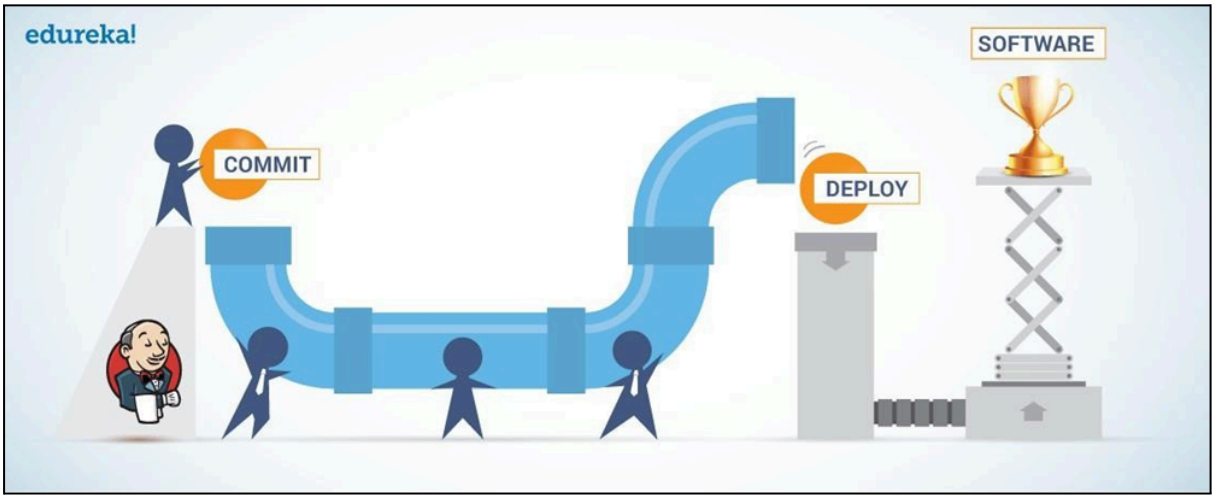
<table>
<tr><td colspan="3">Name of Student: Pushkar Sane</td></tr>
<tr><td colspan="2">Roll Number: 45</td><td>Lab Assignment Number: 10</td></tr>
<tr><td colspan="3">Title of Lab Assignment: To implement Jenkins Master/Slave architecture.</td></tr>
<tr><td colspan="2">DOP: 28-03-2024</td><td>DOS: 05-04-2024</td></tr>
<tr><td>CO Mapped:
CO3</td><td>PO Mapped:
PO2, PO3, PO5, PSO1, PSO2</td><td>Signature:</td></tr>
</table>

Practical No. 10

Aim: To implement Jenkins Master/Slave architecture.

Introduction:

What is Jenkins?



Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis and much more.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allow the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example Git, Maven 2 project, Amazon EC2, HTML publisher, etc.

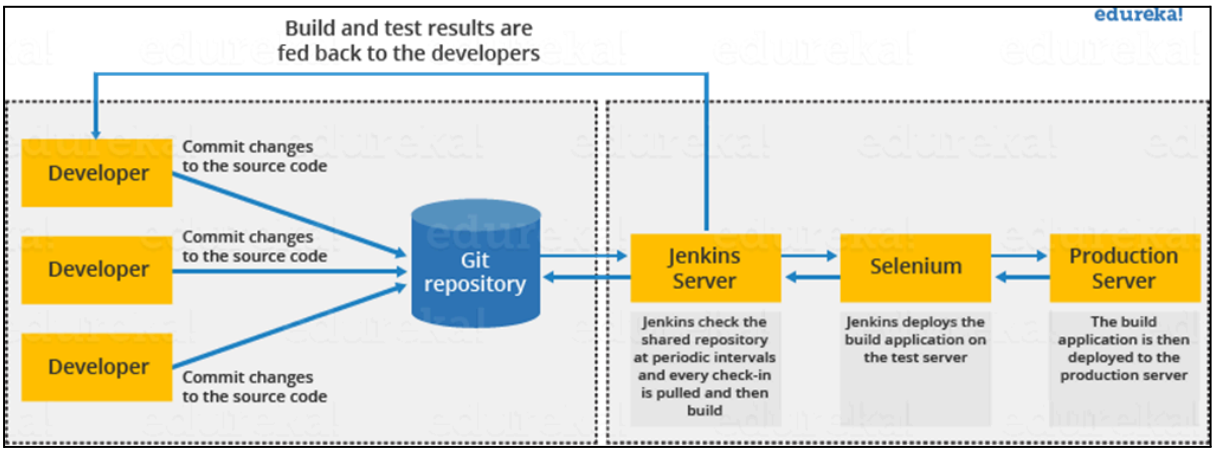
Advantages of Jenkins include:

- It is an open-source tool with great community support.
- Too easy to install.

- It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share it with the community.
- It is free of cost.
- It is built with Java and hence, it is portable to all the major platforms.

Jenkins Architecture

Let us have a look at the Jenkins Architecture below diagram depicts the same.



This single Jenkins server was not enough to meet certain requirements like:

- Sometimes you might need several different environments to test your builds. This cannot be done by a single Jenkins server.
- If larger and heavier projects get built on a regular basis then a single Jenkins server cannot simply handle the entire load.

To address the above-stated needs, Jenkins distributed architecture came into the picture.

Jenkins Distributed Architecture

Jenkins uses a Master-Slave architecture to manage distributed builds. In this architecture, Master and Slave communicate through TCP/IP protocol.

Jenkins Master

Your main Jenkins server is the Master. The Master's job is to handle:

- Scheduling build jobs.
- Dispatching builds to the slaves for the actual execution.
- Monitor the slaves (possibly taking them online and offline as required).
- Recording and presenting the build results.
- A Master instance of Jenkins can also execute build jobs directly.

