**2. Working with Maven**

**Creating a Maven Project, Understanding the POM File, Dependency Management and Plugins.**

**Creating Maven Project - Using Command:**

* Maven is used for managing Java-based projects particularly build automation, dependency management, and project configuration
* Open CMD 🡪 mvn archetype:generate -DgroupId=com.mvncmd.example -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
* Key things to understand
  + groupId: A unique identifier for the group (usually the domain name).
  + artifactId: A unique name for the project artifact (your project) (Mandatory).
  + archetypeArtifactId: The template you want to use for the project.

When you provide the archetypeArtifactId you are informing maven what archetype to use to create the initial structure of the project.

**Commands used to Build and Run Projects**

* + **mvn compile :** This will compile the source code.
  + **mvn test :** This will run unit tests
  + **mvn package :** Package the project into a JAR
  + **Run Application :** java -cp target/myapp-1.0-SNAPSHOT.jar com.example.App

**Key element in pom.xml:**

* **<groupId>: The group or organization that the project belongs to.**
* **<artifactId>: The name of the project or artifact.**
* **<version>: The version of the project (often follows a format like 1.0-SNAPSHOT).**
* **<packaging>: Type of artifact, e.g., jar, war, pom, etc.**
* **<dependencies>: A list of dependencies the project requires.**
* **<build>: Specifies the build settings, such as plugins to use**

**Creating Maven Project – Using IDE:**

* Open Visual Studio Code.
* Navigate to the Extension ([https://code.visualstudio.com/docs/java/java-project#\_generate-project-from-maven-archetype](https://code.visualstudio.com/docs/java/java-project))
* Click on Install the Extension Pack for Java
* Once installed 🡪 Select the option mentioned in the below Screen shot to Get Started with the Java Project and click on “Mark Done”
* Click Ctrl+Shift+P to search and run command. Type Maven to see all maven related operations.
* Select Maven: New Project 🡪 No Archetype 🡪 Enter groupId and artifactId (click Enter to confirm the name)
* Select the Directory to which you have to save the Maven File
* Once selected the Maven file will be created

**Dependency Management**

Maven uses the <dependencies> tag in the pom.xml to manage external libraries or dependencies that your project needs

**Types of Dependencies**

* Direct dependencies: These dependencies are explicitly included in a pom.xml file in the <dependencies> section.
* Transitive dependencies: Maven automatically resolves transitive dependencies, if you add a library that depends on other libraries, Maven will also download those.

**Different Dependencies Scopes**

* compile (default): Available in all build phases.
* provided: Available during compilation but not at runtime (e.g., a web server container).
* runtime: Needed only at runtime, not during compilation.
* test: Required only for testing.

**Plugins**

* Maven plugins are used to perform tasks during the build lifecycle, such as compiling code, running tests, packaging, and deploying.
* You can specify plugins within the <build> section of your pom.xml.
* Some of the Common Plugins
  + maven-compiler-plugin: Compiles Java code.
  + maven-surefire-plugin: Runs unit tests.
  + maven-jar-plugin: Packages the project as a JAR file.
  + maven-clean-plugin: Cleans up the target/ directory.
  + maven-javadoc-plugin: It is used for Generates Javadoc documentation for the project.
  + URL: https://maven.apache.org/plugins/

**PROGRAM-3:**

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

* Gradle is a build automation tool used to compile, test, and package applications—mainly Java-based projects, though it also supports other languages like Kotlin, Groovy, Scala, and even C/C++

**What is Gradle used for?**

* **Building Java Projects** (compiling code, running tests, generating JAR/WAR files) ;
* **Managing Dependencies** from repositories like Maven Central ;
* **Automating Repetitive Tasks** such as testing, code analysis, and deployment ;
* **Creating Custom Build Logic** using Groovy or Kotlin scripts ;
* **Integrating with CI/CD Tools** like Jenkins, Azure DevOps, GitHub Actions

Gradle uses a DSL (Domain-Specific Language) to define the build scripts. Gradle supports two DSLs:

* **Groovy DSL** (default)
* **Kotlin DSL** (alternative)
* Groovy DSL: This is the default language used for Gradle build scripts (build.gradle)
* Kotlin DSL: Gradle also supports Kotlin for its build scripts (build.gradle.kts)

**Create Project – Command:**

* Use command -gradle –type java-application
* Use command – gradle run 🡪 to run / execute the project
* Use command-gradle build-to build the project.

**Difference Between Gradle and Kotlin**

* **Syntax**: Groovy uses a more concise, dynamic syntax, while Kotlin offers a more structured, statically-typed approach.
* **Error handling**: Kotlin provides better error detection at compile time due to its static nature.

Run Custom Task:

Use Command – gradle <task-name> 🡪 to execute the custom task

**Program-4**

**Build and Run a Java Application with Maven, Migrate the Same Application to Gradle.**

**Steps:**

* Create a new Maven Project
* Initialize the Gradle from the Maven Project Path.
* Add ClassPath to build.gradle file
* Run the Gradle Application to verify

**Create a new Maven Project:**

Use Command 🡪 mvn archetype:generate -DgroupId=com.example -DartifactId=maven-example -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

**Update Maven Project – If needed**

* Update the Project with following code in **App.java**.
* File path **\maven-example\src\main\java\com\example\App.java**

**Build and Run Maven Project**

Navigate to Project Path and Run Maven Project.

Mvn compile

Mvn package

Java -cp package name path name

**Initialize Gradle**

* Navigate to Project Path and Initialize Gradle. Use Commands 🡪 gradle init
* It will ask Found a Maven build. Generate a Gradle build from this? (default: yes) [yes, no] 🡪 Type Yes
* Select Groovy as build Script

Type Yes for Generate build using new APIs

**Add Class Path Gradle File**

* main = 'com.example.App’ 🡪 This tells Gradle which class has the main() method
* classpath = sourceSets.main.runtimeClasspath 🡪 Use the output of my main source folder and any runtime libraries (like JAR files) depends on.

task run(type:JavaExec){

main=‘com.example.App’ (path for java file)

classpath= sourceSets.main.runtimeClasspath

}

**Build and Run Gradle**

Use Commands 🡪 gradle build to build the Project and

Use Command 🡪 gradle run to run the Project

**Program 5: Introduction to Jenkins**

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

* Jenkins is a free, open-source automation server that helps with software development. It can be used to automate tasks like building, testing, and deploying software. Jenkins is a popular DevOps tool that's used by many development teams.

**What it’s used for:**

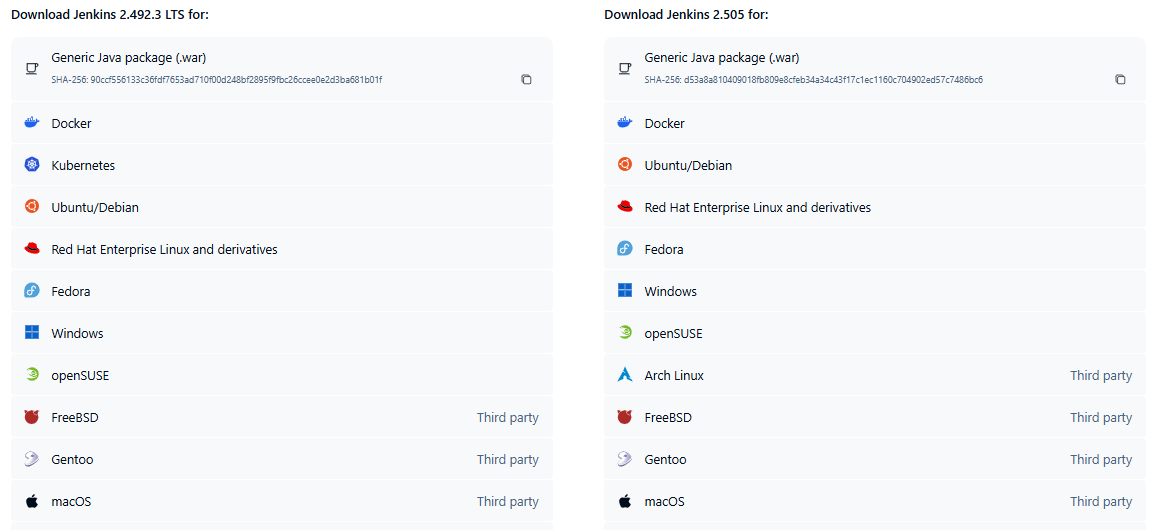
* Automating build and test processes (CI/CD)
* Running jobs on a schedule
* Integrating with tools like Git, Maven, Gradle, Docker, Ansible, etc.
* Triggering workflows when code is pushed (e.g., to GitHub)

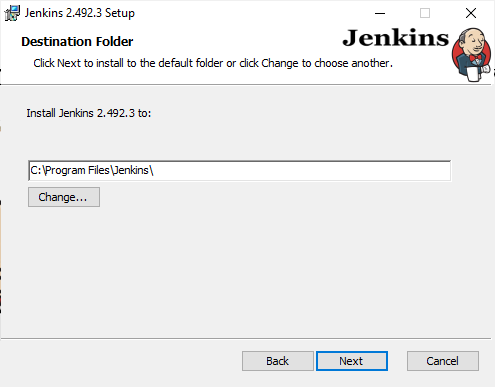
**Prerequisites**

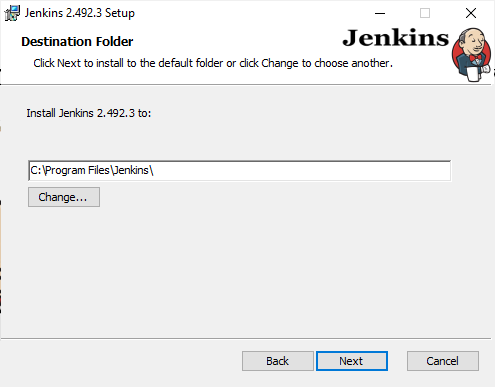
* Java JDK 11 or 17 installed and configured
* Check with: java –version
* Set JAVA\_HOME environment variable

