Academic Year: 2025-26 **Semester: V**

Class / Branch: TEIT

Subject: DevOps Lab

Name of Instructor: Prof. Sujata Oak

Experiment No. 6

Aim: To implement Jenkins Master-Slave Architecture with Scaling.

Theory:

Objective

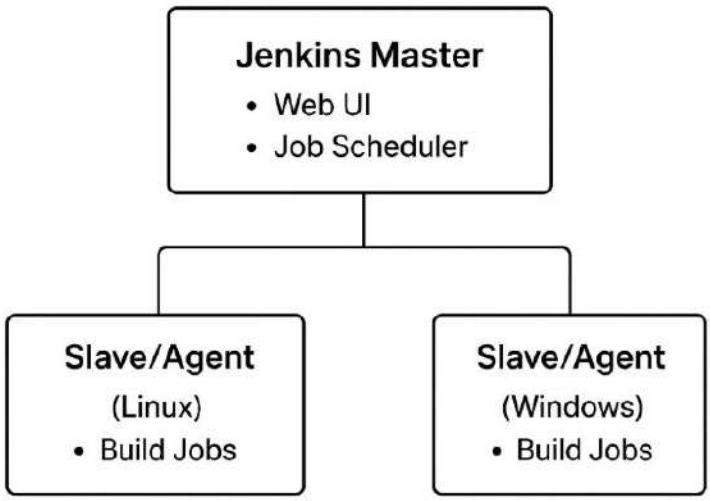
To understand and implement Jenkins' distributed build architecture where a central Jenkins Master (Controller) coordinates tasks and multiple Slave (Agent) nodes execute builds in parallel.

Purpose of Distributed Builds

- Large projects often require running builds and tests on different operating systems, environments, or hardware.
- A single server can become a bottleneck. Distributed builds increase **scalability**, **speed**, and **fault tolerance**.

Key Roles

- **Master (Controller):**
  - Hosts the Jenkins web UI and job configurations.
  - Schedules jobs, monitors nodes, and aggregates build results.
  - Decides *what* to build and *where* to build it.
- **Slave (Agent):**
  - A remote machine (physical/VM/container) where the actual build steps run.
  - Communicates with the master through an SSH or JNLP (Java Web Start) connection.
  - Can have specific labels (e.g., linux, windows) to run platform-dependent jobs.



Jenkins Master and Slave Concept

A Jenkins master comes with the basic installation of Jenkins, and in this configuration, the master handles all the tasks for our build system.
If we are working on multiple projects, we may run multiple jobs on each project. Some projects need to run on some nodes, and in this process, we need to configure slaves. Jenkins slaves connect to the Jenkins master using the Java Network Launch Protocol (JNLP).

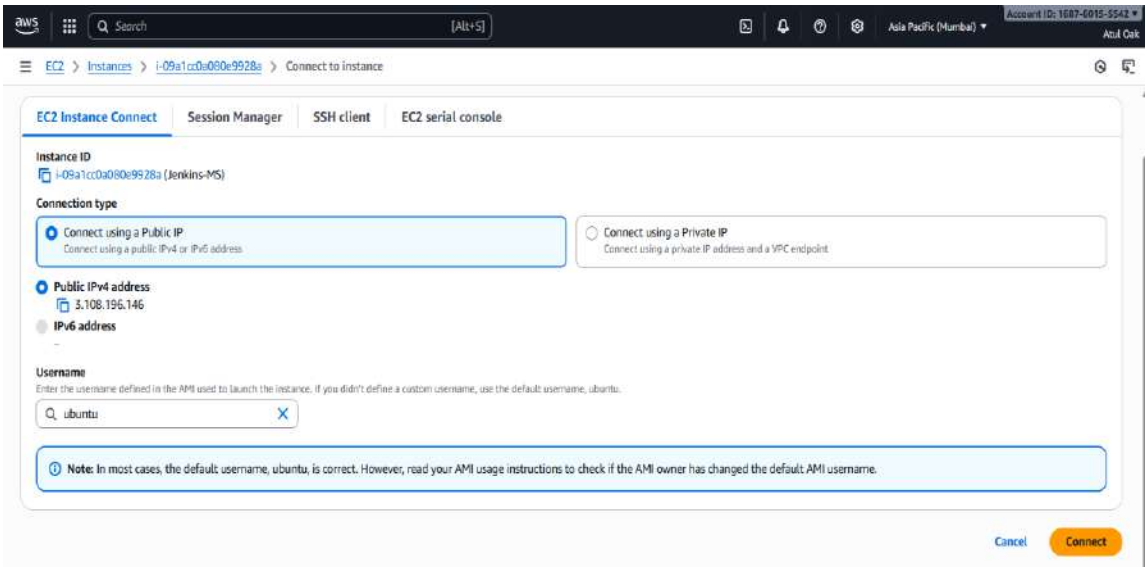
The Jenkins master acts to schedule the jobs, assign slaves, and send builds to slaves to execute the jobs.
It will also monitor the slave state (offline or online) and get back the build result responses from slaves and the display build results on the console output. The workload of building jobs is delegated to multiple **slaves**.

