

## Jenkins setup on AWS EC2 with Ubuntu instances.

We will be creating two Ec2 instances, 1 is master and second is slave machine. This is the test lab hence we will be using master VM with 4GB ram, in production environment we need to have good amount of ram, its better to have T2 large, xtra large, instance with multi AZ depending about the load, in case of multi region deployment and automation.

Here I have used the below link to download and setup Master instance, for slave machine we just need to have java installed with SSH key authentication so that master node can get authentication.

1. Login to AWS console and click on Launch Ec2 instance.

**Name and tags** [Info](#)

Name

Jenkinsmaster

[Add additional tags](#)

**Summary**

Number of instances

[Info](#)

2

When launching more than 1 instan

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Debian

debian

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-04b4f1a9cf54c11d0 (64-bit (x86)) / ami-0a7a4e87939439934 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

**Description**

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture

AMI ID

Username

[Info](#)

64-bit (x86)

ami-04b4f1a9cf54c11d0

ubuntu

Verified provider

2. We will be using Ubuntu AMI version – Ubuntu 24.04 LTS.

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Debian

debian

Q

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

#### Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type  
ami-04b4f1a9cf54c11d0 (64-bit (x86)) / ami-0a7a4e87939439934 (64-bit (Arm))  
Virtualization: hvm    ENA enabled: true    Root device type: ebs

Free tier eligible

#### Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

#### Architecture

64-bit (x86)

#### AMI ID

ami-04b4f1a9cf54c11d0

#### Username

ubuntu



Verified provider

#### Recents Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Debian

debian

Q

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

#### Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type  
ami-04b4f1a9cf54c11d0 (64-bit (x86)) / ami-0a7a4e87939439934 (64-bit (Arm))  
Virtualization: hvm    ENA enabled: true    Root device type: ebs

Free tier eligible

#### Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

#### Architecture

64-bit (x86)

#### AMI ID

ami-04b4f1a9cf54c11d0

#### Username

ubuntu



Verified provider

We need to create Keypair, we can login to Jenkins and perform steps to download and install Jenkins.  
We are creating Public key .pem

## Create key pair



### Key pair name

Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

### Key pair type



RSA

RSA encrypted private and public key pair



ED25519

ED25519 encrypted private and public key pair

### Private key file format



.pem

For use with OpenSSH



.ppk

For use with PuTTY

Security group we are allowing port 22. There are other ports we need to allow.

### Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.



Create security group



Select existing security group

We'll create a new security group called 'launch-wizard-7' with the following rules:



Allow SSH traffic from

Helps you connect to your instance

Anywhere

0.0.0.0/0



Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server



Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

Hard disk we are allowing 20 GB, in production environment it will be more. Since this is test lab environment, we are using 20GB- General purpose

### ▼ Configure storage [Info](#)

[Advanced](#)

1x

GiB

gp3



Root volume, 3000 IOPS, Not encrypted



Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage



We can now see Two instances are created and checks are passed. I have renamed the second instance as Jenkins slave so that we can differentiate.

<input type="checkbox"/>	Jenkinsmaster	i-075535fe1fde238b2	Running	t2.medium	2/2 checks passed	<a href="#">View alarms</a> +
<input checked="" type="checkbox"/>	Jenkinsslave	i-0d26316f98cb5933a	Running	t2.medium	2/2 checks passed	<a href="#">View alarms</a> +


We can now connect to our Ubuntu instance, using git bash.



EC2 Instance Connect


Session Manager

SSH client

EC2 serial console

**Instance ID**  
 i-075535fe1fde238b2 (Jenkinsmaster)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is jenkins.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
 `chmod 400 "jenkins.pem"`
4. Connect to your instance using its Public DNS:  
 `ec2-18-212-223-91.compute-1.amazonaws.com`

Example:  
 `ssh -i "jenkins.pem" ubuntu@ec2-18-212-223-91.compute-1.amazonaws.com`

We can launch Git bash and go to downloads folder. This is path where we have downloaded the public key.

```

MINGW64:/c:/Users/ALLTECHACC/Downloads
ALLTECHACC@Santopc MINGW64 ~ (master)
$ cd ~/Downloads/
ALLTECHACC@Santopc MINGW64 ~/Downloads (master)
$ |

