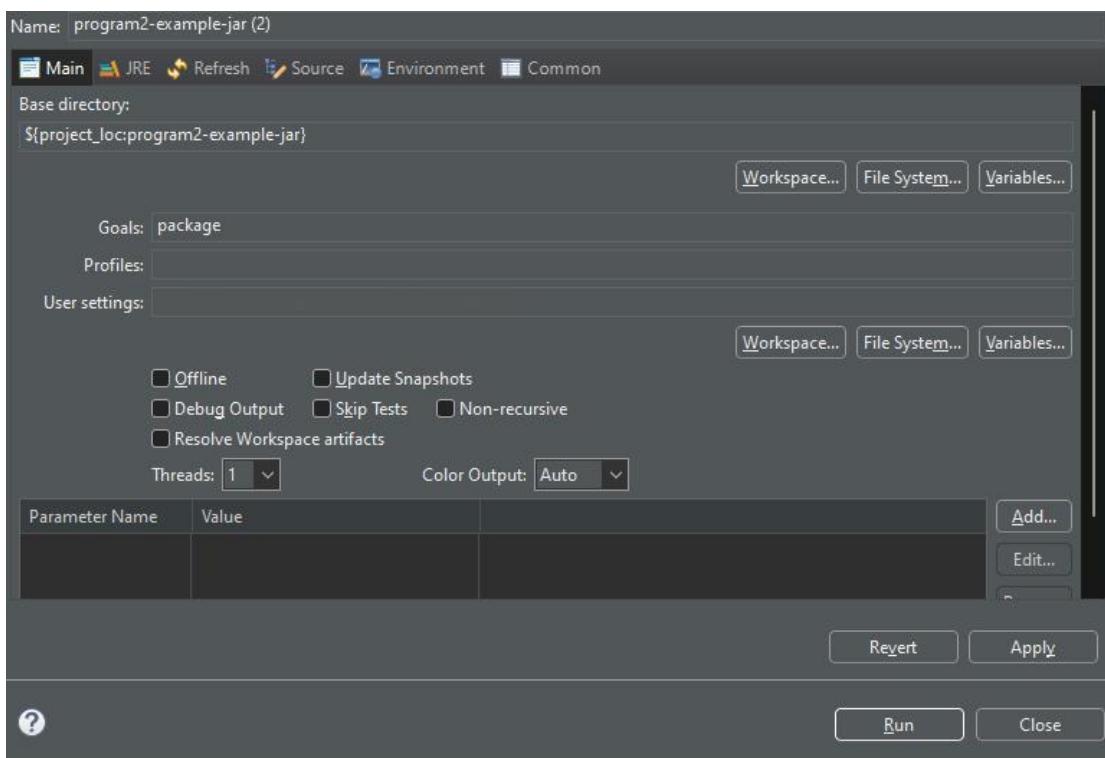
STEP5: Now its time to build Maven Project

Go to Maven Project ----->Right Click on the Project and select Maven Build

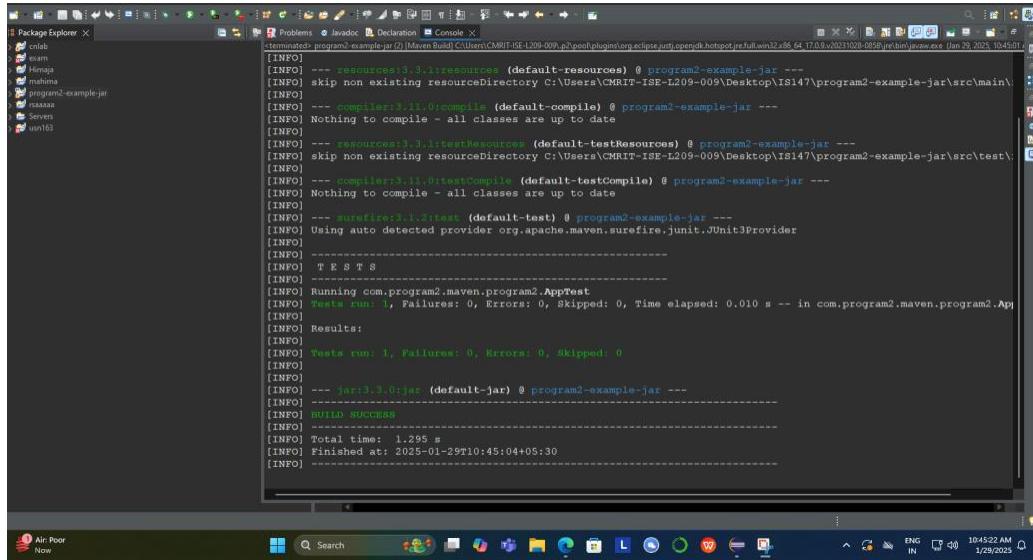


After the above procedure is done

Select Goal as package



**And Click Run
The result be as in below**

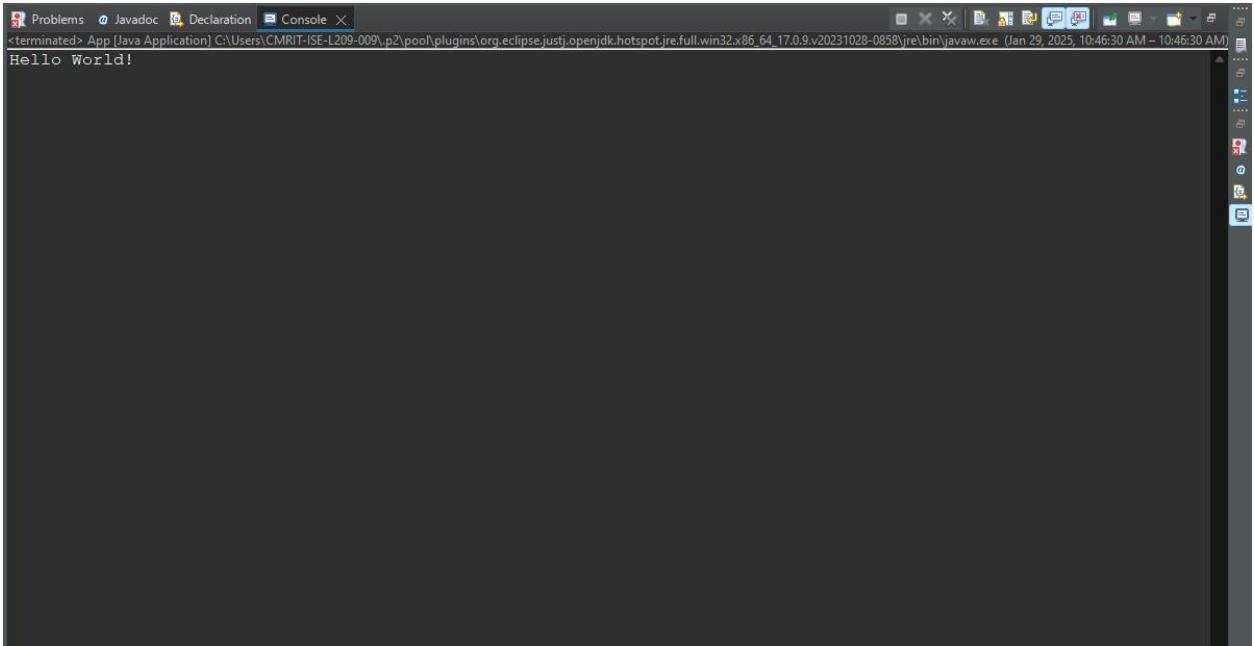


```

[INFO] --- resources:3.3.1:resources (default-resources) @ program2-example-jar ---
[INFO] skip non existing resourceDirectory C:/Users/CMRIT-ISE-L209-009/Desktop/18147/program2-example-jar/src/main/
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- compiler:3.1.1:compile (default-compile) @ program2-example-jar ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ program2-example-jar ---
[INFO] skip non existing resourceDirectory C:/Users/CMRIT-ISE-L209-009/Desktop/18147/program2-example-jar/src/test/
[INFO]
[INFO] --- compiler:3.1.1:testCompile (default-testCompile) @ program2-example-jar ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- surefire:3.1.2:test (default-test) @ program2-example-jar ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit.JUnit3Provider
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.program2.maven.program2.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.010 s -- in com.program2.maven.program2.App
[INFO] Results:
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- jax:3.3.0:jar (default-jar) @ program2-example-jar ---
[INFO]
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.295 s
[INFO] Finished at: 2025-01-29T10:45:04+05:30
[INFO] -----

```

Now goto App.java finally run java application



```

[INFO] Problems @ Javadoc Declaration Console X
<terminated> App [Java Application] C:\Users\CMRIT-ISE-L209-009\.p2\pool\plugins\org.eclipse.jdt.core\jdt.core.jar\bin\javaw.exe (Jan 29, 2025, 10:46:30 AM - 10:46:30 AM)
Hello World!

```

Description

What is groupId in maven ?

groupId identifies a particular project uniquely across all projects, so we should follow a naming convention. A very simple and commonly used way of doing this is to use the reverse of your domain, i.e. com.javarewind.maven.

A good way of maintaining the integrity of groupId is to use the project structure. In case the project consists of multiple modules then every module should append an identifier to the parent groupId. i.e. com.javarewind.maven, com.java rewind.spring, com.javarewind.struts .. etc.

What is artifactId in maven ?

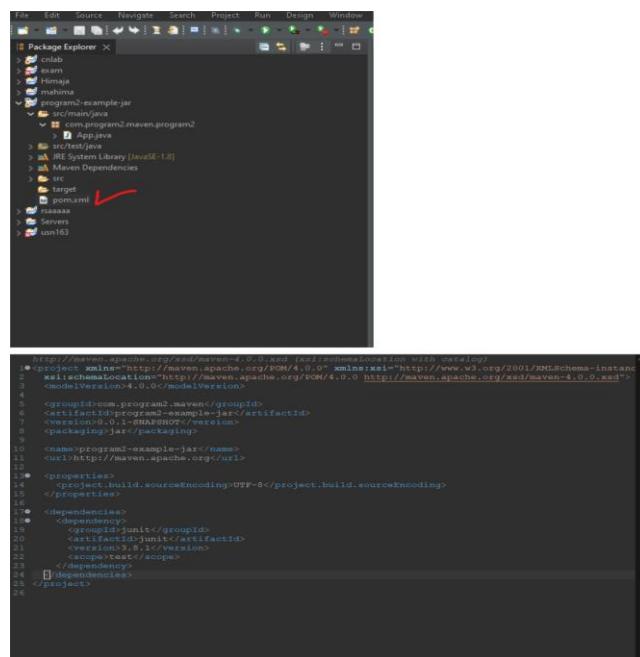
artifactId is the name of war file without version, if you are creating it by yourself you are free to took any name of your choice in lower case and without any strange symbol. But if this is a third party jar than we have to take the name of the jar as suggested by its distribution.

What is the archetype in maven ?

Archetype is a Maven project templating toolkit which tells the maven the type of project we are going to create. Archetype enables the maven to create a template project of the user's choice so that the user can get the project up and running instantly.

“archetype:generate” generates a new project from the provided archetype or updates the actual project if using a partial archetype. Maven provides a number of predefined archetypes, see more details from [Maven Documentation](#).

HOW POM.XML LOOKS IS AS IN SCREEN BELOW



```

<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.program2.maven.program2</groupId>
  <artifactId>program2-example-jar</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>program2-example-jar</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>4.12</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>

```

PROGRAM 3:

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

STEP1:Let's do this in cmd prompt

Goto Command Prompt

Then first make a new directory the command is

mkdir pgm3

For changing to a current directory the command is

cd pgm3

Now run

gradle init

After execution of command the screen shows as in below where we opt for build type select as 1

```
C:\Users\CMRIT-ISE-L209-009\gradletest>gradle init
Starting a Gradle Daemon (subsequent builds will be faster)

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
```

After selecting application type next it asks for Implementation language select as groovy

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

After selecting Implementation language it will ask for Java version and project name

```
Enter selection (default: Java) [1..6] 3  
Enter target Java version (min: 7, default: 21): 21  
Project name (default: gradletest): gradletest
```

After providing version and project name

Select application structure as Single application structure and
Domain Specific Language as Kotlin

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

After every procedure is over it shows Build successful

```
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/8.12.1/samples/sample\_building\_groovy\_applications.html  
  
BUILD SUCCESSFUL in 2m 2s  
1 actionable task: 1 executed
```

STEP2:Now its time to Build the script

Just type the command as:

gradlew run

It will take atleast 3-5 minutes to run the configuration script we have set through steps finally the output be as in below If You want to see the structure of an application run the command as **tree**

```
C:\Users\CMRIT-ISE-L209-009\gradletest>gradlew run
Calculating task graph as no cached configuration is available for tasks: run

> Task :app:run
Hello World!
```

```
C:\Users\CMRIT-ISE-L209-009\gradletest>tree
Folder PATH listing for volume Windows
Volume serial number is CA13-EB08
C:.
    .gradle
        8.12.1
            checksums
            executionHistory
            expanded
            fileChanges
            fileHashes
            vcsMetadata
            buildOutputCleanup
            configuration-cache
                87330068-729b-48ed-8a38-57771bbaae67
                8gybe7c3ykh3sf9t2sllkie4w
            vcs=1
    .settings
    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

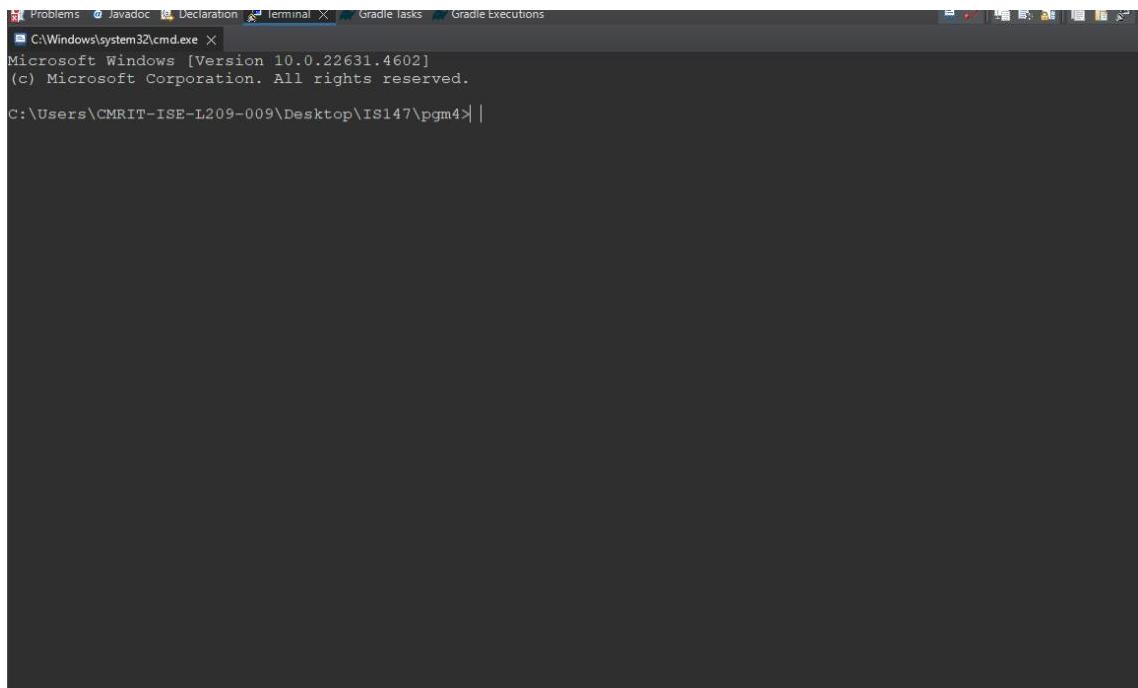
```

PROGRAM 4:

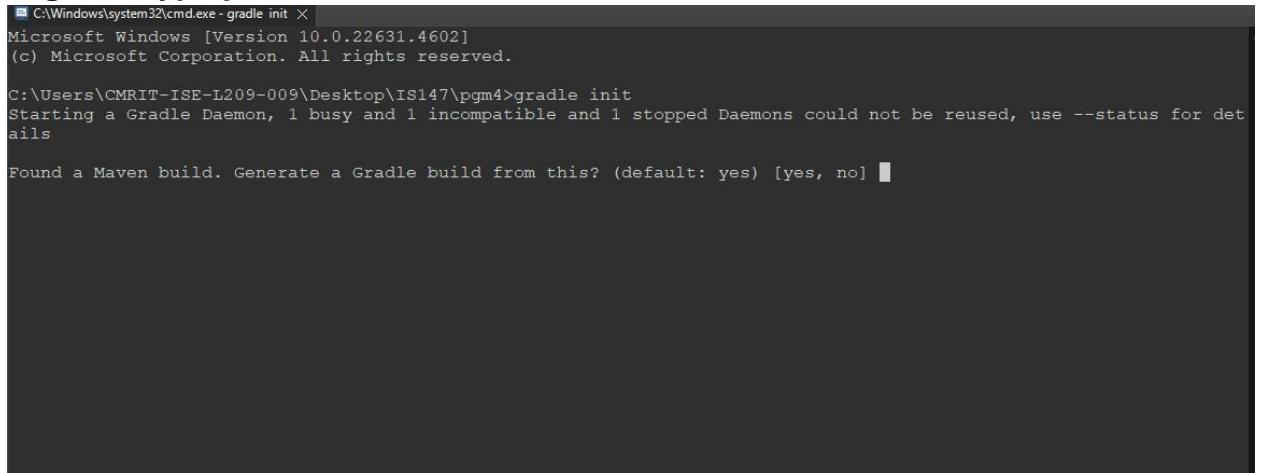
Practical Exercise: Build and Run a Java Application with Maven, Migrate the Same Application to Gradle

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

STEP2: Then to migrate to gradle use shortcut Key Ctrl+Alt+Shift+T To get Terminal screen as in below:

A screenshot of a terminal window titled "Terminal". The window shows a Microsoft Windows command prompt (cmd.exe) running on version 10.0.22631.4602. The path "C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4>" is visible at the top. The main body of the terminal is entirely blank, showing only the cursor character '|'. The window has tabs for "Problems", "Javadoc", "Declaration", "Gradle tasks", and "Gradle Executions".

