TABLE OF CONTENTS

Ex.No:1 **Create Maven Build Pipeline in Azure**

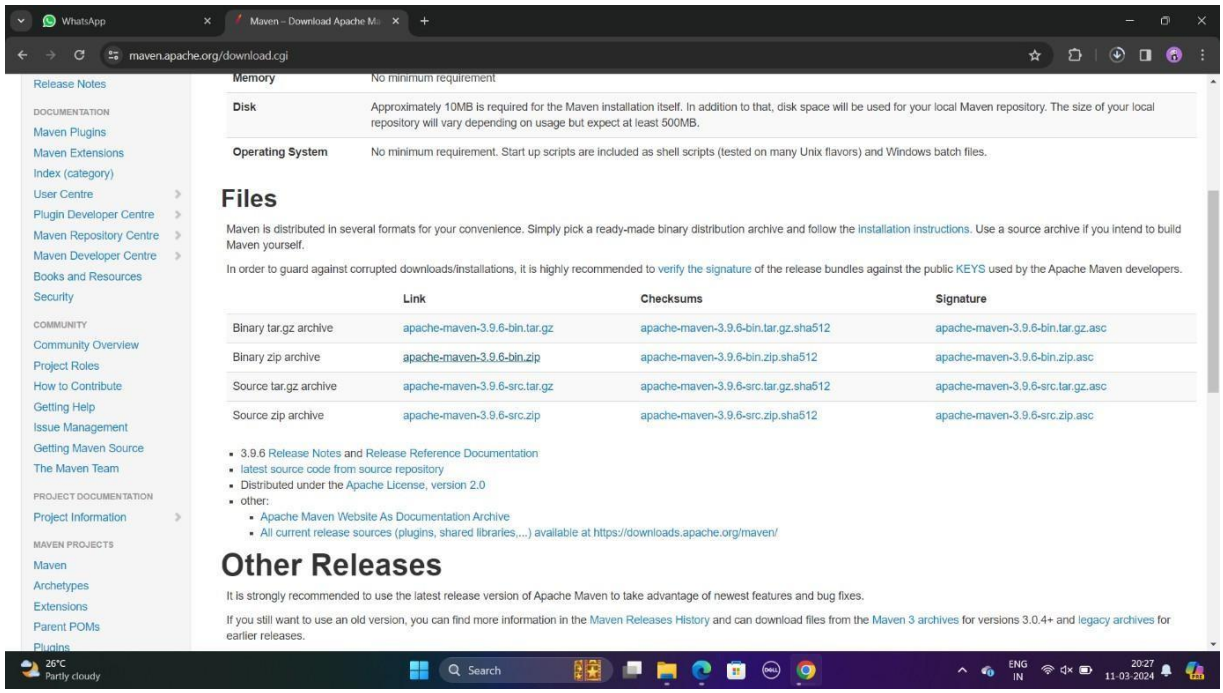
Date:

Aim:

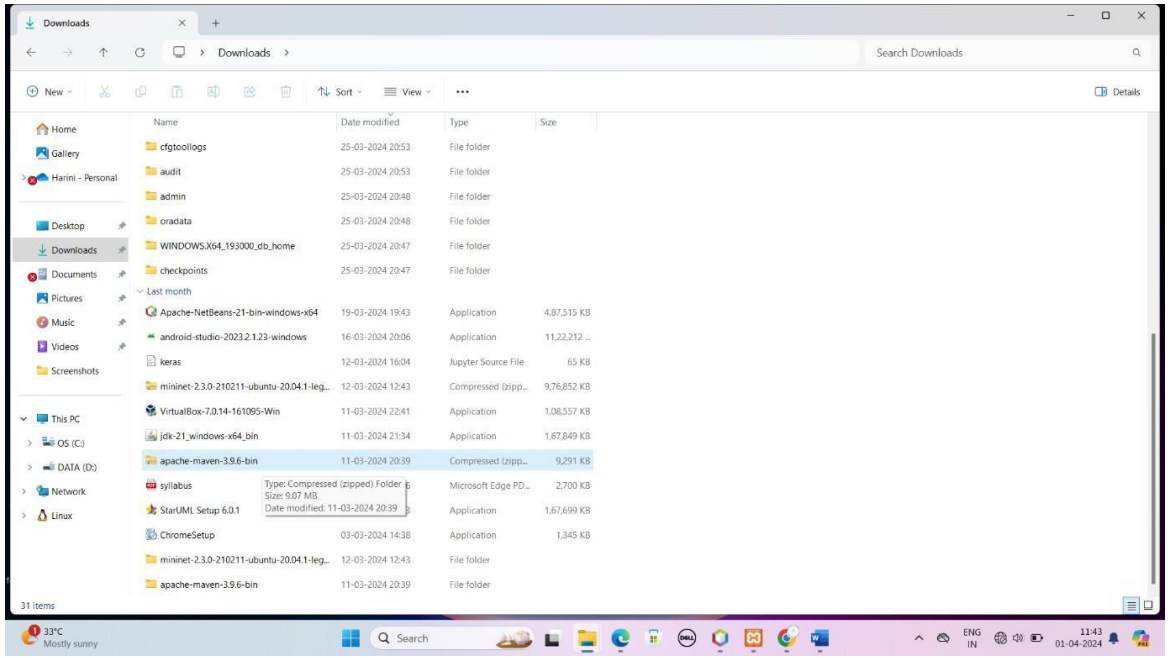
To create maven build pipeline in Azure

Procedure:

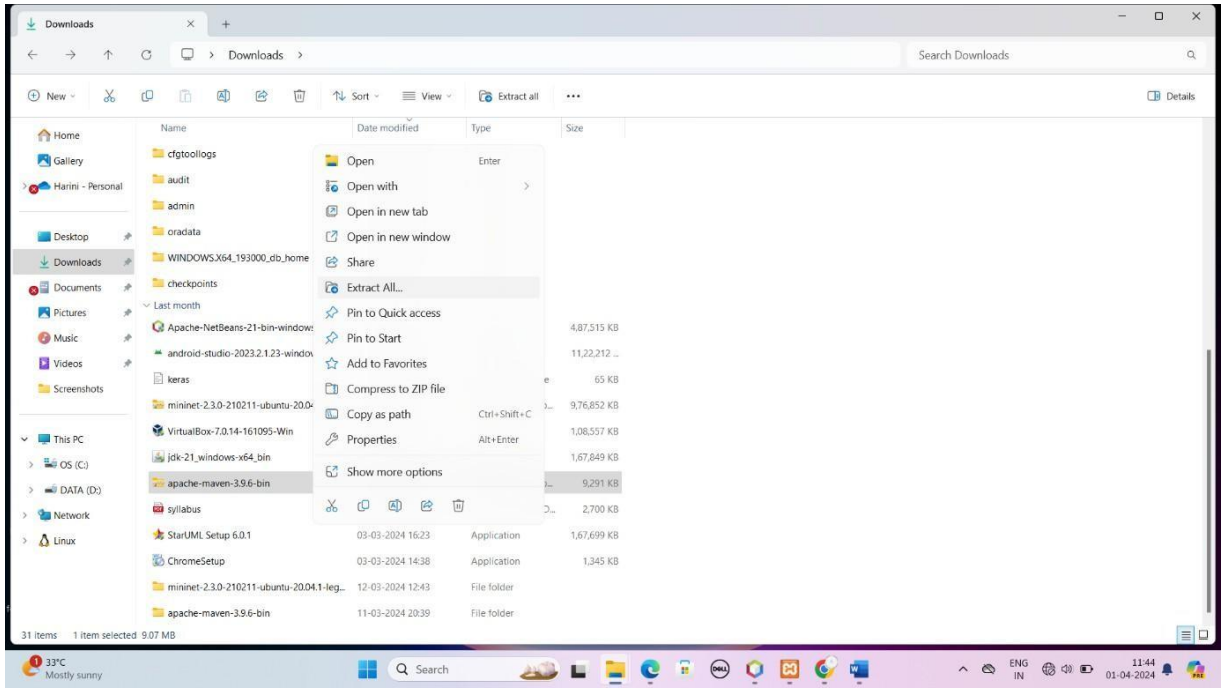
Step 1: Install Apache Maven Zip file



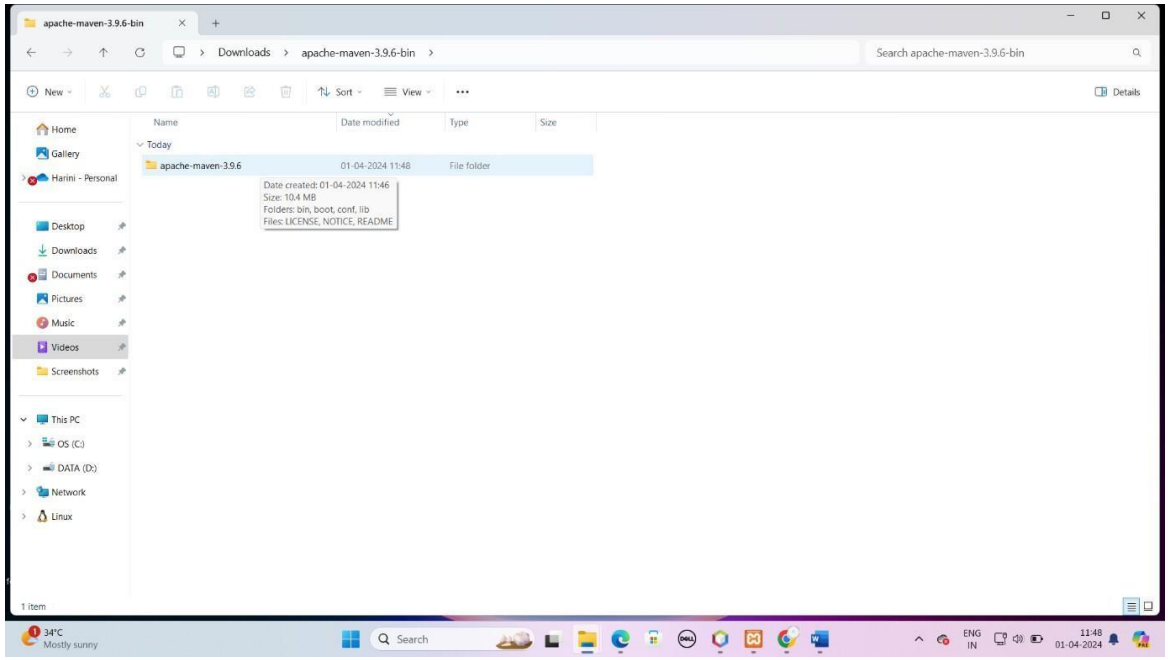
Step 2: Go to folder where apache maven downloaded



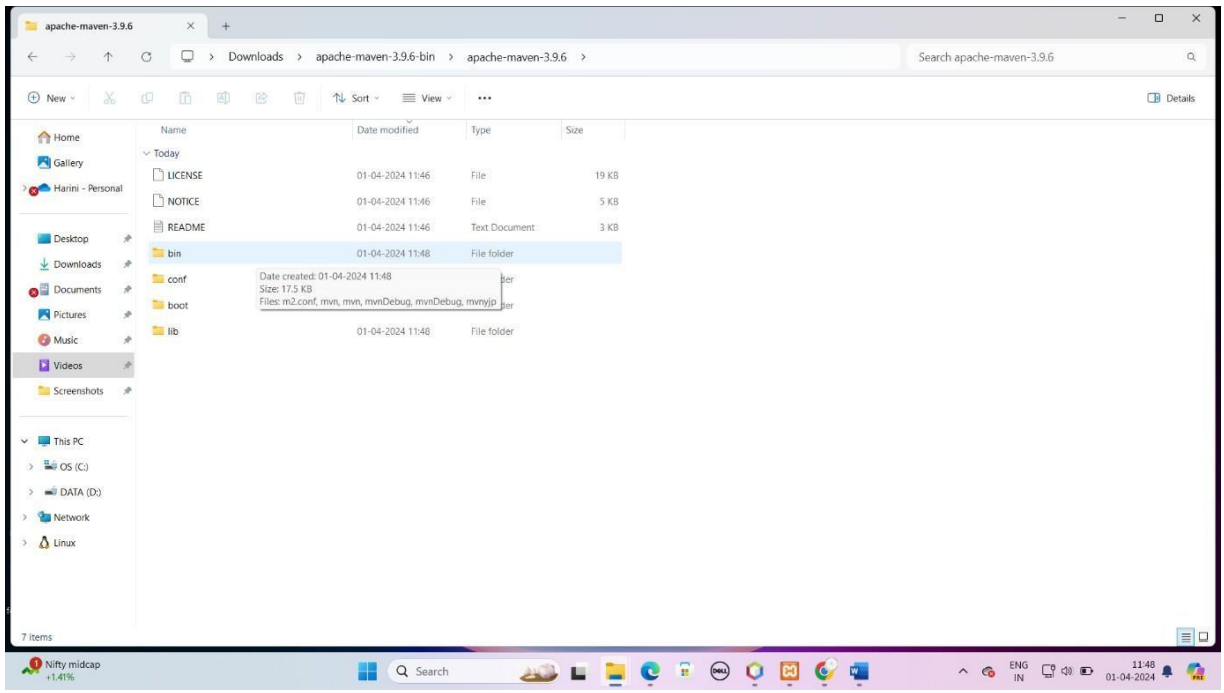
Step 3: Right Click & Extract the Zip file



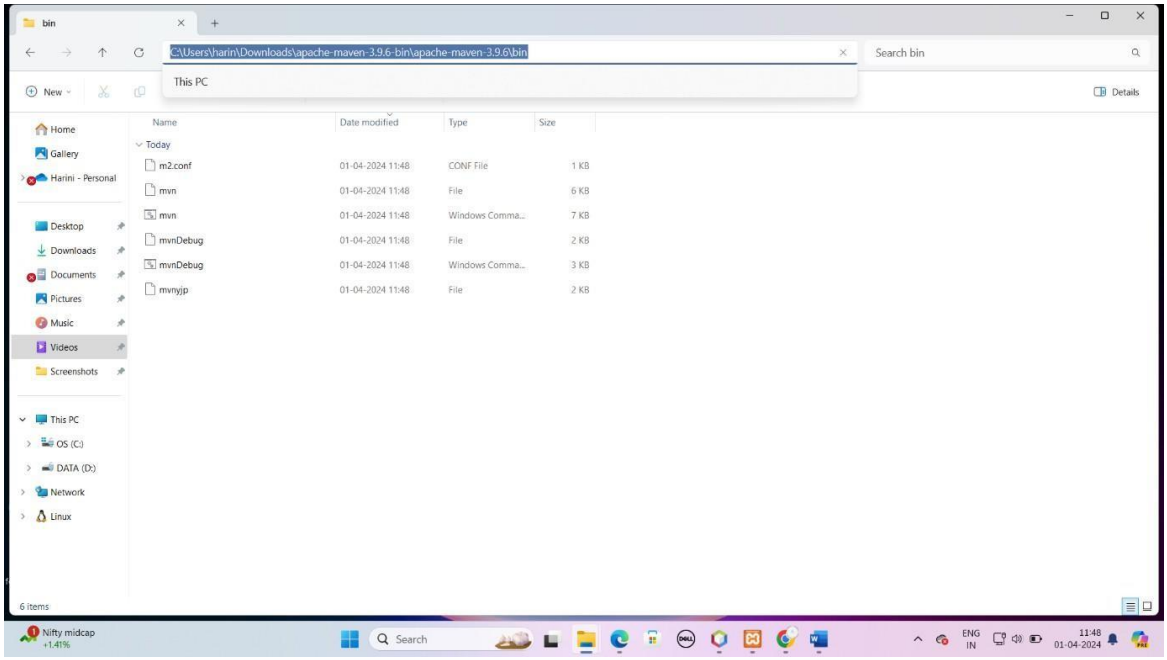
Step 4: Click the Extracted file



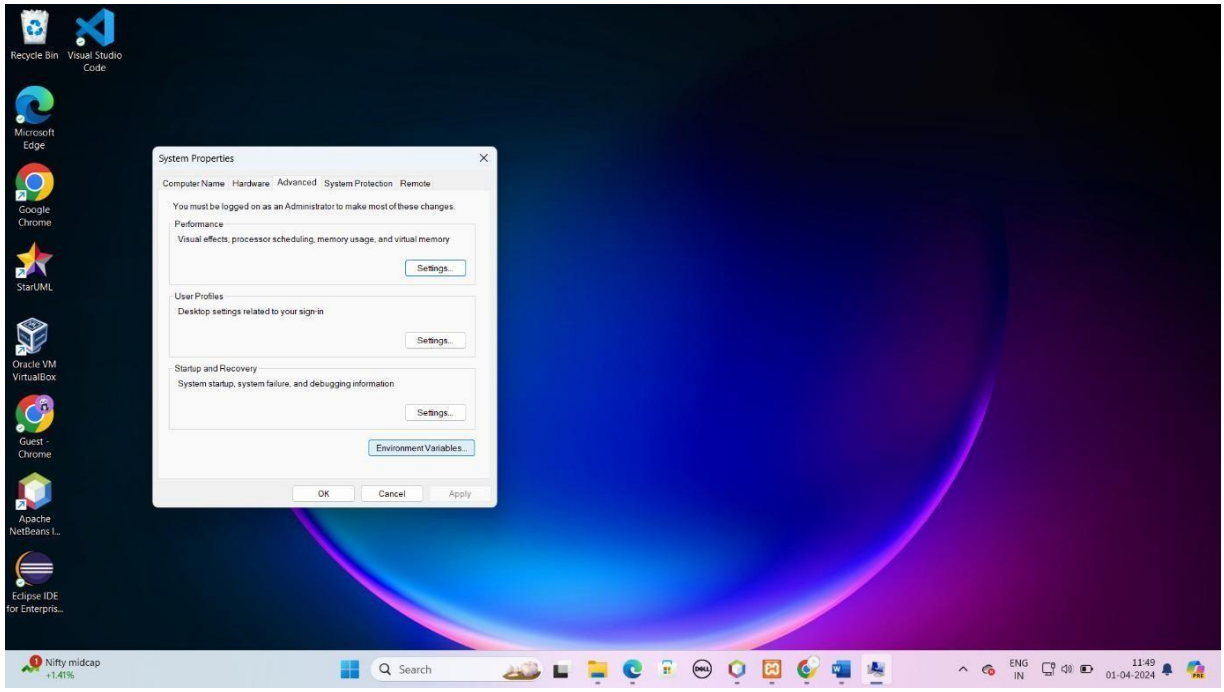
Step 5: Select bin folder



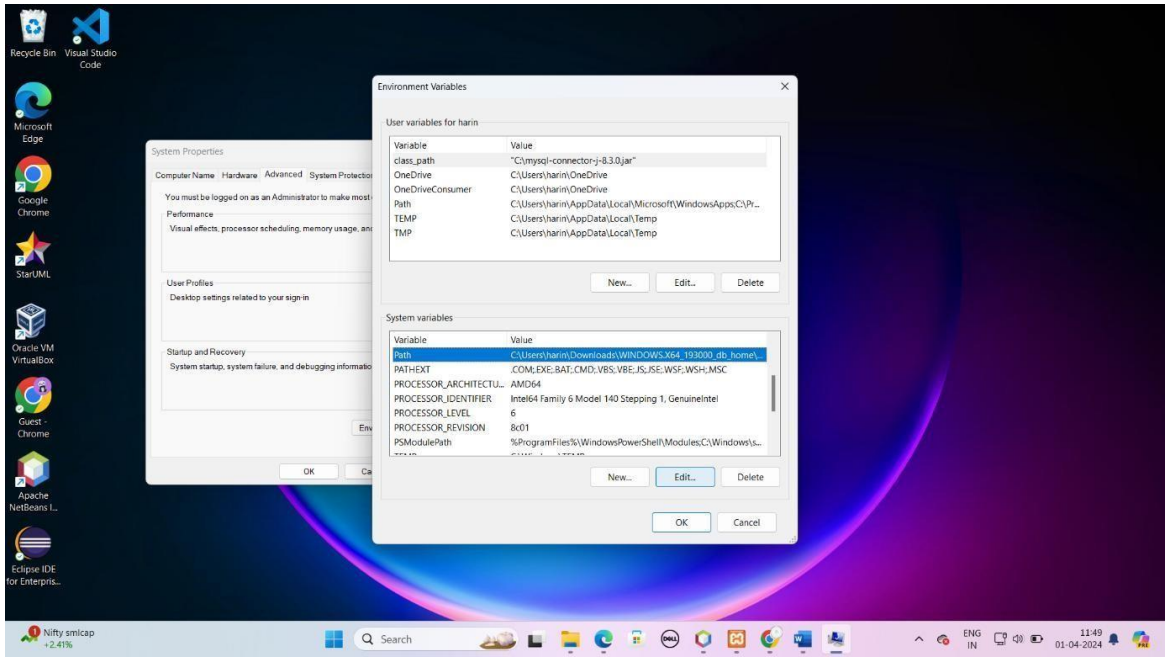
Step 6: Copy bin folder path



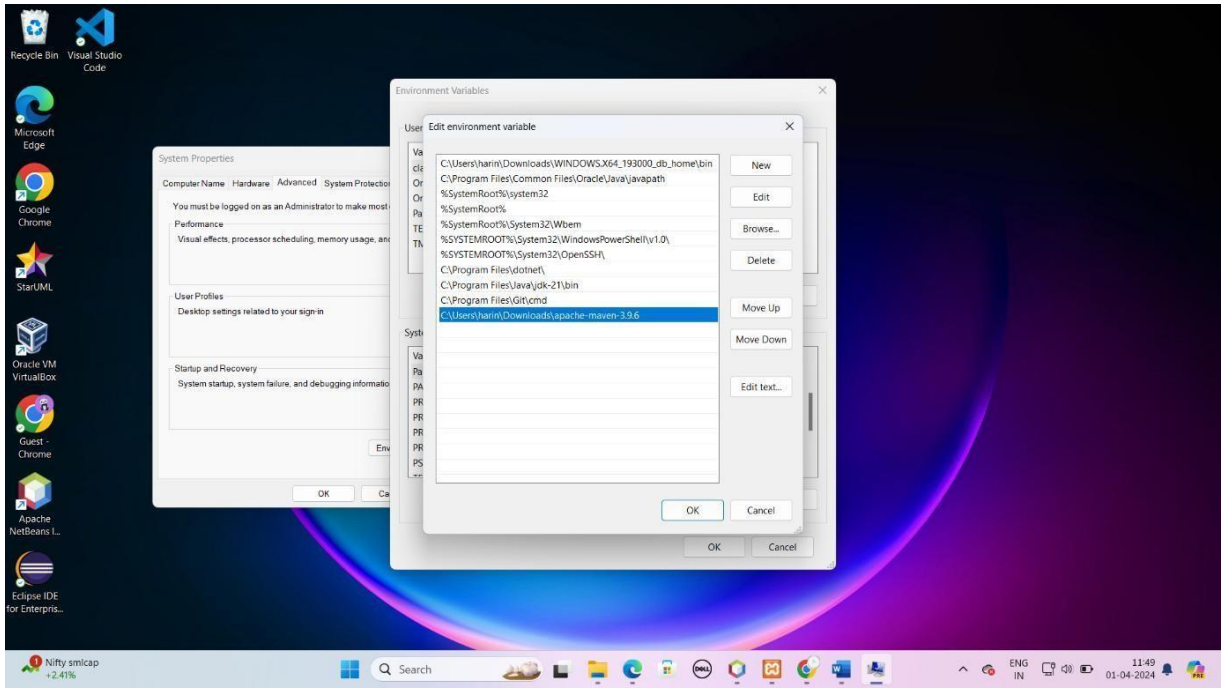
Step 7: Open Environmental Variable



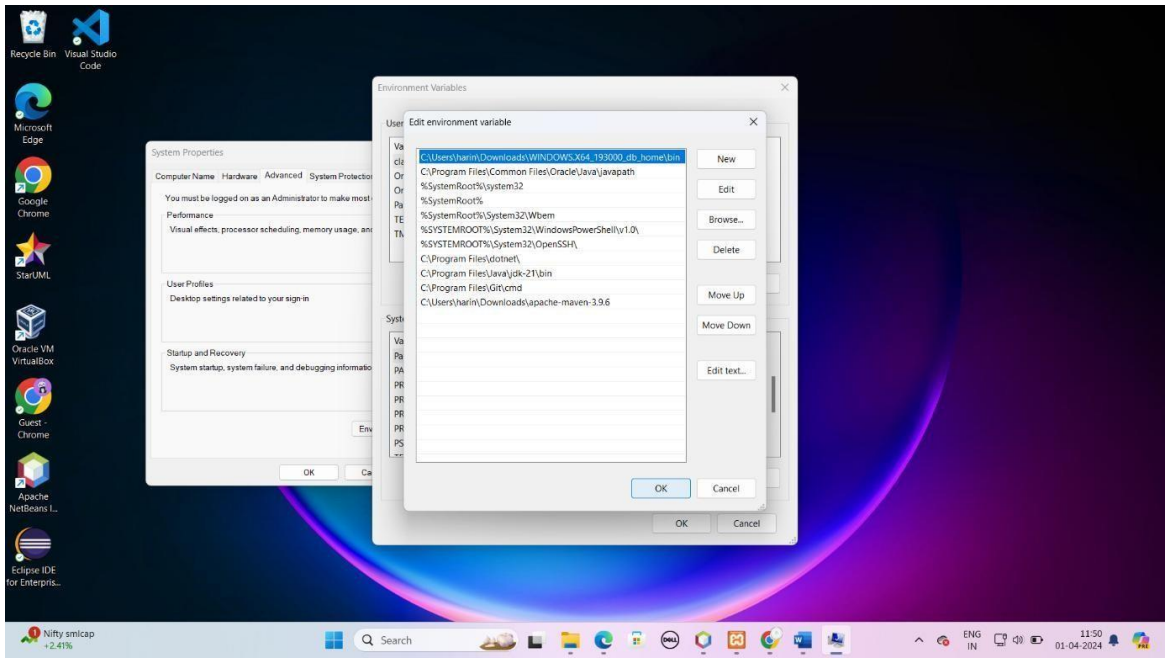
Step 8: Select path and click edit



Step 9: Click New and paste the copied path

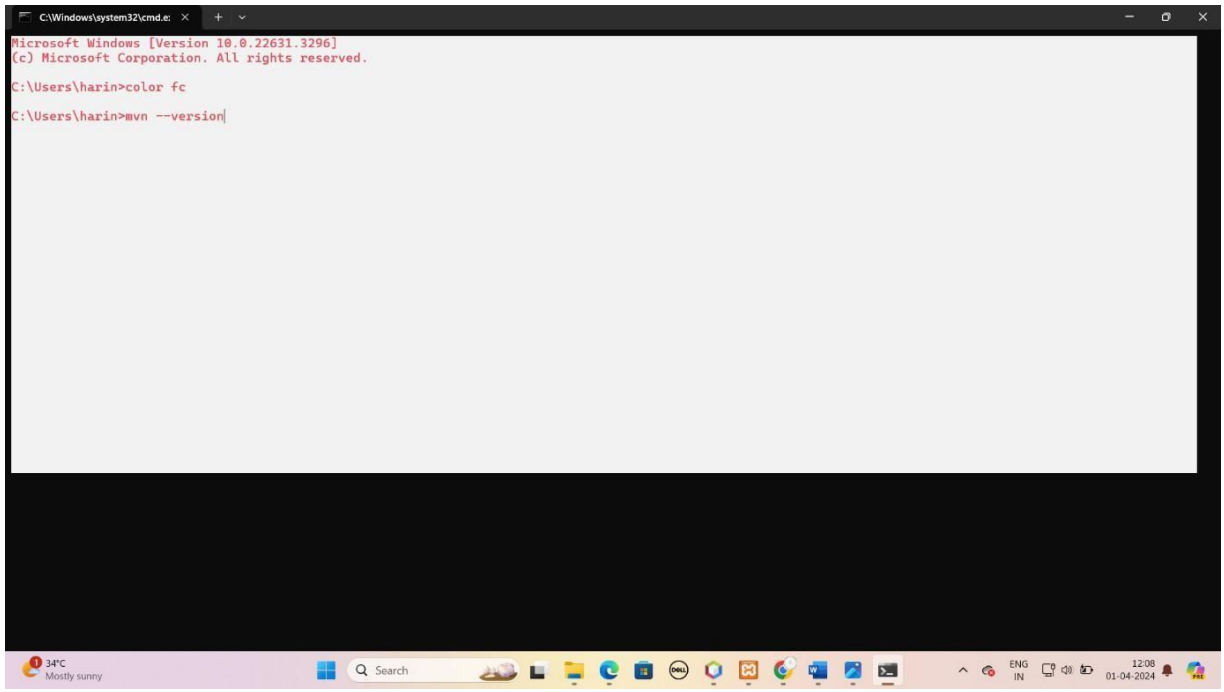


Step 10: Click ok

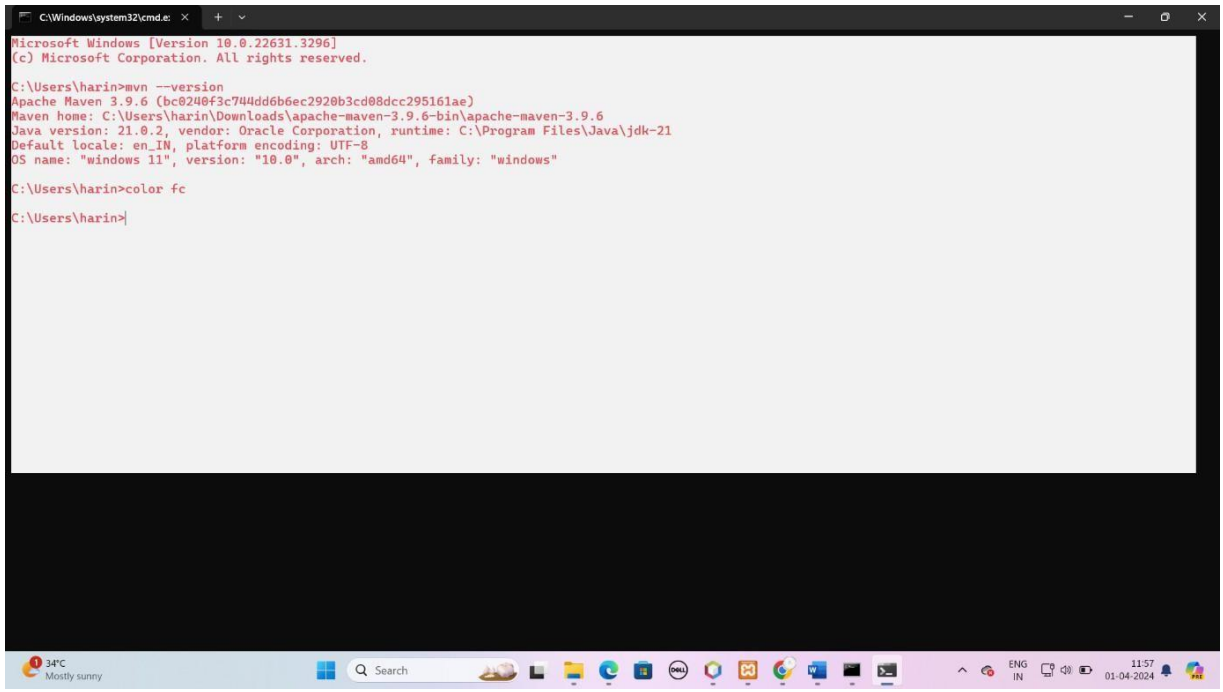


Step 11: Open Command Prompt

>> mvn --version



Step 12: Maven Verified



Result:

Thus, creating maven build pipeline in Azure done successfully.

Ex.No: 2 **Run regression tests using Maven Build pipeline in Azure**

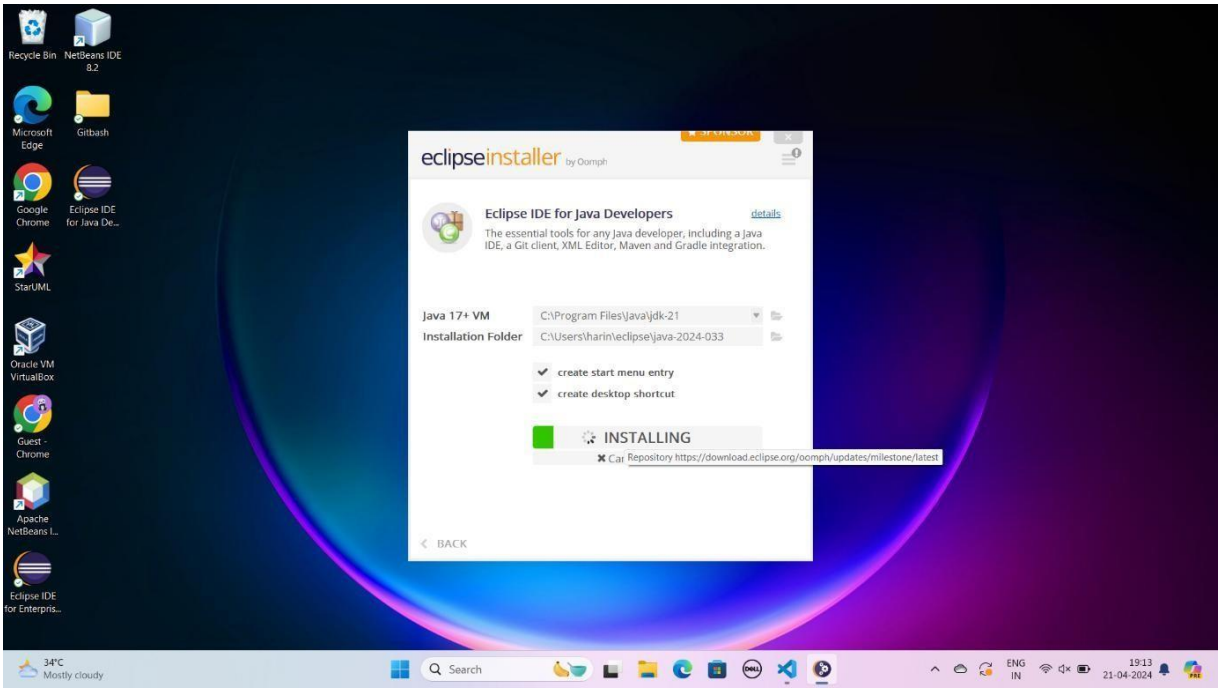
Date:

Aim:

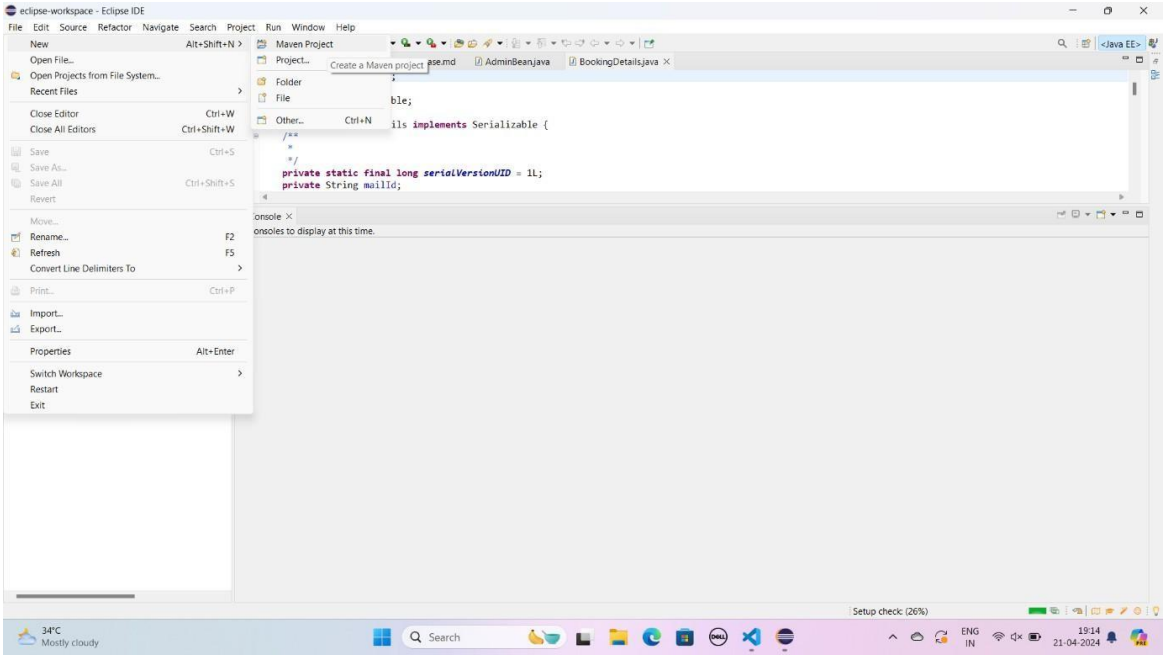
To run regression tests using Maven Build pipeline in Azure.

