

EXPERIMENT: 6

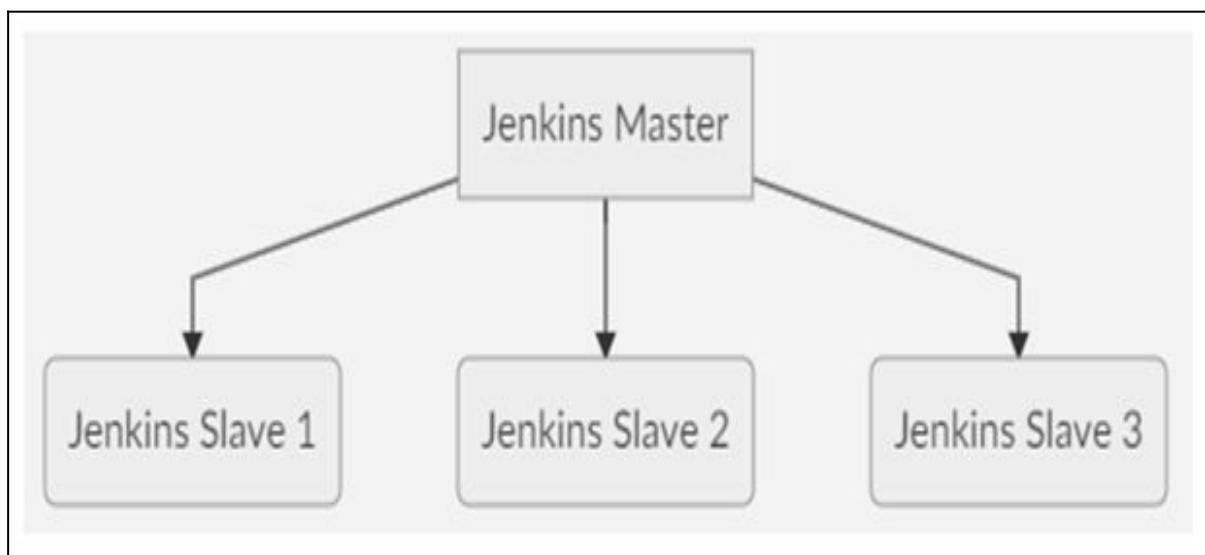
AIM:

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

PROBLEM DEFINITION:

Understand Jenkins Master-Slave architecture and expand a standalone Jenkins setup by adding slave nodes.

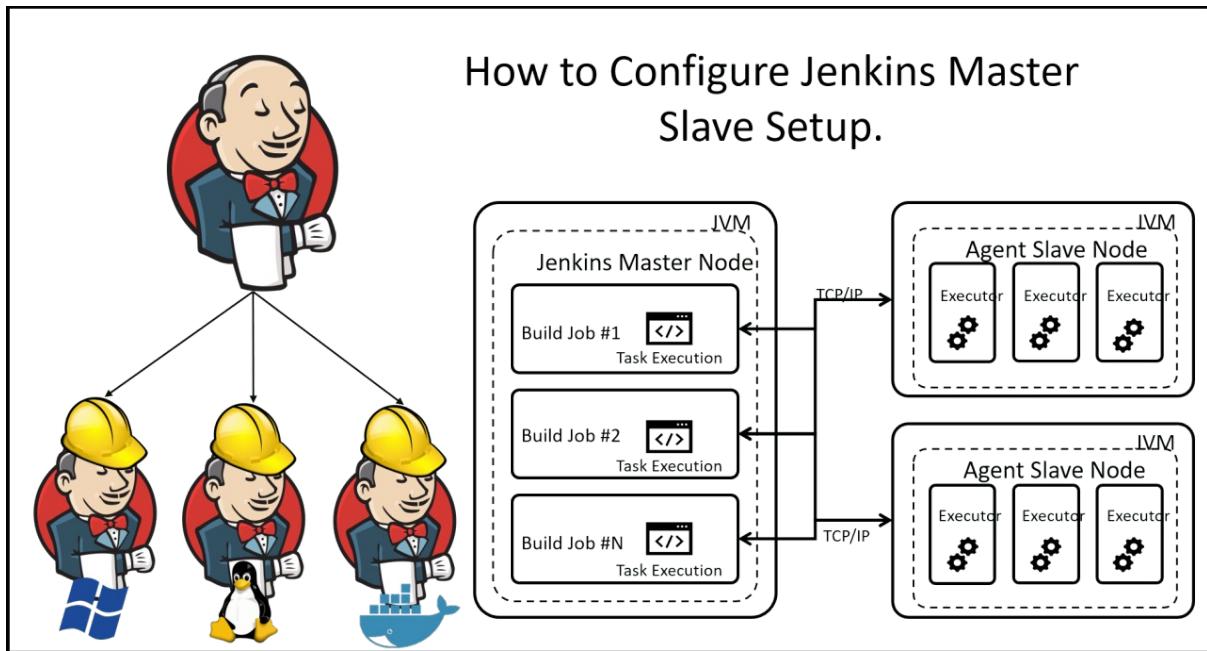
THEORY:



