Jenkins Project

In this project we will deploy a Java App on a separated Linux Server using tool **“Jenkins” .**

- **Tools we need for this project.**

1. **Jenkins :** (Jenkins is a Java-based open-source automation platform with plugins designed for continuous integration. It is used to continually create and test software projects, making it easier for developers and DevOps engineers to integrate changes to the project and for consumers to get a new build. It also enables you to release your software continuously by interacting with various testing and deployment methods.)

2. **Maven :** (Maven is a popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project management. Maven is written in Java and is used to build projects written in C#, Scala, Ruby, Java, etc.)

   Note: in this project we will install “**Maven**” as **Jenkins Plugin,** then we will integrate it “**Maven**” with **Jenkins,** so we can use it to build and deploy our App**.**

3. **VMware Workstation :** (VMware Workstation is a Desktop Hypervisor products which let users run virtual machines, containers, and Kubernetes clusters. VMware Workstation is a virtual machine software that is used for x86 and x86-64 computers to run multiple operating systems over a single physical host computer. Each virtual machine can run a single instance of any operating system (Microsoft, Linux, etc.) simultaneously. VMware Workstation strongly supports hardware compatibility and works as a bridge between the host and virtual machine for all kinds of hardware resources including hard disks, USB devices and CD-ROMs. All device drivers are installed via the host machine.)

- **Steps**:

1. First we will create 2 Vms using **VMware Workstation** , one as a Master node for installing "**Jenkins"**, and the other as a Slave node for the App.
2. Then on the Master node we will install and configure **Jenkins** to be ready to start our first job.
3. After that, we can Access **Jenkins** using any web browser, to install **Maven Plugin** and integrate it with **Jenkins,** and create and configure our Slave node and joint it to Master node.
4. At end we will create a job to build and deploy our App on the other Vm (Slave node).

$~ Lab Setup ~$

➢ Configure and setup Slave Node (app02):

```
# Install Java and Git on Slave Node
sudo dnf update -y
sudo dnf install -y java-17-openjdk  git
# Create a user and ssh keys on slave node
sudo  useradd Jenkins-slave
sudo  su  -  Jenkins-slave
sudo  ssh-keygen
cd  ~/.ssh/
cat  id_rsa.pub >> authorized_keys
```

➢ Configure and setup Master Node (jen01):

- Install Jenkins on Master Node .

# Download **Jenkins** repo.

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

# Get GPG kay for **Jenkins**.

```
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

# Update OS with latest patches.

```
  dnf update -y
```

# Install Java.

```
  sudo yum install fontconfig java-17-openjdk
```

# Install **Jenkins** and **Git**.

```
  sudo yum install Jenkins git
  sudo systemctl daemon-reload
```

# Allowing the port 8080 to access **Jenkins.**

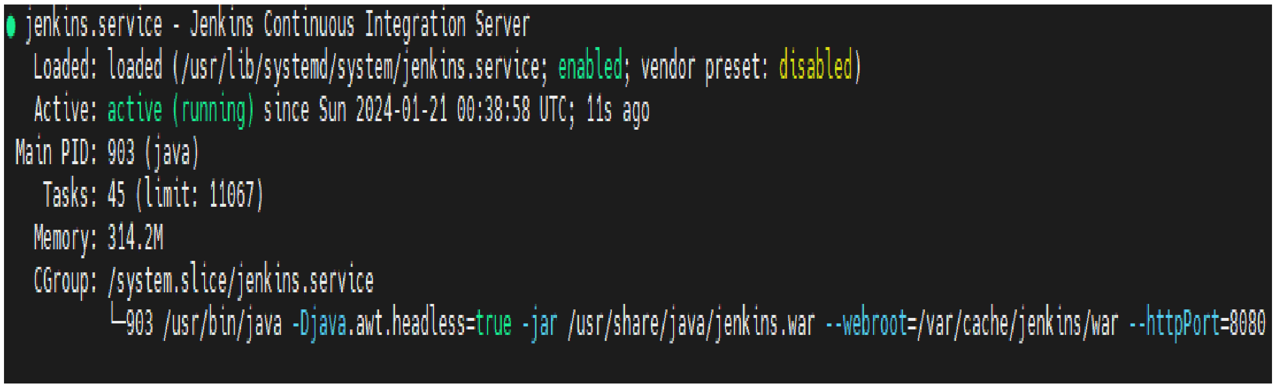
```
sudo firewall-cmd --permanent --add-port=8080/tcp
sudo firewall-cmd –reload
```

# Enable and start **Jenkins.**

```
  sudo systemctl enable jenkins
```

```
sudo systemctl start jenkins
sudo systemctl status jenkins
```

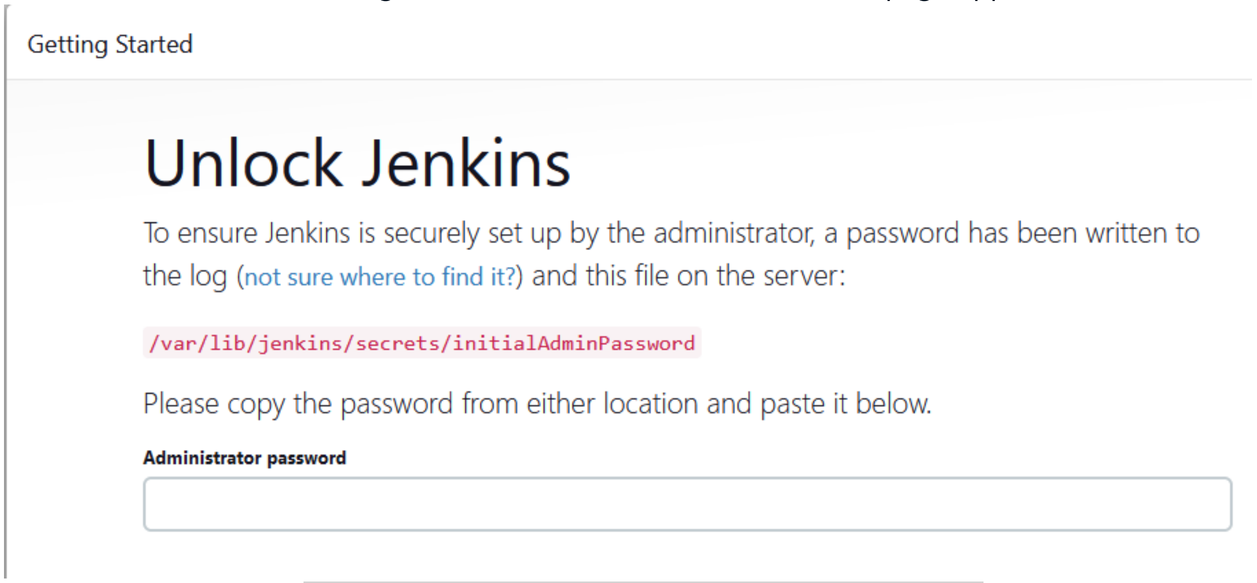
- If everything has been set up correctly, you should see an output like this:



