

Day 28 Task: Jenkins Agents

Jenkins Master (Server)

The Jenkins Master acts as the control centre for all the activities that take place within a Jenkins environment. It manages the distribution of tasks to different Jenkins agents (also called "slaves" or "nodes"), which are responsible for performing the actual builds, tests, and deployments.

The Jenkins Master also provides a web-based user interface (UI) for managing the Jenkins environment, configuring jobs, and viewing build logs and reports. It allows users to define different job types, such as freestyle jobs, pipeline jobs, and multi-configuration jobs, to automate different aspects of the software development process.

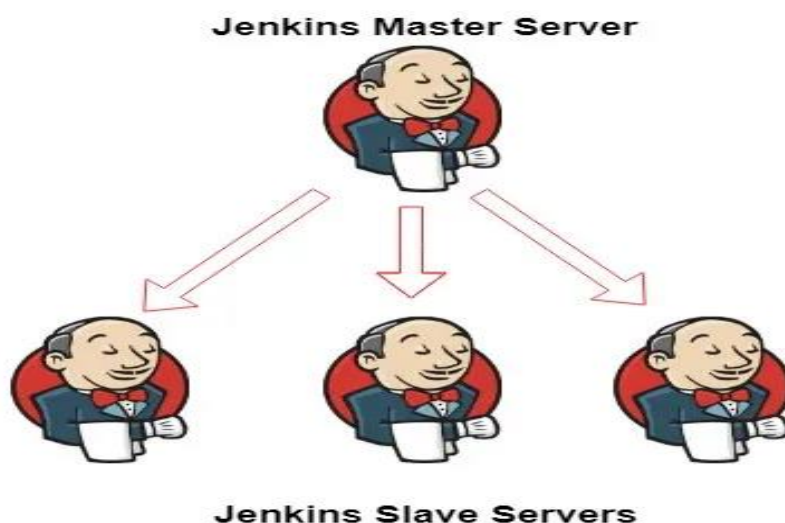
Jenkins Agent

Jenkins Agent, also known as a "slave" or "node", is a separate software component that works with the Jenkins Master (Server) to execute build, test, and deployment tasks in a distributed manner.

When a build or test job is submitted to Jenkins, the Jenkins Master assigns the task to an available Agent that meets the job's requirements. The Agent then downloads the necessary code, dependencies, and configurations from the Jenkins Master and performs the build or test task in a separate workspace.

After the task is complete, the Agent reports the result back to the Jenkins Master, which consolidates the results from all Agents and generates a report that can be viewed through the web-based user interface.

Jenkins Agents provide the ability to distribute build and test workloads across multiple machines, which can significantly improve the performance and scalability of a Jenkins environment. By allowing for parallel execution of tasks, Jenkins Agents can help to reduce the time required to complete builds and tests, and improve the overall productivity of development teams.



Pre-requisites

Let's say we're starting with a fresh Ubuntu 22.04 Linux installation. To get an agent working make sure you install Java (same version as jenkins master server) and Docker on it.

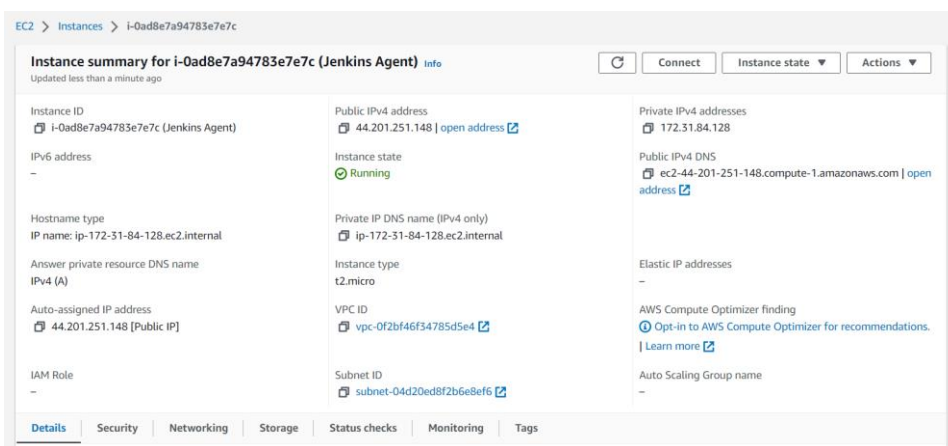
Note:- While creating an agent, be sure to separate rights, permissions, and ownership for jenkins users.

