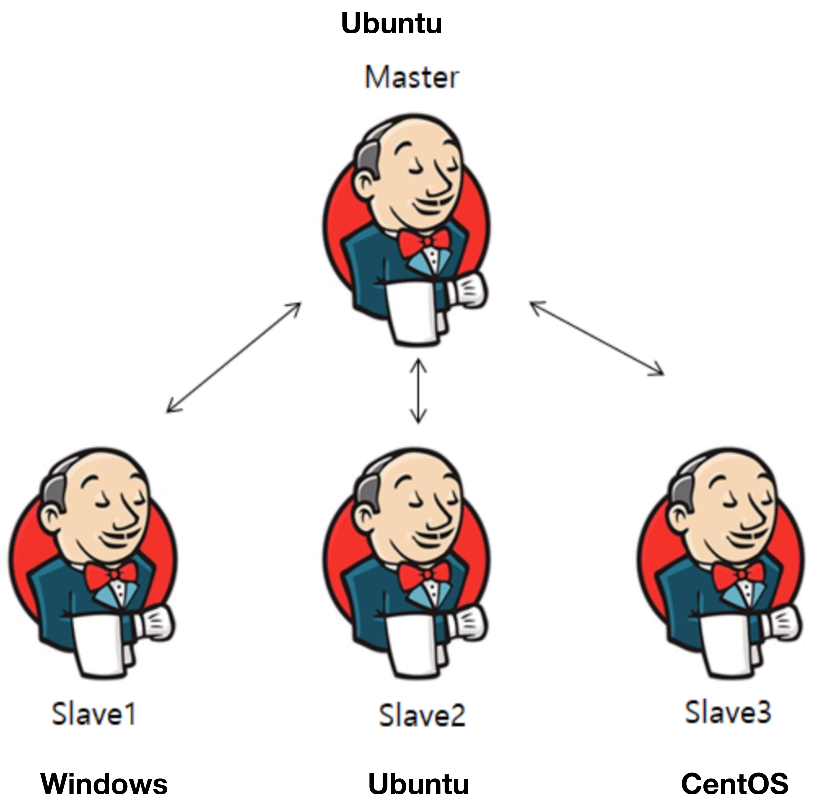
Jenkins Master-Slave Setup



==

What is Jenkins?

Jenkins is a tool that helps automate the process of building, testing, and deploying software.

Why we should use Jenkins?

1. Automates repetitive tasks.
2. Saves time and effort.
3. Reduces human errors.
4. Supports Continuous Integration (CI) and Continuous Delivery (CD).
5. Works with almost any tool and platform.

What is the need of Jenkins Master and Slave?

1. To distribute the workload across multiple machines.
2. To run multiple jobs parallel.
3. To use different systems for different types of tasks (Linux, Windows, Mac).
4. To improve speed, scalability, and performance.

What is Jenkins Master and Slave?

- **Jenkins Master** = The controller. It manages, schedules, and monitors the jobs.
- **Jenkins Slave (Agent)** = The worker. It performs the actual tasks like build, test, or deploy when instructed by the Master.

How Jenkins Master and Slave works?

```
+-------------+            +----------------+              +------------+
| Developer   |  Commit    | Jenkins        |  Assign      | Jenkins    |
| (Git Repo)  +----------->| Master         +------------->| Slave 1    |
+-------------+  New Code  | (Controller)   |  Job         | (Worker)   |
                           +----------------+              +------------+
                                   |                             |
                                   |  Assign                     |
                                   |---------------------------> |
                                   |                             |
                                   |                          +------------+
                                   |                          | Jenkins    |
                                   |                          | Slave 2    |
                                   |                          | (Worker)   |
                                   |                          +------------+
```

Explanation step-by-step:

1. **Code Commit:**
   Developer pushes code to the Git repository.

2. **Jenkins Master detects change:**
   Jenkins Master is configured to monitor the Git repository. It detects the code change automatically.

3. **Job Scheduling:**
   Master checks available Slaves and picks an idle one.

4. **Job Assignment:**
   Master sends job instructions (Build, Test, Deploy) to an available Slave.

5. **Slave Executes:**
   The Slave does the actual work — compiling code, running tests, building artifacts, deploying applications.

6. **Result Reporting:**
   The Slave reports the result (Success or Failure) back to the Master.