```
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2024-01-21 00:38:58 UTC; 11s ago
 Main PID: 903 (java)
    Tasks: 45 (limit: 11067)
   Memory: 314.2M
   CGroup: /system.slice/jenkins.service
           └─903 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
```

# **Unlocking Jenkins** : (When you first access a new Jenkins instance, you are asked to unlock it using an automatically-generated password).

1) Browse to http:// Jenkins-Machine-IP:8080 (or whichever port you configured for Jenkins when installing it) and wait until the **Unlock Jenkins** page appears.



Getting Started

Unlock Jenkins

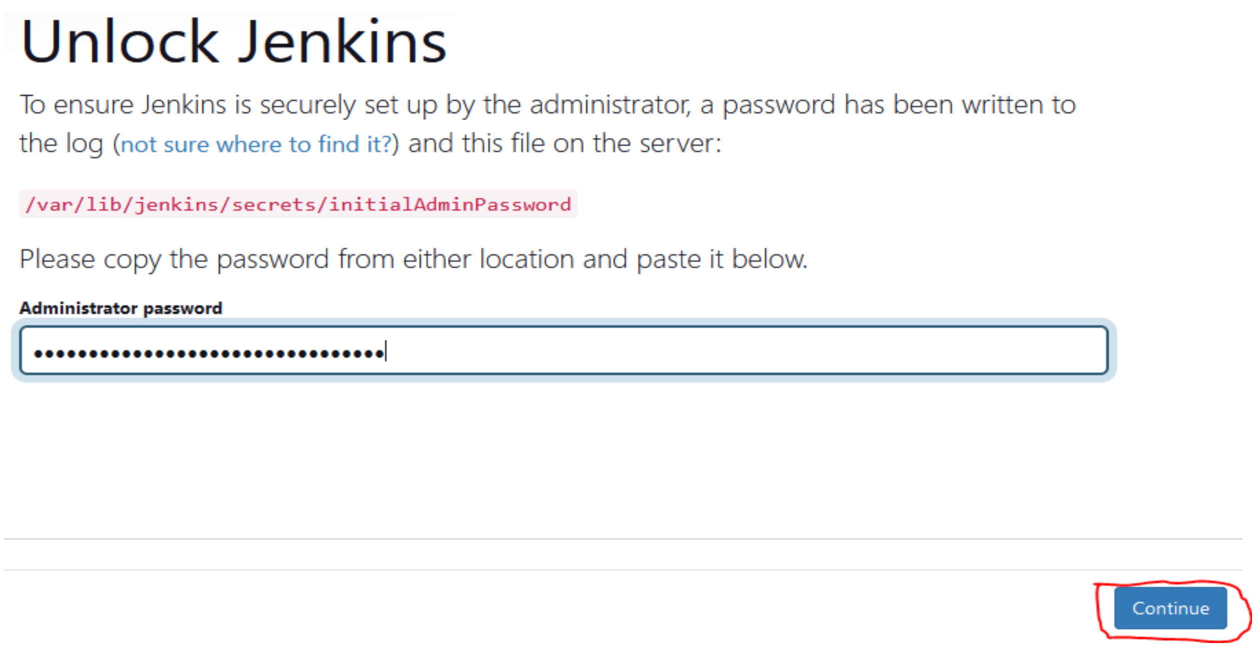
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

2) Run command: `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` to print the password.
3) On the **Unlock Jenkins** page, paste this password into the **Administrator password** field and click **Continue**.



Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

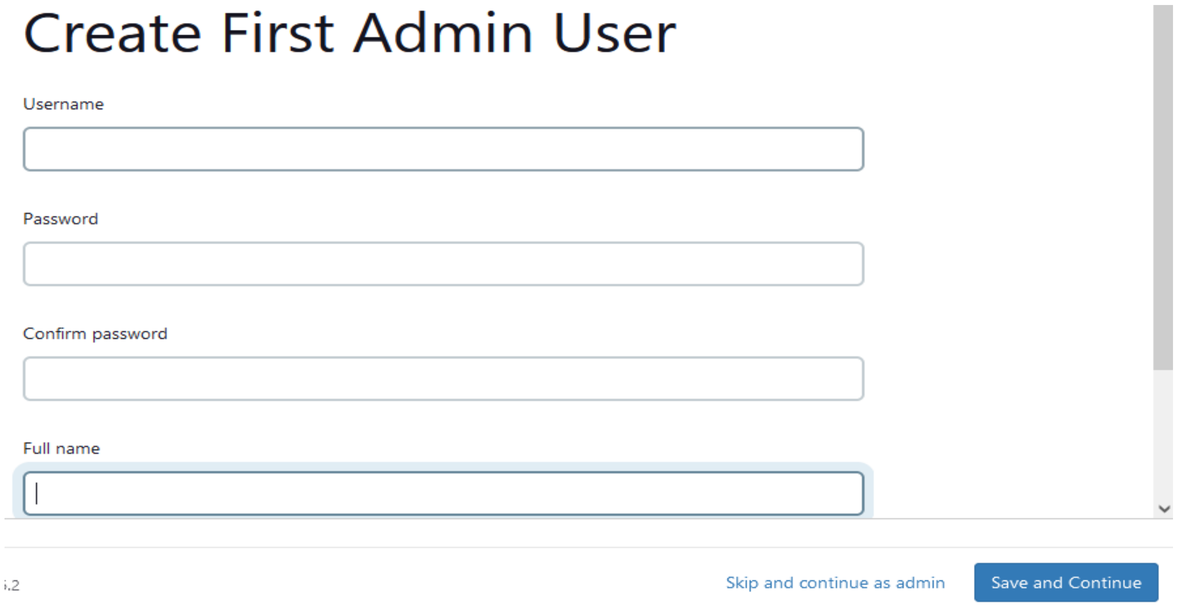
# **Customizing Jenkins with plugins** : (After unlocking Jenkins, the Customize Jenkins page appears. Here you can install any number of useful plugins as part of your initial setup.)

Click one of the two options shown:

- **Install suggested plugins** - to install the recommended set of plugins, which are based on most common use cases.
- **Select plugins to install** - to choose which set of plugins to initially install. When you first access the plugin selection page, the suggested plugins are selected by default.

# Creating the first administrator user :

1) When the **Create First Admin User** page appears, specify the details for your administrator user in the respective fields and click **Save and Finish.**


Create First Admin User

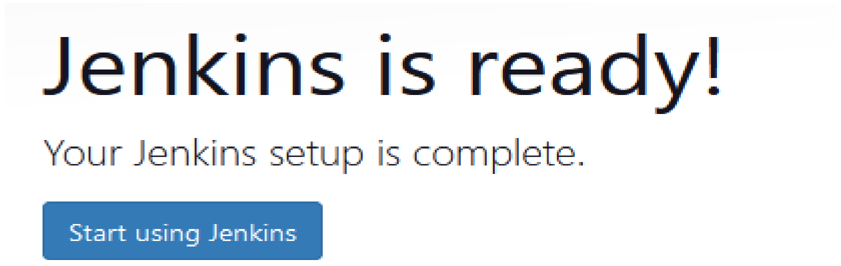
Username

Password

Confirm password

Full name

2) When the **Jenkins is ready** page appears, click **Start using Jenkins**.


Jenkins is ready!

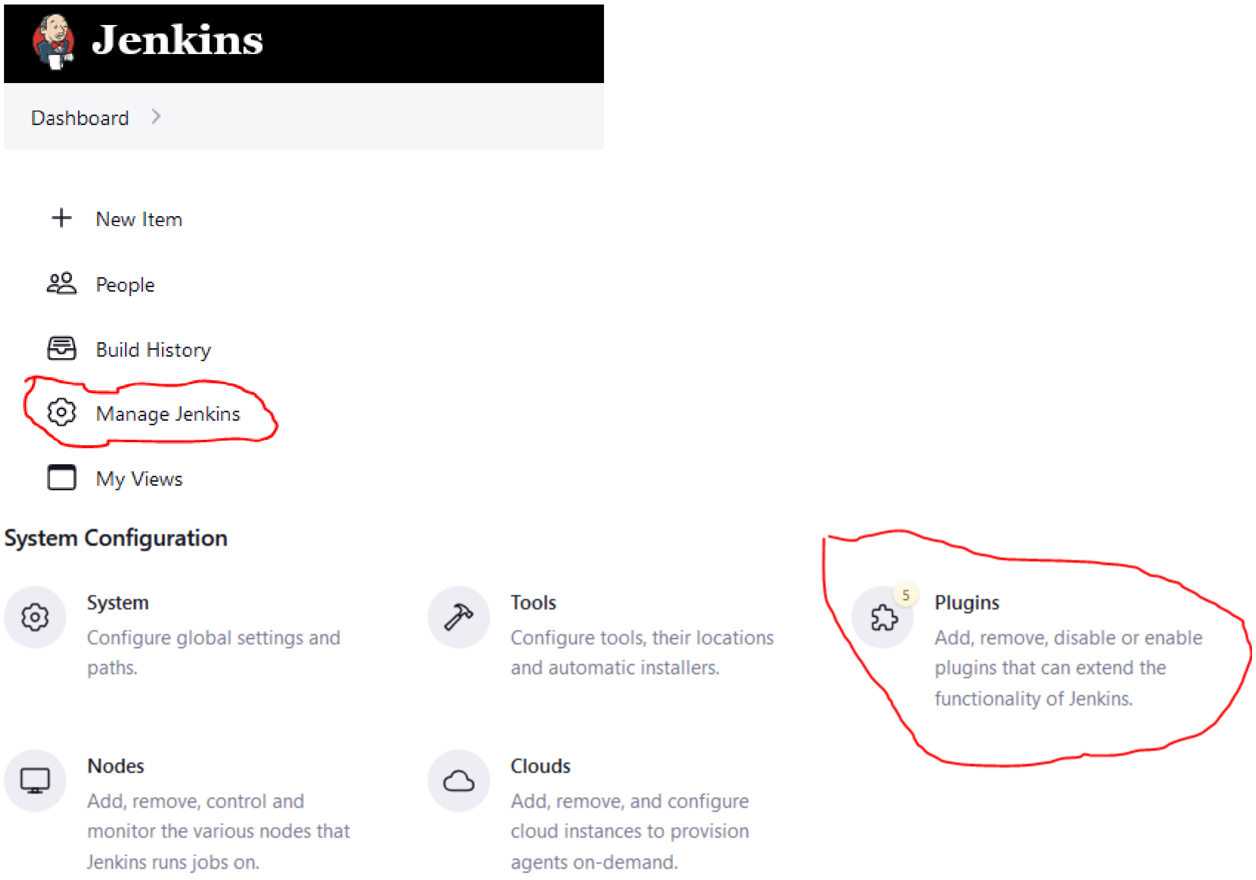
Your Jenkins setup is complete.

3) If required, log in to Jenkins with the credentials of the user you just created and you are ready to start using **Jenkins**.

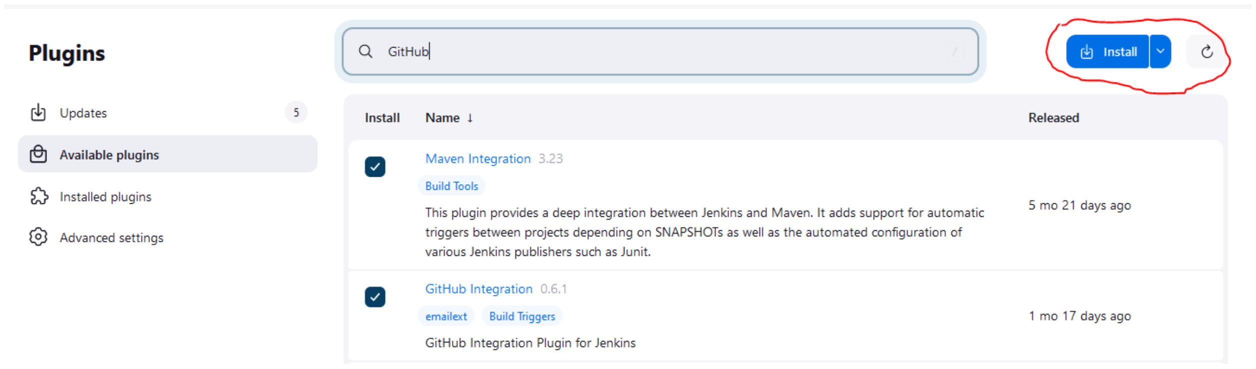
➢ Install Maven Plugin and integrate it with Jenkins:

# After we successfully unlock Jenkins web page, now we need to install maven plugin and integrate it with Jenkins. so we will follow this steps to do it successfully.

i. Navigate to **Dashboard->> Manage Jenkins ->> Plugin**

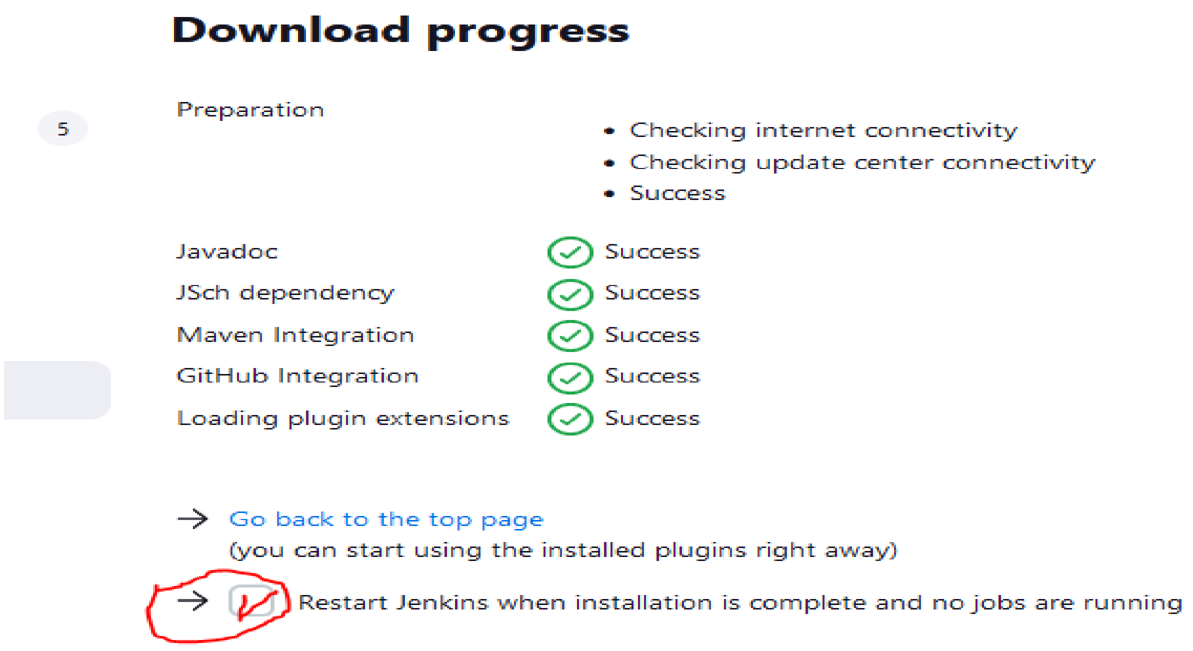

Jenkins
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
System Configuration
System
Configure global settings and paths.
Tools
Configure tools, their locations and automatic installers.
Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

ii. Now search for **"GitHub Integration Plugin , Maven Integration Plugin "** and click on **Install**.



Install	Name ↓	Released