Procedure:

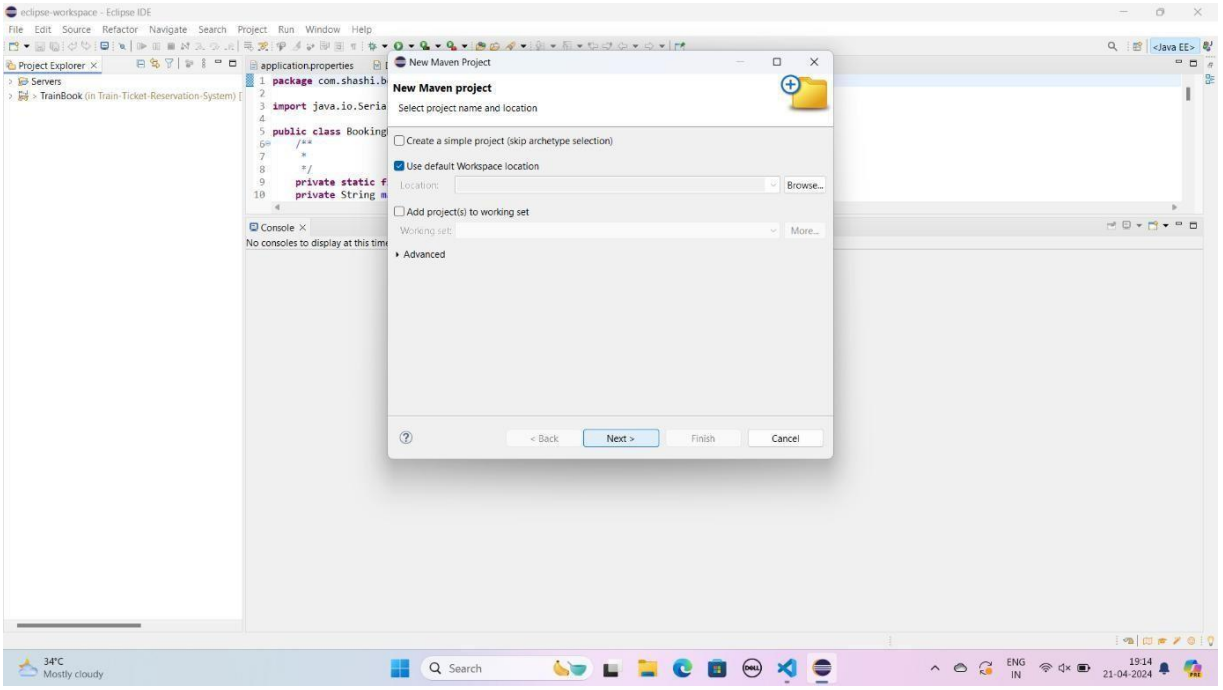
Step 1: Download Eclipse and install Eclipse IDE for java Developers



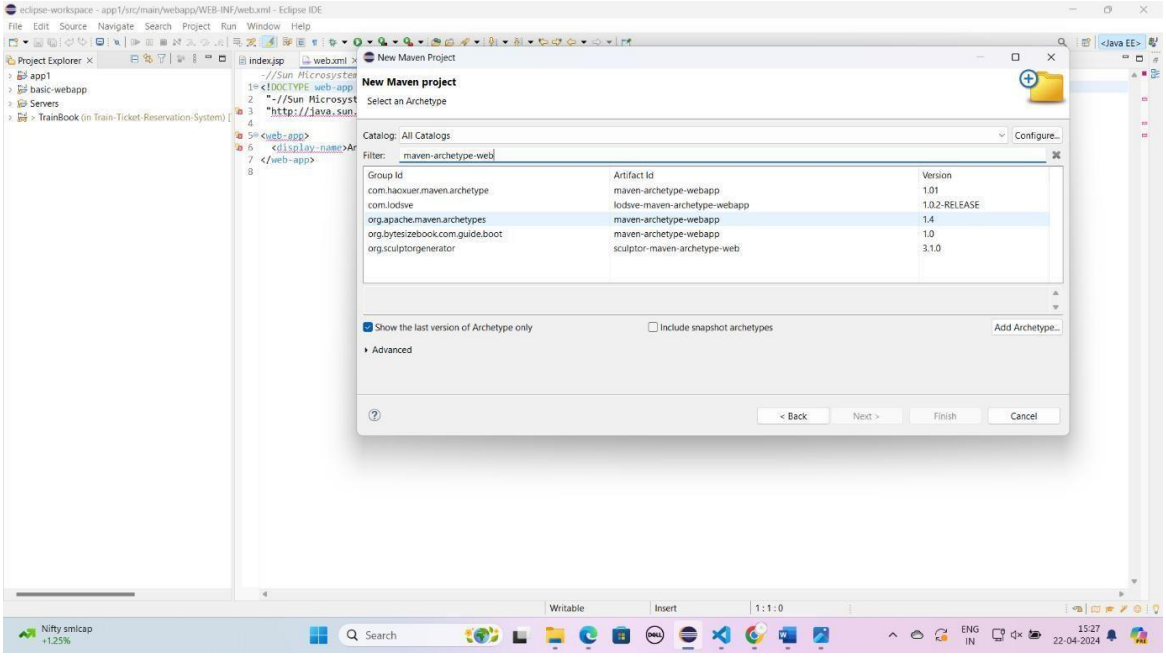
Step 2: Open file select new and create maven project



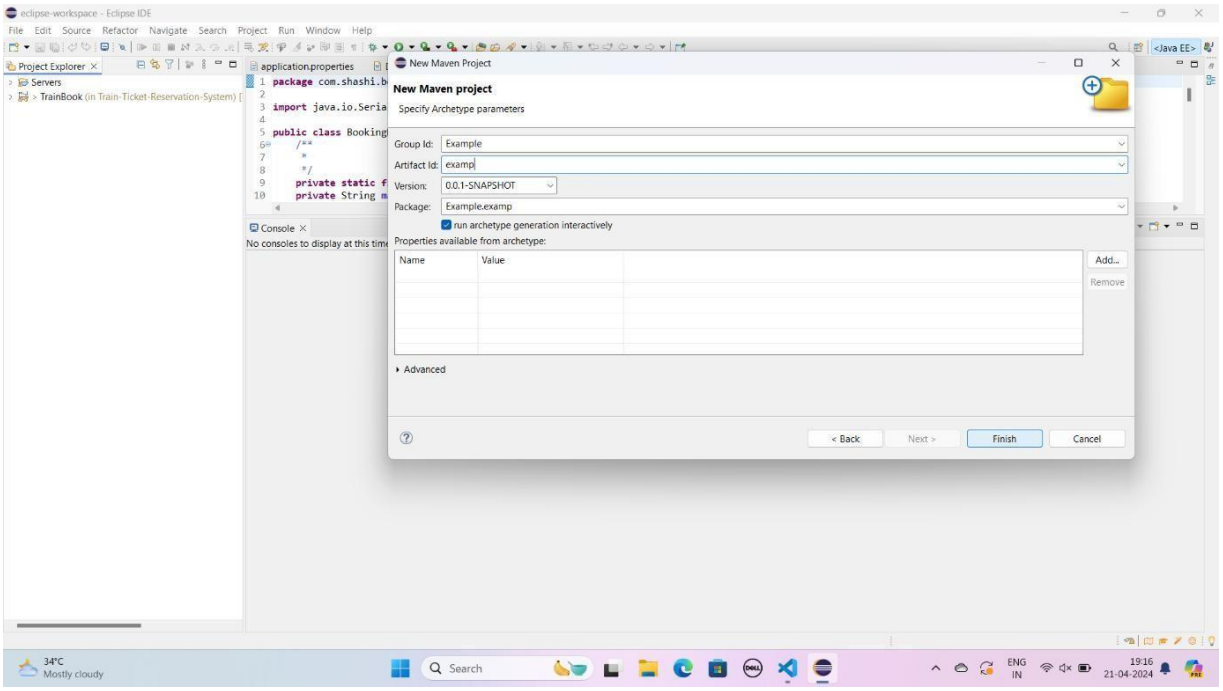
Step 3: Just Click next



Step 4: In New Maven project select Maven-archetype-webapp Version 1.4 and Click next



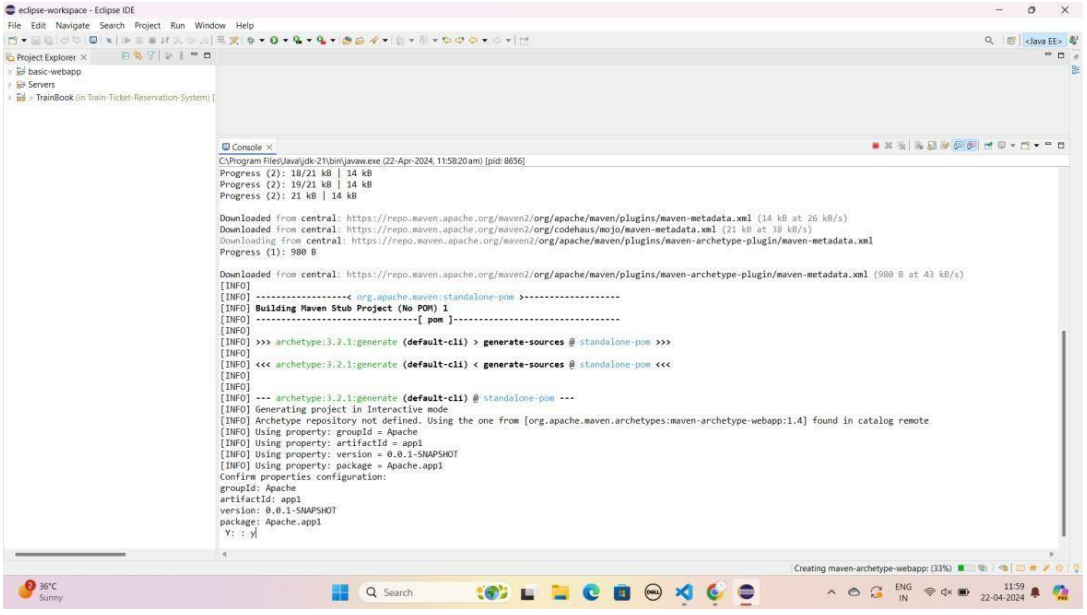
Step 5: Provide the Group Id and Artifact Id then click finish



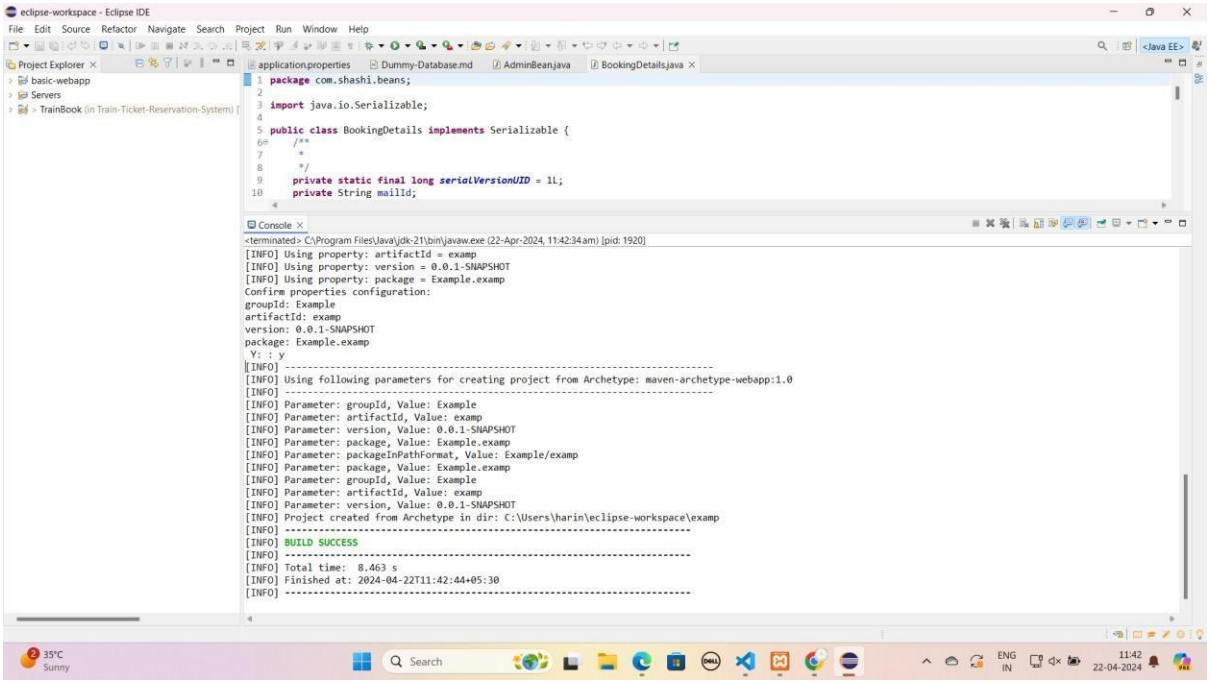
Step 6: Creating maven-archetype-webapp

At 33% completion it will ask Y :: y

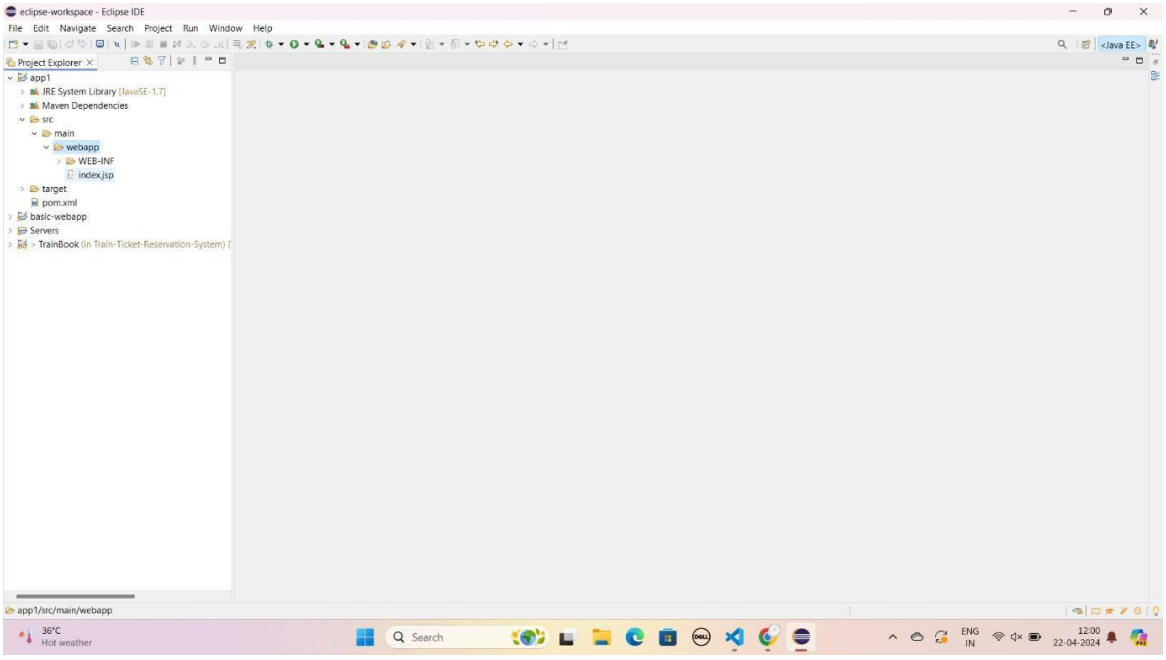
type y and it will continue the process



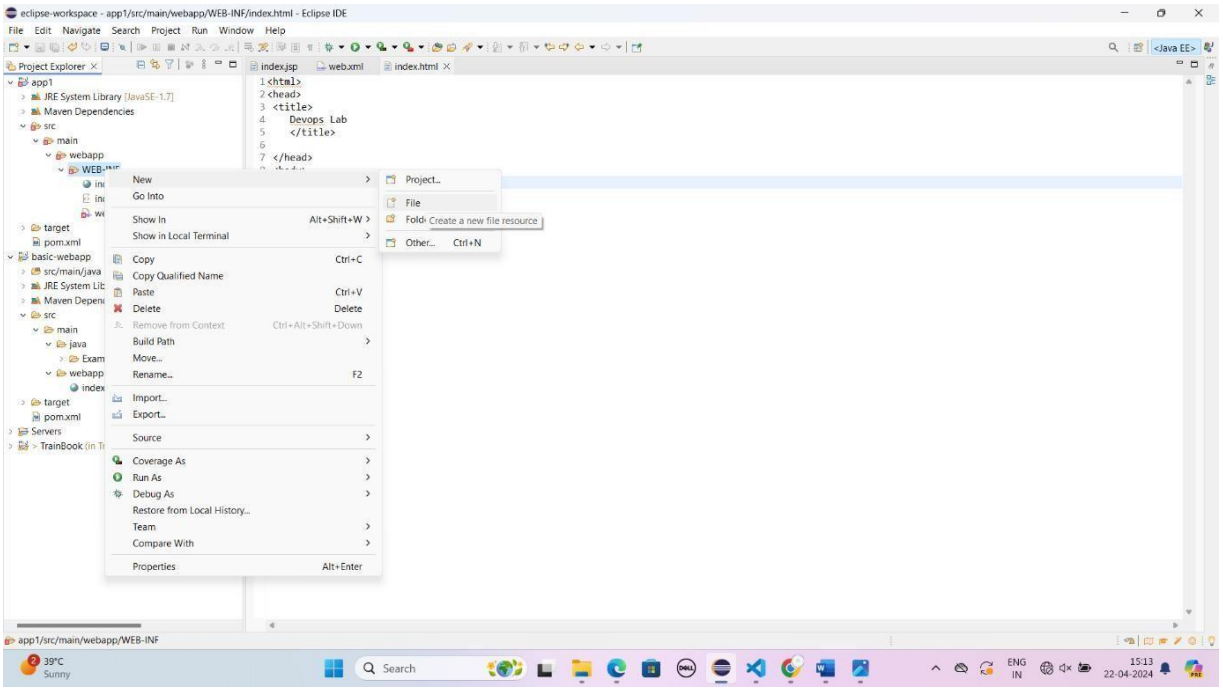
Step 7: It Shows Build Success



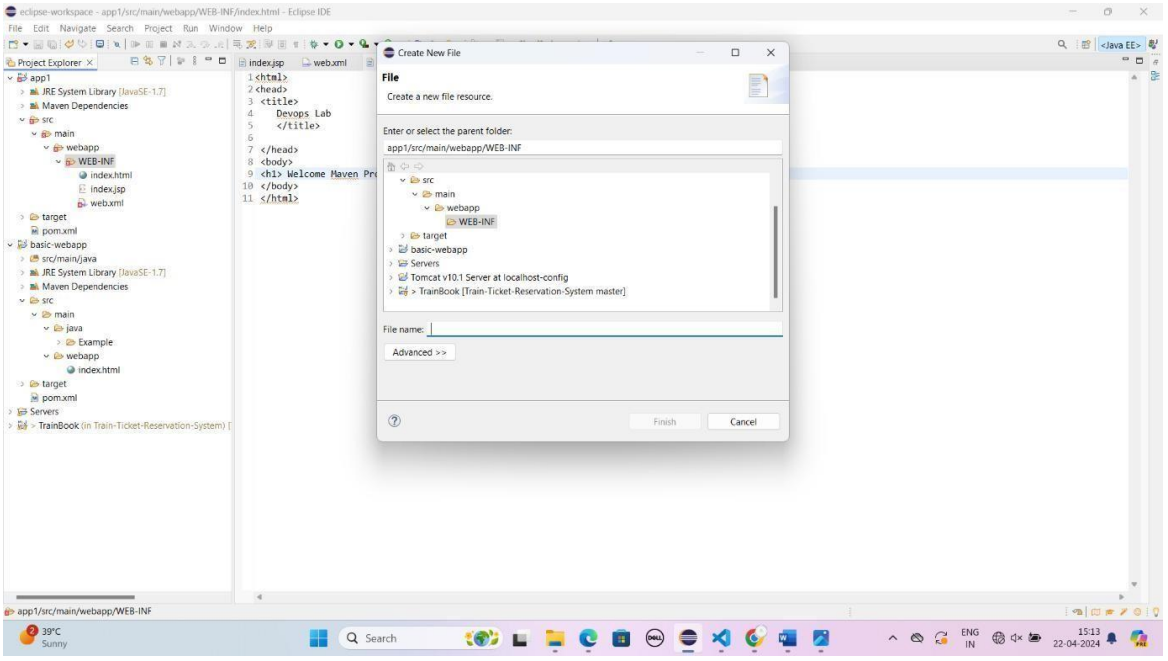
Step 8: Click app1 -> src -> main -> webapp -> WEB –INF



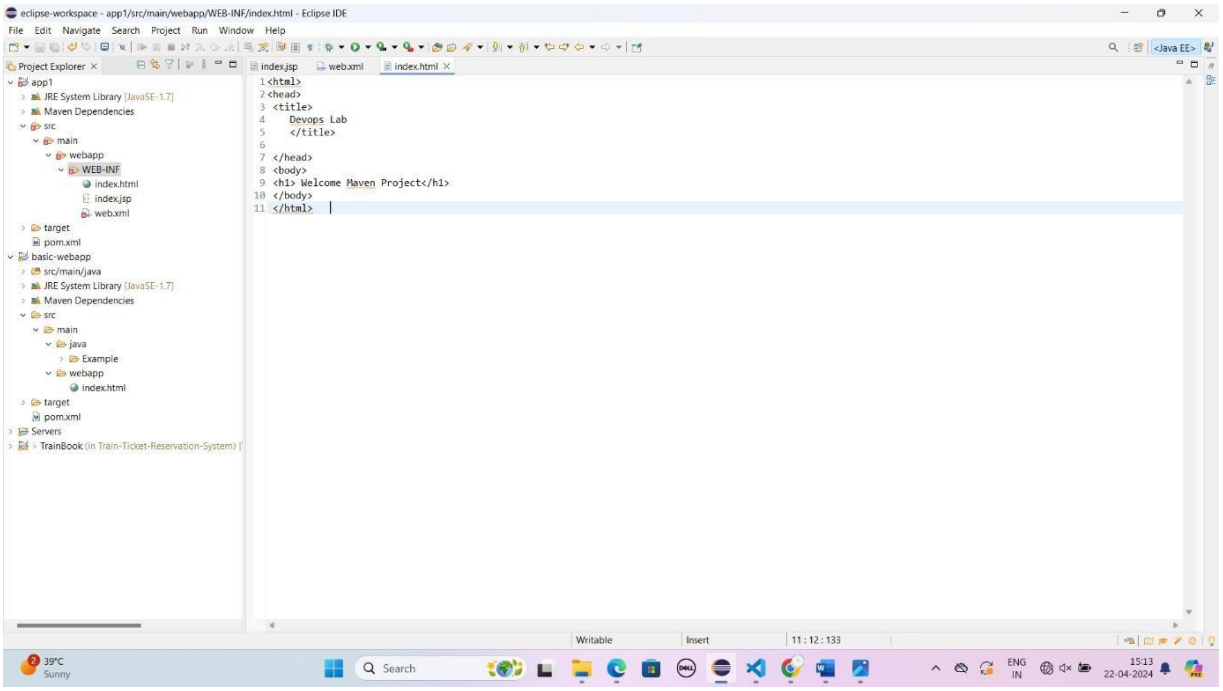
Step 9: Right click on WEB-INF select new and click on file



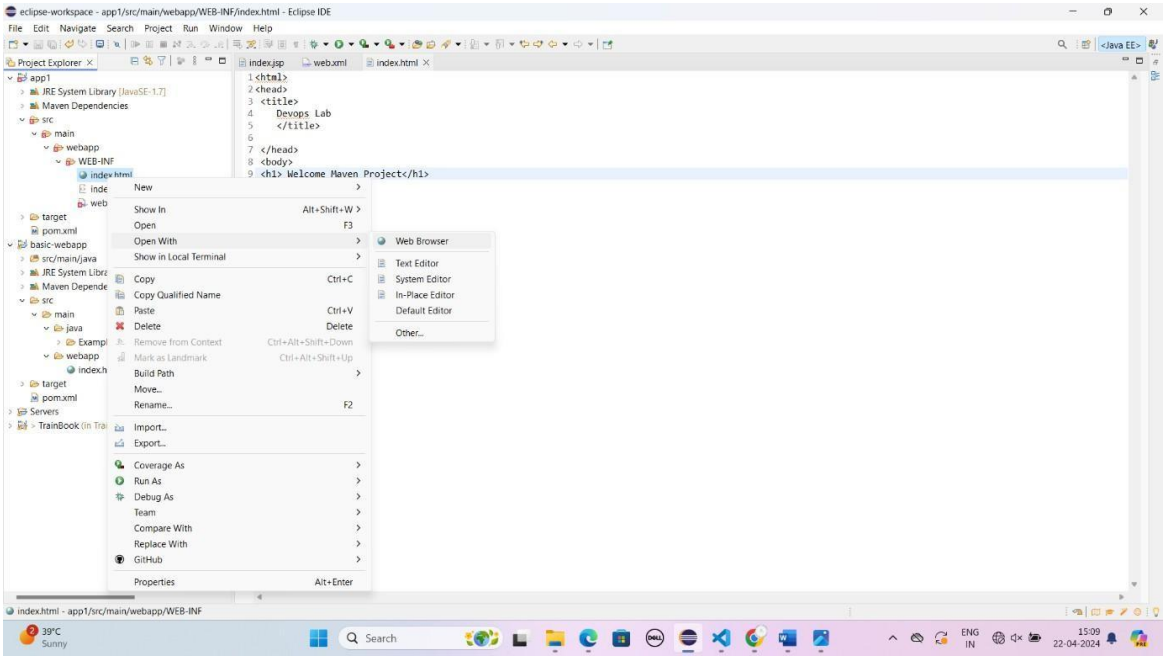
Step 10: create new file and click finish



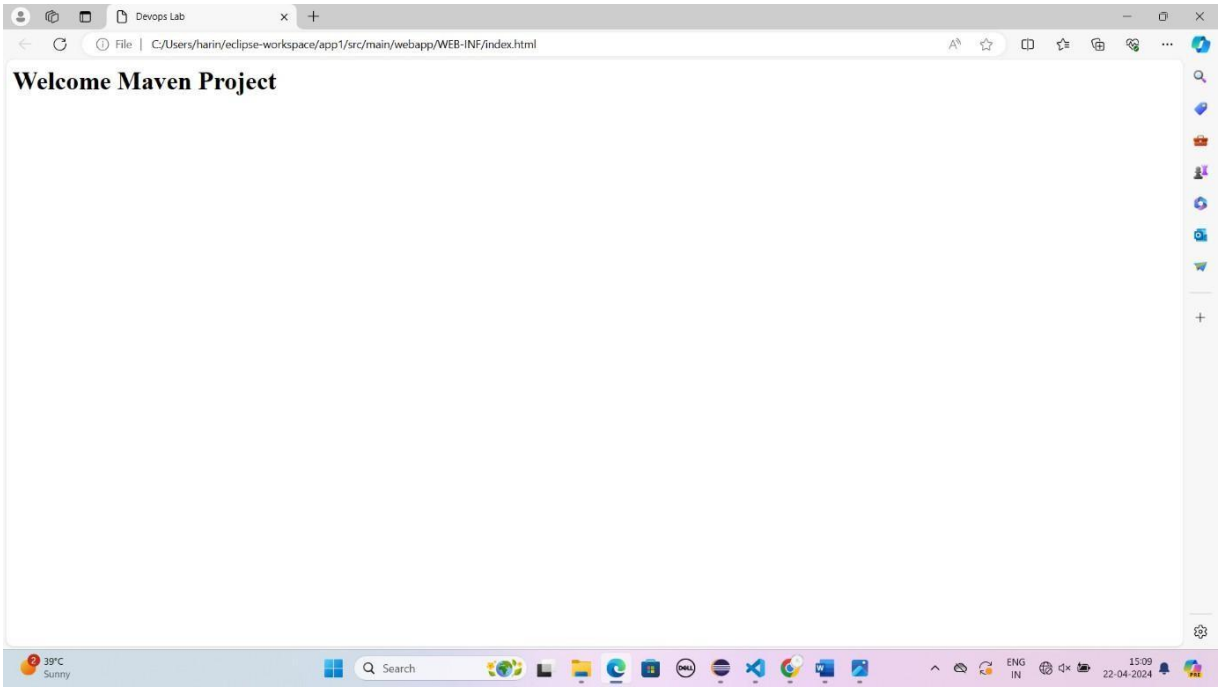
Step 11: Create a simple HTML program



Step 12: Run html program in web browser



Step 13: Output Verified



Result:

Thus regression tests using Maven Build pipeline in Azure done successfully.

Ex.No:3 **Install Jenkins in Cloud**

Date:

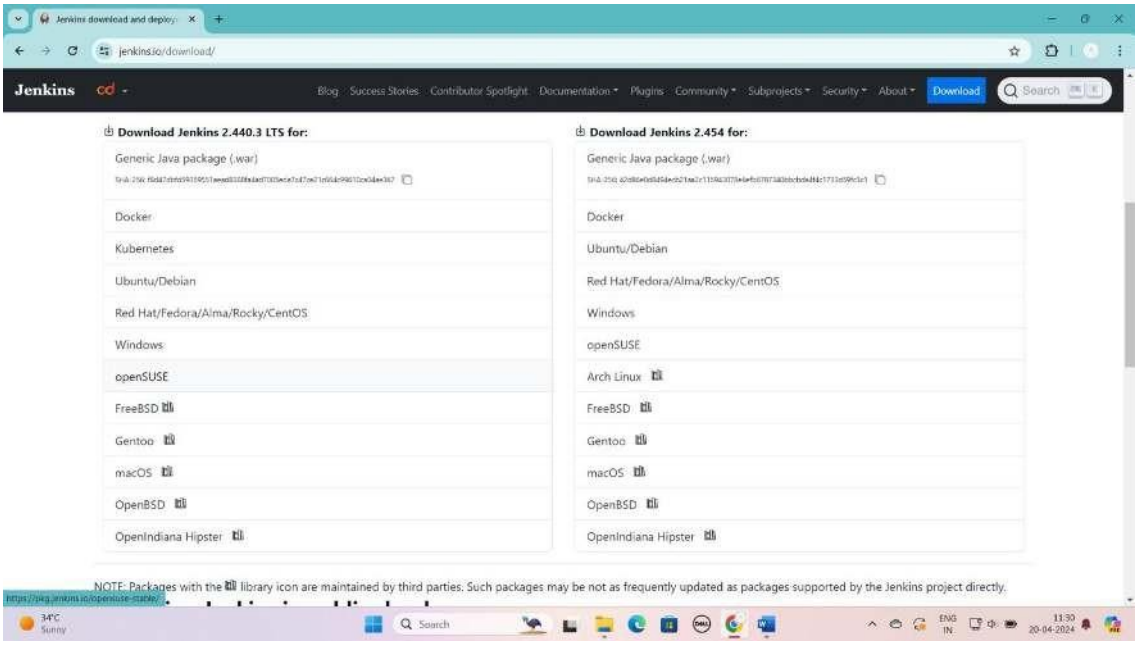
Aim:

To install Jenkins in Cloud.