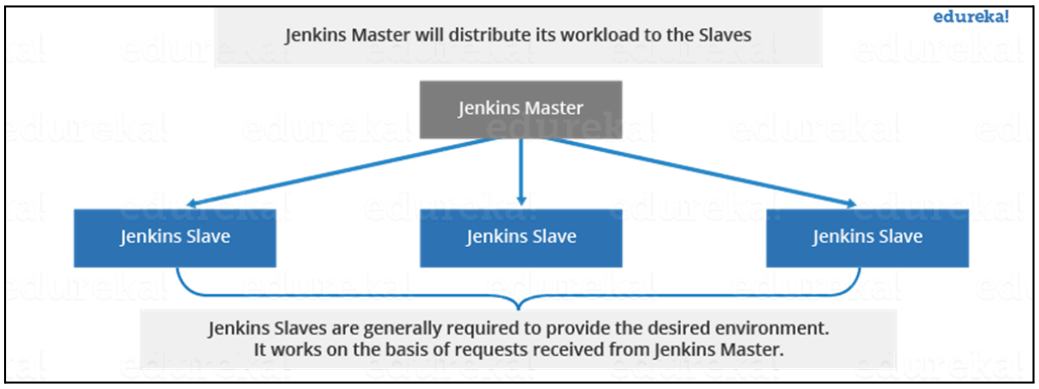
Jenkins Slave

A Slave is a Java executable that runs on a remote machine. Following are the characteristics of Jenkins

Slaves:

- It hears requests from the Jenkins Master instance.
- Slaves can run on a variety of operating systems.
- The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.
- You can configure a project to always run on a particular Slave machine or a particular type of Slave machine, or simply let Jenkins pick the next available Slave.

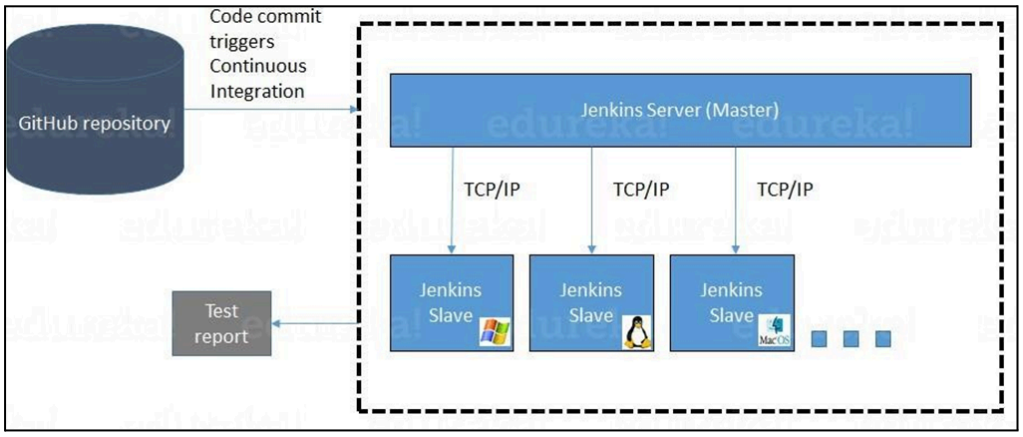
The diagram below is self-explanatory. It consists of a Jenkins Master which is managing three Jenkins Slave.



How does Jenkins Master and Slave Architecture work?

Now let us look at an example in which we use Jenkins for testing in different environments like Ubuntu, MAC, Windows, etc.

The diagram below represents the same:

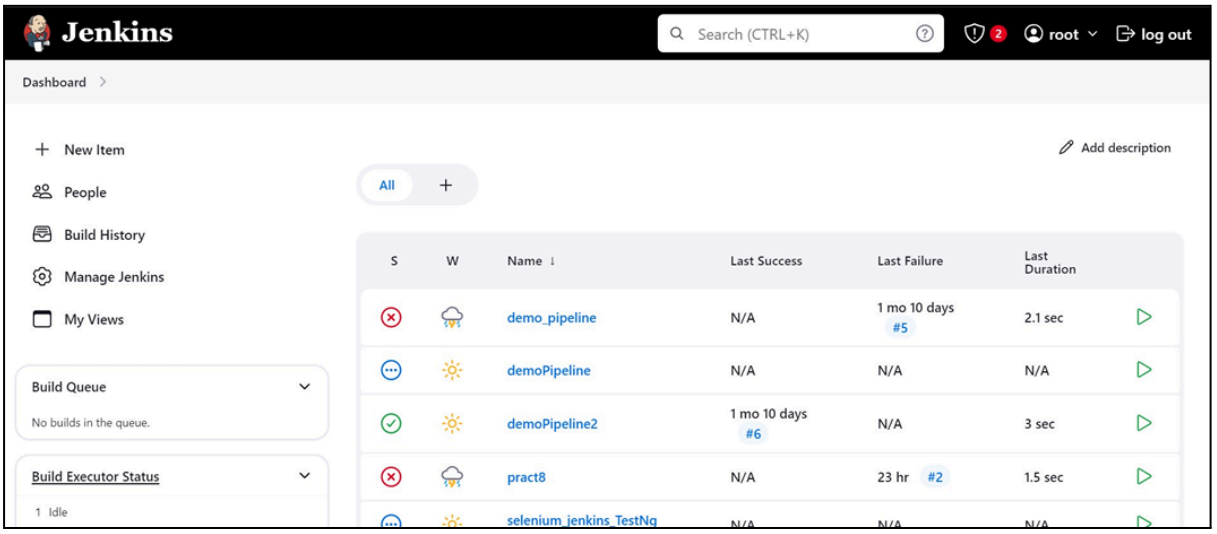


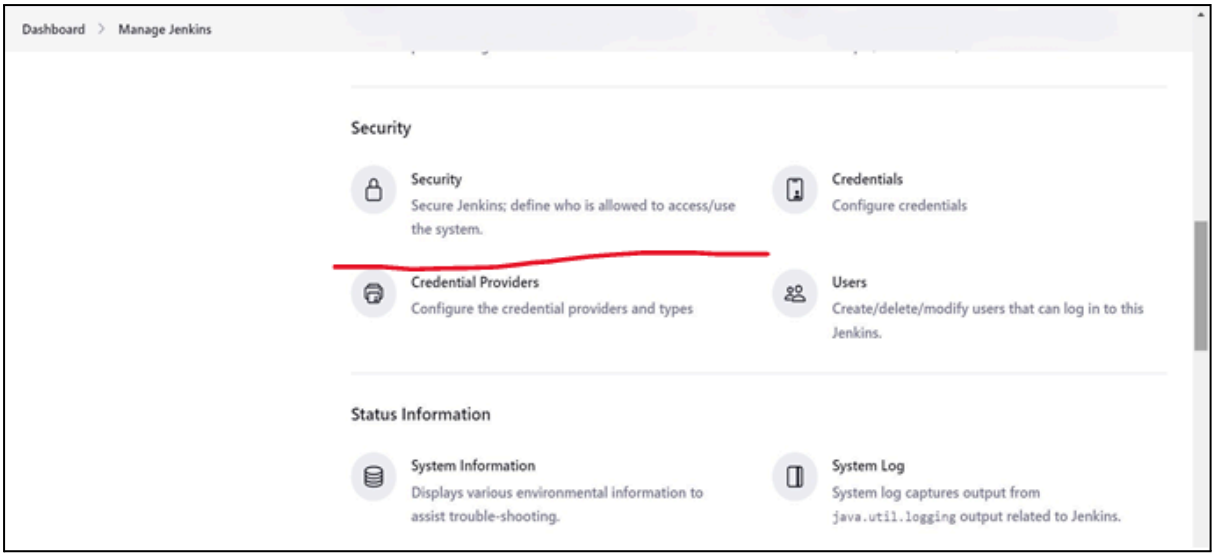
The above image represents the following functions:

- Jenkins checks the Git repository at periodic intervals for any changes made in the source code.
- Each build requires a different testing environment which is not possible for a single Jenkins server. In order to perform testing in different environments, Jenkins uses various Slaves as shown in the diagram.
- Jenkins Master requests these Slaves to perform

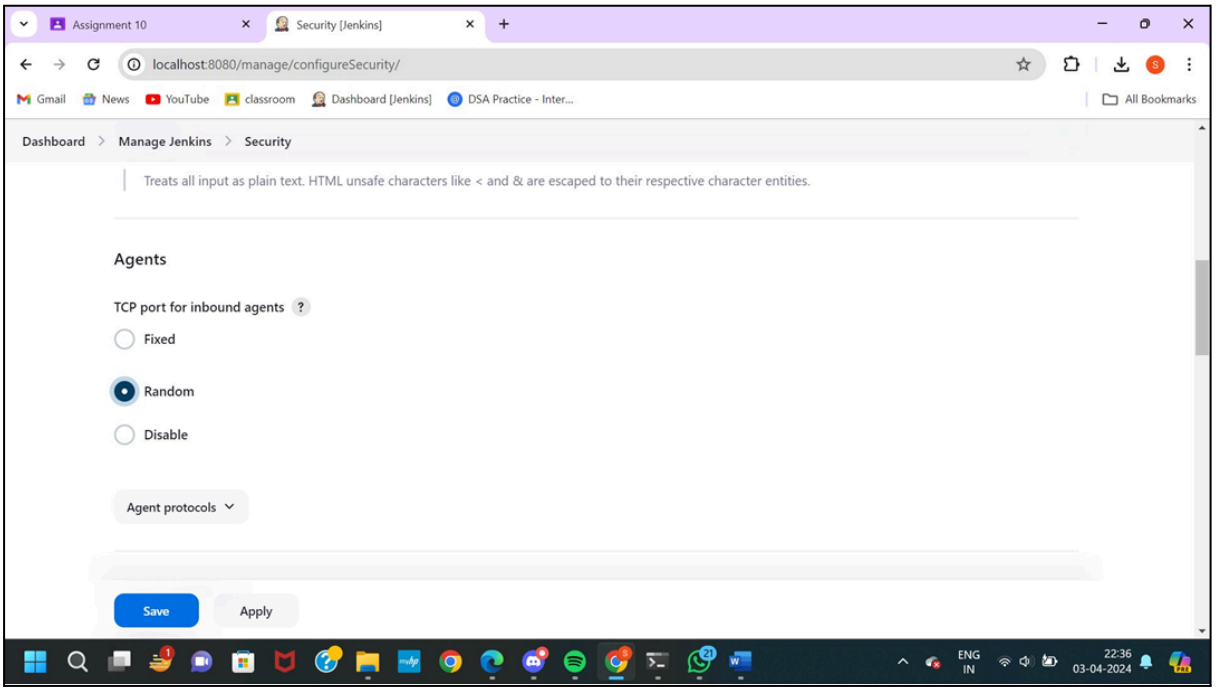
Steps:

Step 1: Open the Jenkins (localhost:8080) and navigate to the “Manage Jenkins” menu. Then select the “Security” option.