Task-01

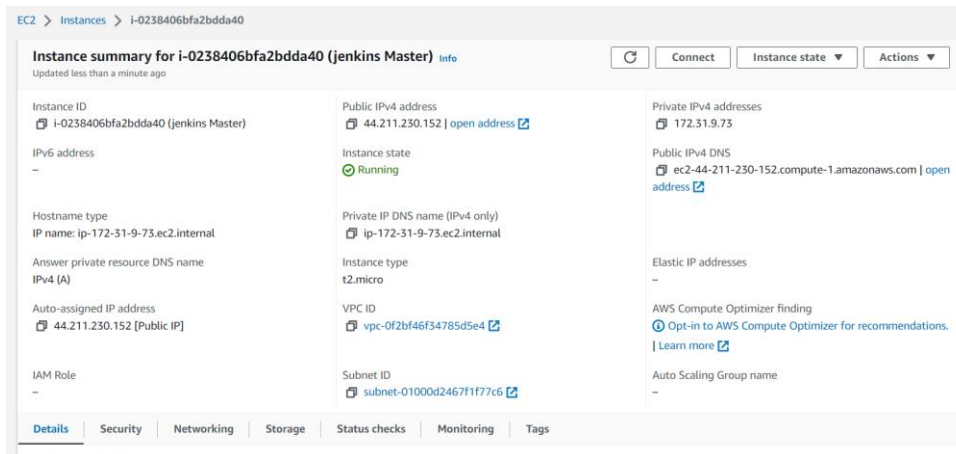
- Create a new AWS EC2 Instance and connect it to master (Where Jenkins is installed)

Here I have created a new EC2 instance called Jenkins Agent which I am going to connect it to another instance called Jenkins master where Jenkins and docker are already installed

1. Jenkins Agent



2. Jenkins Master



- The connection of master and agent requires SSH and the public-private key pair exchange.

Step 1 - Generate SSH keys on “Jenkins-agent” EC2 instance

Step 2 - Add public key from “Jenkins-agent” instance to “Jenkins-master” instance under location “.ssh/authorized_keys”

Id_rsa.pub key from Jenkins agent :

```

ubuntu@ip-172-31-84-128:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:z5GgwBoahDotP/tp18/oJ53Fza4YWou6y+WreJirCts ubuntu@ip-172-31-84-128
The key's randomart image is:
+---[RSA 3072]-----+
| o . . |
| +. . . |
| +o + . |
| . o . . |
| . . S.oo |
| o o +o.o |
| . o.o+oo. |
| .o. ++*+o . |
| o.E+==OOB.. |
+---[SHA256]-----+
ubuntu@ip-172-31-84-128:~$ cd /home/ubuntu/.ssh/
ubuntu@ip-172-31-84-128:~/.ssh$ ls
authorized_keys id_rsa id_rsa.pub
ubuntu@ip-172-31-84-128:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCscHjLdETj3GNzqfV091f8Czt648hXdpOFXHQ3cypbboUZ0DuOu/EB+cFumvbGB4oqZm8wKNxaEZvk
PrF8U53tR4OzudIZcmP07ohddW6g0TmhPLS/Tof7nxOWmwsb3o6hggFIHibhYmTMOEnLtQ8GgAxEBHm5yEv6SgzAk3oq/FYITS0zi4RrqUiy/s/mnbZhsG
ZOTC/KrIjbiDio83enhgdbhlc4aus7WaeBw16/8hZhJnCE6bnCqx89I8rquS3+WtasMXTFwhmhiQABJ8GZE2vv/9jibfjLUZrBf1Lz1X14OBp1+7fp/7
31-84-128
ubuntu@ip-172-31-84-128:~/.ssh$