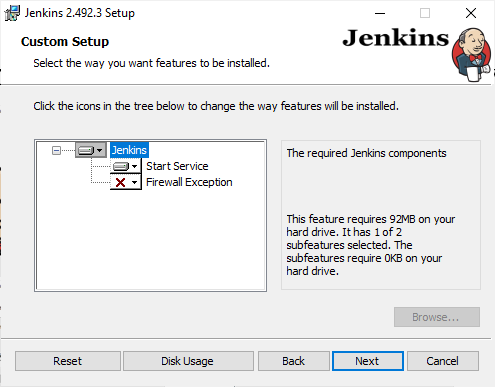
**Download Jenkins**

* Go to <https://www.jenkins.io/download/>



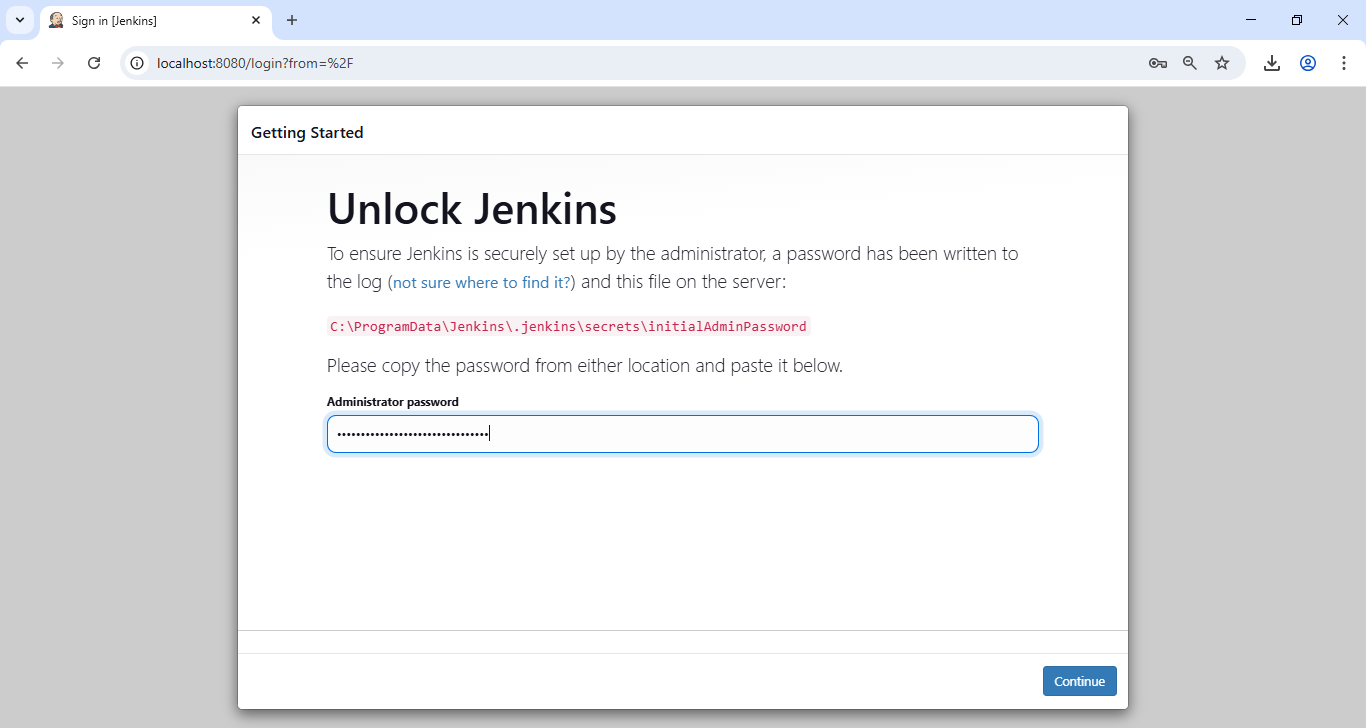






Setup Jenkins

* Open Browser and navigate to <http://localhost:8080/>
* Navigate to the mentioned Jenkins Path to obtain the Administrator Password C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
* Click Continue
* Select “Install Suggested Plugin”
* Setup the Admin User Account



**Program 6: Continuous Integration with Jenkins**

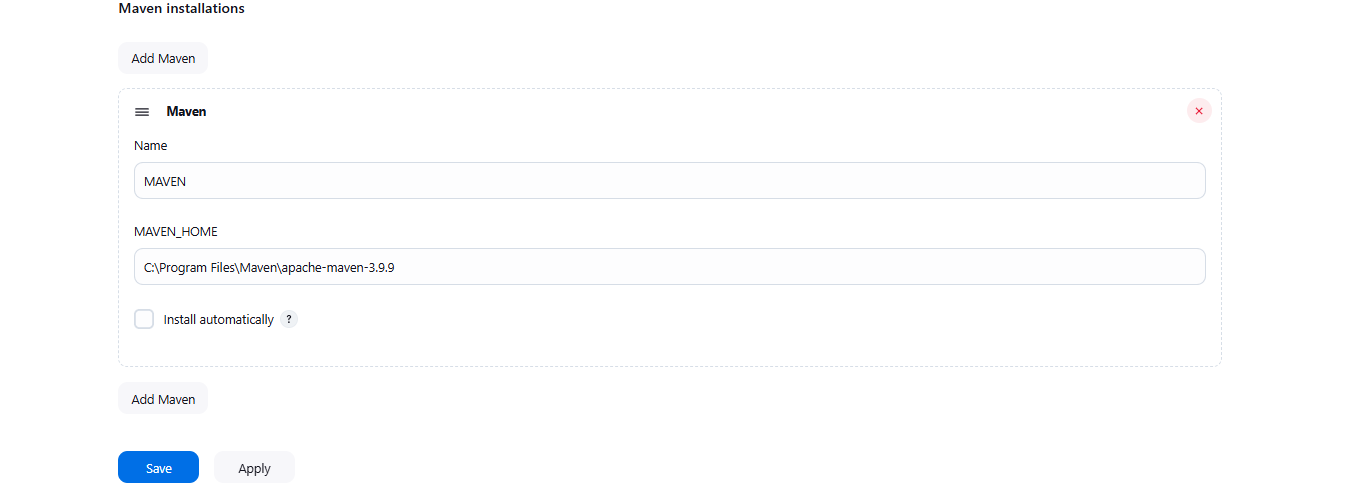
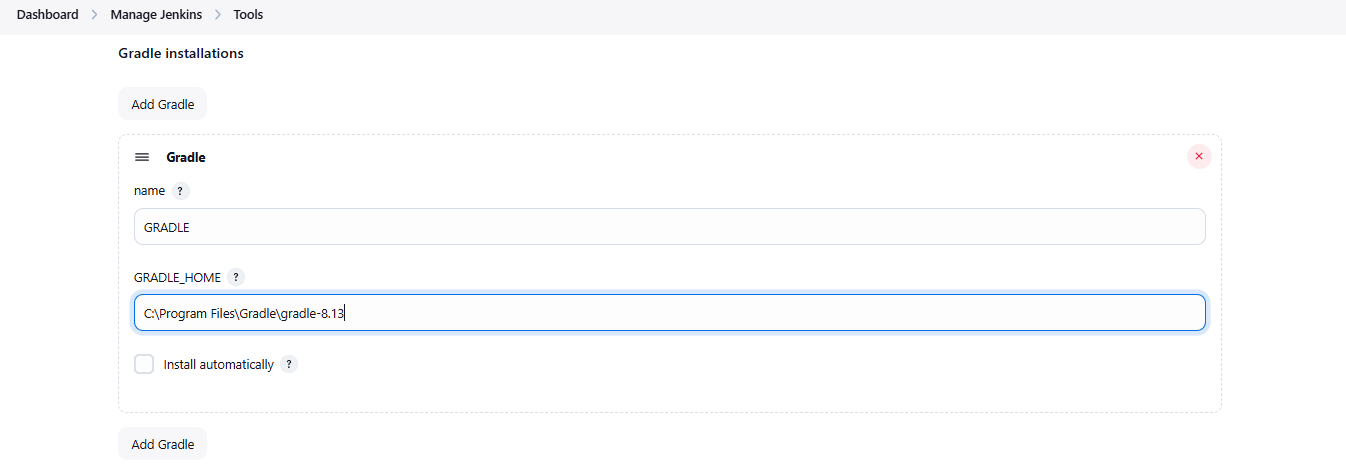
Setting Up a CI Pipeline, Integrating Jenkins with Maven/Gradle, Running Automated Builds and Tests

**Configure Jenkins with Required Tools**

Go to: Manage Jenkins > Global Tool Configuration

Maven: Add Maven installation (e.g., Maven 3.8.6)

Gradle (if needed): Same as Maven



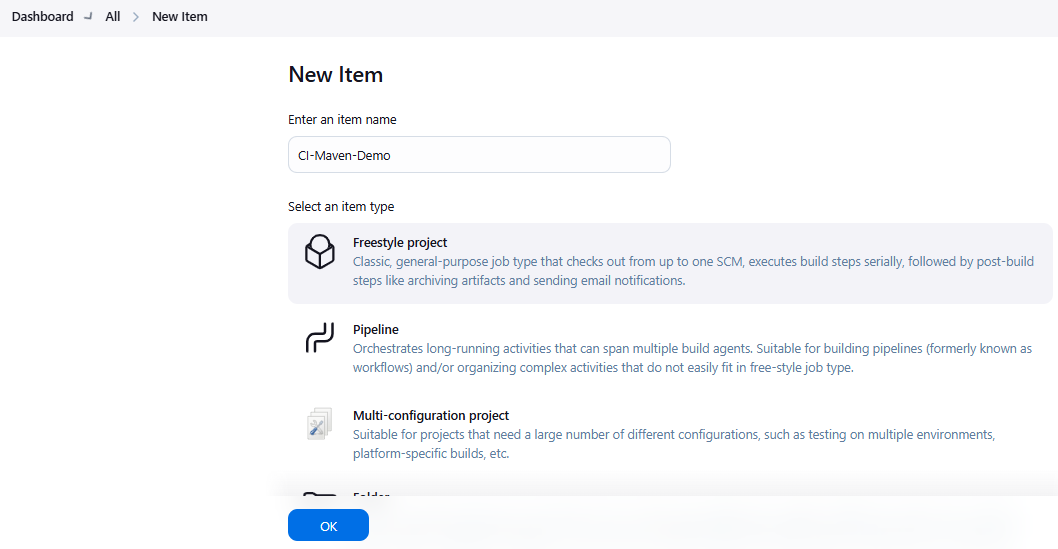
Create a

Create a Freestyle Project in Jenkins:

1.Open Jenkins → click New Item

2. Enter a name: CI-Maven-Local

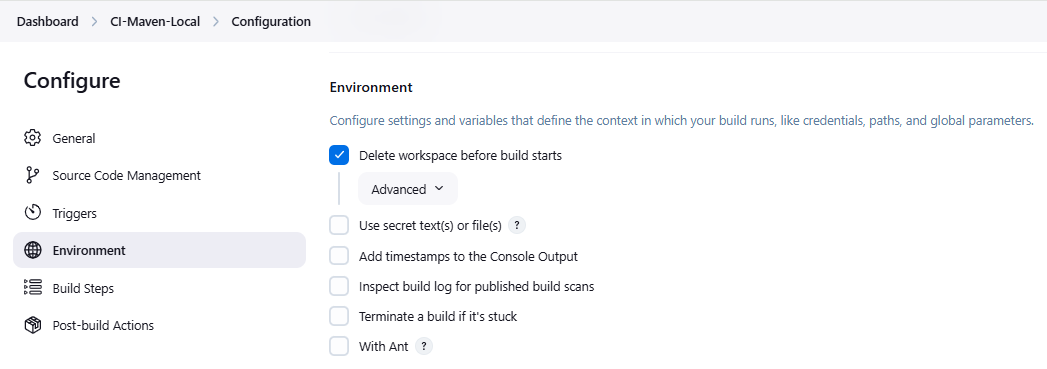
3. Choose Freestyle project → click OK



Configure Source (Local Filesystem)

