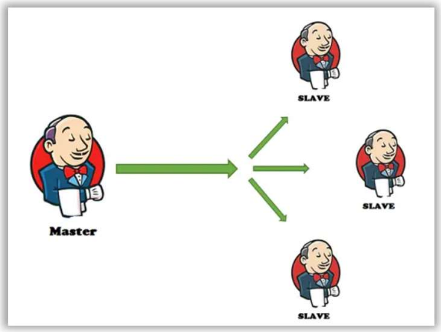
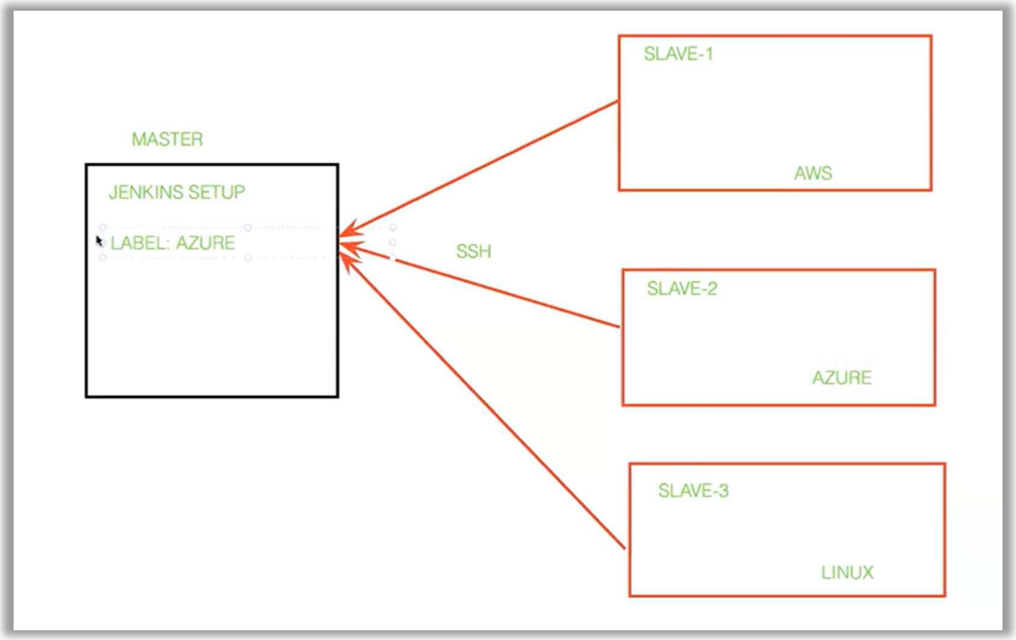
Jenkins Class – 6

if we have multiple task in the same instance it will get more load on the server so it may get shutdown due to over loading. But, we have solution for this. Just create few instance which is called **slave** then integrate it with **Master.**



We need to install Jenkins in master for making continuous integration. But no need to install Jenkins in slaves but, need to install java11 on each slave. Because master will give the task to slave, if they want to do the Jenkins related task they might have java which is dependency right?.

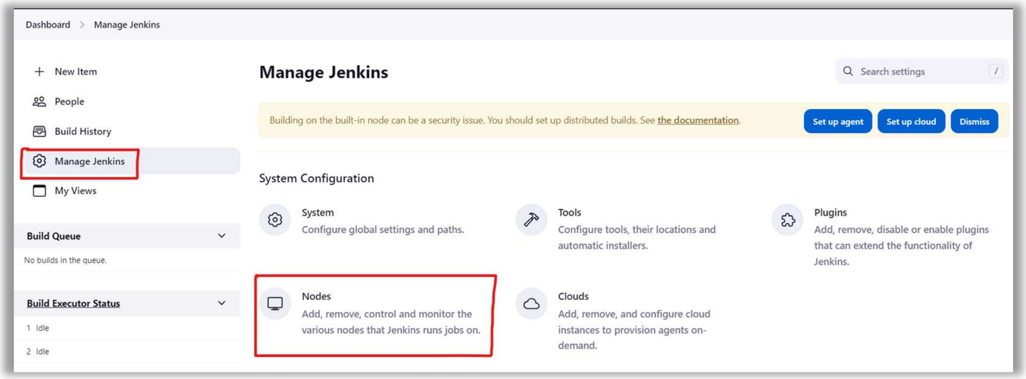
So, we need to install java11 on each slave. While we create the slave, we need to set the label to each slave. Because if we want to call the slave, how can we call them? So if we create the label to each slave we can call them by label. From the below snap, if we call label **AZURE** the task will assign to the **slave – 2**. Like wise we can assign the task to specific slave by calling label name. If we didn't mention any label the task will take over by the master.