Advantages

- Parallel execution → faster CI/CD pipeline.
- Flexibility to run jobs on specific environments.
- Load distribution prevents the master from being overloaded.

Steps to Configure Jenkins Master and Slave Nodes

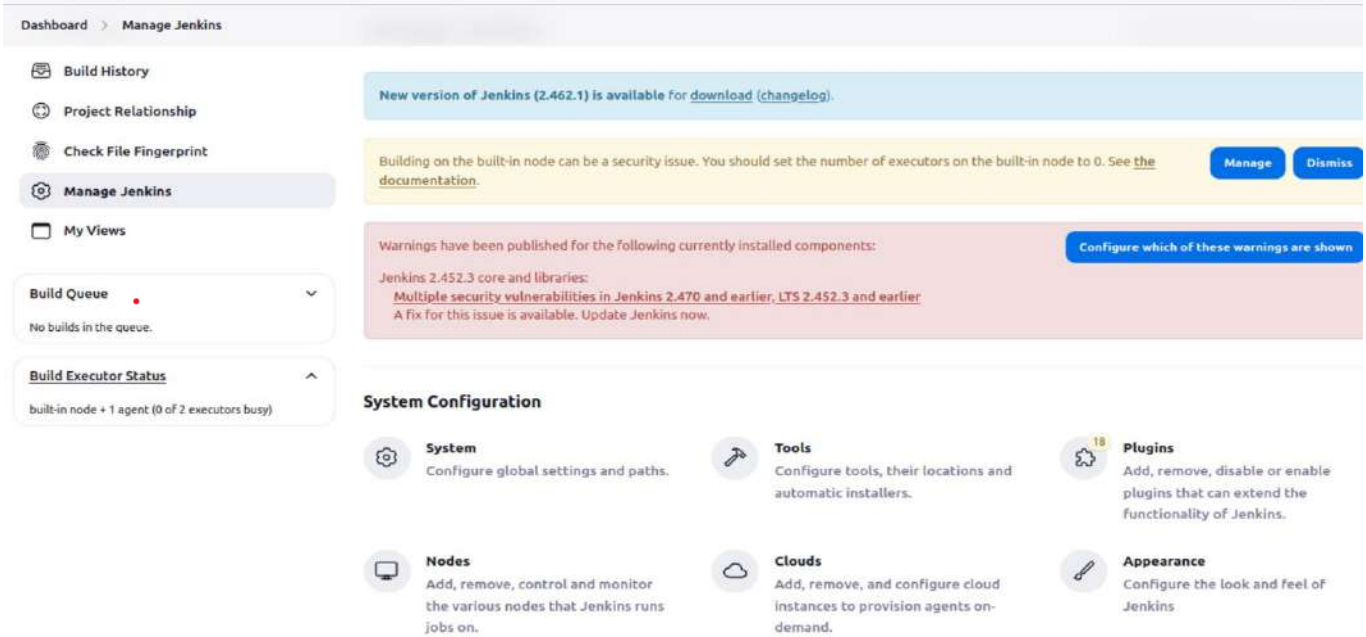
STEPA: Sign-In to AWS MANAGEMENT CONSOLE



