Experiment -08.

86 practice Exercise : set up a Jenkins CI pipeline for a maven project use Ansible to Deploy Artifacts Generated by Jenkins

Aim :- to set up a Jenkins CI pipeline for a Maven project Automating the build And test processes And then use Ansible to Deploy the generated Artifacts to a target Server ensuring a Streamlined CI/CD workflow

Step1 :- Install Java
* Sudo apt update && sudo apt upgrade -y

step 2 :- Install and configure Jenkins.
install Jenkins :
Sudo apt update
Sudo apt install Jenkins -y
start And Enable Jenkins :-
Sudo system start Jenkins
Sudo systemctl enable Jenkins.

Get Jenkin Admin Password :-
Sudo cat /var/lib/Jenkins/secrets/initialAdminPassword.
Access Jenkins UI : http://localhost:8080

step3 :- install and configure Maven in Jenkins
install maven
sudo apt install maven -y

Output :-
openjdk version "11.0.x"

Output :-
[INFO] Building myapp 1.0
[INFO] packing JAR
[INFO] BUILD SUCCESS

Run the Ansible playbook to Deploy

Task [Copy Artifact from jenkins workspace] ***********
changed : [localhost]

Task [Run the Java Application] ***********
changed: [localhost]

Task [Run the Java Applicatio] ***********

play RECAP **********
localhost : OK=2 changed=2 unreachable=0 failed=0

Verify Deployment

Output :- Hello welcome to my Application

Add Maven in Jenkins
-> Open Jenkins Dashboard
-> Go to Manage Jenkins -> global tool configuration
-> Under Maven Add a new installation
-> Name it Maven-3 and set path to.

Step 4 :- Create a Jenkins CI pipeline for a Maven Project
Create a Maven Project.
-> in Jenkins Dashboard -> New item -> Maven project.
-> Name it Maven-CI
-> Under Source code Management And a Git Repository
http://github.com/sample-user/sample-maven-project.git
Under Build Enter :
clean Package.

* Run the Jenkins Build
-> click build Now
-> Jenkins fetches the project Compiles the code run tests And generates a JAR/WAR file in target/directory

Step 5 :- Install Configure Ansible.
install Ansible
Sudo apt install ansible -y
Configure Ansible inventory
nano inventory.ini
Add:
[local]
localhost ansible_connection=local

Step 6 :- Use Ansible to Deploy the Artifact Create an Ansible playbook (deploy.yml)

nano deploy.yml

Add

```
- name : Deploy Move Pacxel
  hosts : local
  become : yes
  tasks:
  - name: Copy Artifact from Jenkins Workspace
    copy:
      src: /var/lib/jenkins/workspace/maven-ci-target
           /myapp.jar
      dest : /opt/myapp.jar
      mode : '0644'
  - name : Run the Java Application
    command : nohup java -jar /opt/myapp.jar &
```

(Save and Exit : CTRL + X -> Y -> Enter)

Step 7 :- Run the Ansible Playbook to Deploy

ansible-playbook -i inventory.ini deploy.yml

Step 8 :- Verify Deployment

checks Running java process.

ps aux | grep java

Access the Application (if web-Based)

curl http://localhost:8080

AM 15/5

Experiment - 09

9) Introduction to Azure DevOps :- Overview of Azure DevOps services, Setting up An Azure DevOps Account And Project

—: Introduction to Azure DevOps :-

* what is Azure DevOps ?

-> Azure DevOps is a Suite of Development tools And Services offered by Microsoft to Support the entire Software Development lifecycle (SDLC)

-> it Enables teams to plan work collaborate on code Development build deploy Application efficiently

—: Overview of Azure DevOps Services :-

- Azure DevOps provides Several Key Services.

1) Azure boards
2) Azure Repos
3) Azure pipelines
4) Azure Test plans
5) Azure Artifacts.

—: Setting up an Azure DevOps Account And project :-

Step1 :- Sign up for Azure DevOps

* visit : https://dev.azure.com
* Sign in using a Microsoft Account (Ex:- Outlook, office 365)

* if you don't have an Account click Create one. And follow the instructions

Step 2 :- Create An Organization

1. Once logged in, click New Organization
2. choose
   * Organization name Ex :- myteam-devops
   * location (choose a geographic region)
3. click Continue

Step 3 :- Create a New project.