Procedure:

Step:1 Download Jenkins from jenkins official page

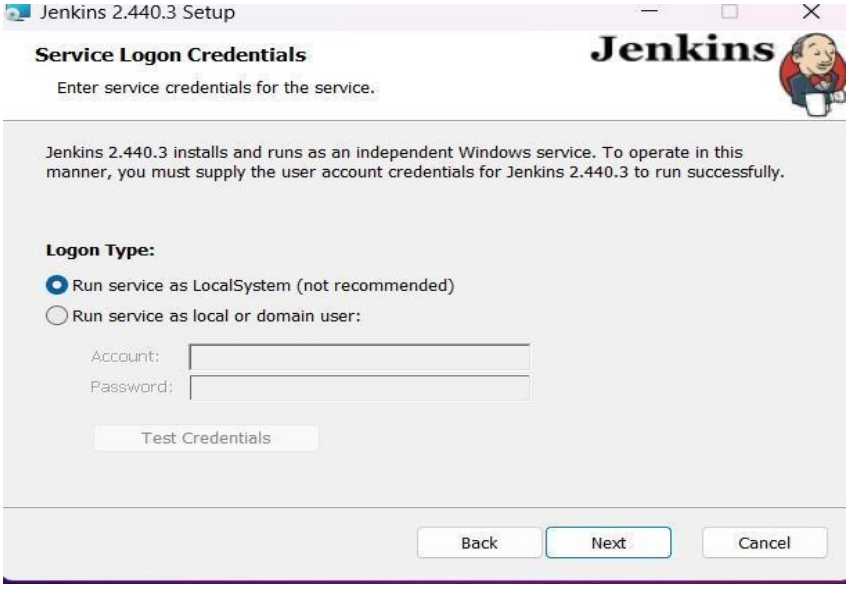
Note: Java JDK version must be 11 / 17/ 21 to install Jenkins on windows



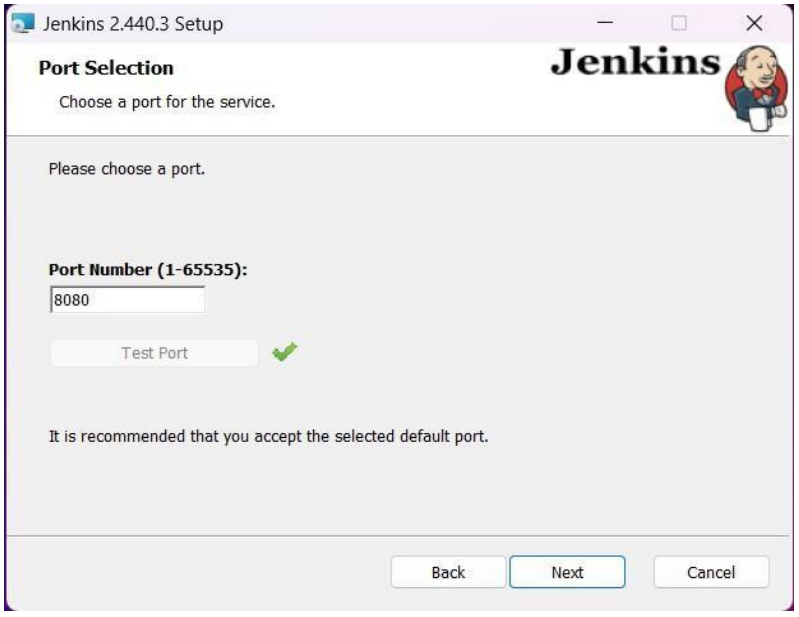
Step:2 Click the downloaded file and run as administrator

Step:3 Click next to install

Step:4 Enter service credentials for the service and click next

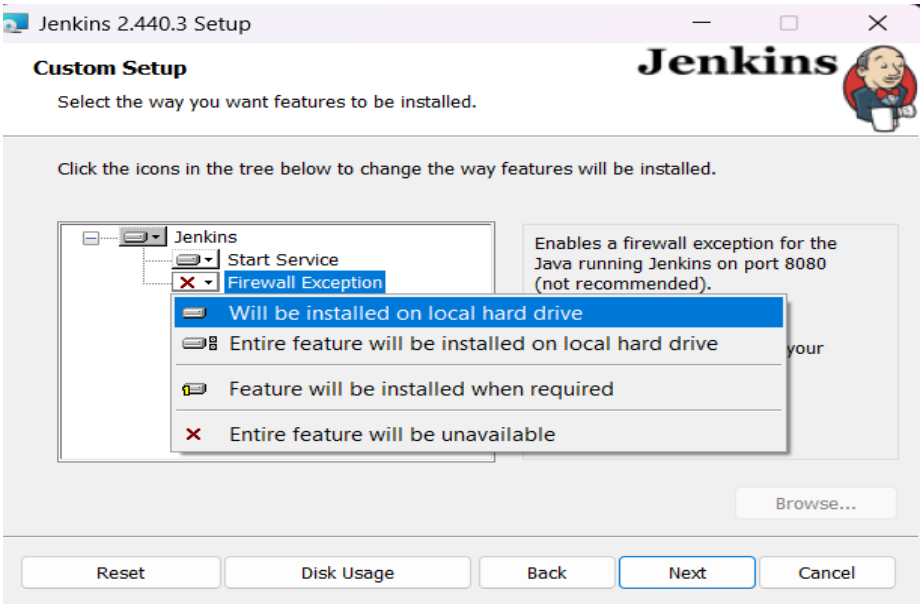


Step:5 Choose a port for the service and click next



Step:6 Select java home directory and click next

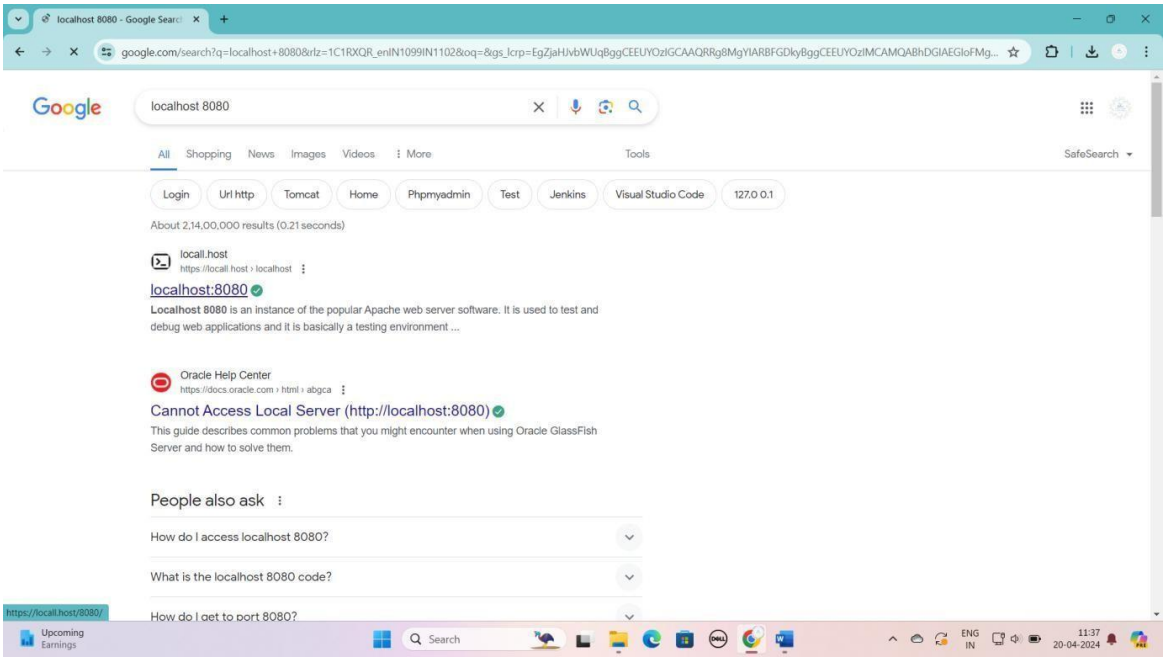
Step:7 Select the features to be installed and click next



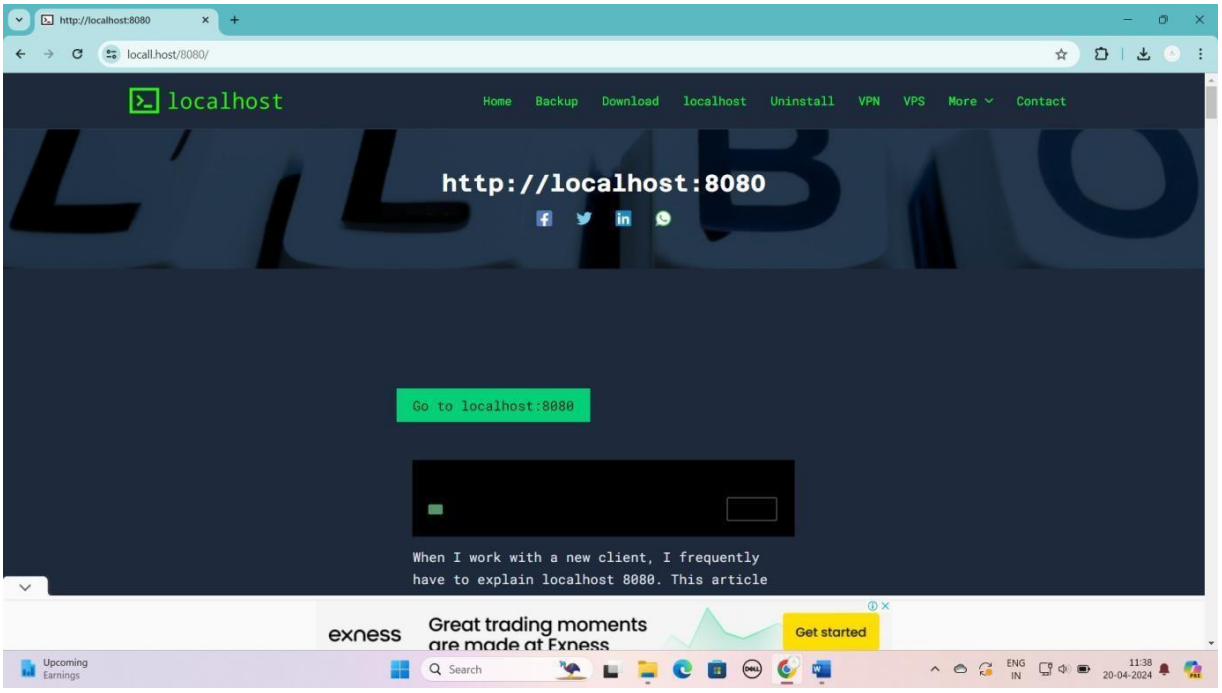
Step:8 Click install to begin the installation

Step:9 Click finish the button to exit the setup

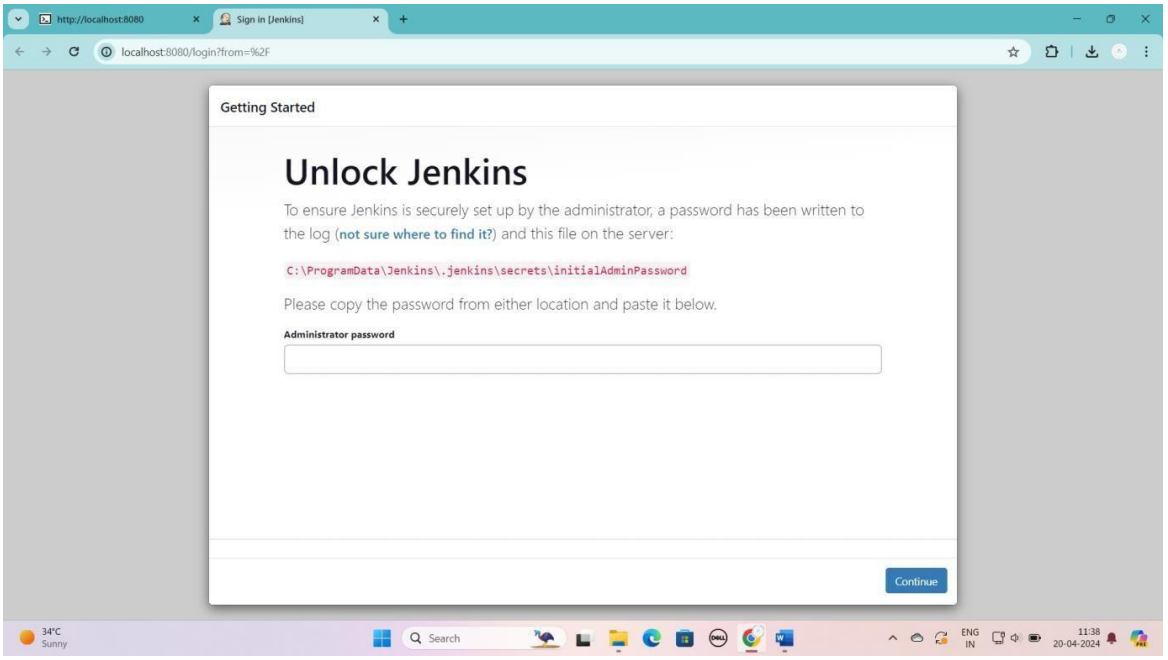
Step:10 Goto google and search locolhost 8080



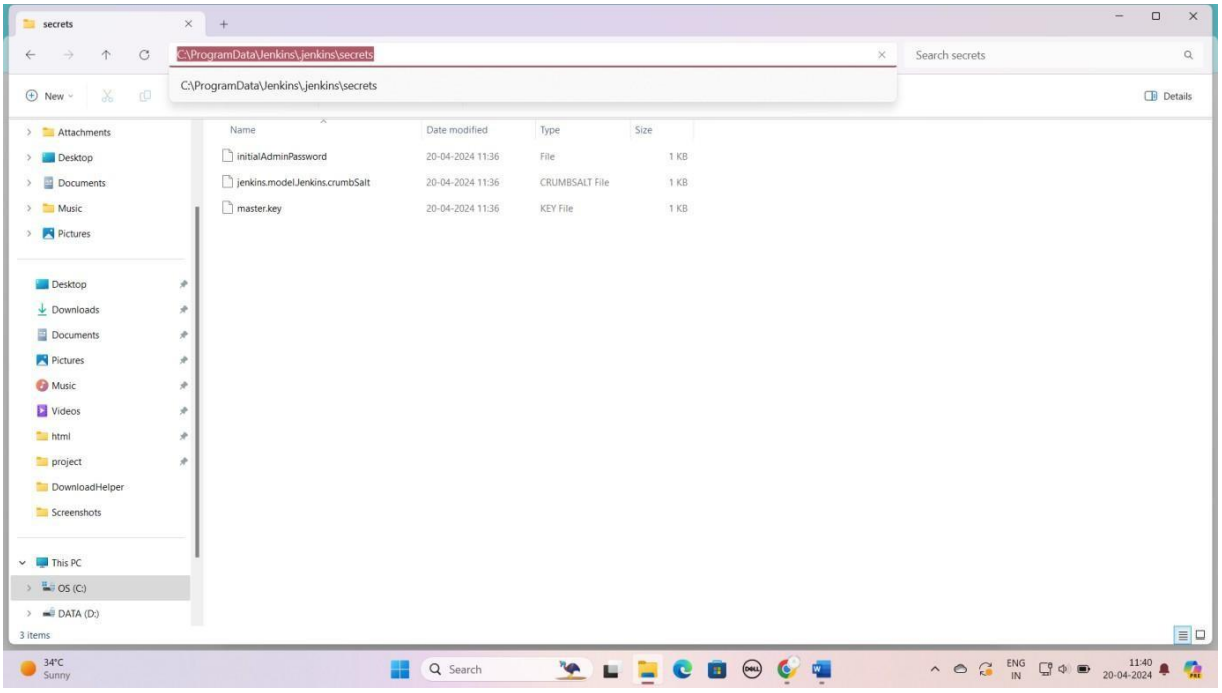
Step:11 Opening localhost 8080



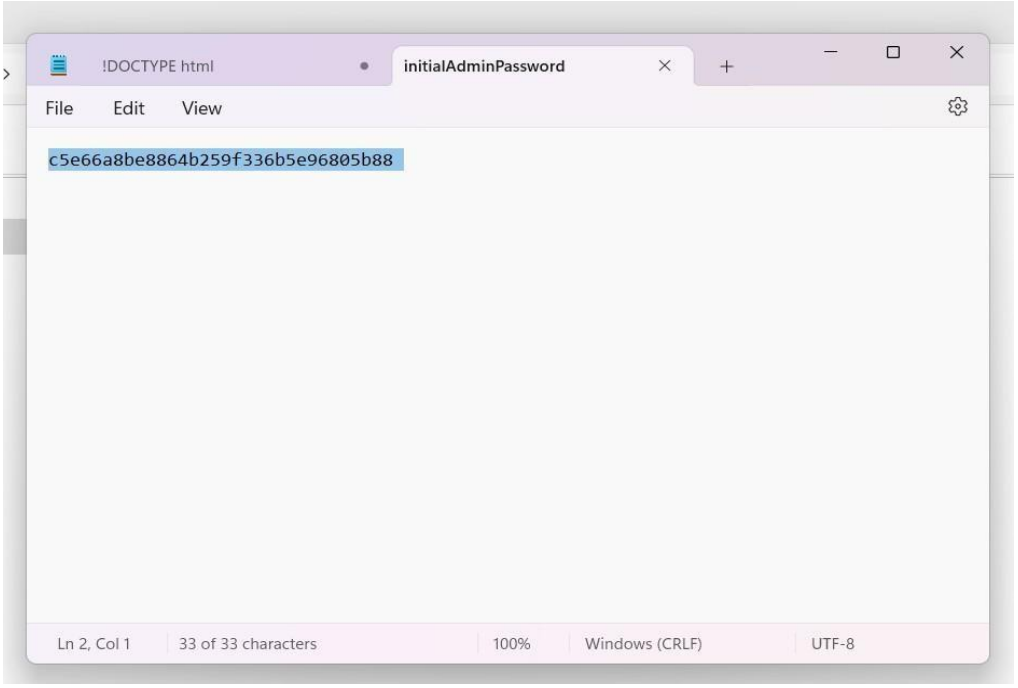
Step:12 Unlock jenkins



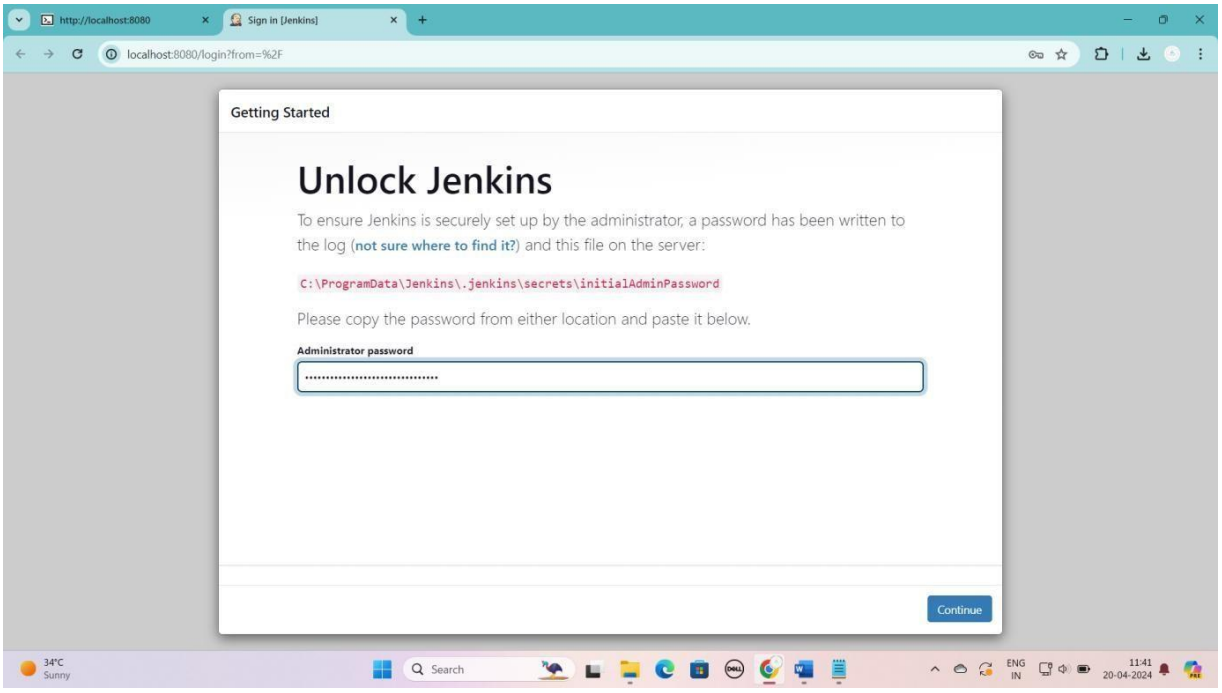
Step:13 Goto jenkins folder and search for intial admin password



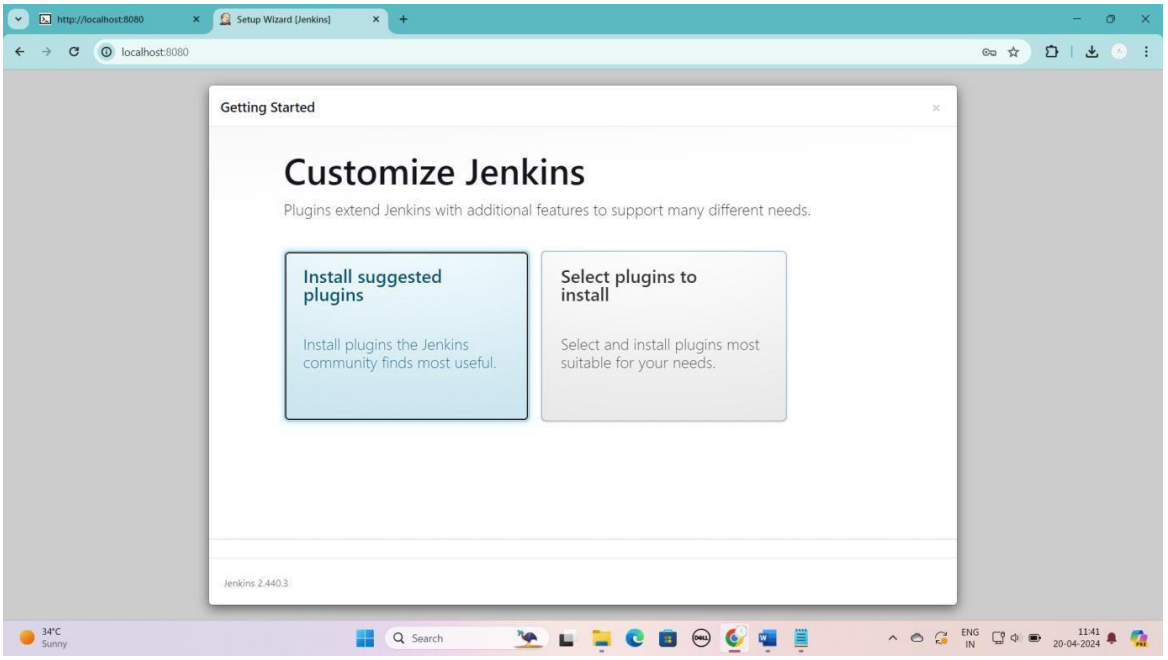
Step:14 Copy the password



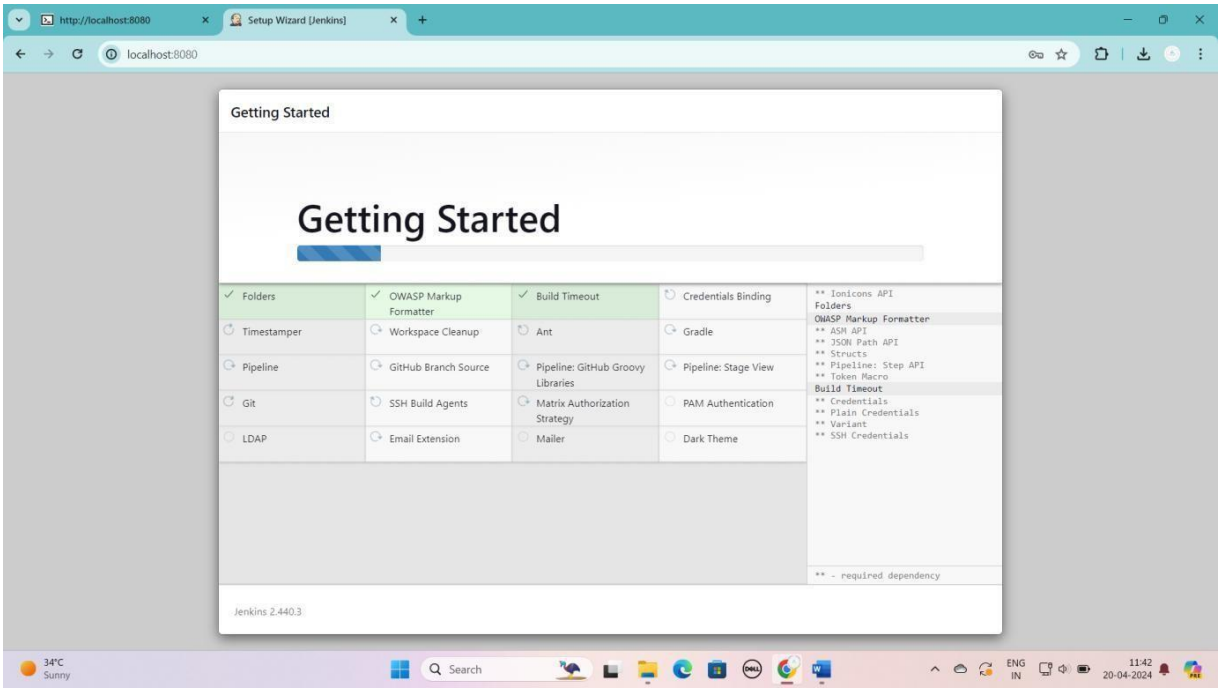
Step:15 To unlock jenkins paste the password and click continue



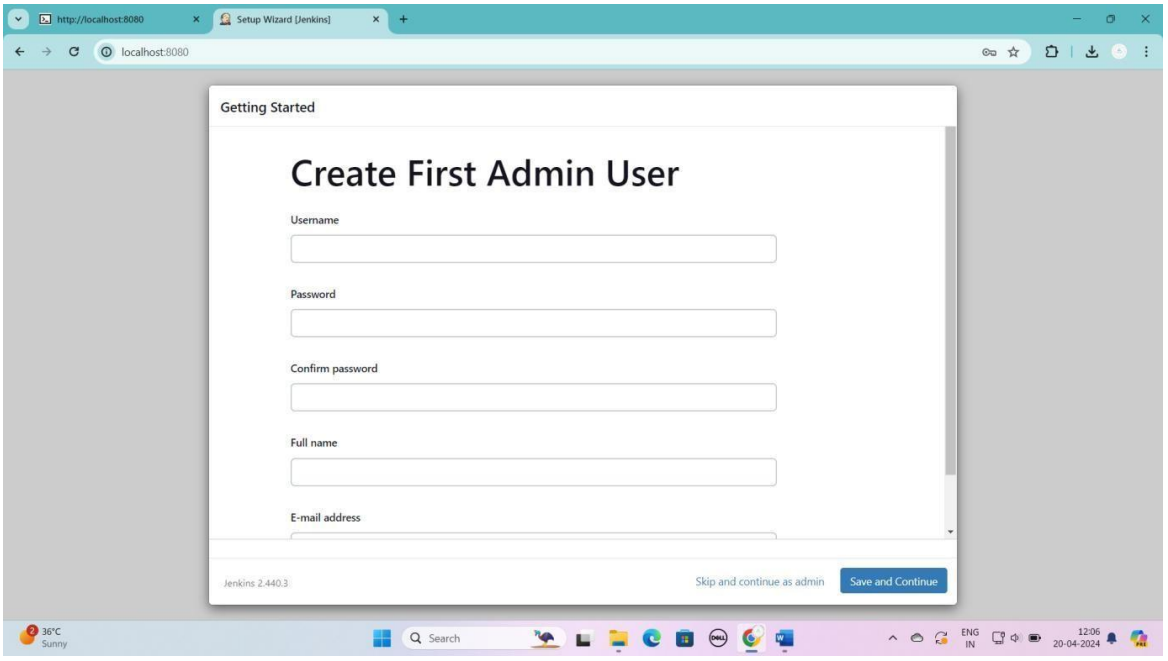
Step:16 Click install suggested plugins



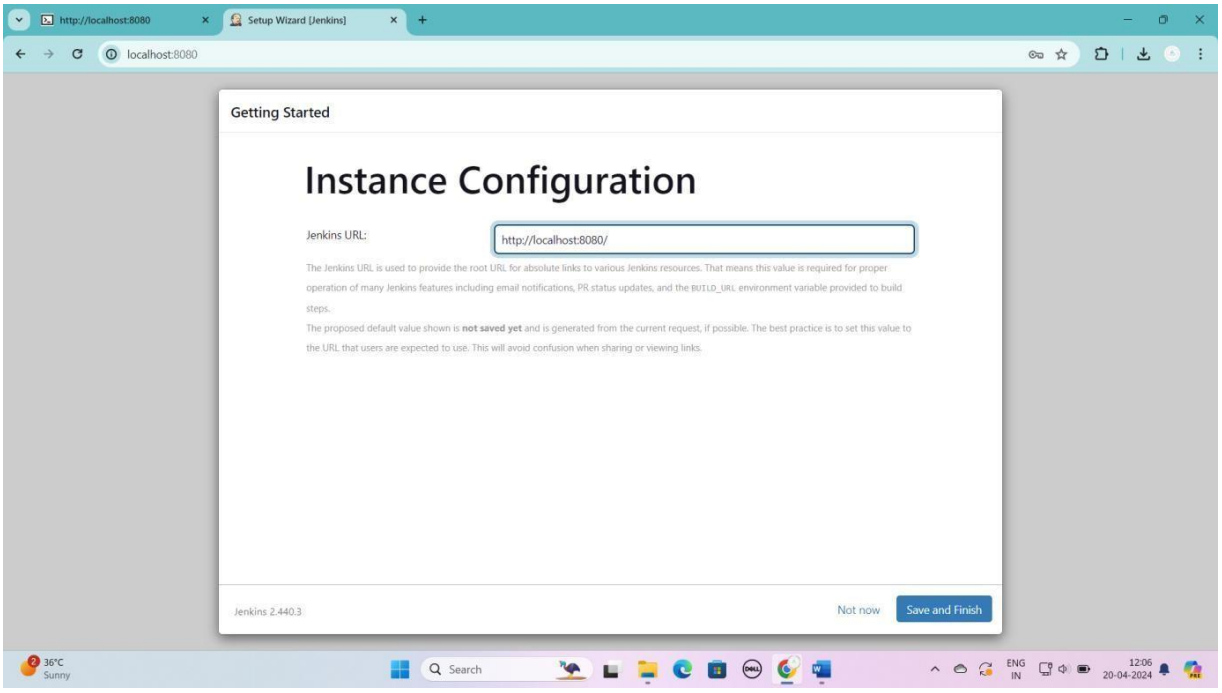
Step:17 plugins installation



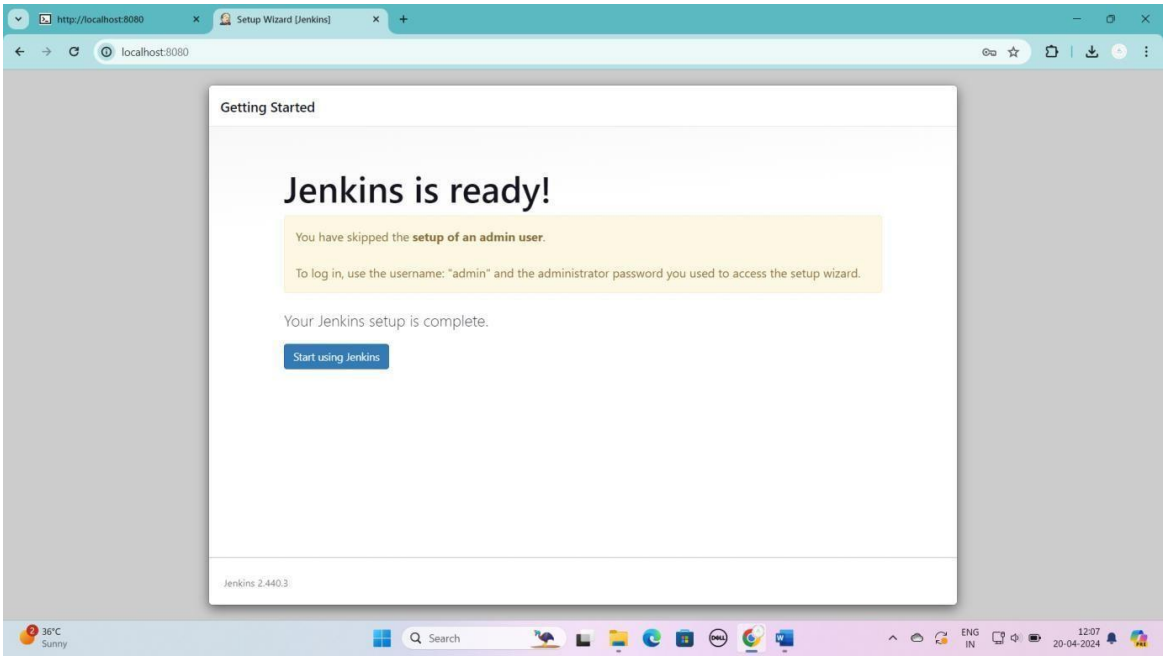
Step:18 Create Admin User & click save and continue



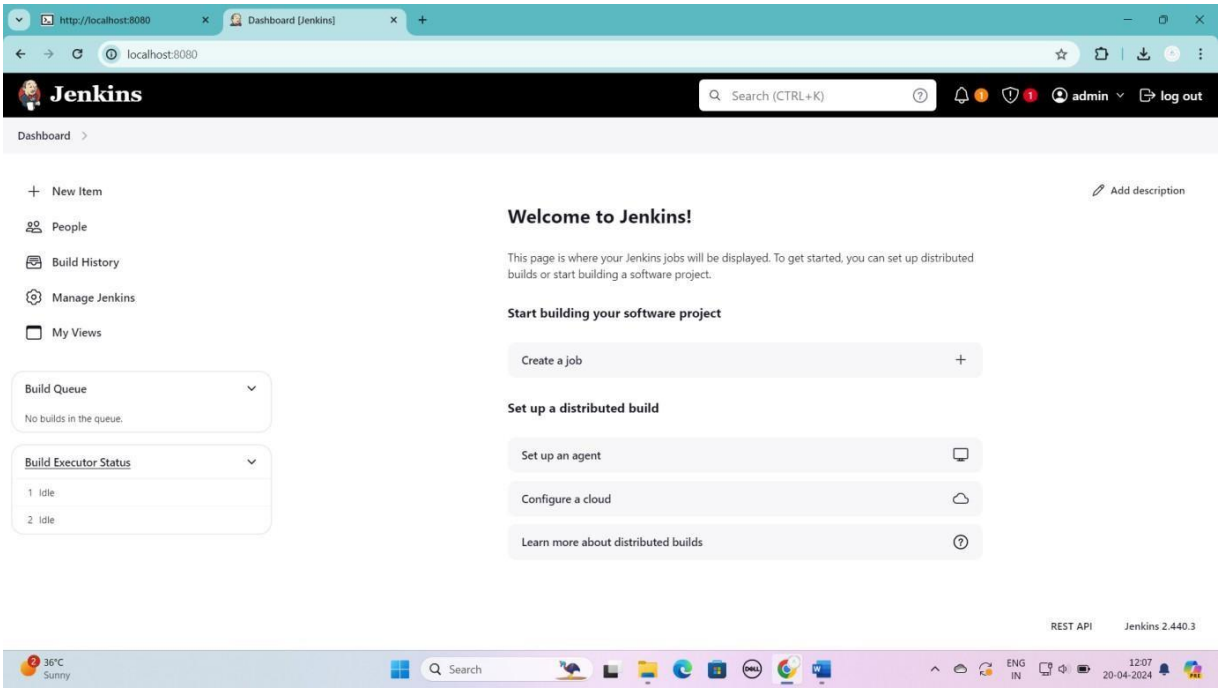
Step:19 Configure jenkins & click save and continue



Step:20 Jenkins setup is complete & click start using jenkins



Step:21 Welcome to jenkins page



Result:

Thus installing Jenkins in Cloud done successfully.