✓	Maven Integration 3.23 Build Tools This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.	5 mo 21 days ago
✓	GitHub Integration 0.6.1 emailext Build Triggers GitHub Integration Plugin for Jenkins	1 mo 17 days ago

iii. After plugins successfully installed we must restart Jenkins.



Download progress

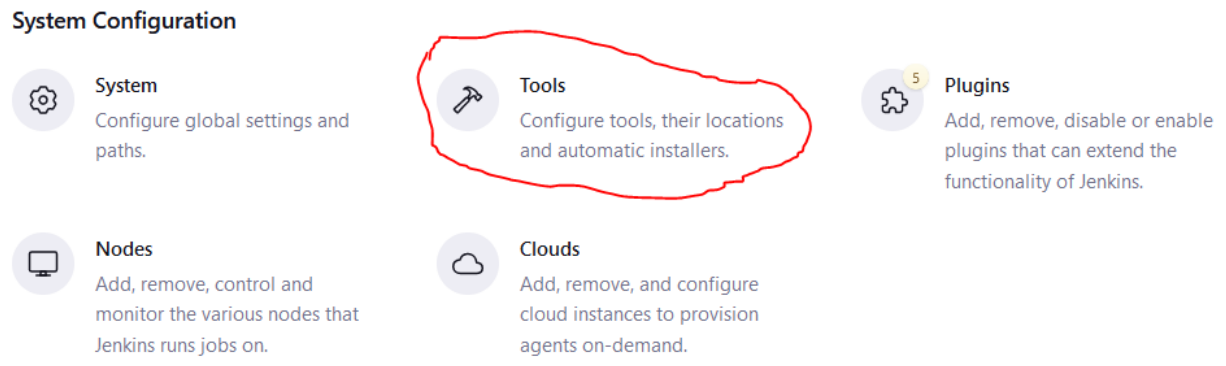
Preparation
- Checking internet connectivity
- Checking update center connectivity
- Success

Javadoc	✓ Success
JSch dependency	✓ Success
Maven Integration	✓ Success
GitHub Integration	✓ Success
Loading plugin extensions	✓ Success

→ Go back to the top page
(you can start using the installed plugins right away)

→ [✓] Restart Jenkins when installation is complete and no jobs are running

iv. To integrate Maven with Jenkins, Navigate to **Dashboard->> Manage Jenkins ->> Tools .**



System Configuration

System
Configure global settings and paths.

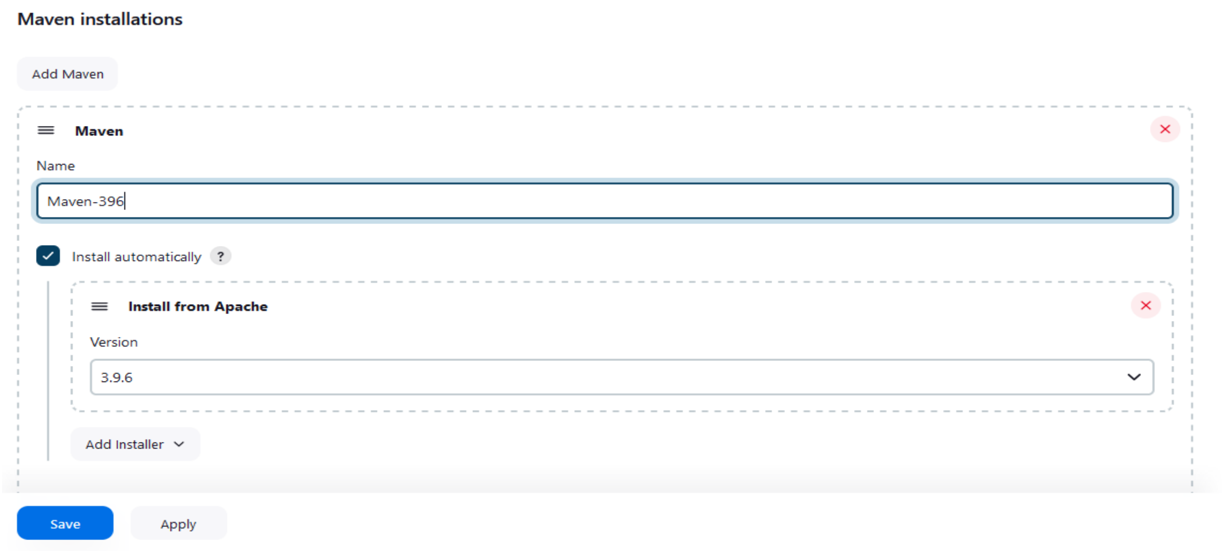
Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

v. click on **Maven->> Add Maven**.



Maven installations

Maven

Name: Maven-396

Install automatically [✓]

Install from Apache

Version: 3.9.6

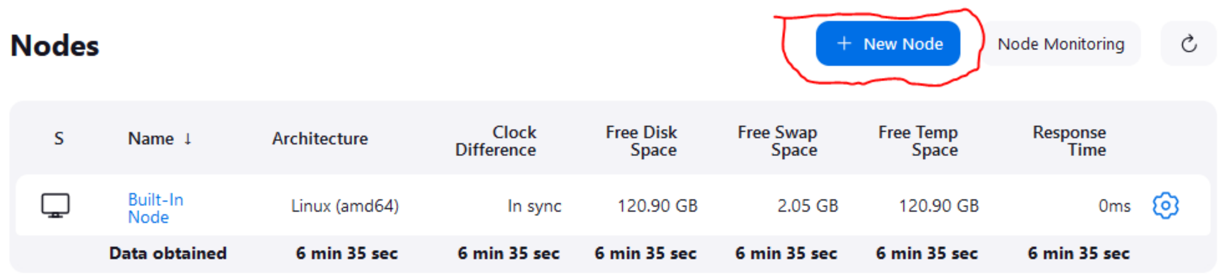
vi. Click on **apply** and **save.**

➢ Create Slave node and join it to Master.

# To join the Jenkins slave node to Jenkins Master, perform below steps:

I. Navigate to **Dashboard->> Manage Jenkins ->> Nodes**

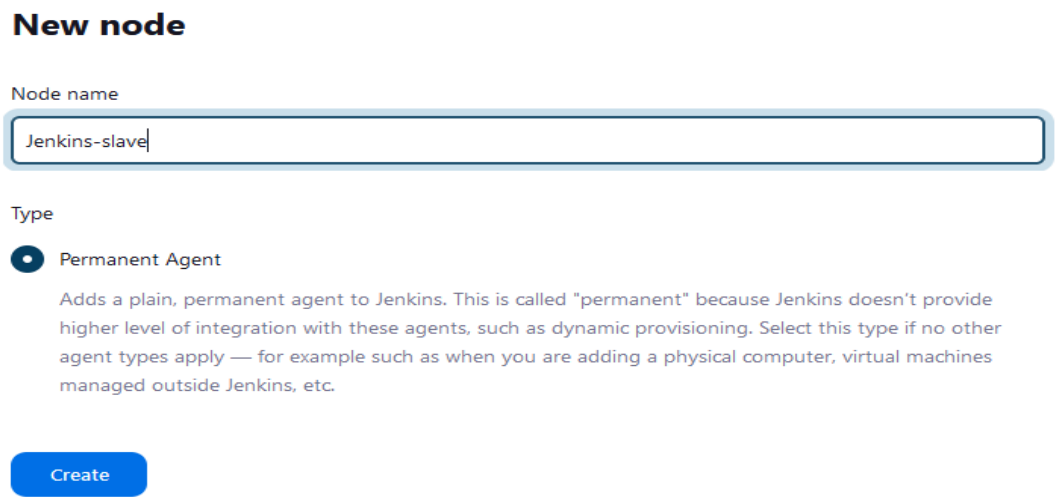
II. Click on **New Node.**



Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	
	Built-In Node	Linux (amd64)	In sync	120.90 GB	2.05 GB	120.90 GB	0ms	
	Data obtained	6 min 35 sec	6 min 35 sec	6 min 35 sec	6 min 35 sec	6 min 35 sec	6 min 35 sec	

III. Enter Node name and select **Permanent Agent.**



New node

Node name

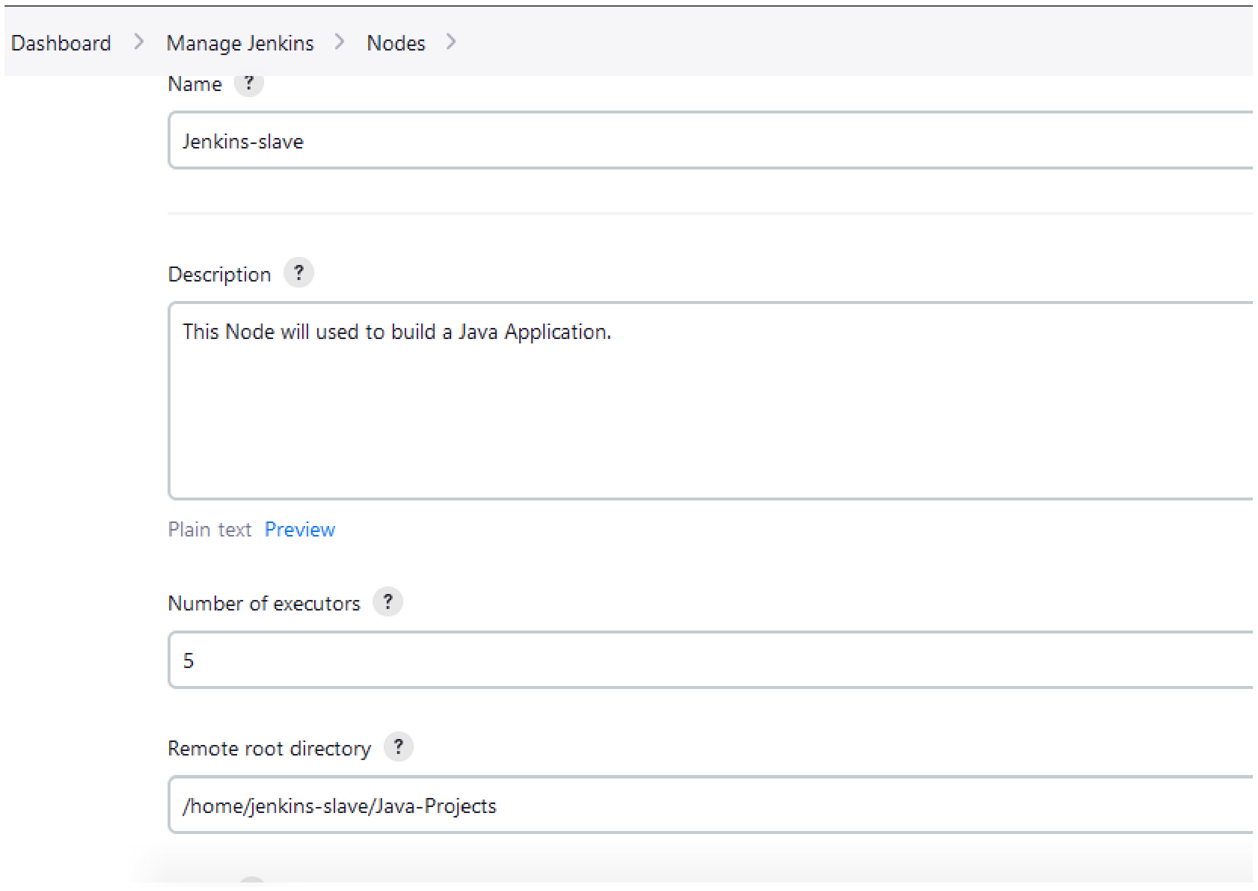
Jenkins-slave

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.

IV. Slave Node setup and configuration info.



Name

Jenkins-slave

Description

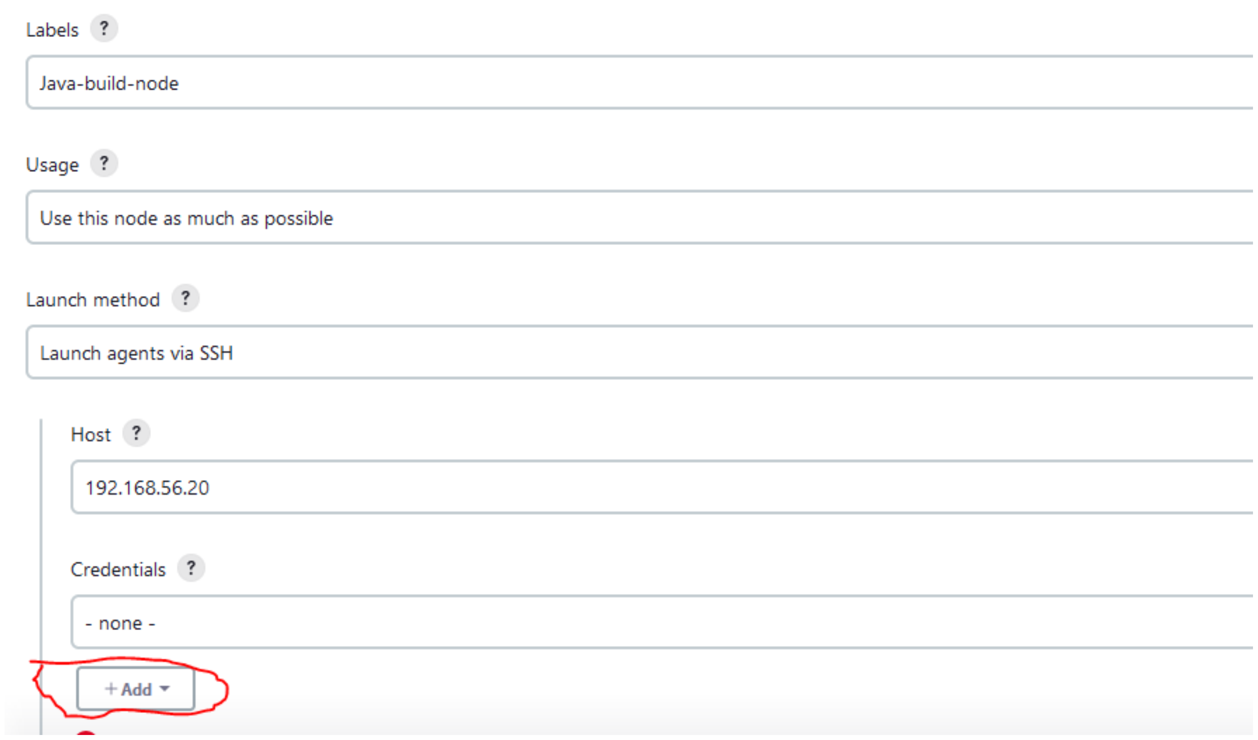
This Node will used to build a Java Application.

Number of executors

5

Remote root directory

/home/jenkins-slave/Java-Projects


Labels ?

Java-build-node

Usage ?

Use this node as much as possible

Launch method ?

Launch agents via SSH

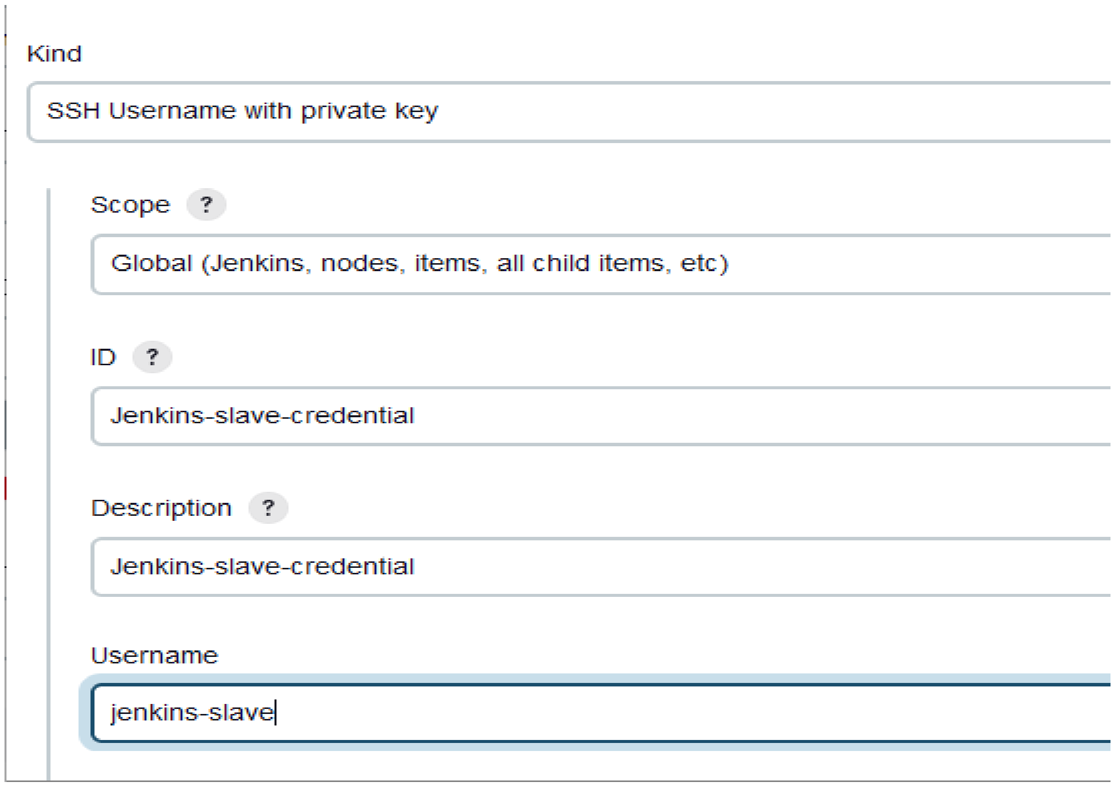
Host ?

192.168.56.20

Credentials ?

- none -

- If you choose “**Launch via SSH**” option , enter your host IP or Hostname , then click on “**Add**” to add **SSH Credential.**


Kind

SSH Username with private key

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

ID ?

Jenkins-slave-credential

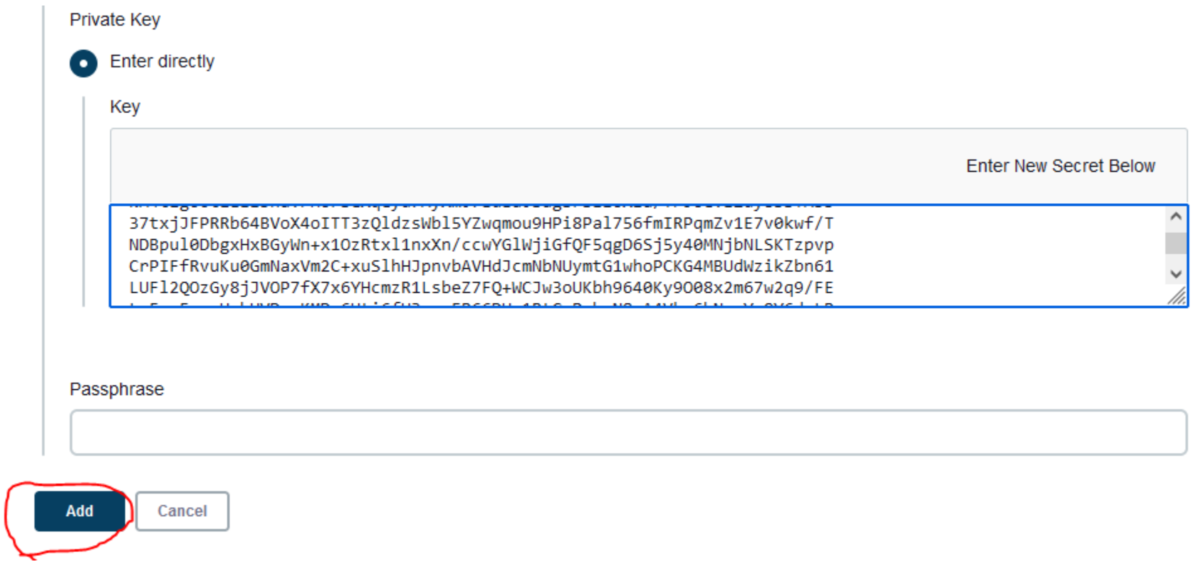
Description ?

Jenkins-slave-credential

Username

jenkins-slave

- On your Slave Node Machine , copy the private key that had been created before , use this command to print it “cat ~/.ssh/id_rsa” .



Private Key

● Enter directly

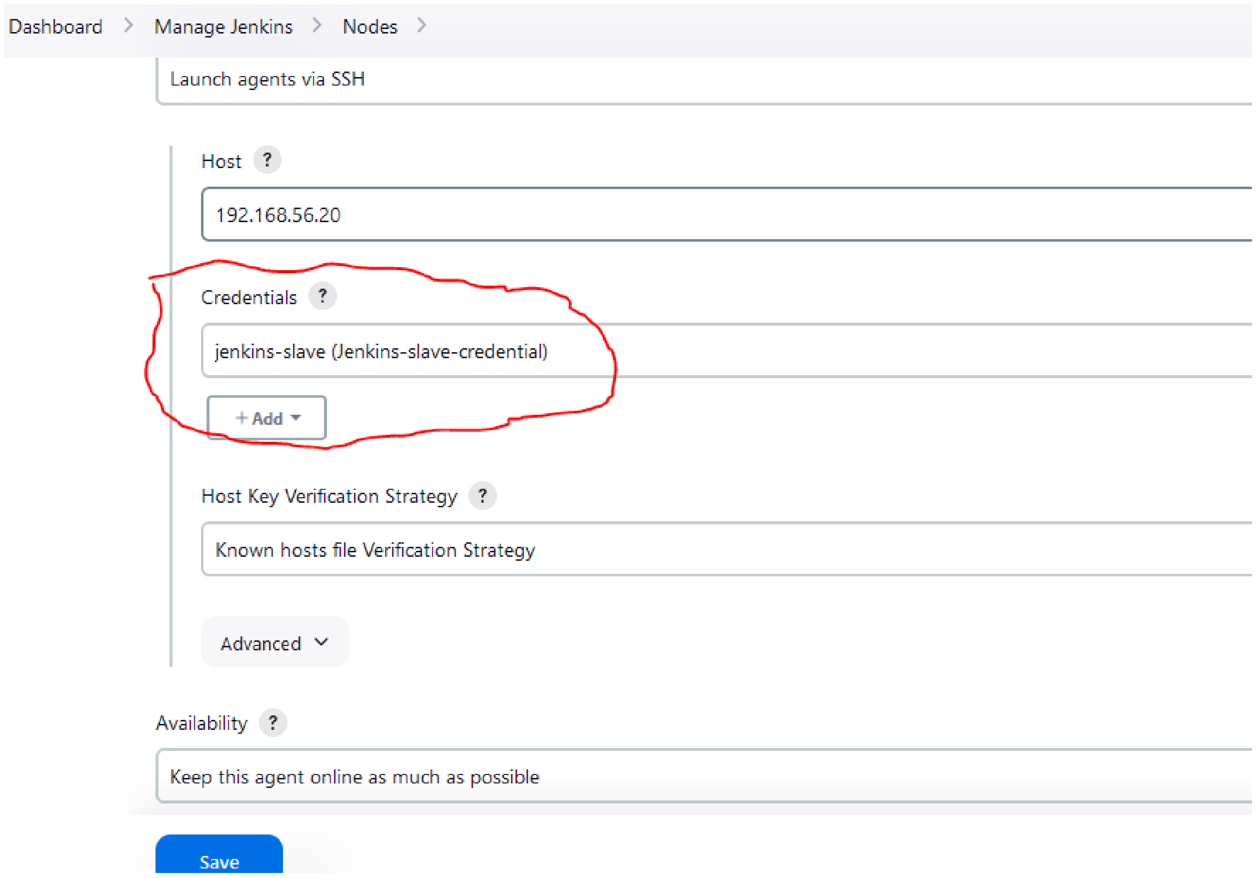
Key

Enter New Secret Below