Click on Connect

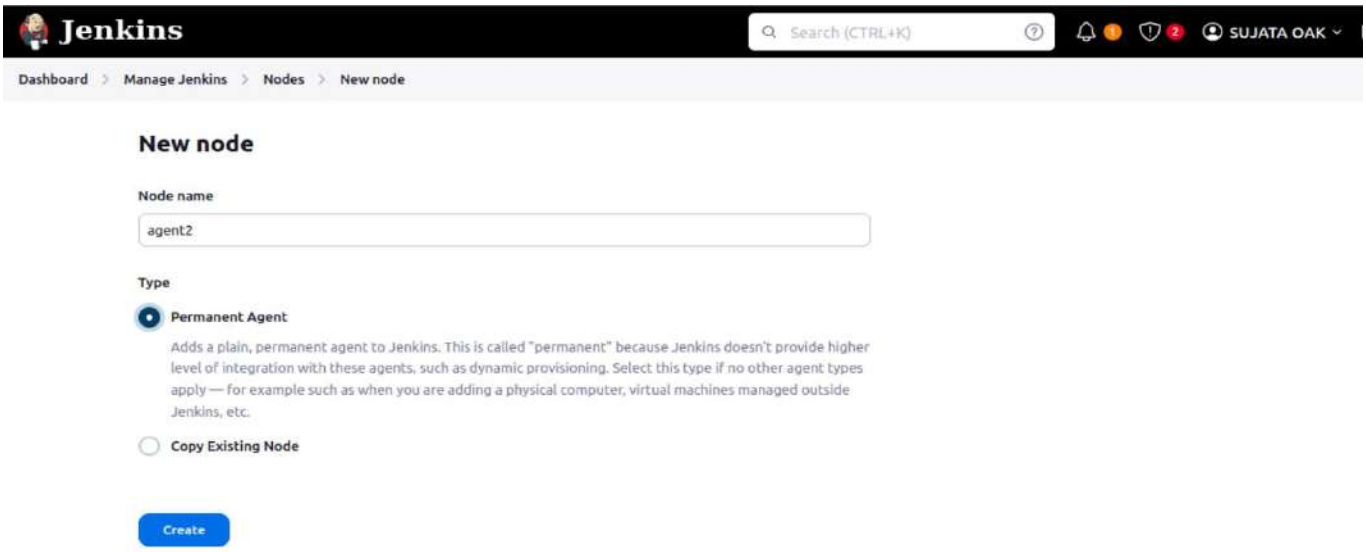
STEP B: Install Jenkins

STEP1: In Jenkins Dashboard Click on Manage Jenkins -> Manage Nodes



STEP 2: Select New Node and enter the name of the node in the Node Name field.

Select Permanent Agent and click the OK button. Initially, you will get only one option, "Permanent Agent." Once we have one or more slaves you will get the "Copy Existing Node" option. Click Create



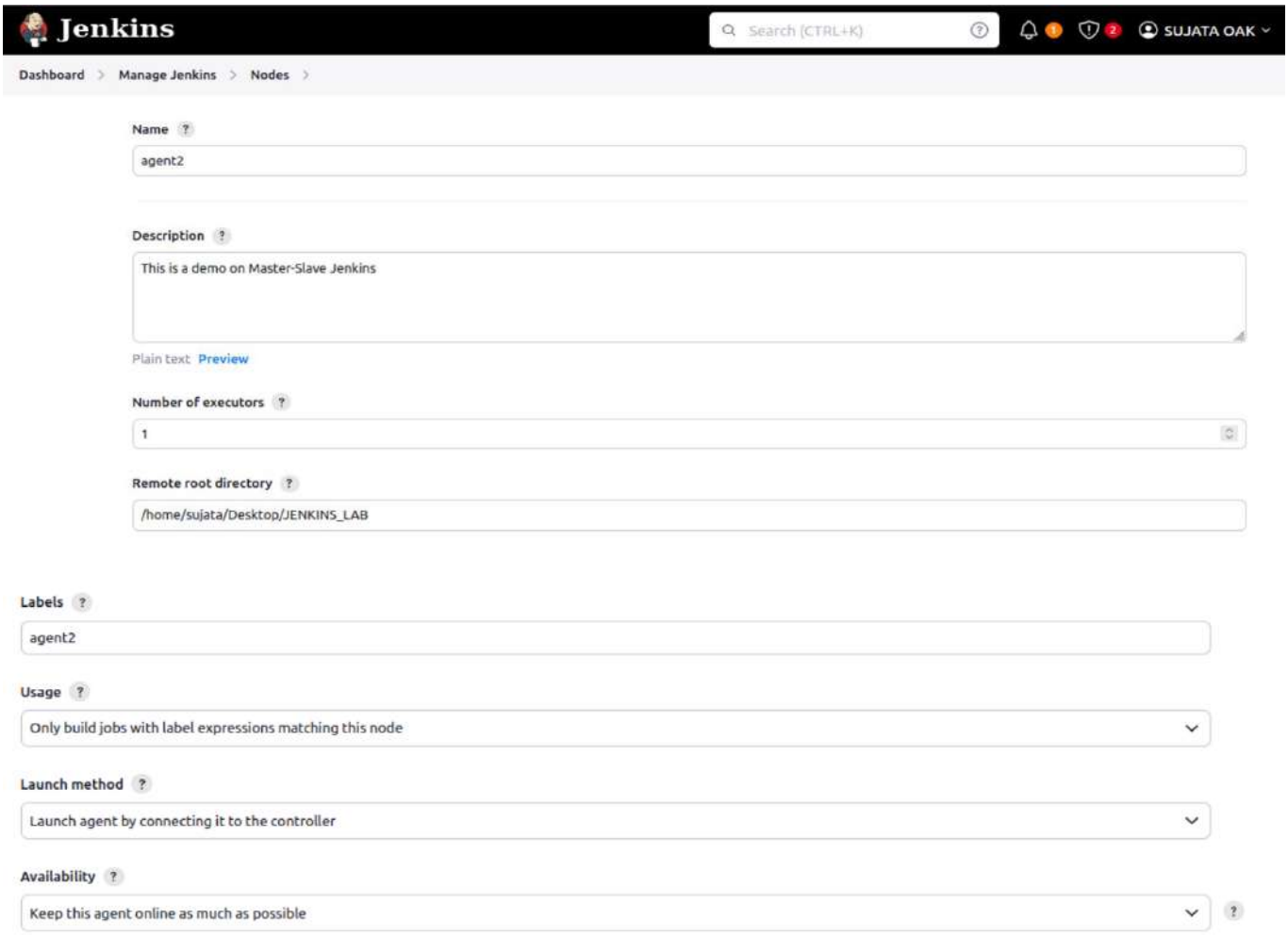
STEP3: Configure node with below details:

```
sujata@Ubuntu:~/Desktop/JENKINS_LAB$ pwd
/home/sujata/Desktop/JENKINS_LAB
```