* Inside your Organization click New project.
* provide :- webappproject
  visibility :- private or public.
* click create.

-: Next Steps :-

After your project is created you can start using the Services.

* Use Boards to plan and track work
* set up your Repos for version Control
* Define And Configure pipelines to Automate builds And Deployments
* Add Test plans for QA
* Use Artifacts if you manage package Dependencies

Output :-

Experiment - 10

10) Creating Build pipeline ; Building a Maven/gradle project with Azure pipeline. integrating code repositories (e.g :- Github, Azure repos) Running Unit tests And Generating Reports

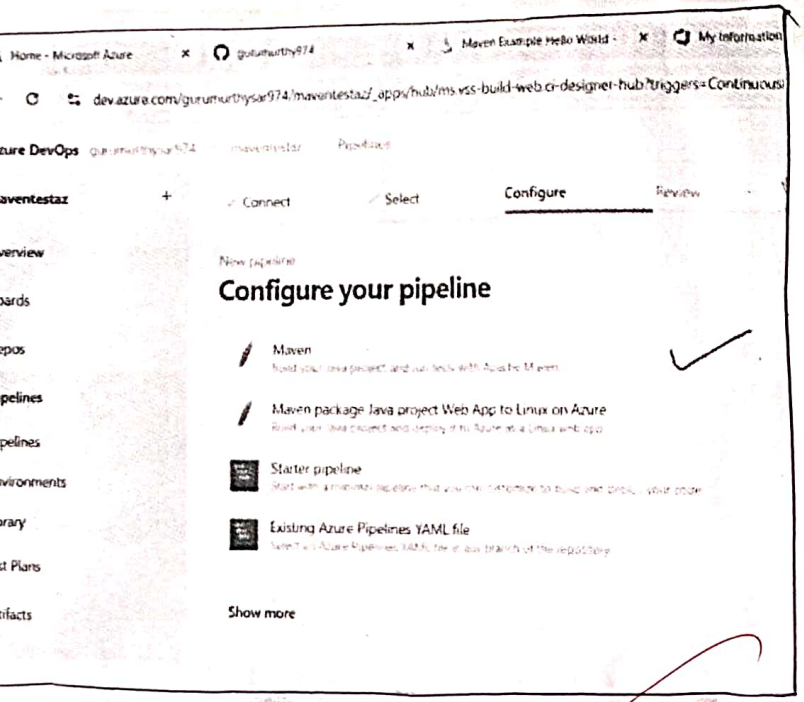
Step 1 :- on Creating Organization goto Organization settings goto policy And Allow public project Active.

Step 2 :- Goto GitBash
Type Commands As in below
cd maventest1

Step 3 :- to Create Simple hellow world maven project type command as in below
mvn archetype:generate -DgroupId=com.directorymaven -DartifactId=JavaTeam -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false.

Step 4 :- to add files from local to github follow the procedure.
a) first Create a repository in github as maventestazure.
b) then come to gitbash and type.
- git init.
git add
git commit -m "azure pipeline example"
git branch -M main.

Output :-



- git remote add origin https://github.com/xxx/maventestazure.git
git push -u origin main

→ After Completion of Above Command my Repository looks.

Step 4 :- Now go to Azure Devops Organization Create public project.