```

authorized key from Jenkins master :

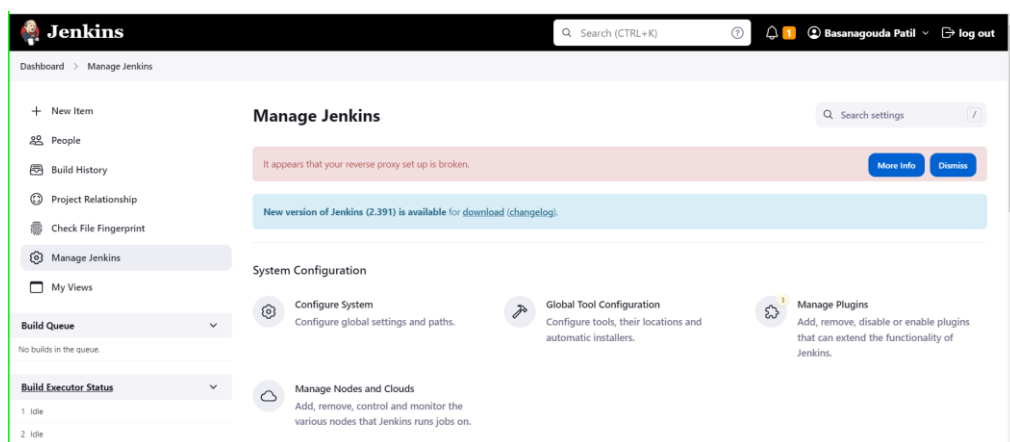
```

ubuntu@ip-172-31-9-73:~$ cd /home/ubuntu/.ssh/
ubuntu@ip-172-31-9-73:~/.ssh$ sudo nano authorized_keys
ubuntu@ip-172-31-9-73:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCscHjLdETj3GNzqfV091f8Czt648hXdpOFXHQ3cypbboUZ0DuOu/EB+cFumvbGB4oqZm8wKNxaEZvk
PrF8U53tR4OzudIZcmP07ohddW6g0TmhPLS/Tof7nxOWmwsb3o6hggFIHibhYmTMOEnLtQ8GgAxEBHm5yEv6SgzAk3oq/FYITS0zi4RrqUiy/s/mnbZhsG
ZOTC/KrIjbiDio83enhgdbhlc4aus7WaeBw16/8hZhJnCE6bnCqx89I8rquS3+WtasMXTFwhmhiQABJ8GZE2vv/9jibfjLUZrBf1Lz1X14OBp1+7fp/7
31-84-128
ubuntu@ip-172-31-9-73:~/.ssh$

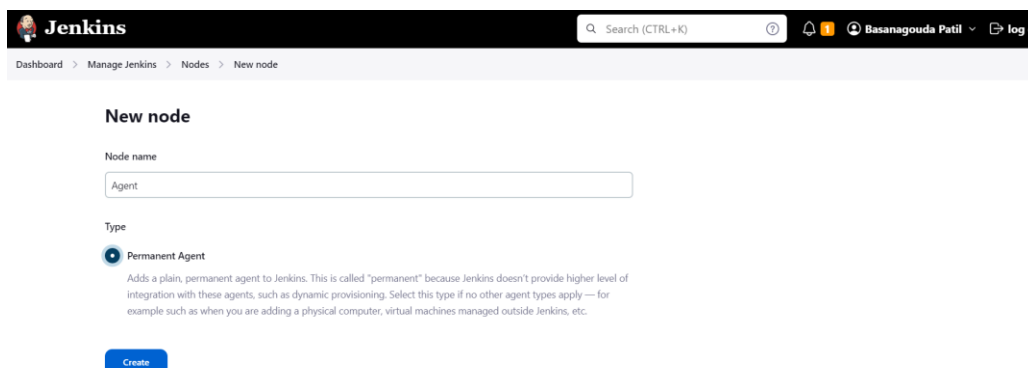
```

- The connection of master and agent requires SSH and the public-private key pair exchange.