7. **Notification:**
   Jenkins Master displays the results on the Dashboard and can also send notifications via email, Slack, etc.

==

Steps that I follow to creates these Master-Slave setup

Jenkins Master-Slave Setup on AWS EC2 (Ubuntu, CentOS, Windows)

Overview

This guide provides step-by-step instructions to set up a **Jenkins Master-Slave architecture** using **Ubuntu, CentOS, and Windows-based EC2 instances** within the same **AWS VPC**.

Prerequisites

- AWS account with necessary permissions.
- Three **EC2 instances** in the **same VPC**:
  - **Master (Ubuntu)**: Runs Jenkins Server.
  - **Slave 1 (Ubuntu)**: Acts as a Jenkins Agent.
  - **Slave 2 (CentOS)**: Acts as a Jenkins Agent.
  - **Slave 3 (Windows)**: Acts as a Jenkins Agent.
- **Security Group Configuration**:
  - Allow **port 8080** for Jenkins Master (Inbound rule: TCP 8080).
  - Allow **port 22 (SSH)** for Ubuntu and CentOS.
  - Allow **port 3389 (RDP)** for Windows.
  - Allow **outbound internet access** for downloading dependencies.
- **Java 17 installed** on all nodes.

Step 1: Install Jenkins on Master Node (Ubuntu)

1. Update System and Install Java idk and ode version 17

```
sudo apt upgrade -y
sudo apt install -y openjdk-17-jdk
```

Verify java using below commands:

```
java -version
Java -version
```

2. Add Jenkins Repository and Install Jenkins

Use below three commands, first to add Jenkins repo, then update and install jenkins. (In case this repo not work for you, do the google and get another repo to add)

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

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

3. Start and Enable Jenkins

```
sudo systemctl enable jenkins
sudo systemctl start jenkins
```

4. Get Initial Admin Password

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

- Open **http://<Master_Public_IP>:8080** in a browser.
- Enter the **admin password** from the above command.
- Install **Suggested Plugins**.
- Create an **Admin User**.
- Set password

Step 2: Set Up Slave Nodes

1. Install Java on Each Slave Node

Ubuntu Slave

```
sudo apt upgrade -y
sudo apt install -y openjdk-17-jdk
```

CentOS Slave

[Notes:

If you are using CentOS 7 and Amazon Linux 2 then Use yum with below command

sudo yum install -y java-17-openjdk

If you are using CentOS 8, CentOS Stream 9, RHEL 8/9 then Use dnf with below command

sudo dnf install -y java-17-openjdk

]

Windows Slave

- Download **JDK 17** from the link: https://download.oracle.com/java/17/archive/jdk-17.0.12_windows-x64_bin.msi
- Install it and set **`JAVA_HOME`** in System Environment Variables:

1. Open **Control Panel** → **System** → **Advanced system settings**.
2. Click **Environment Variables**.
3. Under **System Variables**, click **New**.
4. Set **Variable name** as `JAVA_HOME` and **Variable value** as `C:\Program Files\Java\jdk-17` (or the installed path if you any different).
5. Click **OK** and close all windows.

- Verify installation by opening **Command Prompt** and running:
  `java -version`

2. Add Slave Nodes in Jenkins Master

- Go to **Manage Jenkins** → **Manage Nodes and Clouds** → **New Node**.
- Enter **Node Name** (e.g., `Slave-1-Ubuntu`, `Slave-2-CentOS`, `Slave-3-Windows`).
- Select **Permanent Agent** → Click **OK**.
- Set **Remote root directory**:
  - For **Ubuntu & CentOS:** `/home/jenkins`
  - **For Windows:** `C:\Jenkins`
- Set **Launch method**: `Launch agent by connecting it to the master.`
- Click **Save**.