```
37txjJFPRRb64BVoX4oITT3zQldzsWbl5YZwqmou9HPi8Pal756fmIRPqmZv1E7v0kwf/T
NDBpul0DbgxHxBGyWn+x1OzRtxl1nxXn/ccwYGlWjiGfQF5qgD6Sj5y40MNjbNLSKTzpvp
CrPIFfRvuKu0GmNaxVm2C+xuSlhHJpnvbAVHdJcmNbNUymtG1whoPCKG4MBUdWzikZbn61
LUFl2QOzGy8jJVOP7fX7x6YHcmzR1LsbeZ7FQ+WCJw3oUKbh9640Ky9O08x2m67w2q9/FE
```

Passphrase

- Paste your private key in this box and click add.



Launch agents via SSH

Host ?

192.168.56.20

Credentials ?

jenkins-slave (Jenkins-slave-credential)

Host Key Verification Strategy ?

Known hosts file Verification Strategy

Availability ?

Keep this agent online as much as possible

- After add the Credential, Click on save .

- On master node add public key of user we created on slave node machine, in /var/lib/jenkins/.ssh/known_hosts , used when selecting option "**Know hosts key strategy**".

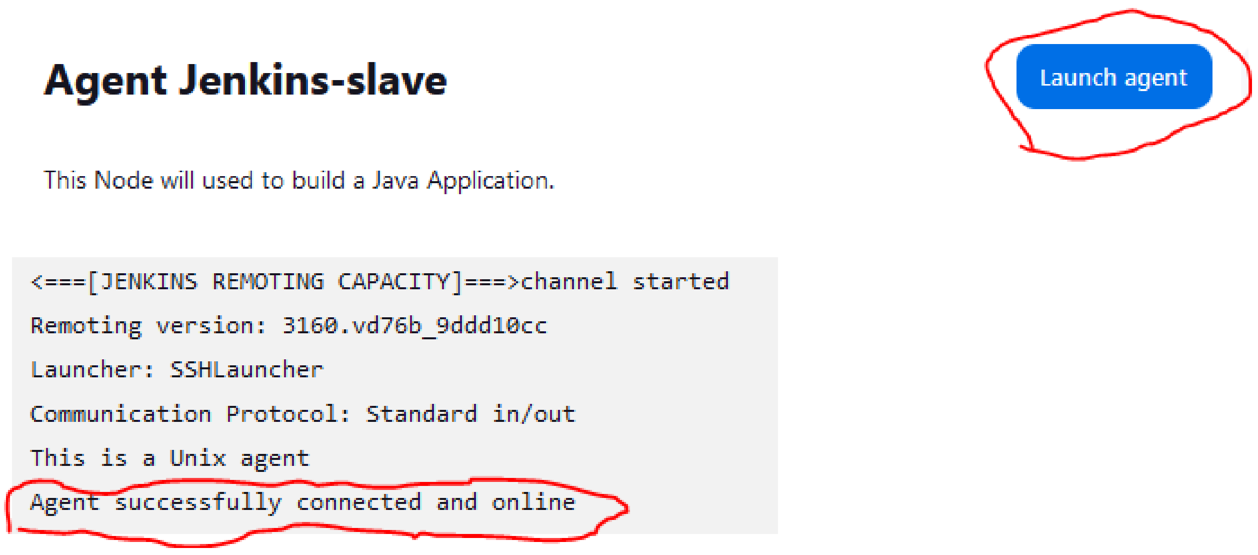
# make "**.ssh**" directory .