Understanding the Master and Slave Architecture

A standalone Jenkins instance can quickly grow into a resource-intensive application. To manage this, Jenkins can be scaled using a slave node architecture, which offloads some responsibilities from the master Jenkins instance. A Jenkins slave node is a device configured to act as an automation executor on behalf of the master. The Jenkins master represents the base Jenkins installation, performing basic operations and serving the user interface, while the slaves handle the heavy lifting.

This distributed computing model allows the Jenkins master to remain responsive to users while offloading automation execution to connected slave(s). For example, a Jenkins master schedules jobs, assigns them to slaves, sends builds to slaves for execution, monitors slave states (online or offline), and displays build results.



Step to Configure Jenkins Master and Slave Nodes:

1. Click on “Manage Jenkins”

Screenshot of the Jenkins Manage Jenkins page:

- Header:** Jenkins / Manage Jenkins
- Top Bar:** Search settings, Upgrade Automatically (button)
- Message Bar:** New version of Jenkins (2.516.2) is available for download ([changelog](#)). Or Upgrade Automatically (button)
- Warning Bar:** Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#). (Buttons: Set up agent, Set up cloud, Dismiss)
- Java 17 end of life in Jenkins:** You are running Jenkins on Java 17, support for which will end on or after 31-Mar-2026. Refer to [the documentation](#) for more details. (Buttons: More Info, Ignore)
- Warnings Section:** Warnings have been published for the following currently installed components:
 - Git client plugin 6.3.0:** File system information disclosure vulnerability. A fix for this issue is available. Go to the [plugin manager](#) to update the plugin.
 - Jakarta Mail API 2.1.3-2:** SMTP command injection vulnerability. A fix for this issue is available. Go to the [plugin manager](#) to update the plugin.
 (Buttons: Go to plugin manager, Configure which of these warnings are shown)

2. Click on “Manage Nodes”

Jenkins / Manage Jenkins / Nodes

Nodes

+ New Node Configure Monitors ⌂

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
💻	Built-In Node	Windows 11 (amd64)	In sync	66.50 GiB	12.60 GiB	66.50 GiB	0ms ⚡
☒	slave		N/A	N/A	N/A	N/A	N/A ⚡
💻	slave-1	Windows 11 (amd64)	In sync	66.50 GiB	12.59 GiB	66.50 GiB	100ms ⚡
	Data obtained	22 min	22 min	22 min	22 min	22 min	22 min

Icon: S M L Legend

3. Select “New Node” and enter the node name in the “Node Name” field.

Jenkins / Manage Jenkins / Nodes / New node

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 required information.
 5. Enter the Hostname in the “Host” field.
 6. Click the “Add” button to add credentials and click “Jenkins.”
 7. Enter Username, Password, ID, and Description.

Jenkins / Manage Jenkins / Nodes

Name ?
slave-2

Description ?

Plain text [Preview](#)

Number of executors ?
1

Remote root directory ?
C:\Program Files\Jenkins\remoting

Labels ?
job1

Save

8. Select the dropdown menu to add credentials into the “Credentials” field.
9. Select the dropdown to add the Host Key Verifications Strategy under “Non-verifying Verification Strategy.”
10. Select “Keep this agent online as much as possible” in the "Availability" field.

```
PS C:\Program Files\Jenkins\remoting> java -jar agent.jar -url http://localhost:8080/ -secret 12f110ef6cb10cf3c23fad6fd227f7717d26894d0d9b014c11425289076ca3
@6 -name "slave-1" -webSocket -workDir "C:\Program Files\Jenkins\remoting"
Sep 10, 2025 6:13:02 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\Program Files\Jenkins\remoting\remoting as a remoting work directory
Sep 10, 2025 6:13:02 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to C:\Program Files\Jenkins\remoting\remoting
Sep 10, 2025 6:13:02 AM hudson.remoting.Launcher createEngine
INFO: Setting up agent: slave-1
Sep 10, 2025 6:13:02 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3309.v27b_9314fd1a_4
Sep 10, 2025 6:13:02 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\Program Files\Jenkins\remoting\remoting as a remoting work directory
Sep 10, 2025 6:13:02 AM hudson.remoting.Launcher$CuilListener status
INFO: WebSocket connection open
Sep 10, 2025 6:13:02 AM hudson.remoting.Launcher$CuilListener status
INFO: Connected
```

