**Puppet ka he**

Step 1: Update your System

sudo apt update && sudo apt upgrade –y

Step 2: Install Puppet Server Repository

wget https://apt.puppet.com/puppet7-release-focal.deb # Use focal for Ubuntu 20.04 and later.

sudo dpkg -i puppet7-release-focal.deb

sudo apt update

Step 3: Install Puppet Server sudo apt install puppetserver –y

Step 4: Configure Puppet Server Memory Allocation sudo nano /etc/default/puppetserver

JAVA\_ARGS="-Xms512m -Xmx512m" # Adjust as necessary.

Step 5: Start and Enable Puppet Server Service

sudo systemctl start puppetserver

sudo systemctl enable puppetserver

sudo systemctl status puppetserver

Step 6: Install Puppet Agent on the Same Machine sudo apt install puppet-agent –y

Step 7: Configure Puppet Server and Agent sudo nano /etc/puppetlabs/puppet/puppet.conf

[main]

certname = puppet.example.com # Replace with your hostname.

server = puppet.example.com # Replace with your hostname.

environment = production

runinterval = 1h

[agent]

server = puppet.example.com

sudo hostnamectl set-hostname puppet.example.com

**Step 8: Open Ports for Puppet Server**

sudo ufw allow 8140/tcp

sudo ufw reload

Step 9: Generate and Sign Certificates sudo /opt/puppetlabs/bin/puppet agent --test --waitforcert 60

sudo /opt/puppetlabs/bin/puppetserver ca list --all

sudo /opt/puppetlabs/bin/puppetserver ca sign --certname <agent\_hostname>

Step 10: Test Puppet Agent Communication

sudo /opt/puppetlabs/bin/puppet agent –test

Step 11: Verify Setup

sudo /opt/puppetlabs/bin/puppet resource service puppet ensure=running

**2.Git**

**Come out of mingw64**

cd ~

nano ~/.bash\_profile

vi ~/.bash\_profile

PS1='\w\$ '

 In nano: Press Ctrl + O to save, then Enter, and Ctrl + X to exit.

 In vi: Press Esc, type :wq, and press Enter.

source ~/.bash\_profile press q to come out

1. Set up Git (if not done already)

git config --global user.name "Your Name"

git config --global user.email “[your.email@example.com](mailto:your.email@example.com)”

git config –list

**Create a directory for your project**

mkdir my-git-project

cd my-git-project

**Initialize the Git repository**

git init

**Create a file**

echo "Hello, World!" > hello.txt

Add the file to the **staging area**:

git add hello.txt

Commit the change:

git commit -m "Added hello.txt"

Update the file:

echo "This is a Git practice file." >> hello.txt

View the changes:

git status

git diff

Add the changes and commit:

git add hello.txt

git commit -m "Updated hello.txt with new content"

Create a new branch:

git branch feature-branch

Switch to the new branch:

git checkout feature-branch

Verify the current branch:

git branch

Create a new file:

echo "This is a feature file." > feature.txt

Add and commit the file:

git add feature.txt

git commit -m "Added feature.txt"

Switch back to the main branch:

git checkout main

Merge the feature branch into the main branch:

git merge feature-branch

Add a remote repository (replace <remote\_url> with your repository URL):

git remote add origin <remote\_url>

Push the main branch to the remote:

git push origin main

Push the feature branch to the remote:

git push origin feature-branch

Pull the latest changes to your local repository:

git pull origin main

View the commit history:

git log --oneline –-graph

Create a tag for a specific commit:

git tag -a v1.0 -m "Version 1.0"

Push the tags to the remote:

git push origin –-tags

Make changes but don’t commit:

echo "Unfinished work" >> hello.txt

Stash the changes:

git stash

git stash list

git stash apply

**JENKINS**

1. Install and configure Jenkins and configure to run python, java and Maven project

**Download Jenkins.msi**

Select without the admin waala

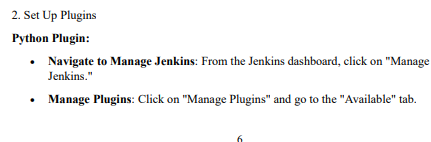
Make sure jdk hona chahiye if not download and set path in environmental variables.

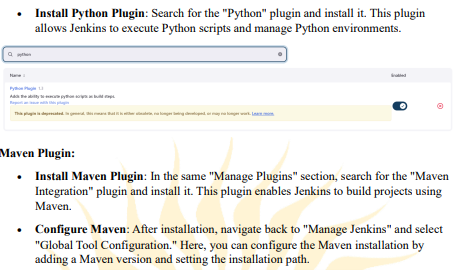
Finish all installation.