3. Connect Slaves to Master

Ubuntu & CentOS Slaves

Here, please copy the command from your nodes, do not use my below command as it is.
Go to your Jenkins master—>Manage Jenkins—>Nodes—->Click on Centos or Ubuntu or Windows and then copy the command [FIRST COMMAND] according to windows, ubuntu and centos.

```
wget http://<Master_Private_IP>:8080/jnlpJars/agent.jar
java -jar agent.jar -url http://<Master_Private_IP>:8080/
-secret <SECRET_KEY> -name "Slave-1-Ubuntu" -webSocket
-workDir "/home/jenkins"
```

Windows Slave

- Open **PowerShell** as Administrator and navigate to the Jenkins directory:
  `cd C:\Jenkins`
- Download agent.jar:
  ```
  Invoke-WebRequest -Uri http://<Master_Private_IP>:8080/jnlpJars/agent.jar
  -OutFile agent.jar
  ```
- Start the agent:
  ```
  java -jar agent.jar -url http://<Master_Private_IP>:8080/ -secret
  <SECRET_KEY> -name "Slave-3-Windows" -webSocket -workDir "C:\Jenkins"
  ```
- The `<SECRET_KEY>` can be found in **Manage Nodes** → **Slave-Name** → **Secret**.

Step 3: Test Job Execution on Slaves

1. Create a Test Job

- Go to **Jenkins Dashboard** → **New Item**.
- Enter a name (e.g., `Test-Agent-Slave`).
- Select **Freestyle Project** → Click **OK**.
- Check **Restrict where this project can be run**.
- In **Label Expression**, enter `Slave-1-Ubuntu` or `Slave-2-CentOS` or `Slave-3-Windows`.
- Scroll to **Build** → **Add build step** → **Execute Shell (Linux)** or **Execute Windows batch command**.
- Enter the following commands:

Linux Slaves (Ubuntu & CentOS)

```
echo "Hello from Slave!"
hostname
```

Windows Slave

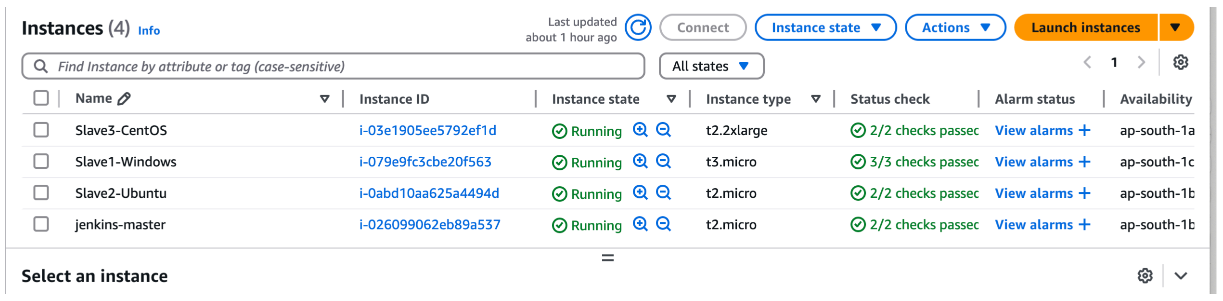
```
echo "Hello from Windows Slave!"
hostname
```

- Click **Save** and **Build Now**.

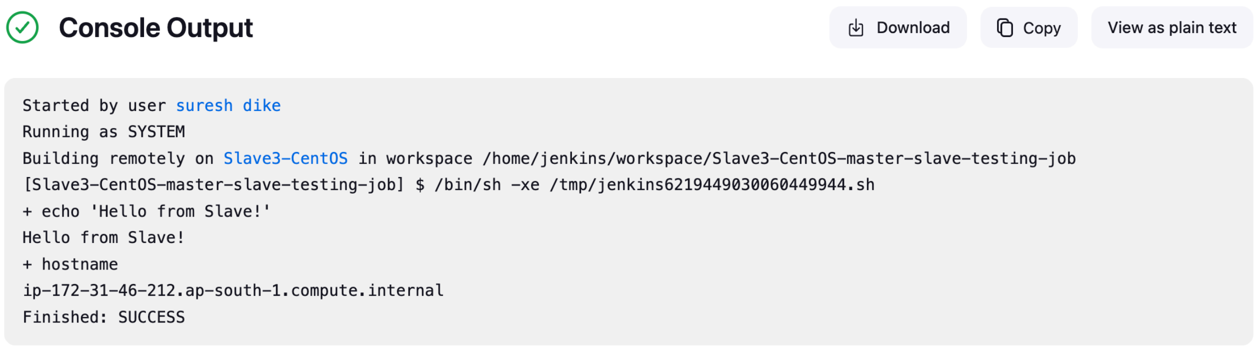
2. Check Console Output

- Click **Build History** → **Latest Build** → **Console Output**.
- You should see output like:
  `Hello from Slave!`
- `ip-172-31-2-59   # or actual hostname of your agent`