In the Build Environment section:

Enable: Delete workspace before build starts (optional for clean builds)



Configure Source (Local Filesystem)

Under Build Steps

1. Add a "Execute Windows batch command“

Set:

xcopy /E /I /Y "E:\Dhivya\DevOps\Exp6\ci-demo\\* " "%WORKSPACE%“

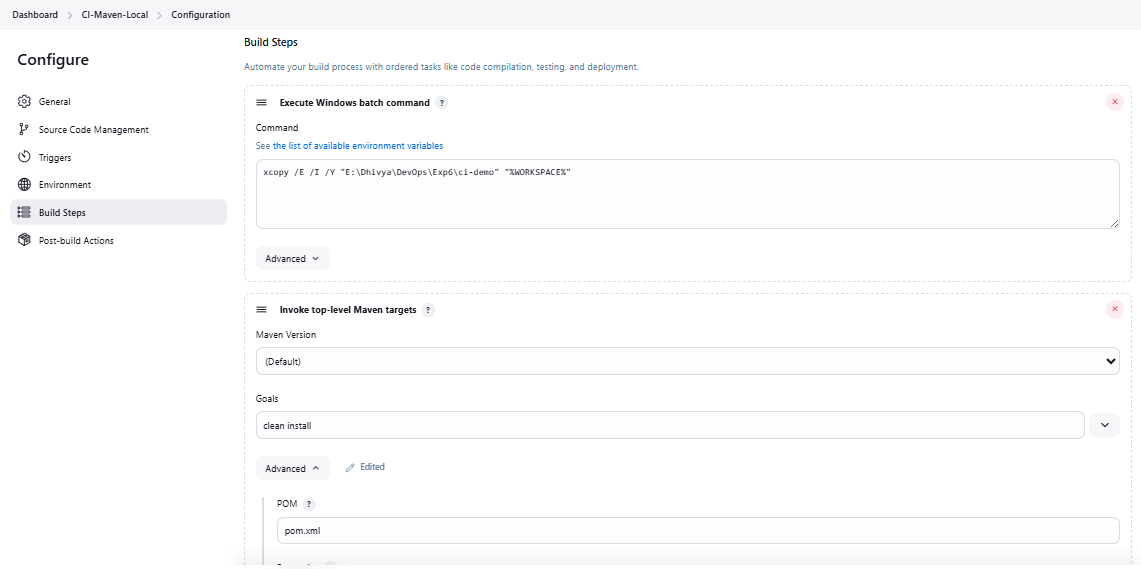
(/E Copy **E**verything; /Y **Y**es to overwrite; /I The destination is a **directory** when you’re copying more than one file)

2. Choose Add build step → Invoke top-level Maven targets

Set:

Goals:cleaninstall   
(**"Goals"** tells Maven **what phases** of the build lifecycle to run)

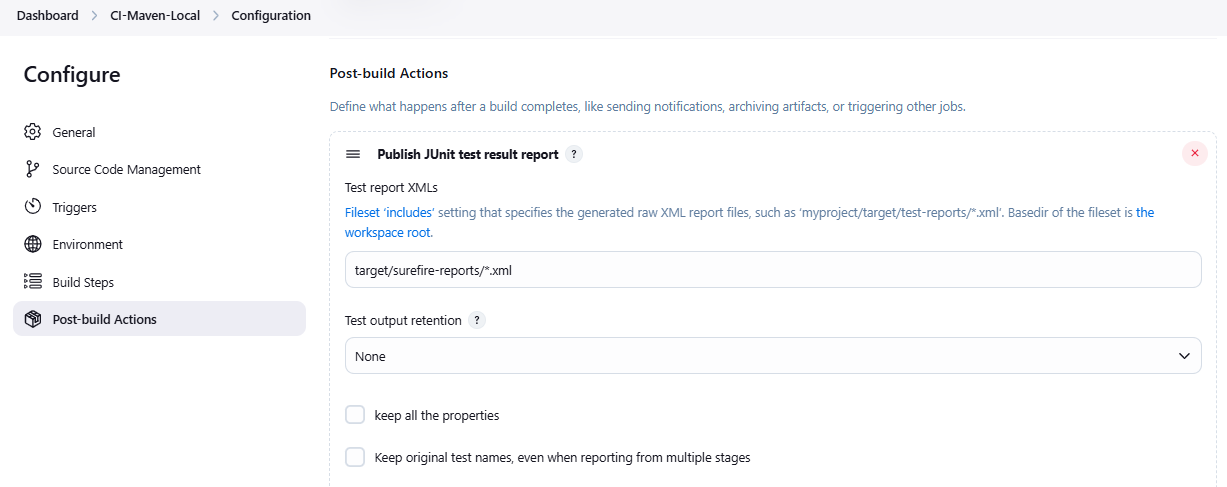
POM: pom.xml



Add Post-Build Actions

Click Add post-build action → Publish JUnit test result report

Report files: target/surefire-reports/\*.xml



Build and View Results

1. Click Save

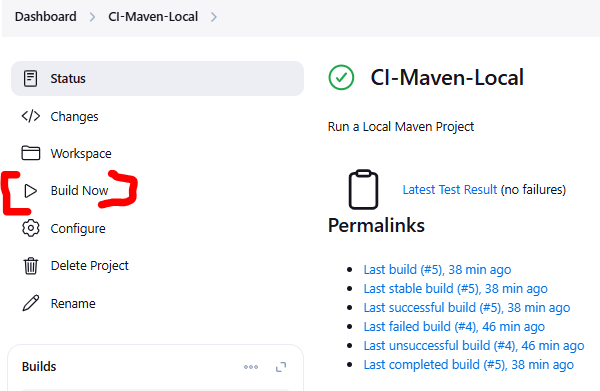
2. Click Build Now

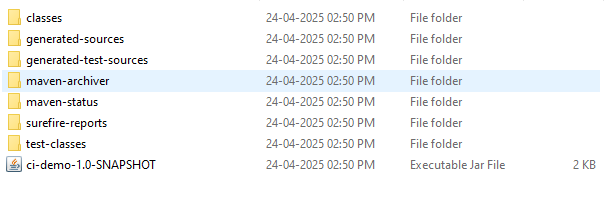
3. Monitor the Console Output to verify:

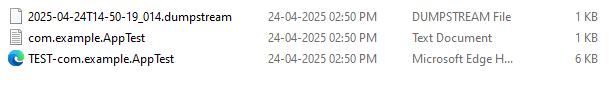
4. Maven is building the project

Tests (if any) are executed

.jar file is generated in the target/ folder







**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

**Ansible Introduction**

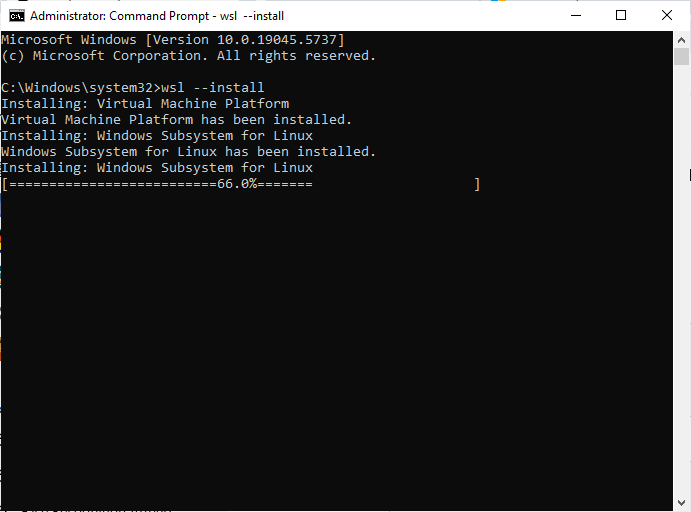
Ansible is an open-source automation tool used for configuring systems, deploying software, and orchestrating tasks. It uses simple YAML-based playbooks and communicates with remote machines over SSH or locally.

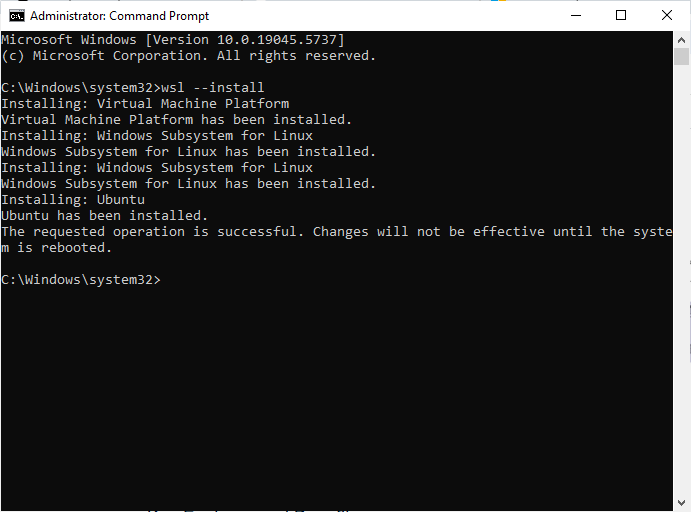
Ansible automates repetitive IT tasks like software installation, updates, and service management. It helps ensure consistency, speed, and error-free execution across multiple servers.

