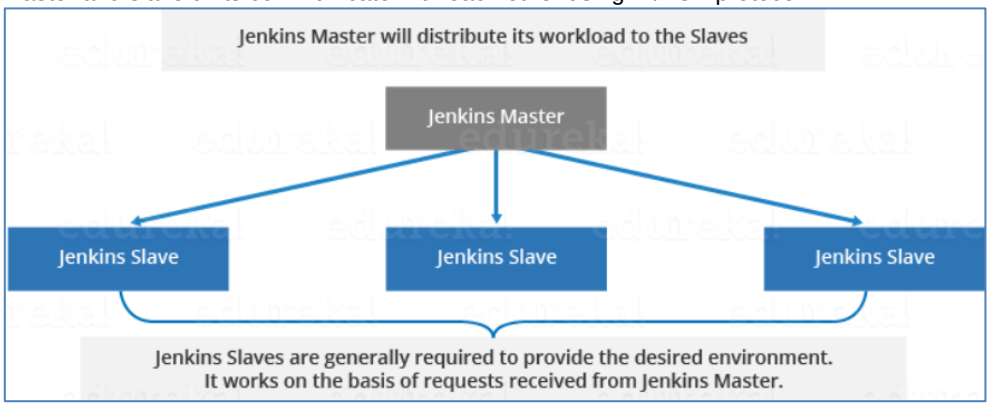
Table of Contents

1. Jenkins Master Slave 2
1.1. Jenkins Master 2
1.2. Jenkins Slave 2
1.3. How Jenkins Master and Slave Architecture works 2
1.4. Implementation 3
1.5. Install Jenkins Master 3
1.6. Configuring Credentials 4
1.6.1. Types of credentials 4
1.6.2. Download required plugins 4
1.6.3. Configuring credentials 4
1.6.4. Adding new global credentials 4
1.7. Setup up Slaves Nodes 5
1.8. Add Slaves Nodes 6
1.9. Prepare Slave Agent Nodes to Execute Build 7
1.10. Creating a Pipeline and Running on The Slave Machine 8

1. Jenkins Master Slave

- Jenkins manages the builds with the help of master-slave architecture.
- Master and slave units communicate with each other using IP/TCP protocol.



1.1. Jenkins Master

- This is the primary server of Jenkins.
- 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.
  - Master Jenkins is capable of directly executing build jobs.

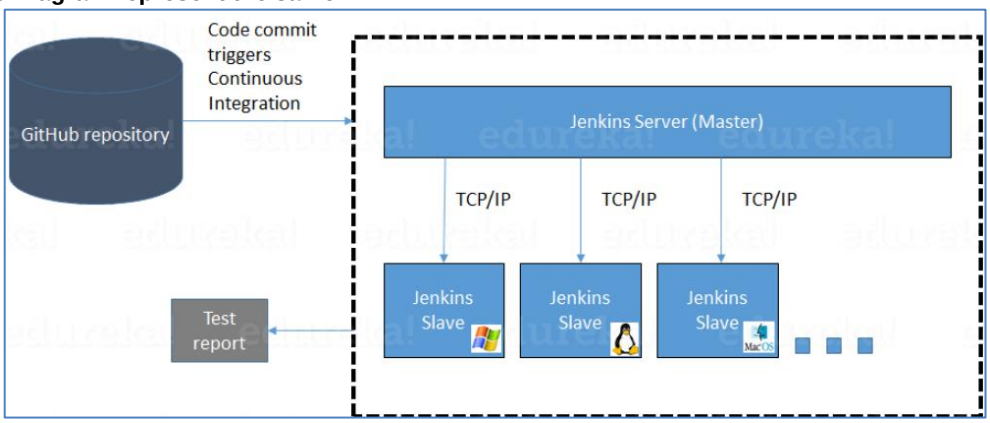
1.2. Jenkins Slave

- A Slave is a Java executable that runs on a remote machine. Following are the characteristics of Jenkins Slaves:
  - It runs on the remote slave and hears requests from the Jenkins Master instance and follow the task.
  - 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.

1.3. How Jenkins Master and Slave Architecture works

- Jenkins Master checks the Git repository at periodic intervals for any changes made in the source code.
- To perform the build for different testing environment Jenkins uses various Slaves.
- Jenkins Master requests these Slaves to perform define testing task and generate the test reports.