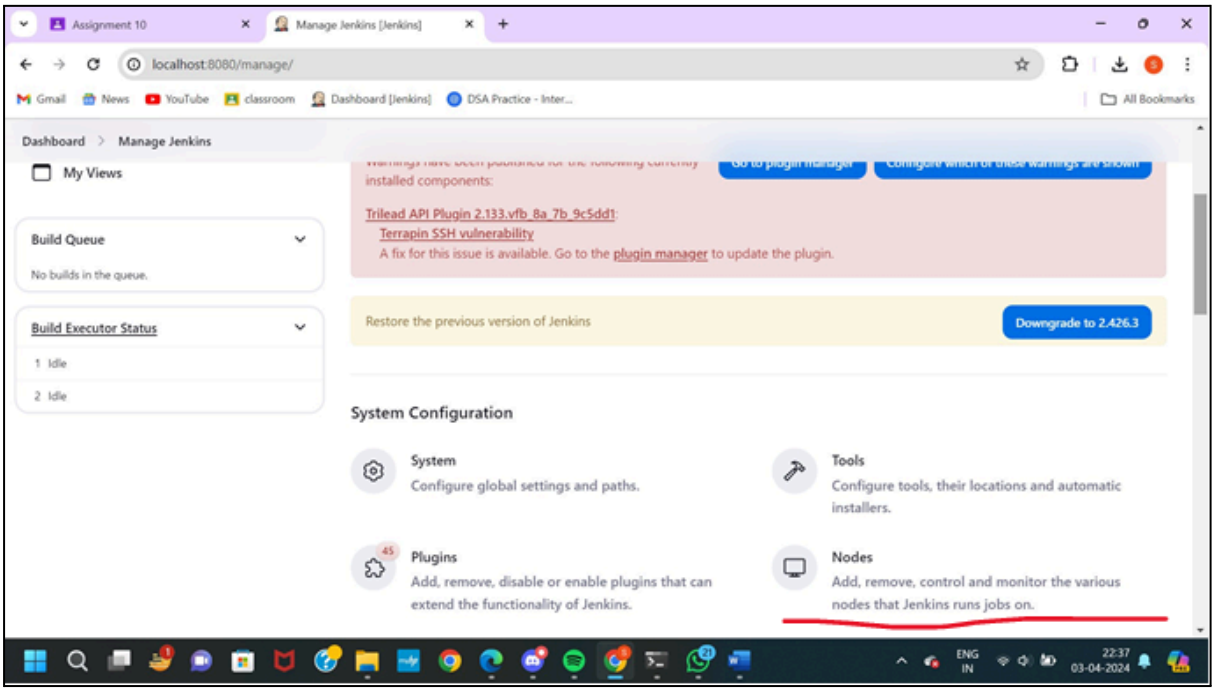




Step 2: Search for the “Agents” option and then select “Random” under the TCP ports for inbound agents menu and then scroll down and click on apply and save.

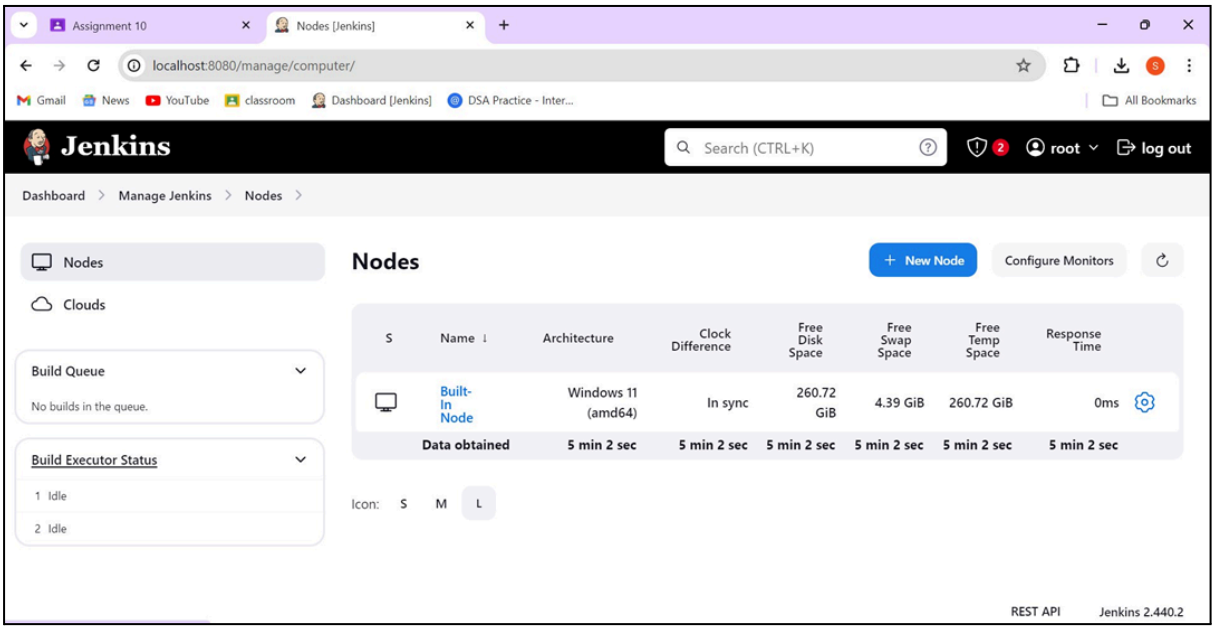


Step 3: Go to the dashboard and again navigate to the Manage Jenkins Menu, and then select the “Nodes” menu.

