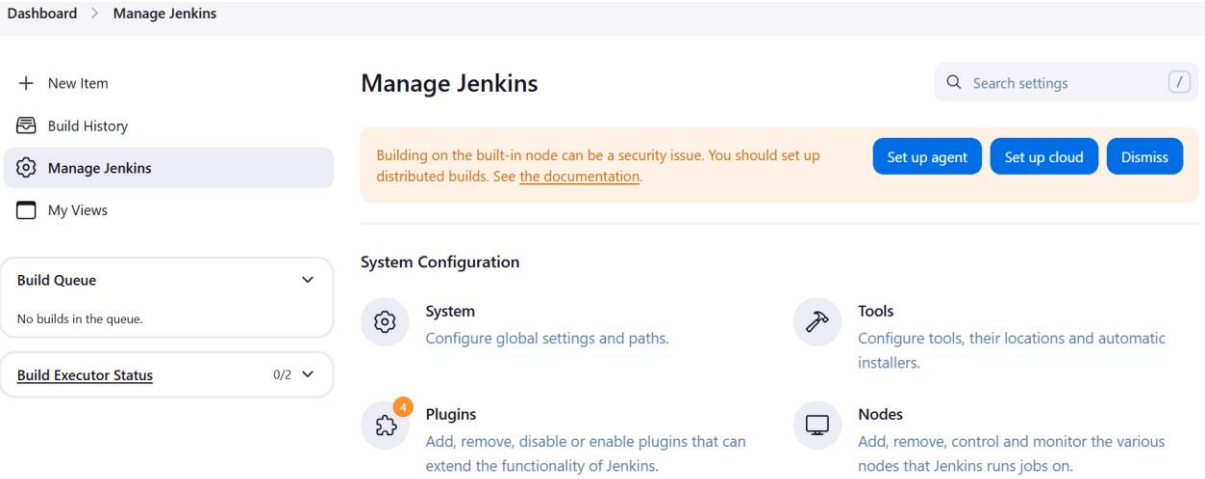
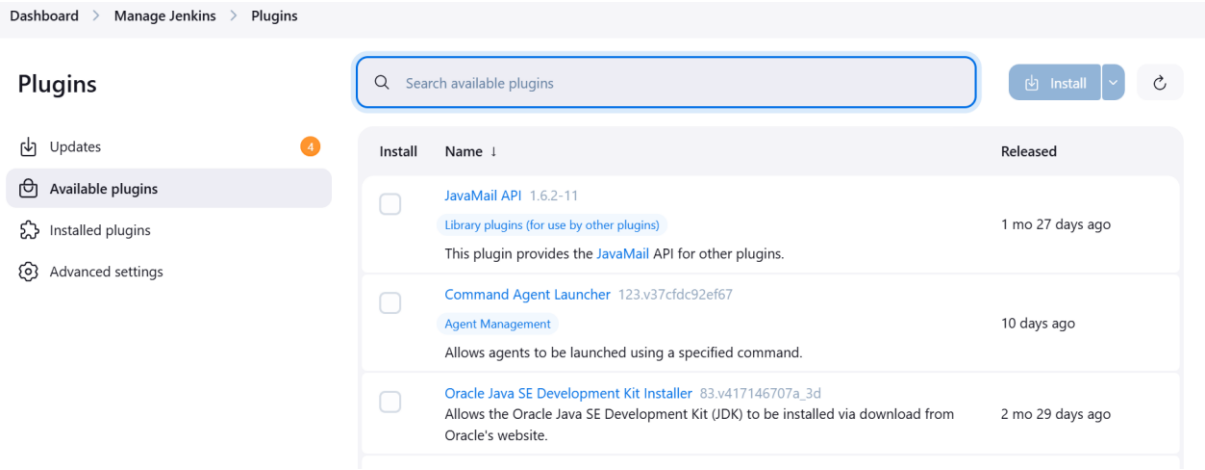
6. Continuous Integration with Jenkins: Setting Up a CI Pipeline, Integrating Jenkins with Maven/Gradle, Running Automated Builds and Tests.