Ex.No:4 **Create CI pipeline using Jenkins**

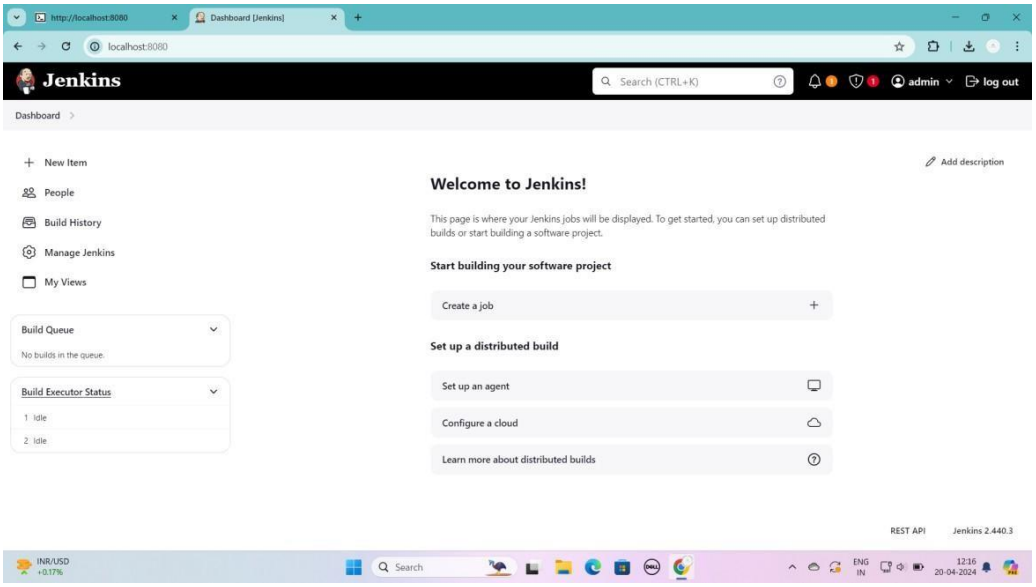
Date:

Aim:

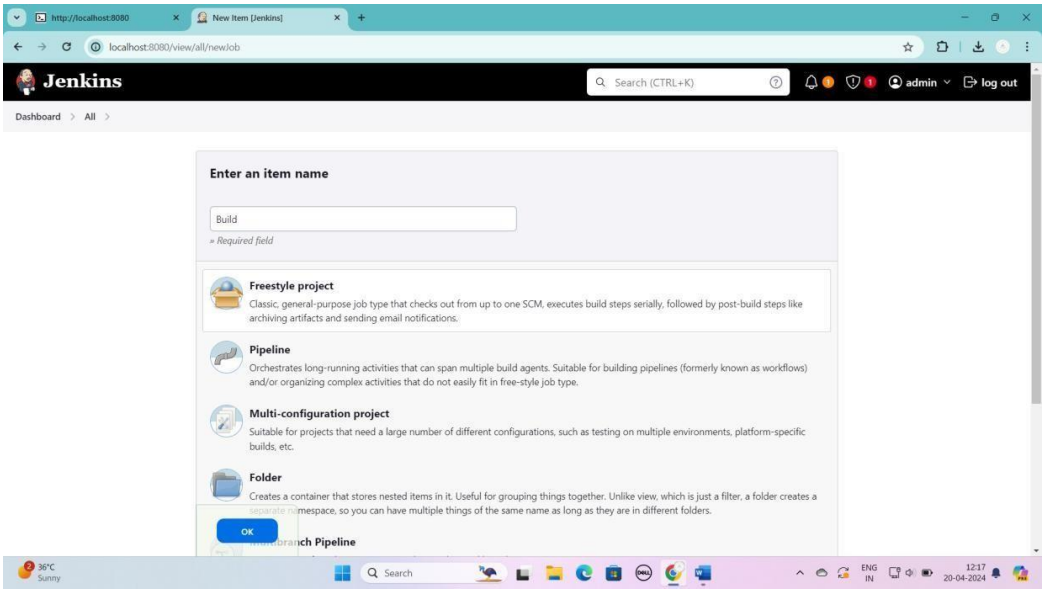
To create CI pipeline using Jenkins.

Procedure:

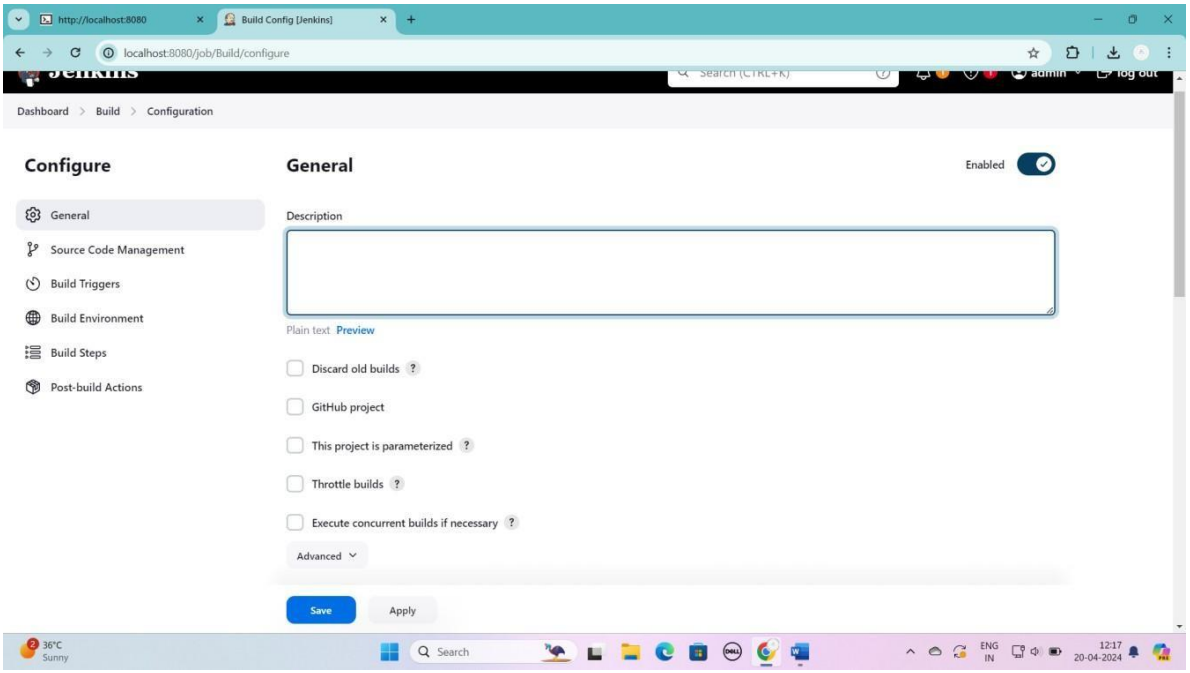
Step1:Go to jenkins page using localhost 8080



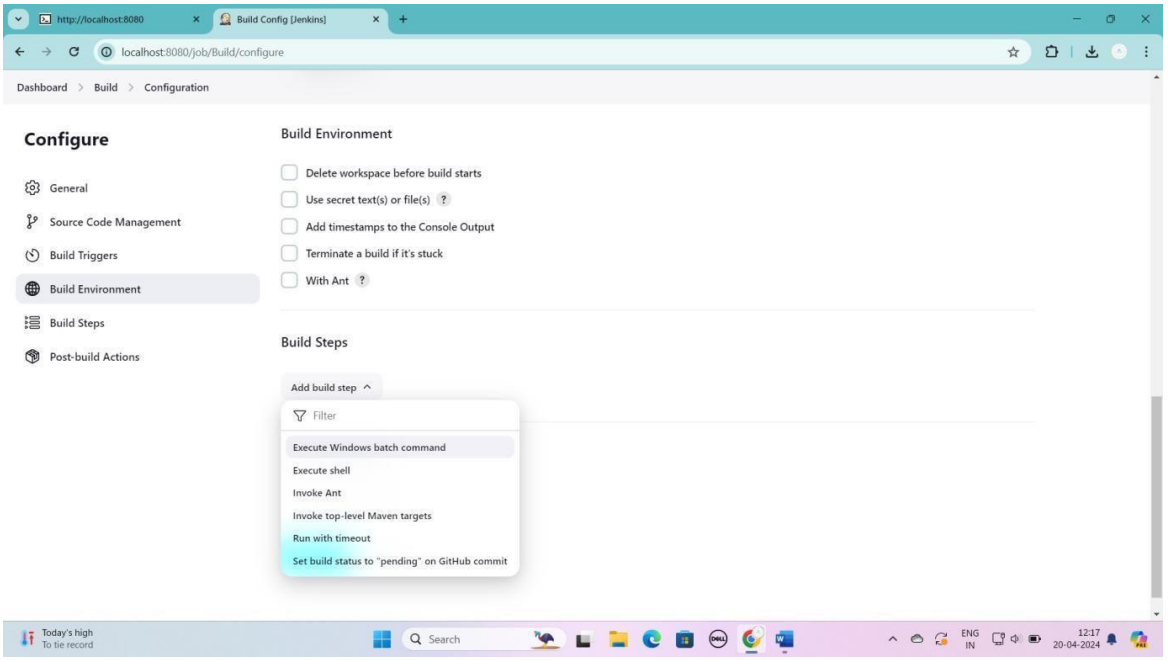
Step 2: Enter an item name as Build and Click Freestyle project& click ok.



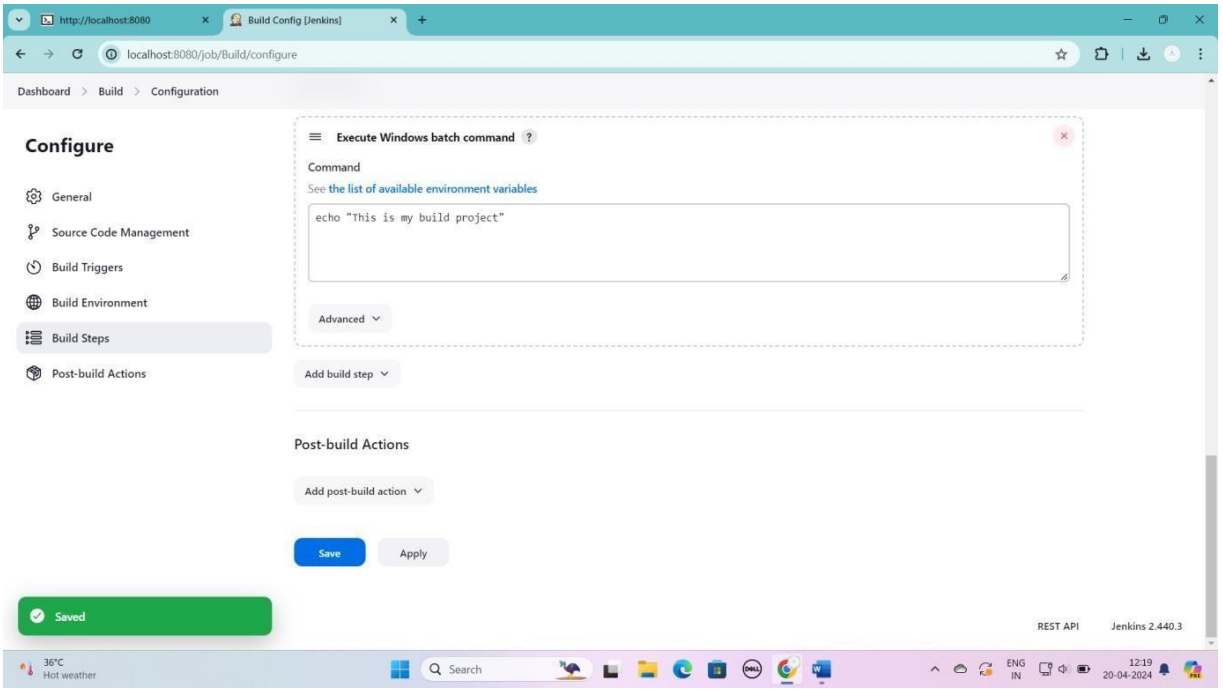
Step 3: From Configure click Build steps and scroll down the page.



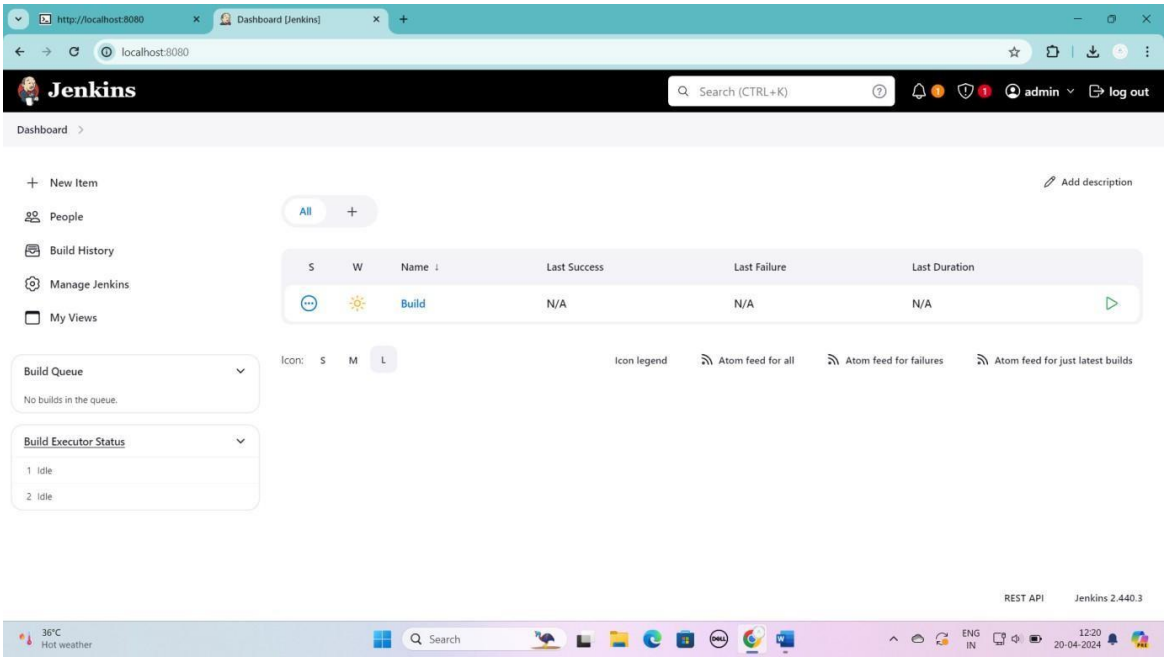
Step 4: Click Execute windows batch command as shown below



Step 5: In command window give a message using echo and click apply and save.

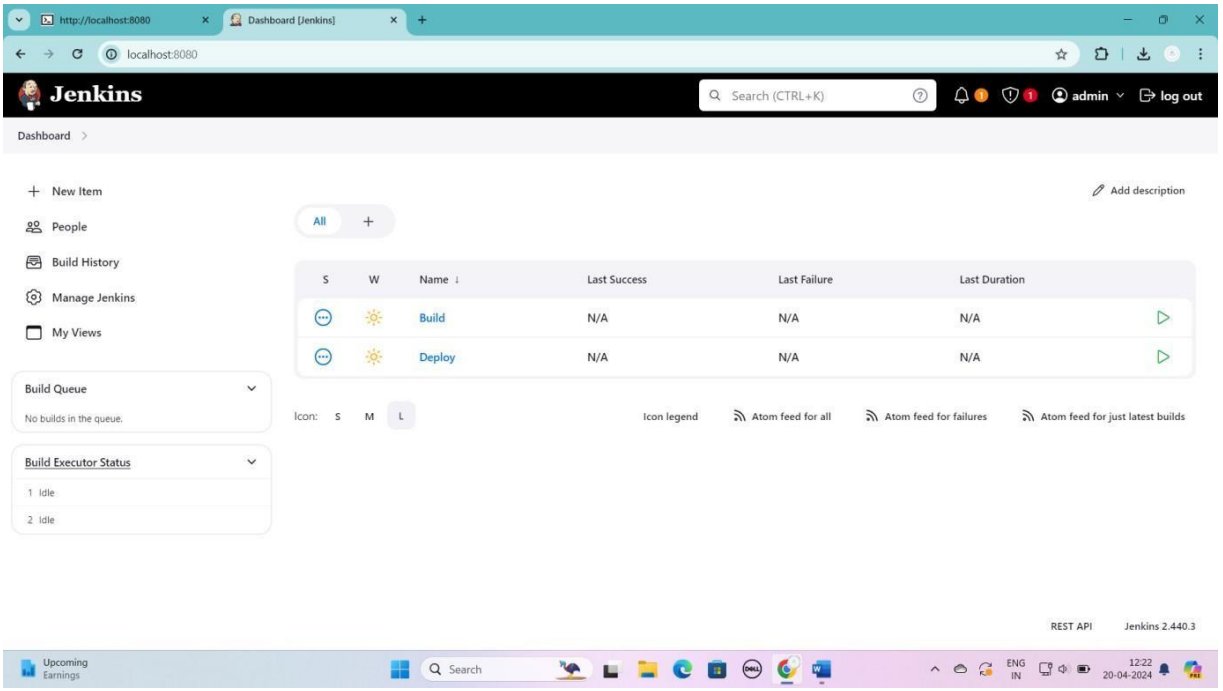


Step 6: The below page will be generated after apply and save.

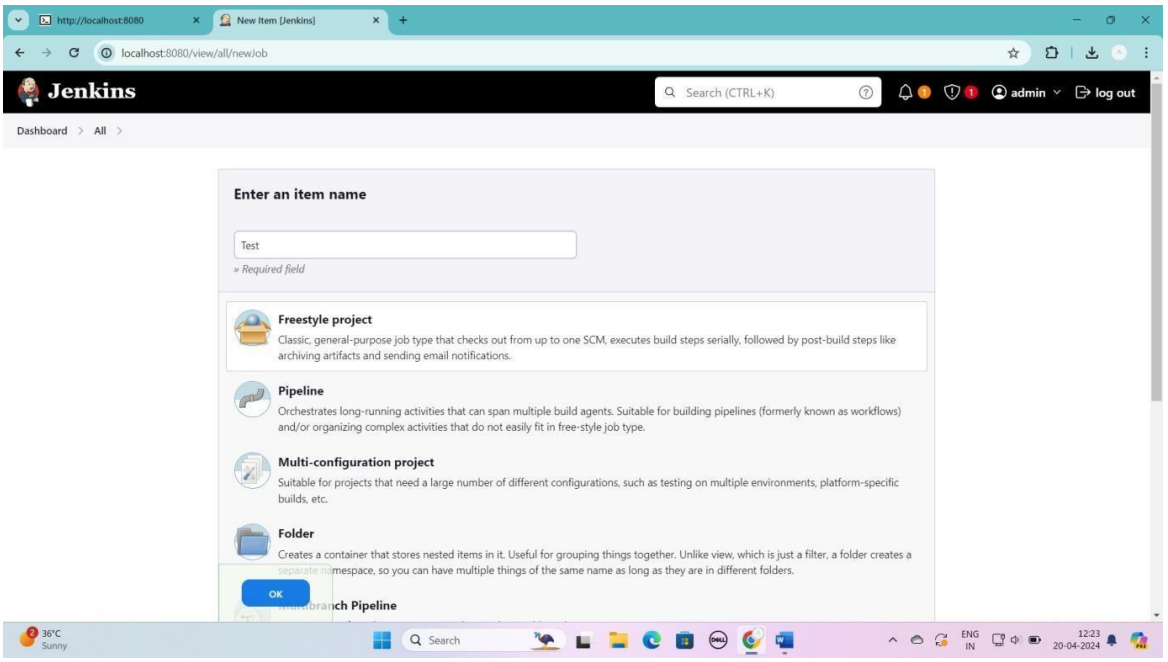


Step 7: The same procedure will be followed for creating Deploy [refer step 2 to step 6].

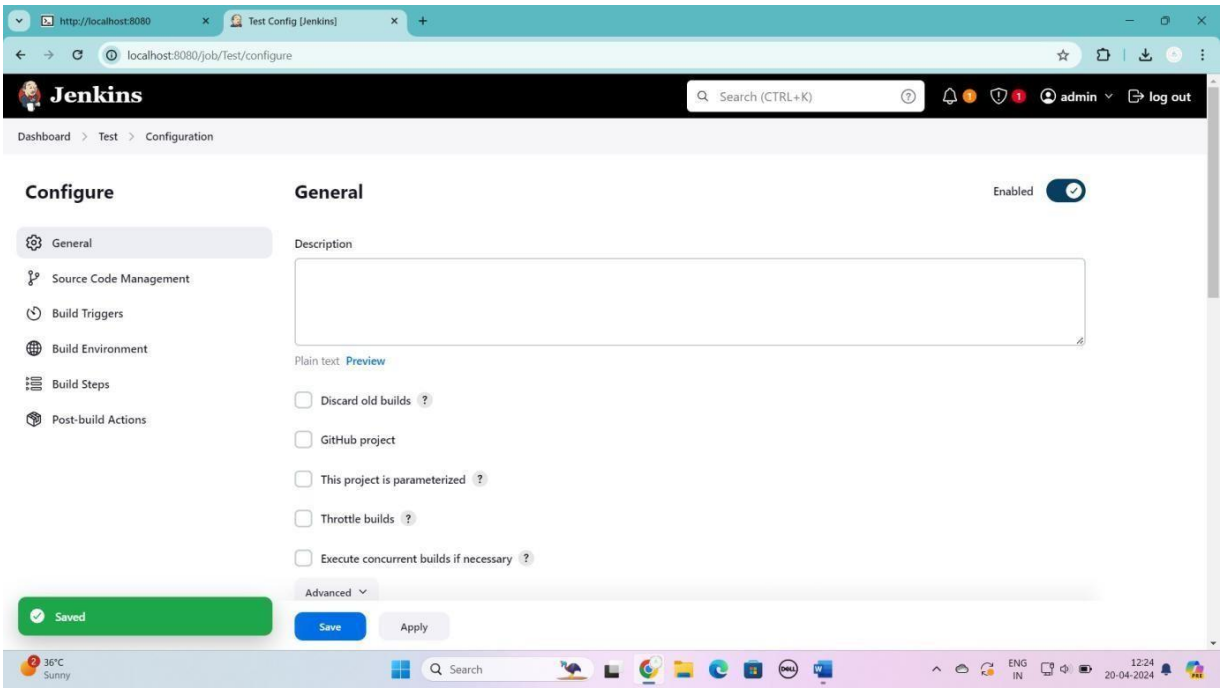
After creating Deploy the below page will be generated and click on the new item.



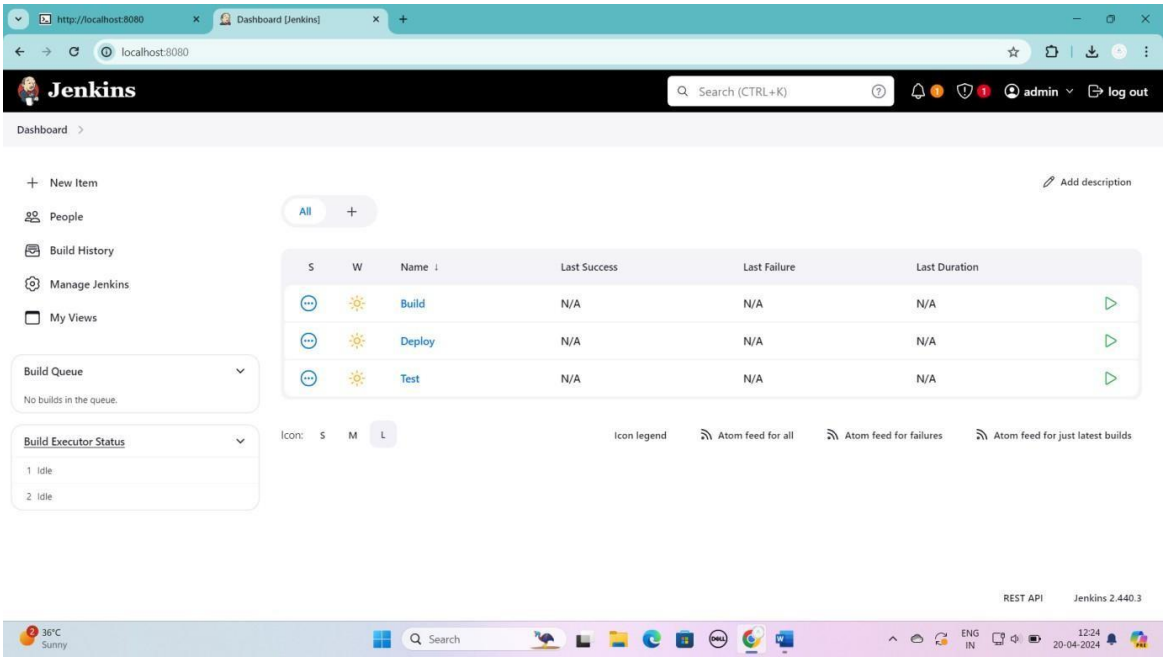
Step 8: Enter an item name as Test &click on freestyle project and click ok.



Step 9: Click apply and save.

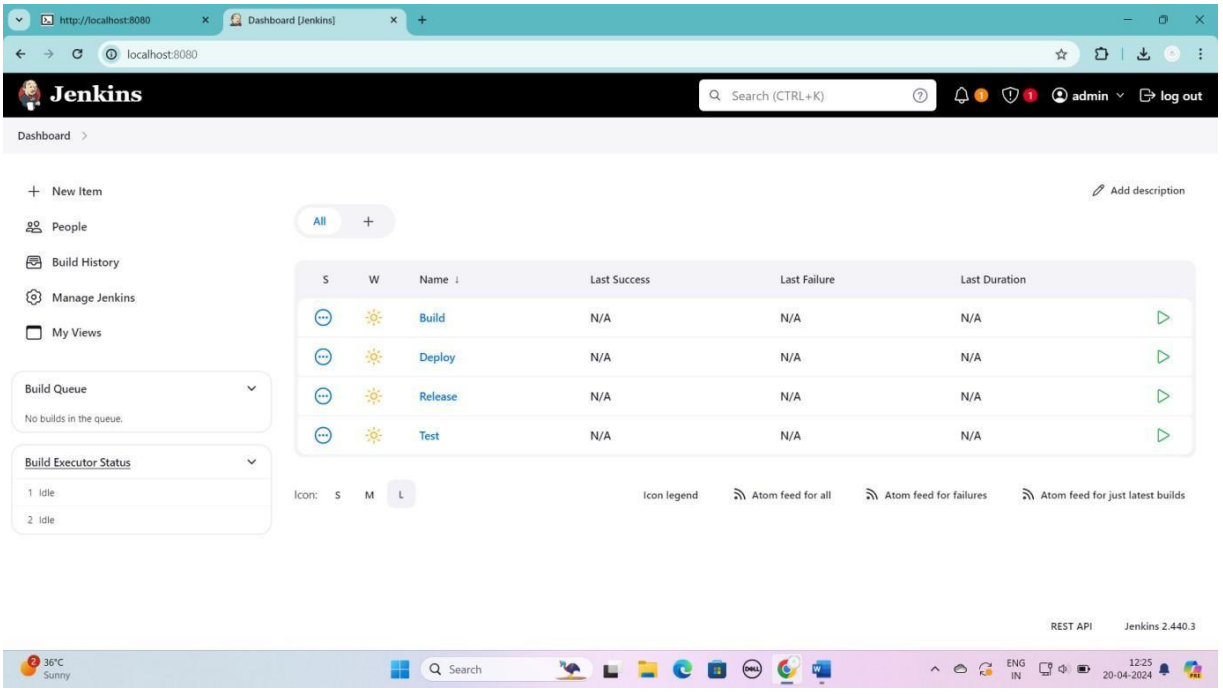


Step 10: The below page will be generated after apply and save

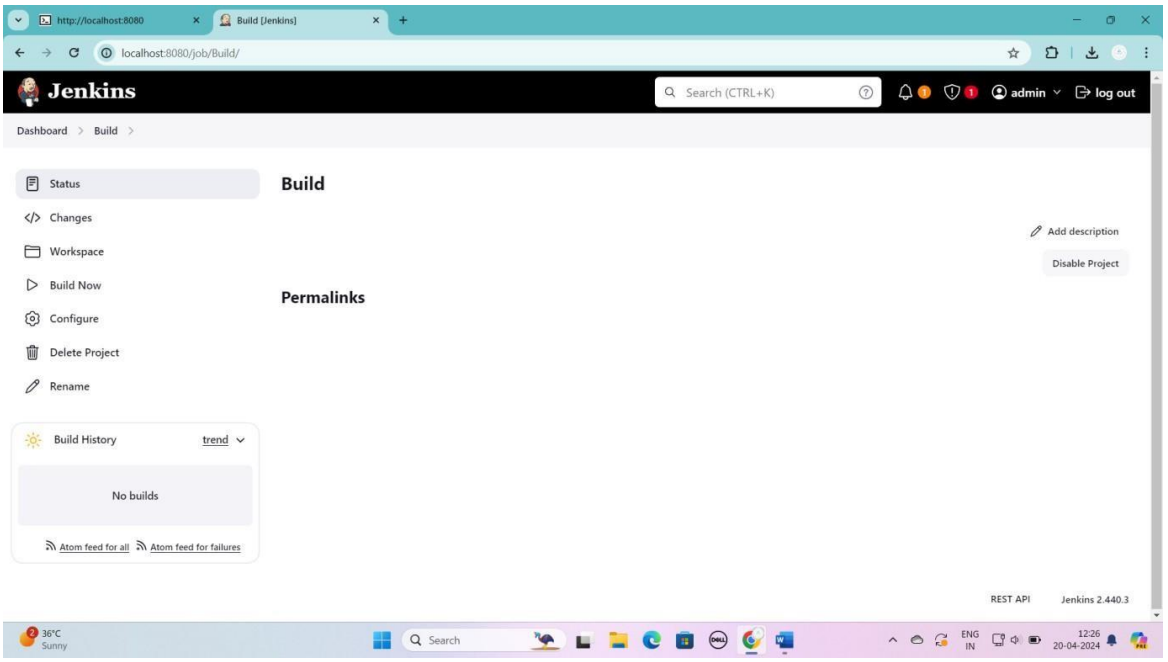


Step 11: The same procedure will be followed for creating Release [refer test step 9, 10].

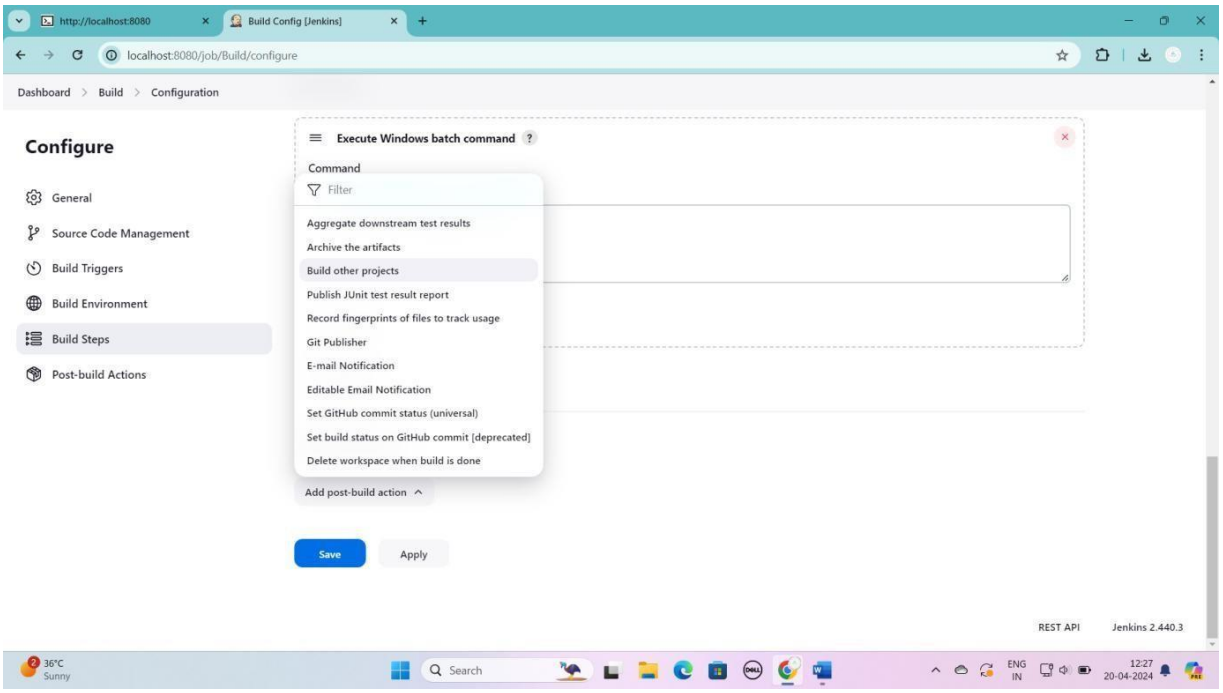
After creating Deploy the below page will be generated and click on the new item



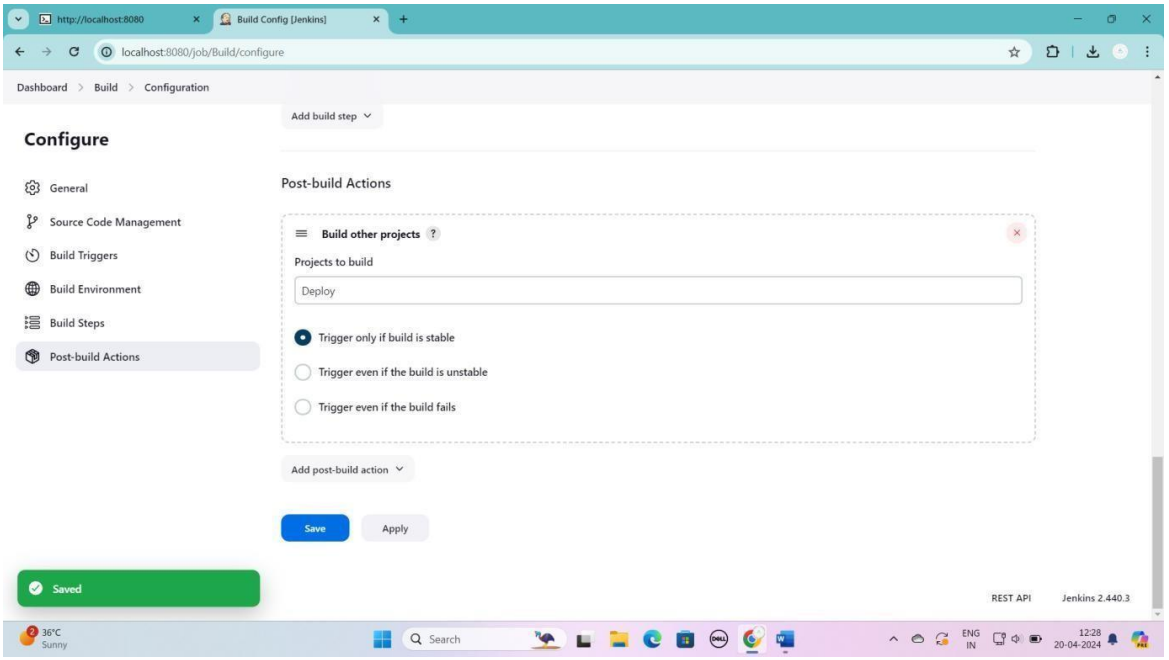
Step 12: Click the Build item and Click on the Configure.