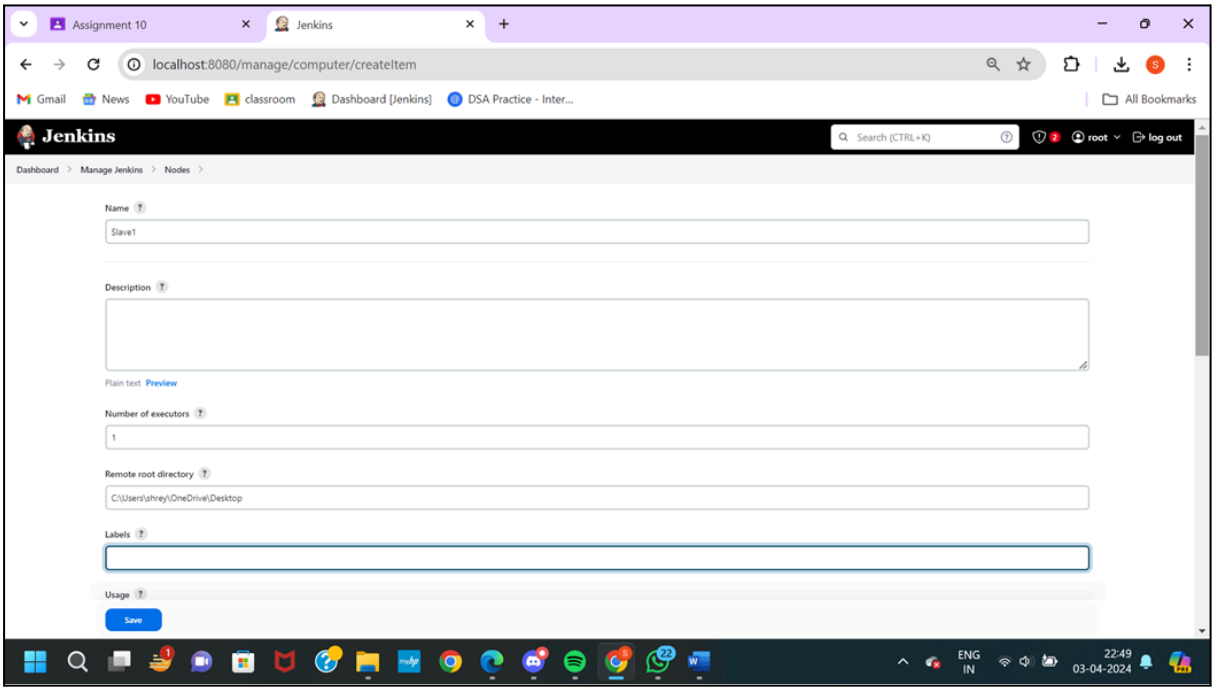


Step 4: You will be able to see the “Built-in Node” in the Nodes list. This node is also known as Master Node which is basically the current OS.

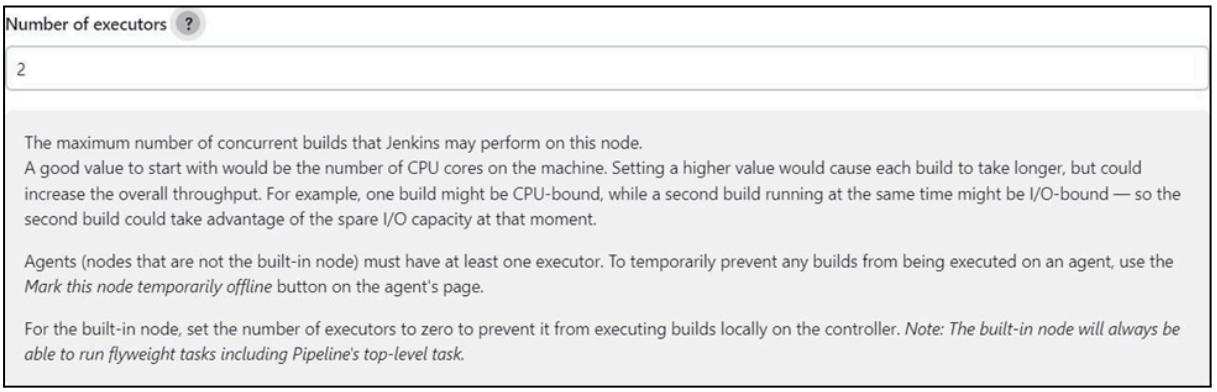
Now click on the “New Node” to create a new Slave node.

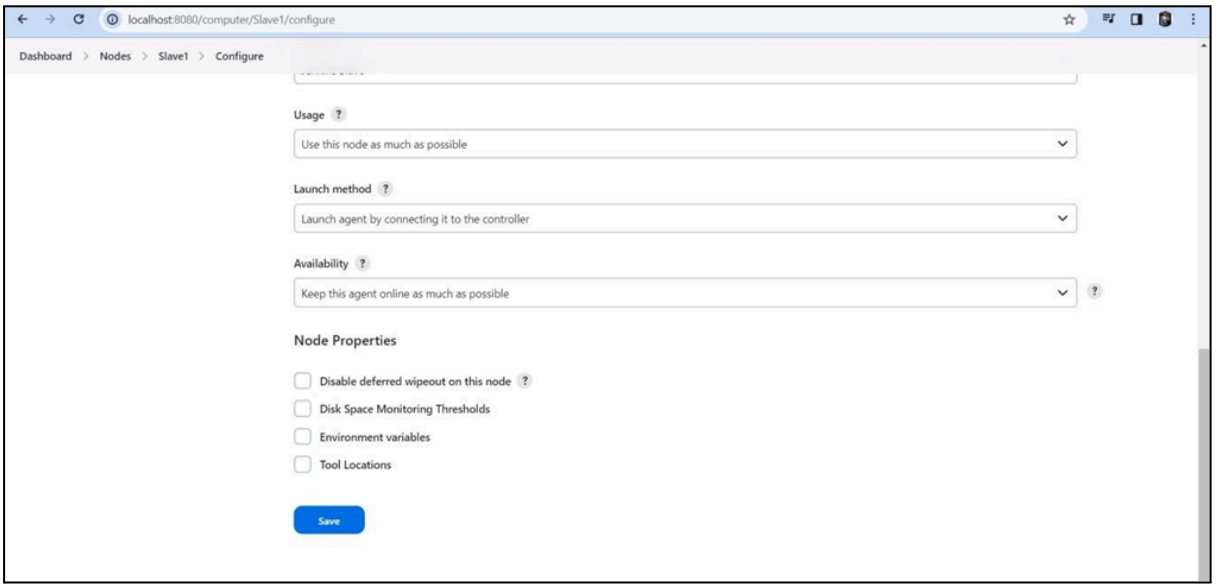


Step 5: Configure the Slave node.



For Remote root directory, create a new folder on your device (Empty) and copy paste the path of that folder.