- Login to Jenkins with username and password.
- Select Dashboard -> Manage Jenkins -> Select Plugins



- Select Available Plugins
- From the list of plugins, choose Pipeline maven plugin database and Gradle, then select install



- Once download progress is success, restart Jenkins or click on goto backpage.
- Login again
- Select -> Manage Jenkins -> Tools
- Git Configuration
- Download git for windows 64 bit version
- Do the setup username and emailed

1. **Set your username**:

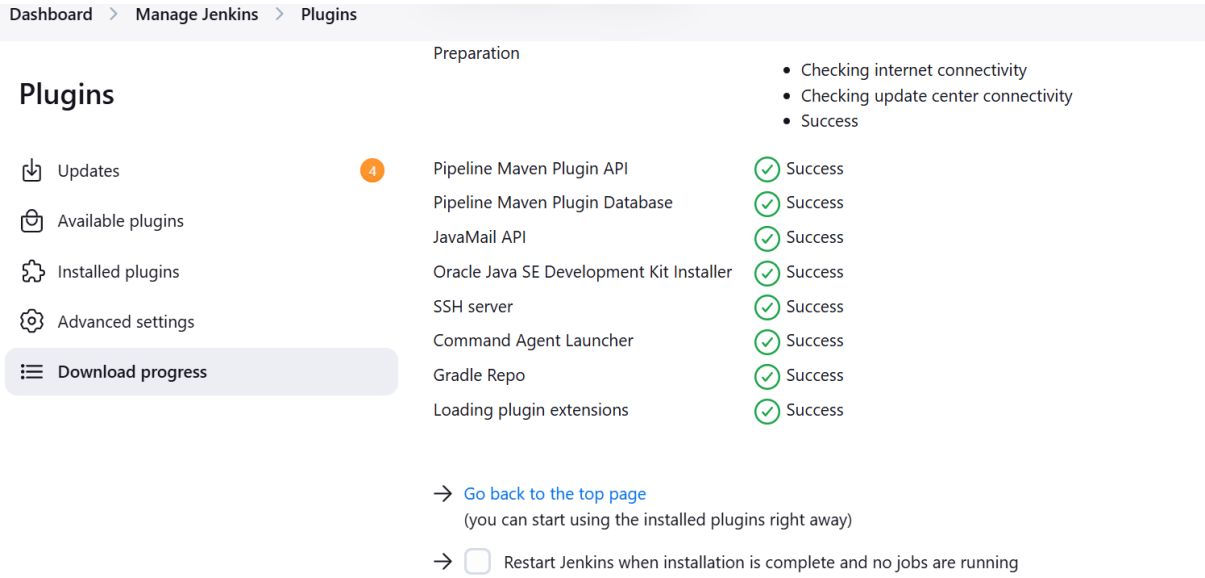
   git config --global user.name "mythilim"

2. **Set your email**:

git config --global user.email mythi.m84@gmail.com

- To verify
  - git config --list

```
Command Prompt

Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

C:\Users\balam>git config --global user.name "mythilim"

C:\Users\balam>git config --global user.email "mythi.m84@gmail.com"

C:\Users\balam>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=mythi.m84@gmail.com
user.name=mythilim

C:\Users\balam>
```



Step 1: Install Required Jenkins Plugins

To integrate Jenkins with Maven/Gradle and automate builds, install the following plugins:

1. Navigate to **Jenkins Dashboard** > **Manage Jenkins** > **Manage Plugins**.
2. Under the **Available Plugins** tab, search for and install:
   - **Pipeline Maven Plugin Database** (ID: pipeline-maven)

- **Gradle Plugin** (ID: gradle)
- **Git Plugin** (ID: git)

3. Restart Jenkins to apply the changes

Note: if the Plugins are not shown in Available Plugins and shown in installed plugin tab, then it indicates that plugins are already installed.