Step 1 – Login to the Jenkins and go to dashboard, and click on “Manage Jenkins” then click on “Manage Nodes and Clouds”



Step 2 - To create a node click on "New Node" and give name to it and make it permanent agent then click on create.

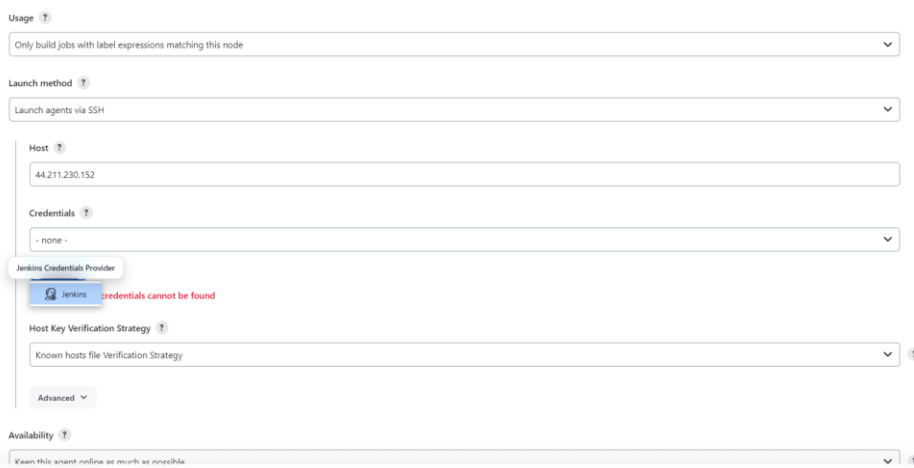


Step 3 - Add details to the node, accordingly.



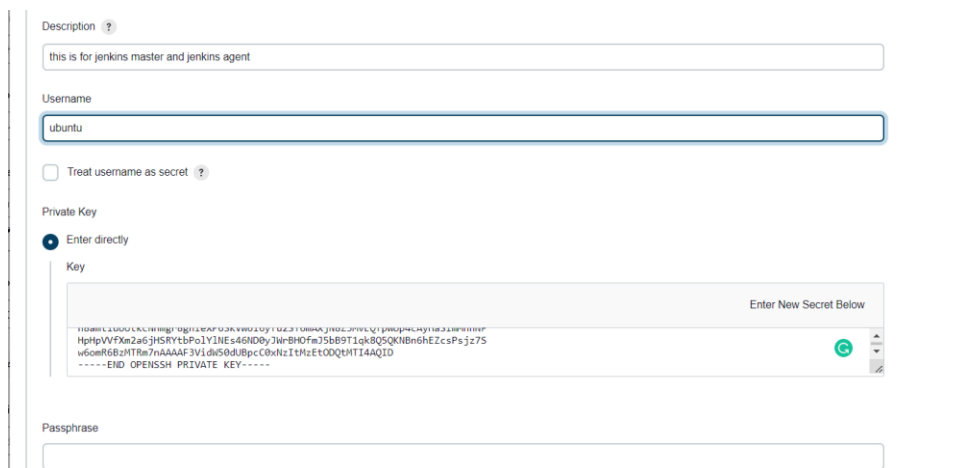
The image shows the Jenkins 'New Node' configuration page. The 'Name' field is set to 'Agent'. The 'Description' field contains 'This is agent'. The 'Number of executors' is set to '1'. The 'Remote root directory' is set to '/home/ubuntu'. The 'Labels' field is set to 'node-agent'. A 'Save' button is at the bottom.

Step 4 – In launch method select Launch agents via SSH and click on ADD Credentials



The image shows the 'Launch method' section of the Jenkins Node Configuration. The 'Launch method' is set to 'Launch agents via SSH'. The 'Host' field is set to '44.211.230.152'. The 'Credentials' dropdown is set to '- none -'. A red error message 'credentials cannot be found' is displayed. The 'Host Key Verification Strategy' is set to 'Known hosts file Verification Strategy'. The 'Availability' dropdown is set to 'Keep this agent online as much as possible'.