Step 5 :- Select Pipeline And then Click on Create pipeline

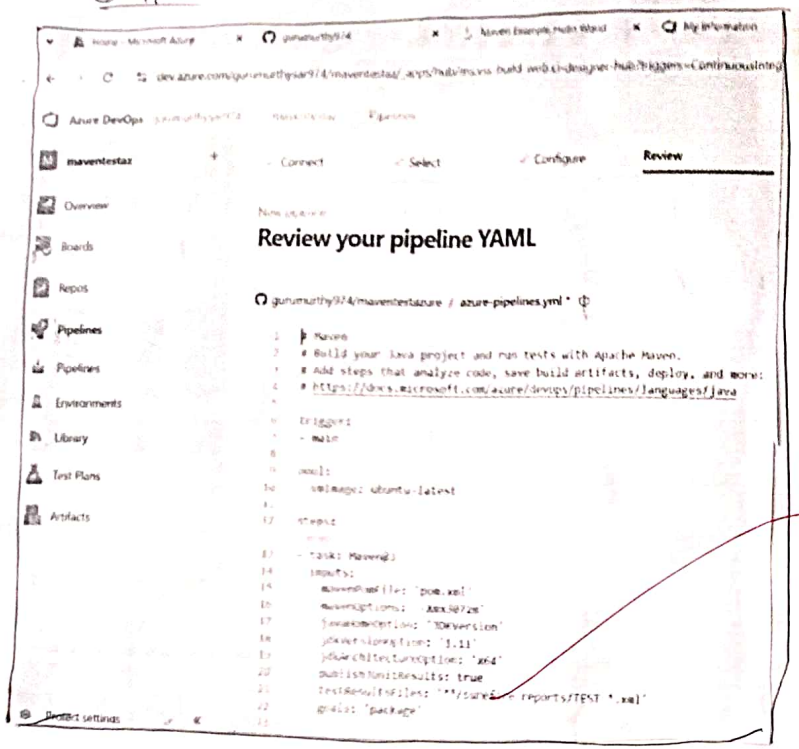
Step 6 :- After Creating pipeline Select type of Repo as github

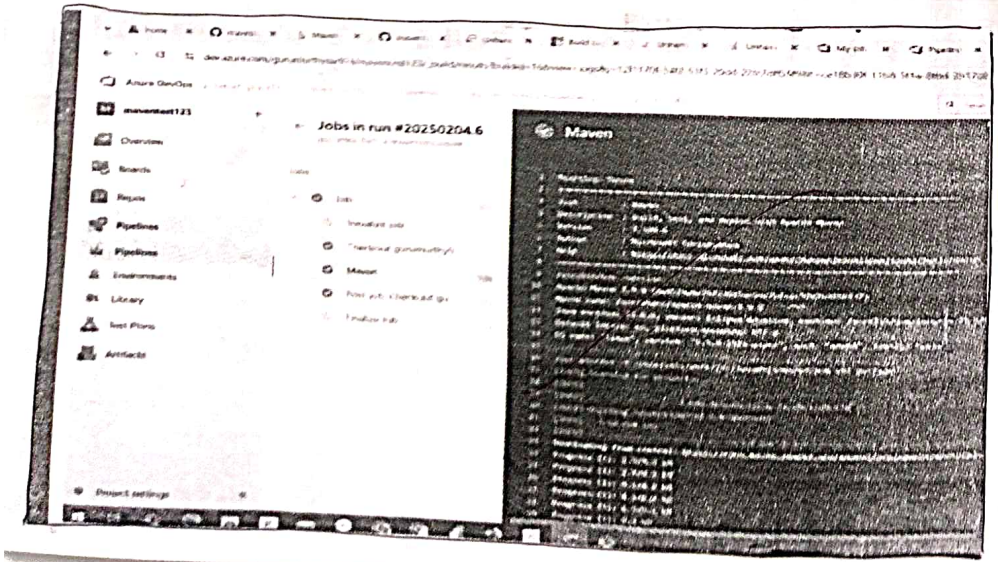
Step 7 :- it asks for Minimum signin Verification After that your Screen be as in below Select required Repository there to Run Maven project project in my Case its maventest123

Step 8 :- After Required Repo is Selected the Screen be as in below.

Drag the Screen Down check Once Again the Selected Repository is Correct or not then Click on Approve And install

Outputs :-





Step 9 :- It Again Verifies Signin Verification of Microsoft Account you be able to see Starter pipeline Select for Maven