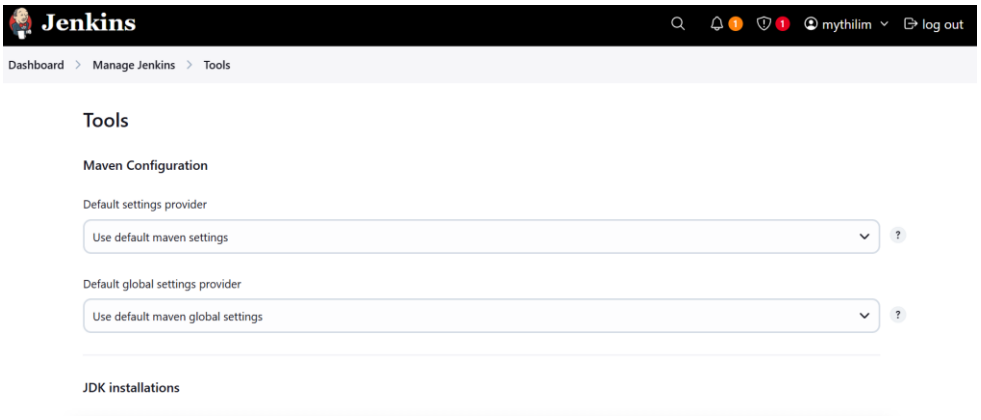
Step 2: Configure Maven/Gradle in Jenkins

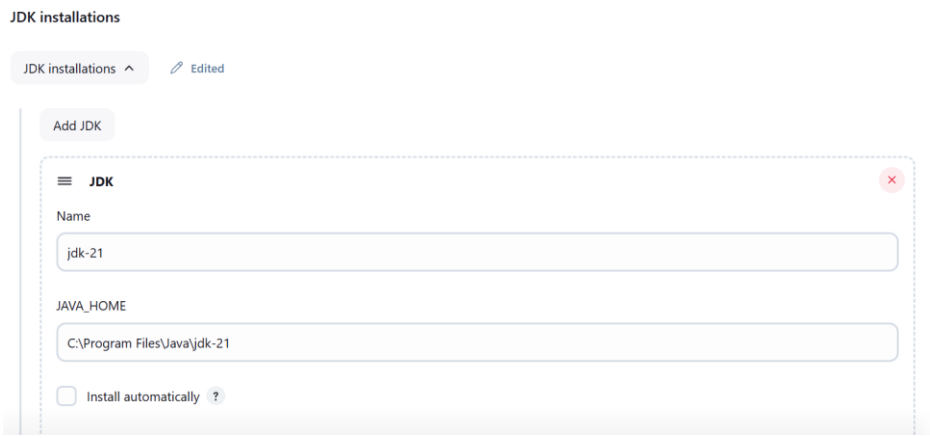
For Maven:

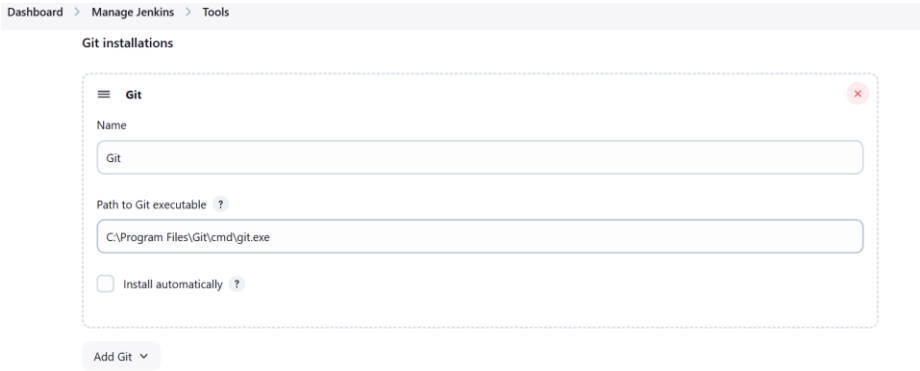
1. Go to **Manage Jenkins** > **Global Tool Configuration**.
2. Scroll to the **Maven** section and click **Add Maven**.
3. Provide a **Name** (e.g., "Maven 3.8") and set the **Maven Home** directory path.
4. Click **Save**.