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| **Inventory** | **A file that lists the IP addresses or hostnames of machines to configure** |
| **Modules** | **Predefined tasks in Ansible (like apt, copy, service)** |
| **Playbook** | **YAML file that defines what Ansible should do** |
| **Task** | **An individual automation action in a playbook** |
| **Ad-hoc Command** | **One-liner Ansible command for quick tasks** |

**Prerequisites on Windows**

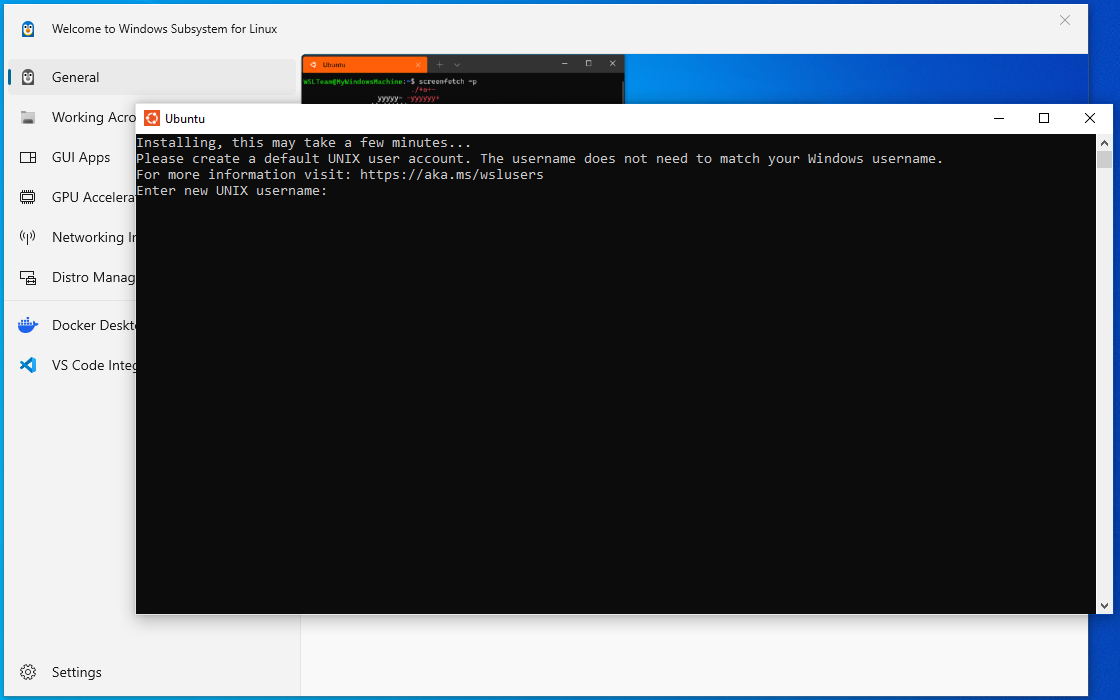
1. Use WSL (Windows Subsystem for Linux)
2. Cmd 🡪 wsl --install

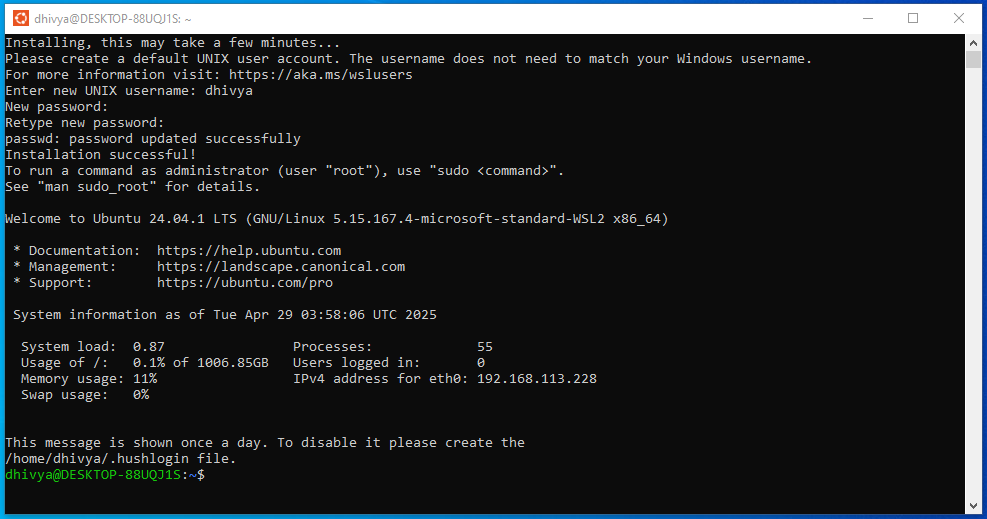




Setup UNIX

. Complete the WSL setup by adding new UNIX credentials

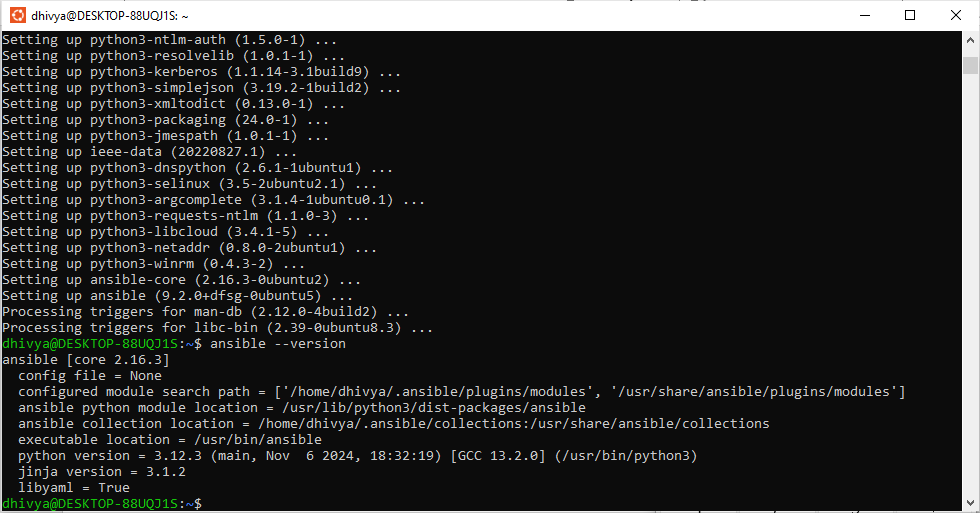




Verify Ansible

1. Execute command

ansible --version to verify if Ansible is installed



Ansible inventory file

1. Why are we creating inventory?   
   We create an inventory file to define the target machines or groups that Ansible should manage.
2. What does nano inventory do?  
   The nano inventory command opens the file in the nano text editor to let you type and save the host configuration.

Create inventory file:

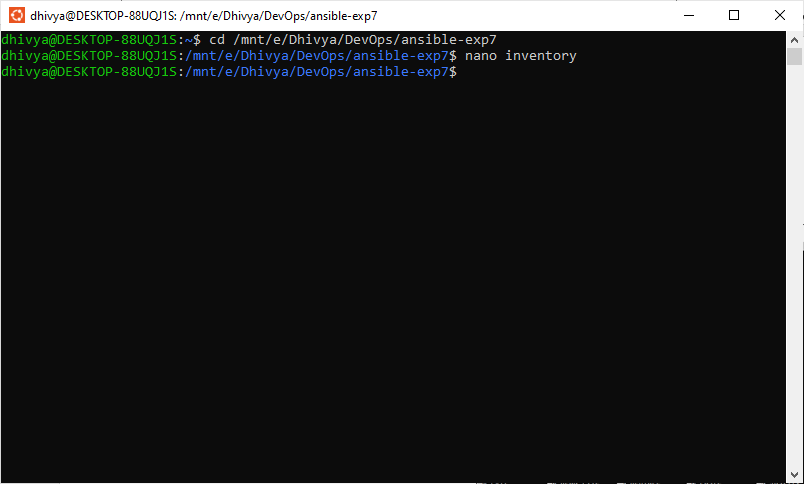
1. Create a folder in the desired directory.
2. Use following command to mount the ubuntu application to directory

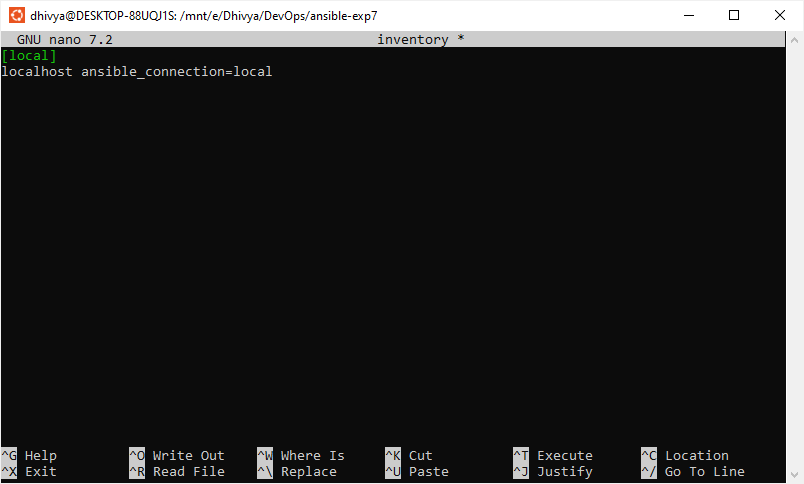
**cd /mnt/e/Dhivya/DevOps/ansible-exp7**

1. Use Command 🡪 **nano inventory** To create an inventory file
2. Type
3. [local]
4. Localhost ansible\_connection=local   
   Ctrl + S to Save ; Ctrl + X to return to main terminal  
     
   [local] – This defines a group name called local (you refer to this name in the playbook under hosts).

localhost – This is the actual machine Ansible will run commands on (your own system).

ansible\_connection=local – This tells Ansible not to use SSH, but to run commands directly on the local machine.





**If you want to run the playbook on a different (remote) machine:**

Let’s say the remote machine IP is 192.168.1.10 and the user is ubuntu.

You update the inventory like this:

[web]

192.168.1.10 ansible\_user=ubuntu ansible\_ssh\_private\_key\_file=~/.ssh/id\_rsa

ansible\_user=ubuntu – This is the Linux user on the remote machine.

ansible\_ssh\_private\_key\_file – Path to your SSH key used to log in (ensure passwordless SSH access).

Your playbook should have:

hosts: web

**If you want to run on a list of machines:**

Example with multiple IPs:

[webservers]

192.168.1.10 ansible\_user=ubuntu

192.168.1.11 ansible\_user=ubuntu

192.168.1.12 ansible\_user=ubuntu

Now, Ansible will run the same playbook across all these machines in parallel (unless serial execution is specified).

Playbook:

hosts: webservers

**Create playbook.yml file**

1. Use Command 🡪 nano playbook.yml

To create a YAML file

Copy following content

- name: Experiment 7 - Run Task on Localhost

hosts: local

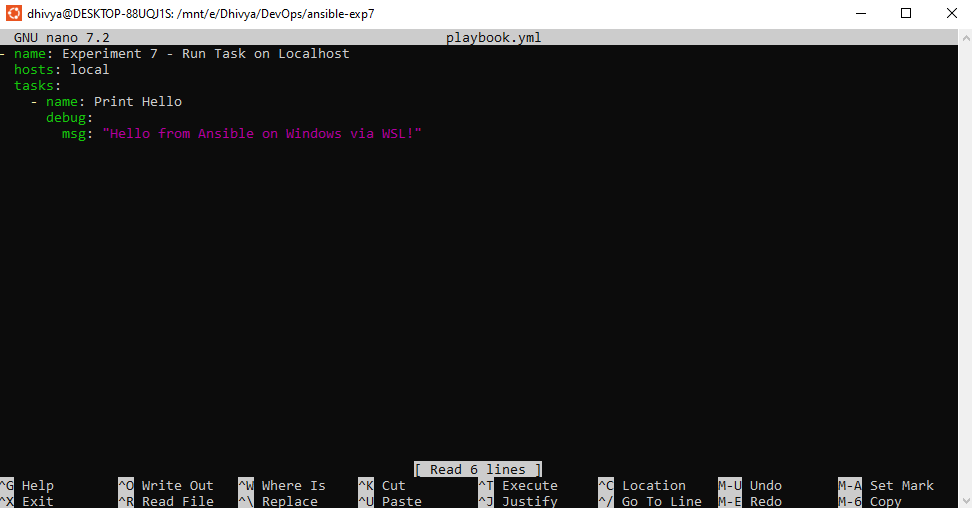
tasks:

- name: Print Hello

debug:

msg: "Hello from Ansible on Windows via WSL!“

1. Ctrl + S to Save ; Ctrl + X to return to main terminal

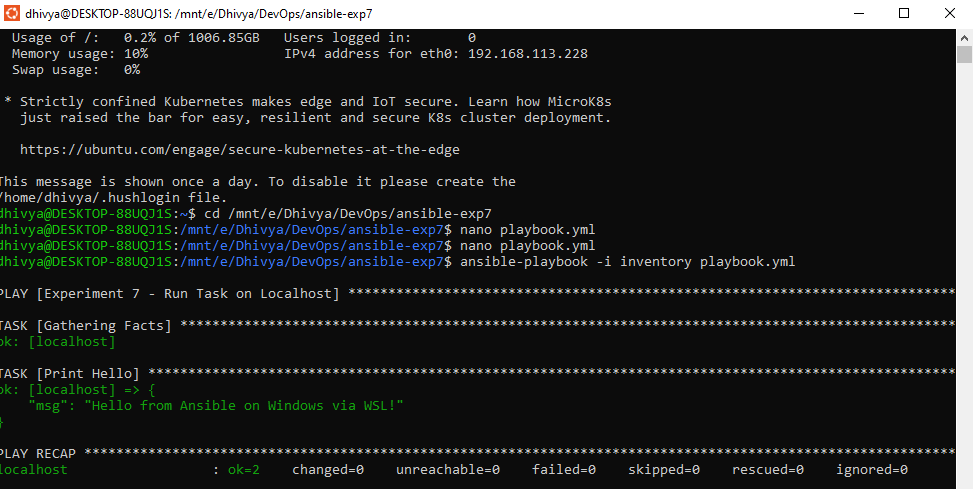


**Run the Playbook:**

1. Use Command 🡪 **ansible-playbook -i inventory playbook.yml**

This command tells Ansible to use the inventory file to find hosts and execute the tasks written in playbook.yml.

Ansible reads the playbook and runs each task on the defined hosts using local or remote connections.



**8. Practical Exercise: Set Up a Jenkins CI Pipeline for a Maven Project , Use Ansible to Deploy Artifacts Generated by Jenkins**

**Create Ansible Project(Program 7)**

1: Log in to Jenkins

2: Click New Item

3: Type Item Name

Click freestyle project

4: click general

5: Go to Environment and enable the check box

Delete workspace before build starts

6: Build Step: Execute Window batch command

Type:

wsl ansible-playbook -i /mnt/e/Dhivya/Devops/ansible-exp7/inventory /mnt/e/Dhivya/Devops/ansible-exp7/playbook.yml

7: Click Save

8: Go to Services –> Search Jenkins –> properties –> tab -> log on –> set system username and system password –> apply –> ok

9: Click Restart the services

10: Log out Jenkins and Re-login

11: Open Ubuntu and check Java version by using the command java –version – If the Java is not installed the ubuntu displays the command to install Java.

12: Select Java Installation command of version17 or above

13: Copy and Run the command installation command

14: Give password

15: Set Environment variables for Ubuntu:

Open ubuntu terminal and type nano ~/.bashrc

16: Scroll Down and type

export JAVA\_HOME=/usr/lib/jvm/java-17-openjdk-amd64

export PATH=$JAVA\_HOME/bin:$PATH

17: SAVE AND EXIT (ctrl+s and ctrl+X)

18: Check JAVA version and path by using echo $JAVA\_HOME

19: Go to Jenkins and Click Build NOW go to Console Output And check the status

20: Create Maven Project using the command (Discussed in program 2)

21: Open Playbook file and type

- name: Experiment 7-Run

  hosts: local

  tasks:

    - name: print hai

      debug:

        msg: "hai ansible"

    - name: Ensure Maven is available

      command: mvn --version

      environment:

        JAVA\_HOME: /usr/lib/jvm/java-17-openjdk-amd64

    - name: Run Maven build

      command: mvn clean install

      args:

        chdir: /mnt/d/Subjects/Lab/Mvnproject/myapp

      environment:

        JAVA\_HOME: /usr/lib/jvm/java-17-openjdk-amd64

    - name: Run Maven-built JAR

      command: java -jar target/myapp-1.0-SNAPSHOT.jar

      args:

        chdir: /mnt/d/Subjects/Lab/Mvnproject/myapp

      environment:

        JAVA\_HOME: /usr/lib/jvm/java-17-openjdk-amd64

      register: jar\_output

    - name: Show output from java JAR

      debug:

        var: jar\_output.stdout



22: Save and Close the playbook file

23: Open POM file in visual studio code and type

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>3.2.0</version>

<configuration>

<archive>

<manifest>

<mainClass>com.mvncmd.example.App</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

</plugins>

</build>

</project>

24: Save and close

25: Open Jenkins And build the project

26: Go to Console Output and check status. You should see the Hello World output from the App.java class file

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

Azure DevOps is a cloud-based platform from Microsoft that provides end-to-end DevOps tools for software development and deployment.

It offers services for:

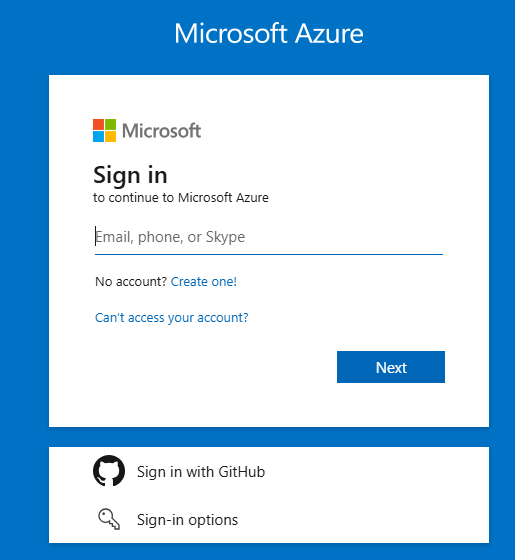
* Azure Repos: Git-based version control
* Azure Pipelines: CI/CD automation
* Azure Boards: Work item and issue tracking
* Azure Artifacts: Package management
* Azure Test Plans: Manual and automated testing

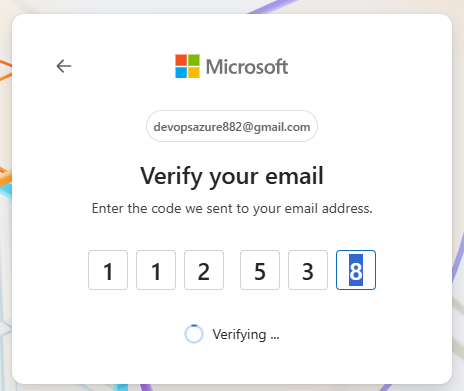
**WHY USE AZURE DEVOPS?**

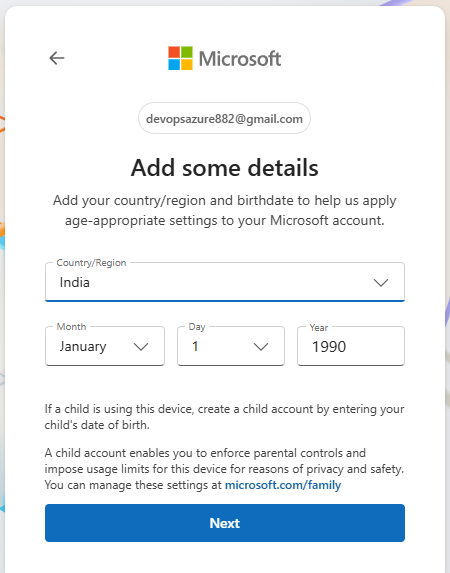
* Full DevOps lifecycle under one platform
* Integrates well with GitHub, Jenkins, and even Ansible
* Works with any language (Java, .NET, Python, Node, etc.)
* Supports both cloud-hosted agents and self-hosted agents
* Cloud-hosted agents are managed by a third-party service provider in the cloud
* Self-hosted agents, on the other hand, are installed and managed by the user on their own infrastructure

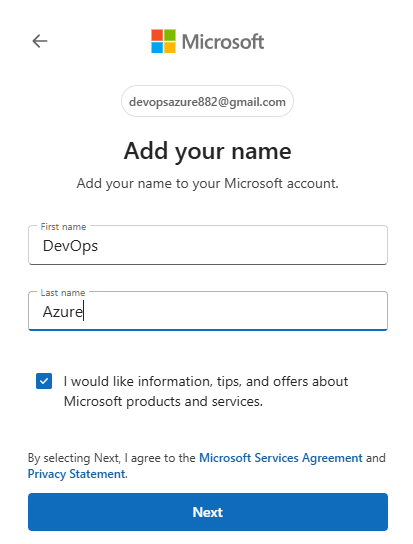
**Setup Azure Account**

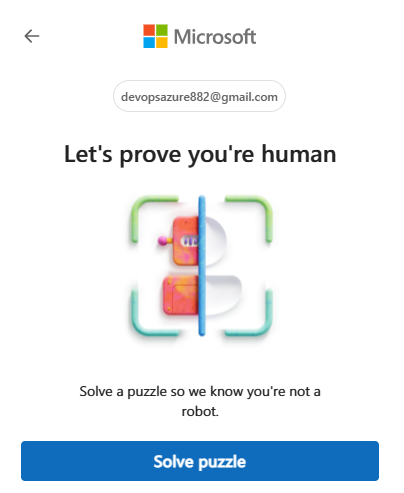
* Login to MS AZURE portal
* Click sign up
* Create new azure account