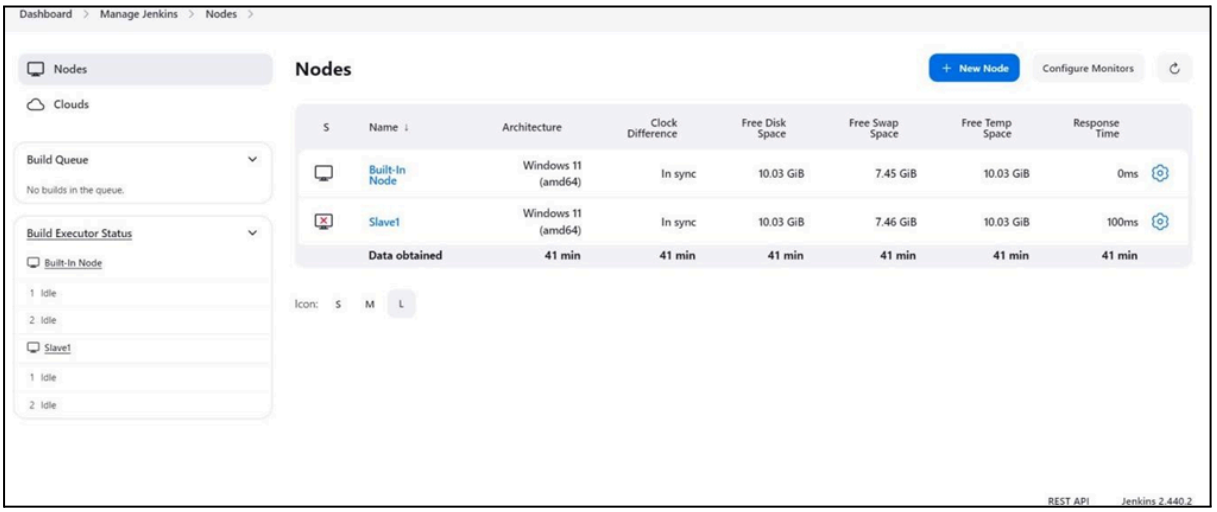
Labels can be anything according to your choice

Select the options exactly as above, if you are not able to see the above options in the dropdown list then go to step 1 and select TCP inbounds as random and check again.

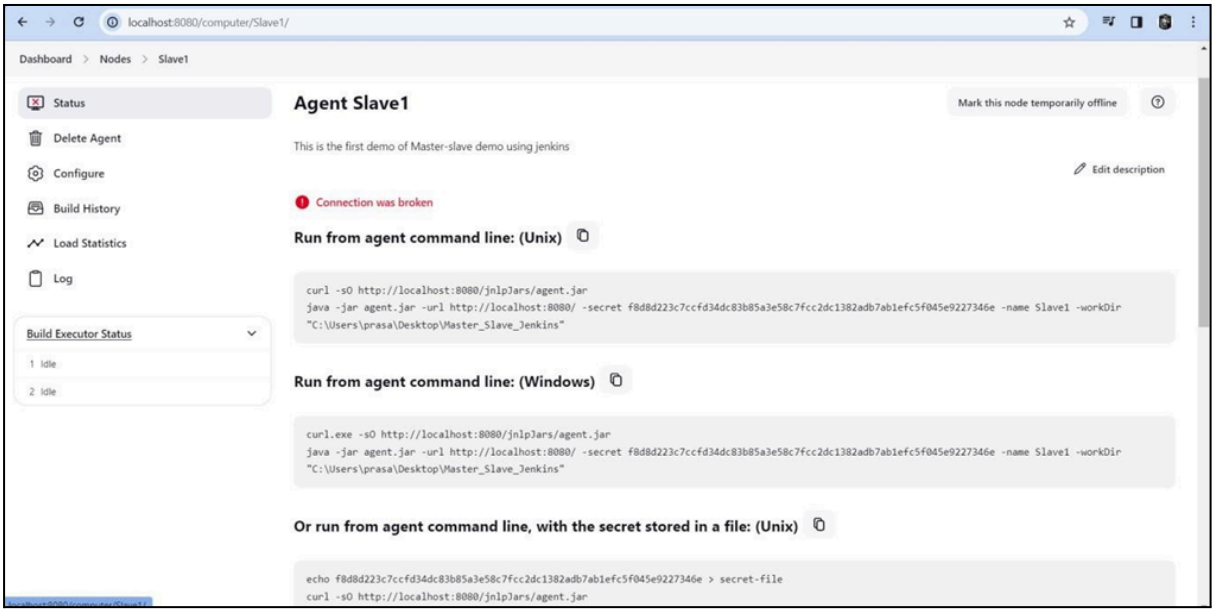
Click on save.

Step 6: Once the node is saved, you will see the cross sign on the slave node which basically means that the slave is not yet connected or running.

To connect the slave, click on the slave node and then select the status option. On the same page you will see multiple commands for different operating systems. Copy the command for windows

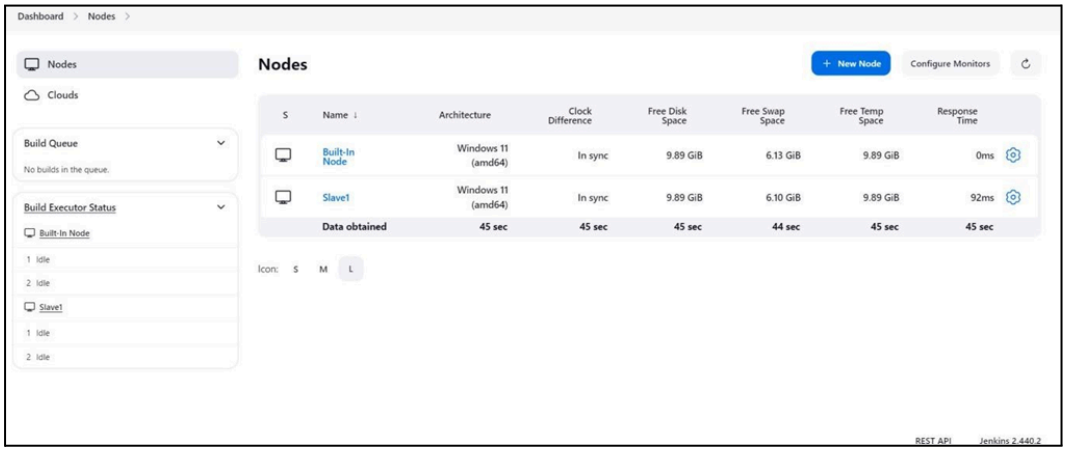


and run it in the command prompt.