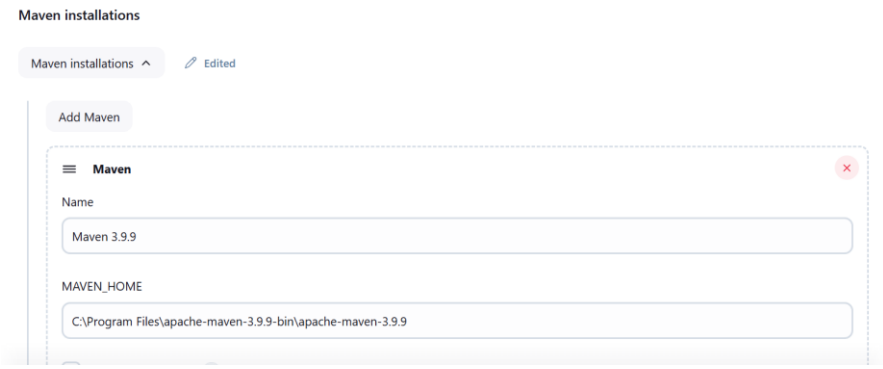
For Gradle:

1. Navigate to **Manage Jenkins** > **Global Tool Configuration**.
2. Scroll to the **Gradle** section and click **Add Gradle**.
3. Provide a **Name** (e.g., "Gradle 7.3") and set the **Gradle Home** path.
4. Click **Save**.

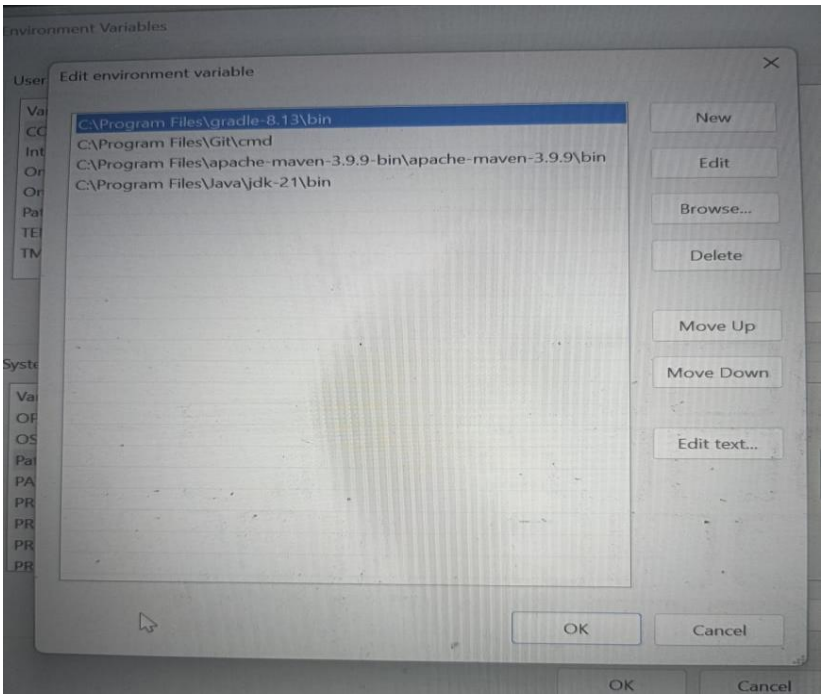








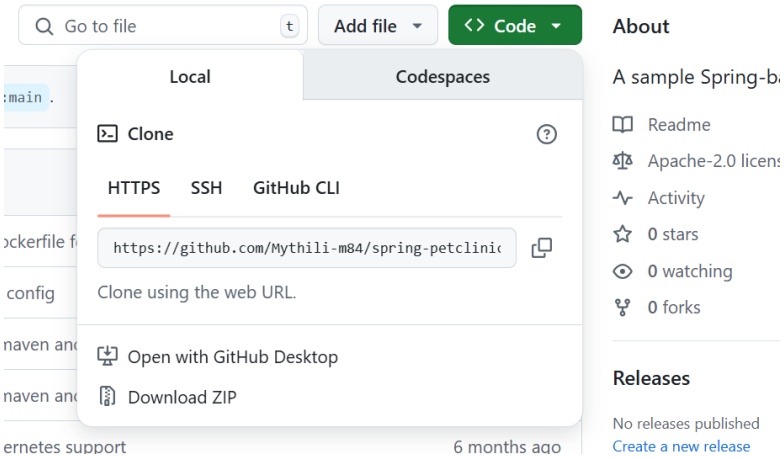
- The same paths should be added in Windows Environment variables.