STEP3: Type command **gradle init it will ask for migrate from maven to gradle type **yes****

A screenshot of a terminal window titled "Terminal". The window shows a Microsoft Windows command prompt (cmd.exe) running on version 10.0.22631.4602. The path "C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4>" is visible at the top. The user types "gradle init" and presses Enter. The terminal then displays a message about starting a Gradle Daemon and handling stopped Daemons. It then asks the user if they want to generate a Gradle build from a Maven build, with options "[yes, no]". The user has not yet responded.

STEP4: After the above command is validated to yes it prompts to select Domain Specific Language as in screen below select 2 (as we have done for Kotlin)

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

STEP5: After selecting Groovy it asks for validating prompt for API Generator just validate as **yes**

```
C:\Windows\system32\cmd.exe - gradle init X

C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4>gradle init

Found a Maven build. Generate a Gradle build from this? (default: yes) [yes, no] yes

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

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

Finally it runs the init phase as been selected

```
> Task :init
Maven to Gradle conversion is an incubating feature.
For more information, please refer to https://docs.gradle.org/8.12.1/userguide/migrating_from_maven.html in the Gradle documentation.

BUILD SUCCESSFUL in 6m 44s
1 actionable task: 1 executed
C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4> |
```

STEP6:

Type the command
gradle build

```
C:\Windows\system32\cmd.exe X

C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4>gradle build
Reusing configuration cache.

BUILD SUCCESSFUL in 759ms
1 actionable tasks: 4 up-to-date
Configuration cache entry reused.
C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4>
```

Now to get exact program output of our java file Locate to build gradle File from ur local repository and Copy paste the code as in below shown in red color

```
plugins {  
    id("java-library")  
    id("maven-publish")  
    id("application")  
}  
  
application {  
    mainClass.set("com.pgm4.test.App") // Use .set() for properties  
}  
  
repositories {  
    mavenCentral()  
    // Uncomment if you need to publish locally  
    // mavenLocal()  
}  
  
dependencies {  
    testImplementation("junit:junit:4.13.2") // Use Kotlin syntax for dependencies  
}  
  
group = "com.pgm4.test"  
version = "0.0.1-SNAPSHOT"  
description = "pgm4"  
java.sourceCompatibility = JavaVersion.VERSION_11 // Consider upgrading  
  
publishing {  
    publications {  
        create< MavenPublication >("maven") {  
            from(components["java"])  
        }  
    }  
}  
  
tasks.withType<JavaCompile>().configureEach {  
    options.encoding = "UTF-8"  
}
```

```
tasks.withType<Javadoc>().configureEach {  
    options.encoding = "UTF-8"  
}
```

AFTER DOING ALL CHANGES FINAL STEP

To run commands

gradle clean build

gradle run

You will get Output as

Hello World! Welcome to pgm4

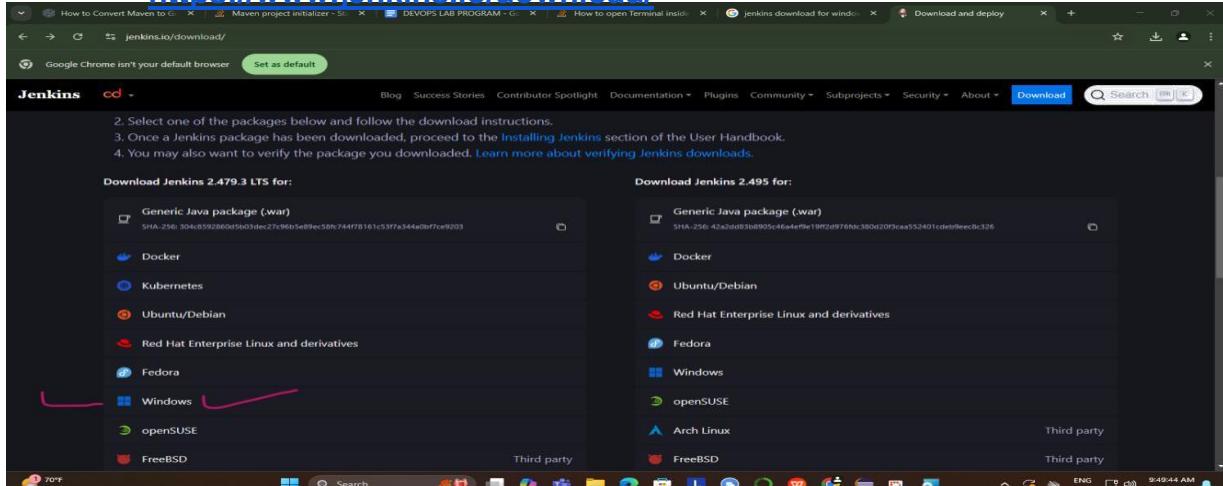
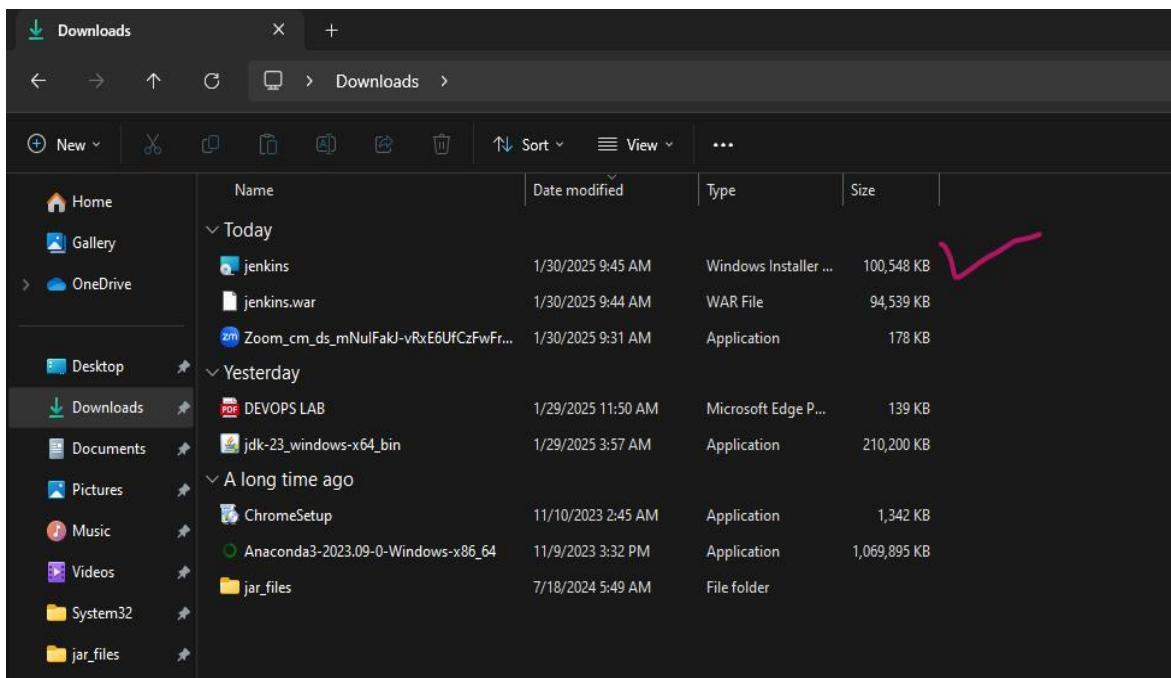
```
BUILD SUCCESSFUL in 1s  
8 actionable tasks: 6 executed, 2 from cache  
Configuration cache entry stored.  
C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4>gradle run  
Calculating task graph as configuration cache cannot be reused because file 'build.gradle' has changed.  
> Task :run  
Hello World! Welcome to pgm4  
  
BUILD SUCCESSFUL in 883ms  
2 actionable tasks: 1 executed, 1 up-to-date  
Configuration cache entry stored.  
C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4>gradlew run  
Reusing configuration cache.  
  