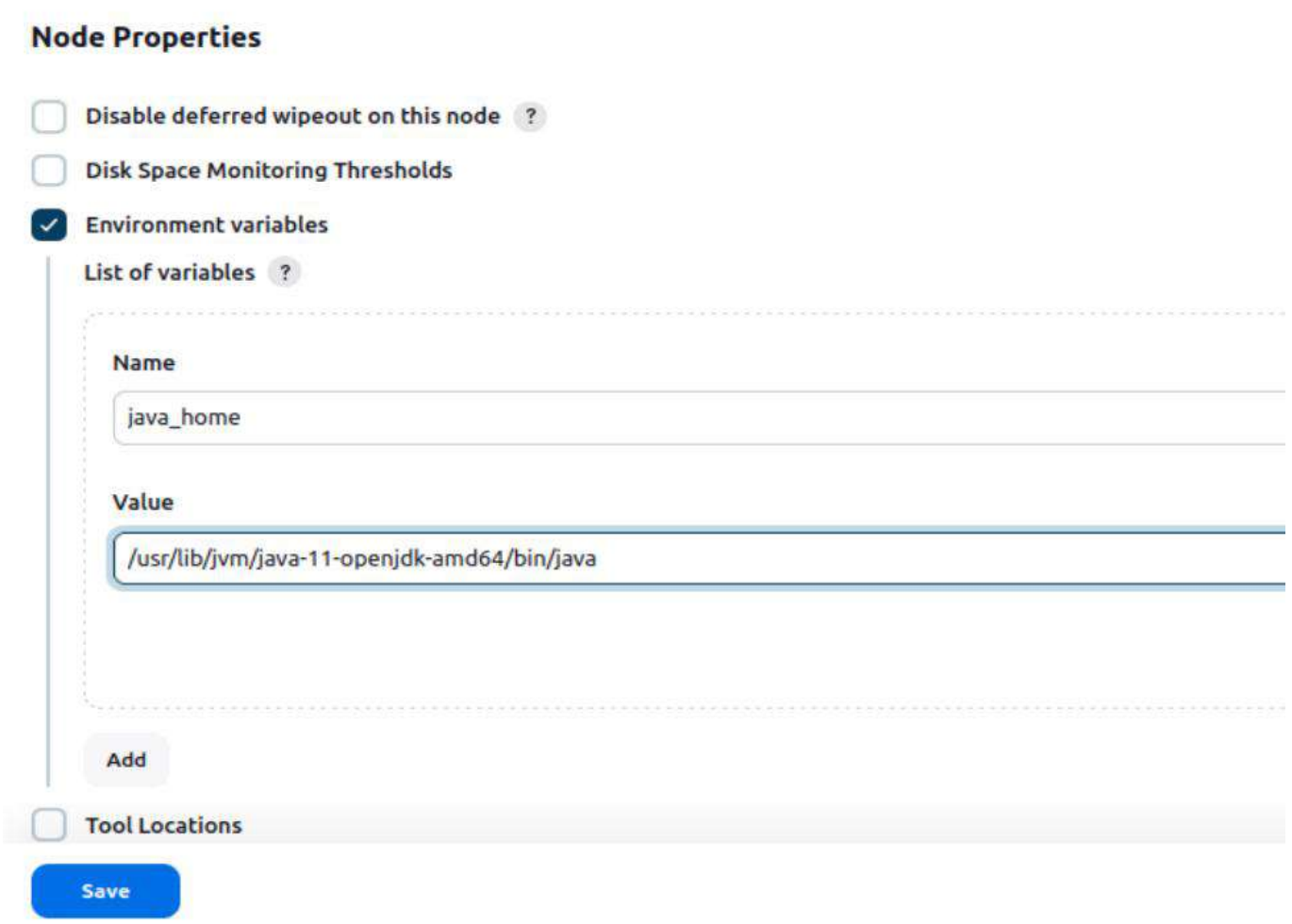
#find / -type f -name java

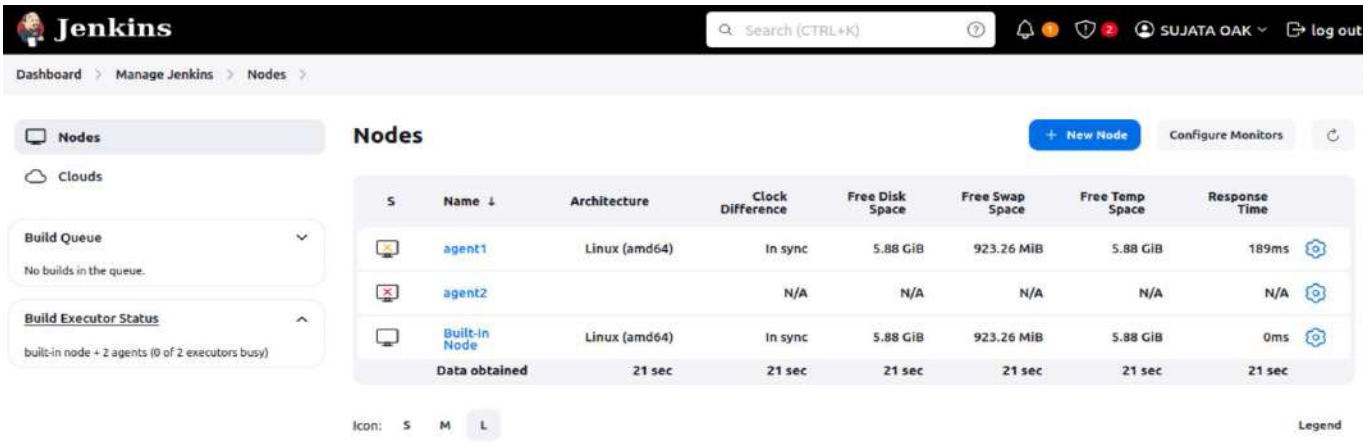
```
sujata@Ubuntu:~/Desktop/JENKINS_LAB$ su root
Password:
root@Ubuntu:/home/sujata/Desktop/JENKINS_LAB# find / -type f -name java
```

```
/usr/lib/jvm/java-11-openjdk-amd64/bin/java
```



Under 'Node Properties', provide jdk path.


Node Properties
Disable deferred wipeout on this node
Disk Space Monitoring Thresholds
Environment variables
List of variables
Name
java_home
Value
/usr/lib/jvm/java-11-openjdk-amd64/bin/java
Add
Tool Locations
Save


Jenkins
Search (CTRL+K)
SUJATA OAK
log out
Dashboard
Manage Jenkins
Nodes
Nodes
Clouds
Build Queue
No builds in the queue.
Build Executor Status
built-in node + 2 agents (0 of 2 executors busy)
New Node
Configure Monitors
S
Name
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
agent1
Linux (amd64)
In sync
5.88 GiB
923.26 MiB
5.88 GiB
189ms
agent2
N/A
N/A
N/A
N/A
N/A
Built-In Node
Linux (amd64)
In sync
5.88 GiB
923.26 MiB
5.88 GiB
0ms
Data obtained
21 sec
21 sec
21 sec
21 sec
21 sec
21 sec
Icon:
S
M
L
Legend