The Diagram represent the same



1.4. Implementation Steps

1. Install Jenkins Master
2. Configure Jenkins Master Credentials
3. Configure Slave Agent Nodes
4. Add New Slave Nodes
5. Prepare Slave Agent Nodes to Execute Build
6. Testing

1.5. Install Jenkins Master

➢ **Install on Ubuntu**

- Jenkins is based on Java, so we need to install Java OpenJDK version 7 on the server.
  - Install python-software-properties with apt command.
    ```
    # apt-get install python-software-properties
    ```
  - add Java PPA repository to the server
    ```
    # add-apt-repository ppa:openjdk-r/ppa
      Just Press ENTER
    ```
  - Update the Ubuntu repository and install the Java OpenJDK
    ```
    # apt-get update
    # apt-get install openjdk-7-jdk
    ```
  - Verify the Java installation
    ```
    # java -version
    ```

- **Install Jenkins**
  - Add Jenkins key and repository to the system
    ```
    # wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
    # echo 'deb https://pkg.jenkins.io/debian-stable binary/' | tee -a /etc/apt/sources.list
    ```
  - Update the repository and install Jenkins
    ```
    # apt-get update
    # apt-get install jenkins
    ```

➢ **Install on RHEL**

- Jenkins is based on Java, so we need to install Java OpenJDK version on the server.
  - Update the Ubuntu repository and install the Java OpenJDK
    ```
    # yum -y update
    # yum install java-1.8.0-openjdk
    ```
  - Setup the JAVA_HOME at start.
    ```
    # vi ~/.bash_profile
    export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-3.b12.el7_3.x86_64/
    export JRE_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-3.b12.el7_3.x86_64/jre
    ```
  - Verify the Java installation
    ```
    # java -version
    ```

- **Install Jenkins**
  - Add the Jenkins repository to your system
    ```
    # wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
    ```
  - Import the GPG key
    ```
    # rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
    ```
  - install Jenkins.
    ```
    # yum -y install jenkins
    ```
  - start Jenkins
    ```
    # systemctl start jenkins
    # systemctl stop jenkins
    # systemctl status jenkins
    ```

1.6. Configuring Credentials

- There are multiple 3rd-party sites and applications that can interact with Jenkins, for example, artifact repositories, cloud-based storage systems and services.
- Those 3rd-party sites and applications can configure credentials in the application for dedicated use by Jenkins.
- Once a Jenkins manager adds/configures these credentials in Jenkins, the credentials can be used by Pipeline projects to interact with these 3rd party applications.
- Credentials stored in Jenkins can be used:
  - anywhere applicable throughout Jenkins (i.e. global credentials).
  - by a specific Pipeline project/item in JenkinsFile.
  - by a specific Jenkins user (as is the case for Pipeline projects created in Blue Ocean)

1.6.1. Types of credentials

- Jenkins can store the following types of credentials
  - **Secret text** - a token such as an API token (e.g. a GitHub personal access token),
  - **Username and password** - which could be handled as separate components or as a colon separated string in the format `username:password`
  - **Secret file** - which is essentially secret content in a file,
  - **SSH Username with private key** - an SSH public/private key pair,
  - **Certificate** - a PKCS#12 certificate file and optional password, or
  - **Docker Host Certificate Authentication** credentials.

1.6.2. Download required plugins

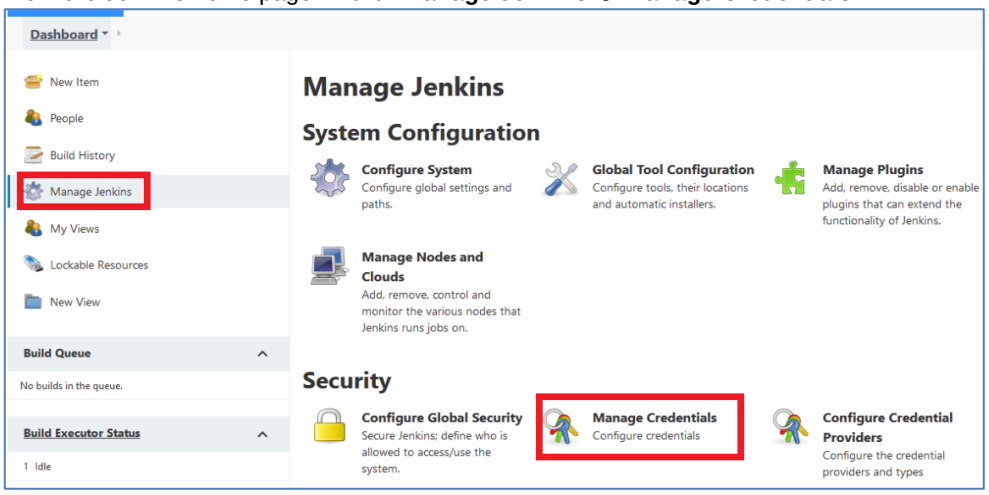
- Following Plugins are required.
  - **Credentials plugin:** provides a centralized way to define credentials that can be used by your Jenkins instance, plugins and build jobs.
  - **Credentials Binding plugin:** allows you to configure your build jobs to inject credentials as environment variables.
  - **Plain Credentials plugin:** a plugin dependency required by the Credentials Binding plugin.