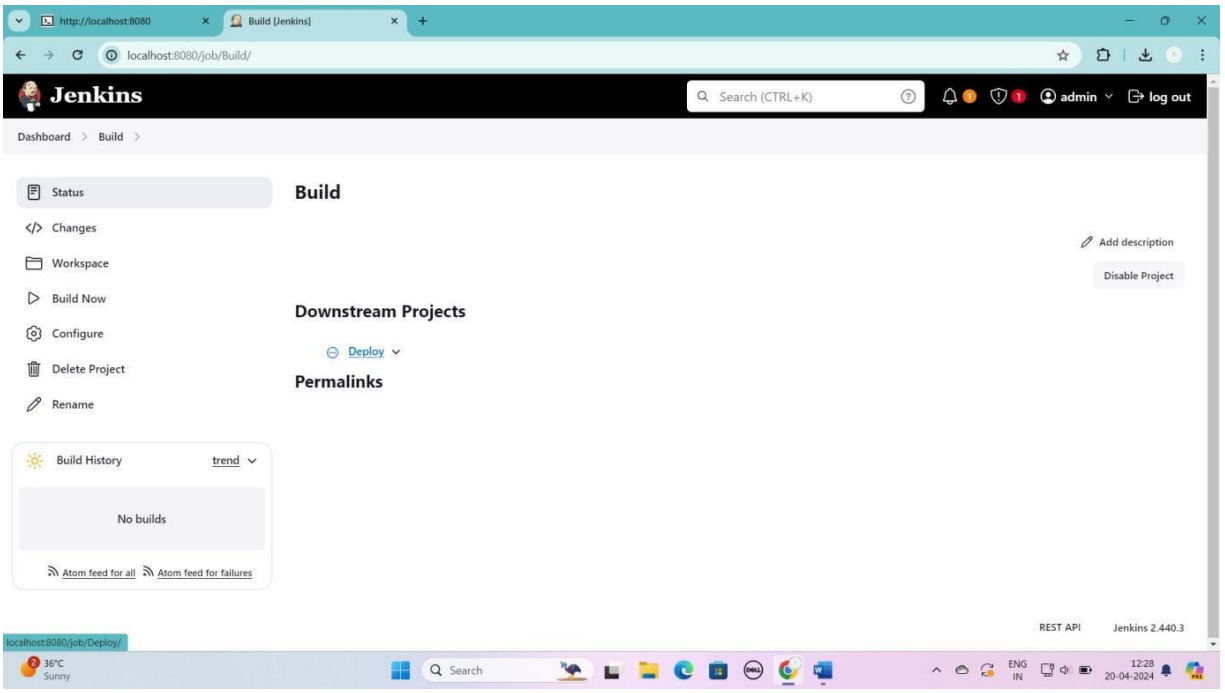
Step 13: Click the Build Steps in Execute Window batch command and click the Build other Projects on drop down list.



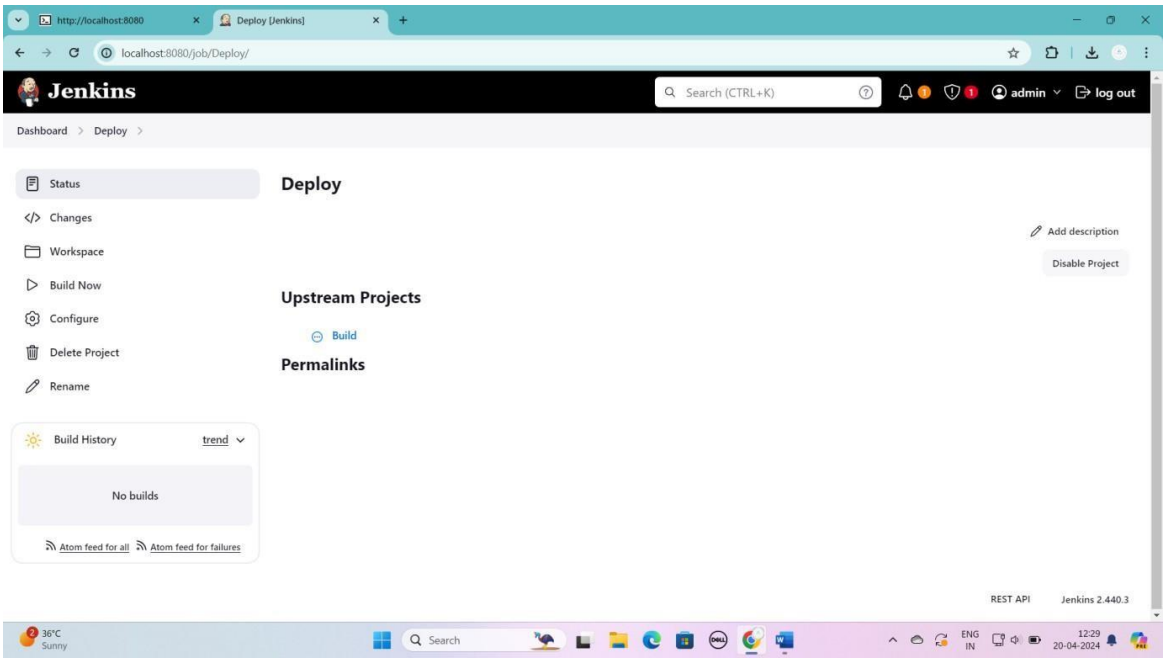
Step 14: Click on the Post build actions in Build other project enter the Deploy as in next item name to connect here and click the Trigger only if build is stable, Click apply and save.



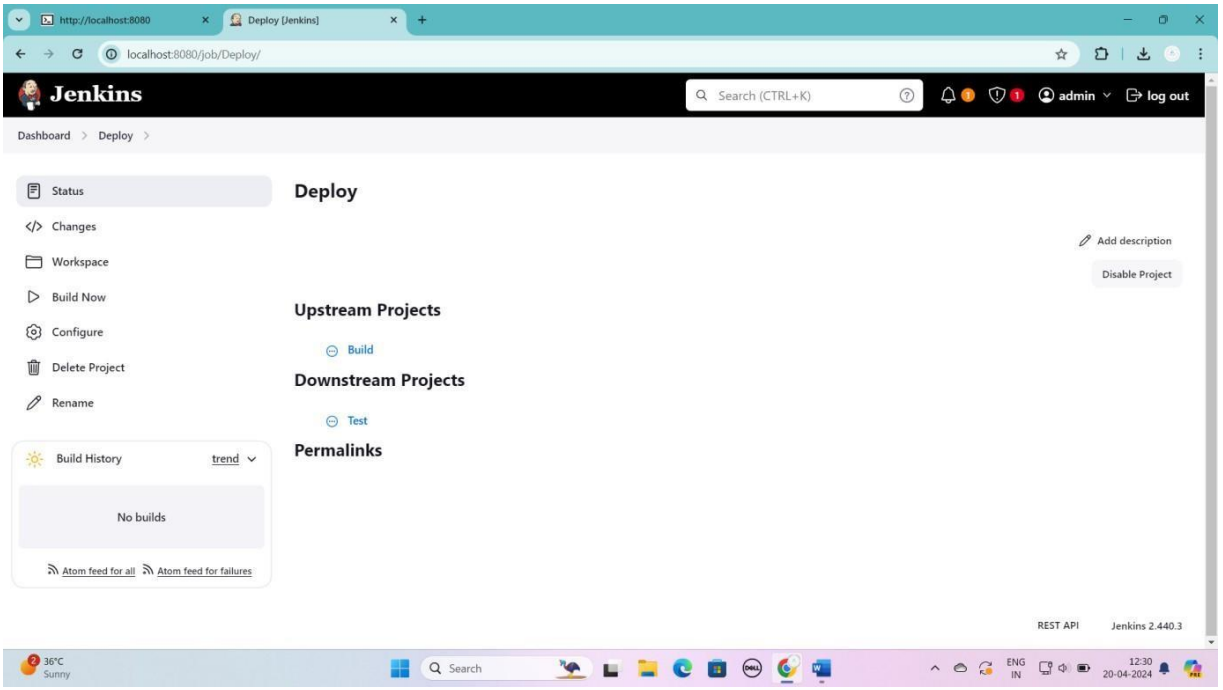
Step 15: In Build item –>Downstream project ->Click Deploy.



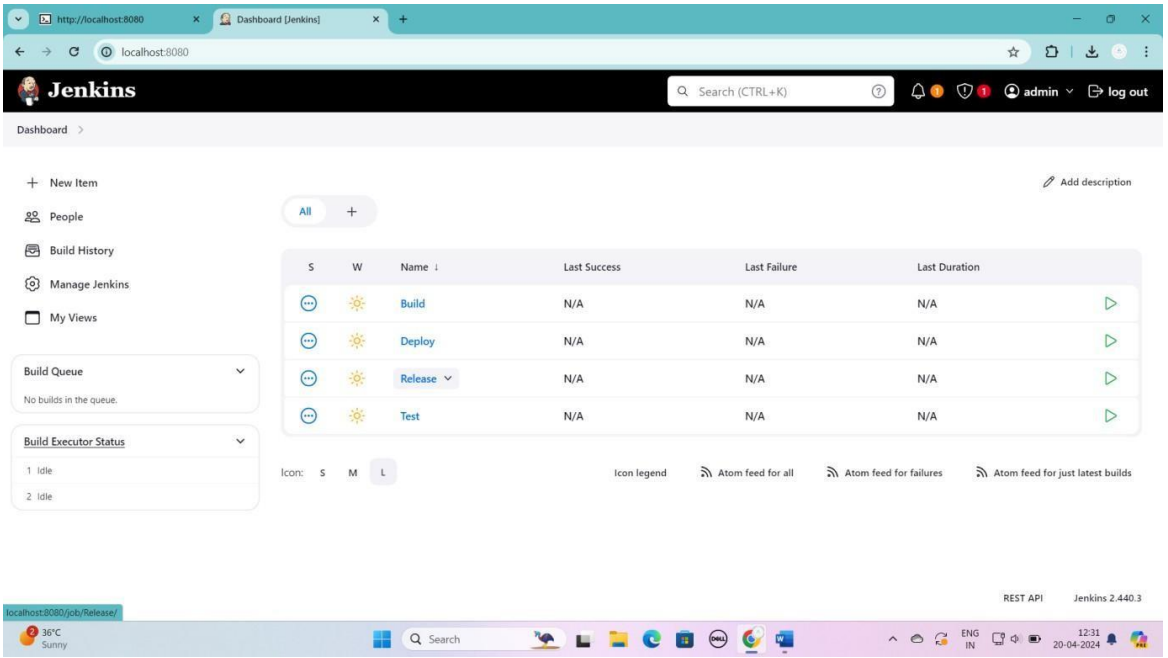
Step 16: After clicking on the Deploy the below page will be displayed Upstream projects as Build.



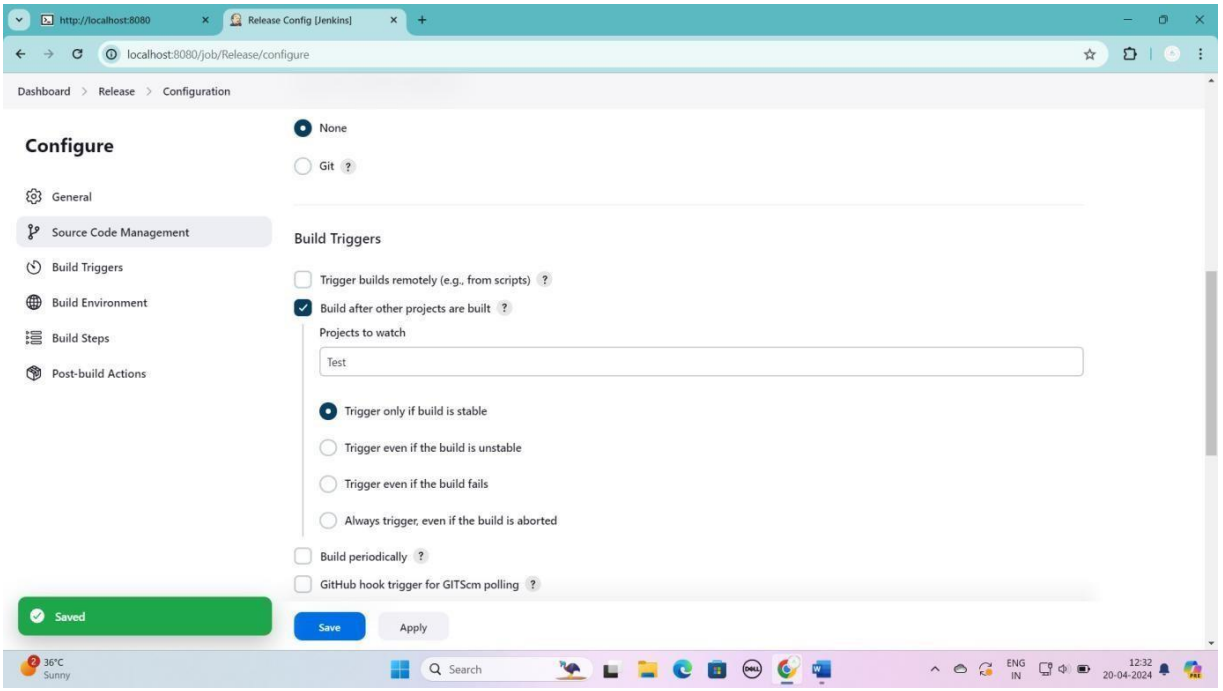
Step 17: The same procedure will be followed for connecting the Test [refer step 12 to 16]. After creating Deploy the below page will be generated and click on the new item, connecting the below page is displayed. It shows Upstream project as Build and Downstream project as Test.



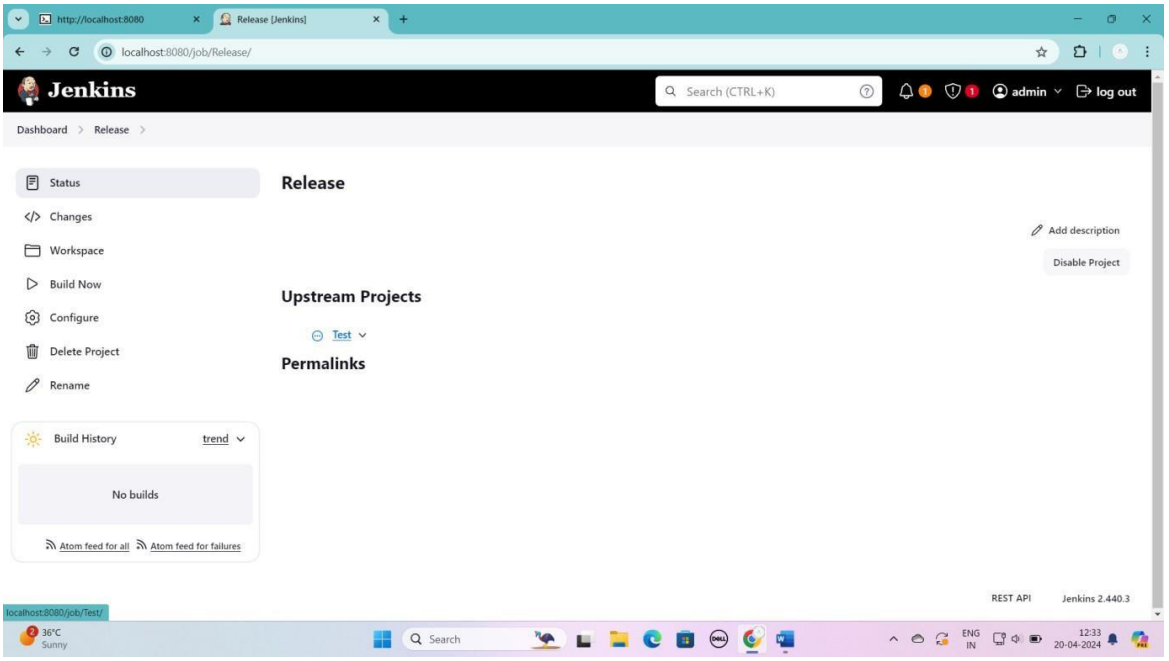
Step 18: After connecting all these items with each other the below page is display. Click the Release item.



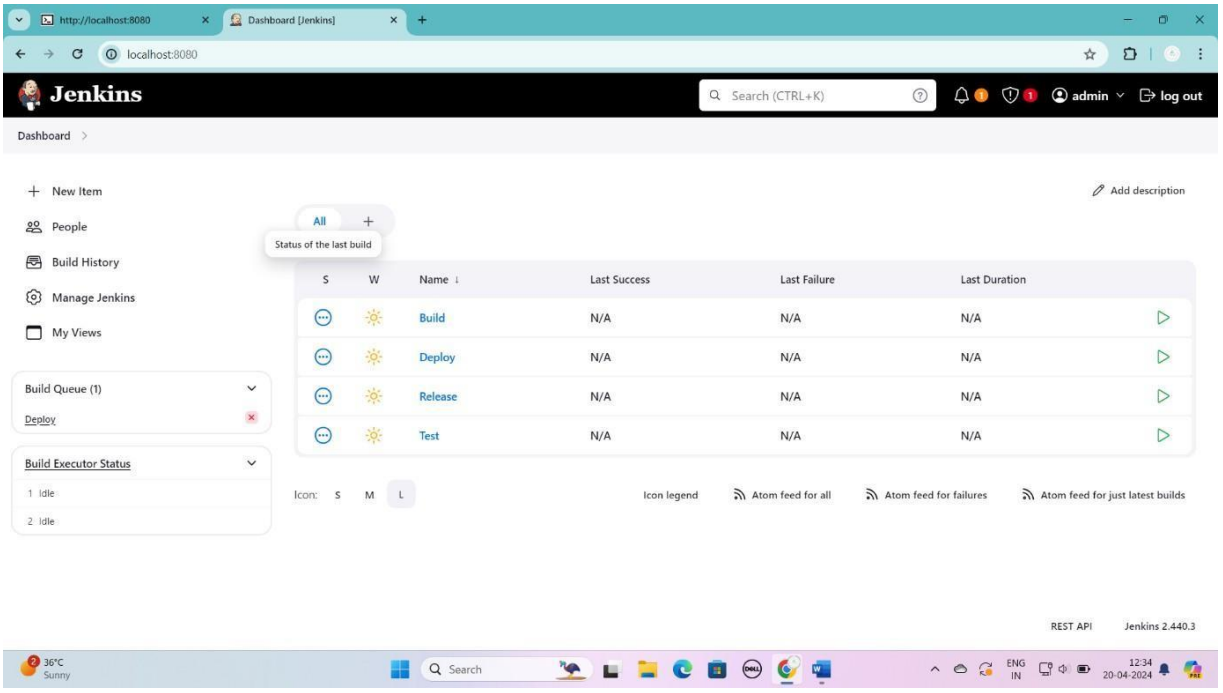
Step 19: Go to configure & click the Build Triggers ->Build after projects are built, Now in "project to match" give the Test and click on the "Trigger only if build is stable", Click apply and save.

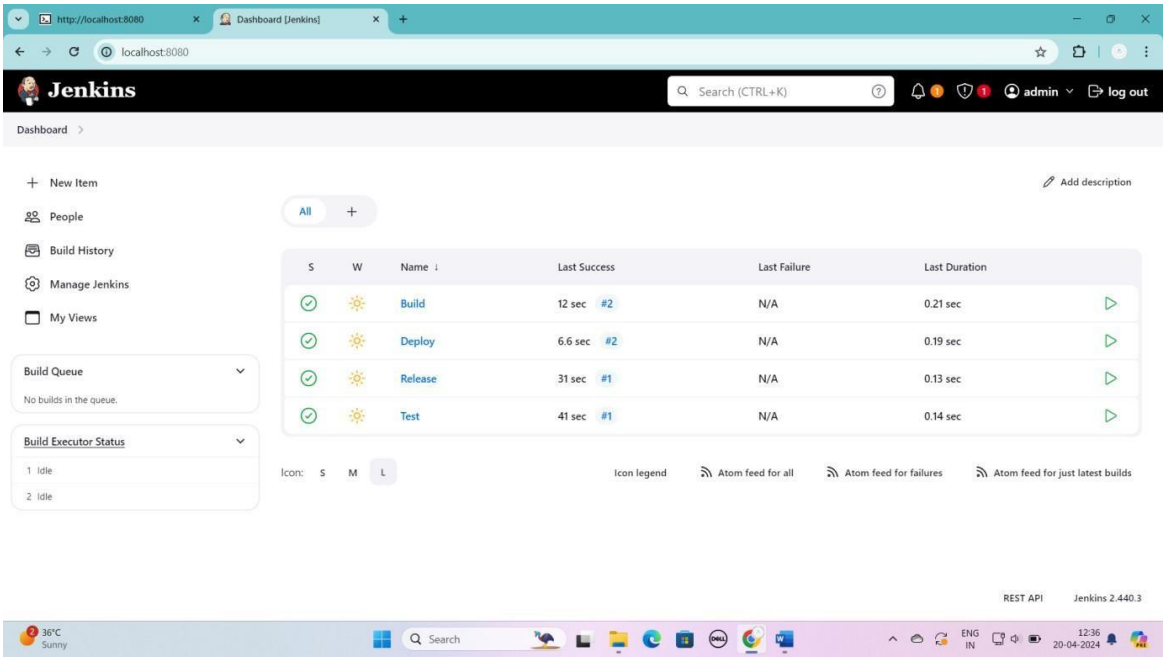


Step 20: After clicking the below page will be displayed the Test as upstream project. Click on the Test.

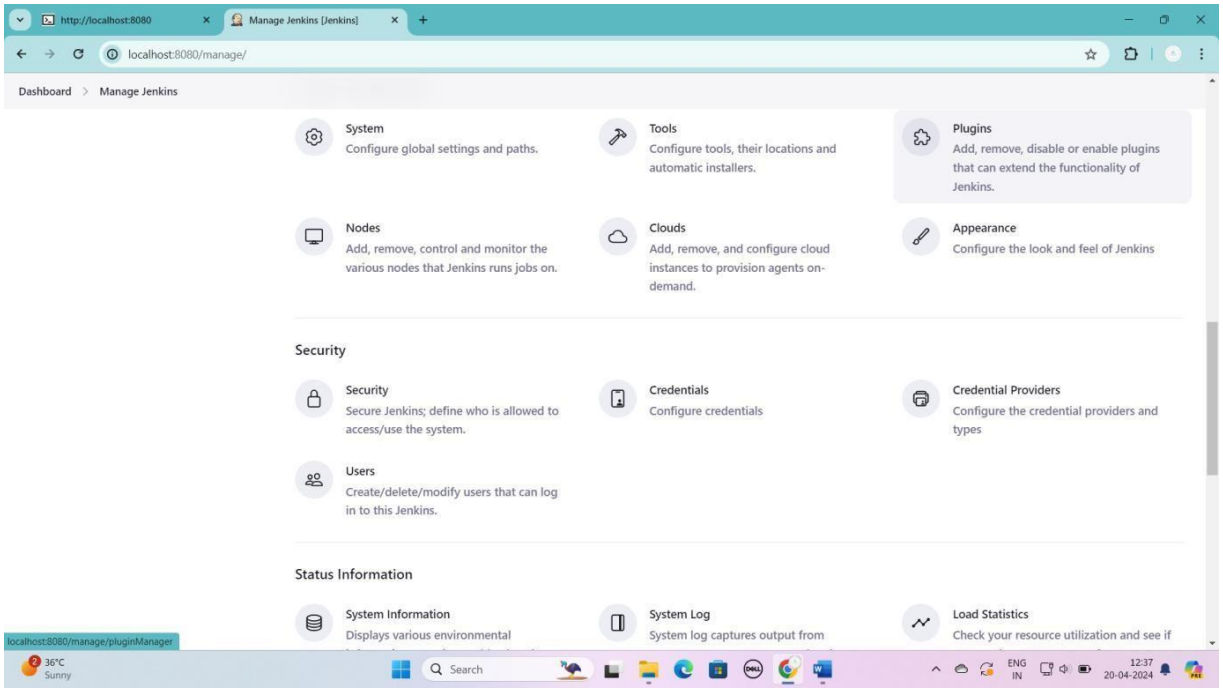


Step 21: After connecting all the items together below image will display. If we run the Build item the connecting Deploy item will display on the Build Queue.

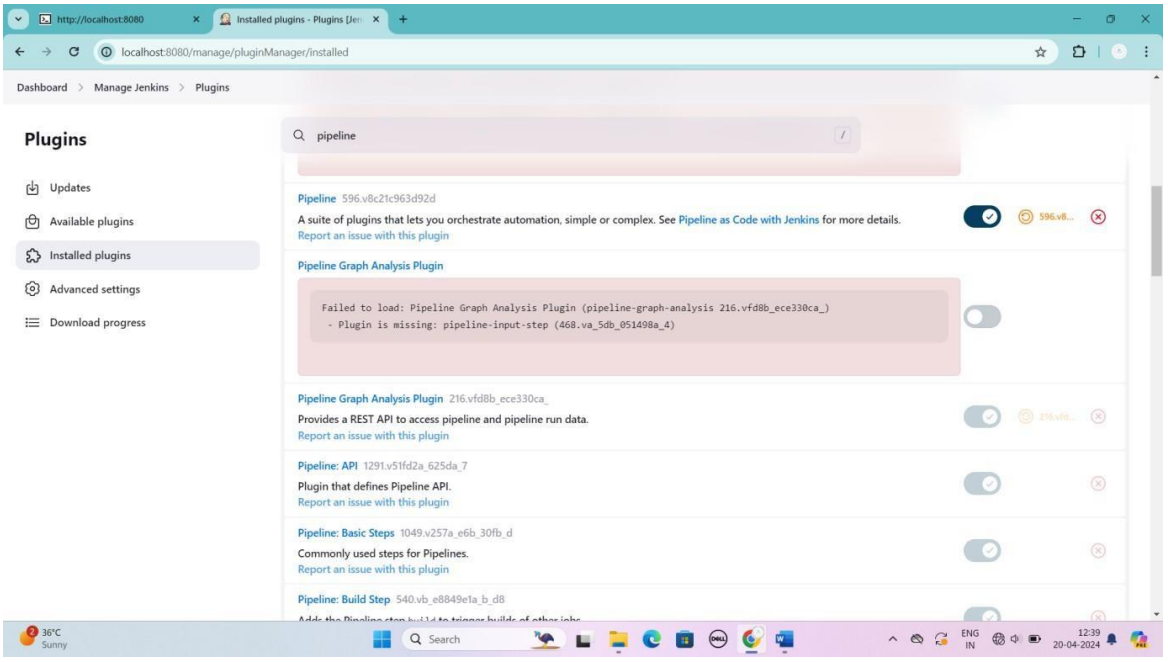




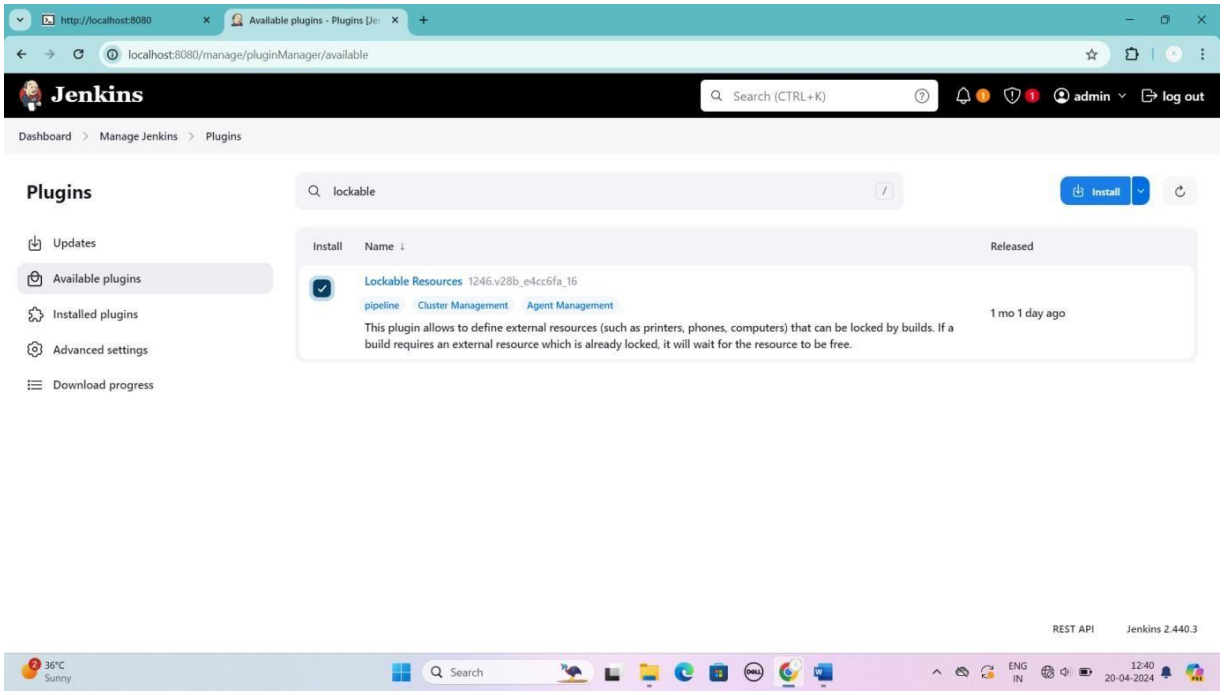
Step 22: Go to Mange Jenkins scroll down the page and click on plugins.



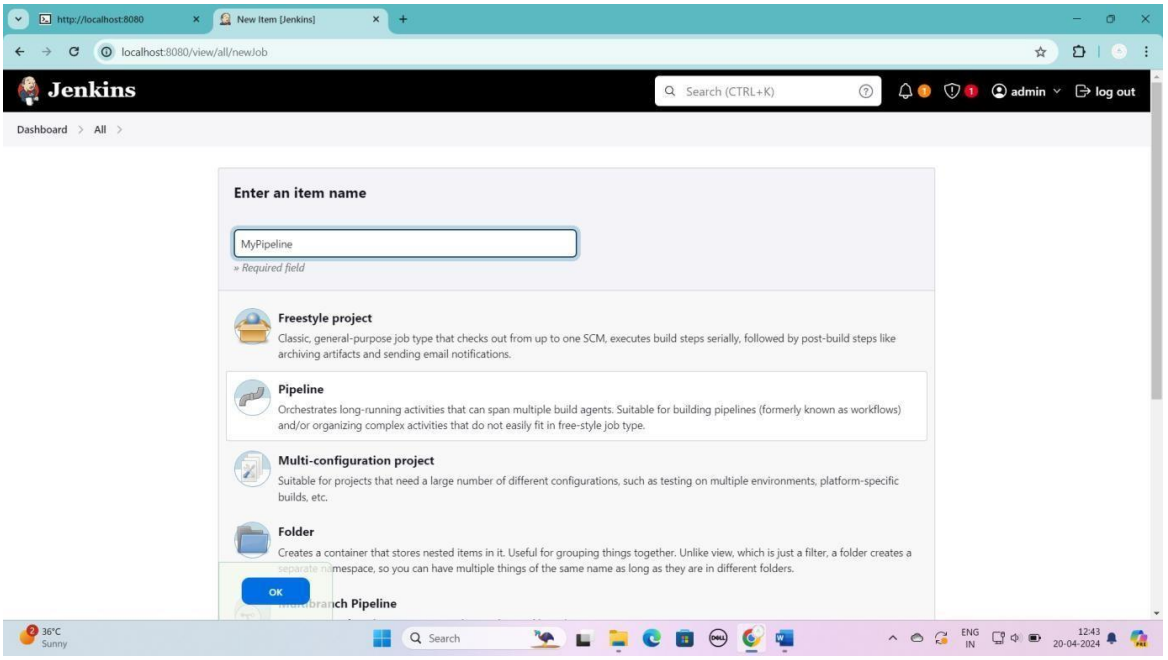
Step 23: Click on the Installed plugins and check the Pipeline and enable it.



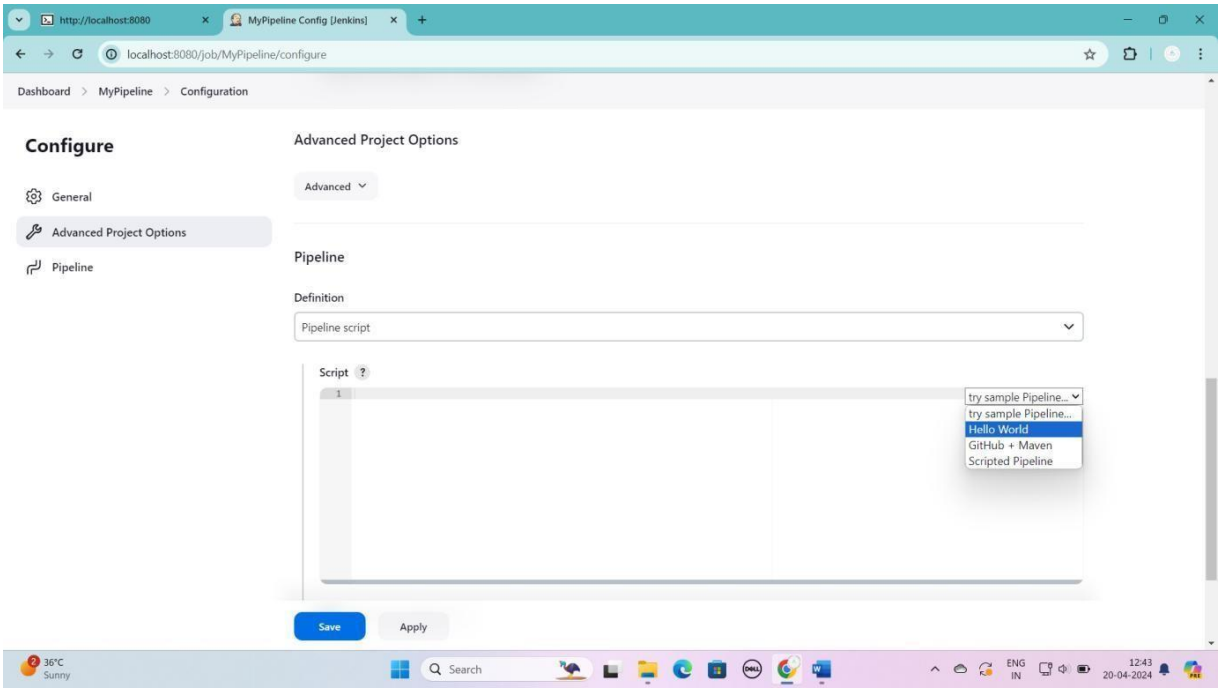
Step 24: Go to Available plugins and install the Lockable Resources.