STEP4: On click of 'Save' will display the below page with error message. Here Jenkins connect with Slave node using Java Web Start and it needs a port to establish the connection.

To configure JNLP port in global security. Now goto Manage Jenkins -> Security

This port has to be allowed to access across firewall, so from Master terminal run the below command,

sudo ufw allow 50000/tcp

This command will allow port 50000 to listen for request.

```
root@Ubuntu:/home/sujata/Desktop/JENKINS_LAB# sudo ufw allow 50000/tcp
Rule added
Rule added (v6)
```

STEP5: Again coming back to Jenkins and navigate to Nodes -> agent2 which will display two ways to connect with Agent node.

To establish connection, run the below command

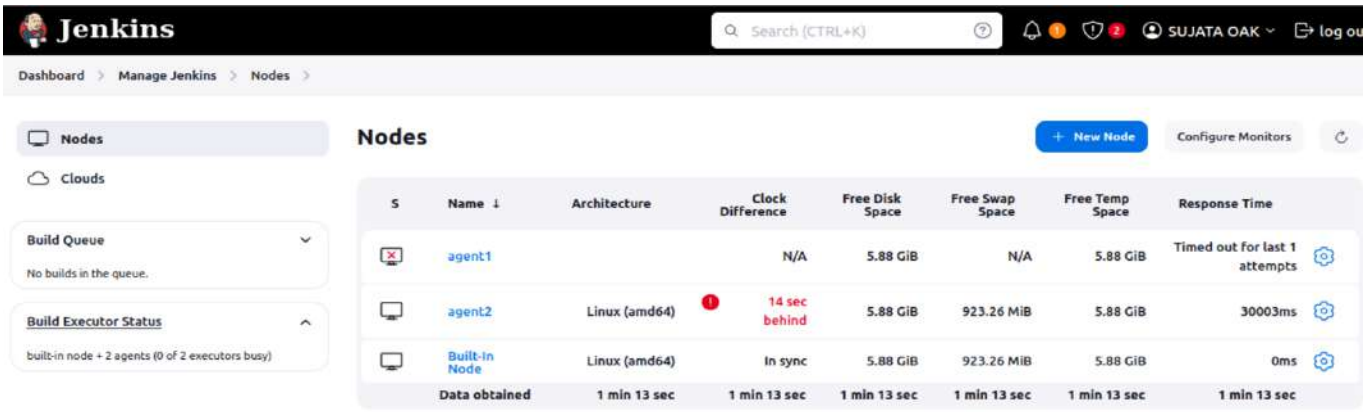
```
root@Ubuntu:/home/sujata/Desktop/JENKINS_LAB# curl -sO http://127.0.0.1:8080/jnlpJars/agent.jar
```

```
root@Ubuntu:/home/sujata/Desktop/JENKINS_LAB# java -jar agent.jar -url http://127.0.0.1:8080/ -secret cacd8d769874ea4f1a2a28392
ffe62d08addd0eeb0ea463cced99fa1f707fad0 -name agent2 -workDir "/home/sujata/Desktop/JENKINS_LAB"
```