> Task :run  
Hello World! Welcome to pgm4  
  
BUILD SUCCESSFUL in 810ms  
2 actionable tasks: 1 executed, 1 up-to-date  
Configuration cache entry reused.  
C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4> |
```

PROGRAM5:

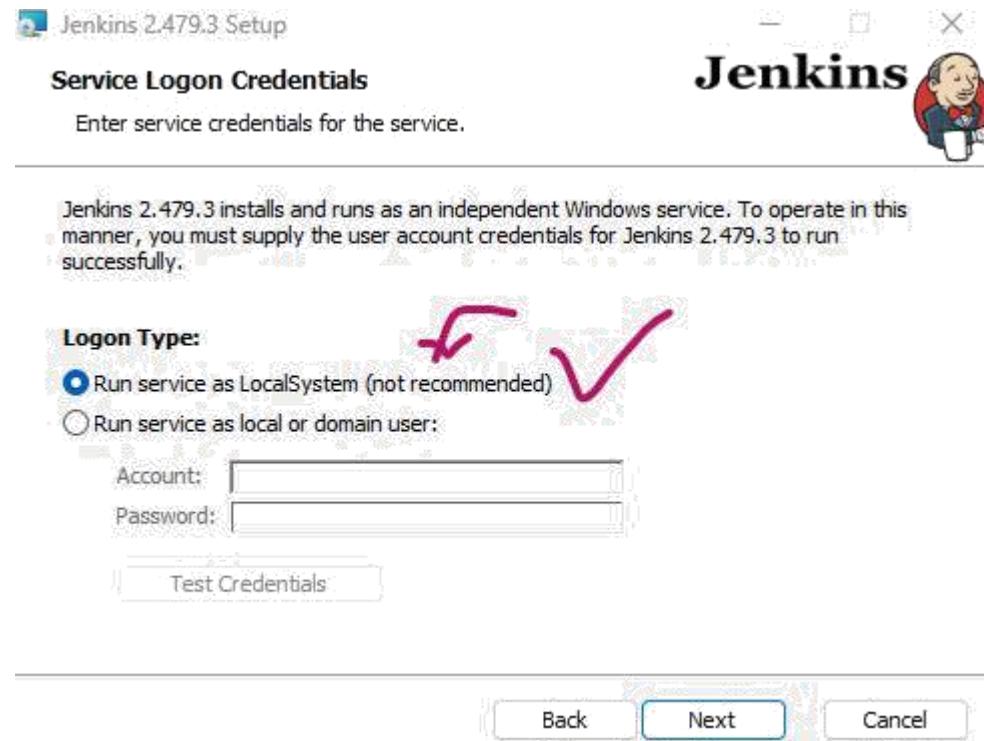
Introduction to Jenkins: What is Jenkins?, Installing Jenkins on Local or Cloud Environment, Configuring Jenkins for First Use .

STEP1: Type jenkins download for windows

<https://www.jenkins.io/download/>

**STEP2: After clicking on Windows Jenkins MSI Installer exe file be installed**

STEP3: Goto Jenkins MSI Installer click on it u opt for “Run service as Local System”



XX Need not to Provide Account & Password XX

STEP4: Choose a port as 8080 and test the port and click Next



STEP5:It takes current jdk version that's available for safer side once goto cmd prompt and type command

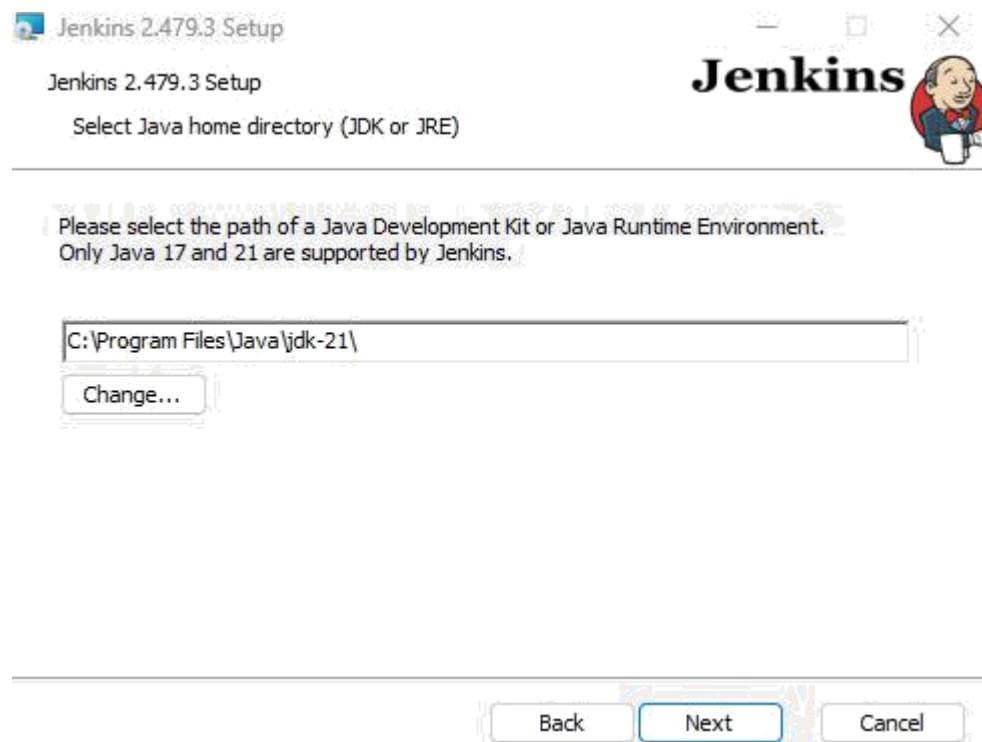
java -version

If matching click next

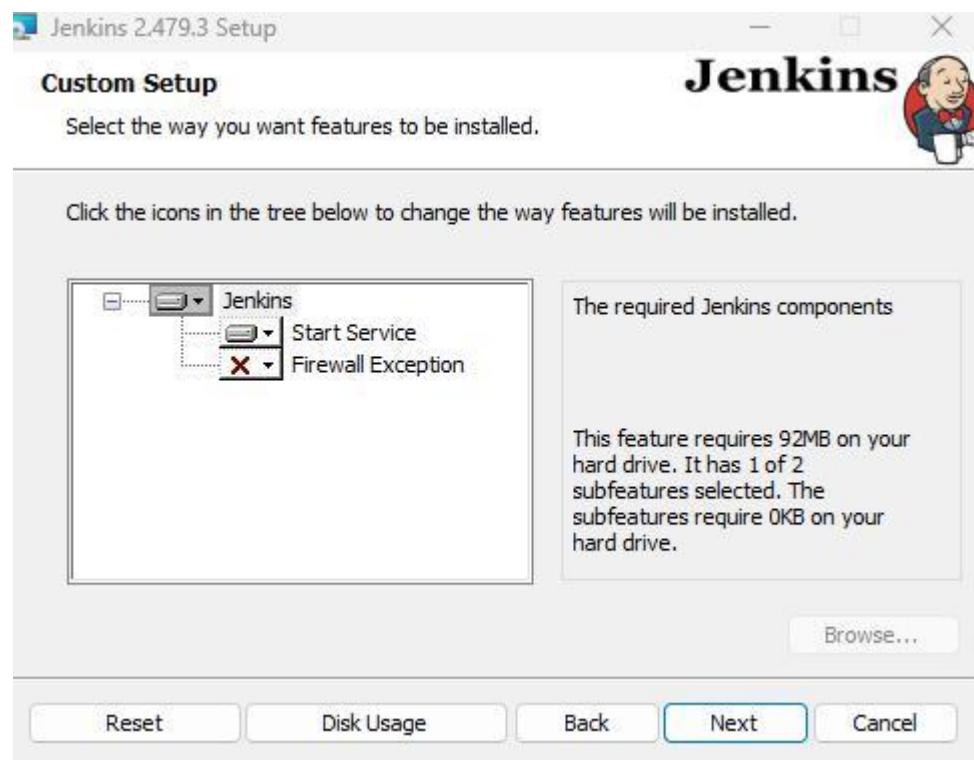
If failed to accept download jdk version 17 to 21 any of it

<https://www.oracle.com/java/technologies/javase/jdk21-archive-downloads.html>

Platform	File Type	Size	Link
Linux x64 Debian Package	http	160.30 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_linux-bin.deb.sha256
Linux x64 RPM Package	http	188.18 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_linux-bin.rpm.sha256
macOS Arm 64 Compressed Archive	http	182.27 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_macos-aarch64_bin.tar.gz.sha256
macOS Arm 64 DMG Installer	https	181.55 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_macos-aarch64_bin.dmg.sha256
macOS x64 Compressed Archive	https	184.51 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_macos-x64_bin.tar.gz.sha256
macOS x64 DMG Installer	https	183.85 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_macos-x64_bin.dmg.sha256
Windows x64 Compressed Archive	https	185.91 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_windows-x64_bin.zip.sha256
Windows x64 Installer	https	164.28 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_windows-x64_bin.exe.sha256
Windows x64 msi Installer	https	165.03 MB	https://download.oracle.com/java/21/archive/jdk-21.0.5_windows-x64_bin.msi.sha256

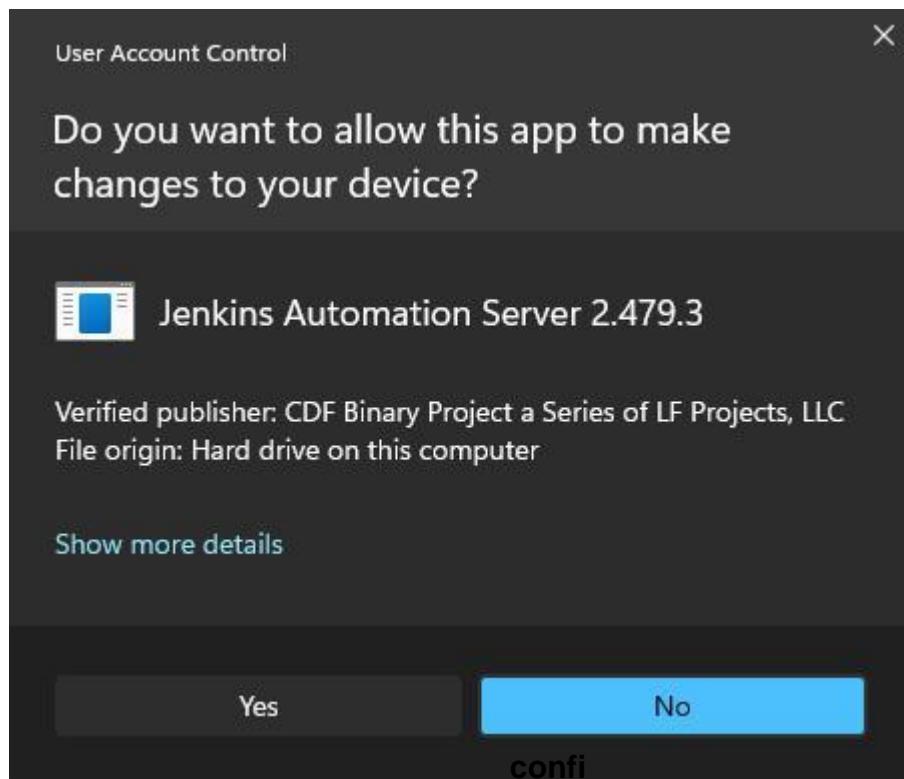


STEP6: Click NEXT after doing above step you will get screen as in Below again continue to click Next



STEP7: The popup allow format comes for Java Automation Server allow it click install -----> Finally

Click Finish



Now By default Our Jenkins run at

<http://localhost:8080/>

Final step very important Once It will ask for Administrator Passowrd 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 mentioned belows administration passowrd

The screenshot shows a "Getting Started" page titled "Unlock Jenkins". It instructs the user to ensure Jenkins is securely set up by an administrator, who has written a password to a log file and a specific file path: **C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword**. It asks the user to copy the password from either location and paste it into a text input field labeled "Administrator password". A red checkmark is placed next to the input field. At the bottom right is a blue "Continue" button.

Then select Install Suggested Plugins it starts to install as in shown below

Getting Started

Getting Started

<input checked="" type="checkbox"/> Folders	<input checked="" type="checkbox"/> OWASP Markup Formatter	<input checked="" type="checkbox"/> Build Timeout	<input checked="" type="checkbox"/> Credentials Binding	folders OWASP Markup Formatter ** ASM API ** JSON Path API ** Structs ** Pipeline: Step API ** Token Macro Build Timeout ** bouncycastle API ** Credentials ** Plain Credentials ** Variant ** SSH Credentials Credentials Binding ** SCM API ** Pipeline: API ** commons-lang3 v3.x Jenkins API Timestamper ** Caffeine API ** Script Security ** JavaBeans Activation Framework (JAF) API ** JAXB ** SnakeYAML API ** - required dependency
<input checked="" type="checkbox"/> Timestamper	<input checked="" type="checkbox"/> Workspace Cleanup	<input checked="" type="checkbox"/> Ant	<input checked="" type="checkbox"/> Gradle	
<input checked="" type="checkbox"/> Pipeline	<input checked="" type="checkbox"/> GitHub Branch Source	<input checked="" type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input checked="" type="checkbox"/> Pipeline Graph View	
<input checked="" type="checkbox"/> Git	<input checked="" type="checkbox"/> SSH Build Agents	<input checked="" type="checkbox"/> Matrix Authorization Strategy	<input checked="" type="checkbox"/> PAM Authentication	
<input checked="" type="checkbox"/> LDAP	<input checked="" type="checkbox"/> Email Extension	<input checked="" type="checkbox"/> Mailer	<input checked="" type="checkbox"/> Dark Theme	

Jenkins 2.479.3

Then it asks for minimum registration You can skip and continue as admin

Getting Started

Create First Admin User

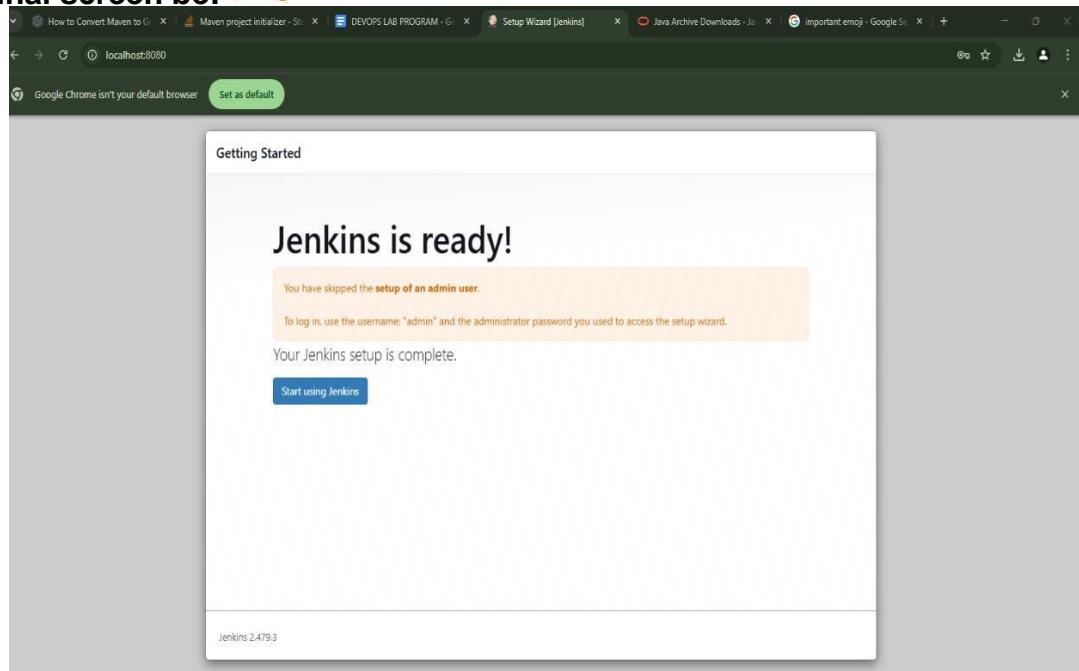
Username	<input type="text"/>
Password	<input type="password"/>
Confirm password	<input type="password"/>
Full name	<input type="text"/>
E-mail address	<input type="text"/>

Skip and continue as admin

Save and Continue

Jenkins 2.479.3

Final screen be: 😊



PROGRAM 6 :**Continuous Integration with Jenkins: Setting Up a CI Pipeline, Integrating Jenkins with Maven/Gradle, Running Automated Builds and Tests****How Is Jenkins Used for Continuous Integration?**

Continuous Integration (CI) is a software development practice where developers integrate code into a shared repository frequently, usually several times a day. Jenkins is an open-source automation server that facilitates CI by automating the build, testing, and deployment processes.

With Jenkins, developers can easily detect and fix integration issues early, improving collaboration and accelerating software delivery. By continuously integrating code, teams can maintain a high level of code quality, reduce development time, and minimize the risk of release failures.

Continuous Integration Features in Jenkins

Continuous integration involves the automatic building and testing of code whenever changes are committed to the version control system. Jenkins provides several features that facilitate CI, including:

- Version control system integration: Jenkins integrates with various version control systems (VCS) such as Git, Subversion, and Mercurial. This allows Jenkins to monitor repositories for changes, trigger builds, and incorporate updates automatically.
- Build automation: Jenkins supports build automation using build tools like Maven, Gradle, and Ant. It can compile, package, and deploy code, ensuring that the latest changes are continuously integrated into the software project.
- Automated testing: Jenkins can execute automated tests for each build, using testing frameworks like JUnit, TestNG, and Selenium. This ensures that any issues introduced during development are quickly detected and reported, allowing developers to address them promptly.
- Pipeline as code: Jenkins Pipeline allows users to define their entire CI/CD pipeline as code using a domain-specific language called “Groovy.” This makes the pipeline easily versionable, shareable, and more maintainable.
- Distributed builds: Jenkins supports distributed builds across multiple build agents, which

It allows for faster and more efficient build processes by distributing the workload across multiple machines.

- Plugins and extensibility: Jenkins offers a vast ecosystem of plugins that extend its functionality, allowing users to customize and adapt Jenkins to their specific needs. Plugins are available for various tasks, such as integrating with different VCS, build tools, notification systems, and more.
- Notifications and reporting: Jenkins can send notifications through various channels like email, Slack, or other messaging systems to keep the team informed about build status, test results, and potential issues. It also generates reports and visualizations for various metrics, such as test results, code coverage, and build trends.
- Access control and security: Jenkins provides fine-grained access control and user management, allowing administrators to control who can access specific projects, pipelines, or configuration settings. It also supports integration with LDAP and Active Directory for centralized user management.
- REST API: Jenkins exposes a REST API that enables users to interact with Jenkins programmatically, allowing for integration with external tools, automation, and custom applications.

Benefits and Drawbacks of Using Jenkins for CI

Jenkins CI offers numerous benefits that can streamline software development processes and improve overall efficiency:

- Shorter development cycles: By automating repetitive tasks such as building, testing, and deployment, Jenkins CI reduces the time developers spend on manual tasks, enabling them to focus on writing code and addressing critical issues. This accelerates the development cycle and speeds up time-to-market.
- Fast code integration: Jenkins CI facilitates frequent code integration into a shared repository, making it easier to detect and fix integration issues early on. This prevents the accumulation of integration problems, leading to more stable and reliable software.
 - Short feedback loops: The automation provided by Jenkins CI allows developers to receive immediate feedback on the success or failure of their code changes. Rapid feedback helps in identifying problems early, ensuring that they can be addressed before they become more difficult and time-consuming to resolve.

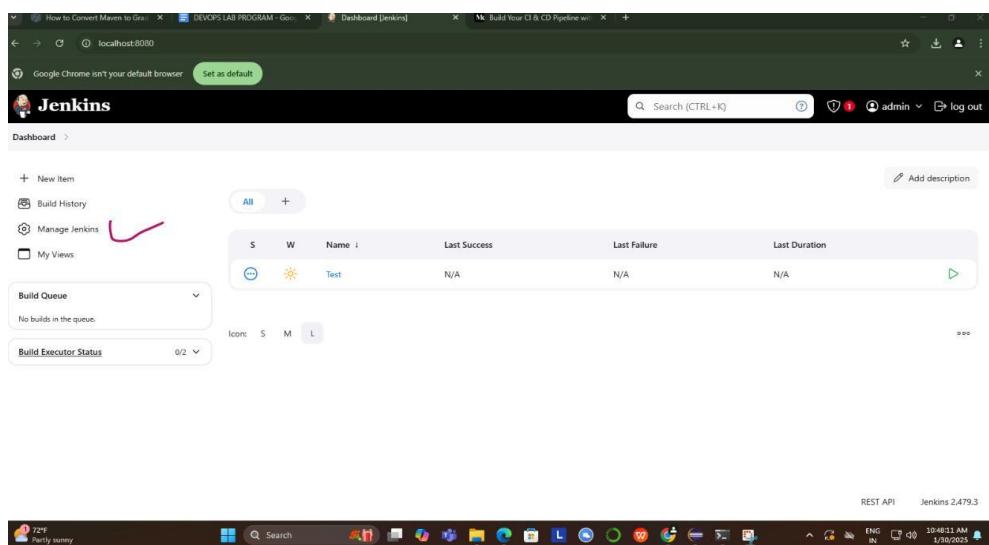
- Automated workflows: Jenkins CI can be configured to trigger automated workflows based on specific events, such as code commits or pull requests. This enables a seamless and efficient flow of work, helping teams maintain a high level of productivity and consistency.

However, there are potential concerns associated with using Jenkins CI:

- Expense: Although Jenkins itself is an open-source tool, the resources and infrastructure required to run and maintain it can be costly, especially for larger projects or organizations. Costs may include hardware, cloud services, or additional plugins and integrations needed for specific use cases.
- Maintenance: Jenkins CI requires regular maintenance to ensure its optimal performance, including updating plugins, monitoring the system for potential issues, and troubleshooting any problems that arise. This maintenance can be time-consuming and may require dedicated personnel with expertise in Jenkins and the underlying technologies.
- Not cloud native: Jenkins was designed before the advent of cloud computing, which means it doesn't naturally lend itself to cloud-based environments. To make Jenkins work in a cloud environment, substantial customization and additional tooling may be needed.

STEP1: Now coming to our Program to set CI Pipeline for Maven

Go to Jenkins Dashboard and Click on Manage Jenkins



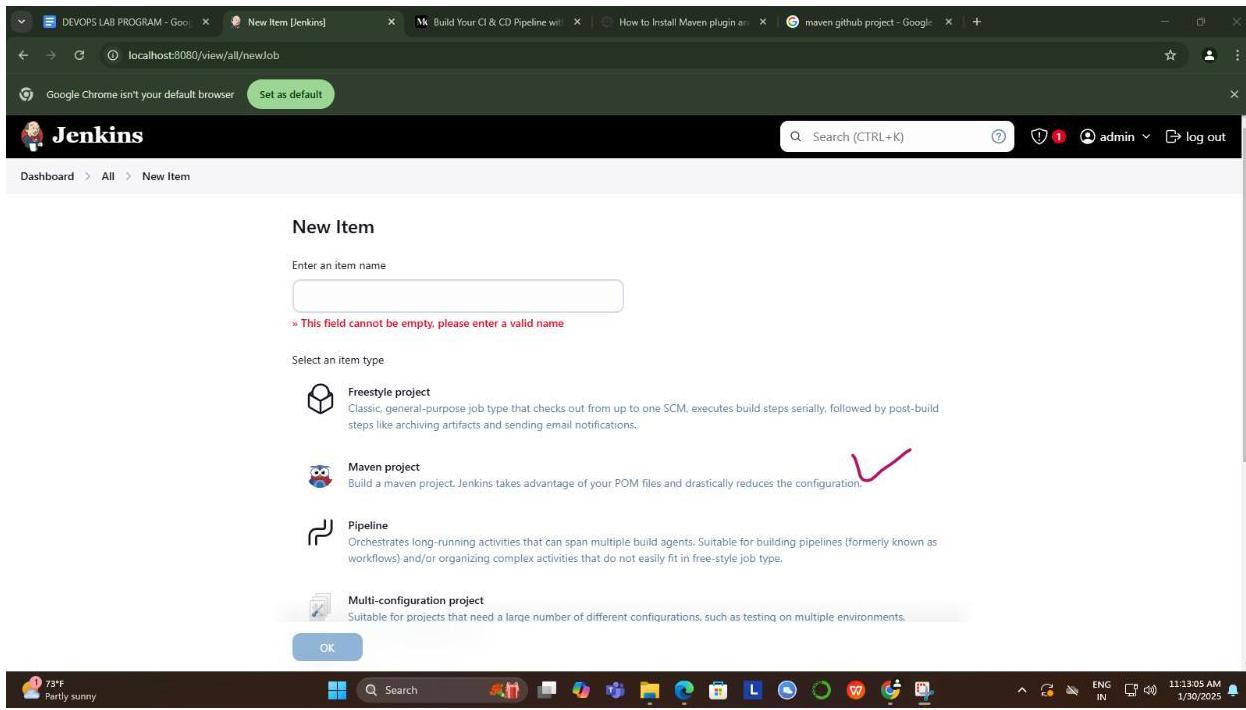
STEP2: Select Plugins

The screenshot shows the Jenkins 'Manage Jenkins' page under the 'System Configuration' section. The 'Plugins' link is circled with a red marker. Other sections visible include 'Build Queue', 'Build Executor Status', 'Nodes', 'Clouds', 'Tools', 'Appearance', 'Security', 'Credentials', and 'Credential Providers'. A status bar at the bottom shows system information like weather and time.

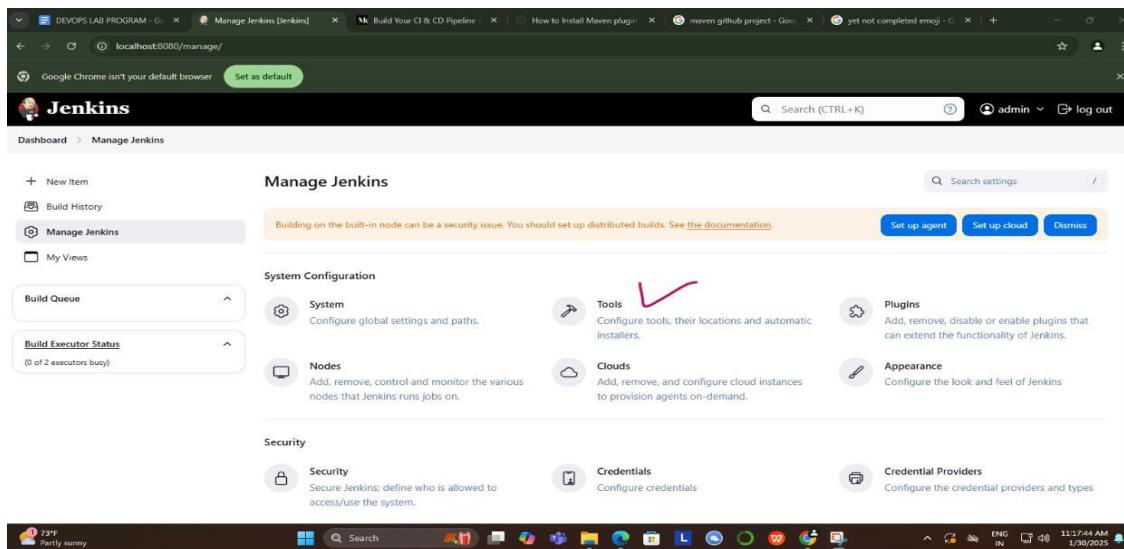
STEP3: Search for Maven IntegrationPlugin in Available Plugins and Install

The screenshot shows the Jenkins 'Manage Jenkins > Plugins' page under the 'Available plugins' tab. A search bar contains the text 'maven integration plugin'. The results table has columns for 'Name', 'Version', and 'Released'. The 'Available plugins' tab is selected. Other tabs shown include 'Updates', 'Installed plugins', 'Advanced settings', and 'Download progress'. A status bar at the bottom shows system information like weather and time.

STEP4: After Maven Integration Plugin is Installed We able to see **Maven Project** as New Item



STEP5: YET not completed **we have to configure the Location to properly Build and Run Maven Project**
So again click on Manage Jenkins and select Tools



STEP6: Now lets not select Maven Project as new Item as we already have Maven project in local systems lets see how we can run the Maven Project with POM.XML

- a) Click on New Item
- b) Provide Item Name and select Freestyle Project

New Item

Enter an item name

Maven Test

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project

OK

- c) Scroll down to 'Build' option. Click on 'Add Build Step' and choose the value 'Invoke top-level Maven targets' from the drop down list.

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Add build step

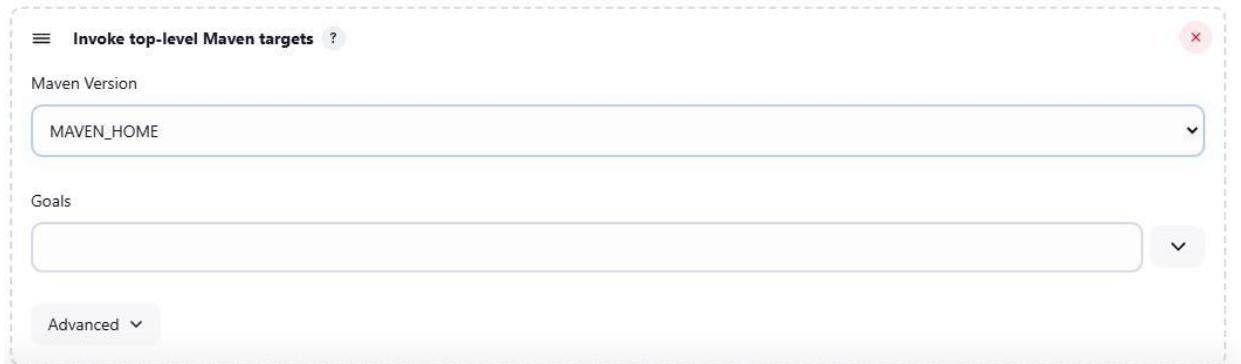
Post-build Actions

Add post-build action

Save Apply

- d) After selecting Invoke top-level Maven targets opt for proper environment version as in set in previous steps in my case its MAVEN_HOME

Build Steps



- e) Enter Goal as

clean install

- f) Before you save and apply just below Goal there is Advance option add pom.xml path



Goto pom.xml of your particular pgm and take path in my case
its **C:\Users\CMRIT-ISE-L209-009\Desktop\IS147\pgm4\pom.xml**

After all Steps is over click on **Build button** the output be as in below

The screenshot shows the Jenkins interface for the 'Maven Test' job. The top navigation bar includes links for 'Dashboard', 'Maven Test', 'admin', and 'log out'. The main content area displays the build status as 'Maven Test #11' with a green checkmark. Below this are sections for 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', and 'Rename'. A 'Builds' section lists the last four builds, all of which were successful. At the bottom right, there is a 'REST API' link.

To see either click on build texts and goto console output or u can goto dashboard and opt to see build scripts.

The screenshot shows a browser window with multiple tabs open. One tab is specifically focused on the Jenkins 'Console Output' for build #11. The output log is displayed, showing the Maven build process. A red checkmark is placed over the 'Console Output' link in the left sidebar of the Jenkins interface, indicating it as a key point of interest.

PROGRAM 7:

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

How Do I Install Ansible on Ubuntu?

Installing Ansible on Ubuntu requires setting up an Ansible control node and connecting it to one or more Ansible hosts. The following steps describe how to perform the necessary configuration and test the new Ansible installation.

STEP 1: Configure Ansible Control Node

The Ansible control node is a system used to connect to and manage Ansible host servers. Proceed with the steps below to set up the control node on the main server:

- 1. Create an administrator-level user for the control node. Use the adduser command:**

sudo adduser [username]

- 2. When prompted, define a strong account password.**