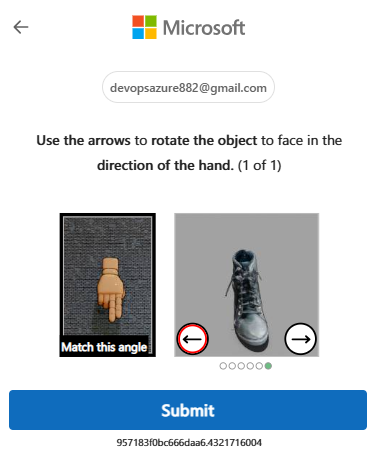
****

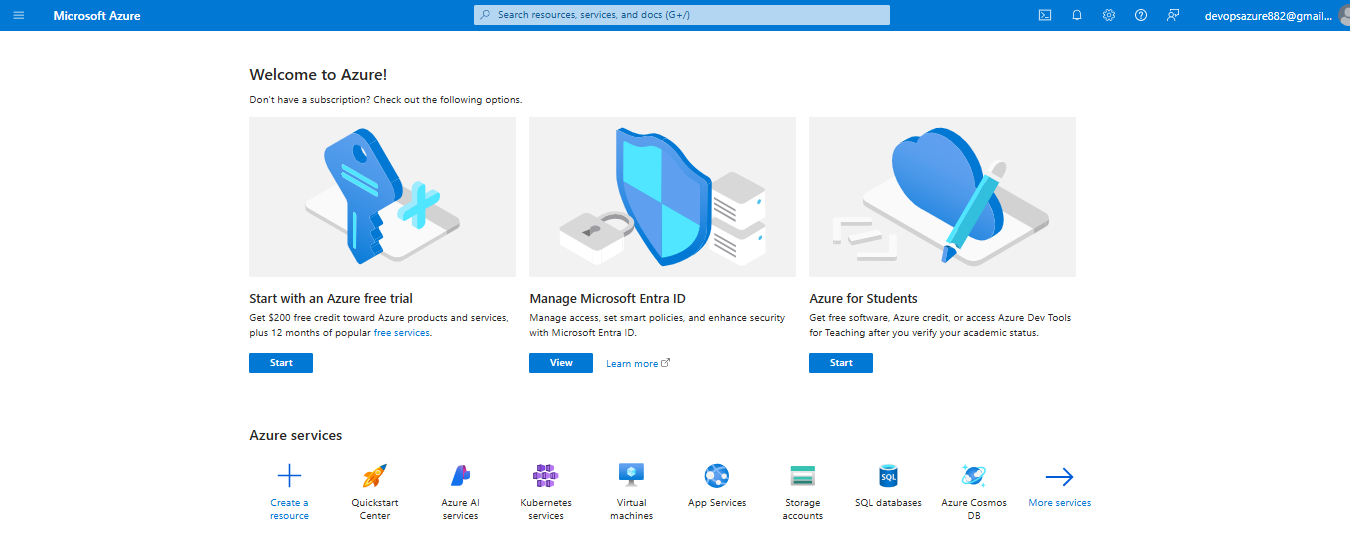
****

****

****

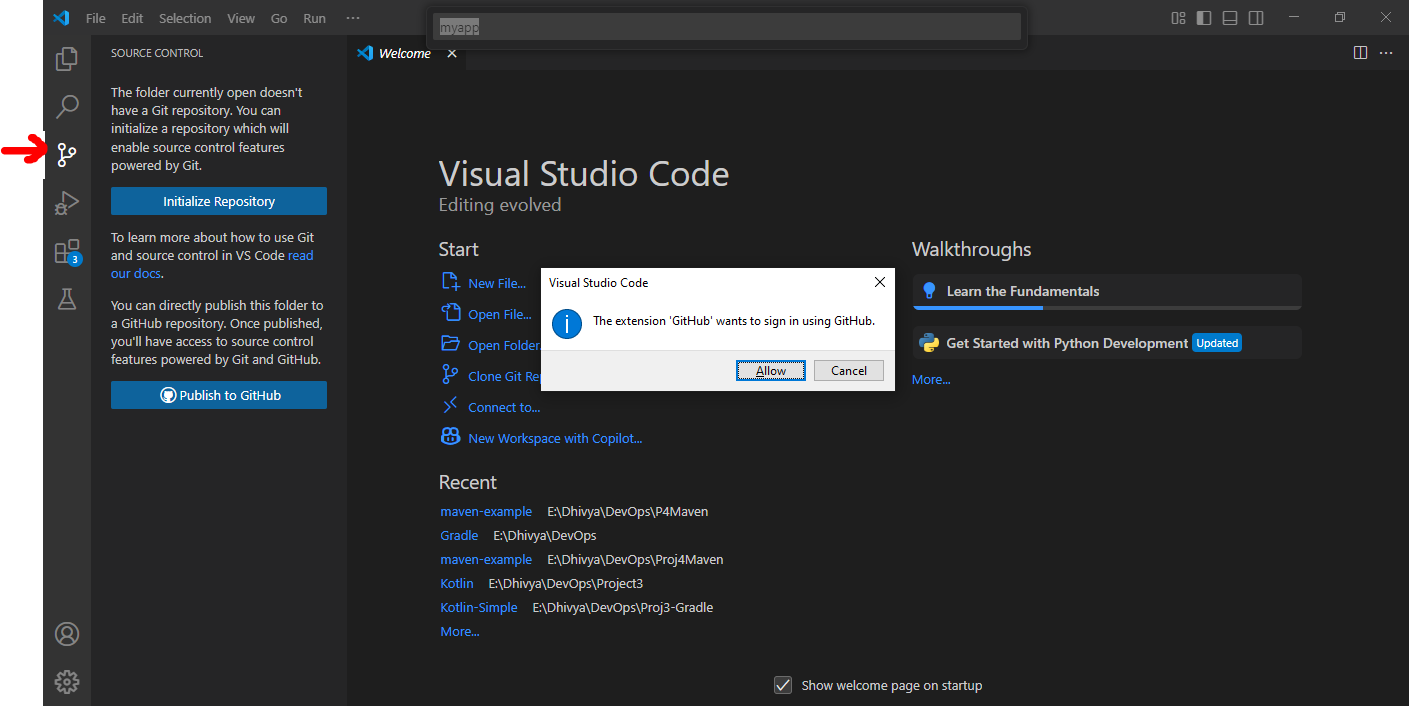
****

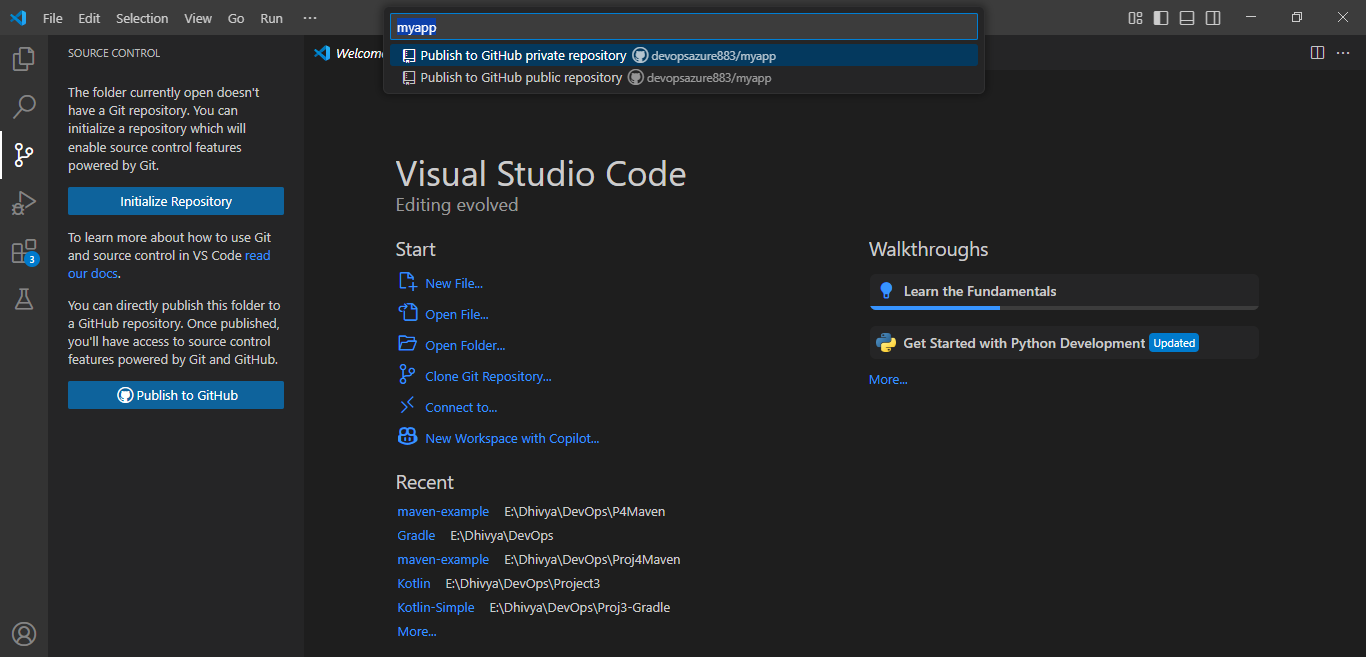
****

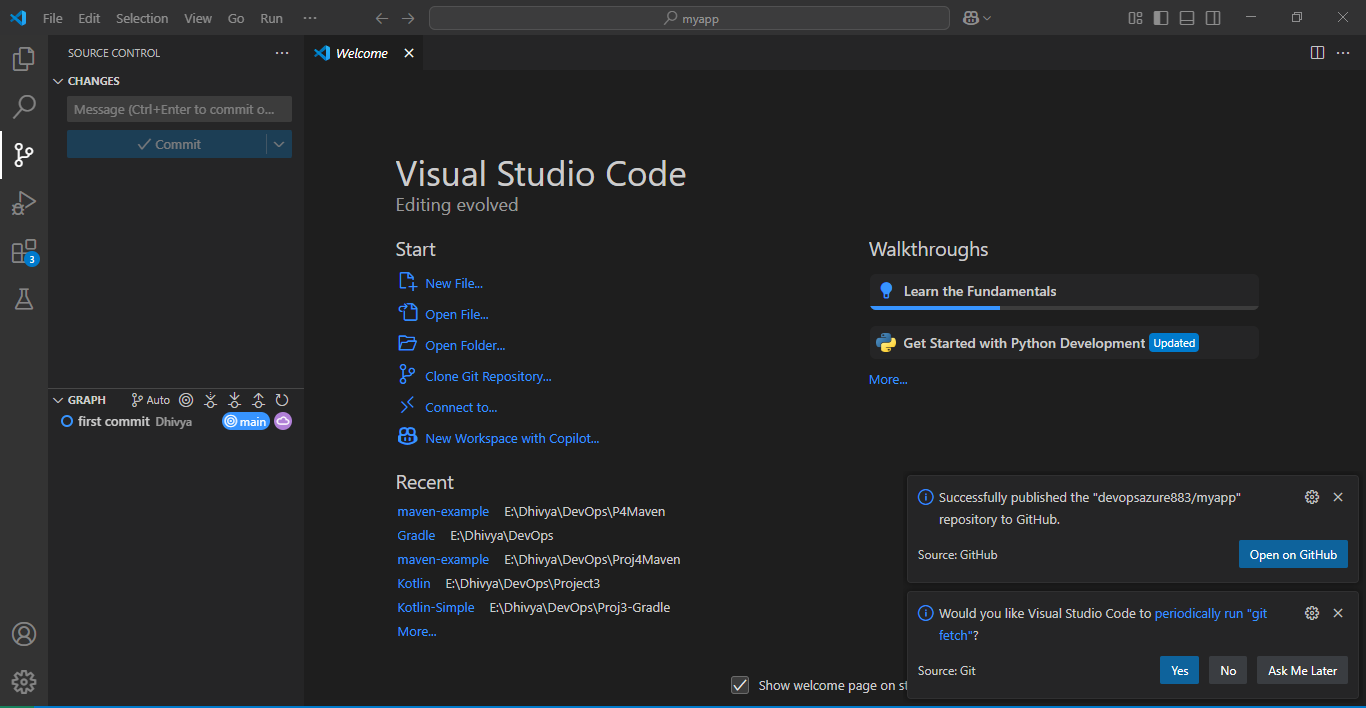
****

**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**

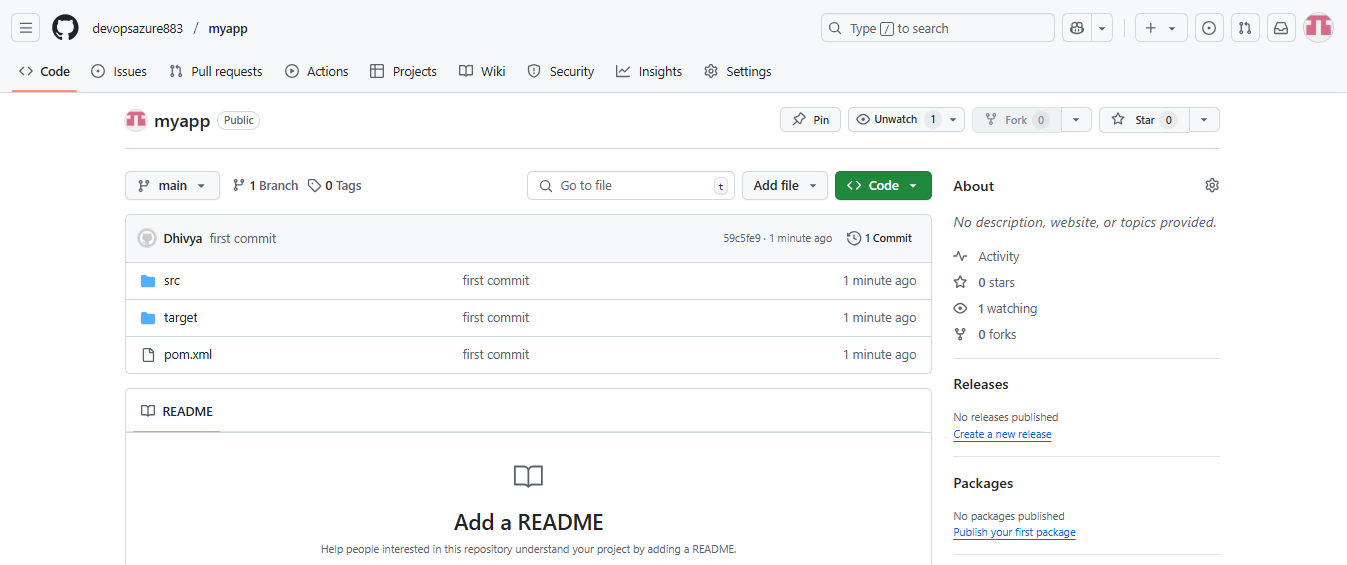
* Create a Maven Project
* Open maven project in visual studio code
* Integrate the code in git repo
* Click source code in VS code