The job log also saved in the slave because the master will give the job to the slave to build it. We should take keypair while create the instance.

Launch the Master then **install Jenkins in Master**. Then go the slave instance and **install java11 on slave**.

After install the java and Jenkins in master, go to the **manage Jenkins->Nodes.**



Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

No builds in the queue.

1 Idle

2 Idle

System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

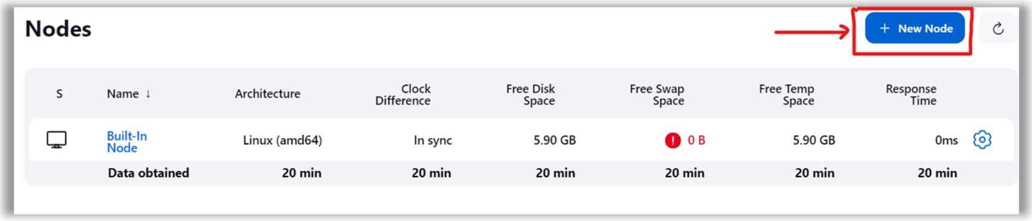
Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

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.

Once go to the Nodes we can see the Build in Node which is whenever we created the Jenkins it will create the default node to the current Jenkins server.

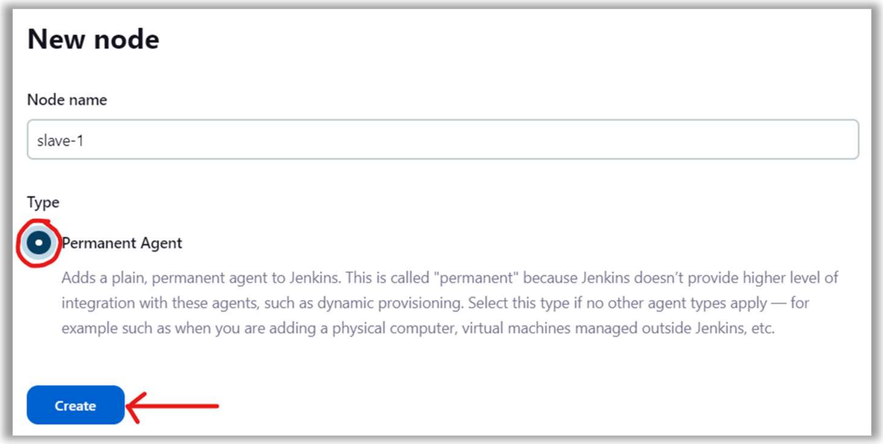
If we want to add the extra server which is called slave or node, click on **New Node.**



Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.90 GB	0 B	5.90 GB	0ms
	Data obtained	20 min	20 min	20 min	20 min	20 min	20 min

Once click on **New Node** we will get the new node creation page as mention below. Now enter the name of your slave/node as you wish and select the **permanent agent** then click on create.