```
marko@phoenixnap:~$ sudo adduser ansible
[sudo] password for marko:
Adding user `ansible' ...
Adding new group `ansible' (1001) ...
Adding new user `ansible' (1001) with group `ansible' ...
Creating home directory `/home/ansible' ...
Copying files from `/etc/skel' ...
New password: ←
Retype new password:
passwd: password updated successfully
Changing the user information for ansible
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] y
marko@phoenixnap:~$ █
```

Optionally, provide more details about the user by answering questions.

Press Enter to skip a question.

3. Use the following usermod command to assign superuser privileges to the account:

sudo usermod -aG sudo [username]

A membership in the sudo group allows the user to utilize the sudo command to perform administrative tasks.

4. Switch to the newly created user on the control node:

sudo su [username]

Note: The Ansible control node can be a dedicated server, a local machine, or a virtual machine running Ubuntu.

STEP 2: Set up an SSH Key pair

The Ansible control node uses SSH to connect to hosts. Generate an SSH key pair for the Ansible user by executing the following steps:

1. Enter the command below using the Ansible control node command line:

ssh-keygen

Note: If an SSH key pair with the same name already exists, SSH displays a warning asking the user to decide whether to overwrite it. Overwriting makes the previous SSH key pair unusable, so ensure the old keys are no longer needed before confirming.

2. When prompted, provide a passphrase. While adding a strong passphrase is recommended, pressing Enter allows the user to skip the passphrase creation.

The system generates the public/private key pair and prints the randomart image.

```
ansible@phoenixnap:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
Created directory '/home/ansible/.ssh'.
Enter passphrase (empty for no passphrase): ←
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_rsa
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:89D+0mk6VsCbHevu/kLk0oiWrrvnjsW+Lsgt3jCdRrE ansible@phoenixnap
The key's randomart image is:
+---[RSA 3072]---+
|                               |
|                               |
|                               |
|                               |
| . . . |
| o .o . |
| E S . = o |
| o o Boo+ |
| .oo+ * =+o.. |
| +=o*. ++=o . |
| ...*XBo.0*.o |
+---[SHA256]---+
ansible@phoenixnap:~$ █
```

STEP 3: Configure an Ansible Host

Ansible hosts are remote servers managed by the Ansible control node. Each host must have the control node's SSH public key into authorized_keys directory. Apply the steps below for each new Ansible host:

1. Use the following ssh-copy-id command on the control node to copy the public key to a host:

ssh-copy-id [username]@[remote-host]

Replace [username] with an existing administrative user on the host system and [remote-host] with the remote host domain or IP address. For example, to copy the key to the user ansible on the host with the local IP address 192.168.0.81, type:

To know IP type command**cat /etc/resolv.conf or hostname -i****ssh ansible@192.168.0.81**

2. Type yes and hit Enter when asked whether to continue connecting to an authenticated host.

3. Enter the remote host account password.

```
ansible@phoenixnap:~$ ssh-copy-id ansible@192.168.0.81
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
The authenticity of host '192.168.0.81 (192.168.0.81)' can't be established.
ED25519 key fingerprint is SHA256:fG67eTA0FjkEJlRAcQyxna/MDc7zX4f0dABzt+aktGM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansible@192.168.0.81's password: ←
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'ansible@192.168.0.81'"
and check to make sure that only the key(s) you wanted were added.

ansible@phoenixnap:~$
```

The utility uploads the public key to the remote host account.

STEP4 : Install Ansible

Use the APT package manager to install the Ansible package on the control node system:

1. Ensure the package index is up to date

sudo apt update

2. Install Ansible on Ubuntu with the following command:

sudo apt install ansible -y

STEP 5: Verify the Installation

Check that Ansible was successfully installed on your Ubuntu system using the `ansible` command:

`ansible --version`

The output displays the Ansible version number, the location of the configuration file, the path to the executable, and other information.

```
ansible@phoenixnap:~$ ansible --version
ansible 2.10.8 ←
  config file = /home/ansible/ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
ansible@phoenixnap:~$
```

STEP 6: Set up the Inventory File

Once Ansible is installed on the control node, set up an inventory file to allow Ansible to communicate with remote hosts. The inventory file contains all the information about the remote hosts managed through the Ansible control node.

Note: For an in-depth overview of creating files on remote hosts, refer to our article [How to Create a File In Ansible](#).

Follow the steps below to create an inventory file on the control node:

1. Create the `ansible` subdirectory in the `etc` directory:

`sudo mkdir -p /etc/ansible`

2. Use a text editor such as Nano to create a file named hosts:

sudo nano /etc/ansible/hosts

3. Add localhost that the control node will manage. Use the following format:

[local]
localhost ansible_connection=local

The [local] line allows for the creation of categories to organize local hosts. The following example adds a local host using its local IP address 192.168.0.81 and sorts it into the servers category:

```
ansible@DESKTOP-VL3E6GS: /home/abhijith
GNU nano 7.2
[local]
localhost ansible_connection=local
```

4. Save the file and exit.

5. Enter the command below to check the items in the inventory:

ansible-inventory --list -y

The output lists the hosts:

```
ansible@DESKTOP-VL3E6GS: /home/abhijith$ ansible-inventory --list -y
all:
  children:
    local:
      hosts:
        localhost:
          ansible_connection: local
ansible@DESKTOP-VL3E6GS: /home/abhijith$
```

STEP 7: Test the Connection

To ensure the Ansible control node can connect to the local hosts and run commands, use the following ansible command to ping the hosts from the control node:

sudo ansible all -m ping

Note: When a user connects to the remote hosts for the first time, Ansible asks for confirmation that the hosts are authentic. To confirm the authenticity, enter yes when prompted.

The output confirms the successful connection.