1.6.3. Configuring credentials

- Credentials can be configured by Jenkins administrator user who has a correct permission.

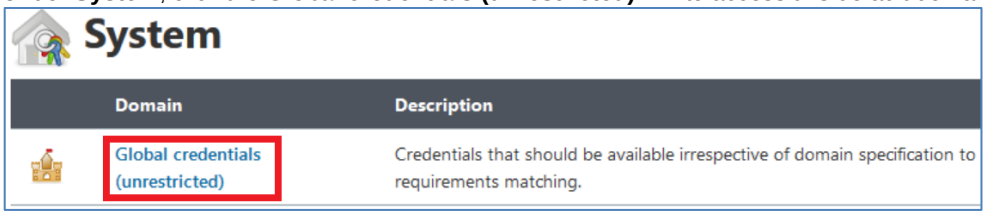
1.6.4. Adding new global credentials

- Logged in to Jenkins as a user with the **Credentials → Create** permission.
- From the Jenkins home page → click **Manage Jenkins → Manage Credentials**.

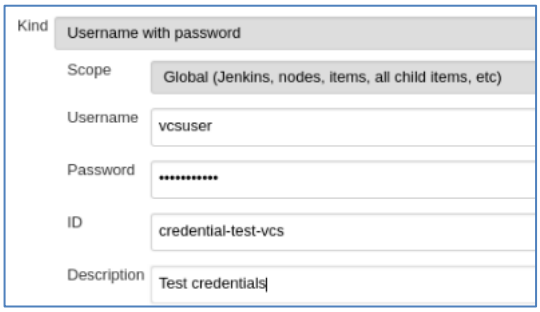


- Under **Stores scoped to Jenkins** on the right, click on **Jenkins**

- Under **System**, click the **Global credentials (unrestricted)** link to access this default domain.



- Click **Add Credentials** on the left.
  - **Kind** field, choose the type of credentials to add
  - From the **Scope** field, choose **Global**
  - Fill other information and Click **OK**



1.7. Setup up Slaves Nodes

- Install Java on Slave

```
# sudo apt install software-properties-common apt-transport-https -y
# sudo add-apt-repository ppa:openjdk-r/ppa -y
# sudo apt install openjdk-8-jdk -y
# java -version
```

- Create Jenkins User on Slave

```
# useradd -m -s /bin/bash jenkins
# passwd Jenkins
```

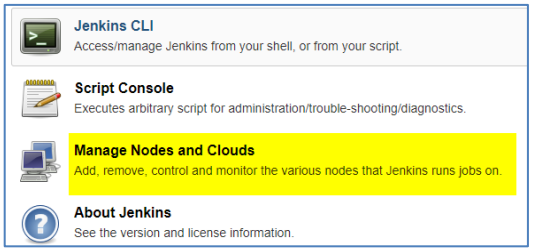
- Copy the SSH Key from Master to Slave

```
# ssh-copy-id jenkins@10.0.15.21
# ssh-copy-id jenkins@10.0.15.22
```

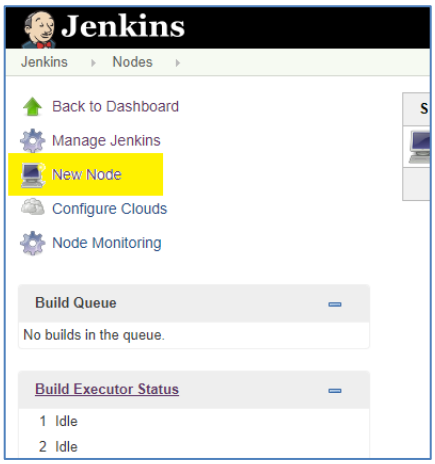
1.8. Add Slaves Nodes

- Click on **Manage Jenkins** in the left corner on the Jenkins dashboard.

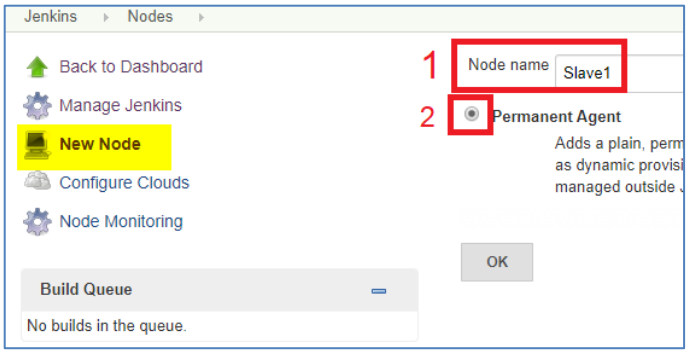
- Click on **Manage Nodes**.