Step 5 – Give description according and past the private key that we created in 'jenkins-agent' instance using ssh-keygen.



The image shows the 'Private Key' section of the Jenkins Node Configuration. The 'Description' field contains 'this is for jenkins master and jenkins agent'. The 'Username' field is set to 'ubuntu'. The 'Private Key' section is set to 'Enter directly'. The 'Key' field contains a long string of characters representing a private key. The 'Passphrase' field is empty.

Step 6 - Below we select ubuntu for credentials and click on 'save' that will create node.

Credentials ?

ubuntu (this is for jenkins master and jenkins agent)

+ Add

Host Key Verification Strategy ?

Known hosts file Verification Strategy

Advanced ▾

Availability ?

Keep this agent online as much as possible

Node Properties

☐ Disable deferred wipeout on this node ?

☐ Environment variables

☐ Tool Locations

Save

- **Verify its status under "Nodes" section.**

The screenshot shows the Jenkins 'Manage nodes and clouds' page. On the left, there are links for 'Configure Clouds' and 'Node Monitoring'. Below these are sections for 'Build Queue' (showing 'No builds in the queue') and 'Build Executor Status' (showing 'Built-in Node' with '1 idle' and '2 idle' executors). The main table lists the following nodes:

| S | Name | Architecture | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Space | Response Time |
|---|---------------|---------------|------------------|-----------------|-----------------|-----------------|---------------|
| | Agent | | N/A | N/A | N/A | N/A | N/A |
| | Built-in Node | Linux (amd64) | In sync | 1.63 GB | 0 B | 1.63 GB | 0ms |
| | Data obtained | 0.31 sec | 0.31 sec | 0.31 sec | 0.3 sec | 0.3 sec | 0.3 sec |

Task-02

- Run your previous Jobs (which you built on Day 26, and Day 27) on the new agent
- Use labels for the agent, your master server should trigger builds for the agent server.

Step -1 Click on “New Item” and enter desired name and select “Freestyle project” and click on Ok.

The screenshot shows the 'Enter an item name' dialog box in Jenkins. The 'Item name' field contains 'Node-Pipeline'. Below the field, there are several project type options:

- 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.
- 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.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.

The 'OK' button is highlighted at the bottom left.

Step 2 – Give description, select GitHub project and enter your repository url from where you will clone all files of your project and add label expressions, in that add your node label that you created.

Dashboard > Node-Pipeline > Configuration

Configure Enabled

General

Description: This is Node-App

[Plain text] [Preview](#)

☐ Discard old builds

☒ GitHub project

Project url:

Advanced

☐ This project is parameterized

☐ Throttle builds

☐ Execute concurrent builds if necessary

☒ Restrict where this project can be run

Label Expression:

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

[Save](#) [Apply](#)

Step 3 – In build Steps select Execute shell and write commands to run , Click on save.

Build Steps

Execute shell

Command:

Advanced

[Add build step](#)

Post-build Actions

[Add post-build action](#)

[Save](#) [Apply](#)

Step 4 - Click on build and check your project console output.

```
Commit message: "update todo.vjs"
> git rev-list --no-walk 4b06932cd6367578eece5b5de71f16339119828 # timeout=10
[node-todo-delivery] $ /bin/sh -xe /tmp/jenkins1285513513208737247.sh
+ docker-compose down
Removing node-todo-delivery_web_1 ...
Removing node-todo-delivery_web_1 ... done
Removing network node-todo-delivery_default
+ docker-compose up -d
Creating network "node-todo-delivery_default" with the default driver
Creating node-todo-delivery_web_1 ...
Creating node-todo-delivery_web_1 ... done
Finished: SUCCESS
```

Past your instance public Ip in browse to check web app is running

← → 🔍 Not secure | 44.211.230.152:8000/todo

GNA Students are Super Duper Awesome

What should I do? [Add](#)

Happy Learning:)