New node

Node name

slave-1

Type

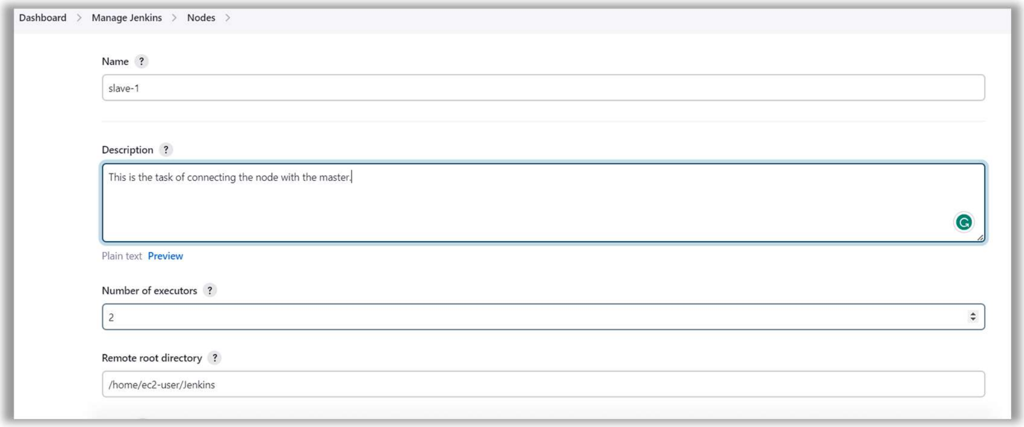
[✓ Permanent Agent]

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.

Number of Executors

Now you can see the one of the options is **Number of executors.** Which is used to execute the job. For example we have interpreter to execute the python code and complier helps to execute the java code like the executors helps to execute the jobs here.

If we have 3 jobs, we should have 3 executors to build the 3 jobs as soon as possible. If we choose 1 executor for 3 jobs, it will take more time to build the jobs as execute the jobs one by one.



Name: slave-1

Description: This is the task of connecting the node with the master.

Number of executors: 2

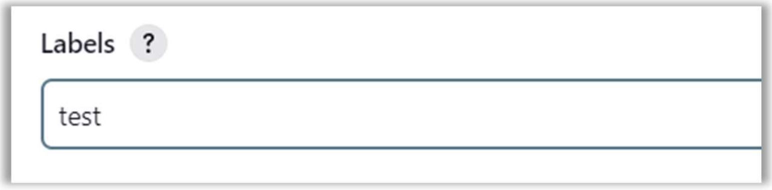
Remote root directory: /home/ec2-user/Jenkins

Remote root directory

We have another option which is Remote root directory. If we build the job in master, it will store the data in default path which is **/var/lib/Jenkins/.** But if we perform the task in slave, where will it save. So we should declare it in this **Remote root directory.** But we should take the path like **/home/ec2-user/<folder name>/.** You can choose the <folder name > as you wish but the format should have like that only. If you take different path, it may get error.

Labels

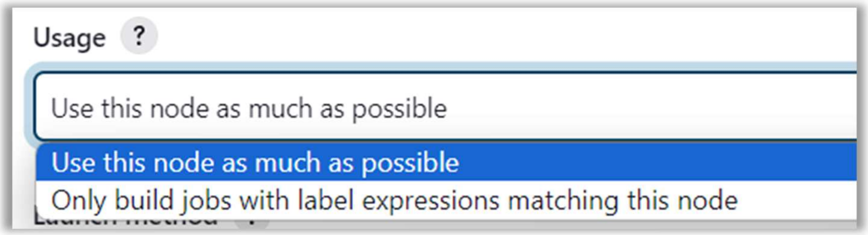
In the label part we can choose any name to the label. Which is used to call the node from the master.



Labels: test

Usage

In this part we have to select **Only build jobs with label expressions matching this node.** Because either we call this node or not the build will assign to the slave which is connected with master if you choose **Use this node as much as possible.** If you choose the second option when you call the node that time only the task will assign to the slave. Rest of the time the task will performed by master only.



Usage ?