Step 3: Setup Github Repository

1. Register or login to github
2. Click on create repository
3. Search for spring-projects/spring-petclinic repository

4. Click on fork to your own repository
5. Click the down arrow next to code and copy the clone link of HTTPS.
6. Install Git in local system. If already installed search for Git cmd in windows.
7. Execute command in cmd “git clone https://github.com/spring-projects/spring-petclinic.git”.
8. All the files related to petclinic will be downloaded. Execute cd spring-petclinic
9. mvn compile, mvn test, mvn package, **java -jar target/spring-petclinic-3.4.0-SNAPSHOT.jar**



Step 4: Create a Jenkins Job

1. From the Jenkins dashboard, click **New Item**.
2. Enter a name for the project (e.g., "My-CI-Pipeline").
3. Select **Freestyle Project**.
4. Click **OK** to create the job.

Step 5: Configure Source Code Management (SCM) with Git

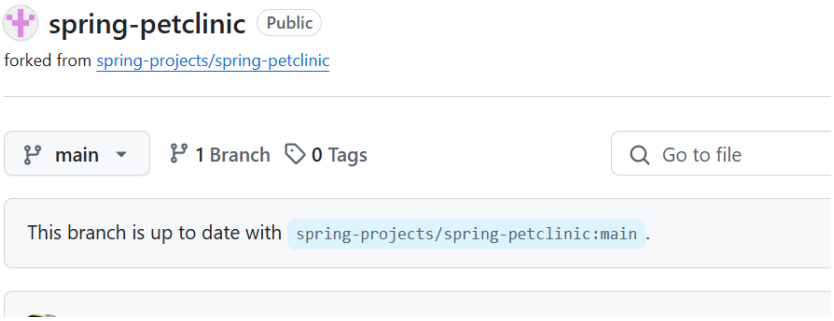
To pull the source code from a Git repository:

Example Project:

1. In the job configuration page, navigate to **Source Code Management**.
2. Select **Git**.
3. Enter the repository URL:

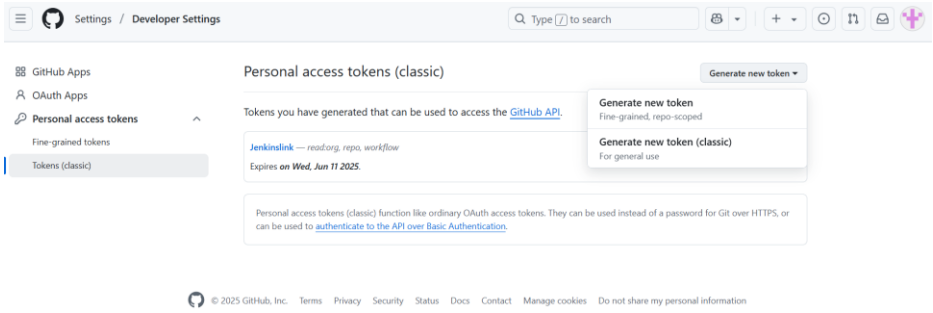
   https://github.com/Mythili-m84/spring-petclinic

