Repeat experiment 6, meaning

1) Create a Maven project by executing the following command

mvn archetype:generate –DgroupId=in.eg -DartifactId=maven4 -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

2) Go inside the newly created project by executing the following command

cd maven4

3) Package the newly created project by executing the following command

mvn package

You will get an error because of java version on executing the above command. For that,

4) Open pom.xml and type the following code (after version tag (or) after name tag)

```
<properties>
   <maven.compiler.source>7</maven.compiler.source>
   <maven.compiler.target>1.7</maven.compiler.target>
</properties>
```

5) Again package the project using the command

mvn package

6) See the list of options that can be used with git by typing

git

7) Add your email by executing the following command

git config --global user.email “Your Github email”

8) Add your name by executing the following command

git config --global user.name “Your Github username”

9) Initialize the Git repository by executing the following command

git init

10) Add pom.xml file by executing the following command

git add pom.xml

11) Add the source folder by executing the following command

git add src

12) Generate one SSH key by executing the following command

ssh-keygen -t ed25519 -C “Your Github email”

For enter file, don’t give anything just press Enter

For enter passphrase, don’t give anything just press Enter

For enter same passphrase again, don’t give anything just press Enter

13) Display the contents of the public key file id_ed25519.pub by executing the following command

sudo cat /home/student/.ssh/id_ed25519.pub

Select the contents of this public key from the terminal, copy and paste it in the 3rd input by logging in to your Github account and in the dashboard, click on the profile pic icon in the top right corner --> Settings --> SSH and GPG keys --> New SSH Key.

Also, you have to enter a key name. For example, mykey or key1

14) Give a commit message

git commit -m “Your commit message”

Create a repository by logging in to your Github account and in the dashboard left side, click on New -> enter a repository name, Give a description (optional) and tick the checkbox “Add a Readme file” and click on the green button “Create repository”

15) Set the origin to your Github account and repository

git remote add origin git@github.com:<Your Github username>/<Your repository name>

16) Set push to Github for saving code to Github repository

git config --global push.autoSetupRemote true

17) Set the branch to main

git branch -M main

18) Push the code to Github repository

git push origin main

19) Rebase your repository to point to the main branch

git pull --rebase origin main

20) Again execute Step# 18

21) In case you executed Step #15 more than once, then execute the command

git remote set-url origin git@github.com:<Your Github username>/<Your repository name> and execute the commands from Step# 16 to Step #20

22) Create an empty jar file, for example t.jar in /home/student path by executing the command

gedit t.jar (and enter any character. Then, save and close the file)

22) Then, see the status of Jenkins by typing the following command

sudo systemctl status jenkins

(If you get a green symbol, it means Jenkins is active/running)

23) Type localhost:8080 in the browser and enter

Username : admin

Password : The password which you get after typing the command

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

24) Click on “New Item” -> Enter an item name like AnsiDeploy

25) Select Pipeline and click on Ok

26) Scroll down and enter the pipeline script in the textarea as shown below

```
pipeline
{
    agent any
    stages
    {
        stage(“Checkout”)
        {
           steps
          {
               git branch: ‘main’, url : ‘<Your github url which ends with .git>’
          }
        }
```

```
stage(“Build”)
{
   steps
  {
       sh ‘mvn clean package’
  }
}

stage(“Test”)
{
   steps
  {
       sh ‘mvn test’
  }
}

 stage(“Archive Artifacts”)
 {
     steps
     {
       archiveArtifacts artifacts: '**/target/*.jar', allowEmptyArchive: true
     }
 }
```

```
        stage("Deploy")
        {
            steps
            {
                sh """
                    export ANSIBLE_HOST_KEY_CHECKING=False
                    ansible-playbook -i hosts.ini mydeploy.yml --extra-vars='ansible_become_pass=exam@cse'
                """
            }
        }
    }
```

and click on Save

27) Create an inventory file hosts.ini by executing the following command

sudo gedit /var/lib/jenkins/workspace/<Your pipeline name>/hosts.ini

Source code :

```
[local]
localhost ansible_connection=local
```

Save and close the file

28) Create a playbook mydeploy.yml by executing the following command

sudo gedit /var/lib/jenkins/workspace/<Your pipeline name>/mydeploy.yml

Source code :

```
---
- name: Deploy Artifact to Localhost
  hosts: localhost
  tasks:
    - name: Copy the artifact to the target location
      become: true
      become_user: student
      become_method: su
      copy:
          src: "/var/lib/jenkins/workspace/< Your pipeline name>/target/<Your Maven project name>-1.0-SNAPSHOT.jar"
          dest:  "/home/student/t.jar"
```

Save and close the file

29) Click on Build now

- If you get a green tick, it means Build is successful else if you get a red cross mark, it means Build has failed. You will have to type the correct code in the pipeline script, inventory file (or) playbook file.

30) Click on **"Stages"** in the Jenkins Dashboard to see the final result showing all the 5 stages in green tick.