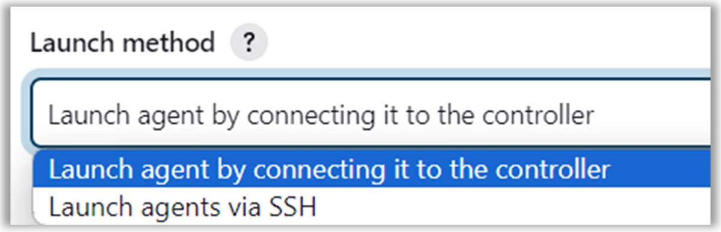
Use this node as much as possible

Use this node as much as possible

Only build jobs with label expressions matching this node

Launch method

We are using SSH protocol to connect the nodes from master to slave to we should select **Launch agent via SSH.** Because we don't have controllers to make the connection between master and slave.



Launch method ?

Launch agent by connecting it to the controller

Launch agent by connecting it to the controller

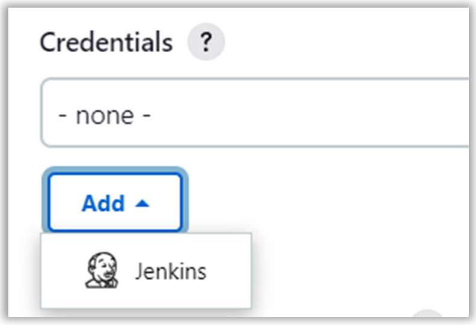
Launch agents via SSH

Host

In the host part we should give the slave's public/private address.

Credentials

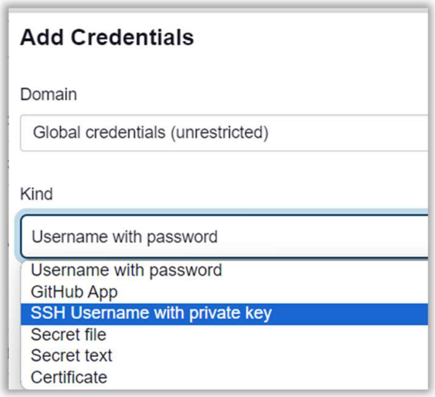
We don't have any credentials to add here. So we need to add the credentials by click on **Add-> Jenkins**



Credentials ?

- none -

In this part we have to select the SSH username with private key. Because we didn't set any username password to the ec2-user server. But we have **private key** of the **key-pair** credential. So we can go with private key.



Add Credentials

Domain

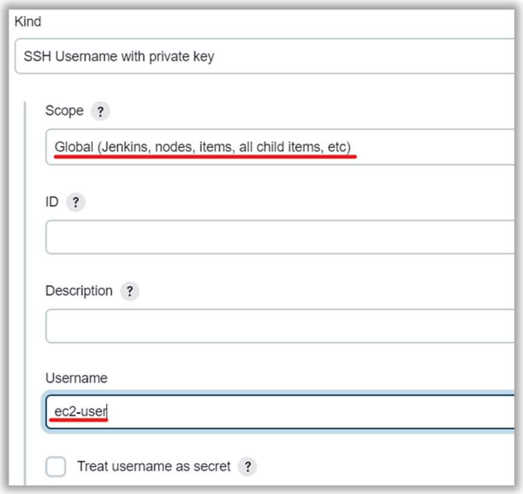
Global credentials (unrestricted)

Kind

Username with password

- Username with password
- GitHub App
- SSH Username with private key
- Secret file
- Secret text
- Certificate

Here we should enter the username as ec2-user. Because the private key is in the ec2-user.



Kind

SSH Username with private key

Scope ?

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

ID ?

