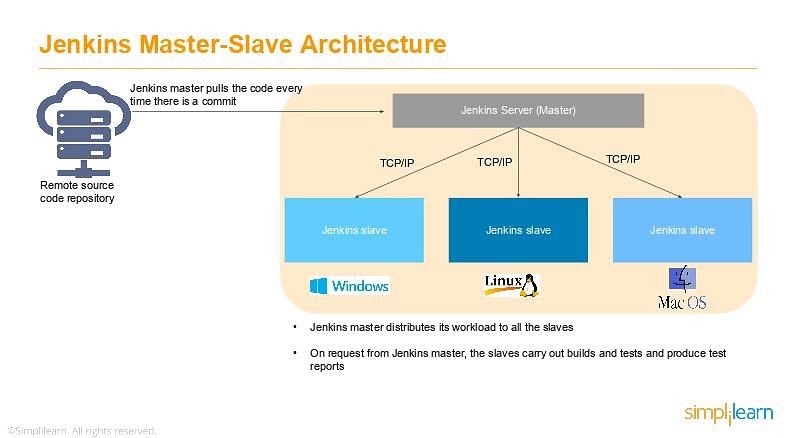
**Experiment No:5**

**Continuous Integration using Jenkins: Configure Jenkins, Jenkins Management.**

**Continuous Integration Using Jenkins**

Jenkins is an open-source automation server that helps in automating software development processes, such as Continuous Integration (CI) and Continuous Deployment (CD). It allows developers to integrate their work frequently, ensuring quick detection of errors.

****

**Installing Jenkins**

**Method 1: Using Windows Installer**

1. Download the Windows .msi installer from the Jenkins official website.
2. Run the installer and follow the setup wizard.
3. Select the installation path (default: C:\Program Files\Jenkins).
4. Choose the default port (8080), or change it if needed.
5. Once installed, Jenkins runs as a **Windows Service**.

**Configuring Jenkins**

**Initial Setup**

1. Open a browser and go to http://localhost:8080
2. Retrieve the administrator password:

cat /var/lib/jenkins/secrets/initialAdminPassword

1. Enter the password and proceed with the installation

**Installing Plugins**

Jenkins plugins extend its capabilities. Essential plugins include:

* **Git Plugin** – To integrate with Git
* **Pipeline Plugin** – For building CI/CD pipelines
* **Maven Integration Plugin** – For Java projects
* **Docker Plugin** – For containerized deployments
* **Blue Ocean Plugin** – A modern UI for Jenkins pipelines

**Jenkins Management**

**Plugin Management**

Plugins enhance Jenkins functionality.

* Navigate to: Manage Jenkins → Manage Plugins
* Install essential plugins:
  + **Pipeline** (for CI/CD automation)
  + **Git** (for version control)
  + **Build Tools** (Maven, Gradle, etc.)
  + **Credentials Binding** (for secret management

**Scheduling build Jobs using Jenkins: POLL SCM, Build Periodically.**

**What is a Pipeline in Jenkins?**

A **Jenkins Pipeline** is a way to automate the process of building, testing, and deploying applications. Instead of clicking buttons manually, you write a script (called a Jenkinsfile) that tells Jenkins what to do.

**Create a New Pipeline**

1. Click **"New Item"** (top left corner).
2. Enter a name (e.g., **MyFirstPipeline**).
3. Select **"Pipeline"** and click **OK**.

**Add Pipeline Script**

Scroll down to the **Pipeline** section.

**You have two options:**

1. **Write the script inside Jenkins**.
2. **Store the script in a file (Jenkinsfile) inside your GitHub repository**.

pipeline {

agent any

stages {

stage('Get Code') {

steps {

git url: 'https://github.com/usadhana/my-ne-repo.git', branch: 'master'

}

}

stage('Build') {

steps {

echo 'Trying to build the project...'

bat 'dir' // List files (Windows equivalent of 'ls')

}

}

stage('Test') {

steps {

echo 'Running tests...'

}

}

stage('Deploy') {

steps {

echo 'Deploying the application...'

}

}

}

}

Your pipeline follows **four main stages**:

1️**Get Code** → Clones the project from GitHub.  
2️ B**uild** → Tries to build the project (right now, it only lists files).  
3️ **Test** → Runs tests (currently just prints a message).  
4️ **Deploy** → Deploys the project (currently just prints a message).