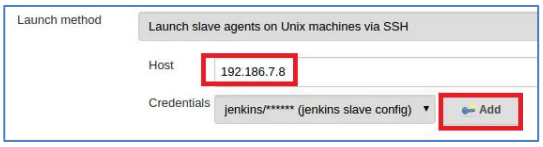
- Click on **New Node**



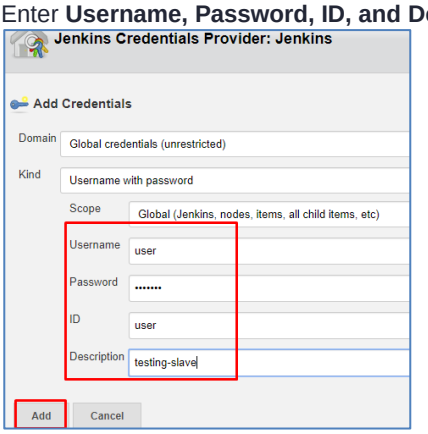
- Type a name for the node (**Slave1**), choose the **Permanent Agent** option and click on **Ok**.



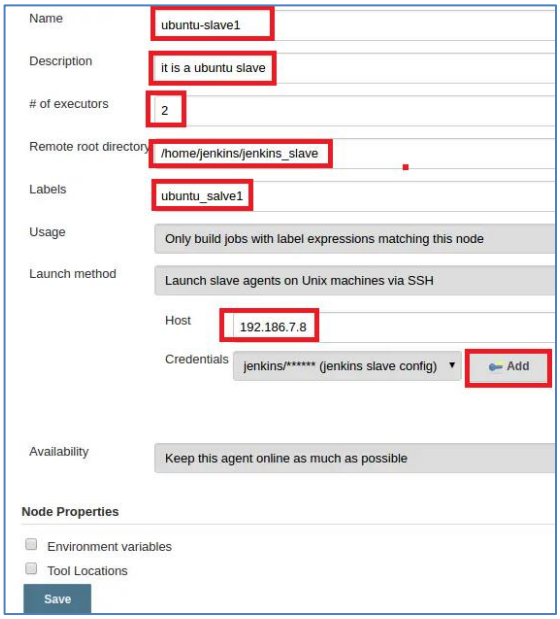
- Enter the details of the node slave machine.
  - **no. of executers** in nothing but no. of jobs that this slave can run parallelly.
  - The **Labels** for which the name is entered as “Slave1” is what can be used to configure jobs.
  - Select **Usage** to **Use this node as much as possible**.
  - For l**aunch method** we select the option of “**Launch agent by connecting it to the master**”.
    - Enter the Hostname in the **Host** field.



    - Select the **Add** button to add credentials. and click **Jenkins**.
    - Enter **Username, Password, ID, and Description** of **Slave1**



  - Enter **Custom WorkDir path** as the workspace of your slave node.
  - In **Availability** select “**Keep this agent online as much as possible**”.



  - Click on **Save**

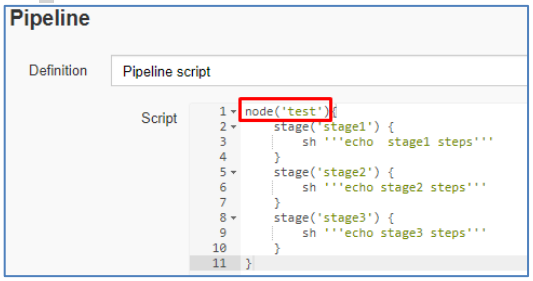
1.9. Prepare Slave Agent Nodes to Execute Build

- Click on the **'Manage Jenkins**' menu and then click **'Configure System**'.
- Now go to the **'Slave Setups'** section and define all.
- Click **'Save'.**

1.10. Creating a Pipeline and Running on The Slave Machine

- Click **New Item** in the top left corner on the dashboard
- Enter the name of your project in the **Enter an item name** field, and select the **Pipeline** project, and click **OK** button.
- Go to the **Pipeline** section, make sure the **Definition** field has the **Pipeline script** option selected.
- Copy and paste the following declarative Pipeline **script** into a script field.

```
node('test1'){
  stage('stage1') {
    sh '''echo stage1 steps'''
  }
  stage('stage2') {
    sh '''echo stage2 steps'''
  }
  stage('stage3') {
    sh '''echo stage3 steps'''
  }
}
```



- Click on **Save**, it will redirect to the Pipeline view page.
- On the left pane, click the **Build Now** button to execute your Pipeline.