The screenshot shows the Jenkins interface for managing a slave node named 'slave-1'. The top navigation bar includes links for Jenkins, Nodes, and slave-1, along with search and settings icons. The main content area has tabs for Status (selected), Delete Agent, Configure, Build History, Load Statistics, Script Console, Log, System Information, and Disconnect. The Status tab displays the message 'Agent is connected.' and a 'Monitoring Data' dropdown menu. The 'Labels' section shows 'job1'. The 'Projects tied to slave-1' section lists 'None'. A 'Build Executor Status' box at the bottom indicates '(0 of 1 executor busy)'. The overall layout is clean and organized, typical of the Jenkins web UI.

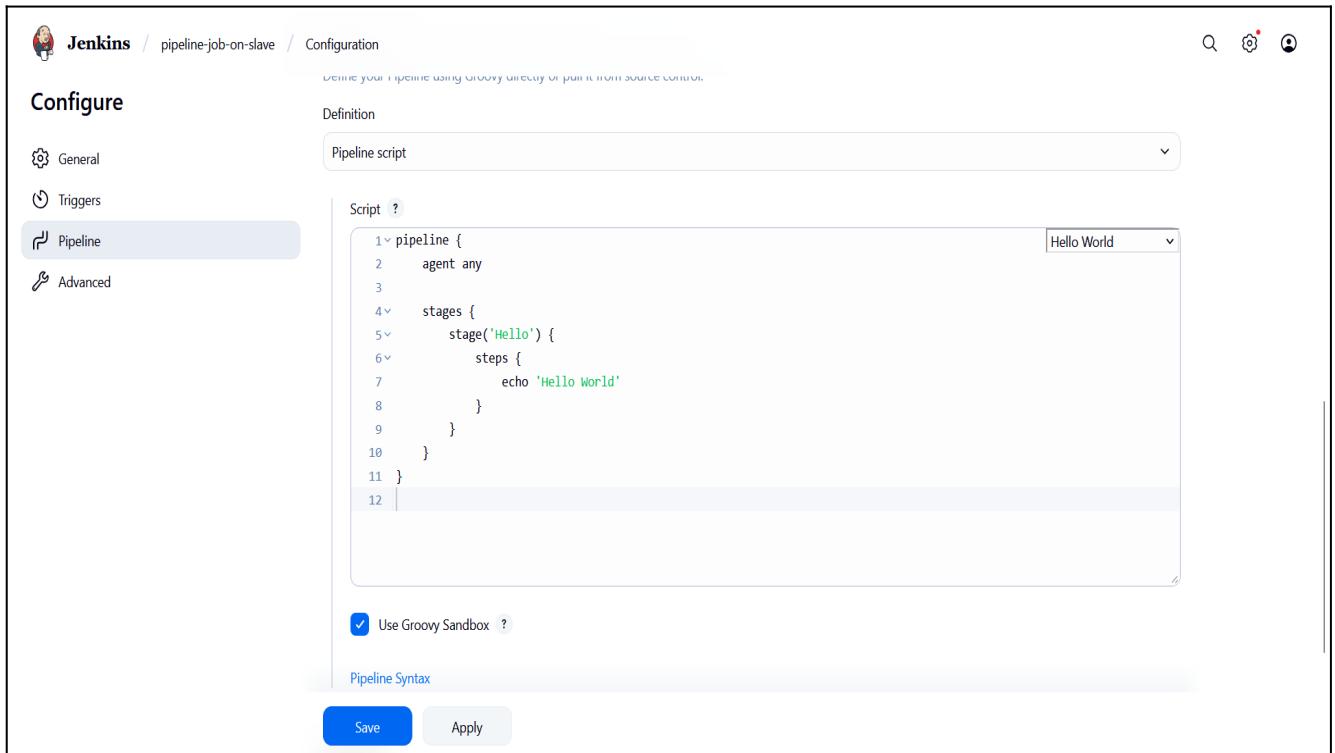
Creating a Freestyle Project and Running on the Slave Machine:

1. Click “New Item”, which will redirect to the job’s new page.
2. Enter the project name, select “Freestyle Project”, and click OK.
3. On the job’s page, click the “Build Now” button to execute your pipeline.
4. Verify the history of the executed build number under “Build History” by checking the build status and output on the remote host.
5. Click on the build number and select “Console Output” to view the executed job and output on the remote host.