* Click publish to GitHub
* Select either private or public repo
* Select the file which the user wants to integrate
* Uncheck target and click ok
* Code will be pushed to GitHub repository

****

****

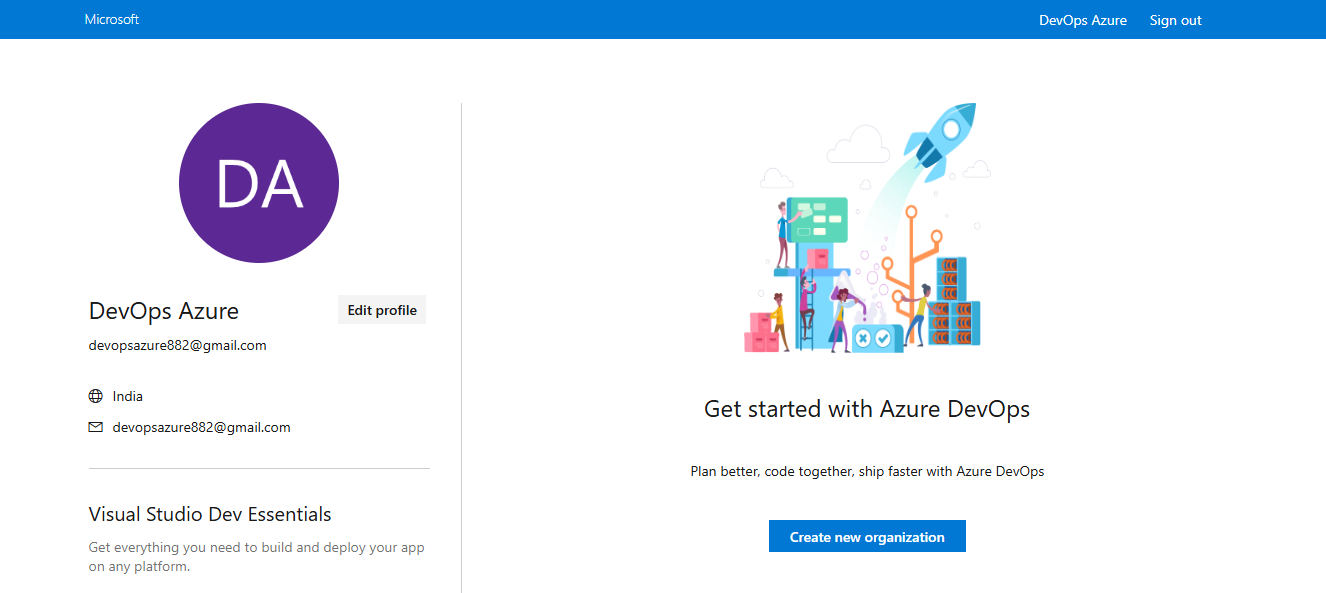
****

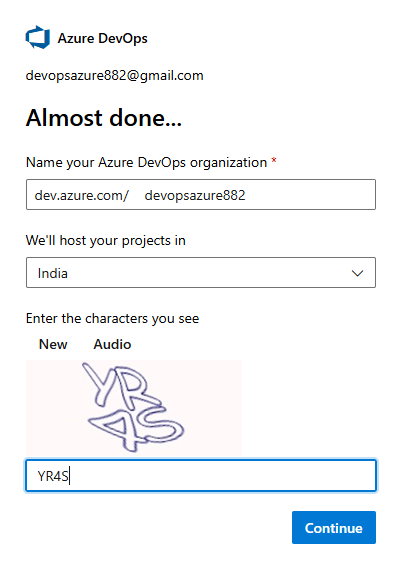
* **Once the code is pushed login to GitHub and check**

****

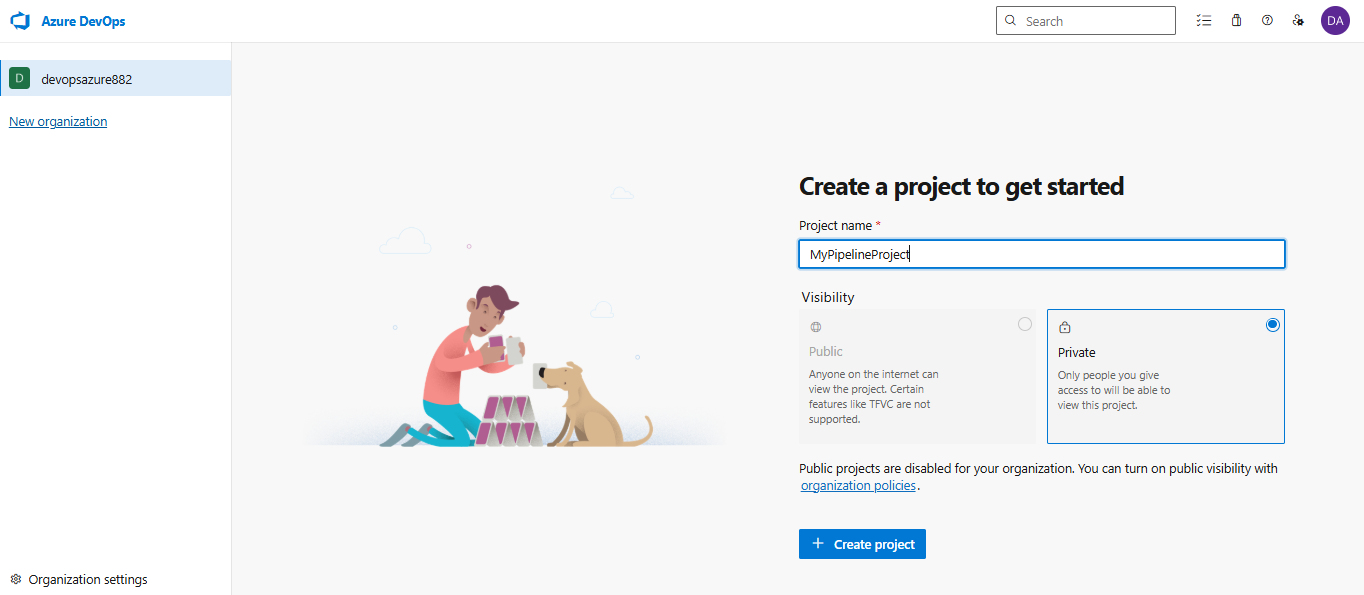
**Login to Azure Devops : https://aex.dev.azure.com/me**

