

## **Aim:**

To understand Jenkins Master-Slave Architecture and scale your Jenkins standalone implementation by implementing slave nodes.

## **Introduction:**

Jenkins offers a simple way to set up a continuous integration or continuous delivery (CI/CD) environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks. While Jenkins doesn't eliminate the need to create scripts for individual steps, it does give you a faster and more robust way to integrate your entire chain of build, test, and deployment tools than you can easily build yourself.

“Don't break the nightly build!” is a cardinal rule in software development shops that post a freshly built daily product version every morning for their testers. Before Jenkins, the best a developer could do to avoid breaking the nightly build was to build and test carefully and successfully on a local machine before committing the code. But that meant testing one's changes in isolation, without everyone else's daily commits. There was no firm guarantee that the nightly build would survive one's commitment.

## **Jenkins automation**

Today Jenkins is the leading open-source automation server with some 1,600 plug-ins to support the automation of all kinds of development tasks. The problem Kawaguchi was originally trying to solve, continuous integration and continuous delivery of Java code (i.e. building projects, running tests, doing static code analysis, and deploying) is only one of many processes that people automate with Jenkins. Those 1,600 plug-ins span five areas: platforms, UI, administration, source code management, and, most frequently, build management.

## **How Jenkins works**

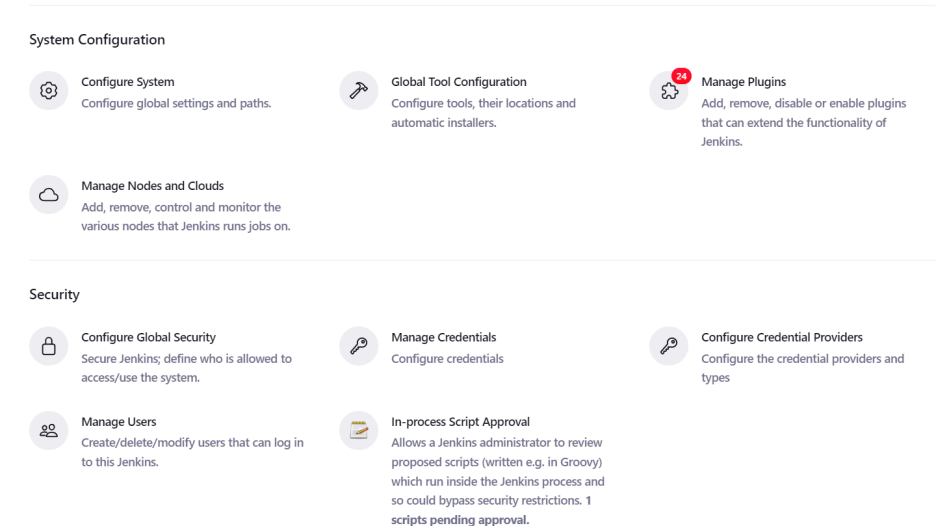
Jenkins is distributed as a WAR archive and as installer packages for the major operating systems, as a Homebrew package, as a Docker image, and as source code. The source code is mostly Java, with a few Groovy, Ruby, and Antlr files.

You can run the Jenkins WAR standalone or as a servlet in a Java application server such as Tomcat. In either case, it produces a web user interface and accepts calls to its REST API.

When you run Jenkins for the first time, it creates an administrative user with a long random password, which you can paste into its initial webpage to unlock the installation.

## Steps:

1. Go to the Manage Jenkins section and scroll down to the section of Manage Nodes.



2. Click on New Node.

Manage nodes and clouds Refresh status

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-in Node	Windows 11 (amd64)	In sync	125.16 GB	9.92 GB	125.16 GB	0ms
	Data obtained	2 min 19 sec	2 min 19 sec	2 min 19 sec	2 min 19 sec	2 min 19 sec	2 min 19 sec

3. Give a name for the node, choose the Permanent Agent option, and click on Ok.

**New node**

Node name

Type

☒ Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

4. Enter the details of the node slave machine.

1. Here no. of executors is nothing but no. of jobs that this slave can run parallelly. Here we have kept it to 2.
2. The Labels for which the name is entered as "Slave1" is what can be used to configure

jobs to use this slave machine.

3. Select Usage to Use this node as much as possible.

4. For the launch method we select the option of “Launch agent by connecting it to the master”.

5. Here in the Agents section click on Random and Save it. Now you will find the required option.

6. Enter the Custom WorkDir path as the workspace of your slave node.

7. In Availability select “Keep this agent online as much as possible”.

8. Click on Save.

Search (CTRL+K) admin log out

Press F11 to exit full screen

Name ?  
node1

Description ?

Number of executors ?  
2

Remote root directory ?  
C:\Node1

Labels ?  
node1

Usage ?  
Use this node as much as possible

Launch method ?  
Launch agent by connecting it to the controller

☐ Disable WorkDir ?