Creating a Pipeline and Running on the Slave Machine:

1. Click “New Item” in the top left corner on the dashboard.
2. Enter the project name in the “Enter an item name” field, select the “Pipeline” project, and click OK.
3. Give a description (optional).
4. Go to the “Pipeline” section.
5. Copy and paste the desired Pipeline script into the script field.
6. Save the changes, which will redirect to the Pipeline new page
7. On the left pane, click the “Build Now” button to execute your pipeline.
8. Verify the history of executed builds under “Build History” by clicking the build status.
9. On each build, the output will display on the results page.

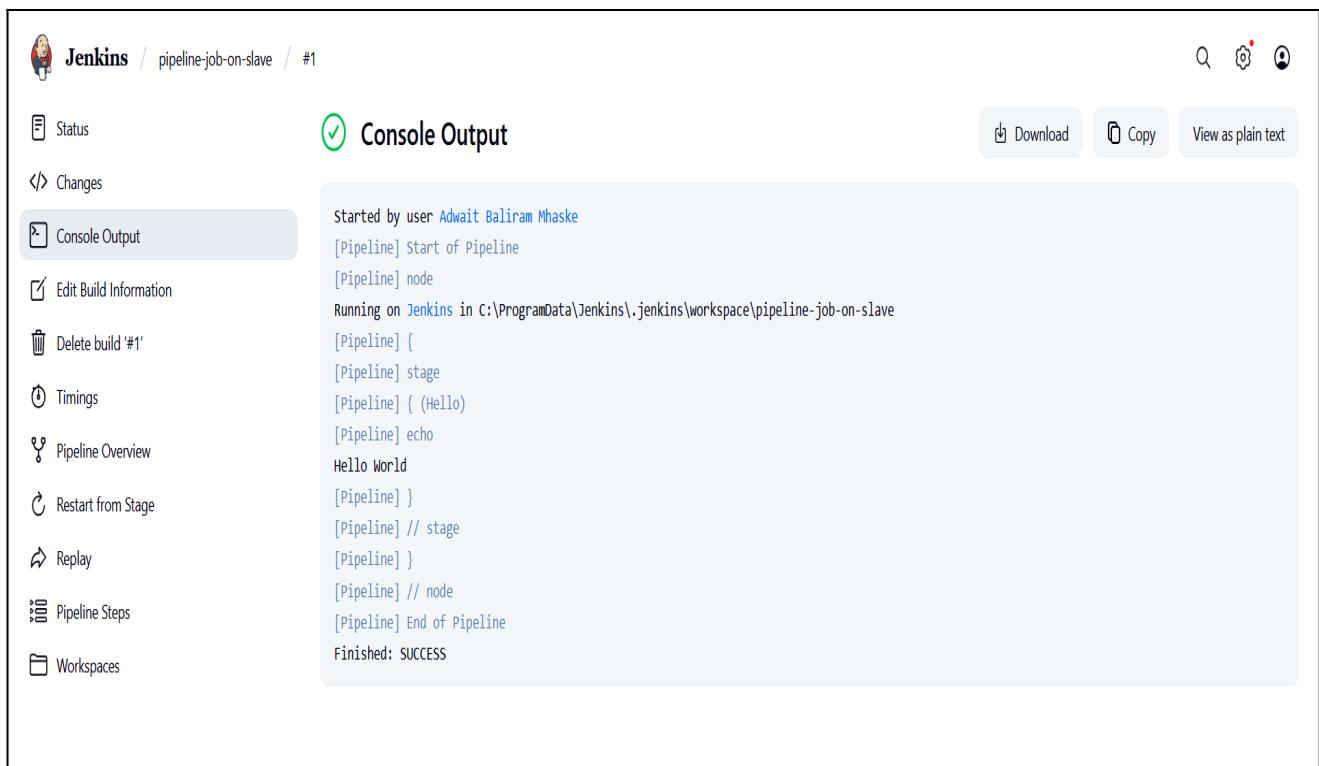
10. Click on the build number and select “Console Output” to see that the pipeline ran on a slave machine.



The screenshot shows the Jenkins Pipeline configuration page for a job named "pipeline-job-on-slave". The "Pipeline" tab is selected in the left sidebar. The main area displays a Groovy pipeline script:

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11}  
12
```

A checkbox labeled "Use Groovy Sandbox" is checked. At the bottom are "Save" and "Apply" buttons.



The screenshot shows the Jenkins console output for build #1 of the "pipeline-job-on-slave" job. The "Console Output" tab is selected in the left sidebar. The output window displays the following log entries:

```
Started by user Adwait Baliram Mhaske  
[Pipeline] Start of Pipeline  
[Pipeline] node  
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\pipeline-job-on-slave  
[Pipeline] {  
[Pipeline] stage  
[Pipeline] { (Hello)  
[Pipeline] echo  
Hello World  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```