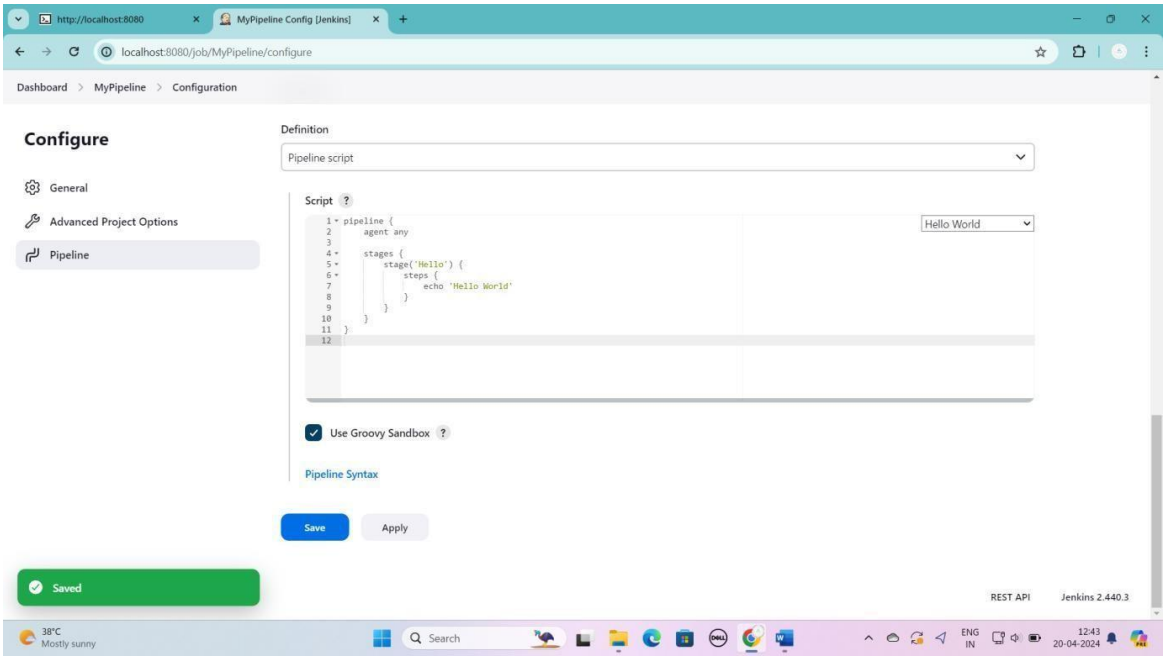
Step 25: After installing, the Pipeline Click on New item and Enter an item name as MyPipeline & click on the Pipeline and click ok to finish.



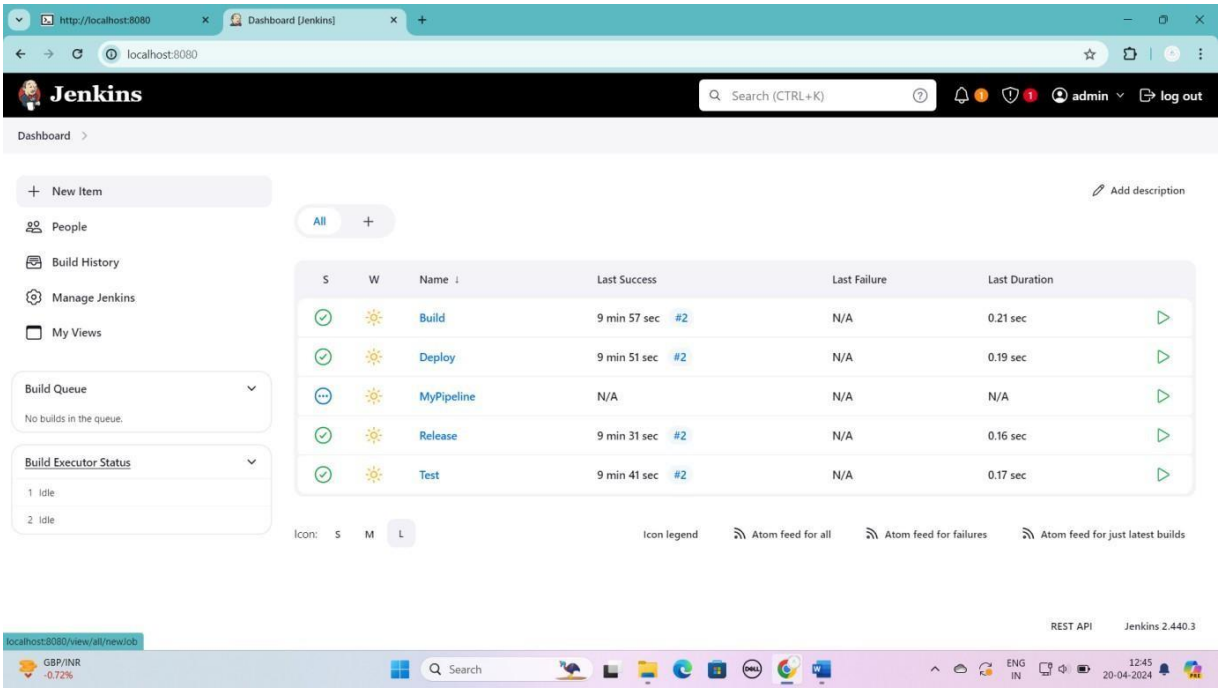
Step 26 : After Creating the below page will displayed and click on the Pipeline in side bar and Choose the Pipeline Script as Definition and Choose the Script default select the Hello world Program in the drop down menu.



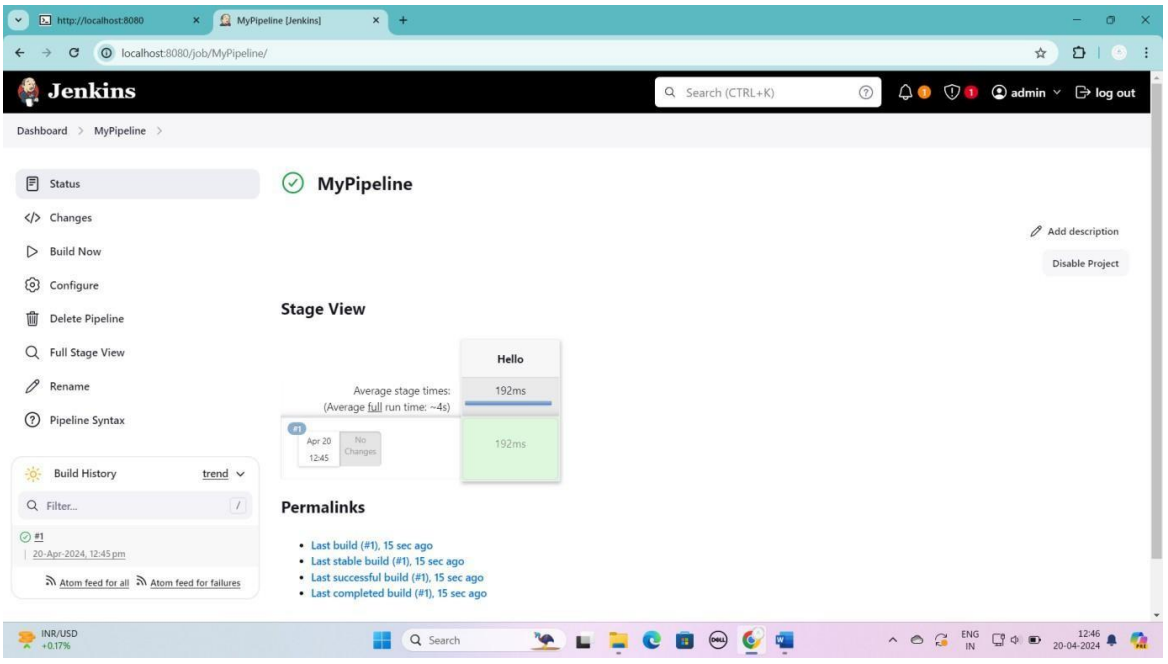
Step 27: After Clicking it the below page is display, Click on the Use Groovy Sandbox and Click Apply and Save.



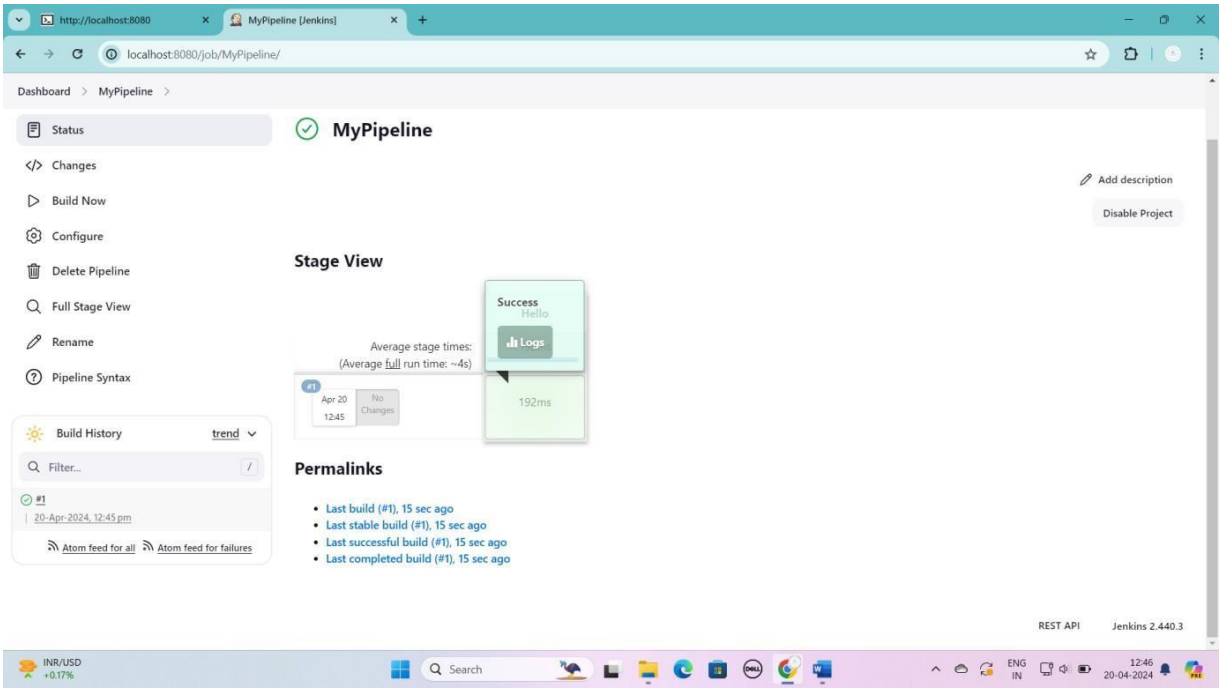
Step 28: After creating the Pipeline the below page is display and click on the run button to run the program and open it.



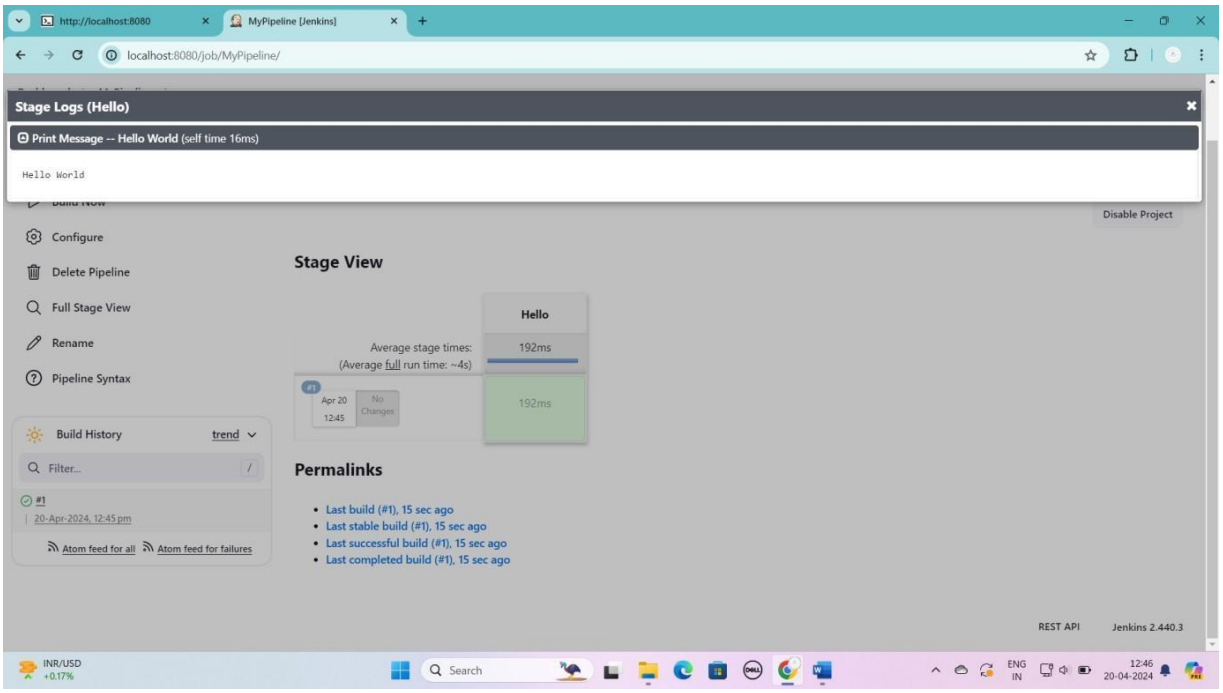
Step 29: After opening the page the below page will display and Click on the Hello.



Step 30: After Clicking Hello the below page will displayed and click on the Logs.



Step 31: By clicking on the Logs the output will display on the top of the screen like below image.



Result:

Thus, Creating CI pipelines using Jenkins done successfully.

Ex.No:4 **Build a simple application using Gradle**

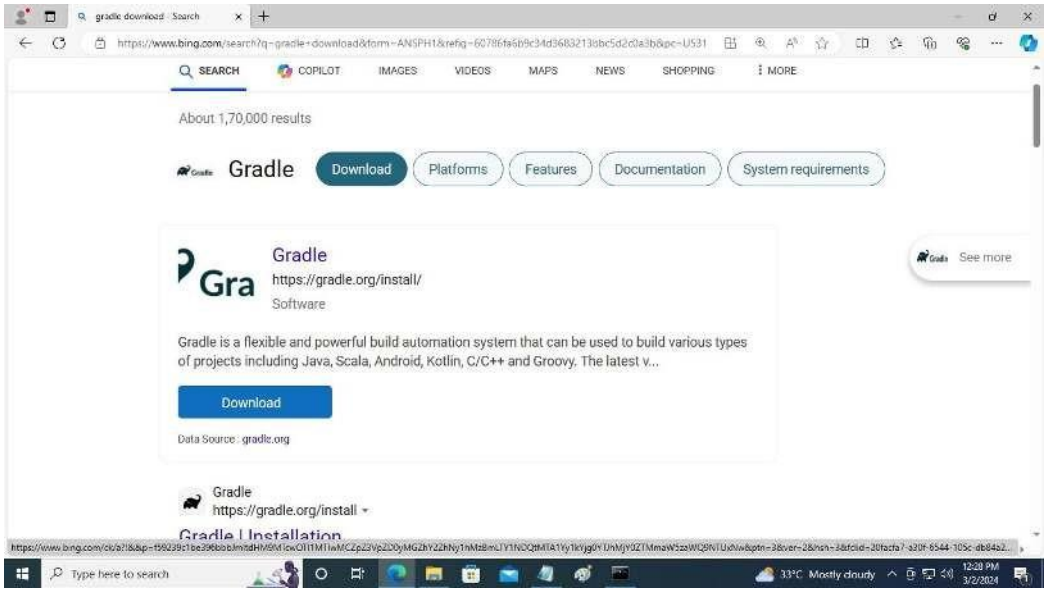
Date:

Aim:

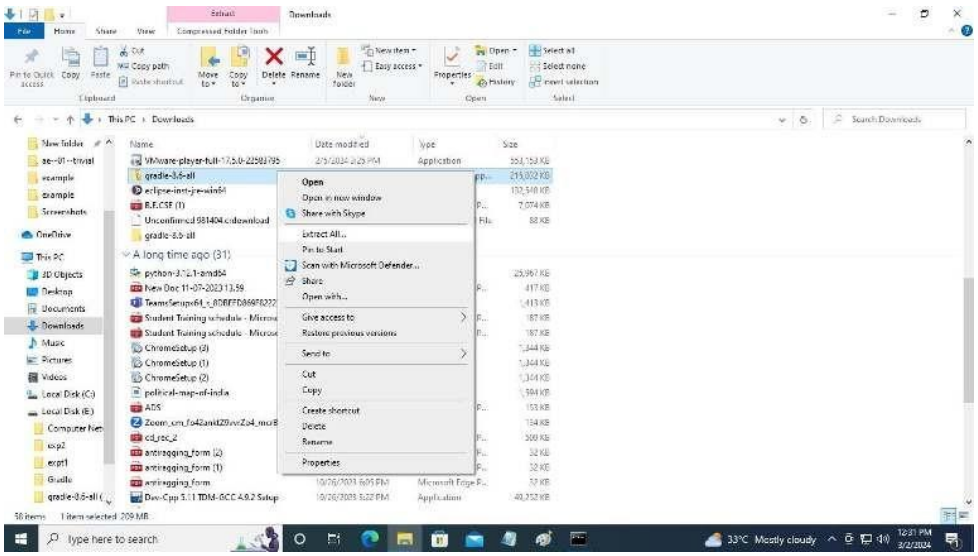
Build a simple application using Gradle

Procedure:

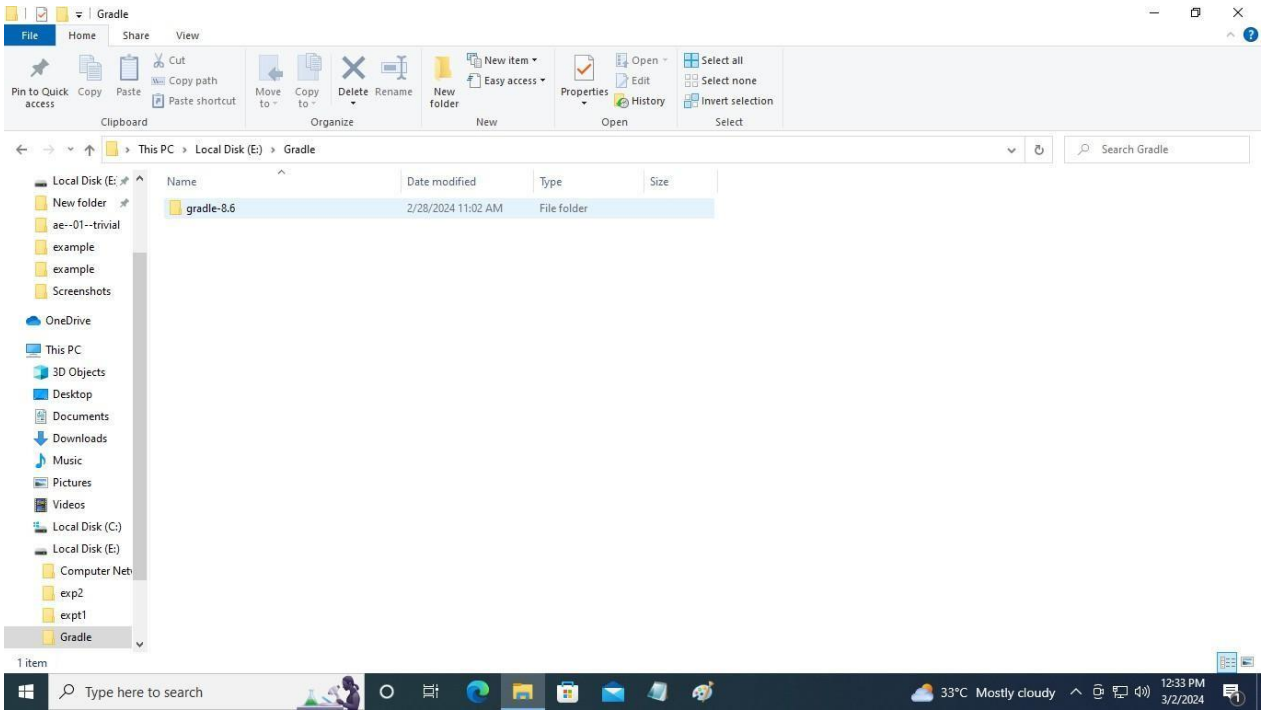
Step 1: Download gradle from official website



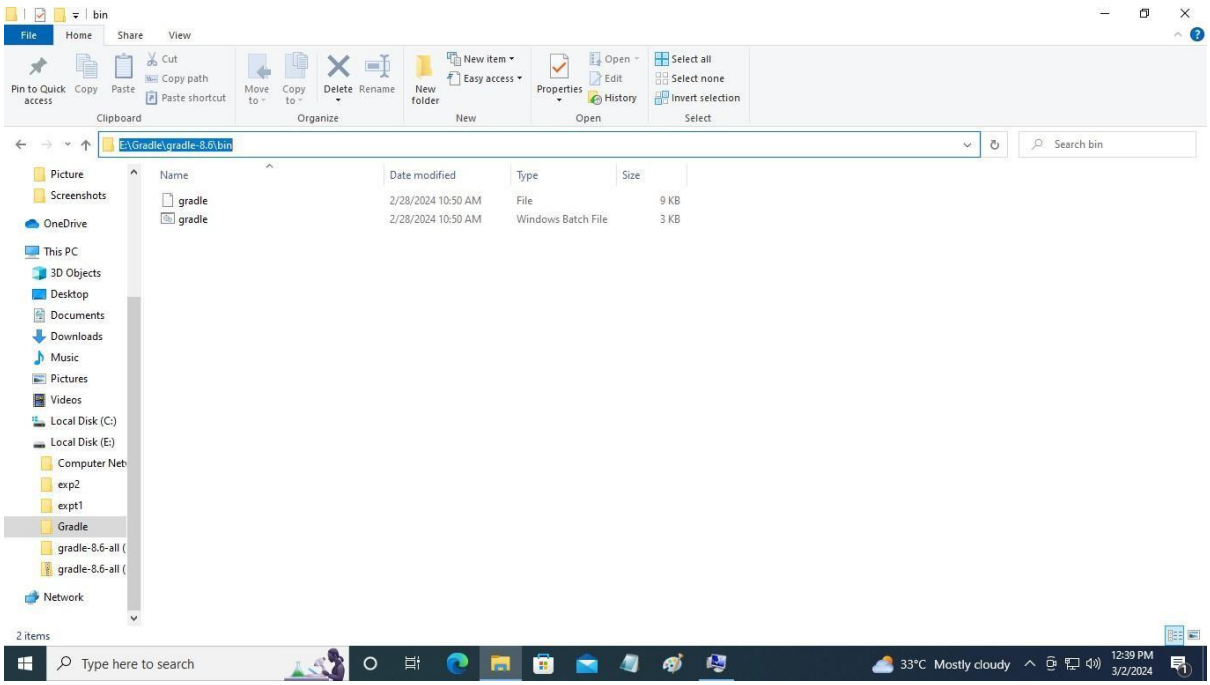
Step 2: Go to downloads and extract the gradle folder



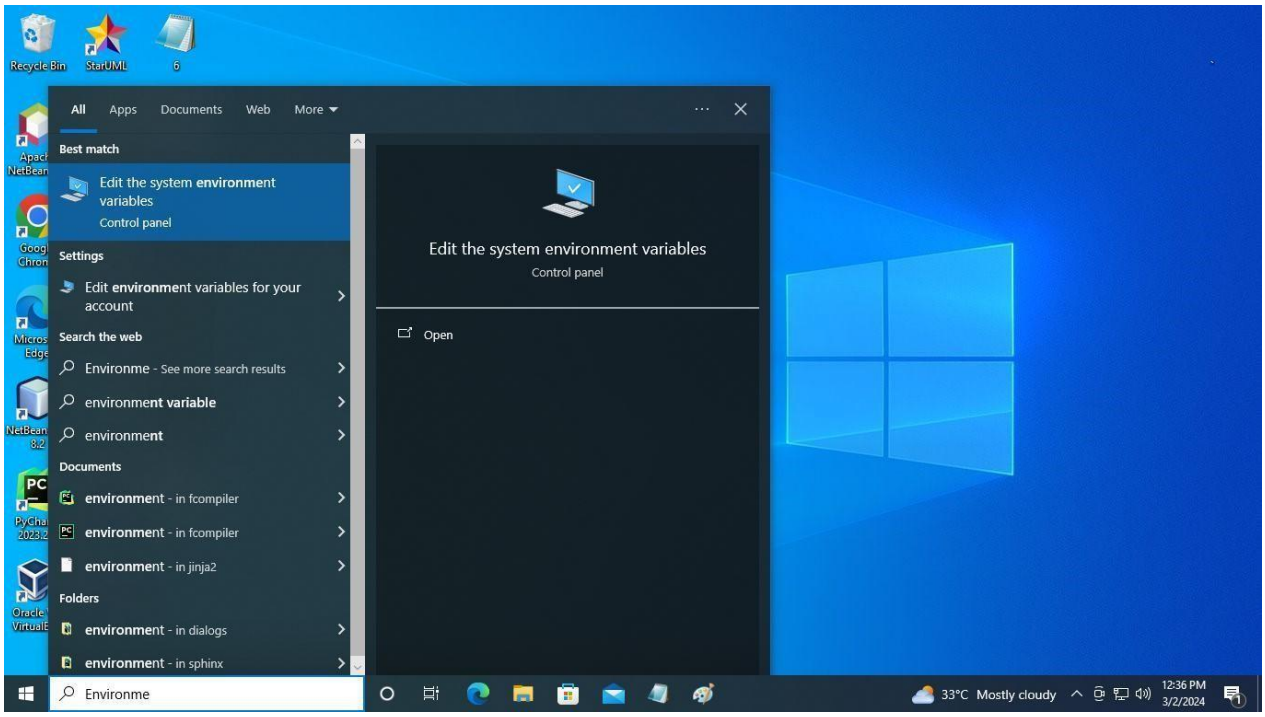
Step 3: Extracted gradle folder



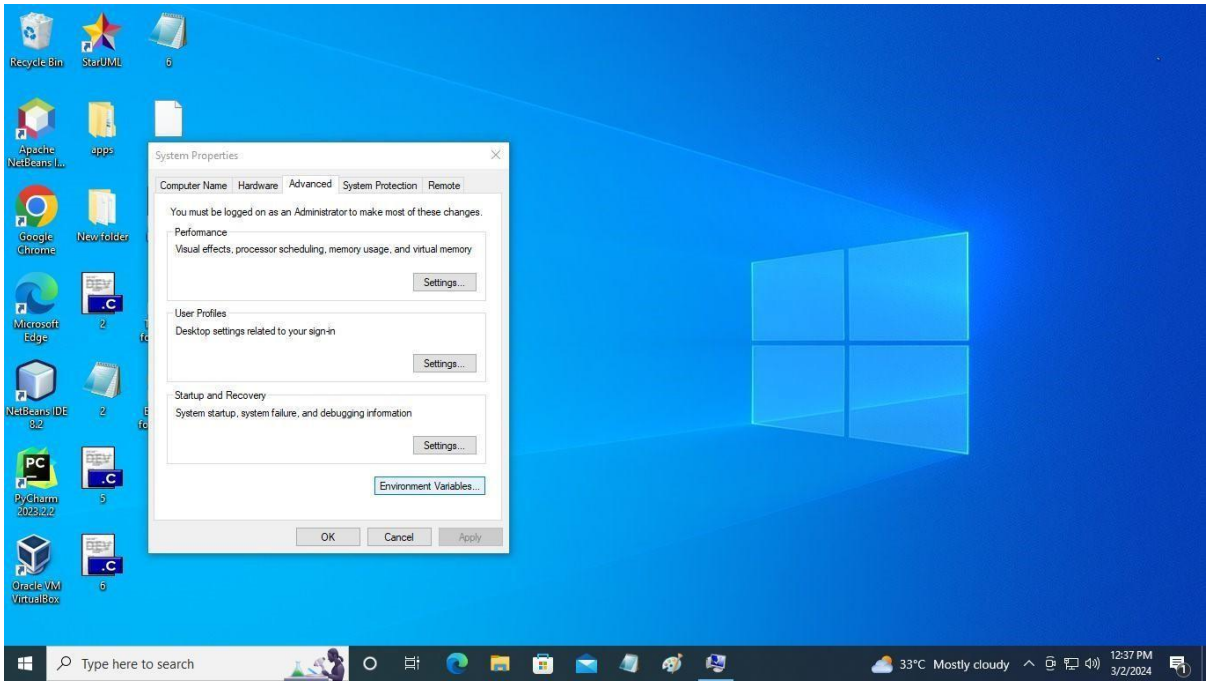
Step 4: Copy the bin path from gradle folder



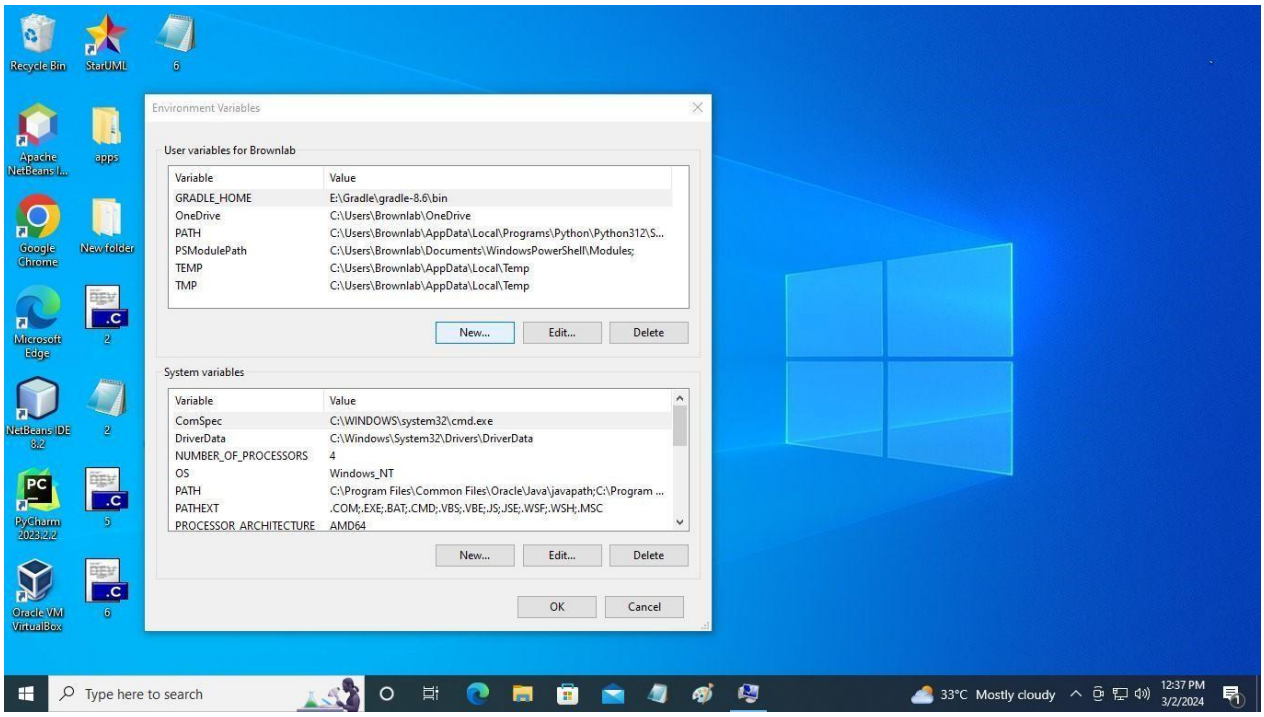
Step 5: Search environment variable to edit the path

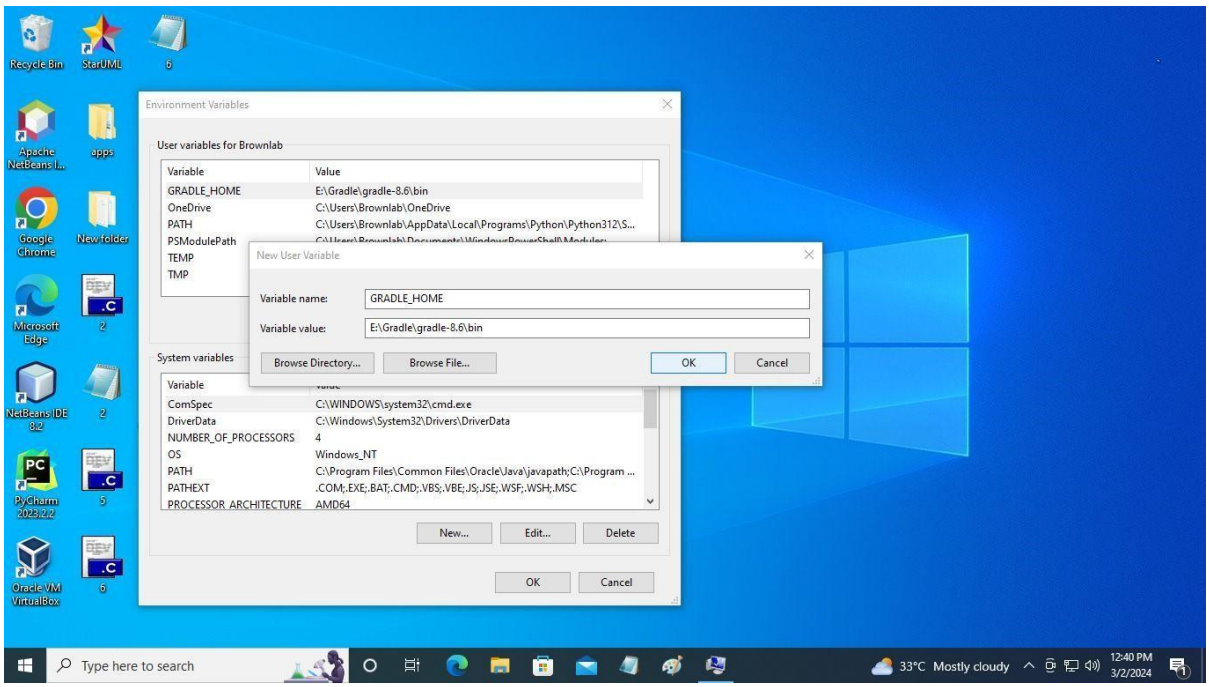


Step 6: Click environment variable

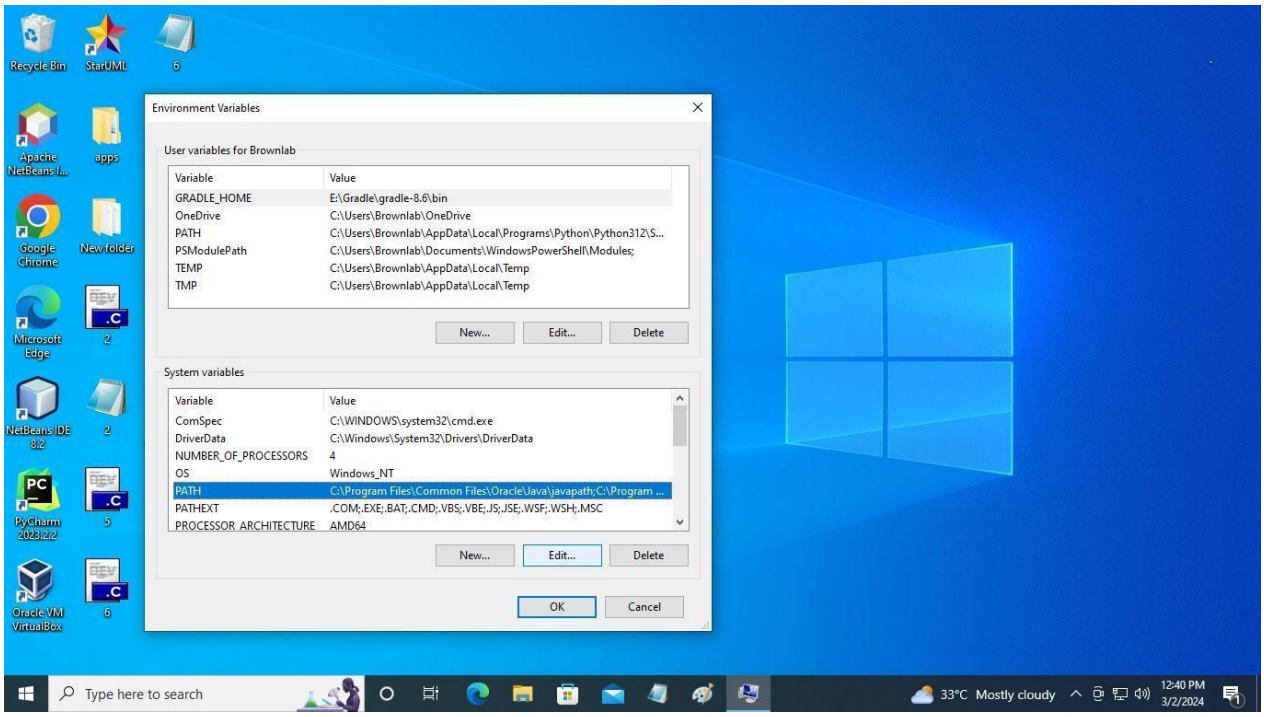


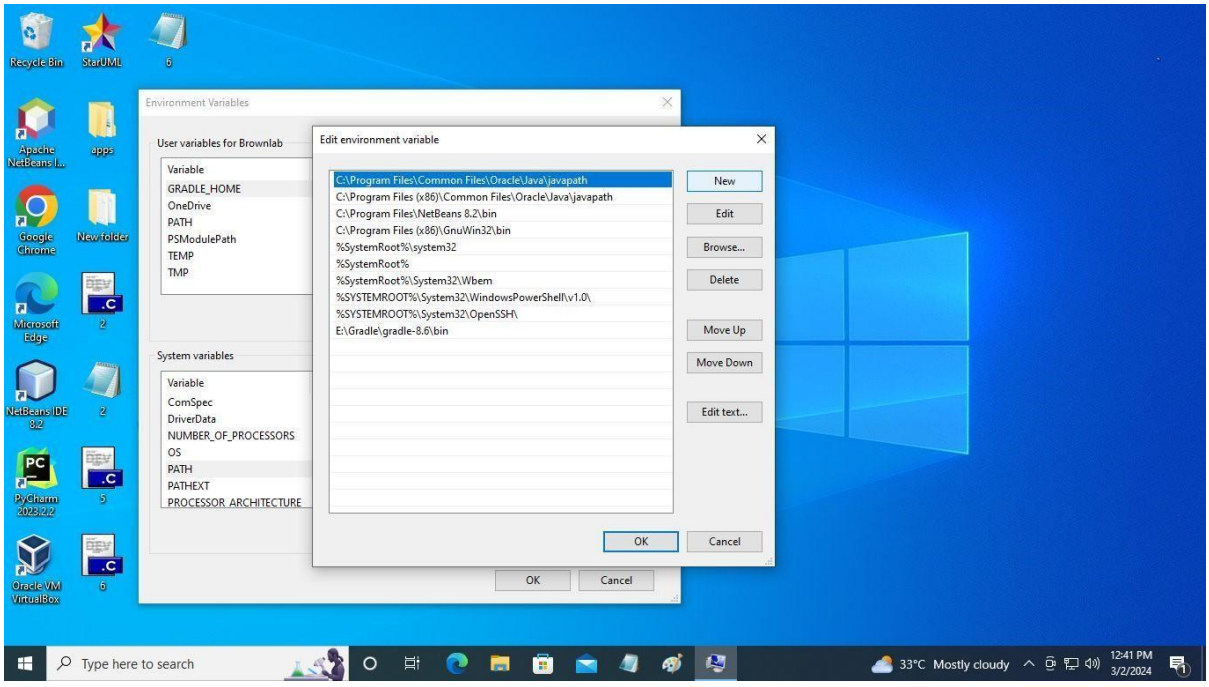
Step 7: Click new and paste the variable name & variable value as below