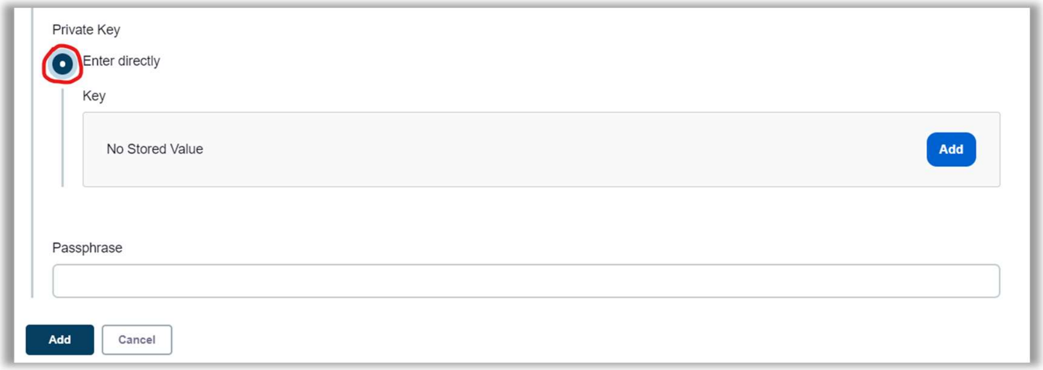
Description ?

Username

ec2-user

Treat username as secret ?

Click on **Enter directly** and click on **Add**



Private Key

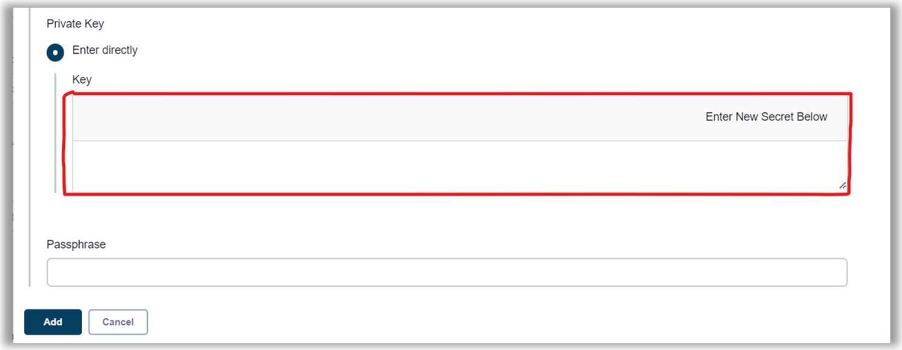
[✓ Enter directly]

Key

No Stored Value

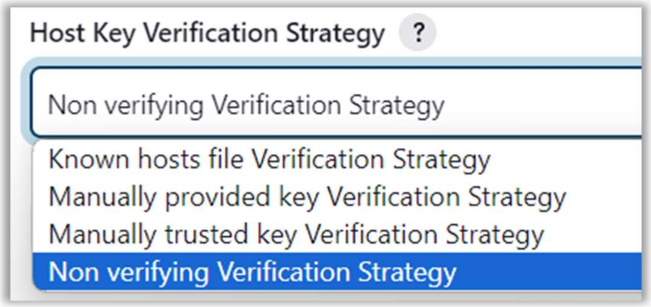
Passphrase

Then you will get the text box field to fill the private key. When you create the key-pair in aws instance that time you got download the private key which is ends with .pem. you should open that file in txt document mode. Then copy the entire text and paste it in this key field. If you have set any passphrase (password) you can enter it in the passphrase field and click on Add.