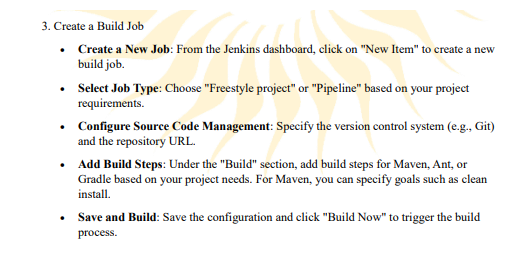
Windows + r and type services.msc do start or restart.

Open Jenkins in <https://localhost/8081> the port u selected.

Then follow the given steps given.







# hello\_world.py

def main():

print("Hello, World!")

# Simple calculation

a = 10

b = 5

result = a + b

print(f"The sum of {a} and {b} is {result}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**3.To Build the pipelines of** jobs using Maven Aim To build pipeline of jobs using Maven/Gradle/Ant in Jenkins ,Create a pipeline script to Test and deploy an application over tomcat server

 **Jenkins** installed and running.

 **Maven** installed and configured in Jenkins.

 **Tomcat** server installed and running.

pipeline {

agent any // Use any available agent

environment {

TOMCAT\_HOME = '/path/to/tomcat' // Update this path to your Tomcat installation

WAR\_FILE = 'my-app.war' // Name of your WAR file

}

stages {

stage('Build') {

steps {

echo 'Building the project...'

script {

// Clean and install the project

sh "mvn clean install"

}

}

}

stage('Test') {

steps {

echo 'Running tests...'

script {

// Run tests (if any)

sh "mvn test"

}

}

}

stage('Deploy') {

steps {

echo 'Deploying to Tomcat...'

script {

// Copy WAR file to Tomcat webapps directory

sh "cp target/${WAR\_FILE} ${TOMCAT\_HOME}/webapps/"

// Restart Tomcat to deploy the new WAR file

sh "${TOMCAT\_HOME}/bin/shutdown.sh"

sh "${TOMCAT\_HOME}/bin/startup.sh"

}

}

}

}

post {

success {

echo 'Pipeline completed successfully!'

}

failure {

echo 'Pipeline failed!'

}

}

}

Aim-To configure Jenkins Master Slave Architecture and scale your Jenkins standalone implementation by implementation slave nodes

### ****Prerequisites****

* Ensure Jenkins Master is installed and running.
* Prepare additional machines (slaves) with:
  + **Java** installed.
  + SSH server running (if using SSH method).

### ****1. Setting Up the Jenkins Master****

* Install Jenkins on the master server.
* Start Jenkins and access the web interface (http://localhost:8080).
* Complete the initial setup.

### ****2. Configuring Slave Nodes****

#### ****Method 1: Using Java Web Start****

* **Open Jenkins Dashboard**.
  + Click **Manage Jenkins**.
  + Select **Manage Nodes and Clouds**.
* **Create a New Node**:
  + Click **New Node**.
  + Enter a name (e.g., slave1), select **Permanent Agent**, and click **OK**.
* **Configure Node Settings**:
  + Fill in:
    - **Description**.
    - **# of Executors**.
    - **Remote Root Directory**.
    - **Labels** (optional).
    - **Usage**.
* **Launch Method**:
  + Select **Launch agent by connecting it to the master**.
  + Click **Save**.
* **Download Slave Agent**:
  + Click on **Launch agent** to download the jnlp file.
* **Run the Agent**:
  + Transfer the jnlp file to the slave node.
  + Run in terminal:

bash

Copy code

java -jar agent.jar -jnlpUrl http://<master-ip>:8080/computer/<slave-node-name>/slave-agent.jnlp -secret <secret>

* **Check Connection**:
  + Go back to the Jenkins dashboard to see the connected slave node.

#### ****Method 2: Using SSH (For Linux Slaves)****

* **Prepare the Slave Node**:
  + Install and run SSH server.
  + Create a user for Jenkins to connect.
* **Configure Node in Jenkins**:
  + Create a new node as in Method 1.
  + In **Launch Method**, select **Launch agent via SSH**.
  + Fill in:
    - **Host**.
    - **Credentials** (add SSH credentials).
  + Click **Save**.
* **Test Connection**:
  + Jenkins will attempt to connect; check for confirmation.

### ****3. Scaling Your Jenkins Implementation****

* **Adding More Slaves**: Repeat the above steps to add more slave nodes.
* **Load Balancing**: Jenkins distributes builds automatically across slaves.

### ****4. Running Jobs on Slave Nodes****

* **Configure Jenkins Jobs**:
  + In job configuration, select **Restrict where this project can be run** and enter the slave node label.
* **Trigger a Build**: Click **Build Now** for the job.

### ****5. Monitoring and Managing Nodes****

* **View Node Status**: Check from **Manage Nodes and Clouds** section.
* **Manage Node Resources**: Adjust executors or labels for optimization.