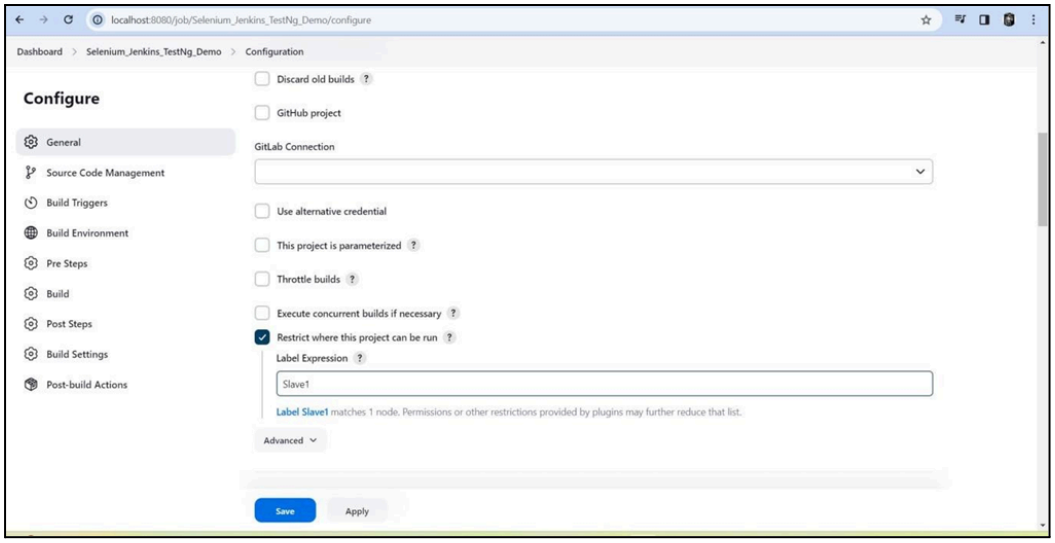


```
Command Prompt - java -jar
Apr 01, 2024 8:14:53 PM hudson.remoting.Launcher createEngine
INFO: Setting up agent: Slave1
Apr 01, 2024 8:14:53 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3206.vb_15dcf73f6a_9
Apr 01, 2024 8:14:53 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\Users\prasa\Desktop\Master_Slave_Jenkins\remoting as a remoting work directory
Apr 01, 2024 8:14:53 PM hudson.remoting.Launcher$CuiListener status
INFO: Locating server among [http://localhost:8080/]
Apr 01, 2024 8:14:53 PM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Apr 01, 2024 8:14:53 PM hudson.remoting.Launcher$CuiListener status
INFO: Agent discovery successful
  Agent address: localhost
  Agent port:    53285
  Identity:      86:28:2c:d6:b0:51:dd:6b:52:6a:b0:17:37:e3:31:38
Apr 01, 2024 8:14:53 PM hudson.remoting.Launcher$CuiListener status
INFO: Handshaking
Apr 01, 2024 8:14:53 PM hudson.remoting.Launcher$CuiListener status
INFO: Connecting to localhost:53285
Apr 01, 2024 8:14:53 PM hudson.remoting.Launcher$CuiListener status
INFO: Server reports protocol JNLP4-connect-proxy not supported, skipping
Apr 01, 2024 8:14:53 PM hudson.remoting.Launcher$CuiListener status
INFO: Trying protocol: JNLP4-connect
Apr 01, 2024 8:14:54 PM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Apr 01, 2024 8:14:54 PM hudson.remoting.Launcher$CuiListener status
INFO: Remote identity confirmed: 86:28:2c:d6:b0:51:dd:6b:52:6a:b0:17:37:e3:31:38
Apr 01, 2024 8:14:54 PM hudson.remoting.Launcher$CuiListener status
INFO: Connected
```

Step 7: Once the slave node is connected and running, we can create a new job for the slave node to perform remotely.

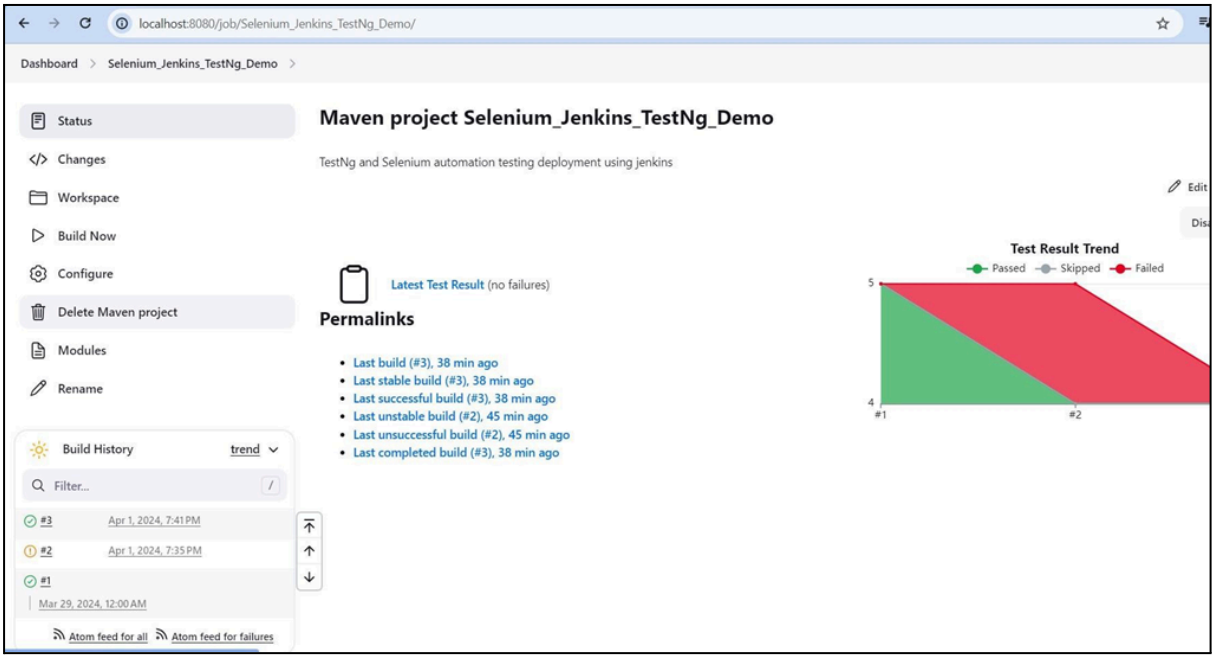


Go to any previously built Jenkins project and click on configure.