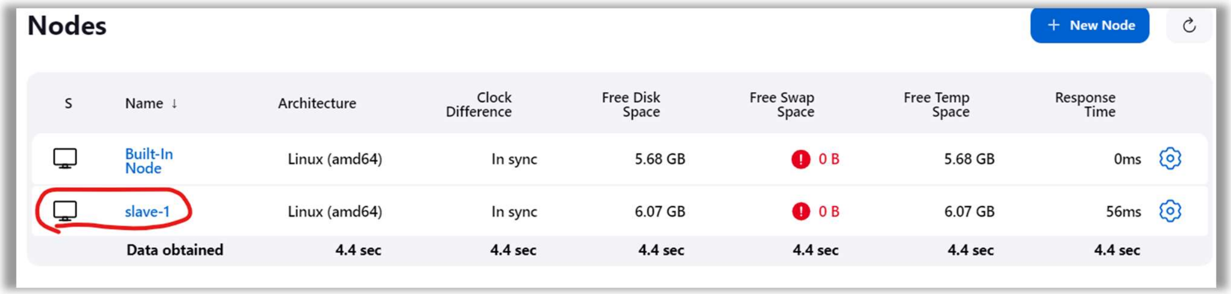
Host Key Verification Strategy

In this part we have to choose Non verifying verification strategy. Because we already set the credentials as well. So no need to se the verification strategy in it.


Host Key Verification Strategy ?
Non verifying Verification Strategy
Known hosts file Verification Strategy
Manually provided key Verification Strategy
Manually trusted key Verification Strategy
Non verifying Verification Strategy

Keep the remaining things. Just click on save now.

After follow all the above steps finally we can see the slave/node in the node dashboard



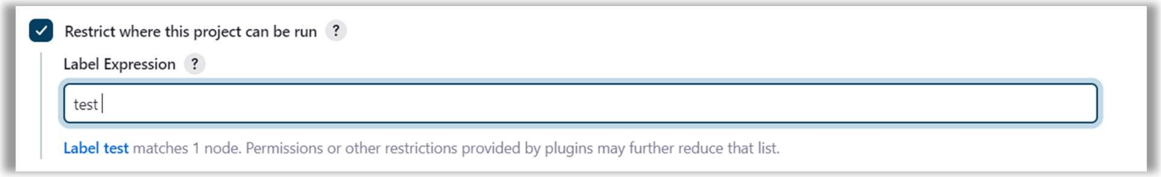
Nodes

+ New Node

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.68 GB	0 B	5.68 GB	0ms
	slave-1	Linux (amd64)	In sync	6.07 GB	0 B	6.07 GB	56ms
	Data obtained	4.4 sec	4.4 sec	4.4 sec	4.4 sec	4.4 sec	4.4 sec

Now lets try to launch the job in slave by follow the below steps.

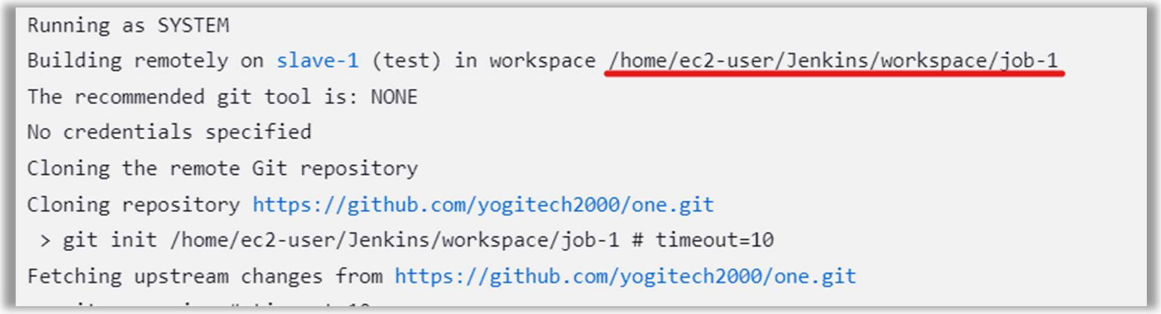
As usual create Jenkins job for GitHub repository as we discussed earlier . now we have to choose the option which is **Restrict where this project can be run.** We should enter the label name which label name has been set to that slave. Now we can see the notification in below that **Label test** matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.



```
test
```