Step 1: Create Organization

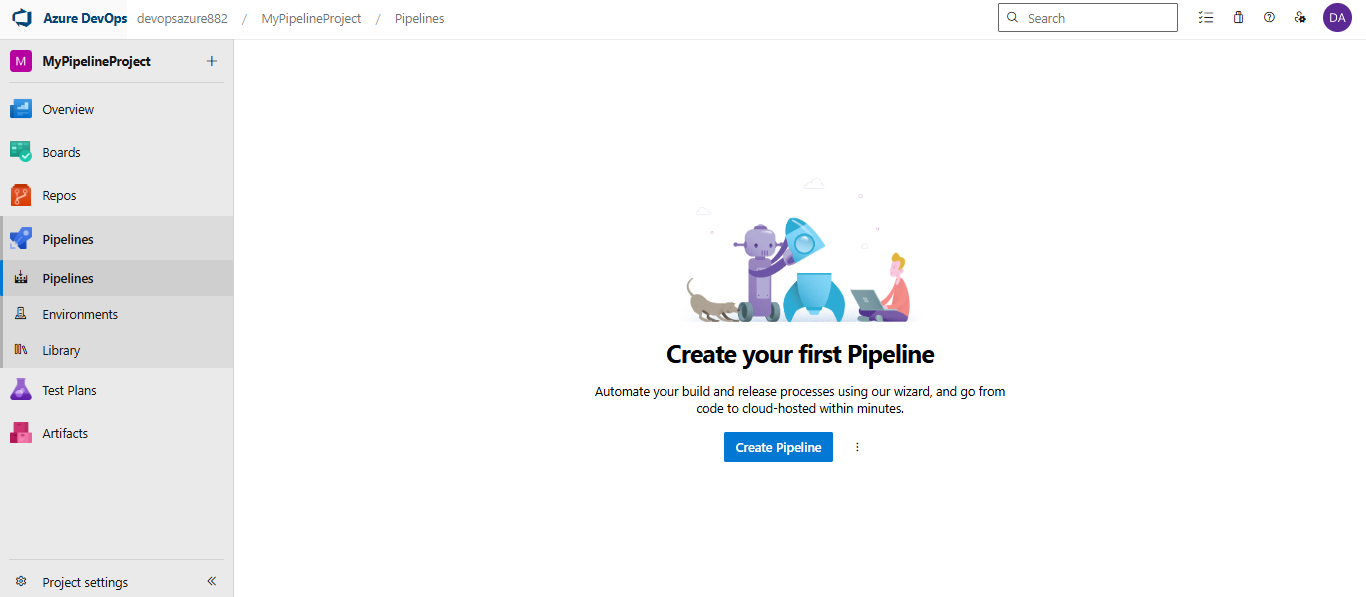


****

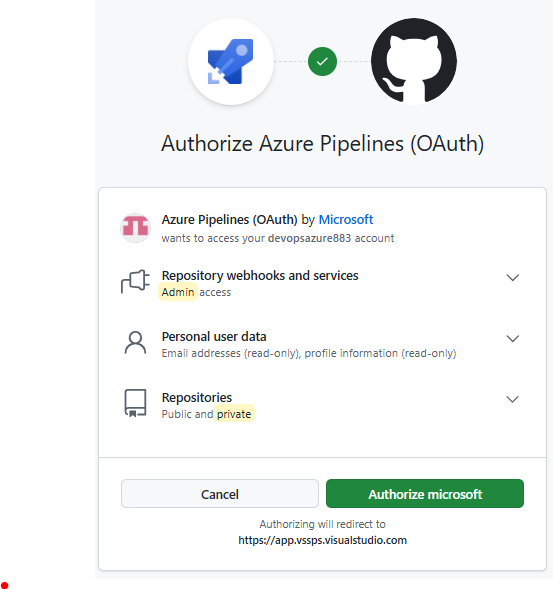
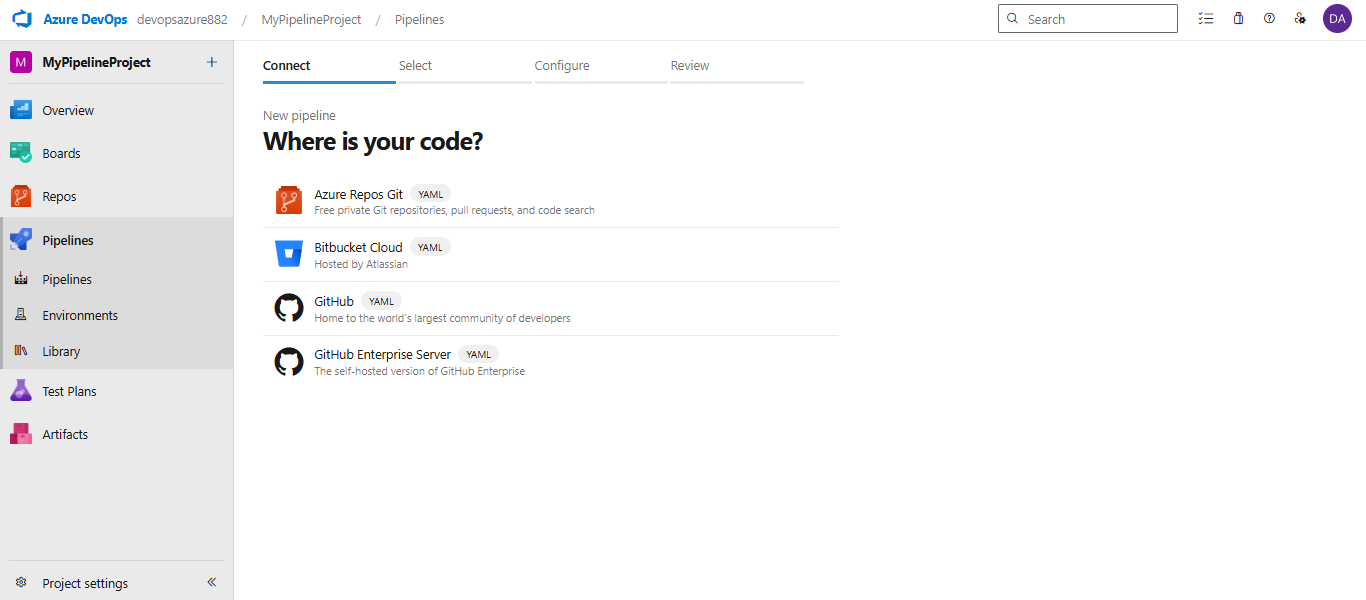
Step 2: Create Project



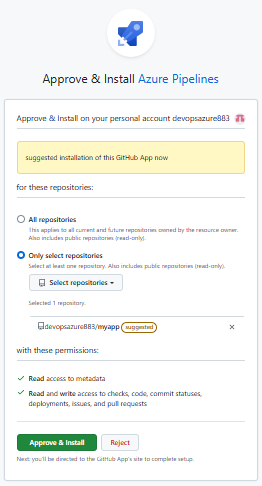
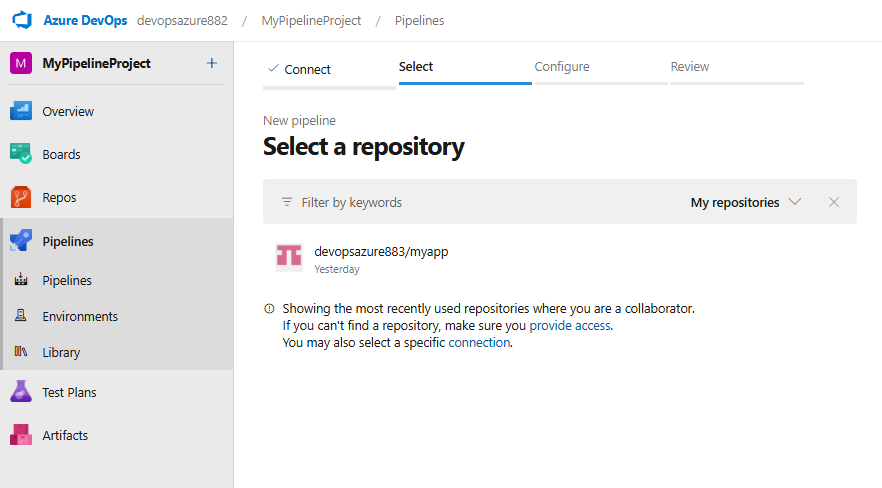
Step 3: Create Pipeline



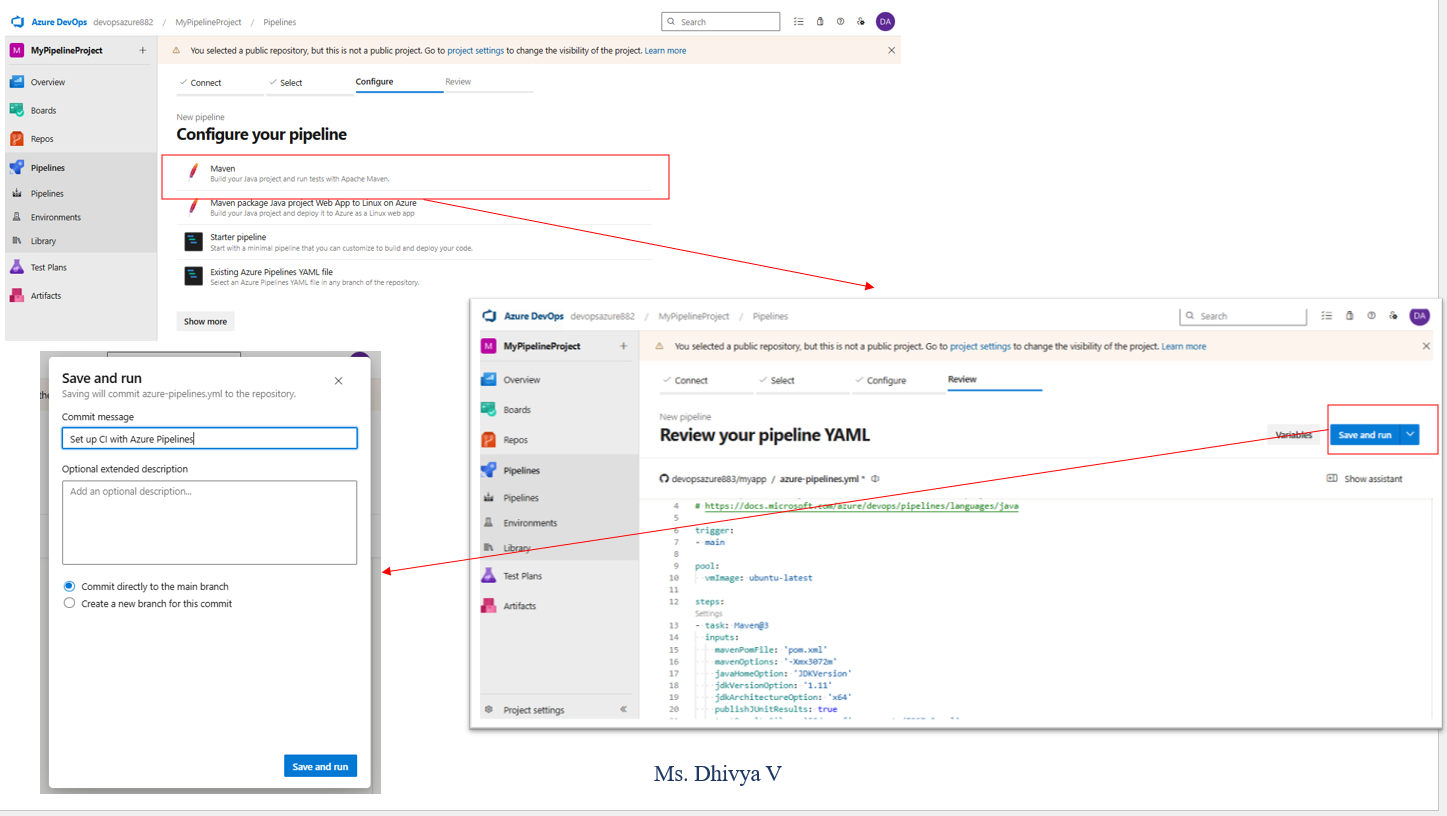
Step 4: Connect to GitHub repo



Step 5: Select Repo



Step 6: Configure Pipeline , select Maven



Step 7: Save and Run the Pipeline

11. Configure Release Pipeline

**App Service overview**

Azure App Service is a platform that lets you run web applications, mobile back ends, and RESTful APIs without worrying about managing the underlying infrastructure.

**Azure Key Vault**

Azure Key Vault provides a way to securely store credentials and other keys and secrets, but your code needs to authenticate to Key Vault to retrieve them.