```
mkdir /var/lib/jenkins/.ssh
chown jenkins:jenkins /var/lib/jenkins/.ssh -R
```

# Create ssh key for our slave-node machine (app02) and add it to **known_hosts** file .

```
ssh-keyscan -H app02 >> /var/lib/jenkins/.ssh/known_hosts
```

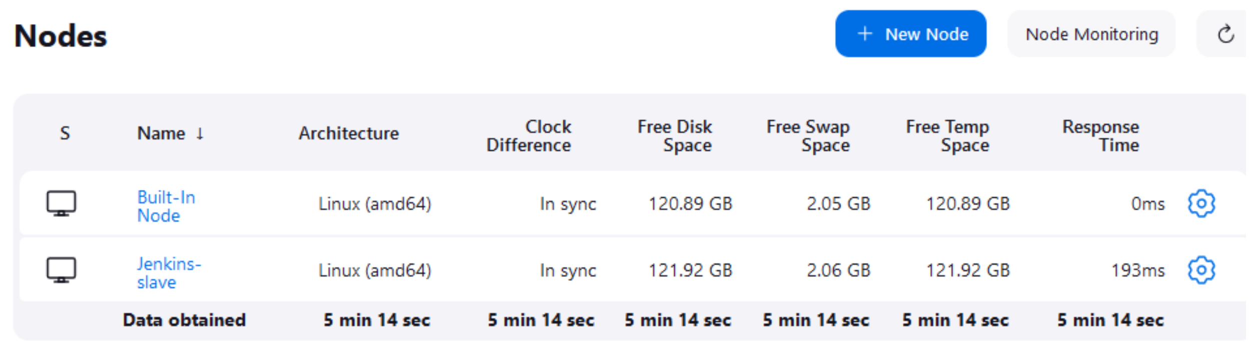
V. Now we can launch our agent node.



Agent Jenkins-slave

This Node will used to build a Java Application.