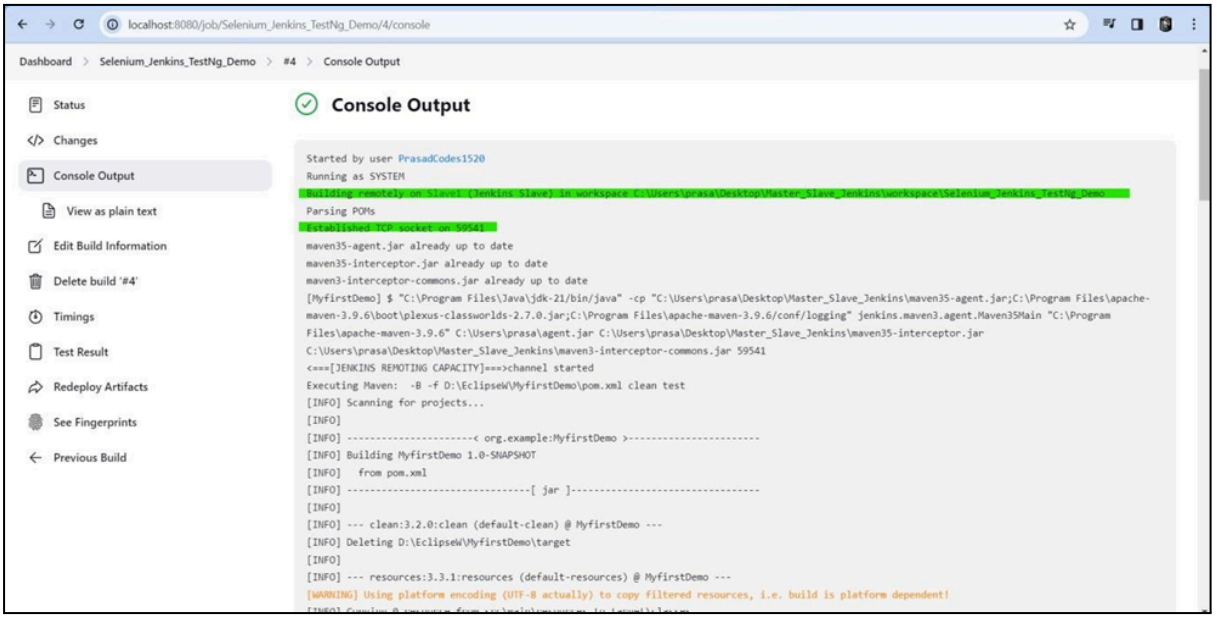
- Then search for the “Restrict where this project can be run” option select it.
- And enter the name of the slave node in the empty field.

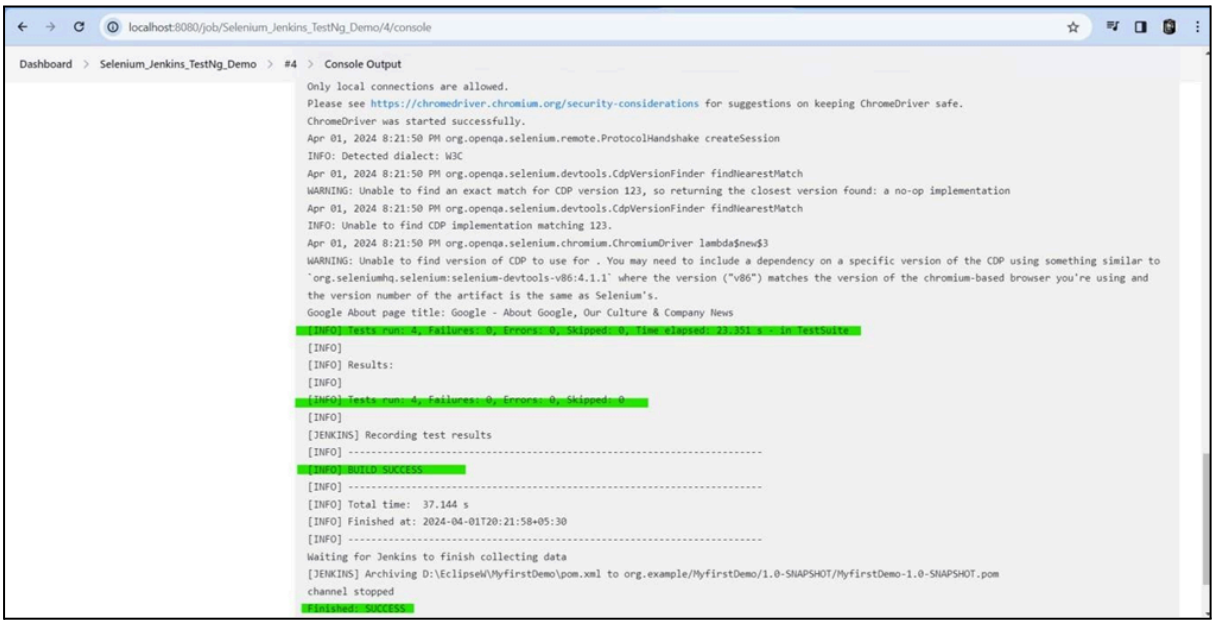
Step 8: Click on save, and then build the Jenkins project. (Build Now).



Once you click on build now, the Jenkins project will be remotely handled by the slave node which we configured.

Step 9: To check if the project has been remotely handled by slave or not, go to the console output of the recently built project. (Step 8)





Conclusion: Implementing Jenkins with a master-slave architecture optimizes resource utilization, scalability, and fault tolerance. By distributing tasks across multiple nodes, it enhances efficiency, reduces build times, and ensures continuous operation. This setup fosters flexibility and agility in software development, promoting faster and more reliable delivery. Overall, it facilitates enhanced productivity and innovation within the organization.