Jenkins Master-Slave Node Configuration Step by Step

In Jenkins, We can build all our applications on a single server. As the requirement grows and changes, we will come cross many issues like what if there are thousands build which needs to be done, what if different builds need to be done on different flavor of OS. In such cases the master-slave architecture comes in picture which takes care of all such odd requirements.

With the growing popularity of Microservices, it is becoming necessary for any company to develop and launch several services at the same time. It's at this point that the CI tools must provide a way for sharing the load across several machines/servers.

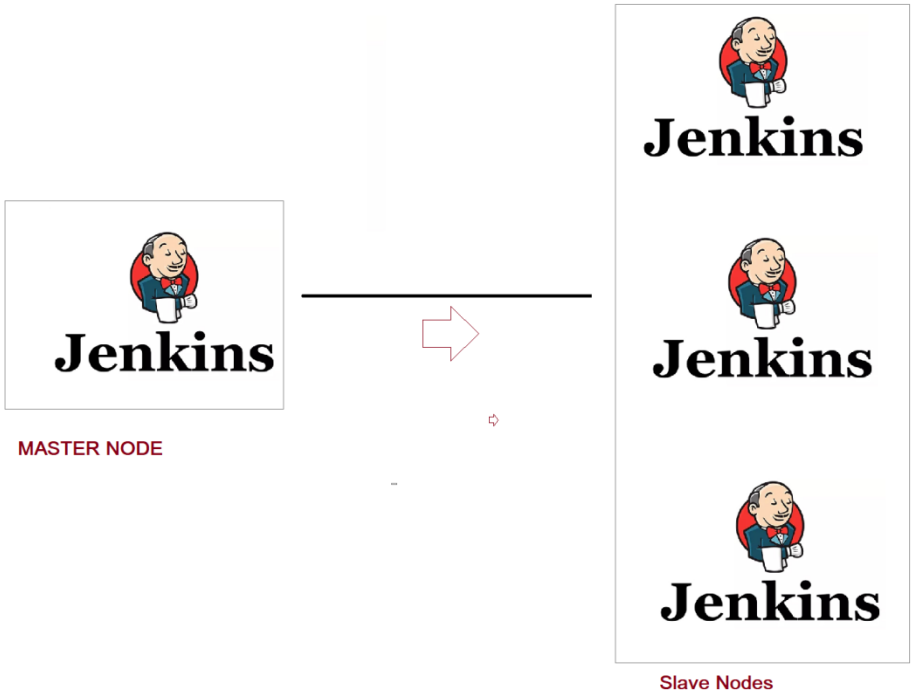
Jenkins also has the capability of distributing build jobs across a large number of systems, which is known as Jenkins distributed builds. We can set up a farm of build servers using Jenkins to distribute the burden or perform the build jobs in different settings.

When we have a large project to build and the load has to be distributed across several machines on the network, distributed builds improve the efficiency of the build process. Jenkins implements the Master/Slave architecture to manage distributed builds

For example there might be an application which needs to be build on Ubuntu. There is another application that needs to be build on CentOS. As a solution we can have two slave nodes(one with Ubuntu installed and another with CentOS installed) and build the respective applications on these servers.

Slave is a java executable which receives the build instructions from the central master node.

We may associate a project with the slave node and then schedule it for the build when we've created and configured it.



Jenkins uses A Master-Slave architecture to manage distributed builds. The machine where we install Jenkins software will be Jenkins master and that run's on port 8080 by default. On the slave machine, we install a program called Agent. This agent requires JVM. This agent executes the tasks provided by Jenkins master. We can launch n numbers of agents and we can configure which task will be run on which agent server from Jenkins master by assigning the agent to the task.

Pre-requisities to configure the Master-Slave Nodes

- ✓ Required Master and Slave Nodes(VMs/AWS EC2)
- ✓ install Jenkins and Java in Jenkins(Master) Node
- ✓ Same version of java need to install on Slave Node
- ✓ Configure Slave Node under manage nodes of Jenkins Master
- ✓ Download agent.jar onto slave node

Configure the Master Node (Master Node)

- It distributes the builds among the numerous slaves for execution.
- It organizes the build projects.
- It keeps an eye on the slaves at all times.
- Master can also run build jobs directly if necessary.
- It keeps track of the build outcomes and shows them.

1. Launch the EC2 node and connect through putty or mobxtream
2. **[root@ip-10-7-1-135 ec2-user]#**

   sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

   sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

   sudo amazon-linux-extras install epel

3. Install java

   [root@ip-10-7-1-135 ec2-user]# amazon-linux-extras install java-openjdk11

4. Install Jenkins

   [root@ip-10-7-1-135 ec2-user]# yum install Jenkins

   After installation of Jenkins done, check the status of Jenkins server

   [root@ip-10-7-1-135 ec2-user]# Service Jenkins status

   (**Note**:if it is not starting , start the Jenkins server and then check the status)

   [root@ip-10-7-1-135 ec2-user]# service Jenkins start

   [root@ip-10-7-1-135 ec2-user]# Service Jenkins status

5. Now, whenever system is restarted, ensure the Jenkins server should start to boot

   [root@ip-10-7-1-135 ec2-user]# chkconfig jenkins on

6. Accessing Jenkins

   As we know that default jenkins runs at port 8080, You can access jenkins at http://<Public IPv4 address>:8080

7. install the plugins and login into jenkins master

Slave Node Configuration

Jenkins Slave runs on a remote machine. A slave is responsible for the following tasks:

- Slaves can be operated on a number of different operating systems.
- It responds to the Jenkins Master's demands.
- Apart from the fact that Jenkins executes the build task on the next available save,
- we can always arrange the project to run on a certain sort of slave computer.
- It also completes construction operations that the Master has dispatched.