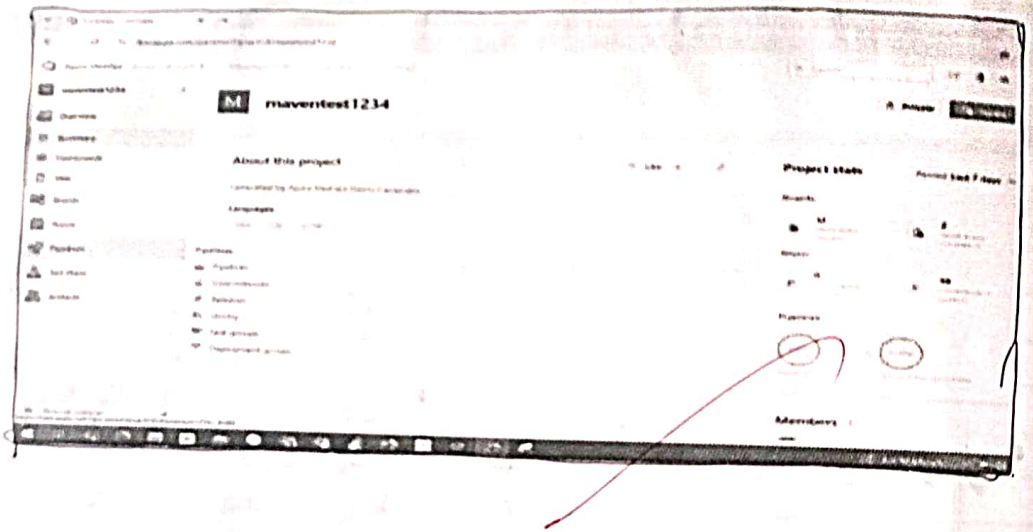
* After Selecting Maven it Asks for Save And run Just Click on it.
* finally you be able to see tasks Running if failed bec we shld Mention proper path for pom.xml

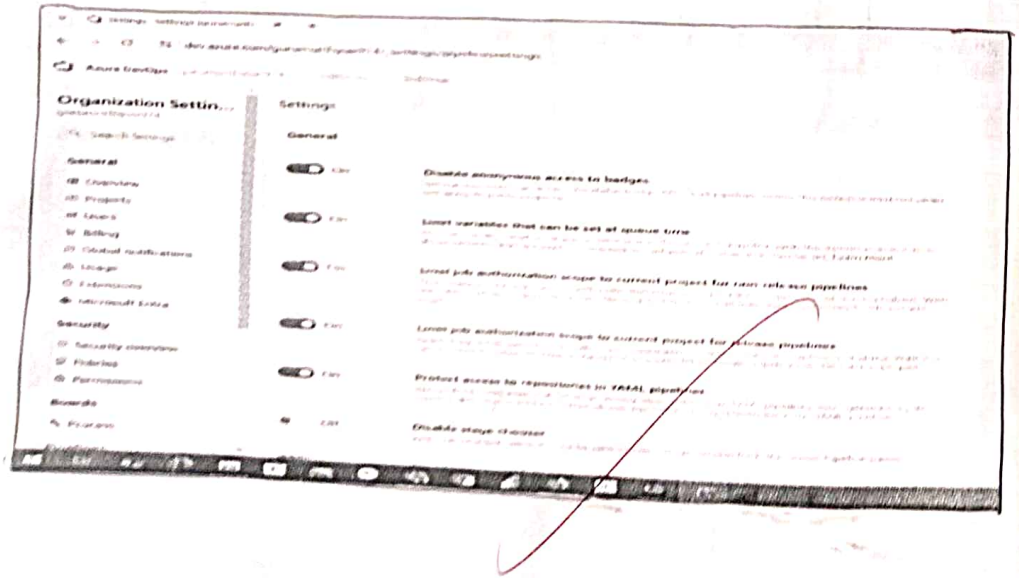
-> the Commits will Also be visible in github we can Download And Also see individual Raw log Reports.

-> if pipeline permission Comes please Complete the Registration form of parallel Jobs after 2 working day (48 hrs) you be Able to Run pipelines for private projects. Same happens to be for public projects.

2025

Outputs :-





Experiment - 11

11& Creating Release pipelines :- Deploying Applications to Azure App Services: Managing Secret And Configuration with Azure key Vault. Hands-on. Continuous Deployment with Azure pipelines.

procedure :-

Step 1 :- prerequisites

- Before starting Ensure you have
- An Azure Devops Account http://dev.azure.com
- Azure App service Created in Azure portal
- An Application Stored in github / Azure Repos.
- Azure CLI installed for Managing Resources

Step 2 :- Create a New Release pipeline

open Azure Devops.

