CONFIGURING JENKINS MASTER SLAVE SET UP

To install Jenkins on Ubuntu and set up a master-slave configuration, you can follow these steps. Jenkins is a popular open-source automation server, and this guide will help you create a Jenkins master and connect one or more Jenkins slave nodes to distribute workloads.

Step 1: Install Jenkins on the Master Node

1. Update the package list and install Java Runtime Environment (JRE):

```
sudo apt update
sudo apt install -y openjdk-11-jre
```

2. Add the Jenkins repository and its GPG key:

```
wget -q -O -
https://pkg.jenkins.io/debian/jenkins.io.key |
sudo apt-key add -
sudo sh -c 'echo deb
http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
```

3. Update the package list and install Jenkins:

```
sudo apt update
sudo apt install -y jenkins
```

4. Start the Jenkins service:

```
sudo systemctl start jenkins
```

5. Enable Jenkins to start on boot:

```
sudo systemctl enable jenkins
```

6. Check the status to ensure Jenkins is running:

```
sudo systemctl status jenkins
```

7. Retrieve the Jenkins initial admin password from the following location:

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

8. Open a web browser and navigate to `http://YOUR_SERVER_IP:8080`. Enter the initial admin password to unlock Jenkins.

9. Follow the on-screen instructions to complete the initial Jenkins setup. Create an admin user and customize Jenkins as needed.

Step 2: Configure Jenkins Master for SSH

To set up passwordless SSH communication between the Jenkins master and slaves, follow these steps on the Jenkins master node:

1. Switch to the Jenkins user:

   ```
   sudo su - jenkins
   ```

2. Generate an SSH key pair for the Jenkins user:

   ```
   ssh-keygen
   ```

   Press Enter for all prompts to use the default values.

3. Copy the public key to the slave server(s) you want to connect:

   ```
   ssh-copy-id slave_user@SLAVE_IP
   ```

Replace `slave_user` with your actual username on the slave server and `SLAVE_IP` with the IP address of the slave. Repeat this step for each slave.

4. Test the SSH connection to ensure it works without requiring a password:

   ```
   ssh slave_user@SLAVE_IP
   ```

   You should be able to log in without entering a password.

Step 3: Set Up Jenkins Slave(s)

On the Jenkins master node, configure the slave(s):

1. Navigate to the Jenkins dashboard (`http://YOUR_SERVER_IP:8080`).
2. Go to "Manage Jenkins" > "Manage Nodes and Clouds."
3. Click "New Node."
4. Enter a name for the node and select "Permanent Agent."

5. Configure the following settings:
    - **# of Executors:** The number of concurrent builds the slave can handle.
    - **Remote root directory:** The workspace directory on the slave.
    - **Labels:** Add labels to identify the slave.
6. Under "Launch method," select "Launch agent via SSH."
7. Configure the SSH settings:
    - **Host:** Slave server's IP address.
    - **Credentials:** Click "Add" and select "SSH Username with private key." Enter the Jenkins user's credentials on the slave, and choose the private key you generated earlier.
8. Click "Save."
9. Click "Launch agent" to start the slave.

Repeat the above steps for each additional slave you want to add.

Step 4: Create and Run Jenkins Jobs

Now that you have set up Jenkins master and slave(s), you can create and run Jenkins jobs. Jenkins will automatically distribute and execute jobs on the available slaves based on labels and workload.

Here's a basic example of creating a Jenkins job:

1. Go to the Jenkins dashboard.
2. Click "New Item."
3. Enter a name for the job and select "Freestyle project."
4. In the job configuration, specify the build steps, source code management, and any other relevant settings.
5. In the "Restrict where this project can be run" section, specify the label(s) of the slave(s) you want to use for this job.
6. Click "Save" to create the job.

7. Build the job, and Jenkins will assign it to an available slave based on the specified label(s).

You now have Jenkins set up with a master-slave configuration, allowing you to distribute and manage build and deployment workloads efficiently. Customize your Jenkins setup further based on your specific requirements and workflows.