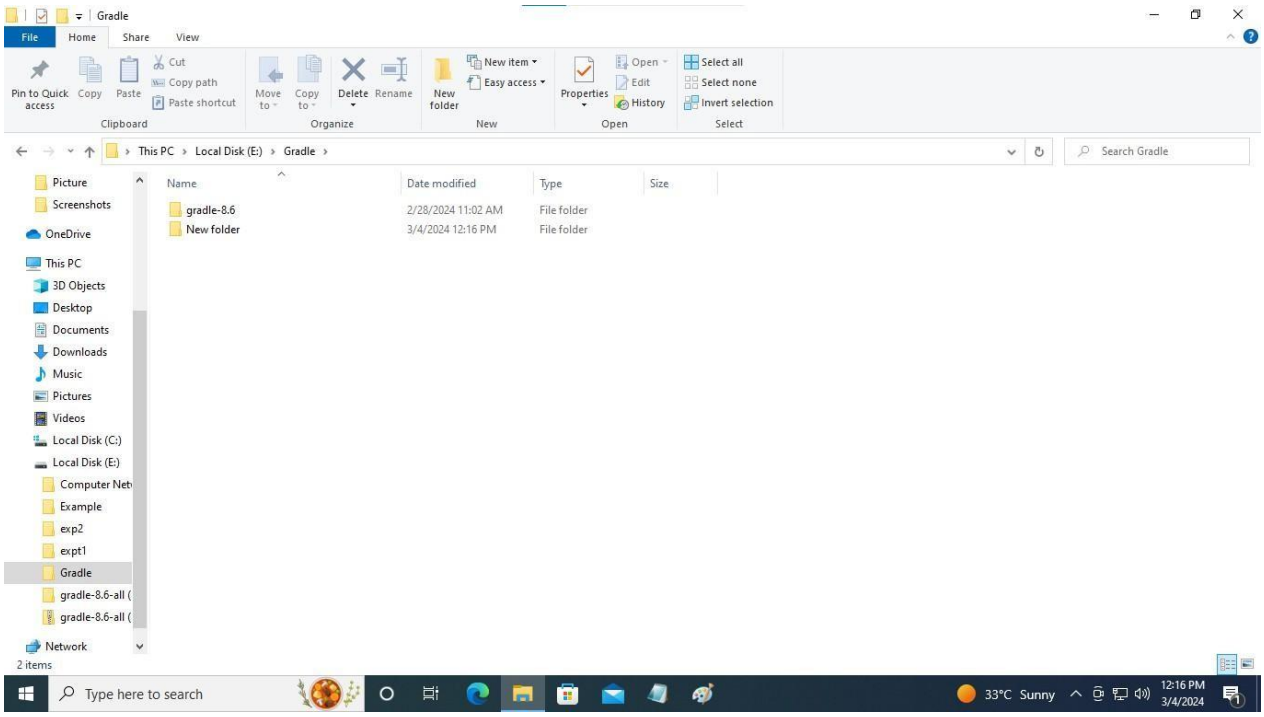


Step 8: In path click edit and paste gradle path

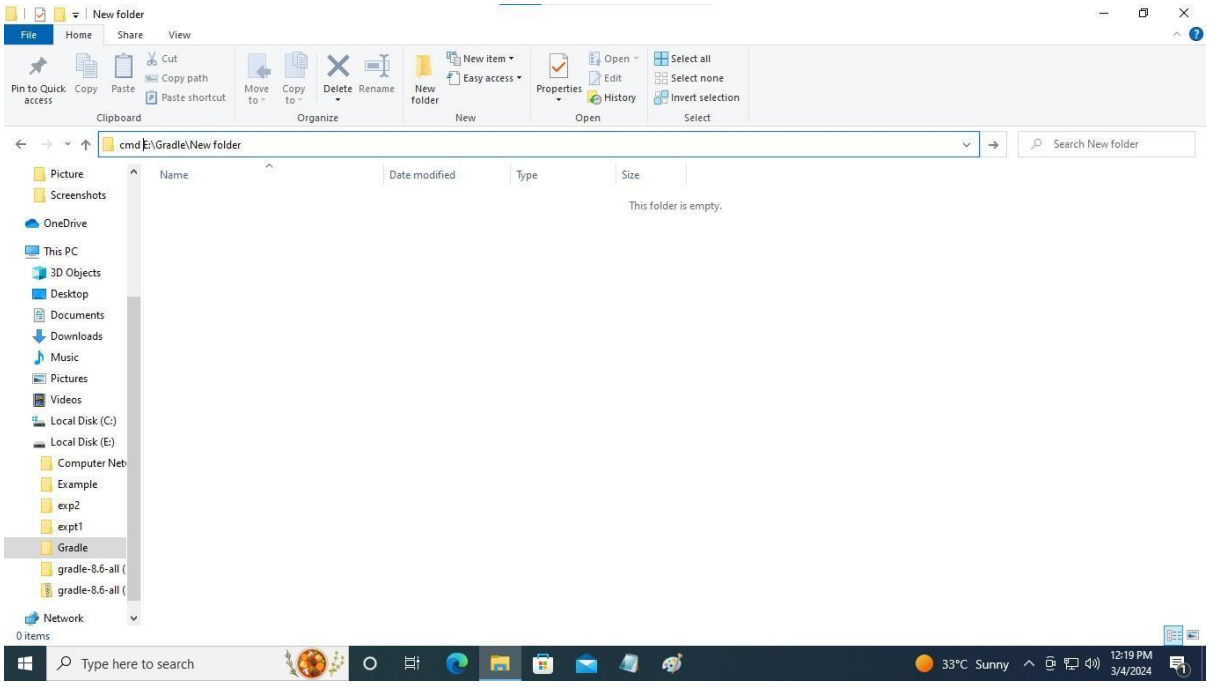




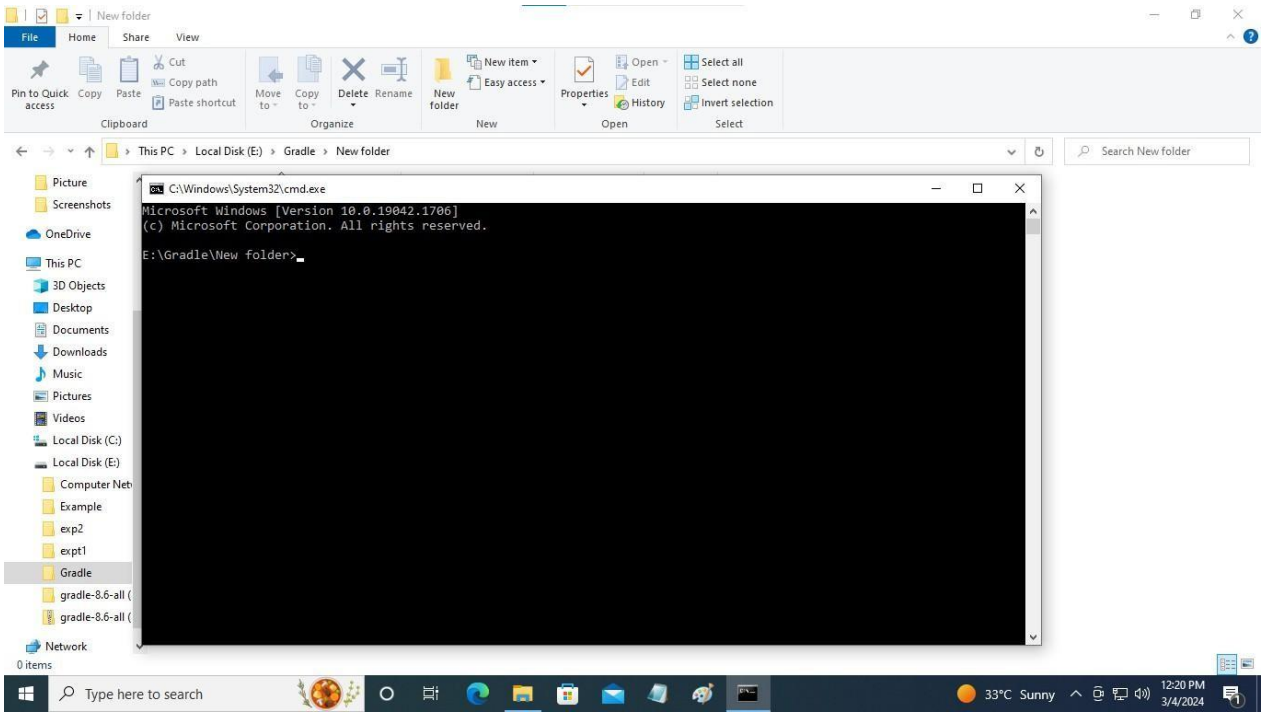
Step 9: Create a new folder in gradle



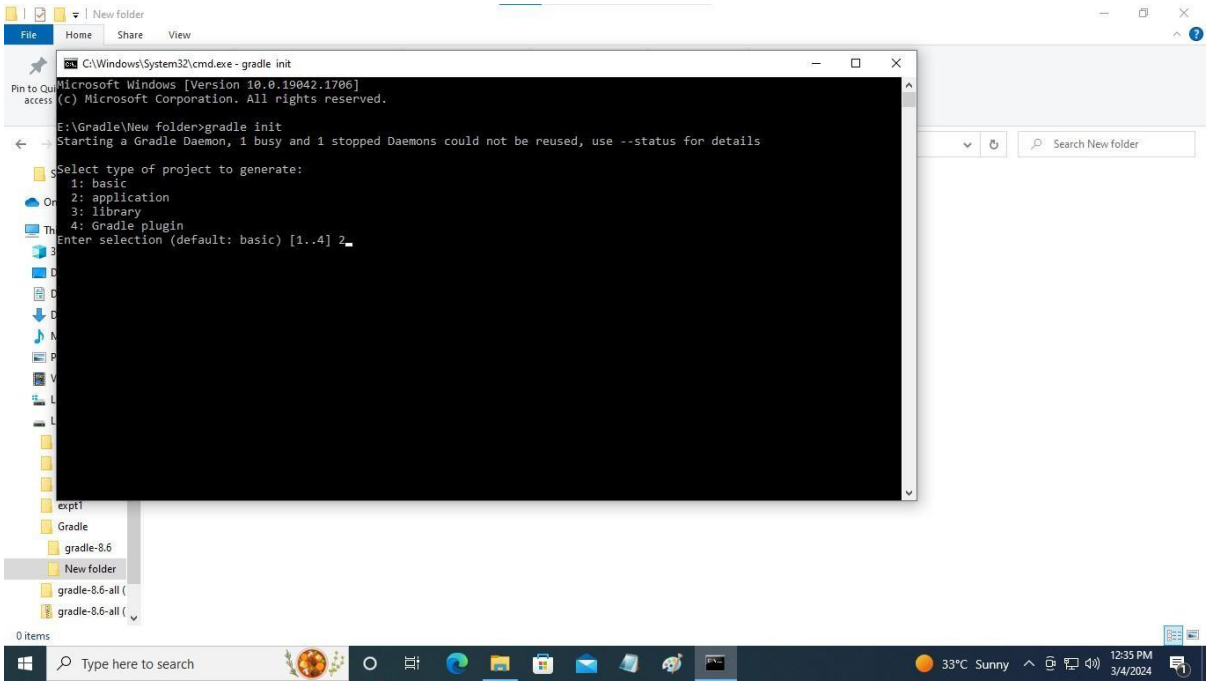
Step 10: Open command prompt in that folder refer below picture



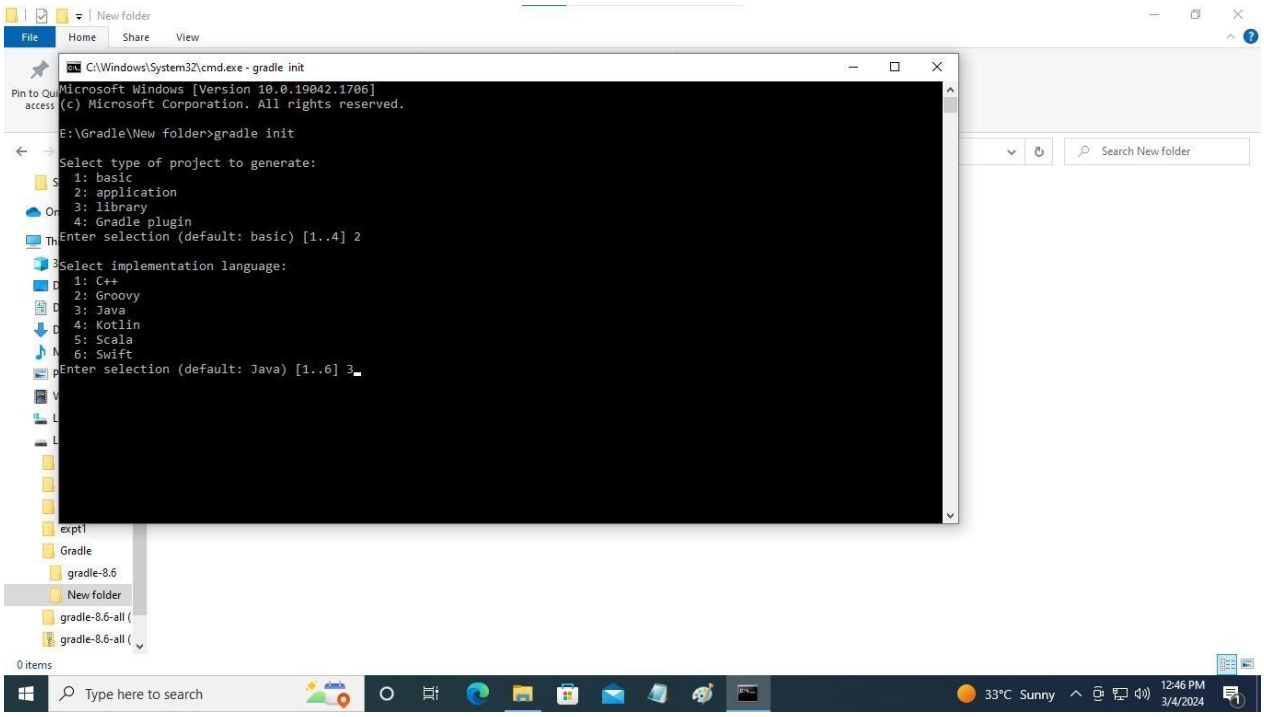
Step 11: Command prompt opened



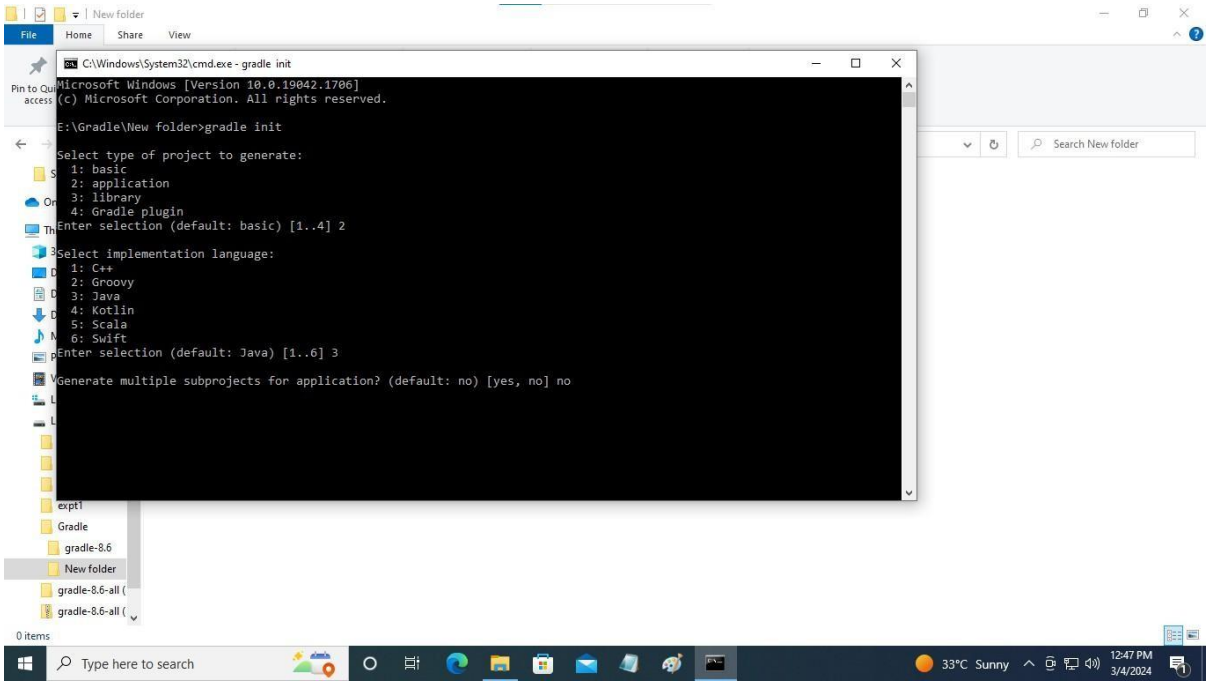
Step 12: Type the command gradle init and press enter .Now Select the type of project to generate and we choose number 2 [application]



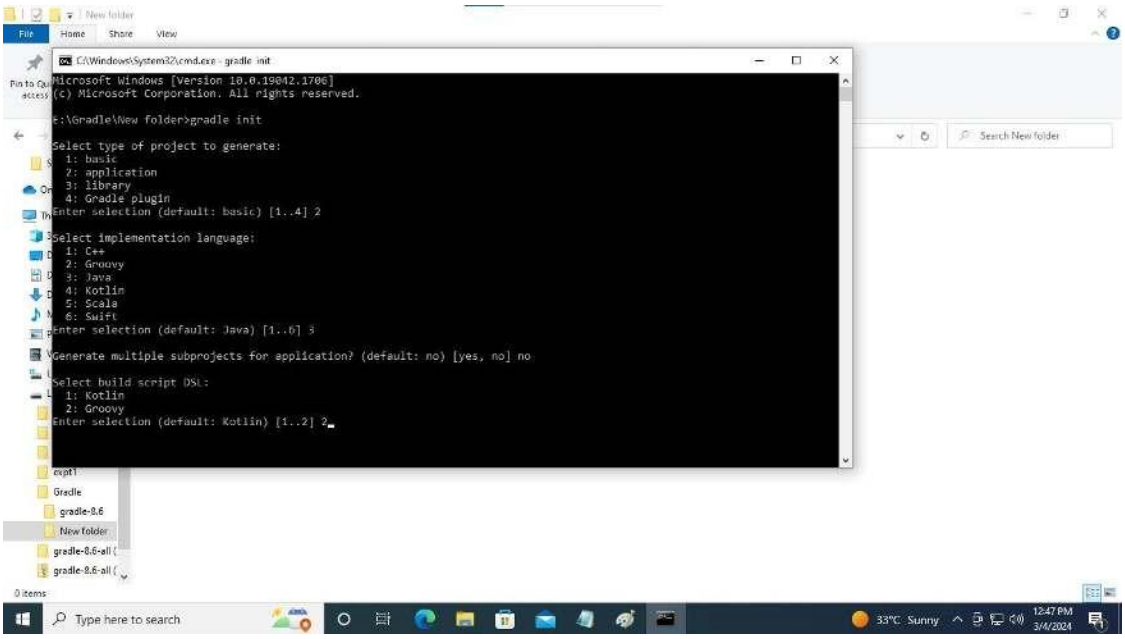
Step 13: Select the implementation language and we choose number 3 [Java]



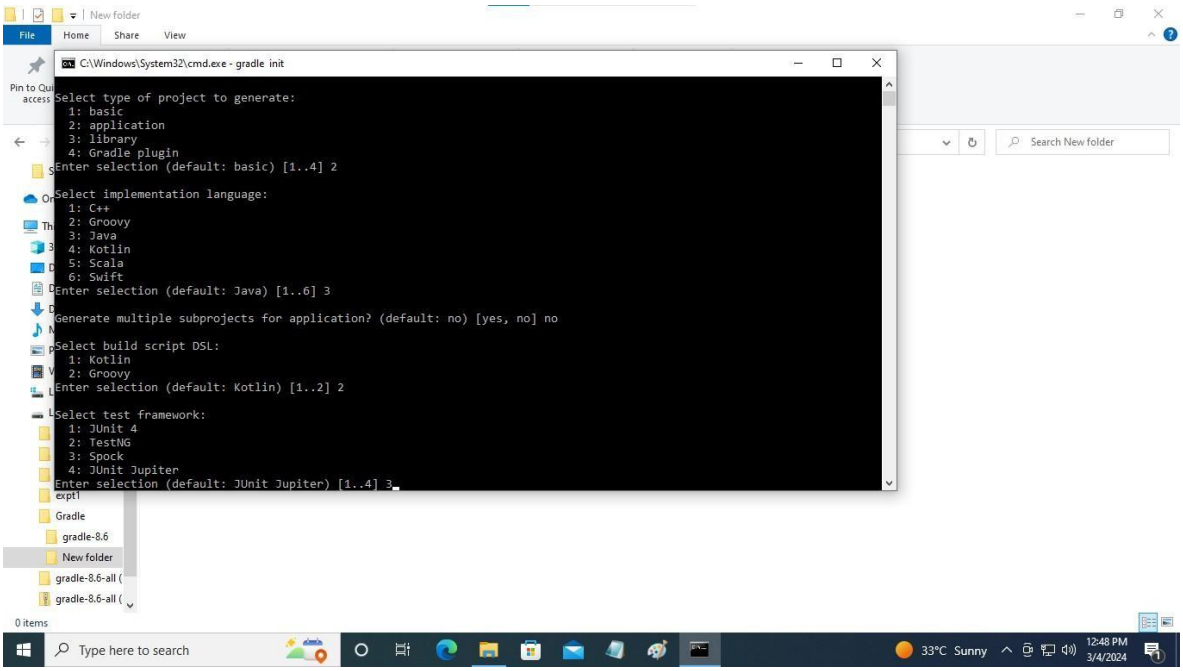
Step 14: Type NO to generate multiple subprojects for application



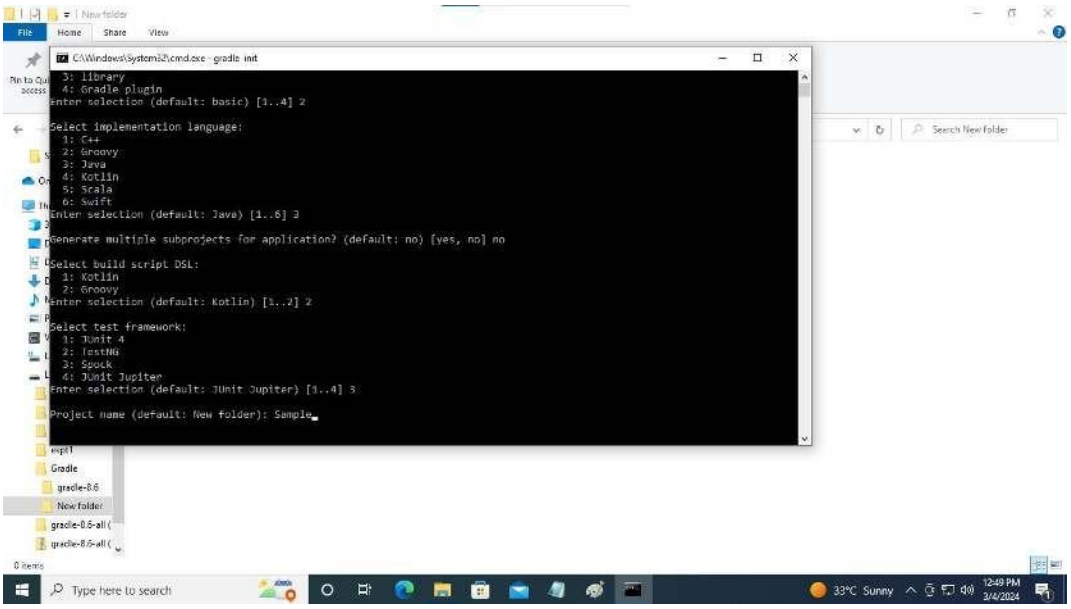
Step 15: Select build script DSL and we choose 2 [Groovy]

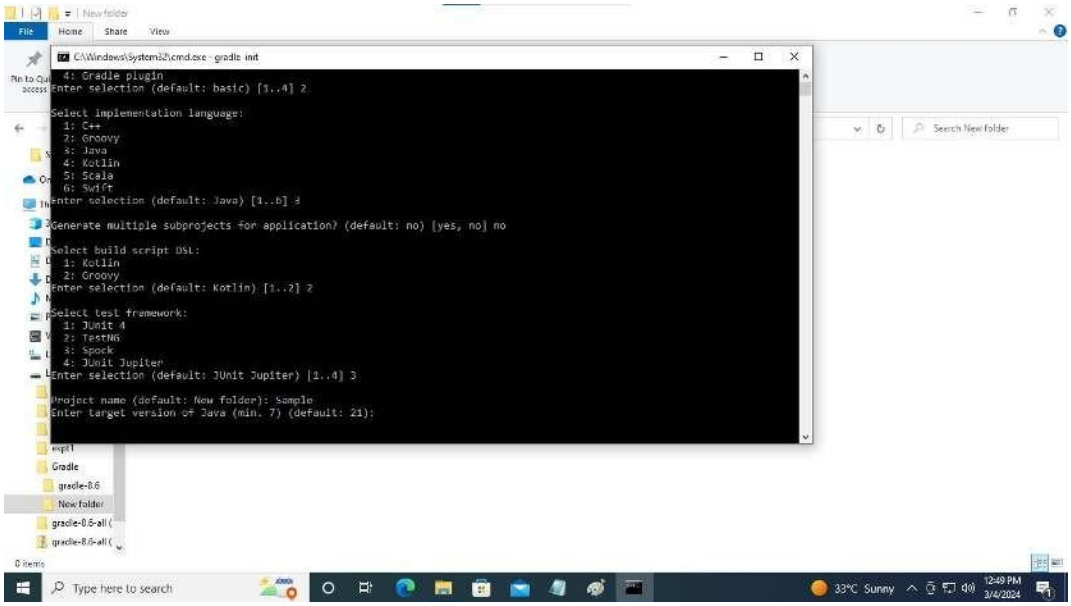


Step16: Select test framework and we choose number 3 [Spock]

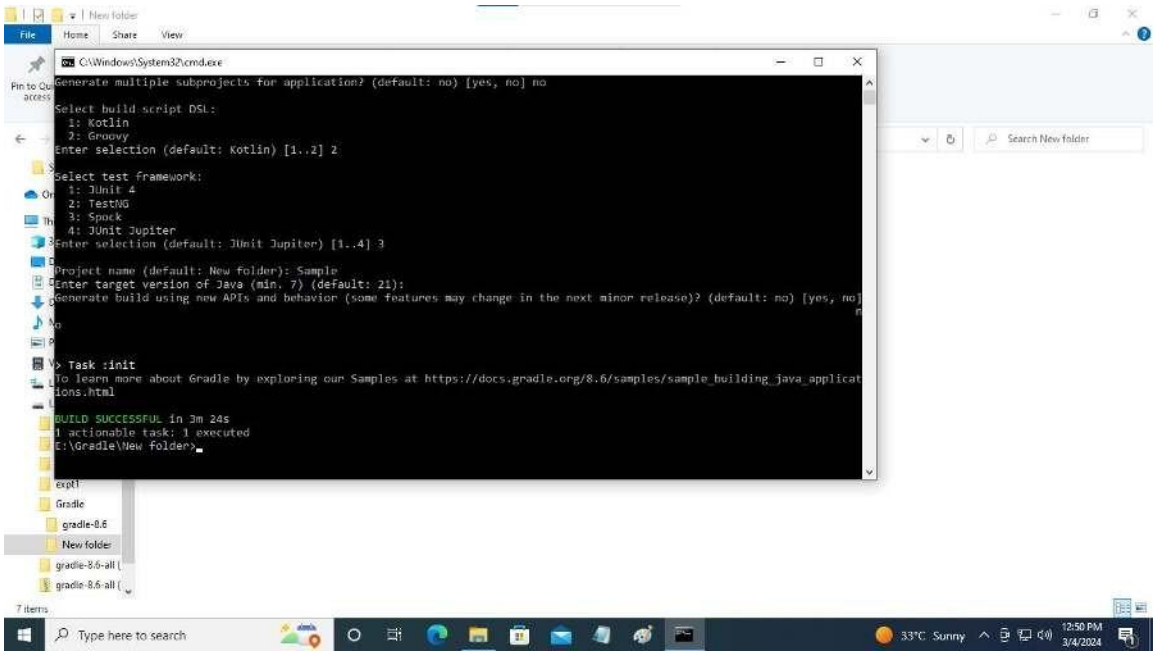


Step 17: Give project name as sample and enter

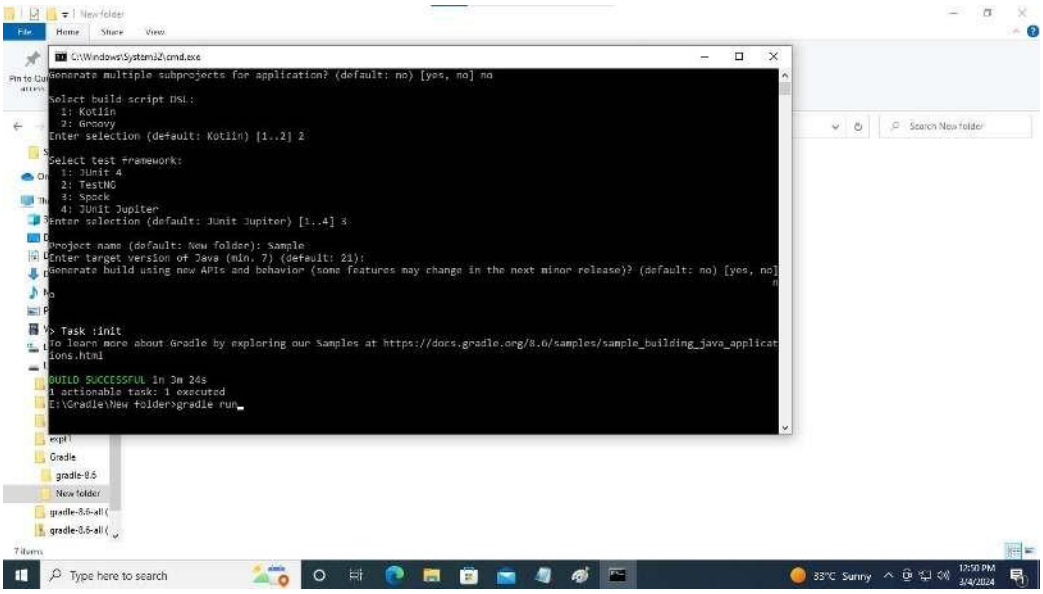




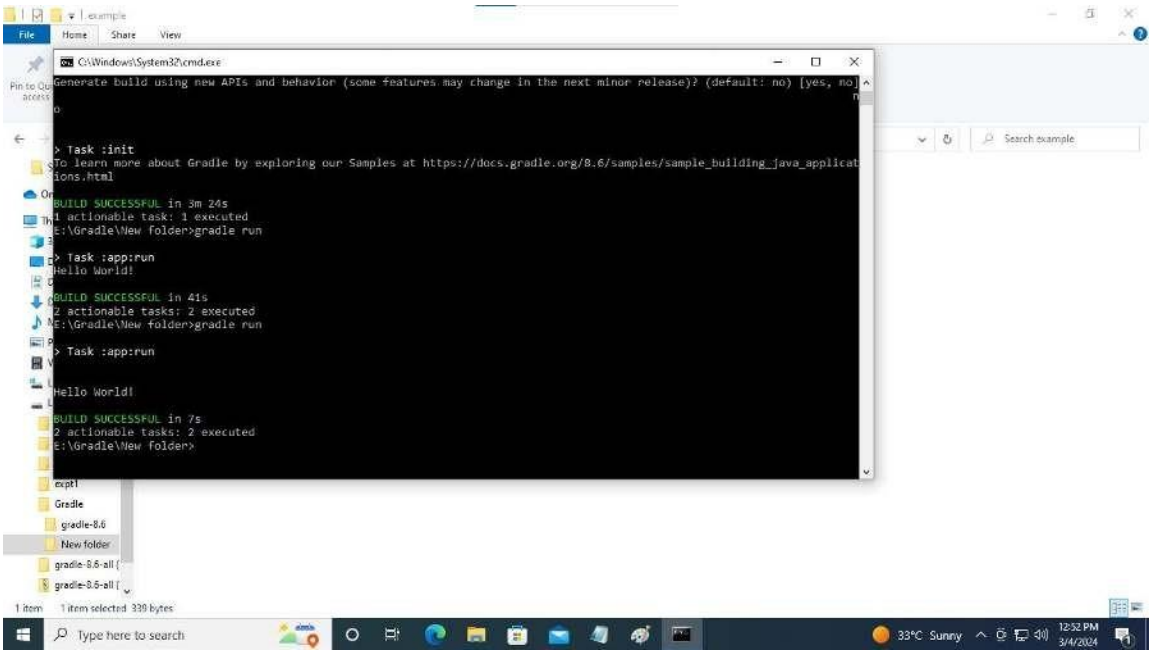
Step 18: Type NO to generate build using new APIs and it gives the Task: init and display the message as BUILD SUCCESSFUL



Step 19: Now run gradle using the command gradle run



Step 20: It gives the output as Hello World!