**Method 1: Create Jobs Using Jenkins UI**

**Step 1: Create Job 1**

1. Open **Jenkins Dashboard**.
2. Click **"New Item"**.
3. Enter a name for your job, e.g., **Job1**.
4. Select **Freestyle Project** and click **OK**.
5. Scroll down to the **Build** section and click **"Add build step" → "Execute Shell"**.
6. Enter the following command:

echo "Job 1 is running"

1. Click **Save**.

**Step 2: Create Job 2**

1. Repeat steps **1-4** and name this job **Job2**.
2. In the **Build** section, add the following shell command:

echo "Job 2 is running"

1. Click **Save**.

**Step 3: Create Job 3**

1. Repeat steps **1-4** and name this job **Job3**.
2. In the **Build** section, add this shell command:

echo "Job 3 is running"

1. Click **Save**.

**Method 2: Make Jobs Run in Sequence**

Now, you can make these jobs run one after another.

1. Open **Job1** → Click **Configure**.
2. Scroll down to **Post-build Actions**.
3. Click **"Add post-build action" → "Build other projects"**.
4. Enter **Job2** and save.
5. Open **Job2**, repeat the same steps, and enter **Job3**.

Now, when you run **Job1**, it will trigger **Job2**, and **Job2** will trigger **Job3**.

**Method 3: Use a Pipeline (Recommended)**

Instead of creating jobs one by one, you can use a **Jenkinsfile**.

1. Go to Jenkins Dashboard → Click **New Item**.
2. Select **Pipeline** → Enter a name like **MyPipeline** → Click **OK**.
3. Scroll to the **Pipeline** section.
4. Paste this code:

pipeline {

agent any

stages {

stage('Job 1') {

steps {

echo 'Job 1 is running'

}

}

stage('Job 2') {

steps {

echo 'Job 2 is running'

}

}

stage('Job 3') {

steps {

echo 'Job 3 is running'

}

}

}

}

1. Click **Save** and then **Build Now**.

Now, this pipeline will execute all three jobs in order.

**Exp 6 Scheduling build Jobs using Jenkins: POLL SCM, Build Periodically.**

In Jenkins, you can schedule build jobs using **POLL SCM** and **Build Periodically** options. These are useful for automating builds based on source code changes or time-based schedules.

**Step 1: Open Jenkins and Create a New Job**

1. **Open Jenkins in your browser**:
   * Type http://your-jenkins-server:8080/ in the address bar and press **Enter**.
2. **Click on "New Item"** (on the left side).
3. **Enter a Job Name** (e.g., MyJob).
4. **Select "Freestyle Project"** (for a simple job).
5. Click **OK**.

**Step 2: Configure Source Code (Git/SVN)**

1. In the job configuration page, scroll to **Source Code Management**.
2. Select **Git** (or SVN if you are using SVN).
3. Enter your **Git Repository URL** (e.g., https://github.com/user/repo.git).
4. If your repo is private, enter your **Git credentials**.

**Step 3: Set Up Build Triggers**

Now, we will configure **POLL SCM** (to build when code changes) and **Build Periodically** (to run at a scheduled time).

**Option 1: Enable POLL SCM (Trigger Build When Code Changes)**

1. Scroll to **Build Triggers**.
2. Check the box **Poll SCM**.
3. In the **Schedule** box, enter a cron expression.

**Examples of Cron Expressions for Polling:**

| **Expression** | **Meaning** |
| --- | --- |
| H/5 \* \* \* \* | Poll every 5 minutes |
| H/15 \* \* \* \* | Poll every 15 minutes |
| H 2 \* \* \* | Poll once daily at 2 AM |

✅ **Jenkins will now check for new commits and build only if there are changes.**

**Option 2: Enable Build Periodically (Trigger Build on Schedule)**

1. Scroll to **Build Triggers**.
2. Check the box **Build Periodically**.
3. In the **Schedule** box, enter a cron expression.

**Examples of Cron Expressions for Scheduling:**

| **Expression** | **Meaning** |
| --- | --- |
| H 0 \* \* \* | Run every day at midnight |
| H 12 \* \* 1 | Run every Monday at 12 PM |
| H 4 \* \* 0 | Run every Sunday at 4 AM |

✅ **Jenkins will now run the job on the specified schedule, even if there are no code changes.**

**Step 4: Add Build Steps**

1. Scroll down to **Build** section.
2. Click **"Add Build Step"** → Select **"Execute Windows batch command.**

echo Starting build

git pull origin master

echo Build completed

Click **Save**.