```
ansible@DESKTOP-VL3E6GS:/home/abhijith$ sudo ansible all -m ping
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ansible@DESKTOP-VL3E6GS:/home/abhijith$
```

The Ansible control node is now set up to control the connected remote hosts.

Conclusion

After following the steps in this guide, you have successfully installed Ansible on Ubuntu and can execute commands and playbooks on remote hosts. The guide provided instructions for setting up the Ansible control node and connecting it with the hosts via SSH.

PROGRAM 9:

Introduction to Azure DevOps: Overview of Azure DevOps Services, Setting Up an Azure DevOps Account and Project.

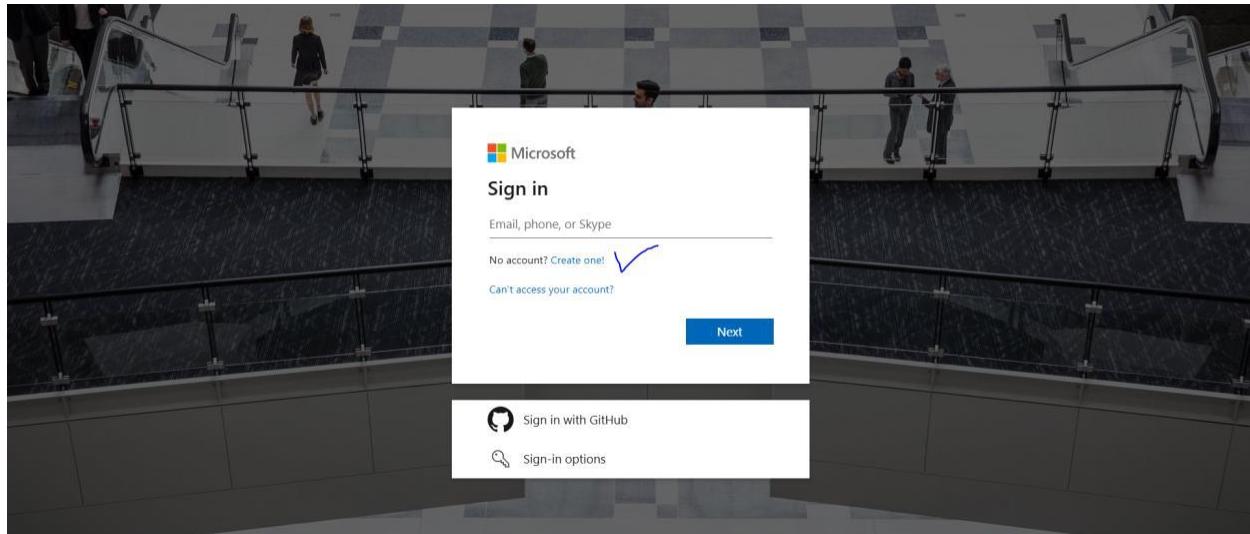
STEP1: Go to Google chrome and type **azure for students**

<https://azure.microsoft.com/en-us/free/students>

Google search results for "azure for students":

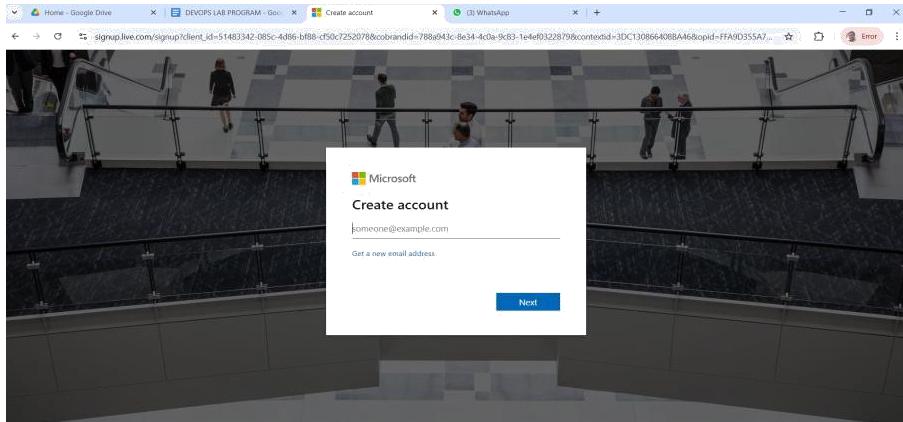
- Azure for Students – Free Account Credit**
With Microsoft Azure for Students, get a \$100 credit when you create your free account. There is no credit card needed and 12 months of free Azure services.
- Azure for College Students—Offer Details**
Students, get Azure for free courtesy of Microsoft Azure.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

STEP2: Click on **Start Free after that u get screen as in below**

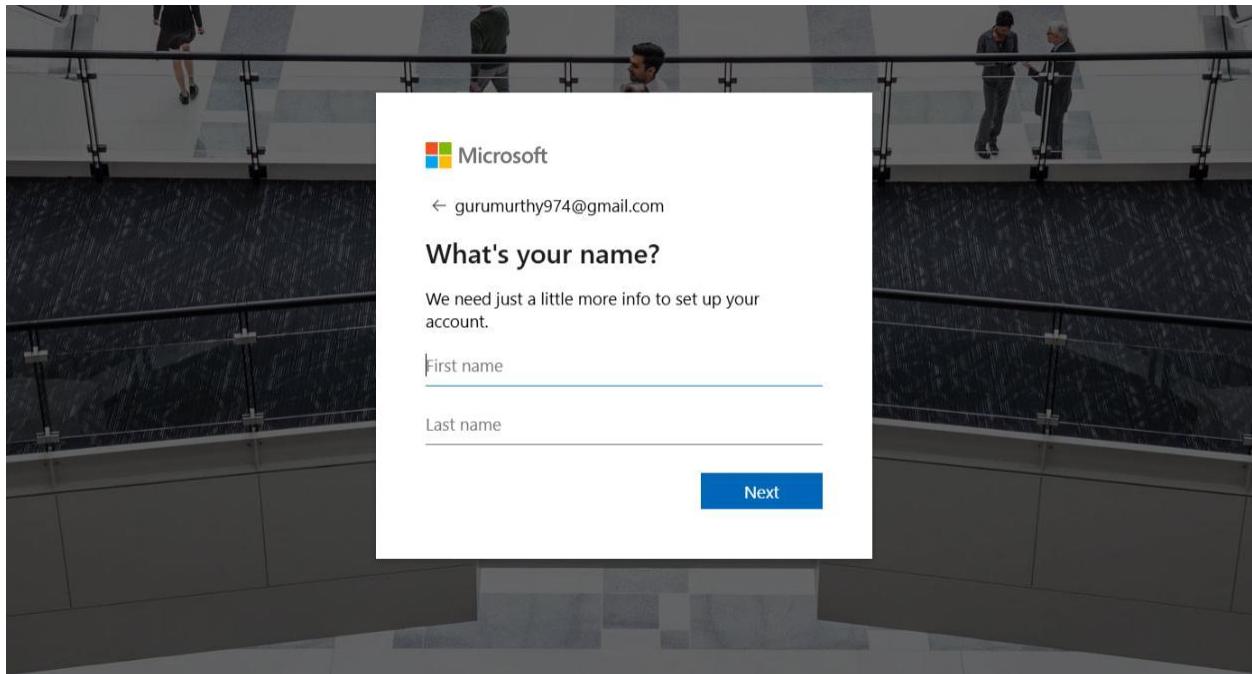


Click on Create one, If u have Github account u can Sign in using github account better way is to create one account

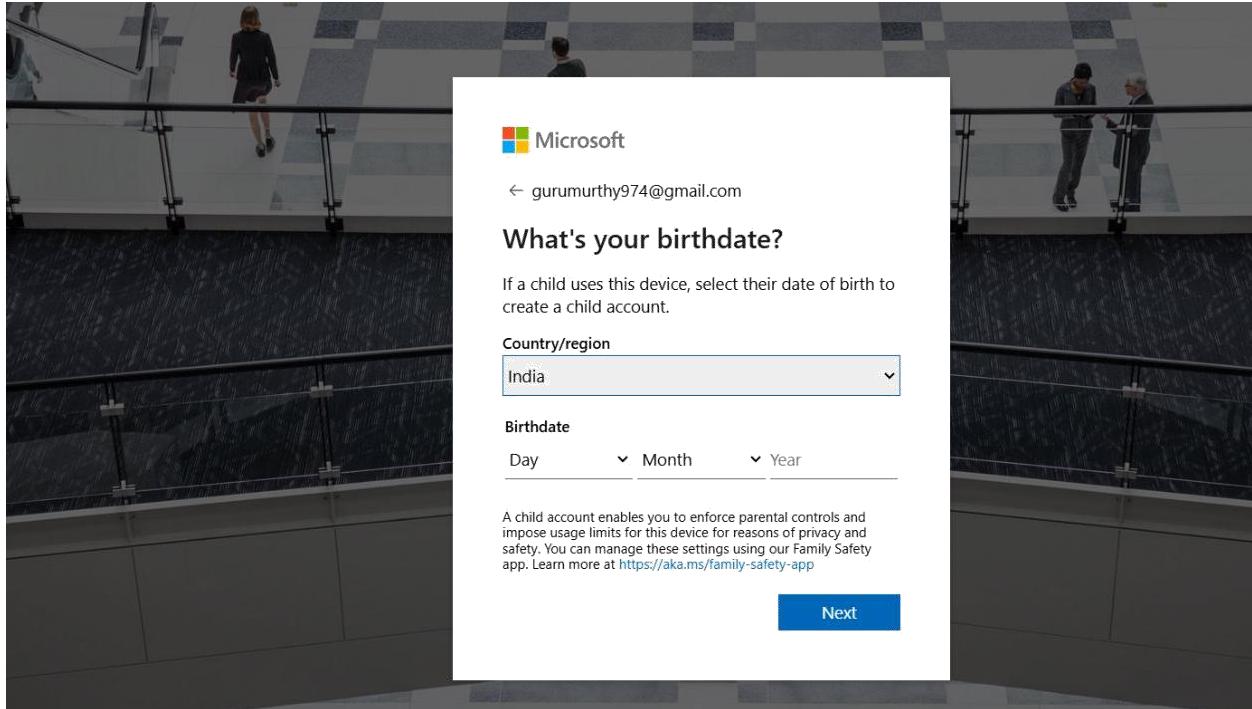
STEP3: Provide your email id at place of create account



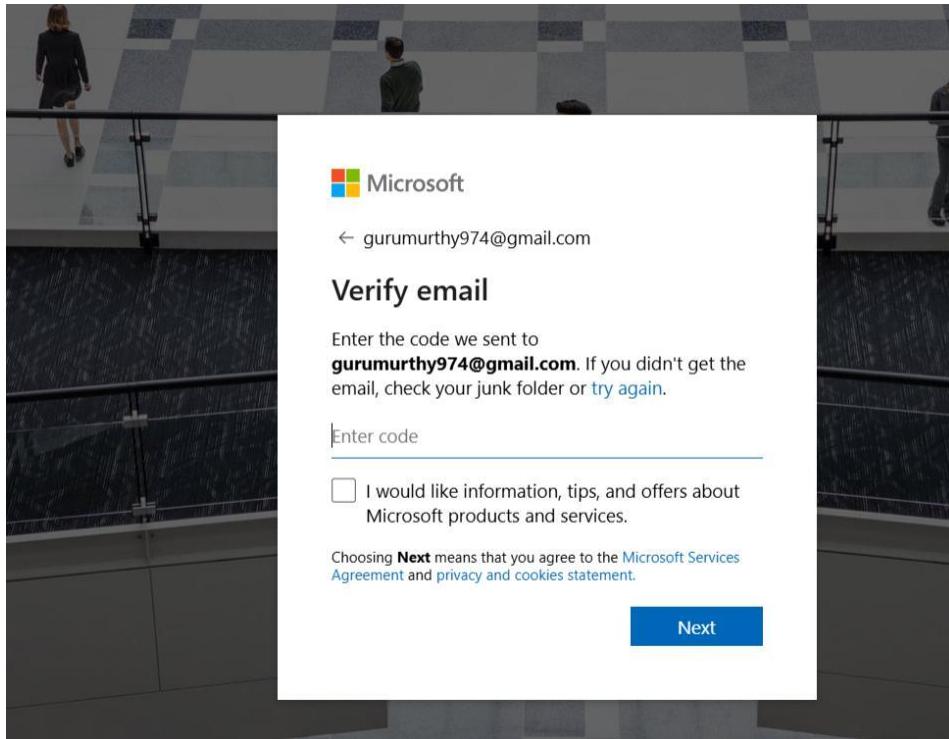
STEP4: After password is set provide your First name Last name



Then provide Country, Date of Birth



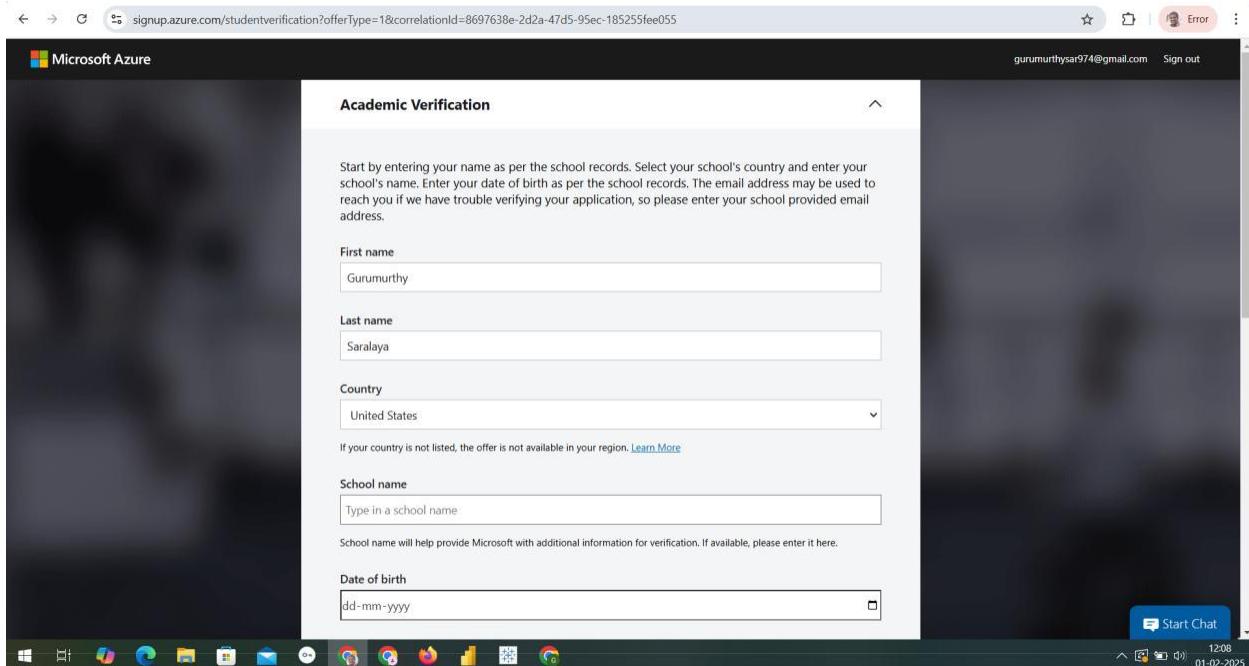
STEP5: Verification code be mailed to the mentioned once kindly type it



**STEP6:After code is verified as u got in the mail
referred U be given an option to solve puzzlegame**



STEP7:This step is the important once where u fill your Academic Details properly where u provide College email id as in provided by your Individual colleges later the verification code again comes



After proper college email is given u get verification mail with link to mail u have provided

Hello,

You have received this email because you recently requested verification via **Microsoft's Academic Verification** service. If you did not submit your email for this program, please disregard this email.

To complete your academic status verification, please click the link below. The link will automatically expire if not used within 5 days.
After clicking the link, your verification status will be confirmed and you will return to the site.

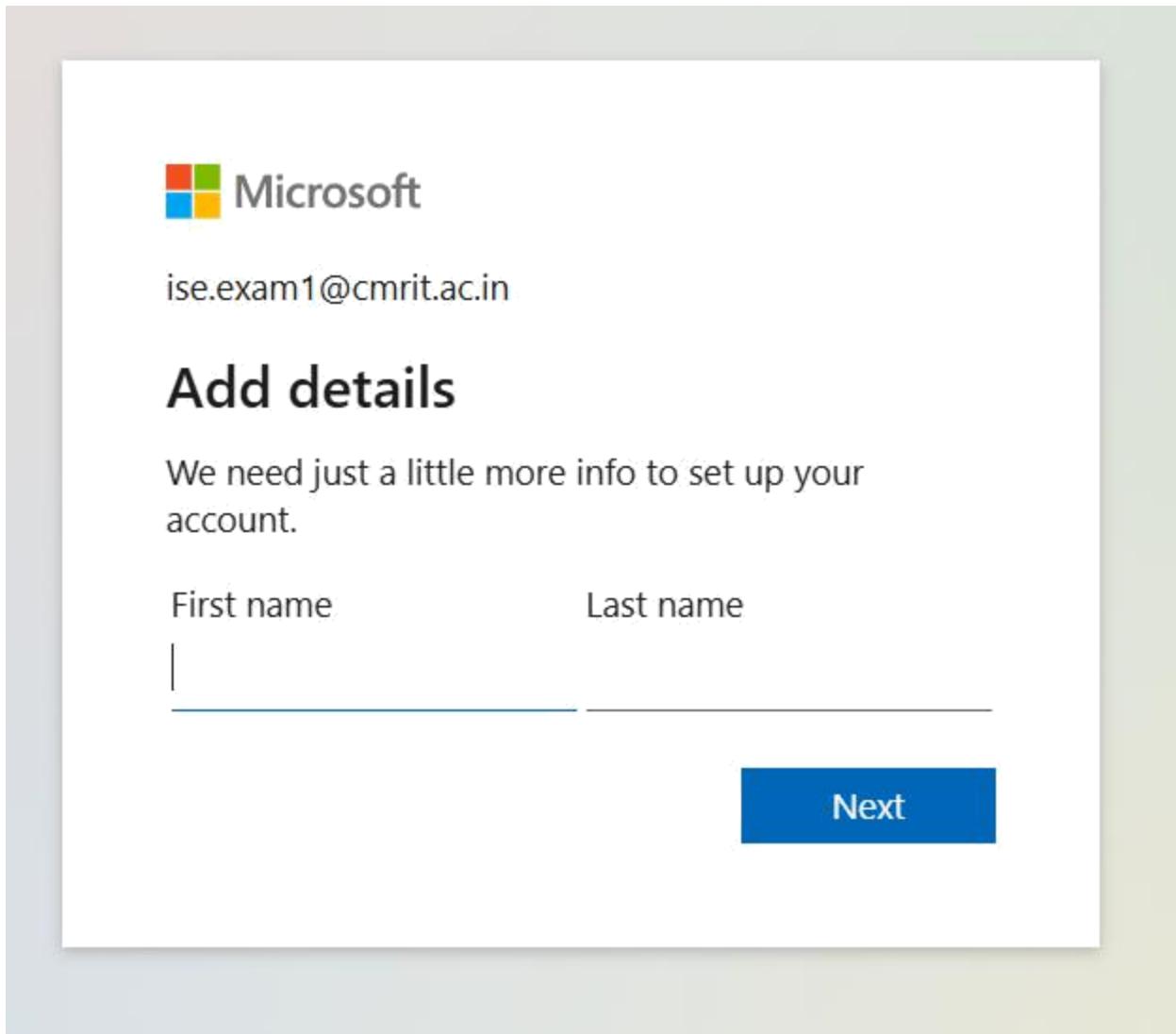
-Navigate to: <https://verifyemail.microsoft.com/v1.0/tokenverification/verify?signature=YPQIdNoTloPJ8nJy8QXZhbyR5Y5wAfwKVaNqhX5hgX6Yb0hOyqw5Aa5CsLqDGTWbHGIFSKjx37ROJFB7B0BoqKqDTCsJ3ADCIXL9McGPZs97Ms2qbP54fx0AGGTgdu7Y2SGVdpLBk98PiP33rVZf%2F2ES3z4lR8zM%2BjjuRvGxNuTK%2B4IM2tfFKm2kSsbp%2Bi754BLiZSJW4216NcWohhks8i%2Fd2qk7fgLurMrUhi8MaiPwQjLzyBmWbd9bjRY4PtX%2FfhwpOZTPGdROYFUpjyUuf6vwYojSylbtfBATxAI1se5jiz7sAL2qdHWV03x8Yhm3gjN4TGYdxzizDLE0CGCn6h8GVjyls%2BQEIO96N0Rvgi%2FnIccNbsdcmtAN0u9ivHcdpoHRnwX22Q58JgggCzyefDnDbFgxuiteaHQn%2BPkgKnOs26OHSdeVVEc%2B0MBsBAQjBTfiHeL7XEkydahHXReTh8a77XxbRAQAdXGXifW0caHpUHafaNKxiQBM28dR%2BPgZB0n3kKFMiR20ltYQtxpSkOGa6DOssh38bumS2WrSQ%3D>

Thank You,

The **Microsoft Academic Verification Team**

Click on link sent

After u Click The screen be as in Below

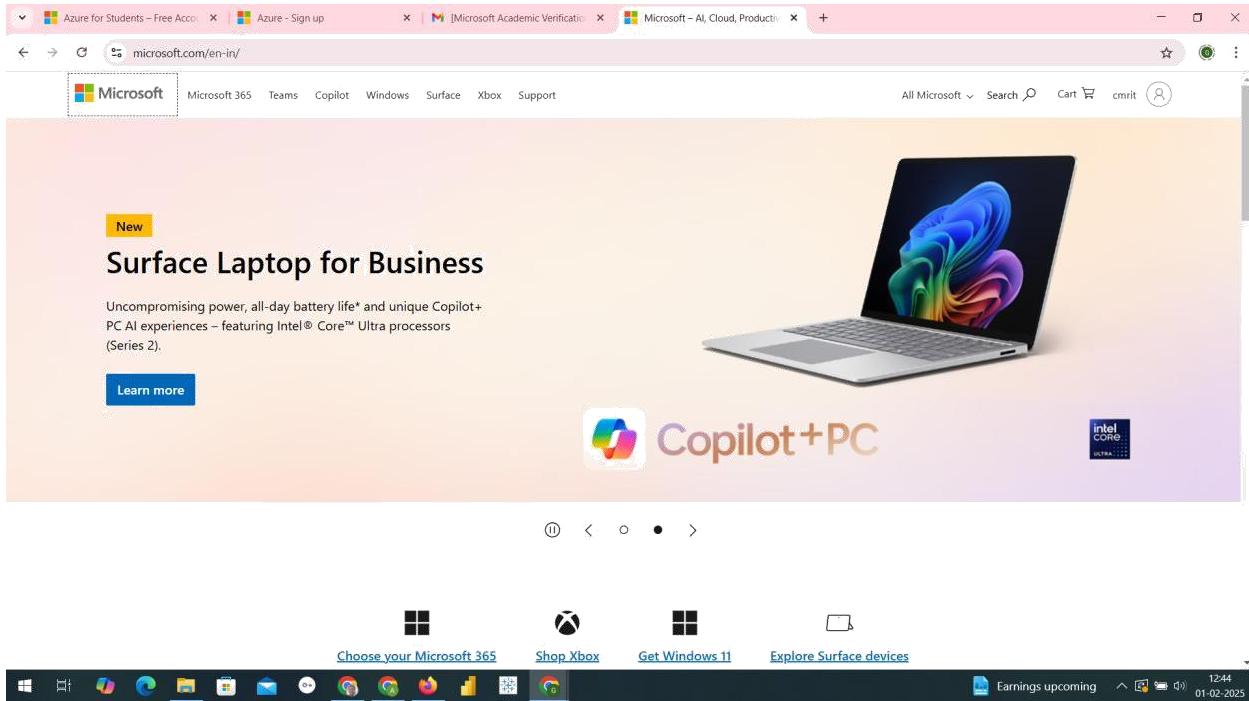


The image shows a Microsoft account setup screen. At the top left is the Microsoft logo. Below it is the email address "ise.exam1@cmrit.ac.in". The main heading is "Add details" in bold. A sub-instruction reads "We need just a little more info to set up your account." Below this, there are two input fields: "First name" with a blank input field and "Last name" with a blank input field. A large blue "Next" button is centered at the bottom.

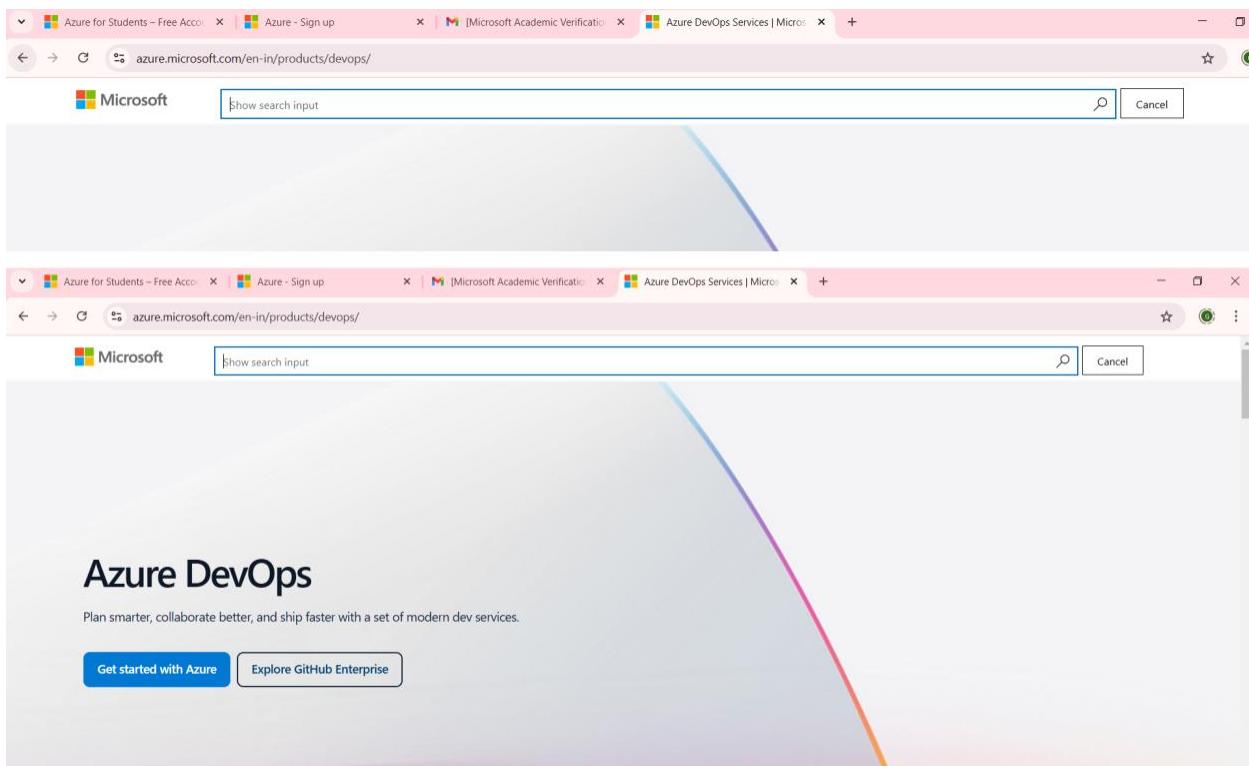
TO MAKE UR ACCOUNT SECURE IT AGAIN HAVE PUZZLE

The screenshot shows the Microsoft Academic Verification homepage. At the top, there's a blue header bar with the Microsoft logo and the text "Microsoft Academic Verification". Below the header, there are three navigation links: "Home" (underlined), "FAQ", and "Support". The main content area features a large shield icon with a checkmark inside it. Below the icon, the text "Protecting your account" is displayed. A sub-instruction reads "Please solve this puzzle so we know you are a real person". A dark blue button labeled "Start Puzzle" is centered below this text. At the bottom of the page, a small URL is visible: "67118200410c18a46.4232473504". The entire page is framed by a thick grey border.

Finally ur screen be like this



Go To search and type Azure Devops



Click on Get started with Azure

After the click u get the screen of Get free need not to do anything just click on Signin You will get screen as in below

Welcome to Azure!

Don't have a subscription? Check out the following options.

Start with an Azure free trial
Get \$200 free credit toward Azure products and services, plus 12 months of popular free services.
[Start](#)

Manage Microsoft Entra ID
Manage access, set smart policies, and enhance security with Microsoft Entra ID.
[View](#) [Learn more](#)

Access student benefits
Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.
[Explore](#) [Learn more](#)

Azure services

[Create a resource](#) [Quickstart Center](#) [Azure AI services](#) [Kubernetes services](#) [Virtual machines](#) [App Services](#) [Storage accounts](#) [SQL databases](#) [Azure Cosmos DB](#) [More services](#)

Resources

THIS IS THE HOME PAGE OF THE MICROSOFT AZURE where you can see n number of services now our target is Azure Devops

In top where have search for services

Welcome to Azure!

Don't have a subscription? Check out the following options.

Start with an Azure free trial
Get \$200 free credit toward Azure products and services, plus 12 months of popular free services.
[Start](#)

Manage Microsoft Entra ID
Manage access, set smart policies, and enhance security with Microsoft Entra ID.
[View](#) [Learn more](#)

Access student benefits
Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.
[Explore](#) [Learn more](#)

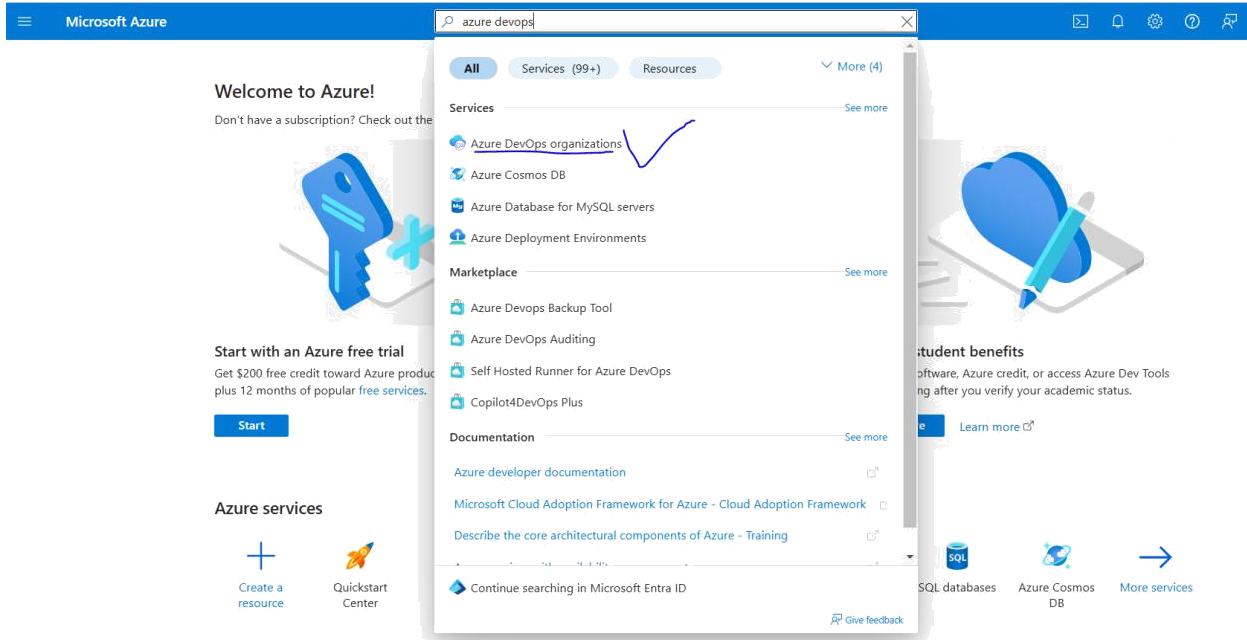
Azure services

[Create a resource](#) [Quickstart Center](#) [Azure AI services](#) [Kubernetes services](#) [Virtual machines](#) [App Services](#) [Storage accounts](#) [SQL databases](#) [Azure Cosmos DB](#) [More services](#)

Resources

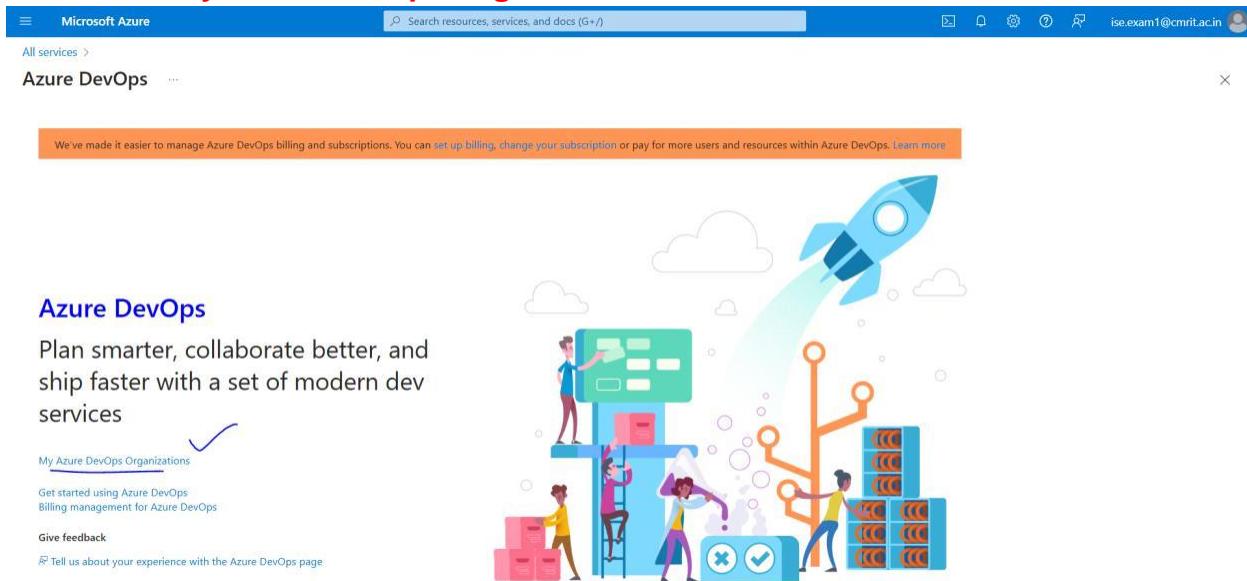
Type Azure Devops

STEP8:Select Azure Devops Organization



STEP9:

**After u opting for Azure Devops Organizations u get a screen as in below
now select **My Azure DevOps Organizations****



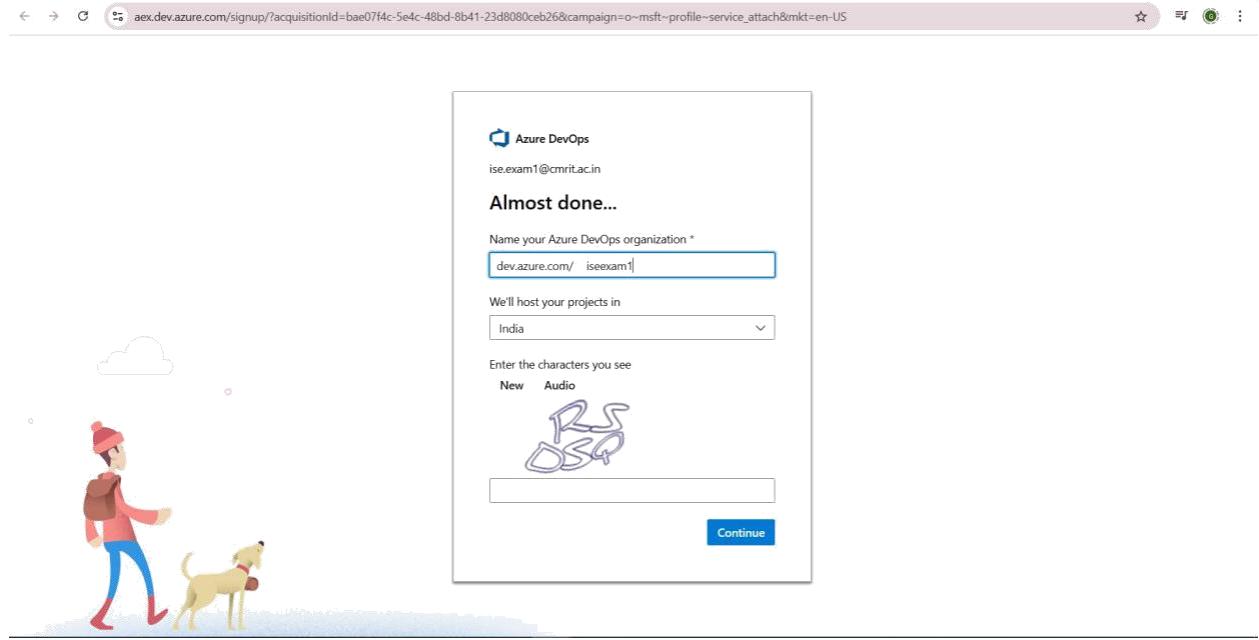
**After above selection it once again reverifies name and email just click
Continue After it U get a Screen**



Get started with Azure DevOps

Plan better, code together, ship faster with Azure DevOps

[Create new organization](#)



You will be able to see Organization is Created

Azure DevOps Organizations

[Create new organization](#)

✓ [dev.azure.com/iseexam10557](#) (Owner)

Create a Team Project and start collaborating with your team now!

New project



Actions

[Open in Visual Studio](#)

Finally After Creating a New Organization

U can create Project of ur choice as per requirement

Every time u need not to signin u can bookmark or add the below link as shortcut

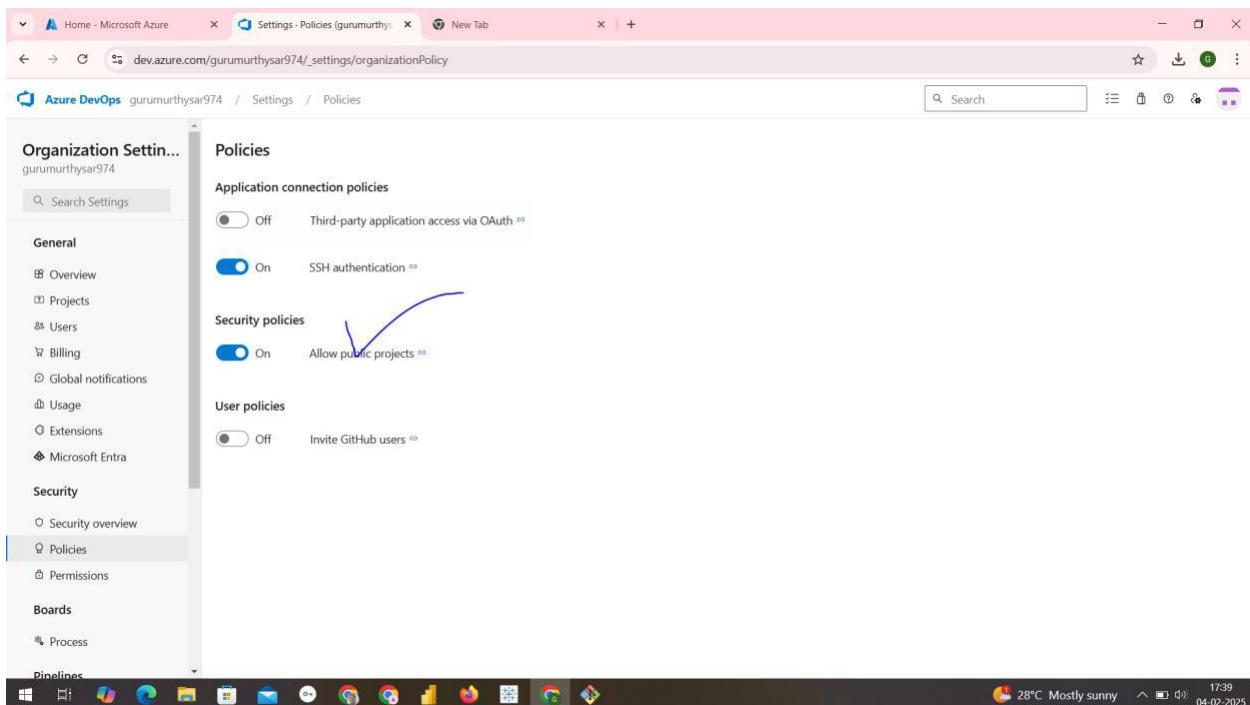
<https://aex.dev.azure.com/>

<https://portal.azure.com/#home>

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.

STEP1:On creating organization goto Organization settings goto Policy And Allow Public Projects active

**STEP2: GOTO GITBASH**

TYPE COMMANDS AS IN BELOW

mkdir maventest1

cd maventest1

STEP3: to create simple hellow world maven project type command as in

below mvn archetype:generate -DgroupId=com.dineshonjava -

DartifactId=Javateam -DarchetypeArtifactId=maven-archetype-quickstart -

DinteractiveMode=false