```
<===[JENKINS REMOTING CAPACITY]===>channel started
Remoting version: 3160.vd76b_9ddd10cc
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
Agent successfully connected and online
```



Nodes

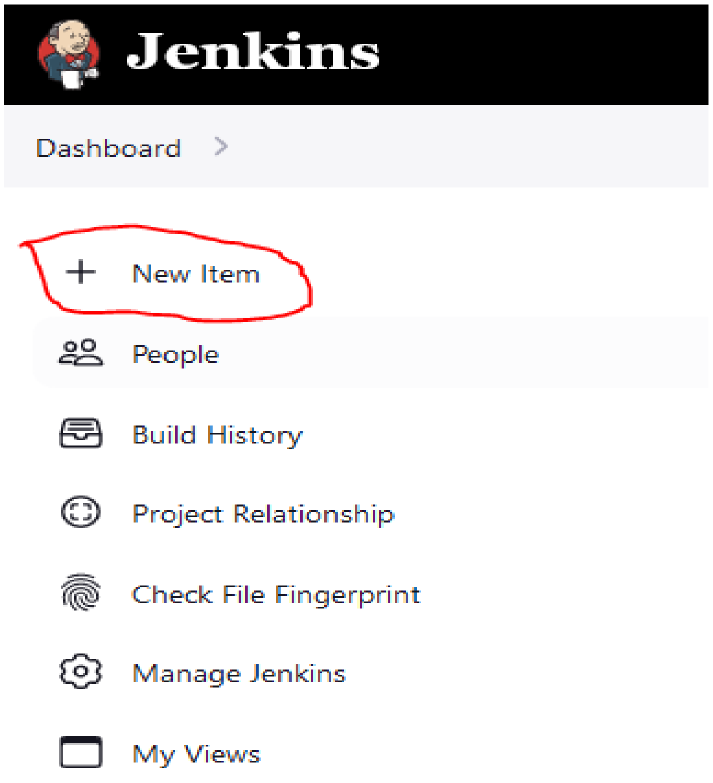
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	120.89 GB	2.05 GB	120.89 GB	0ms
	Jenkins-slave	Linux (amd64)	In sync	121.92 GB	2.06 GB	121.92 GB	193ms
	Data obtained	5 min 14 sec	5 min 14 sec	5 min 14 sec	5 min 14 sec	5 min 14 sec	5 min 14 sec

- Now we have add our Agent (**Jenkins-slave**) successfully.

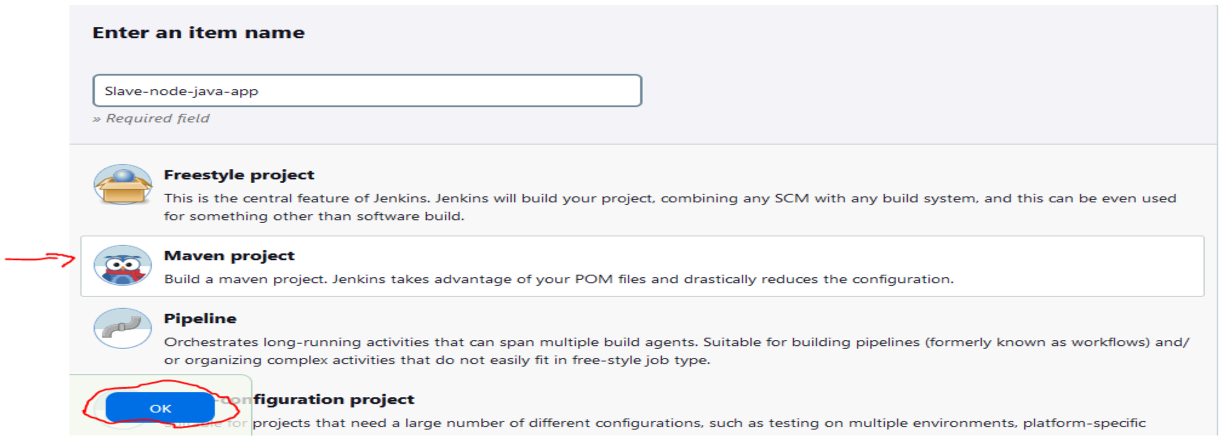
➢ Create job to test connection between two servers (Master node and Slave node).

# we will create a **Maven Project** on master node (**jen01**) to build a Java App on slave node (**app02**) .

i. Navigate to **Dashboard->> New item**



ii. Enter the job name and select **maven project**, then click **ok**.



Enter an item name

Slave-node-java-app

» *Required field*

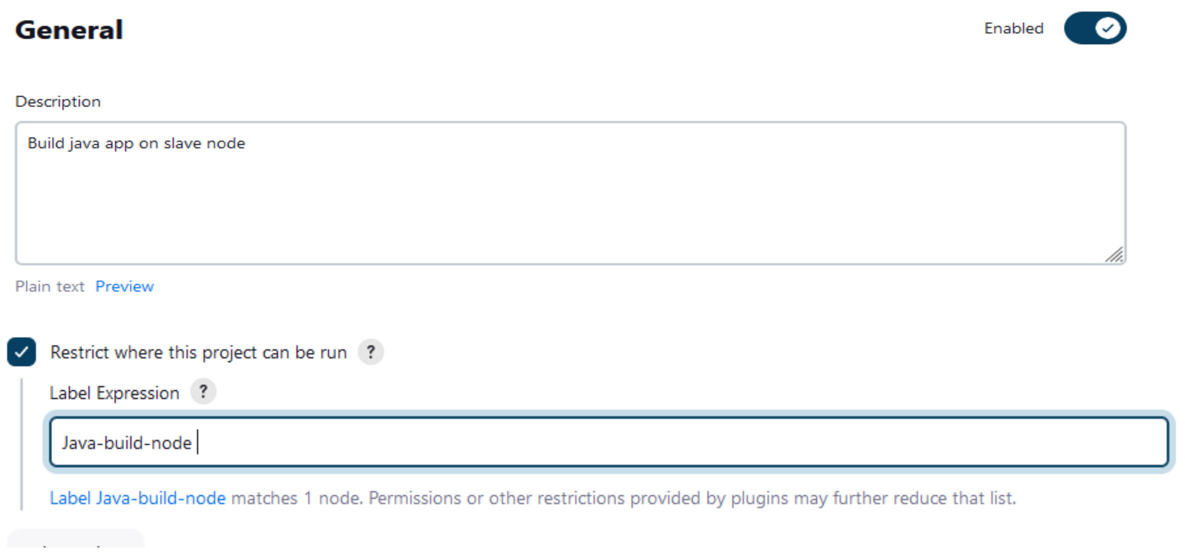
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

onfiguration project
projects that need a large number of different configurations, such as testing on multiple environments, platform-specific

iii. Enter your job details.


General

Enabled ✓

Description

Build java app on slave node

Plain text Preview

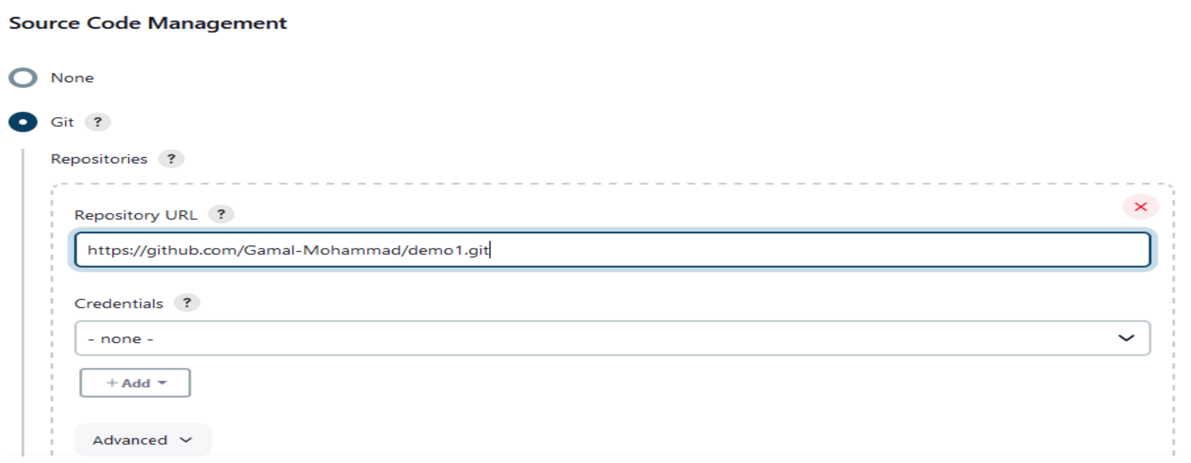
✓ Restrict where this project can be run

Label Expression

Java-build-node

Label Java-build-node matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

- Enter the label of your slave node .


Source Code Management

○ None

● Git

Repositories

Repository URL

https://github.com/Gamal-Mohammad/demo1.git

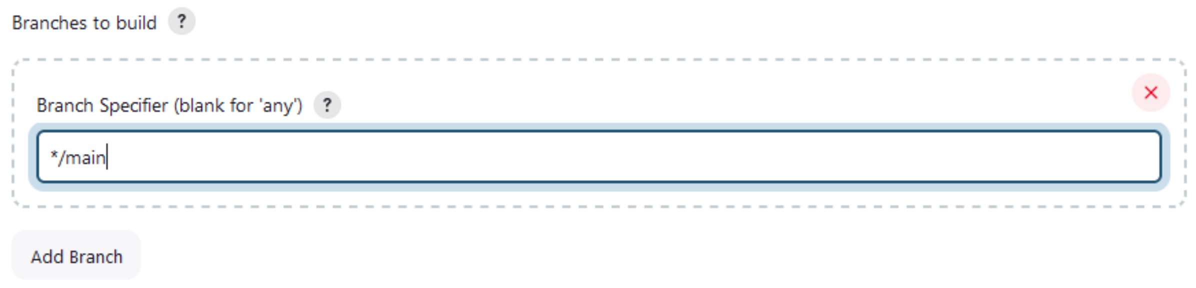
Credentials

- none -

+ Add

Advanced

- If your Github account not public, you can add your private account credentials.

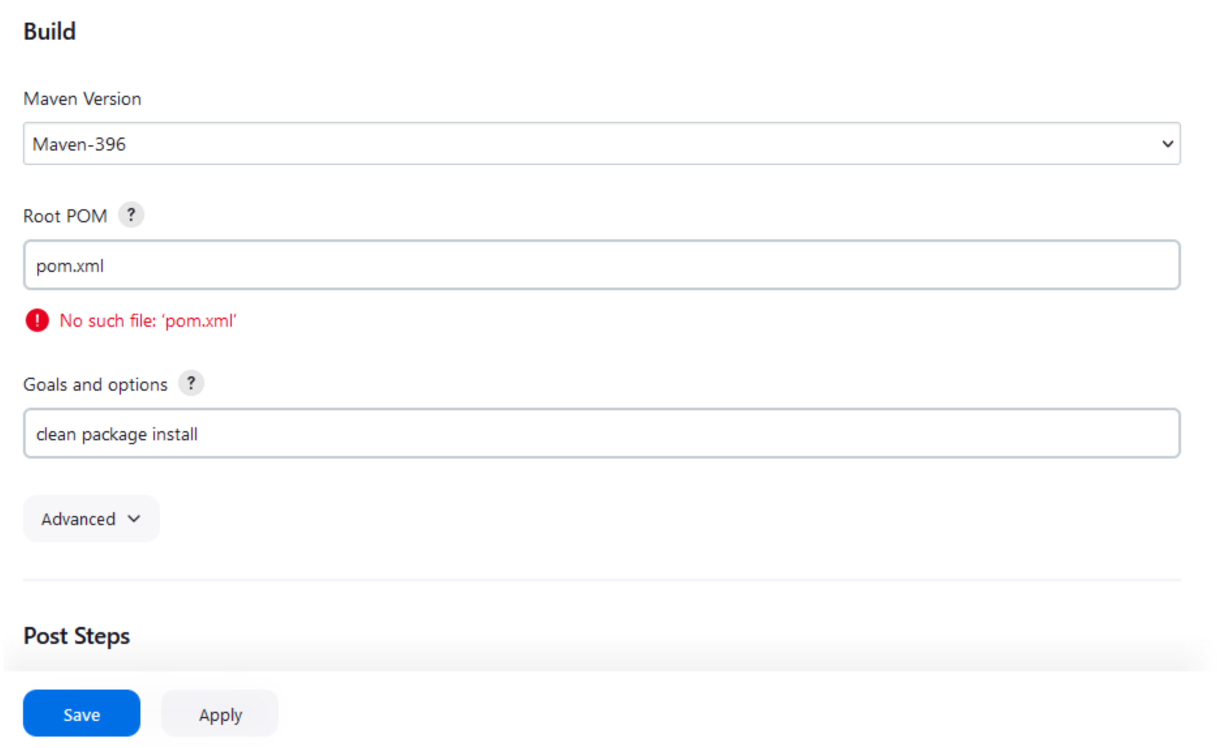

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch

- Enter your project branch in Github.



Build

Maven Version

Maven-396

Root POM

pom.xml

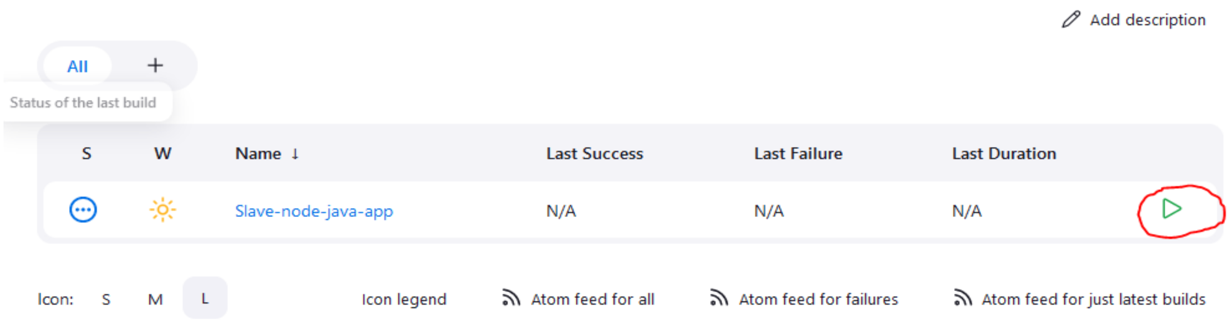
No such file: 'pom.xml'

Goals and options

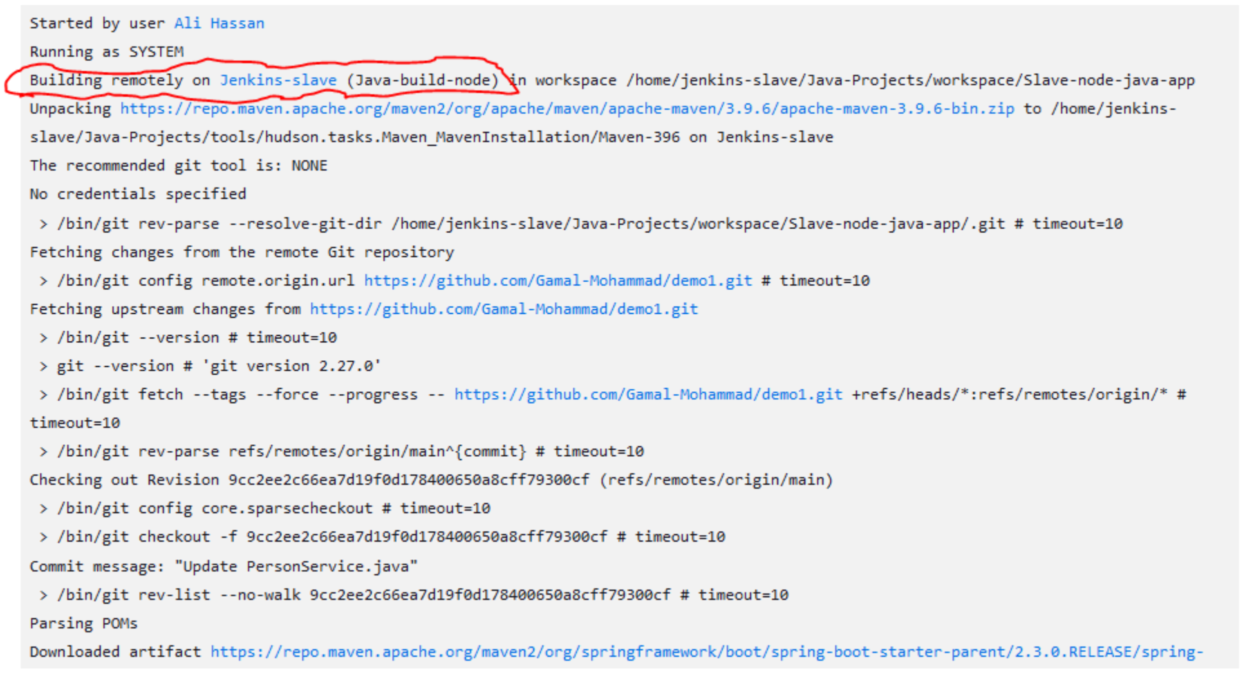
clean package install

- Click **Apply** and **Save**

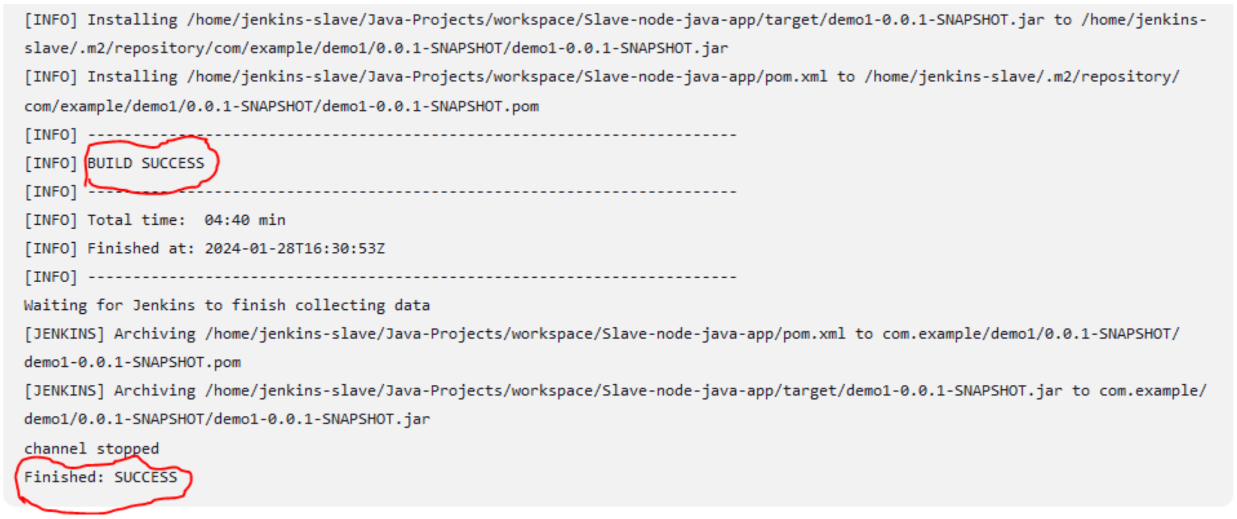
- Now we have our job ready to build.



S	W	Name ↓	Last Success	Last Failure	Last Duration
		Slave-node-java-app	N/A	N/A	N/A

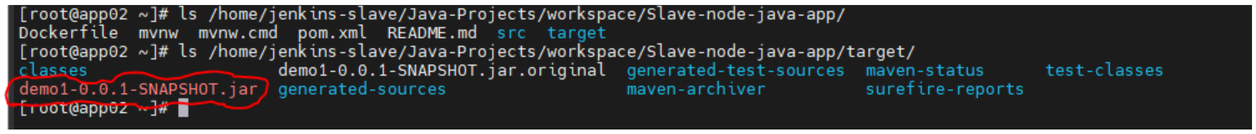