First, I setup all Nodes, one "Master" and three "Slave" on AWS cloud

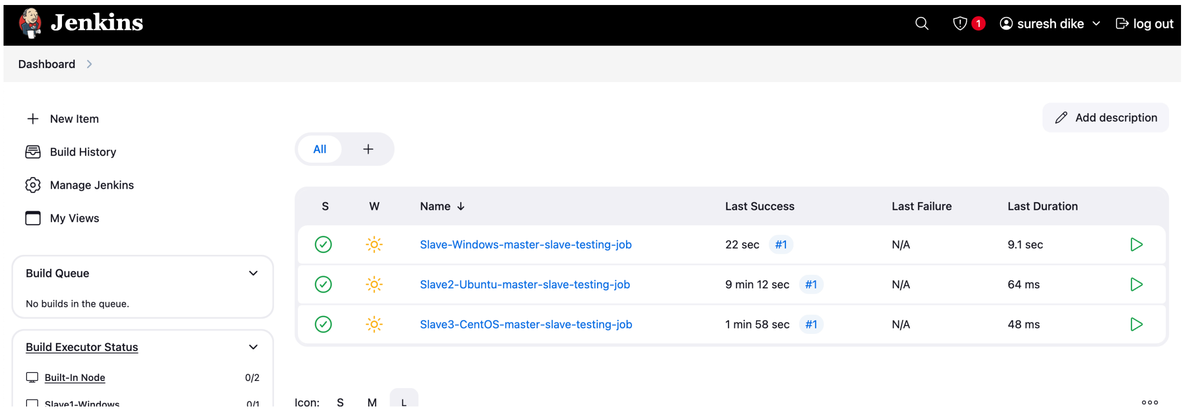


Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
Slave3-CentOS	i-03e1905ee5792ef1d	✓ Running	t2.2xlarge	✓ 2/2 checks passec	View alarms +	ap-south-1a
Slave1-Windows	i-079e9fc3cbe20f563	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	ap-south-1c
Slave2-Ubuntu	i-0abd10aa625a4494d	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	ap-south-1b
jenkins-master	i-026099062eb89a537	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	ap-south-1b



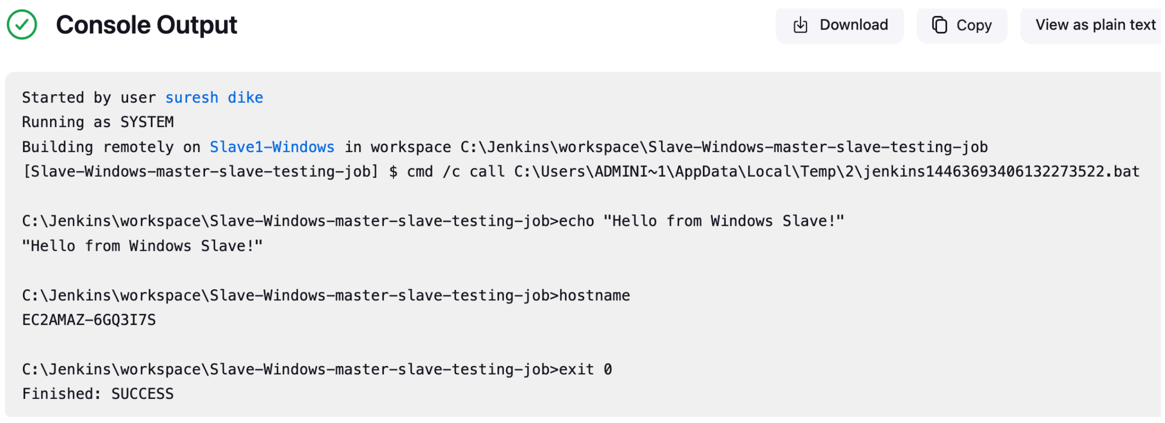
```
Started by user suresh dike
Running as SYSTEM
Building remotely on Slave3-CentOS in workspace /home/jenkins/workspace/Slave3-CentOS-master-slave-testing-job
[Slave3-CentOS-master-slave-testing-job] $ /bin/sh -xe /tmp/jenkins6219449030060449944.sh
+ echo 'Hello from Slave!'
Hello from Slave!
+ hostname
ip-172-31-46-212.ap-south-1.compute.internal
Finished: SUCCESS
```