```

User@DESKTOP-VL3E6GS MINGW64 ~/maventest1 (main)
$ mvn archetype:generate -DgroupId=com.dineshonjava -DartifactId=JavaHelloWorld -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
[INFO] Scanning for projects...
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/3.1.2/maven-install-plugin-3.1.2.pom (8.5 kB at 9.8 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/3.1.2/maven-install-plugin-3.1.2.pom (8.5 kB at 9.8 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/42/maven-plugins-42.pom (7.7 kB at 60 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/42/maven-plugins-42.pom (7.7 kB at 60 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/3.1.2/maven-install-plugin-3.1.2.jar (32 kB at 360 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-deploy-plugin/3.1.2/maven-deploy-plugin-3.1.2.pom (9.6 kB at 171 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-deploy-plugin/3.1.2/maven-deploy-plugin-3.1.2.pom (9.6 kB at 171 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-deploy-plugin/3.1.2/maven-deploy-plugin-3.1.2.jar (40 kB at 964 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-assembly-plugin/3.7.1/maven-assembly-plugin-3.7.1.pom (15 kB at 524 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/41/maven-plugins-41.pom
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/41/maven-plugins-41.pom (7.4 kB at 104 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-assembly-plugin/3.7.1/maven-assembly-plugin-3.7.1.jar (240 kB at 1.6 MB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-dependency-plugin/3.7.0/maven-dependency-plugin-3.7.0.pom (19 kB at 301 kB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-dependency-plugin/3.7.0/maven-dependency-plugin-3.7.0.jar (207 kB at 4.2 MB/s)
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-dependency-plugin/3.7.0/maven-dependency-plugin-3.7.0.jar (207 kB at 4.2 MB/s)