4. Under **Branch Specifier**, enter */main

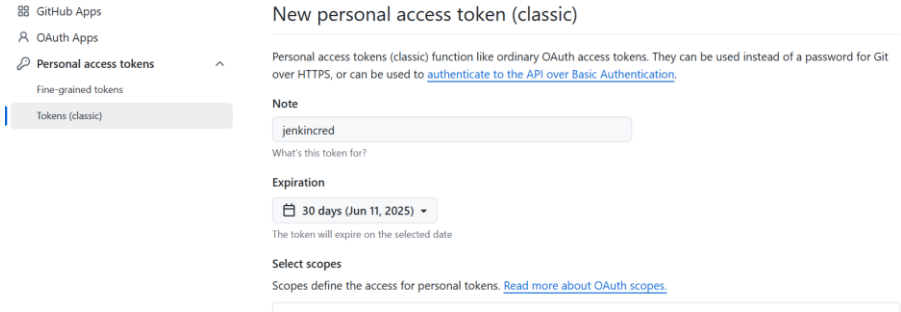


5. If the repository is private, click **Add Credentials** and enter your GitHub username and access token.

    1. Click on github profile and select settings
    2. Choose the last option in the left side menu “Developer settings”
    3. Click on Personal Access Token
    4. Generate new token (Classic)



    5. Specify the name for the token in Note.



    6. Select the below checkboxes repo, workflow and read.org
    7. Click on generate token
    8. Copy the secret token and paste it in Jenkins credential password.

6. Click **Save**.

Step 6: Configure Build Steps

For Maven Projects:

1. Scroll to the **Build** section and click **Add Build Step** > **Invoke Top-Level Maven Targets**.

2. In the **Goals** field, enter:

clean install

3. Click **Save**.

Step 6: Configure Post-Build Actions

1. Scroll to **Post-Build Actions**.

2. Click **Add Post-Build Action** > **Publish JUnit Test Result Report**.

3. In the **Test Report XMLs** field, enter:

    target/surefire-reports/*.xml (for Maven)

build/test-results/test/*.xml (for Gradle)

4. Click **Save**.

Step 7: Trigger Automatic Builds

To trigger builds automatically upon code changes:

1. In the **Build Triggers** section, check **Poll SCM**.
2. Click **Save**.

Step 8: Run and Verify the Build

1. Navigate to the Jenkins dashboard and select the newly created job.
2. Click **Build Now**.
3. Monitor the build status in the **Build History** section.
4. Click on the latest build and navigate to **Console Output** to review logs.
5. Verify that the build completes successfully and test results are generated.

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.**

What is Ansible?

Ansible is an open-source configuration management and automation tool. It helps system administrators and DevOps engineers to automate:

- Software installation and configuration
- Server provisioning
- Application deployment
- System updates and patching

🔑 Key Features of Ansible:

- Agentless: No agent is needed on the target machine. It uses SSH to communicate.
- Simple syntax: Uses YAML to define tasks in Playbooks.
- Idempotent: Running the same playbook multiple times won't cause unintended changes.
- Modular: Uses built-in modules like apt, yum, copy, service, etc.

🔧 Ansible Components:

Component	Description
Inventory	List of target machines to automate
Modules	Reusable scripts used to perform actions
Playbook	YAML file that contains tasks to be executed
Task	A single unit of work to be executed on a host

Step 1: Install ansible in ubuntu

1. To give root access rights su –
2. Install ansible sudo apt install ansible -y
3. **Step-by-Step to Create a Local Inventory**
    1. **Create an inventory file**:
        nano inventory.ini
    2. Type this in inventory.ini
        [local]
        localhost ansible_connection=local
    3. Press Ctrl+O to save the content and press enter.
    4. Press Ctrl+X to exit
4. To test the inventory
    ansible -i inventory.ini local -m ping