OUTPUT:

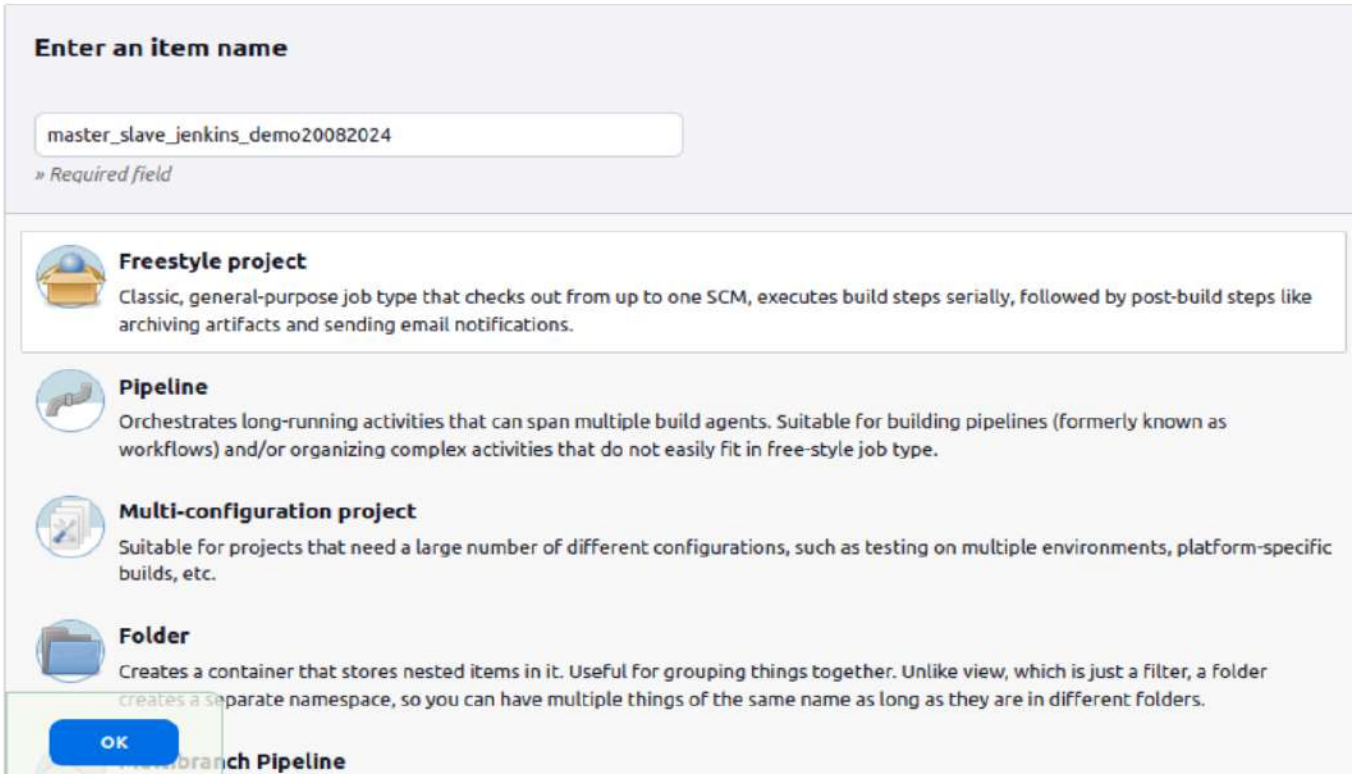
```
INFO: Both error and output logs will be printed to /home/sujata/Desktop/JENKINS_LAB/remoting
Aug 20, 2024 10:24:53 AM hudson.remoting.Launcher createEngine
INFO: Setting up agent: agent2
Aug 20, 2024 10:24:53 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3206.vb_15dcf73f6a_9
Aug 20, 2024 10:24:53 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/sujata/Desktop/JENKINS_LAB/remoting as a remoting work directory
Aug 20, 2024 10:24:54 AM hudson.remoting.Launcher$CuiListener status
INFO: Locating server among [http://127.0.0.1:8080/]
Aug 20, 2024 10:24:54 AM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve
INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]
Aug 20, 2024 10:24:54 AM hudson.remoting.Launcher$CuiListener status
INFO: Agent discovery successful
  Agent address: 127.0.0.1
  Agent port:    50000
  Identity:      80:21:52:35:ca:60:ed:97:f1:2a:65:7a:50:b9:27:77
Aug 20, 2024 10:24:54 AM hudson.remoting.Launcher$CuiListener status
INFO: Handshaking
Aug 20, 2024 10:24:54 AM hudson.remoting.Launcher$CuiListener status
INFO: Connecting to 127.0.0.1:50000
Aug 20, 2024 10:24:54 AM hudson.remoting.Launcher$CuiListener status
INFO: Server reports protocol JNLP4-connect-proxy not supported, skipping
Aug 20, 2024 10:24:54 AM hudson.remoting.Launcher$CuiListener status
INFO: Trying protocol: JNLP4-connect
Aug 20, 2024 10:24:54 AM org.jenkinsci.remoting.protocol.impl.BIONetworkLayer$Reader run
INFO: Waiting for ProtocolStack to start.
Aug 20, 2024 10:24:54 AM hudson.remoting.Launcher$CuiListener status
INFO: Remote identity confirmed: 80:21:52:35:ca:60:ed:97:f1:2a:65:7a:50:b9:27:77
Aug 20, 2024 10:24:54 AM hudson.remoting.Launcher$CuiListener status
INFO: Connected
```