```
Started by user Ali Hassan
Running as SYSTEM
Building remotely on Jenkins-slave (Java-build-node) in workspace /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app
Unpacking https://repo.maven.apache.org/maven2/org/apache/maven/apache-maven/3.9.6/apache-maven-3.9.6-bin.zip to /home/jenkins-
slave/Java-Projects/tools/hudson.tasks.Maven_MavenInstallation/Maven-396 on Jenkins-slave
The recommended git tool is: NONE
No credentials specified
 > /bin/git rev-parse --resolve-git-dir /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/.git # timeout=10
Fetching changes from the remote Git repository
 > /bin/git config remote.origin.url https://github.com/Gamal-Mohammad/demo1.git # timeout=10
Fetching upstream changes from https://github.com/Gamal-Mohammad/demo1.git
 > /bin/git --version # timeout=10
 > git --version # 'git version 2.27.0'
 > /bin/git fetch --tags --force --progress -- https://github.com/Gamal-Mohammad/demo1.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
 > /bin/git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 9cc2ee2c66ea7d19f0d178400650a8cff79300cf (refs/remotes/origin/main)
 > /bin/git config core.sparsecheckout # timeout=10
 > /bin/git checkout -f 9cc2ee2c66ea7d19f0d178400650a8cff79300cf # timeout=10
Commit message: "Update PersonService.java"
 > /bin/git rev-list --no-walk 9cc2ee2c66ea7d19f0d178400650a8cff79300cf # timeout=10
Parsing POMs
Downloaded artifact https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/2.3.0.RELEASE/spring-
```



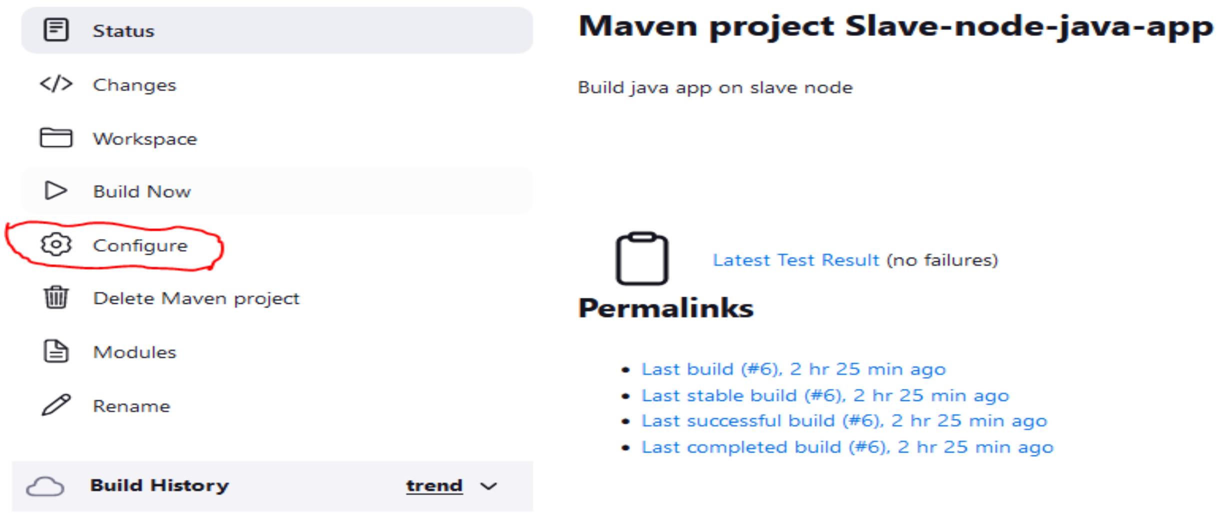
```
[INFO] Installing /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/target/demo1-0.0.1-SNAPSHOT.jar to /home/jenkins-
slave/.m2/repository/com/example/demo1/0.0.1-SNAPSHOT/demo1-0.0.1-SNAPSHOT.jar
[INFO] Installing /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/pom.xml to /home/jenkins-slave/.m2/repository/
com/example/demo1/0.0.1-SNAPSHOT/demo1-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  04:40 min
[INFO] Finished at: 2024-01-28T16:30:53Z
[INFO] ------------------------------------------------------------------------
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/pom.xml to com.example/demo1/0.0.1-SNAPSHOT/
demo1-0.0.1-SNAPSHOT.pom
[JENKINS] Archiving /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/target/demo1-0.0.1-SNAPSHOT.jar to com.example/
demo1/0.0.1-SNAPSHOT/demo1-0.0.1-SNAPSHOT.jar
channel stopped
Finished: SUCCESS
```

- After we have successfully build our job, now we have our App "**jar file**", ready to deploy.



```
[root@app02 ~]# ls /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/
Dockerfile  mvnw  mvnw.cmd  pom.xml  README.md  src  target
[root@app02 ~]# ls /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/target/
classes                    demo1-0.0.1-SNAPSHOT.jar.original  generated-test-sources  maven-status     test-classes
demo1-0.0.1-SNAPSHOT.jar  generated-sources                  maven-archiver          surefire-reports
[root@app02 ~]# 
```

- Now We will configure our job to deploy our App.

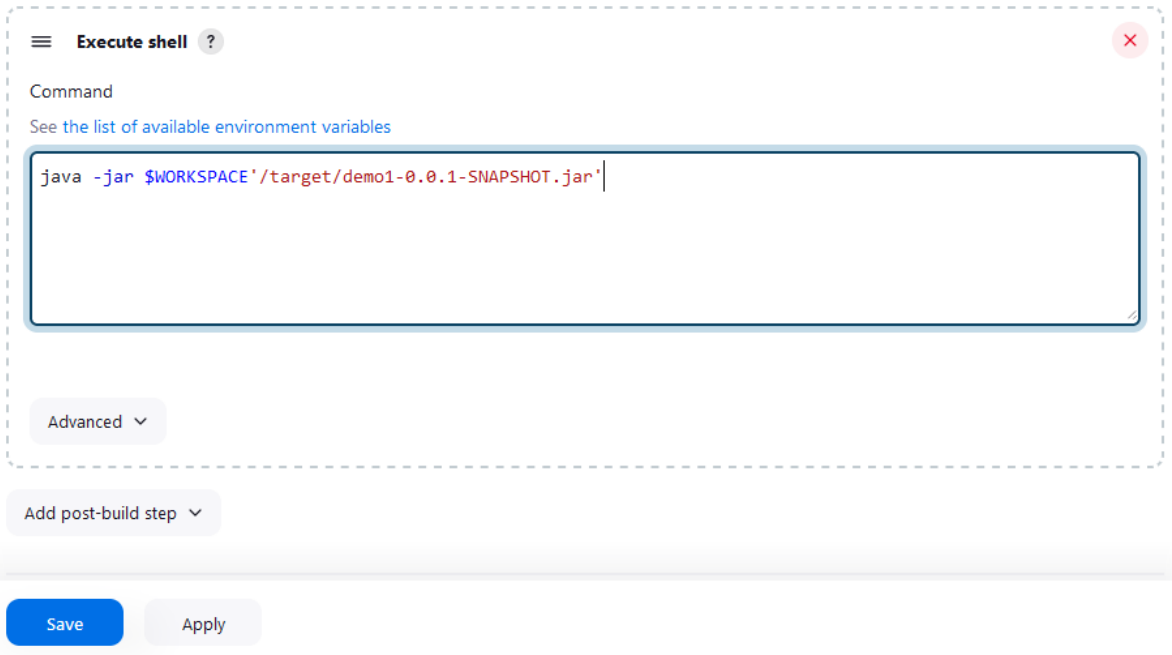


Maven project Slave-node-java-app

Build java app on slave node

Permalinks

- Last build (#6), 2 hr 25 min ago
- Last stable build (#6), 2 hr 25 min ago
- Last successful build (#6), 2 hr 25 min ago
- Last completed build (#6), 2 hr 25 min ago

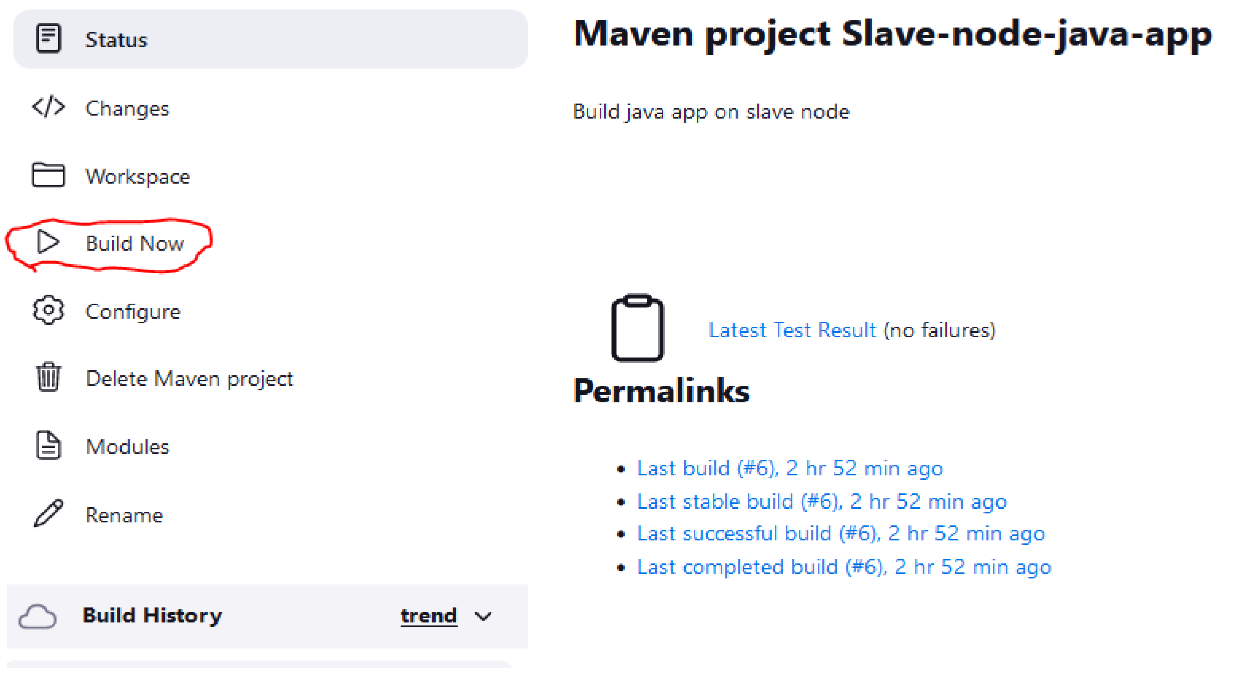


Command

See the list of available environment variables