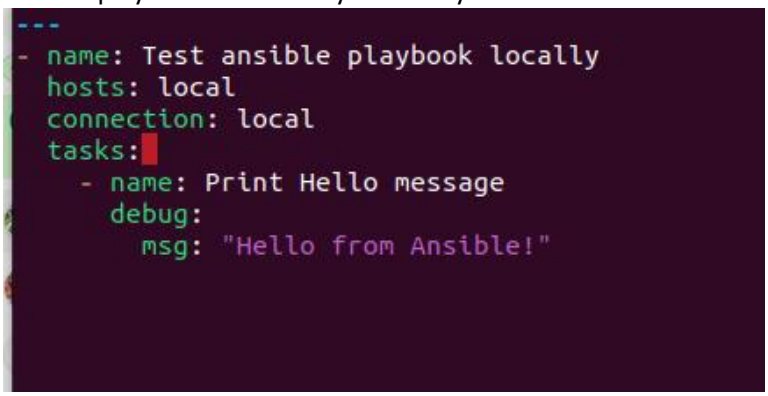
```
root@ubuntu22:~# nano inventory.ini
root@ubuntu22:~# ansible -i inventory.ini local -m ping
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

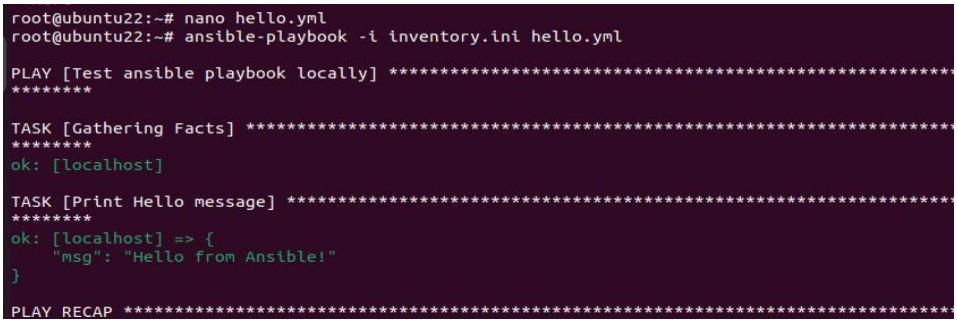
5. Create a Playbook (.yaml file)
    1. nano hello.yml