This will establish connection with the configured Slave node.

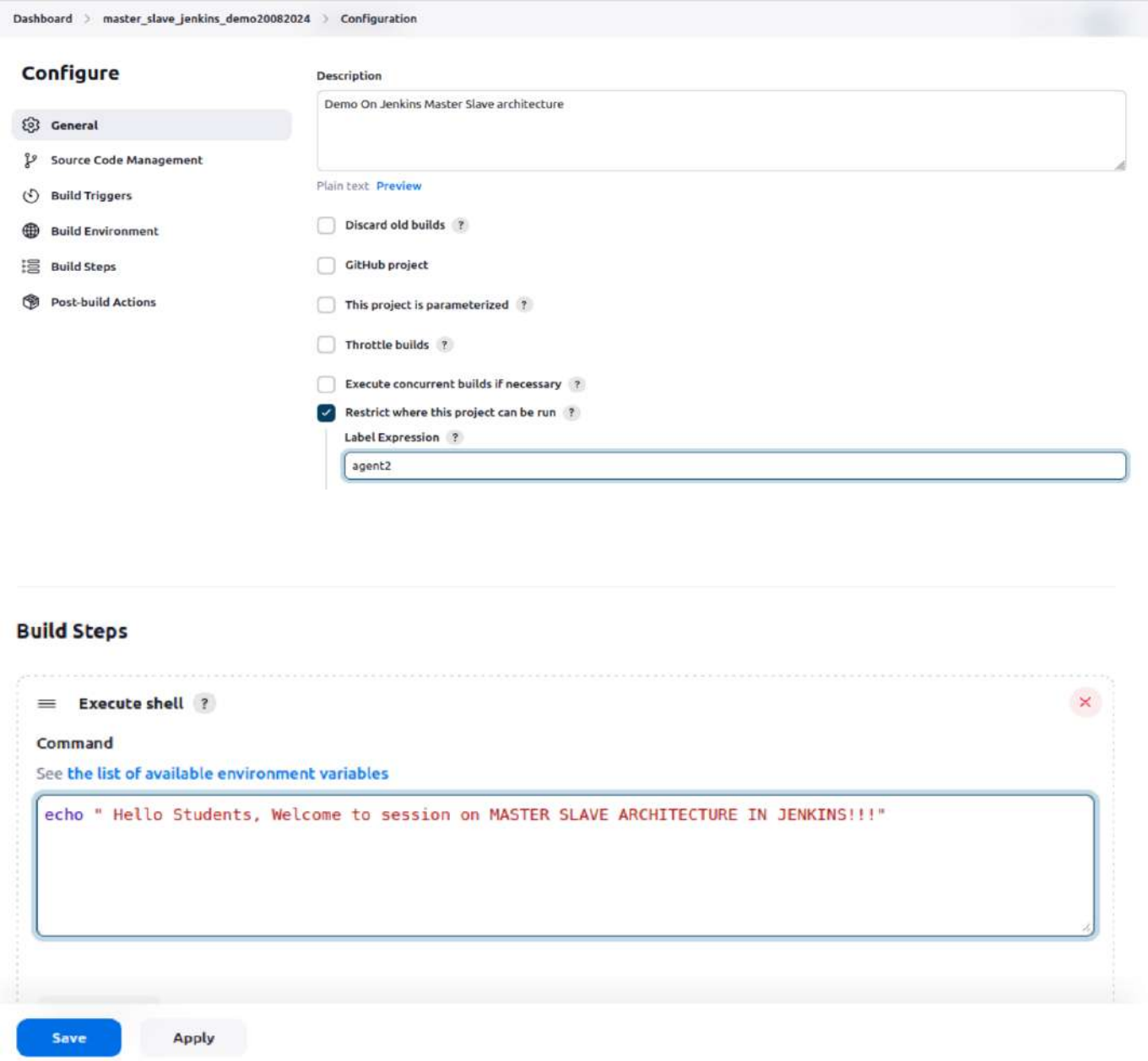


Now Jenkins Slave node is ready to run any job. This node's label name should be mentioned in the corresponding Job configuration as below:

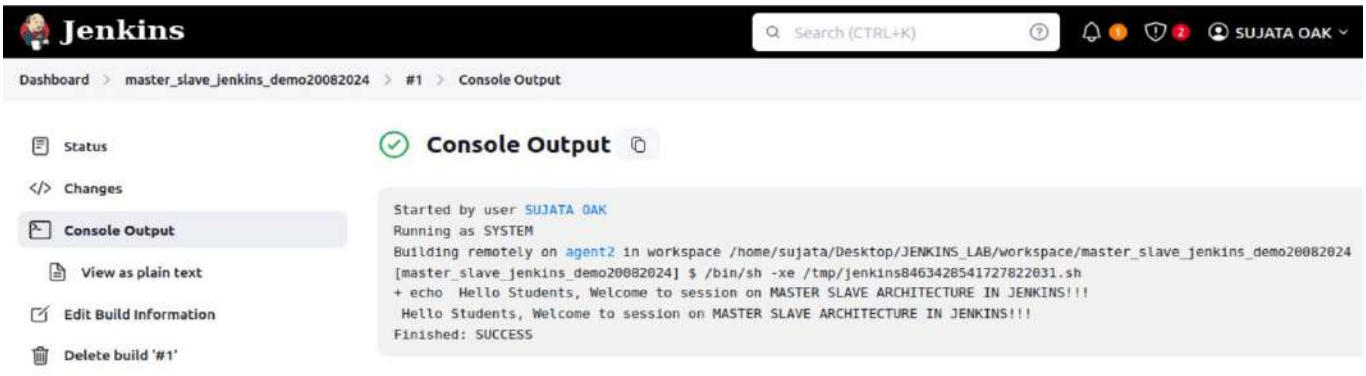
STEP 6: Create a New Job in Jenkins dashboard



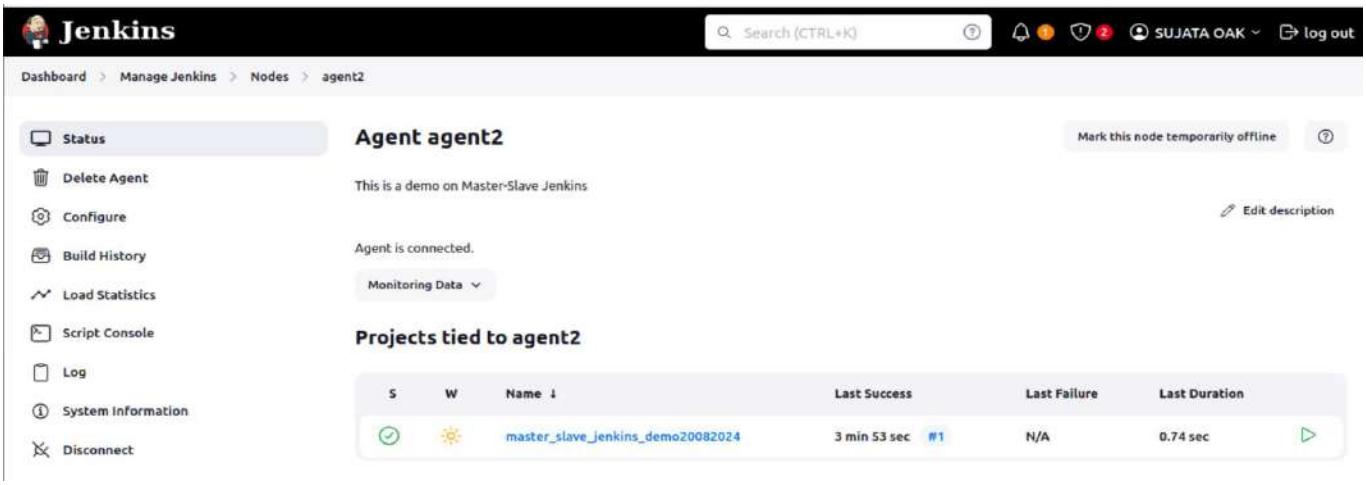
STEP 7: Configure the page with following:



Click on Build-Now, Console Output



STEP 8: Goto Jenkins Dashboard->Manage Jenkins->Nodes->agent2



Conclusion: Jenkins Master–Slave (Controller–Agent) architecture allows scalable, parallel, and environment-specific builds. It is essential for real-world CI/CD pipelines where multiple teams and platforms are involved.