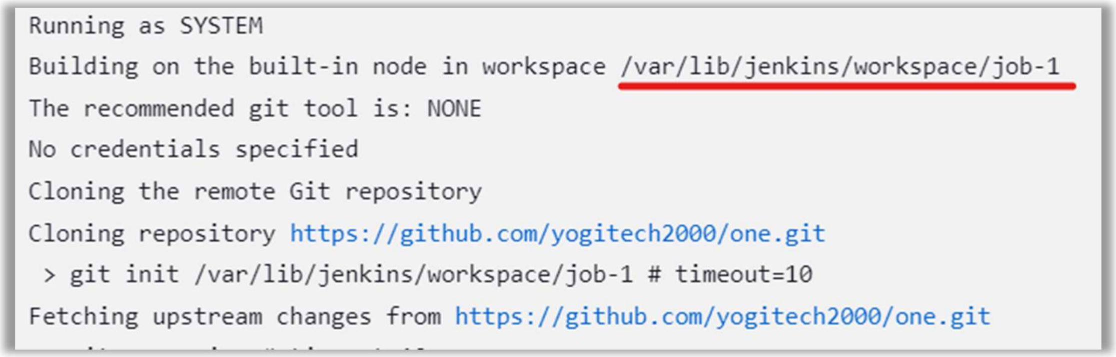
Label test matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Now click on save and build it. We can see the output which is saved the git repository to the slave as we mentioned the path.



```
Running as SYSTEM
Building remotely on slave-1 (test) in workspace /home/ec2-user/Jenkins/workspace/job-1
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/yogitech2000/one.git
 > git init /home/ec2-user/Jenkins/workspace/job-1 # timeout=10
Fetching upstream changes from https://github.com/yogitech2000/one.git
```

If we remove the label, the build will happen in the master.



```
Running as SYSTEM
Building on the built-in node in workspace /var/lib/jenkins/workspace/job-1
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/yogitech2000/one.git
 > git init /var/lib/jenkins/workspace/job-1 # timeout=10
Fetching upstream changes from https://github.com/yogitech2000/one.git
```

In the same way we can connect the slave with master and given the task by call the label.

Note: we can use the same credentials(private key) to the other slave if we launch the instance with the same key-pair.

Like wise we can do any task in slave or master based on the labels.

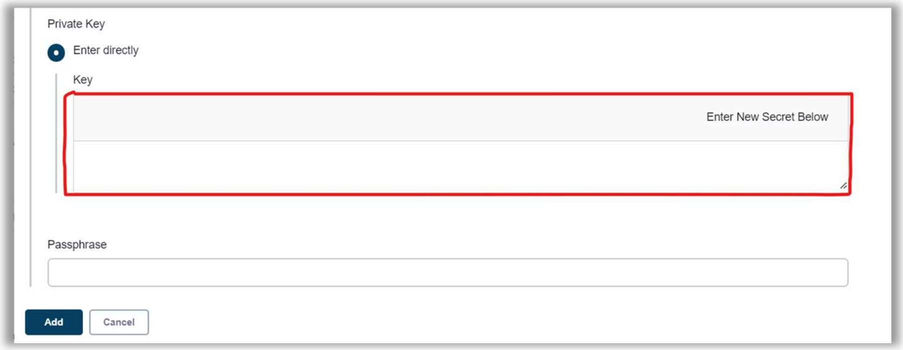
Connect the master(with key) slave(without key):

For exercise with example

We need to launch the two instances. One is with keypair another one is without keypair.

Now to connect each other follow the below steps,

1. Need to generate the ssh key in master (with key). **Command: ssh-keygen.**
2. **Go to: .ssh/id_rsa.pub.** copy the whole content.
3. **Go to:** slave (without key) path: **.ssh/authorized_keys.** Paste the copied content here.
4. Then go to master(with key) path: **.ssh/id_rsa** copy the private key. Paste it in one of the node credentials and add it.



For webhook integration:

Add the master (installed Jenkins) public IP to the GitHub webhook URL. Because the GitHub integrated to master. Then master assign the task to the slave. So, we need to link the master public IP with GitHub account.