Result

Thus, a simple application using Gradle was Build successfully.

Ex.N0:6 **Install Ansible and Configure Ansible Roles and**

Date: **to write playbooks**

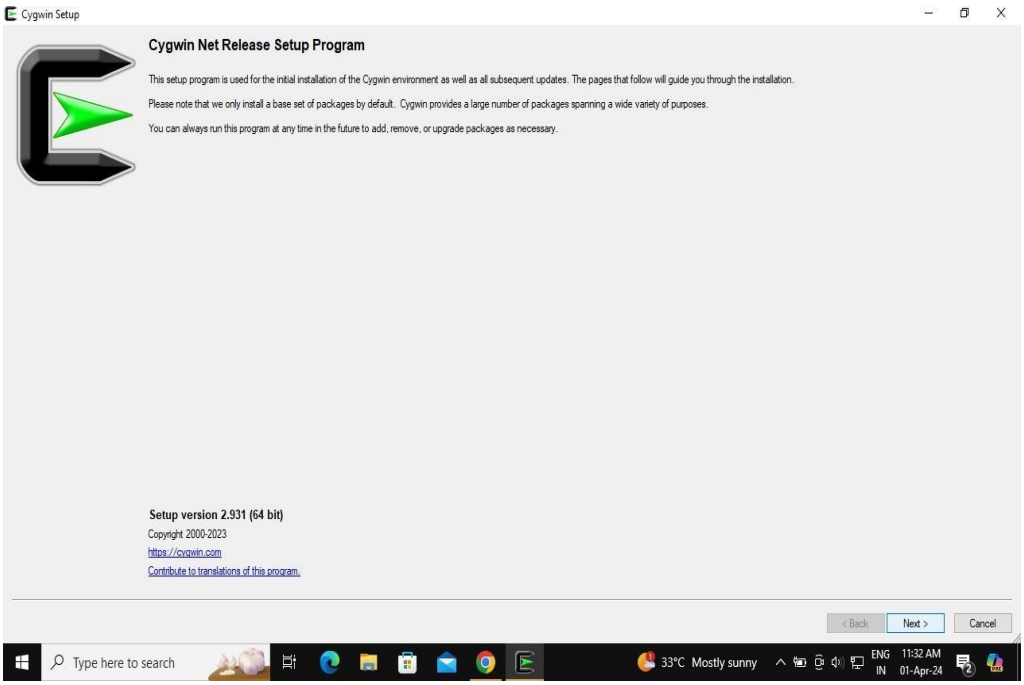
Aim:

To install Ansible ,Configure Ansible roles and to write playbooks

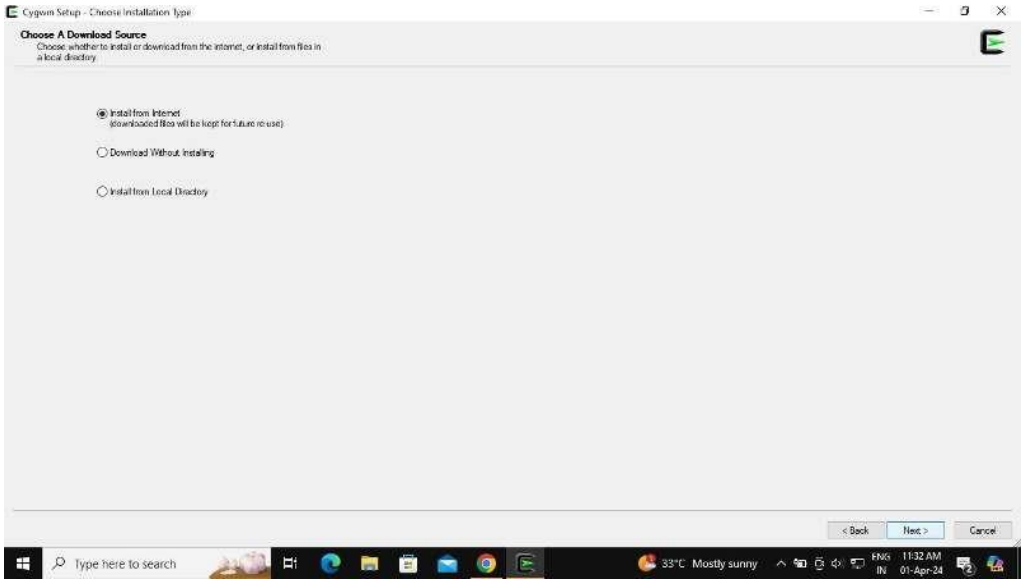
Procedure:

Step 1: Install Cygwin64 Terminal:

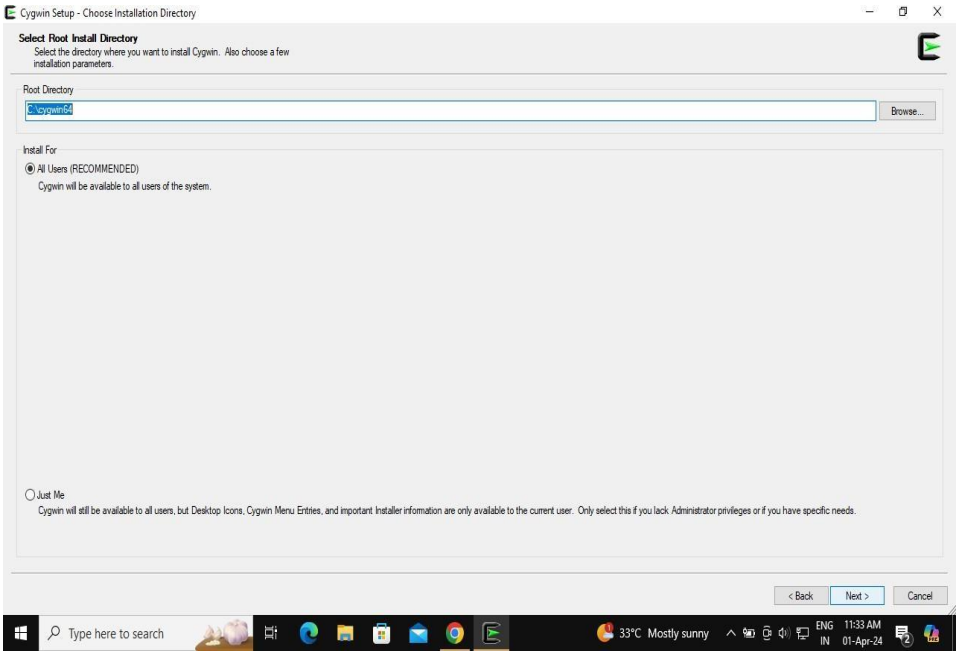
Step 2: Cygwin Net Release setup program



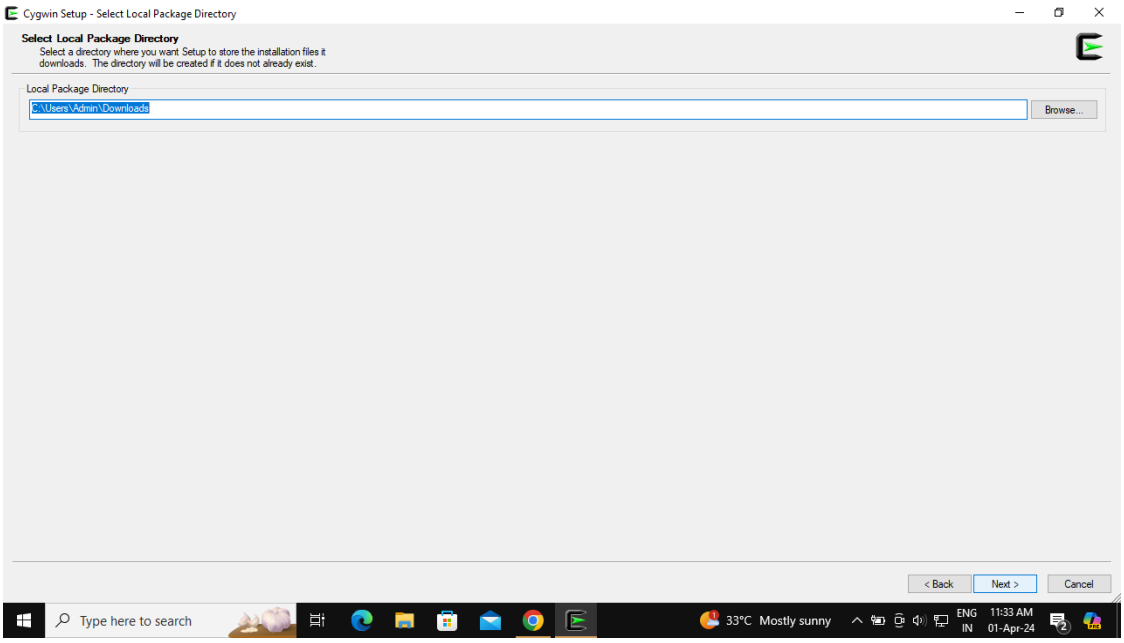
Step 3: Here we choose the source



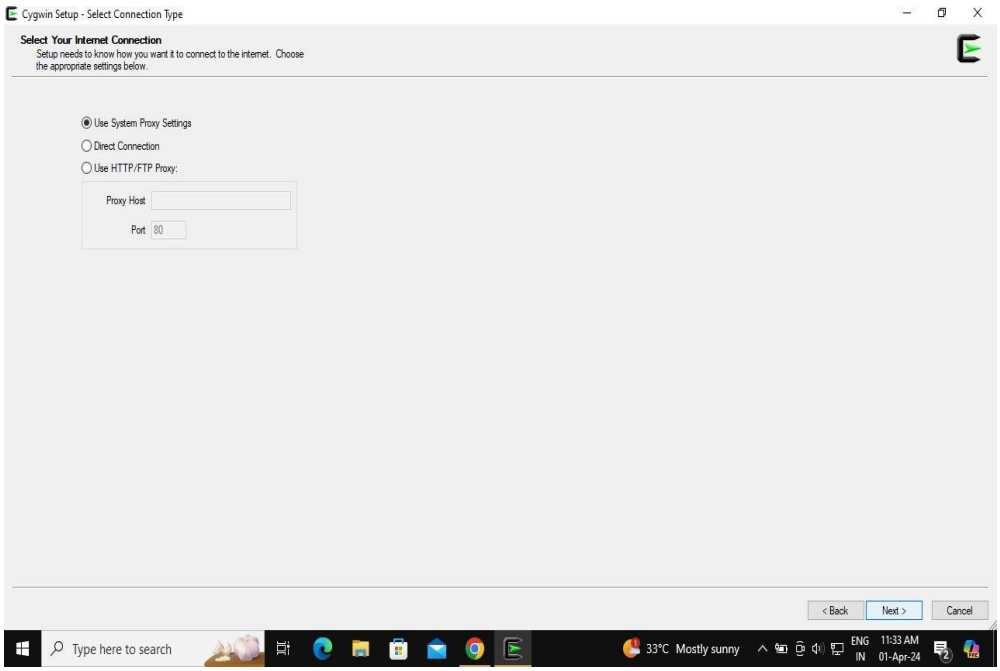
Step 4: Select root to install directory



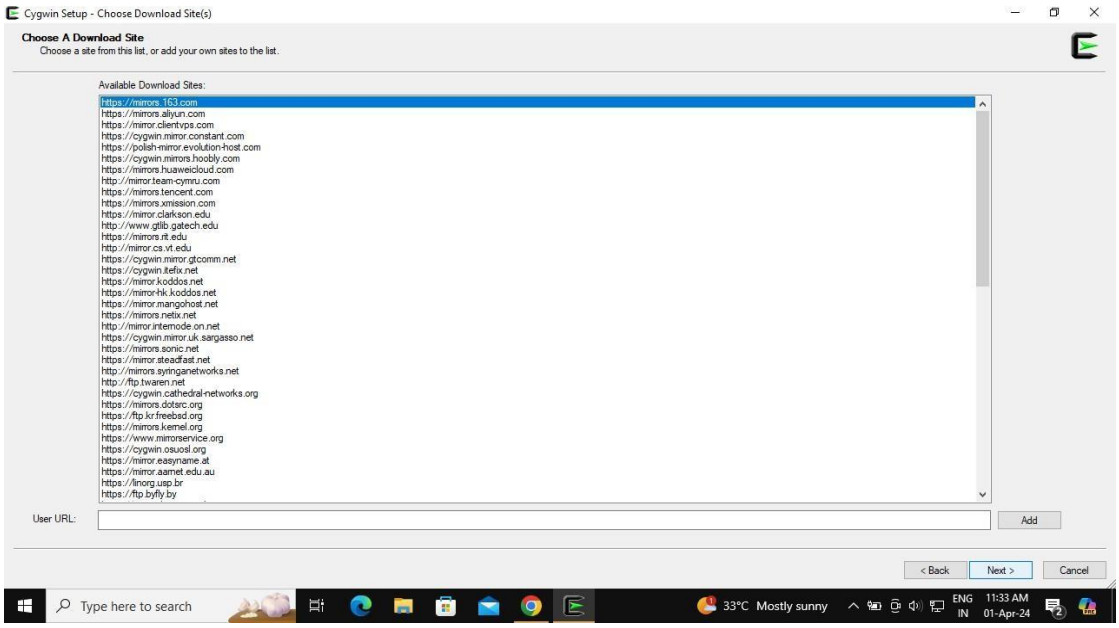
Step 5: Select local package directory



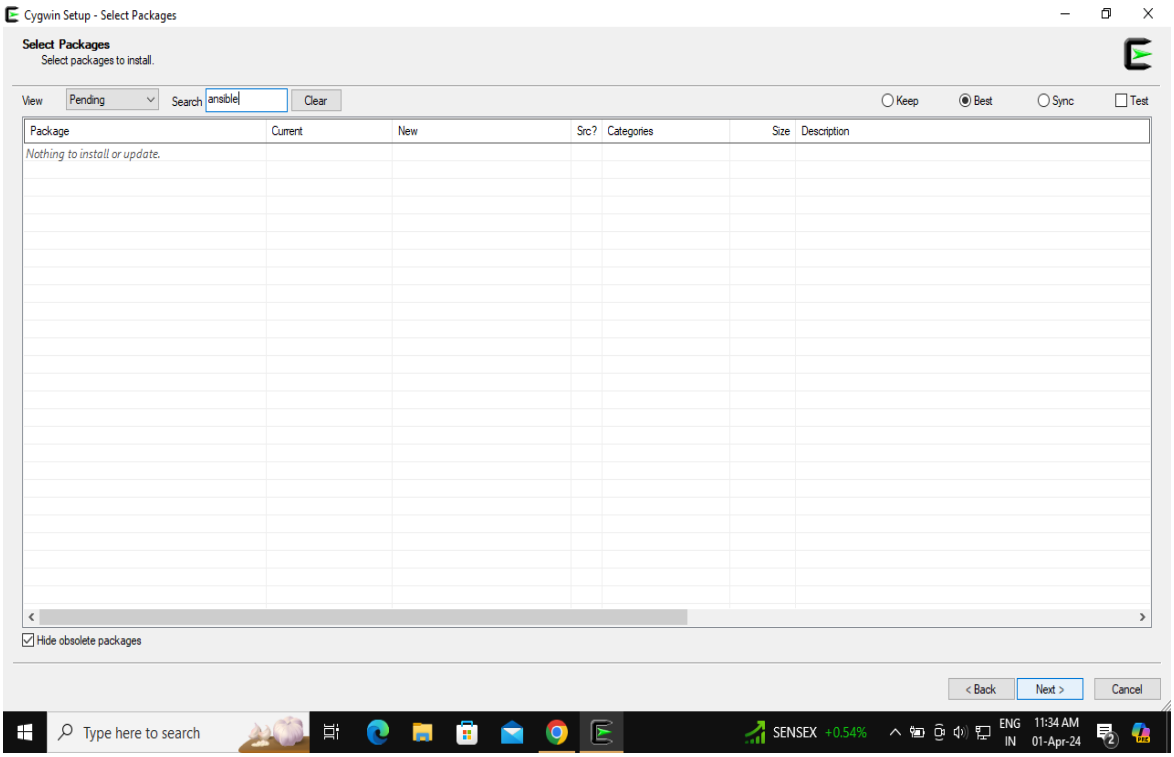
Step 6: Here select your internet connection



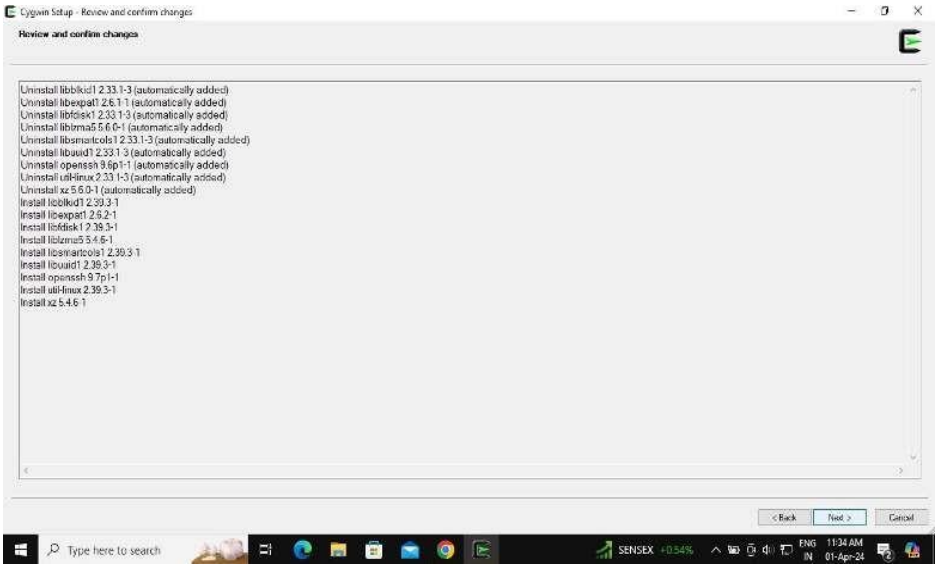
Step 7: Choose a downloaded site



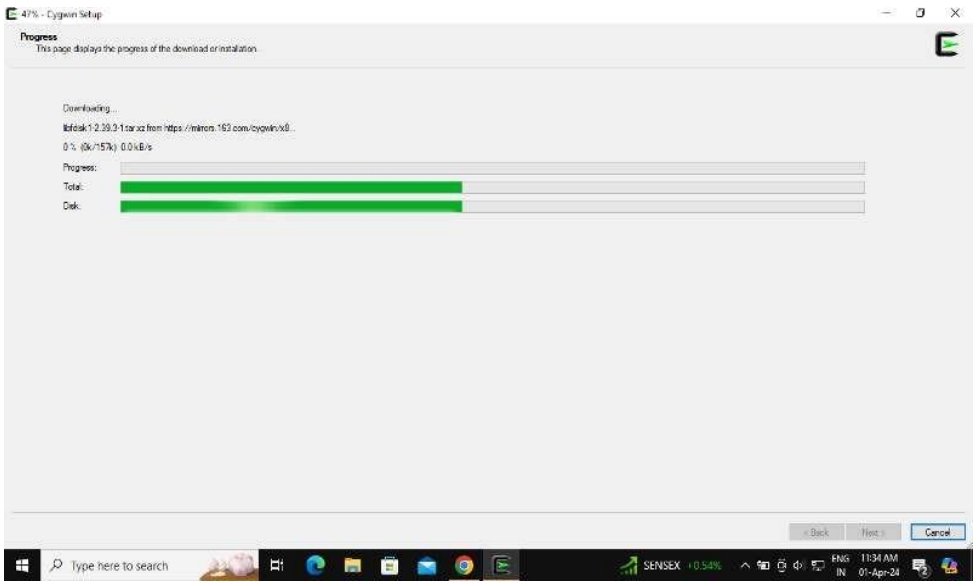
Step 8: Select packages to install



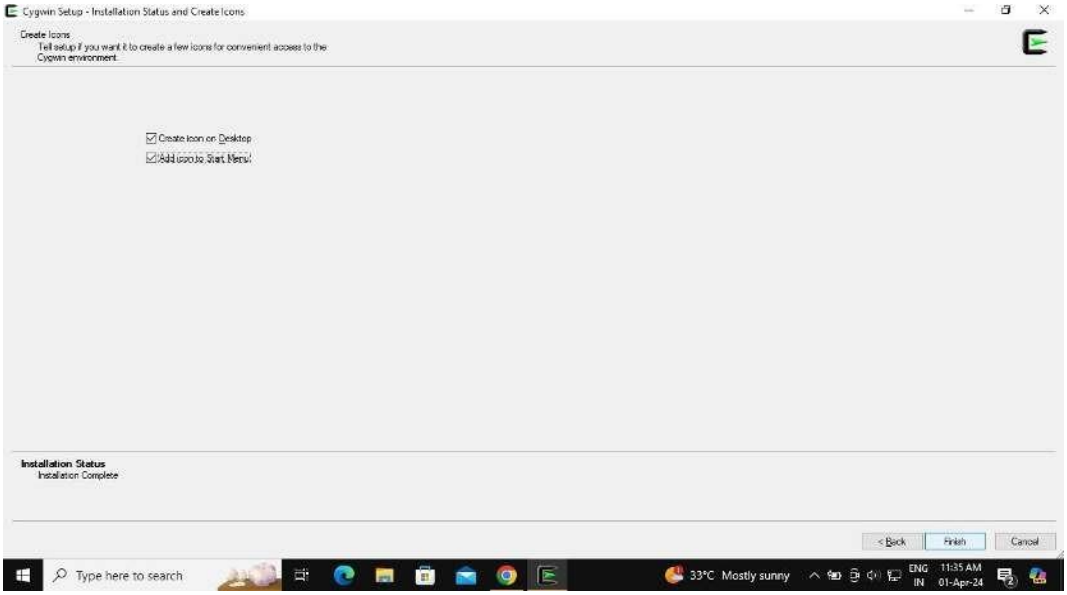
Step 9: Review and confirm changes



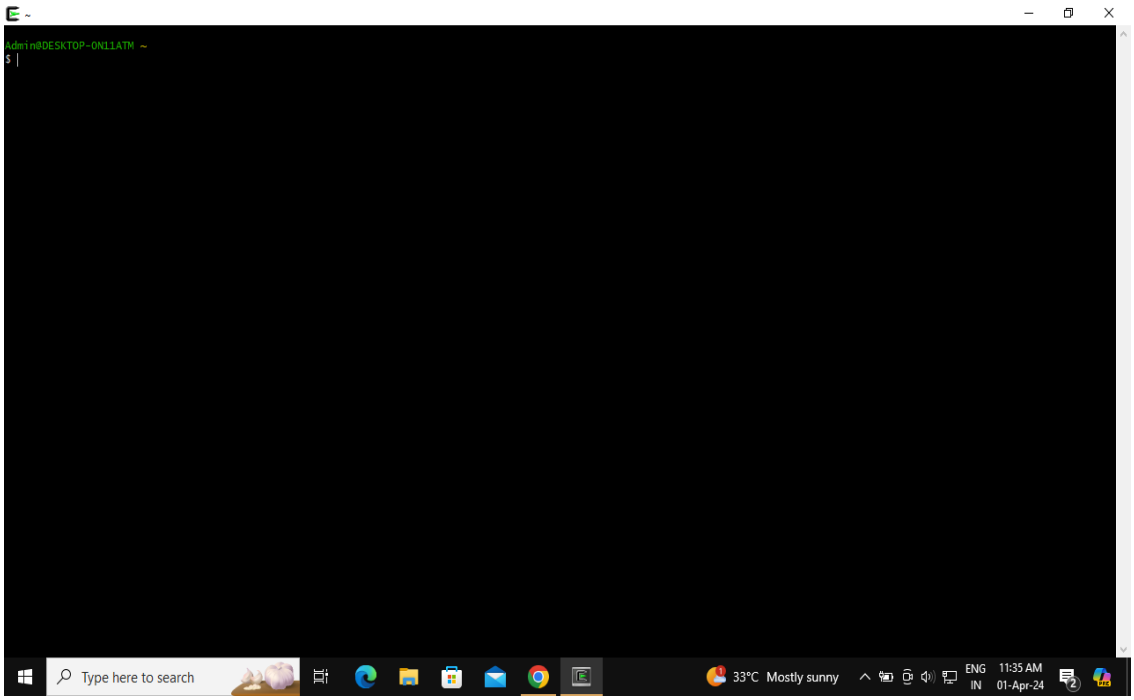
Step 10: Downloading



Step 11: installation complete

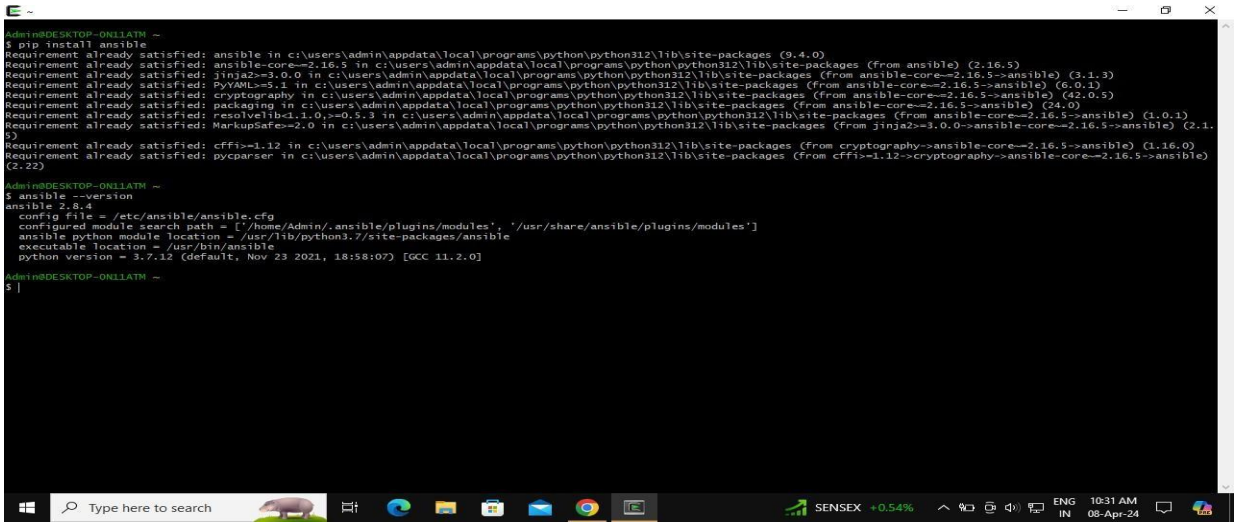


Step 12: The cygwin was installed successfully.



INSTALL ANSIBLE:

Step 13: Install Ansible in cygwin command prompt as pip install ansible



Step 14: Create Ansible roles Directory

Step 15: Using change directory command Create ansible roles

Step 16: Organize the role Structure

```
Admin@DESKTOP-ON11ATM ~/ansible-roles
$ cd common

Admin@DESKTOP-ON11ATM ~/ansible-roles/common
$ mkdir tasks handlers defaults files
mkdir: cannot create directory ‘tasks’: File exists
mkdir: cannot create directory ‘handlers’: File exists
mkdir: cannot create directory ‘defaults’: File exists
mkdir: cannot create directory ‘files’: File exists
```

Step 17: Write Tasks And Handlers

```
Admin@DESKTOP-ON11ATM ~/ansible-roles/common
$ mkdir -p ~/tasks

Admin@DESKTOP-ON11ATM ~/ansible-roles/common
$ cd ~/tasks
```

```
Admin@DESKTOP-ON11ATM ~/tasks
$ mkdir main.yml

Admin@DESKTOP-ON11ATM ~/tasks
$ cd main.yml

Admin@DESKTOP-ON11ATM ~/tasks/main.yml
$
```

Step 18: Create Ansible Playbooks Directory

```
Admin@DESKTOP-ON11ATM ~
$ mkdir ~ansible-playbooks

Admin@DESKTOP-ON11ATM ~
$ cd ~/ansible-playbooks

Admin@DESKTOP-ON11ATM ~/ansible-playbooks
$
```

Step 19: Open a notebook and type the program save it as my_playbook.yml

my_playbook - Notepad

File Edit Format View Help

```
---
- name: Example Playbook
  hosts: localhost
  tasks:
    - name: Print Hello World
      debug:
        msg: "Hello, World!"
```

Step 20: Run the program and we get the output

```
Admin@DESKTOP-0N11ATM ~
$ cd ansible-playbook

Admin@DESKTOP-0N11ATM ~/ansible-playbook
$ ansible-playbook my_playbook.yml
 [WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [Example Playbook] ********************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]

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

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


Admin@DESKTOP-0N11ATM ~/ansible-playbook
$
```

Result:

Thus, installing ansible, configure ansible roles and to write playbooks done successfully.

EX.No: **Create a CD pipeline in Jenkins and deploy in Cloud**

Date:

Aim :

To create a Continuous Deployment (CD) pipeline using Jenkins for automating the deployment of applications on a cloud platform, ensuring faster and reliable delivery of updates.

Procedure :

Step 1: Set Up Jenkins

- Install Jenkins on a server or use a cloud-hosted Jenkins instance.
- Configure Jenkins by installing necessary plugins (e.g., Git, Pipeline, Cloud-specific deployment plugins).
- Create a Jenkins user with appropriate permissions.

Step 2: Prepare the Cloud Environment

- Choose a cloud provider (e.g., AWS, Azure, GCP).
- Set up the infrastructure required for deployment (e.g., virtual machines, Kubernetes clusters, or app services).
- Generate and securely store the cloud credentials (API keys, access tokens).

Step 3: Create a Jenkins Pipeline

- Navigate to Jenkins and create a new pipeline project.
- Define the pipeline stages in a Jenkinsfile using the following typical structure:
    - Checkout Code: Pull code from the version control system (e.g., GitHub).
    - Build: Compile and build the application (e.g., Maven, Gradle).
    - Test: Run automated tests.
    - Deploy: Deploy the application to the cloud environment.

Step 4: Write the Jenkinsfile

Example structure:

```
pipeline {
    agent any
    stages {
        stage('Checkout Code') {
            steps {
                git branch: 'main', url: 'https://github.com/username/repository.git'
            }
        }
        stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                sh 'aws deploy push --application-name AppName --s3-location
s3://bucket-name/'
            }
        }
    }
}
```

Step 5: Integrate Jenkins with Cloud

- Configure cloud provider plugins in Jenkins (e.g., AWS CLI, Azure DevOps).
- Store the cloud credentials in Jenkins using the Credentials Manager.
- Test connectivity between Jenkins and the cloud environment.

Step 6 : Trigger and Monitor the Pipeline

- Start the pipeline manually or configure it to trigger on code commits or schedule.
- Monitor each stage in the Jenkins dashboard for successful execution.

Step 7: Verify Deployment

- Access the deployed application in the cloud environment.
- Perform manual or automated tests to confirm the deployment was successful.

Result :

The experiment successfully demonstrates the creation of a Continuous Deployment pipeline using Jenkins and its integration with a cloud platform.

EX.No: Create an Ansible playbook for a simple web application infrastructure

Date:

Aim :

To create an Ansible playbook for provisioning and configuring a basic web application infrastructure, including a web server and database server, ensuring consistency and automation in deployment.

Procedure :

Step 1: **Set Up Ansible**

- Install Ansible on a control node (e.g., Linux system).
- Configure SSH access to managed nodes (web server and database server) by setting up passwordless authentication using SSH keys.
- Define the inventory file (hosts) to list the target nodes.

Step 2: **Define the Infrastructure**

- Decide the components of the infrastructure:
  - A web server (e.g., Nginx or Apache).
  - A database server (e.g., MySQL).
- Identify required software packages and configurations.

Step 3: **Create an Inventory File**

- Example hosts file:

[web]

192.168.1.10

[db]

192.168.1.11

Step 4: **Write the Ansible Playbook**

- Create a YAML file (e.g., web_app_playbook.yml).
- Example playbook:

```
- name: Set up a simple web application infrastructure
hosts: all
become: yes
tasks:
  - name: Update the system
    apt:
      update_cache: yes
      upgrade: dist

  - name: Install web server
    when: "'web' in group_names"
    apt:
      name: apache2
      state: present

  - name: Copy website files
    when: "'web' in group_names"
    copy:
      src: /path/to/website/
      dest: /var/www/html/
      owner: www-data
      group: www-data
      mode: '0644'

  - name: Install database server
    when: "'db' in group_names"
    apt:
```

```
    -name: mysql-server
    state: present

- name: Start services
  service:
    name: "{{ item }}"
  state: started
  enabled: yes
  with_items:
    - apache2
    - mysql
```

Step 5:

Execute the Playbook

- Run the playbook using the following command:

bash

ansible-playbook -i hosts web_app_playbook.yml

Step 6: **Verify the Infrastructure**

- Check the web server by accessing the application in a browser using the web server's IP address.
- Verify the database server by logging into MySQL and ensuring it is functional.

Step 7: **Test and Validate**

- Test the web application for functionality.
- Confirm the infrastructure meets the requirements and is consistent across the nodes.

Result :

The Ansible playbook successfully provisions and configures a simple web application infrastructure.