Launch method ?

Launch agent by connecting it to the controller

☐ Disable WorkDir ?

Custom WorkDir path ?

C:\Node1

Internal data directory ?

remoting

☐ Fail if workspace is missing ?

☐ Use WebSocket ?

❗ Either WebSocket mode is selected, or the TCP port for inbound agents must be enabled

Advanced...

Availability ?

Keep this agent online as much as possible

Node Properties

☐ Disable deferred wipeout on this node ?

☐ Environment variables

☐ Tool locations

- Once you complete the above steps, the new node machine will initially be in an offline state but will come online if all the settings in the previous screen were entered correctly. One can at any time make the node slave machine offline if required. If the agent does not come online, click on the Agent's name. And use the command given there to start the agent.

## Manage nodes and clouds

Refresh status

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Windows 11 (amd64)	In sync	125.16 GB	9.83 GB	125.16 GB	0ms
	node1		N/A	N/A	N/A	N/A	N/A
Data obtained		1 min 45 sec	1 min 45 sec	1 min 45 sec	1 min 45 sec	1 min 45 sec	1 min 45 sec

### Agent node1

Man

Run from agent command line:

```
curl -s0 http://localhost:8080/jnlpjars/agent.jar
java -jar agent.jar -jnlpUrl http://localhost:8080/computer/node1/jenkins-agent.jnlp -secret 7aaead1d532eddb164de280f5b41c7ab866a523b57c73f37a03e74c39f4d6681 -workDir "C:\Node1"
```





Or run from agent command line, with the secret stored in a file:

```
echo 7aaead1d532eddb164de280f5b41c7ab866a523b57c73f37a03e74c39f4d6681 > secret-file
curl -s0 http://localhost:8080/jnlpjars/agent.jar
java -jar agent.jar -jnlpUrl http://localhost:8080/computer/node1/jenkins-agent.jnlp -secret @secret-file -workDir "C:\Node1"
```

```
Windows PowerShell
PS C:\Node1> java -jar agent.jar -jnlprUrl http://localhost:8080/computer/node1/jenkins-agent.jnlp -secret 7a
b164de280f5b41c7ab866a523b57c73f37a83e74c39f4d6681 -workDir "C:\Node1"
Oct 27, 2022 7:36:12 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\Node1\remoting as a remoting work directory
Oct 27, 2022 7:36:12 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to C:\Node1\remoting
Oct 27, 2022 7:36:12 PM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: node1
Oct 27, 2022 7:36:12 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3844.vb_940a_a_e4f72e
Oct 27, 2022 7:36:12 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\Node1\remoting as a remoting work directory
Oct 27, 2022 7:36:12 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Locating server among [http://localhost:8080/]
Oct 27, 2022 7:36:12 PM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Oct 27, 2022 7:36:12 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Agent discovery successful
Agent address: localhost
Agent port: 5868
Identity: 6a:bd:a2:d5:1a:82:2a:41:f1:09:d8:af:7a:40:53:66
Oct 27, 2022 7:36:12 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Handshaking
Oct 27, 2022 7:36:12 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connecting to localhost:5868
Oct 27, 2022 7:36:12 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Trying protocol: JNLP4-connect
Oct 27, 2022 7:36:12 PM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Oct 27, 2022 7:36:12 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Remote identity confirmed: 6a:bd:a2:d5:1a:82:2a:41:f1:09:d8:af:7a:40:53:66
Oct 27, 2022 7:36:12 PM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

## Manage nodes and clouds

Refresh status

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Windows 11 (amd64)	In sync	125.16 GB	9.83 GB	125.16 GB	0ms 
	node1		N/A	N/A	N/A	N/A	N/A 
Data obtained		2 min 40 sec	2 min 40 sec	2 min 40 sec	2 min 40 sec	2 min 40 sec	2 min 40 sec

6. Now since your slave agent is up and running, let's execute a job on it.

1. For that I already have an existing job and I will run this job on this slave. Open this job and click on configure.
2. Nowhere in the General section, click on “Restrict where this project can be run”.
3. Here in Label Expression, enter the name of the slave and save it.
4. Click Save.

☐ Discard old builds ?

☐ GitHub project

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

☒ Restrict where this project can be run ?

Label Expression ?

node1

Advanced...

7. Now click on Build now and see the output of this job. If everything is correct you will see the output as Success.

### Console Output

```
Started by user admin
Running as SYSTEM
Building remotely on node1 in workspace C:\Model\workspace\masterSlave
[masterSlave] $ sh -xe C:\Users\Wakil\AppData\Local\Temp\jenkins4521270518021198897.sh
+ echo 'Build Successful!'
Build Successful!
Finished: SUCCESS
```

## **Conclusion:**

Thus, we understood Jenkins Master-Slave Architecture and scaled Jenkins standalone implementation by implementing slave nodes.