Program - 06

Date: 08-05-2025

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

⇒

✱ Prerequisites :→

- Jenkins installed and running (can be local or remote server) on http://localhost:8080.
- Java and Maven installed
- A sample Maven Project (Java-based)
- GitHub account (to host your project)

✱ Configure Tools in Jenkins :→

1. Go to Manage Jenkins > Global Tool Configuration.

2. Under JDK, click Add JDK:
   - Name: JDK-17
   - JAVA_HOME : C:\Program Files\Java\jdk-17

3. Under Maven, click Add Maven:
   - Name : Maven-3.9.9
   - MAVEN_HOME : C:\apache-maven-3.9.9

4. Under Git, ensure Git is installed or specify the path to your Git executable.

Date 02-05-2025
Page No. 34

⑧ Install Required Plugins :→

1. Go to Manage Jenkins > Manage Plugins.

2. Under the Available tab, search for and install the following plugins:
   - Git
   - Maven Integration
   - JUnit
   - Pipeline

⑨ Create a Sample Git Repository :→

For demonstration purposes, we'll use a simple Java Maven project hosted on GitHub.

1. Clone the repository to your local machine.

```
>> git clone https://github.com/spring-projects/spring-petclinic.git

>> cd spring-petclinic
```

2. Ensure the project builds correctly:

```
>> mvn clean install
```

Date: 02-05-2025
Page No. 34

⊛ Install Required Plugins :→

1. Go to Manage Jenkins > Manage Plugins.

2. Under the Available tab, search for and install the following plugins:
   - Git
   - Maven Integration
   - JUnit
   - Pipeline

⊛ Create a Sample Git Repository :→

For demonstration purposes, we'll use a simple Java Maven project hosted on GitHub.

1. Clone the repository to your local machine

```
>> git clone https://github.com/spring-projects/spring-petclinic.git
```

```
>> cd spring-petclinic
```

2. Ensure the project builds correctly:

```
>> mvn clean install
```

✱ Create a Jenkins Pipeline Job :-

1. In Jenkins, click New Item.
2. Enter a name for your job (e.g., SpringPetclinicPipeline).
3. Select Pipeline and click OK.
4. In the Pipeline section, set :
   - Definition : Pipeline script
   - Paste :

```
pipeline{
  agent any
   tools{
     jdk 'jdk-17'      //Make sure JDK-17 is configured in Jenkins
     maven 'Maven-3.9.9'    //Ensure Maven is installed in Jenkins global tools
   }
   environment{
     MAVEN_OPTS = "-Dmaven.test.failure.ignore=true"
   }
   stages{
    stage('Checkout Code'){
     steps{
        git url: 'https://github.com/spring-projects/spring-petclinic.git', branch: 'main'
```

Date: 08-05-2025
Page No. 36

```
        //git credentialsId: 'github-creds', url:
        'https://github.com/your-org/your-repo.git',
            branch: 'main'
        }
    }
    stage('Build with Maven'){
      steps{
        sh 'mvn clean compile'
      }
    }
    stage('Run Tests'){
      steps{
        sh 'mvn test'
      }
    }
    stage('Archive Test Results'){
      steps{
        junit 'target/surefire-reports/*.xml'
      }
    }
    stage('Package App'){
      steps{
        sh 'mvn package'
        archiveArtifacts artifacts: 'target/*.jar',
                         fingerprint: true
```

```
        //git credentialsId: 'github-creds', url:
        'https://github.com/your-org/your-repo.git',
            branch: 'main'
        }
    }
    stage('Build with Maven'){
      steps{
        sh 'mvn clean compile'
      }
    }
    stage('Run Tests'){
      steps{
        sh 'mvn test'
      }
    }
    stage('Archive Test Results'){
      steps{
        junit 'target/surefire-reports/*.xml'
      }
    }
    stage('Package App'){
      steps{
        sh 'mvn package'
        archiveArtifacts artifacts: 'target/*.jar',
                         fingerprint: true
```

```
        }
    }
}
post{
    success{
        echo "✓ Build and tests succeeded."
    }
    failure {
        echo "✗ Build or tests failed."
    }
}
}
```

click : Save

This pipeline script defines the following stages:

- Checkout : Clones the repository.
- Build : Compiles the project and skips tests.
- Test : Runs the unit tests.
- Deploy : Placeholder for deployment steps.

✱ Trigger the Pipeline :→

1. In Jenkins, navigate to your pipline job.
2. Click "Build Now" to trigger the pipeline.
3. Monitor the build progress in the Build History section.

Program - 07

Date: 12-05-2025
Page No: 37

7) Configuration Management with Ansible : Basics of Ansible : Inventory, Playbooks, and Modules, Automating Server Configurations with Playbooks, Hands-On: Writing an Running a Basic Playbook.

→

Ansible is a powerful automation tool used for IT tasks such as configuration management, application deployment, and orchestration.

It uses simple, human-readable YAML files to define automation tasks.

⊙ Inventory :- An inventory in Ansible is a file that contains information about the servers or machines you want to manage.

This file is often in INI or YAML format.