1. Launch the Another EC2 instance
2. Install java on Slave node

-ensure same java version in master node to be install in Slave node as well

[root@ip-10-7-1-135 ec2-user]# java -version

openjdk version "11.0.16" 2022-07-19 LTS

(Login into the Slave Node)

need to install the amazon extra packages, before that need to check the list of package in amazon

[root@ip-10-7-2-182 ec2-user]# amazon-linux-extras list

install java

[root@ip-10-7-2-182 ec2-user]# amazon-linux-extras install java-openjdk11

after successful installation, now check the java version

[root@ip-10-7-2-182 ec2-user]# java –version

Configure the Jenkins Master slave configuration

(Get back tot he Master machine)

Goto Manage Nodes

Manage Jenkins --> Manage Nodes and Clouds --> New Node

Add the node name as Permanent Agent

Provide below information to add Jenkins agent

Name: uniquely identifies an agent within this Jenkins installation

Description:

Number of executors: 2

Remote root directory: /opt/build

(this is the directory path in slave machine, once Jenkins job builds in Master node, it creates the build directory in this location)

Labels: javaapp-build-node

Usage:

Use this node as much as possible(select)

Launch method:

Launch agent by connecting it ot the controller(select)

Custom WorkDir path:

(custom Remoting work directory will be used instead of the Agent Root Directory)

Internal data directory:remoting

- Use WebSocket [x]

(**Note**: the red X mark represents that Slave node is not activated. that means it not yet communicated with your master node)

Availability:

Keep this agent online as much as possible

(select)

(**save** this configuration)

(Master Node)

1. Click on the SlaveNode after the above steps

   Run from agent command line:

   curl -sO http://52.91.38.89:8080/jnlpJars/agent.jar

   java -jar agent.jar -jnlpUrl http://52.91.38.89:8080/computer/java%2Dnode/jenkins-agent.jnlp -secret 5d204273e21e1869c4879ab02ae43f1bdb34aa8eb62d9f67661d1a3de3cb54ef -workDir "/opt/build"

   - Download the agent. war file

(Slave Node)

- copy the agent.jar file into the slave node into the path (/home/ec2-user/)

[root@ip-10-7-2-182 ec2-user]# pwd

/home/ec2-user

(drag and drop agent.jar file into this path in mobxtream app)

move the agent.jar file into the /opt directory

[root@ip-10-7-2-182 ec2-user]# mv agent.jar /opt

[root@ip-10-7-2-182 ec2-user]# cd /opt

[root@ip-10-7-2-182 opt]# ls

agent.jar aws rh

(In Master Node url is generate at the time of slave node creation)

java -jar agent.jar -jnlpUrl http://52.91.38.89:8080/computer/java%2Dnode/jenkins-agent.jnlp -secret 5d204273e21e1869c4879ab02ae43f1bdb34aa8eb62d9f67661d1a3de3cb54ef -workDir "/opt/build"

(copy this url and paste in the Slavenode /opt location)

[root@ip-10-7-2-182 opt]# java -jar agent.jar -jnlpUrl http://52.91.38.89:8080/computer/java%2Dnode/jenkins-agent.jnlp -secret 5d204273e21e1869c4879ab02ae43f1bdb34aa8eb62d9f67661d1a3de3cb54ef -workDir "/opt/build"

Note:Once its connected.

(Error Scenario:
Sometimes, when AWS EC2 instance is restarted its <publicIPv4> might get changed, then the following url doesn't work

java -jar agent.jar -jnlpUrl http://52.91.38.89:8080/computer/java%2Dnode/jenkins-agent.jnlp -secret 5d204273e21e1869c4879ab02ae43f1bdb34aa8eb62d9f67661d1a3de3cb54ef -workDir "/opt/build"

in that case, need make the following changes to get your 'Slave Node' works

(Master Node)
Dashboard=>Manage Jenkins=> Configure Systems
jenkins Location

http://54.166.235.28:8080/)

(Get back to the Master Node machine)

Check the java-node(slave node name) (In Jenkins Master)

Agent java-node (this node helps you to build java applications)

Agent is connected.

(Master Machine)

Build sample Job (free style job)

1.(build steps)

uptime

echo $WORKSPACE

2. now you can specify where your job should get run i.e in your slave node machine

Restrict where this project can be run(Select)

from the drop down list

javaapp-build-node

once your job in master node machine builds successfully