```
---
- name: Test Ansible Playbook Locally
  hosts: local
  connection: local
  tasks:
    - name: Print Hello Message
      debug:
      msg: "Hello from Ansible!"
```

6. Maintain the indent in yaml file.
7. Test the Playbook

ansible-playbook -i inventory.ini hello.yml





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

What is Azure DevOps?
Azure DevOps is a cloud-based service from Microsoft that provides developer services to support teams in planning work, collaborating on code development, and building & deploying applications.

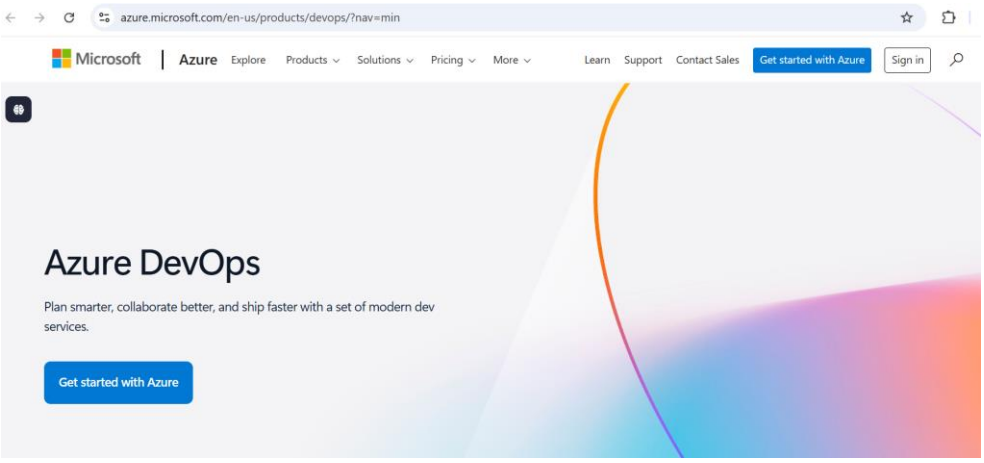
🔗 Azure DevOps Services – Overview
Azure DevOps offers the following main services:

1. Azure Repos
   - Provides Git repositories or Team Foundation Version Control (TFVC).
   - Used for source code version control.
   - Supports pull requests, branching, and code reviews.
2. Azure Pipelines
   - Used for Continuous Integration (CI) and Continuous Delivery (CD).
   - Supports building, testing, and deploying code automatically to any platform.
3. Azure Boards
   - Agile planning tools: Kanban boards, backlogs, and dashboards.
   - Helps manage tasks, bugs, and user stories.
4. Azure Test Plans
   - Provides manual and exploratory testing tools.
   - Helps ensure software quality and coverage.
5. Azure Artifacts
   - Manages package dependencies (e.g., NuGet, npm, Maven).
   - Allows teams to create, host, and share packages.

Setting Up an Azure DevOps Account and Project

✅ **Step 1: Create a Microsoft Account (if not already available)**

- Visit: https://signup.live.com
- Use this account to sign in to Azure DevOps.



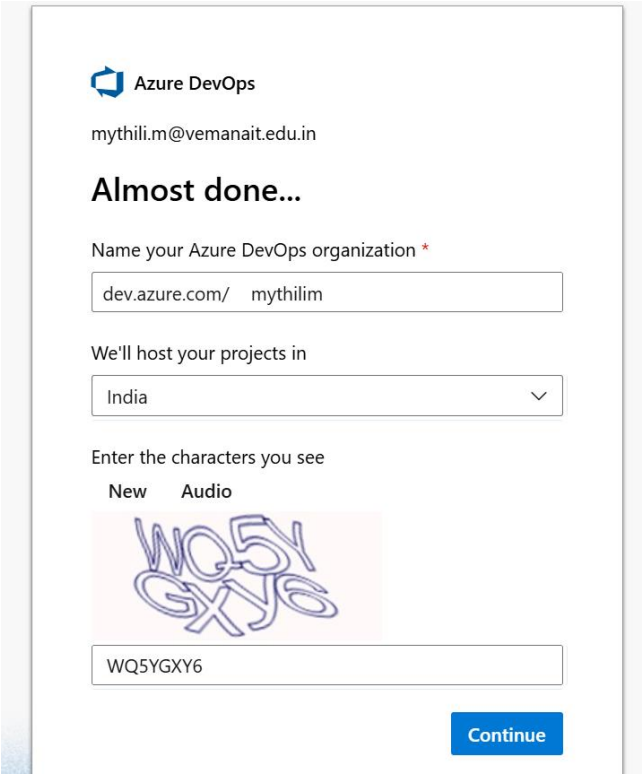
- Signin with college mailid. Authentication with one time password should be done using Authenticator app.

Step 2: Sign in to Azure DevOps

- Go to: https://dev.azure.com
- Use your Microsoft credentials to log in.

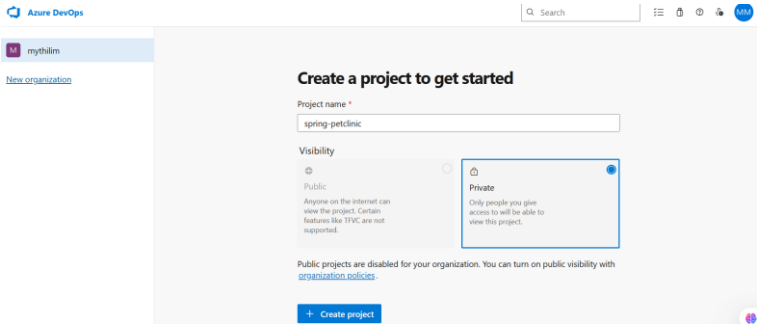
Step 3: Create a New Organization

- An **organization** in Azure DevOps groups related projects.
- Click on **"New Organization"**, provide a name, select a region.



Step 4: Create a Project

- Inside your organization, click **"New Project"**.
- Fill in:
  - **Project name**
  - **Visibility**: Public or Private



Step 5: Explore Services

- After project creation, you can start using:
  - **Repos** for code
  - **Boards** for task tracking
  - **Pipelines** for CI/CD
  - **Artifacts** for packages
  - **Test Plans** for testing