1. Navigate to Azure Devops → pipelines → Releases
2. Click New pipeline

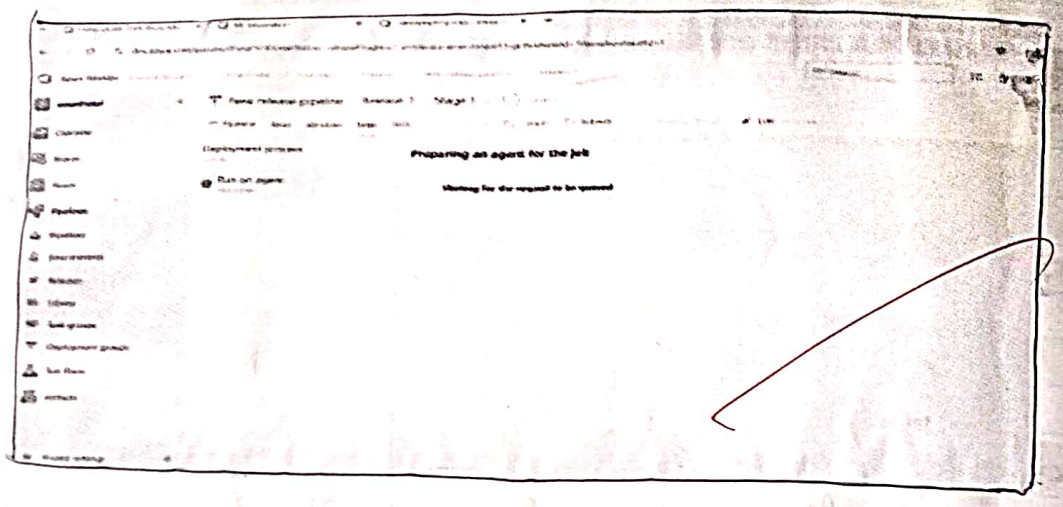
Select Deployment Template.

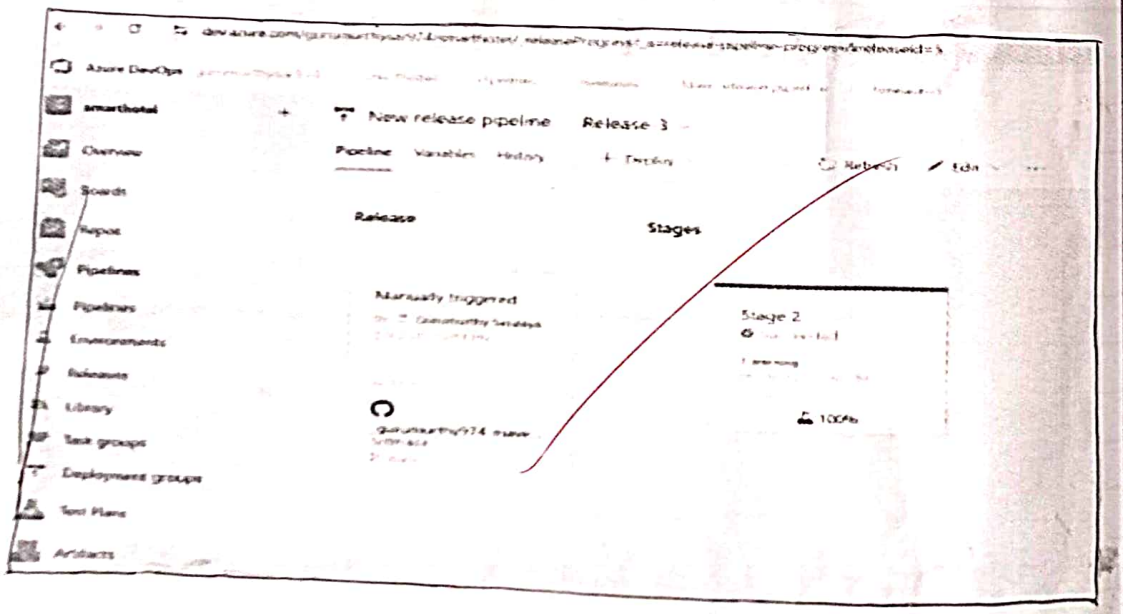
1. choose "Azure App Service Deployment"
2. Click "Apply"

Step 3 :- Configure the Release pipeline.