```
java -jar $WORKSPACE'/target/demo1-0.0.1-SNAPSHOT.jar'
```

- We will add this shell command , click **Apply** and **Save** , and build the job again.



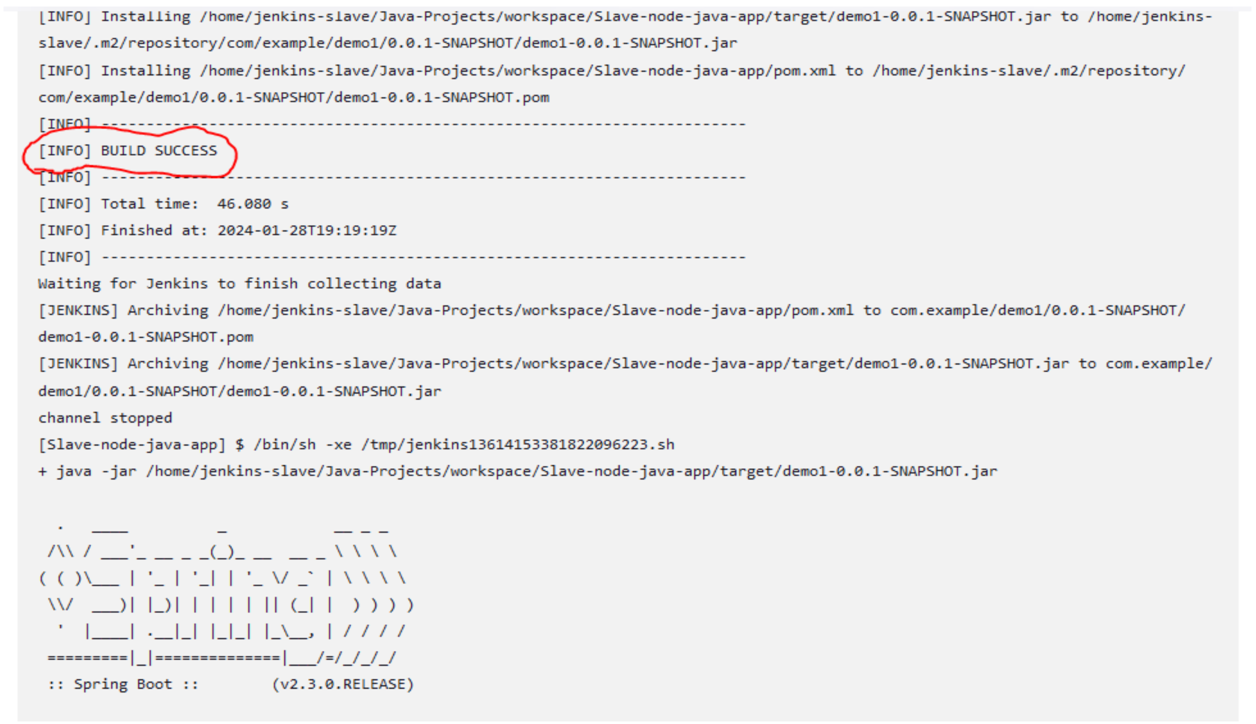
Maven project Slave-node-java-app

Build java app on slave node

Latest Test Result (no failures)

Permalinks

- Last build (#6), 2 hr 52 min ago
- Last stable build (#6), 2 hr 52 min ago
- Last successful build (#6), 2 hr 52 min ago
- Last completed build (#6), 2 hr 52 min ago

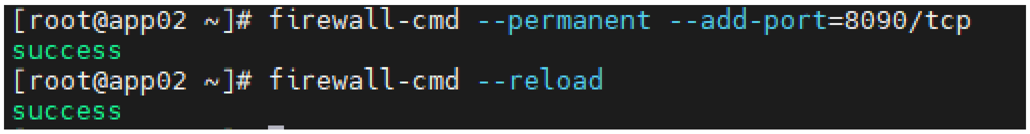


```
[INFO] Installing /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/target/demo1-0.0.1-SNAPSHOT.jar to /home/jenkins-
slave/.m2/repository/com/example/demo1/0.0.1-SNAPSHOT/demo1-0.0.1-SNAPSHOT.jar
[INFO] Installing /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/pom.xml to /home/jenkins-slave/.m2/repository/
com/example/demo1/0.0.1-SNAPSHOT/demo1-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  46.080 s
[INFO] Finished at: 2024-01-28T19:19:19Z
[INFO] ------------------------------------------------------------------------
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/pom.xml to com.example/demo1/0.0.1-SNAPSHOT/
demo1-0.0.1-SNAPSHOT.pom
[JENKINS] Archiving /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/target/demo1-0.0.1-SNAPSHOT.jar to com.example/
demo1/0.0.1-SNAPSHOT/demo1-0.0.1-SNAPSHOT.jar
channel stopped
[Slave-node-java-app] $ /bin/sh -xe /tmp/jenkins13614153381822096223.sh
+ java -jar /home/jenkins-slave/Java-Projects/workspace/Slave-node-java-app/target/demo1-0.0.1-SNAPSHOT.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.3.0.RELEASE)
```

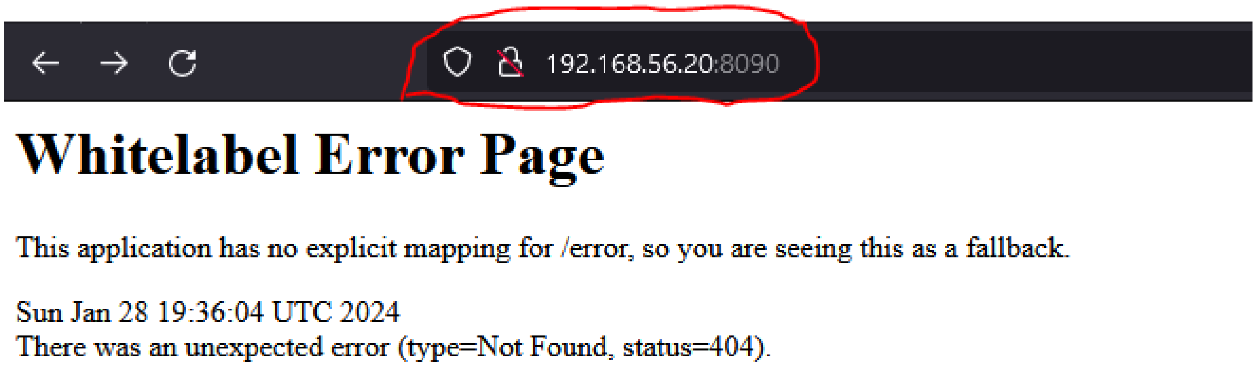
- We have successfully deployed our App.

# Enabling the firewall and allowing port 8090 to access our App.



```
[root@app02 ~]# firewall-cmd --permanent --add-port=8090/tcp
success
[root@app02 ~]# firewall-cmd --reload
success
```

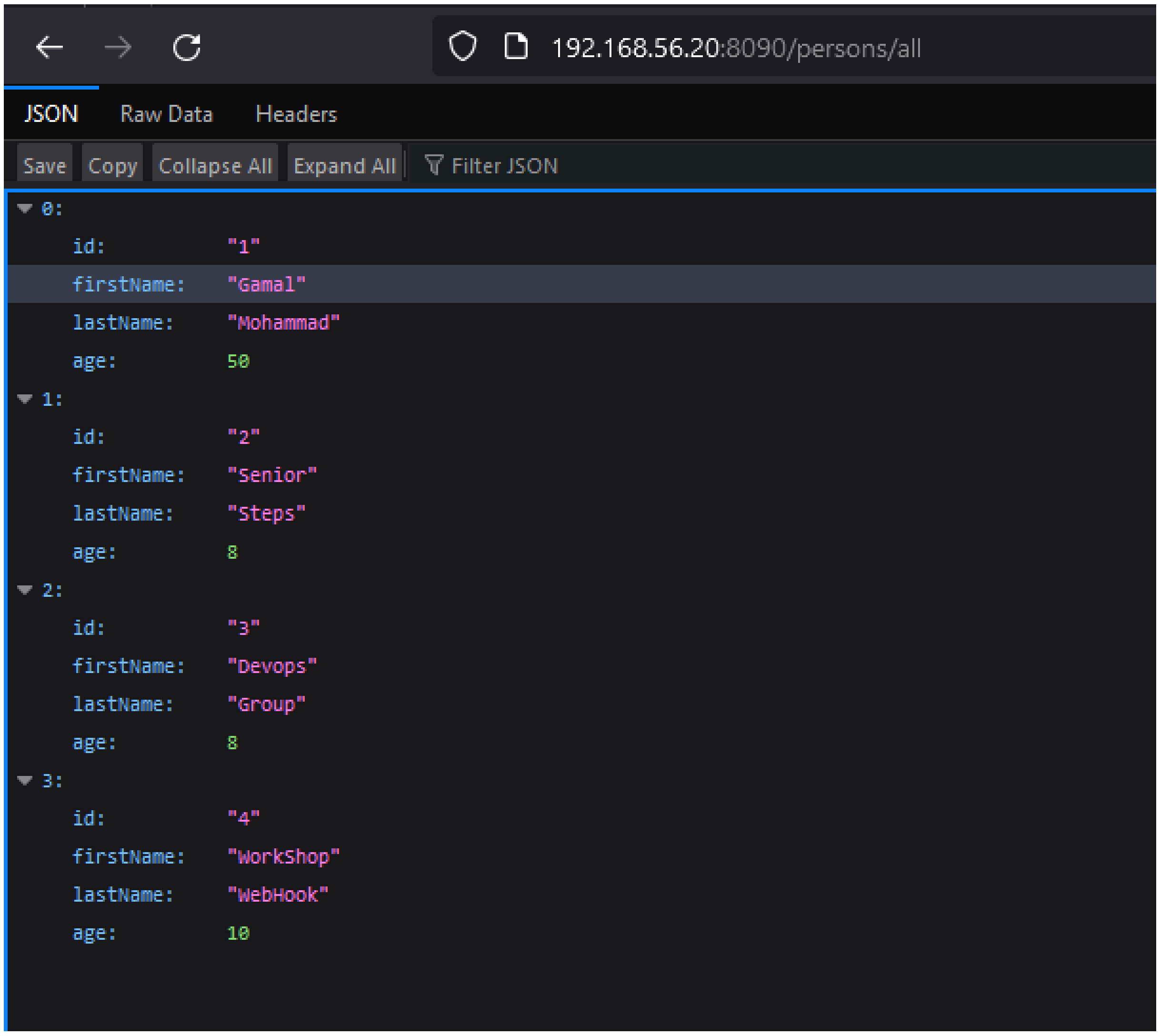
- On any Web browser we can access our app by Host "Slave node server" ip and port number "192.168.56.20:8090" .


192.168.56.20:8090

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun Jan 28 19:36:04 UTC 2024
There was an unexpected error (type=Not Found, status=404).



```
▼ 0:
    id:         "1"
    firstName:  "Gamal"
    lastName:   "Mohammad"
    age:        50
▼ 1:
    id:         "2"
    firstName:  "Senior"
    lastName:   "Steps"
    age:        8
▼ 2:
    id:         "3"
    firstName:  "Devops"
    lastName:   "Group"
    age:        8
▼ 3:
    id:         "4"
    firstName:  "WorkShop"
    lastName:   "WebHook"
    age:        10
```

➢ Conclusion:

# In this project we have covered:

1. How to install **Jenkins**, and access it’s web page.
2. How to add plugins (Git and Maven) and how to integrate them with **Jenkins**.
3. How to Create Slave node and join it to Master.
4. How to Build and Deploy Java Project on Slave node using Maven in Jenkins, and using a GitHub(public repository).