• INI-style format :

```
Ini
[web_servers]
web1.example.com
web2.example.com

[db_servers]
db1.example.com
db2.example.com
```

• YAML-style format :

```
yaml
all:
  children:
    web_servers:
      hosts:
        web1.example.com
    db_servers:
      hosts:
        db1.example.com
```

⊙ Playbooks:- A playbook is a YAML file that defines a series of tasks to be executed on one or more hosts.

Playbooks can define multiple plays, where each play is a set of tasks run on a group of hosts.

⊙ Modules:- Ansible modules are the building books of Ansible tasks. Each module performs a specific function (e.g., install packages, copy files, start servic

Example of a module in a playbook:

```yaml
- name: Install Nginx
  apt:
    name: nginx
    state: present
    update_cache: yes
```

⊙ Automating Server Configurations with Playbooks:- You can use playbooks to automate server configurations. For example, automating the installation and configuration of software packages, setting up users, managing services, and more.

Date: 12-05-2025

Page No. 41

⊙ Hands-On: Writing and Running a Basic Playbook :→

Step1 :- Install WSL (Windows Subsystem for Linux)

Open Powershell as Administrator and run this:

wsl --install

Step2 :- Open Ubuntu (WSL) → set Linux username and password then setup Ansible in it.

① >> sudo apt update ② >> sudo apt install ansible -y

③ verify Ansible → ansible --version

Step3 :- Create a Working Directory and an Inventory File.

① mkdir ~/ansible-lab

② cd ~/ansible-lab

③ nano hosts :→ Type as below:

[local]

localhost ansible_connection = local

Then save :→ Press CTRL+O → Enter → Press CTRL+X

Step4 :- Test Ansible is Working.

> ansible -i hosts local -m ping

Step 5 :- Create a Sample Playbook.

>nano install_nginx.yml and type as below:

```
---
- name : Install and start NGINX on localhost
  hosts : local
  become : yes
  tasks :
    - name : Install NGINX
      apt :
        name : nginx
        state : present
        update_cache : yes
    - name : Ensure NGINX is running
      service :
        name : nginx
        state : started
        enabled : yes
```

Save and exit (CTRL+O, Enter, then CTRL+X)

Step 6 :- Run the Playbook and Verify the Result.

◉ Run :→ ansible-playbook -i hosts install_nginx.yml

◉ Result :→ curl http://localhost

Program-08

Date: 15-05-2025
Page No. 44

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

→ ⊛ Environment Overview :→

- Jenkins is installed and accessible via web interface.
- Java JDK 17 and Maven 3.9.9 are installed.
- You have WSL (Ubuntu) installed and Ansible configured there.
- You're on a Windows machine.

Step 1: Jenkins Initial Configuration :→

1.1 Launch Jenkins

Open your browser: http://localhost:8080

1.2 Install Jenkins Plugins

Go to: Manage Jenkins → Plugins → Install

- Git Plugin
- Maven Integration
- Pipeline
- Ansible Plugin

1.3 Configure Global Tools ⊛

Go to: Manage Jenkins → Global Tool Configuration

Maven : Name : Maven-3.9.9

MAVEN_HOME : C:\apache-maven-3.9.9

JDK: Name : jdk-17

JAVA_HOME : C:\Program Files\Java\jdk-17\

Step 2: Create a Sample Maven Project :→

2.1 Use a Sample GitHub Repo

https://github.com/spring-projects/spring-petclinic

Step 3: Configure Jenkins Job :→

3.1 New Job : • Go to : New Item

• Name : maven-ci-pipeline

• Type : Freestyle Project

3.2 Source Code Management

• Git → Add your repo URL

3.3 Build Environment

• Check : "Delete workspace before build starts"

3.4 Build • Select Invoke top-level Maven targets

• Maven Version : Maven-3.9.9

• Goals : clean package

3.5 Post-Build Actions

• Archive the artifacts → Files to archive : target/*.jar

Save and Build Now.
You should now have a JAR file generated and archived by Jenkins.

Step4: Share Artifact with Ansible (via WSL)

You need to make the artifact accessible to Ansible. Easiest way is to copy it to a known shared folder.

4.1 Create a Shared Folder :- makefile C:\ci-artifacts
copy target*.jar C:\ci-artifacts\
Now Ansible in WSL can access it at:
/mnt/c/ci-artifacts/

Step5: Configure Ansible in WSL :->

5.1 Install Ansible (if not done)
>sudo apt update >sudo apt install ansible

5.2 Inventory File
Create /home/sunil/ansible/inventory :

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

Date 15-05-2025
Page No. 47

5.3 Ansible Playbook

Create /home/sunil/ansible/deployment.yml :

> mkdir -p ~/deploy

> nano /home/sunil/ansible/deploy.yml

- hosts : local

  tasks :

    - name : Copy JAR to deployment directory

      copy :

        src : /mnt/c/ci-artifacts/spring-petclinic-3.4.0-SNAPSHOT.jar

        dest : /home/sunil/deploy/app.jar

Step 6 : Run Ansible Playbook :→

>cd ~/ansible

>ansible-playbook -i inventory deploy.yml

You've now deployed the artifact from Jenkins to a target using Ansible.

⊛ Optional : Run Ansible from Jenkins

Install WSL Plugin or configure Execute shell:

> wsl ansible-playbook /home/sunil/ansible/deploy.yml