Step 4 :- Manage Secrets with Azure Key Vault.

Outputs :-





* Create An Azure key Vault

az keyvault create -name mykeyVault -resource-group myResourceGroup -location eastus

-Add a Secret

az keyVault secret set -vault-name mykeyVault --name "DatabasePassword" -value "SuperSecure123!"

Outputs

Step 1 :-

cloning into 'your-repo'...
remote: Enumerating objects: 100% done.
Receiving objects: 100% done.

Step 2 :-

Jenkins installed and running on http://localhost:8080

Experiment - 12

12. practide Exercise And wrap-up : Build And Deploy a Complete DevOps pipeline Discussion on Best practices And Question Answer.

Aim :- To build And Deploy a Complete DevOps pipeline by integrating the Continous integration (CI) Continuous Deployment (CD) And Configuration Management tools the exercise includes Hands-on- implementation Discussion on the best practices And question Answer Session to Reinforce DevOps principles And streamline Software delivery

procedure :-

Step 1 :- set up version Control (Github / Azure-Repos)
initialize a Git Repository
- if using github
git clone https://dev.azure.com/your-org-name/your-project/_git/your-repo
- cd your-repo

Step 2 :- Set up CI/CD pipeline (Jenkins (or) Azure pipelines)
install Jenkins on ubuntu

Sudo apt update (or) sudo apt install openjdk-11-jdk -y
wget -q -https://pkg.Jenkins.io/debian/Jenkins.io.key | sudo apt-key add - sudo sh -c echo deb

binary /etc/apt/source.list.d/Jenkins.list'
-> Sudo apt update -y
Sudo apt install Jenkins -y
Sudo systemctl strat Jenkins
Sudo systemctl enable Jenkins

Step 3 :- Configure Build in Jenkins.
Create a New Jenkins pipeline.
1. Open Jenkins UI -> New item -> pipeline
2) Configure the Git Repository URL
3) Add a Jenkinsfile in the Repo:

```
pipeline {
  agent any
   stages {
     stage ('Checkout') {
       steps {
         git 'http://github.com/your-username/your-repo.git'
       }
     }
    stage ('Build') {
      steps {
        sh 'mvn clean package'
      }
    }
     stage('Test') {
     }
}
```

Step 3 :-

pipeline runs Successfully building and testing the Application.

Step 4 :-

web App Successfully created.

Steps :

Ansible installed Successfully


```
stage('Deploy') {
    steps {
        sh 'scp target/myapp.jar user@server:/opt
        /myapp.jar'
    }
}
}
}
```

Step 4 :- Deploy to Azure App Services
Create An Azure App Service

```
az webapp create --name MyAppService --resource group
MyResourceGroup --plan MyAppServicePlan
--runtime "JAVA|11"
```

Step 5 :- Configure Deployment Task in Azure pipelines

modify azure-pipelines.yml

```
trigger:
- main
pool:
vmImage: 'ubuntu-latest'
steps:
- task: Maven@3
  inputs:
    MavenPomFile: 'pom.xml'
```

```
goals: 'clean Package'
- task: AzureWebApp@1
  inputs:
    azureSubscription: 'MyService Connection'
    appName: 'MyAppService'
    package: '$(System.DefaultWorkingDirectory)
              /target/*.jar'
```

Steps :- Configuration Management with Ansible

install Ansible

```
sudo apt update.
Sudo apt install ansible -y
```

Step 6 :- Create an Ansible playbook (deploy.yml)

```
- name: Deploy Application
  hosts: localhost.
  become: yes.
  tasks:
  - name: Copy JAR to Server.
    Copy:
    src: /var/lib/Jenkins/workspace/Maven-CI/
         target/myapp.jar
    dest: /opt/myapp.jar
  - name: Start Application.
    Command: nohup java -jar /opt/myapp.jar &
```

Step 6 :-

Ansible playbook executed Application deployed Successfully.