[INFO] [INFO] -----> org.apache.maven:standalone-pom <-----
[INFO] [INFO] Building Maven Stub Project (No POM) 1
[INFO] [INFO] [ pom ]
[INFO] [INFO] >>> archetype:3.3.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO] [INFO] <<< archetype:3.3.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO] [INFO]
[INFO] [INFO] -----> archetype:3.3.1:generate (default-cli) @ standalone-pom <-----
[INFO] [INFO] Generating project in batch mode...
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar
Downloaded From central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (4.3 kB at 102 kB/s)
[INFO] [INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] [INFO] -----
[INFO] Parameter: basedir, value: C:\Users\User\maventest1
[INFO] Parameter: package, value: com.dineshonjava
[INFO] Parameter: groupId, value: com.dineshonjava
[INFO] Parameter: artifactId, value: JavaHelloWorld
[INFO] Parameter: packageName, value: com.dineshonjava
[INFO] Parameter: version, value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\User\maventest1\JavaHelloWorld
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.691 s
[INFO] Finished at: 2025-02-04T17:58:22+05:30
[INFO] -----
```

STEP3: to add files from local to github Follow the procedure

- First create a repository in github as maventestazure
- Then come to gitbash and type

git init

git add .

git commit -m "azure pipeline example"

git branch -M main

git remote add origin <https://github.com/gurumurthy974/maventestazure.git>

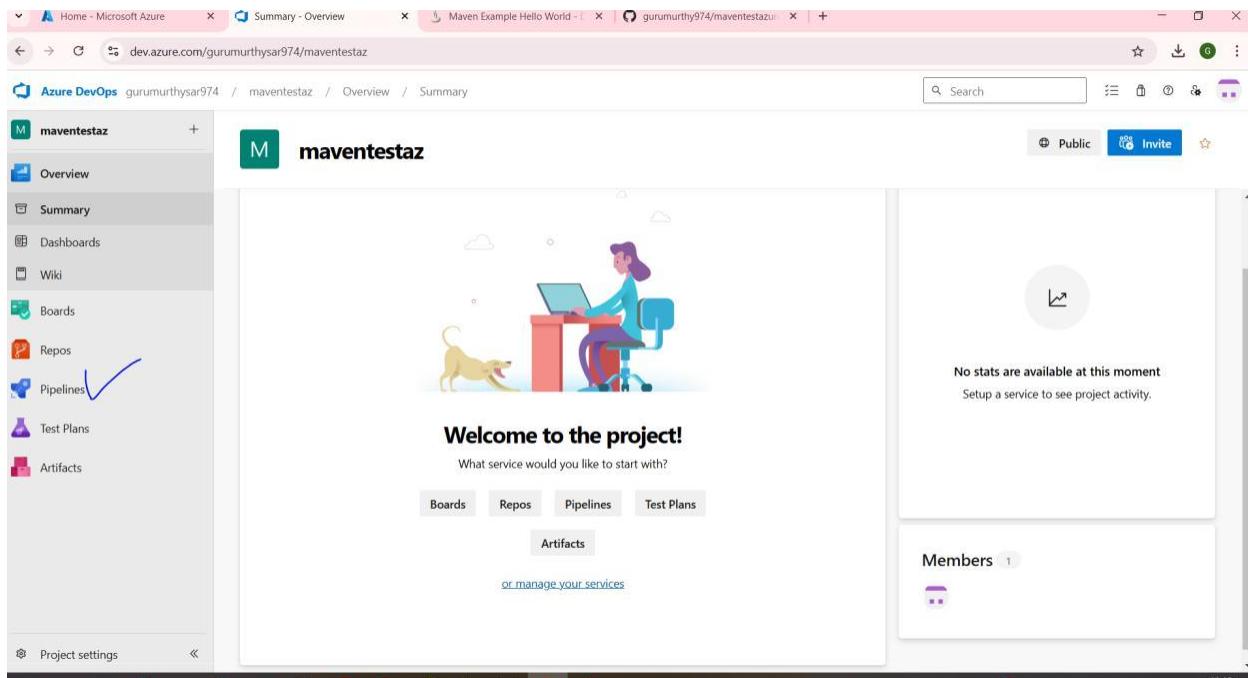
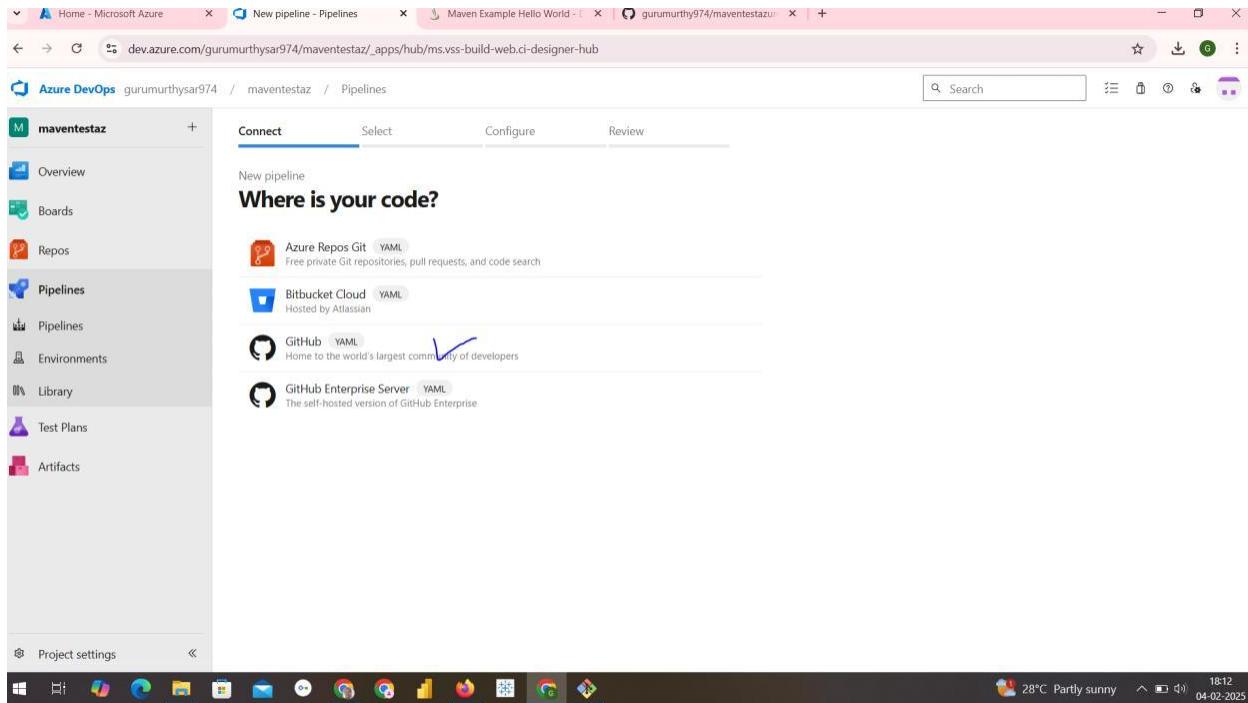
git push -u origin main

After completion of above command my repository looks

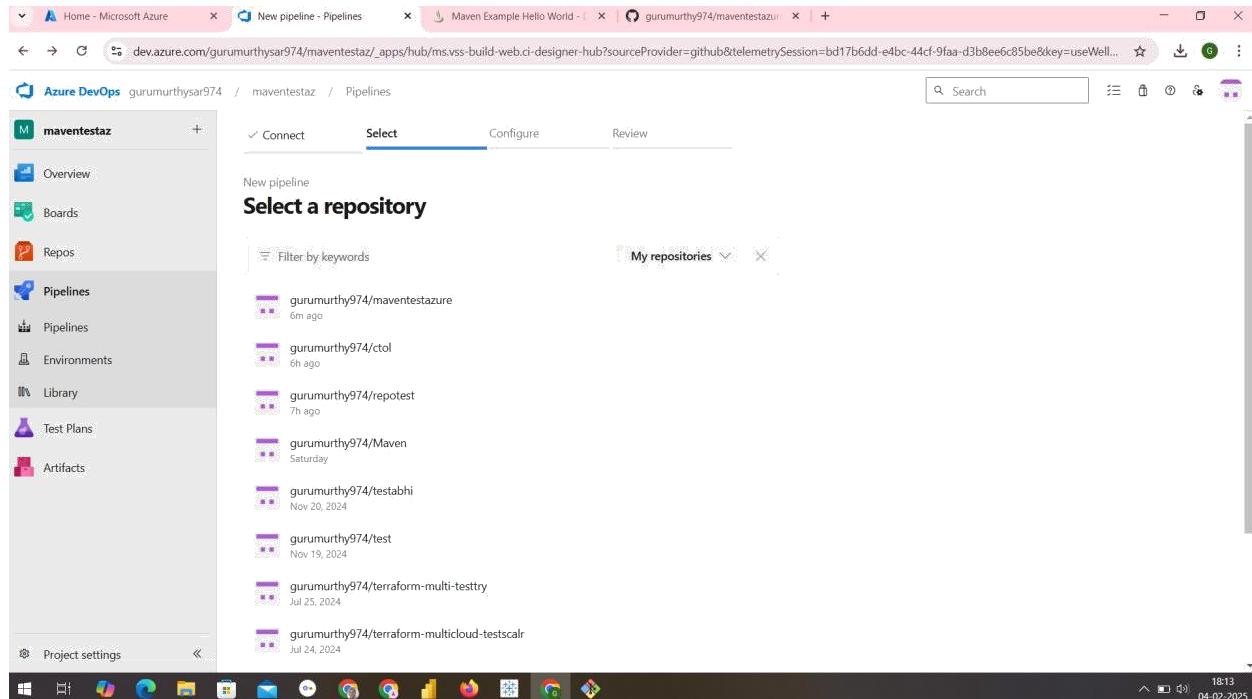
The screenshot shows a GitHub repository named 'maventestazure'. It has 1 branch and 0 tags. A commit from 'gurumurthy974' titled 'first' is listed, with a checkmark next to it. The repository is public and has 1 commit. The README file section is present. On the right side, there are sections for About, Activity, Releases, Packages, and Languages, all of which are currently empty or have no data.

STEP4: Now goto Azure Devops Organization create Public Project

The screenshot shows the 'Create a project to get started' interface in Azure DevOps. The project name is 'maventestaz'. The visibility is set to 'Public', indicated by a checkmark. The 'Advanced' button is visible below the visibility options. By creating this project, you agree to the Azure DevOps code of conduct.

STEP5: SELECT PIPELINE and then click on create pipeline**STEP6: After creating Pipeline select type of repo as Github**

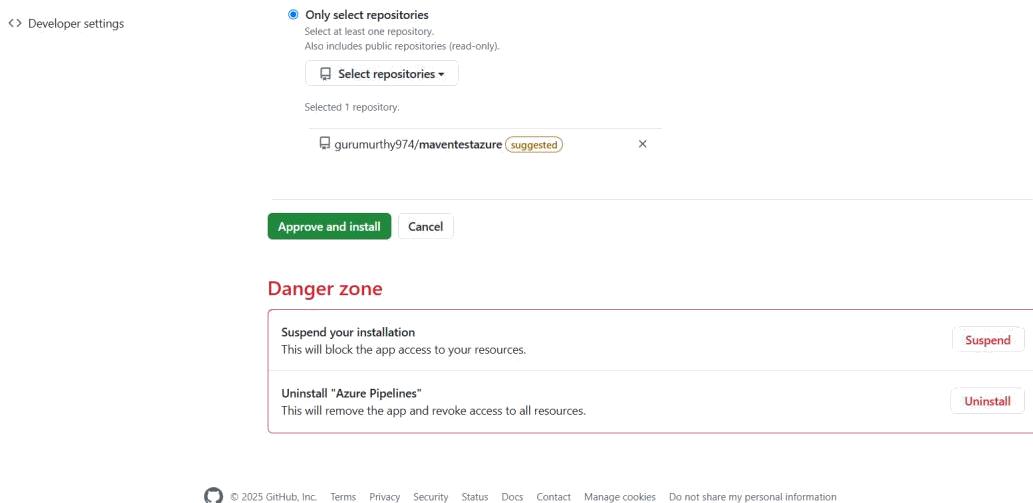
STEP7: It asks for minimum signin verification after that ur screen be as in below select required repository there to run maven project in my case its **maventest123**



STEP8: AFTER REQUIRED REPO IS SELECTED the screen be as in below

The screenshot shows the Azure Pipelines landing page for the 'maventestaz' repository. The left sidebar includes sections for Public profile, Account, Appearance, Accessibility, Notifications, Access (Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys, Organizations, Enterprises, Moderation), Code, planning, and automation (Repositories, Codespaces, Packages, Copilot, Pages), and Permissions (Read access to metadata). The main content area is titled 'Azure Pipelines' and describes it as 'Continuously build, test, and deploy to any platform and cloud'. It highlights features like Any language, platform, and cloud; Native container support; Advanced workflows and features; Extensible; and Free, to you from Azure Pipelines. The URL https://azure.microsoft.com/services/devops/pipelines/ is provided.

Drag the screen down check once again the selected repository is correct or not then click on Approve and Install



STEP9: It again verifies signin verification of microsoft account

You be able to see starter pipeline select for Maven

The screenshot shows the Azure DevOps Pipelines configuration page for a Maven project named 'maventestaz'. The left sidebar lists 'Overview', 'Boards', 'Repos', 'Pipelines' (selected), 'Environments', 'Library', 'Test Plans', and 'Artifacts'. The main area is titled 'Configure your pipeline' and shows four pipeline templates: 'Maven' (selected, indicated by a blue checkmark), 'Maven package Java project Web App to Linux on Azure', 'Starter pipeline', and 'Existing Azure Pipelines YAML file'. The 'Configure' tab is active. The URL in the browser is https://dev.azure.com/gurumurthysar974/maventestaz/_apps/hub/ms.vss-build-web.ci-designer-hub?triggers=ContinuousIntegration%2CPullRequest&connectionId=fd4da095-7394-4e4c-ae7c-563c1c2...

After selecting maven it asks for save and run just click on it

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with options like Overview, Boards, Repos, Pipelines (selected), Environments, Library, Test Plans, and Artifacts. The main area is titled "Review your pipeline YAML" and shows the following YAML code:

```

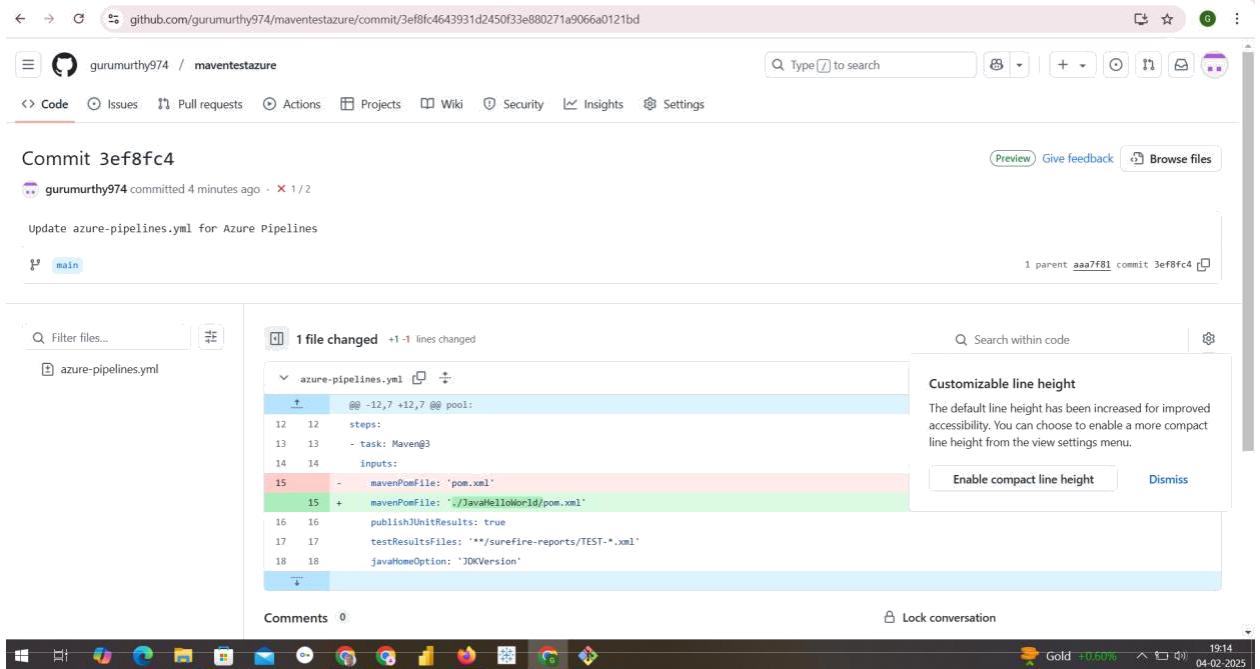
1   # Maven
2   # Build your Java project and run tests with Apache Maven.
3   # Add steps that analyze code, save build artifacts, deploy, and more:
4   # https://docs.microsoft.com/azure/devops/pipelines/languages/java
5
6   trigger:
7     - main
8
9   pool:
10    - vmImage: ubuntu-latest
11
12   steps:
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 buttons for "Variables", "Save and run", and "Show assistant".