My Testing: (After all Master Slave Configuration)



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	Slave-Windows-master-slave-testing-job	22 sec #1	N/A	9.1 sec
✓	☀	Slave2-Ubuntu-master-slave-testing-job	9 min 12 sec #1	N/A	64 ms
✓	☀	Slave3-CentOS-master-slave-testing-job	1 min 58 sec #1	N/A	48 ms

My all three nodes connected to Master and run the build successfully via each nods



```
Started by user suresh dike
Running as SYSTEM
Building remotely on Slave1-Windows in workspace C:\Jenkins\workspace\Slave-Windows-master-slave-testing-job
[Slave-Windows-master-slave-testing-job] $ cmd /c call C:\Users\ADMINI~1\AppData\Local\Temp\2\jenkins14463693406132273522.bat

C:\Jenkins\workspace\Slave-Windows-master-slave-testing-job>echo "Hello from Windows Slave!"
"Hello from Windows Slave!"

C:\Jenkins\workspace\Slave-Windows-master-slave-testing-job>hostname
EC2AMAZ-6GQ3I7S

C:\Jenkins\workspace\Slave-Windows-master-slave-testing-job>exit 0
Finished: SUCCESS
```

==

Learnings & Things to Remember

1.You may face the problem to install java versions 17 specifically.

Solutions: commands are slightly different as per the linux flavours that you are using, use specific commands [as given above]

2.You may face the problem when Windows slave machine not connecting to Jenkins master.

Solutions: it's generally happen due to environment variables are not set correctly.

Make sure Java is installed and environment variables are set correctly.

1. **Download JDK 17** properly
2. **Install JDK** → Keep the default installation path (e.g., `C:\Program Files\Java\jdk-17`)
3. **Set Environment Variables Properly:**

   Steps to set it:

   Right-click on **This PC** → Click **Properties**

   Click **Advanced system settings**

   Click **Environment Variables**

To Add JAVA_HOME Variable

Under **System Variables**, click **New**

Variable Name: `JAVA_HOME`

Paste Variable Value like: `C:\Program Files\Java\jdk-17`

Click **OK**

Update `Path` Variable

In **System Variables**, scroll down and select **Path**

Click **Edit** → Click **New**

Add the following entry:

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

Click **OK**

4. Verify Installation with below commands run. From cmd:

```
java -version
javac -version
```

Now, Download & Start Jenkins Agent on Windows Again.

1. Open **Command Prompt** as **Administrator**
2. Create a working directory for Jenkins agent:

   ```
   mkdir C:\Jenkins
   ```
3. `cd C:\Jenkins`
4. **Download the agent.jar from Jenkins Master:**

   ```
   curl -sO http://<JENKINS_MASTER_IP>:8080/jnlpJars/
   agent.jar
   ```

5. **Start the Jenkins Agent** (Here copy & use the key from your Jenkins Manage—Nodes—>):
   eg.
   ```
   java -jar C:\Jenkins\agent.jar -url http://
   <JENKINS_MASTER_IP>:8080/ -secret <SECRET_KEY> -name
   "Slave1-Windows" -webSocket -workDir "C:\Jenkins".
   ```
   Remember

```
✅ Ensure there is NO SPACE in workDir path
❌ Incorrect: -workDir "Windows: C:\Jenkins"
✅ Correct: -workDir "C:\Jenkins"
```

3. Your build still not successful?

Solutions: check the below points

1.Open your Jenkins job where build has been failed,
click on the configure from left side list, should have check or enable
"Restrict where this project can be run"

2.Check if in the **"Label Expression"** you have given the correct slave machine name?

3.Check in the "**Build Steps—>Add build step"** in drop down have you selected option according to the windows and linux?

Execute Windows batch command	[this is for windows]
Execute shell	[this is for linux]

4. Have you used **proper** build commands according to Windows or Linux in the box which you choose?
E.g.

If I choose, **Execute shell**, I will use this,

```
echo "Hello from Slave!"
hostname
```

If I choose, **Execute Windows batch command**, I will use this,

```
echo "Hello from Windows Slave!"
hostname
```

Hope your Jenkins Master-Slave configuration completed Successfully. 😊😊😊👍🏻👍🏻

Thanks
Suresh.