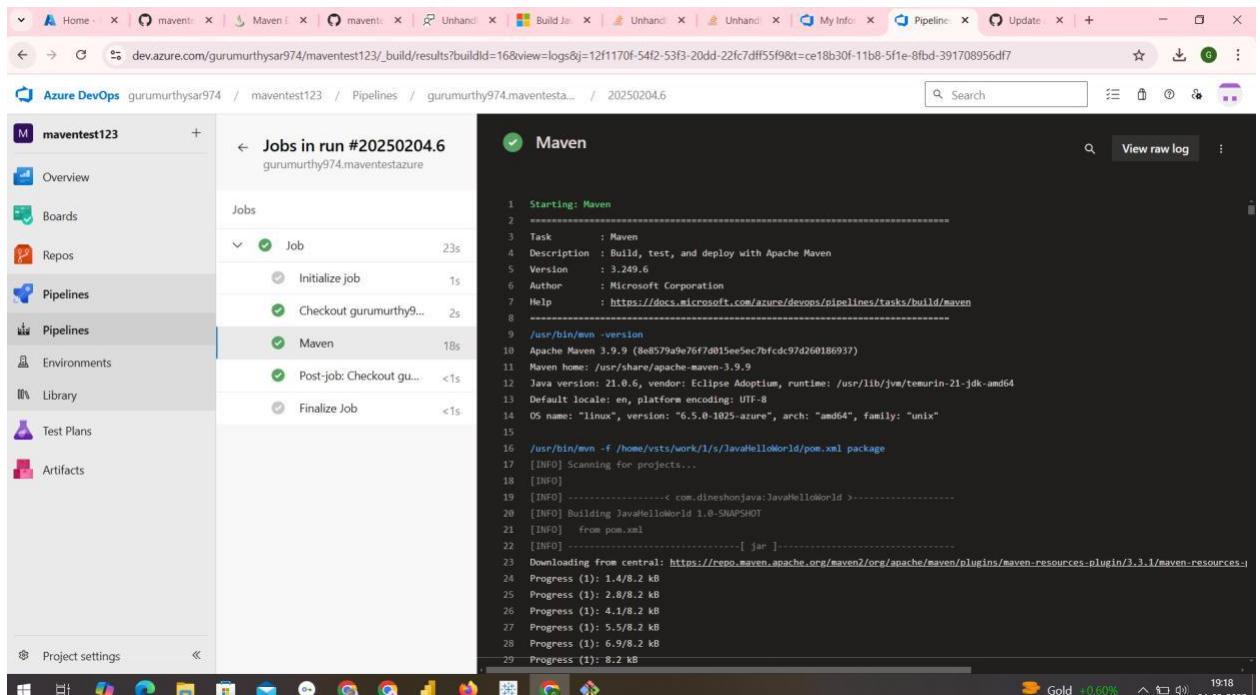
Finally You be able to see tasks running its failed bec we shld mention proper path For pom.xml

The screenshot shows the build summary page. It includes sections for "Manually run by" (Gurumurthy Saralaya), "Repository and version" (gurumurthy974/maventestazure, main commit f6a6b586), "Time started and elapsed" (Just now, 30s), "Related" (0 work items, 0 artifacts), and "Tests and coverage" (Get started). At the bottom, there are links for "Errors 2" and "Warnings 2".



The commits will also be visible in github

WE can download and also see individual Raw Log Reports



If pipeline permission error comes please complete the registration form of Parallel Jobs after 2 working days (48hrs) u be able to run pipelines for private projects same happens to be for public Projects.

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 Pipelines.

STEP1: Click on Organization setting and click on Pipeline Settings You get screen as in below

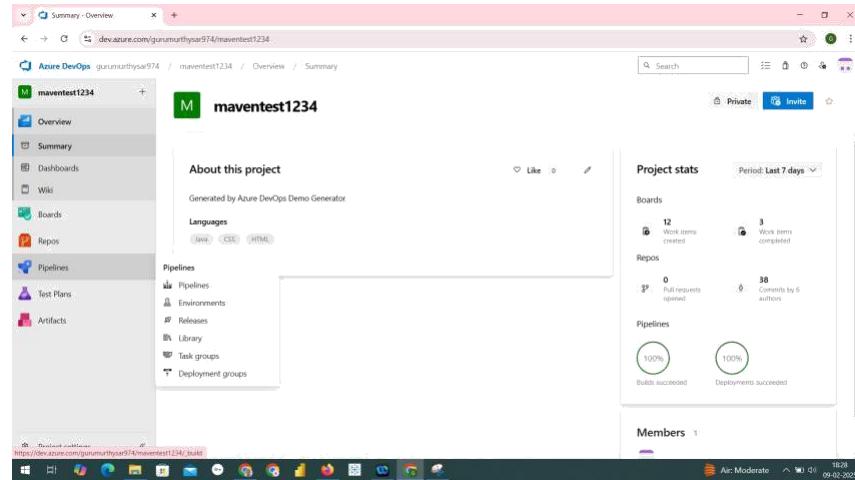
The screenshot shows the 'Organization Settings' page for the 'gurumurthysar974' organization. The left sidebar lists sections like General, Security, Boards, and Pipelines. Under Pipelines, the 'Disable stage chooser' option is highlighted with a blue oval. The main content area displays several pipeline settings with toggle switches:

- Disable anonymous access to badges**: On
- Limit variables that can be set at queue time**: On
- Limit job authorization scope to current project for non-release pipelines**: On
- Limit job authorization scope to current project for release pipelines**: On
- Protect access to repositories in YAML pipelines**: On
- Disable stage chooser**: Off (highlighted)

STEP2: Off the Disable creation of classic pipeline

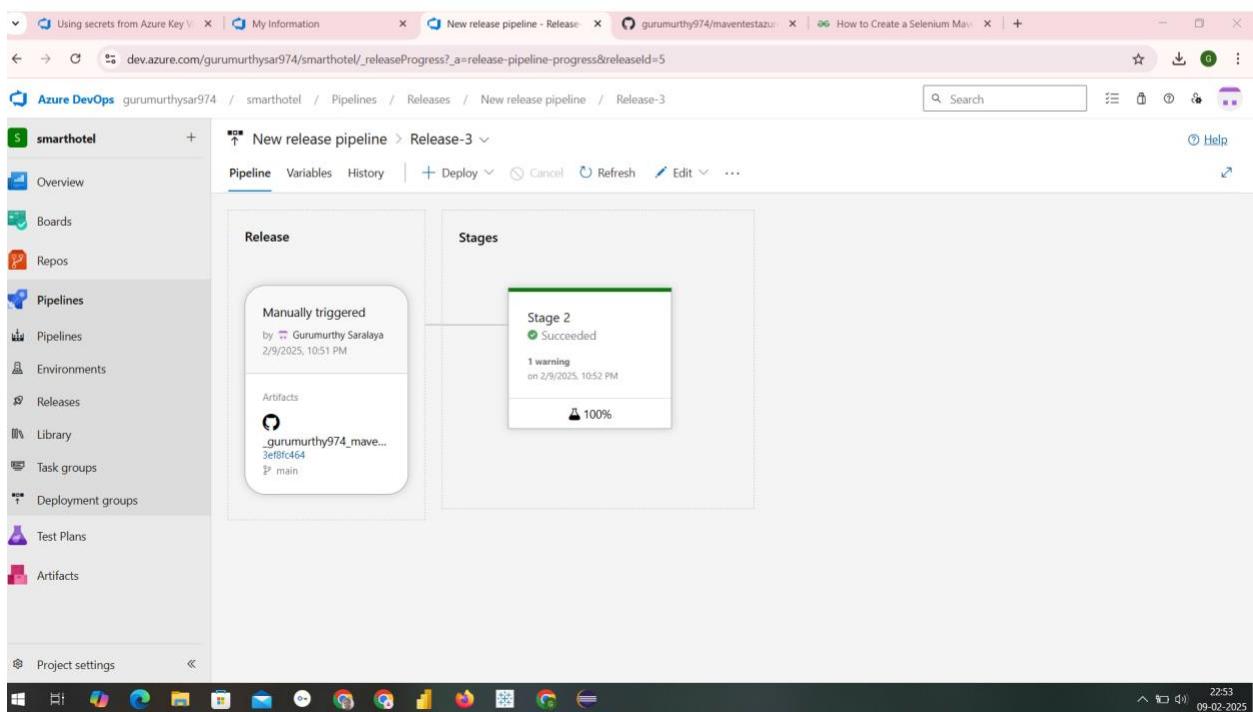
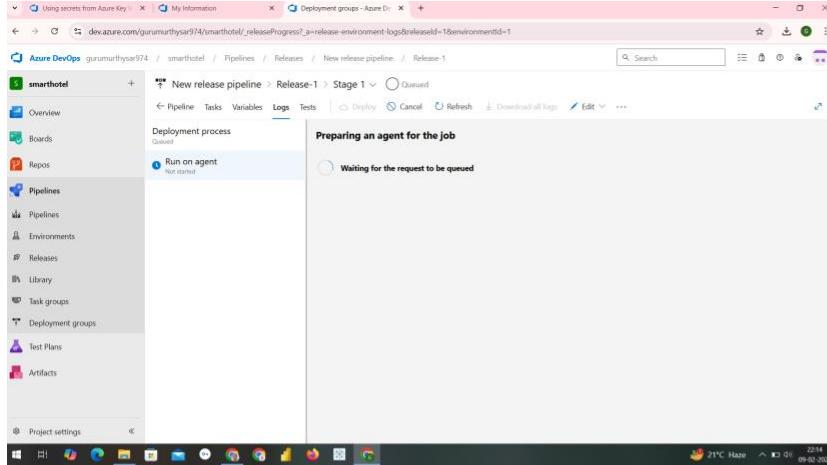
This screenshot shows the same 'Organization Settings' page as above, but with a different focus. The 'Disable creation of classic build pipelines' option is highlighted with a blue oval. The toggle switch for this option is currently set to 'Off'. Other pipeline settings shown are all 'On': Limit job authorization scope to current project for release pipelines, Protect access to repositories in YAML pipelines, and Disable stage chooser.

STEP3: Now you be able to see the visibility of Release for any pipeline creation as in screen below.



STEP4: You can run simple test plans

We can build tasks and run them



We can see our Agent Job releases logs

This screenshot shows the 'Logs' tab for 'Stage 2' of the 'New release pipeline' under 'Release-3'. The 'Agent job' section shows the log output for the 'Hosted Windows 2019' pool. The log details the execution of tasks: 'Initialize job - succeeded', 'Download Artifacts - succeeded', 'Maven D:\a\1\a_gurumurthy974_maventestazure\javaHelloWorld\pom.xml - succeeded 1 warning', and 'Finalize Job - succeeded'. The total duration was 44s. The log was started at 2/9/2025, 10:52:10 PM.

Task	Status	Duration
Initialize job	succeeded	7s
Download Artifacts	succeeded	4s
Maven D:\a\1\a_gurumurthy974_maventestazure\javaHelloWorld\pom.xml	succeeded 1 warning	32s
Finalize Job	succeeded	<1s

VIVA Question & Answers

1. Introduction to Maven and Gradle

Q1: What is Maven, and what is it used for?

A: Maven is a build automation tool used primarily for Java projects. It simplifies project management, dependency management, and build processes using a standardized approach through a Project Object Model (POM).

Q2: How does Gradle differ from Maven?

A: The key differences are:

- **Maven** uses XML-based POM files, while **Gradle** uses a Groovy/Kotlin DSL.
- Gradle is faster due to incremental builds and caching mechanisms.
- Gradle allows more flexibility in defining tasks compared to Maven's rigid lifecycle.

Q3: How do you install Maven and Gradle?

A:

- **Maven:** Download from Apache Maven's official website, set up MAVEN_HOME environment variable, and add bin directory to the system path.
- **Gradle:** Download from the Gradle website or use a package manager (e.g., sdk install gradle via SDKMAN).

2. Working with Maven

Q4: What is the POM file in Maven?

A: The POM (Project Object Model) file is an XML configuration file that defines a Maven project's dependencies, build configuration, and plugins.

Q5: How does Maven manage dependencies?

A: Maven uses a central repository (mvnrepository.com) and a local repository to resolve and cache dependencies. It fetches dependencies based on their groupId, artifactId, and version.

Q6: What are Maven plugins?

A: Plugins extend Maven's functionality by adding specific tasks like compiling code (maven-compiler-plugin), running tests (surefire-plugin), and packaging applications (maven-jar-plugin).

3. Working with Gradle**Q7: What are the two types of Gradle build scripts?**

A:

1. **Groovy DSL (build.gradle)** – Uses Groovy scripting for configuration.
2. **Kotlin DSL (build.gradle.kts)** – Uses Kotlin, providing better type safety and autocompletion.

Q8: How does Gradle handle dependency management?

A: Gradle fetches dependencies from repositories (e.g., Maven Central, JCenter) and uses a dependencies block in build.gradle to define them.

Q9: What is task automation in Gradle?

A: Gradle allows users to define tasks (task myTask { doLast { println "Hello, Gradle!" }}) and automate builds, testing, and deployments efficiently.

4. Jenkins and Continuous Integration**Q10: What is Jenkins?**

A: Jenkins is an open-source automation server used for Continuous Integration/Continuous Deployment (CI/CD). It automates building, testing, and deploying applications.

Q11: How do you install Jenkins?

A: Jenkins can be installed using a .war file (java -jar jenkins.war), as a Docker container, or on cloud environments like AWS, Azure, or Kubernetes.

Q12: What is a Jenkins pipeline?

A: A Jenkins pipeline is a series of automated steps (written in a Jenkinsfile) that define the CI/CD workflow, including building, testing, and deploying code.

Q13: How do you integrate Jenkins with Maven or Gradle?

A: In a Jenkins job, specify Maven or Gradle as a build step and configure the path to the corresponding build file (pom.xml or build.gradle).

5. Configuration Management with Ansible**Q14: What is Ansible?**

A: Ansible is an open-source configuration management and automation tool that uses YAML-based playbooks to manage server configurations and deployments.

Q15: What are the key components of Ansible?

A:

- **Inventory:** A list of managed nodes.
- **Playbooks:** YAML files containing automation tasks.
- **Modules:** Pre-built scripts that perform tasks (e.g., apt, yum, copy).

Q16: How do you write a basic Ansible playbook?

A:

```
- name: Install Apache
  hosts: web_servers
  tasks:
    - name: Install Apache
      apt:
```

```
name: apache2
state: present
```

Q17: How does Ansible differ from other configuration management tools?

A: Ansible is agentless (no need to install software on remote machines) and uses SSH for communication, unlike Puppet or Chef, which require agents.

6. Azure DevOps and Pipelines

Q18: What is Azure DevOps?

A: Azure DevOps is a cloud-based DevOps platform that provides tools for source control, CI/CD pipelines, testing, and deployment.

Q19: How do you set up an Azure DevOps project?

A:

1. Create an Azure DevOps account.
2. Create a new project and repository.
3. Set up a build pipeline using Azure Pipelines.

Q20: What is an Azure Pipeline?

A: Azure Pipelines automate the building, testing, and deployment of applications using YAML or classic GUI-based workflows.

Q21: How do you deploy an application using Azure Pipelines?

A:

1. Define a **build pipeline** to compile and package the application.
2. Define a **release pipeline** to deploy the application to Azure services like App Service or Kubernetes.

7. Hands-on and Best Practices

Q22: How do you migrate a Maven project to Gradle?

A: Use the command:

```
gradle init --type pom
```

This converts pom.xml to build.gradle. Then, manually adjust dependencies and plugins.

Q23: What are best practices for DevOps pipelines?

A:

- Use **Infrastructure as Code (IaC)** (e.g., Ansible, Terraform).
- Implement **automated testing** in CI/CD pipelines.
- Follow **version control** best practices (e.g., Git branching strategies).
- Use **monitoring and logging** for proactive issue detection.

Q24: How does Jenkins work with Ansible for deployment?

A: Jenkins triggers an Ansible playbook to configure servers and deploy artifacts after a successful build.