```

```
MINGW64:/c/Users/ALLTECHACC/Downloads
ALLTECHACC@Santopc MINGW64 ~ (master)
$ cd ~/Downloads/

ALLTECHACC@Santopc MINGW64 ~/Downloads (master)
$ ssh -i "jenkins.pem" ubuntu@ec2-18-212-223-91.compute-1.amazonaws.com
The authenticity of host 'ec2-18-212-223-91.compute-1.amazonaws.com (18.212.223.91)' can't be established.
ED25519 key fingerprint is SHA256:q5QGcEvDTE699v4XfSrY6d9UM8B3c0JsIQR1Xk++G3I.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes|
```

When we get prompt to add the key, we can add the key, type yes, and it will use the key and login to server. We have now logged in successfully to the server.

```
0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-19-251:~$ uptime
 21:56:36 up 27 min,  1 user,  load average: 0.02, 0.01, 0.00
```

We now need to perform the steps as given in the website, for ubuntu machine.

Below is the link for installing jenkins in the master node.

<https://www.jenkins.io/doc/book/installing/linux/#debianubuntu>

## Master node configuration

1. Log into the master node and perform the steps as mentioned in the document.

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins

sudo apt update
sudo apt install fontconfig openjdk-17-jre
java -version
openjdk version "17.0.13" 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Debian-2)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Debian-2, mixed mode, sharing)
```

**We can also perform the automation of the above task by saving it in shell script file and running the script file.**

After performing the above steps on ubuntu machine, we have successfully installed the jenkins on master node.

Now we need to start the services. Below are the steps taken from the website to start the services.

## Start Jenkins

You can enable the Jenkins service to start at boot with the command:

```
sudo systemctl enable jenkins
```

You can start the Jenkins service with the command:

```
sudo systemctl start jenkins
```

You can check the status of the Jenkins service using the command:

```
sudo systemctl status jenkins
```

After performing the above steps, we have now successfully started the services.

```

ubuntu@ip-172-31-19-251:~$ sudo systemctl start jenkins
ubuntu@ip-172-31-19-251:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Tue 2025-02-25 22:15:55 UTC; 10s ago
     Main PID: 5017 (java)
       Tasks: 52 (limit: 4676)
      Memory: 622.4M (peak: 634.5M)
         CPU: 15.550s
    CGroup: /system.slice/jenkins.service
            └─5017 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Feb 25 22:15:52 ip-172-31-19-251 jenkins[5017]: 3afc818feb5c46a390b71bb52a36a62
Feb 25 22:15:52 ip-172-31-19-251 jenkins[5017]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Feb 25 22:15:52 ip-172-31-19-251 jenkins[5017]: *****
Feb 25 22:15:52 ip-172-31-19-251 jenkins[5017]: *****
Feb 25 22:15:52 ip-172-31-19-251 jenkins[5017]: *****
Feb 25 22:15:55 ip-172-31-19-251 jenkins[5017]: 2025-02-25 22:15:55.648+0000 [id=30] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
Feb 25 22:15:55 ip-172-31-19-251 jenkins[5017]: 2025-02-25 22:15:55.666+0000 [id=23] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Feb 25 22:15:55 ip-172-31-19-251 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Feb 25 22:15:55 ip-172-31-19-251 jenkins[5017]: 2025-02-25 22:15:55.941+0000 [id=48] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Feb 25 22:15:55 ip-172-31-19-251 jenkins[5017]: 2025-02-25 22:15:55.942+0000 [id=48] INFO hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1



```

Now need to enable port 8080 and http port from security group to allow access to jenkins.  
We need to go to AWS console and make the necessary changes.

Inbound rules <a href="#">Info</a>						
Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>		
sgr-038d26457e9537eac	SSH	TCP	22	Custom	<input type="text" value="0.0.0.0/0"/>	<input type="button" value="X"/>
-	Custom TCP	TCP	8080	Anyw...	<input type="text" value="0.0.0.0/0"/>	<input type="button" value="X"/>
-	HTTP	TCP	80	My IP	<input type="text" value="0.0.0.0/0"/>	<input type="button" value="X"/>

After performing the above steps we are now able to access the jenkins url.  
We need to use Public ip of our Ec2 instance.

#### Public IPv4 address

 18.212.223.91 | [open address](#) 

## 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**

We now need to get the Admin password. We can get the password from the above path.

We need to use `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`

```
ubuntu@ip-172-31-19-251:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
3afc818feb5c46a390b71bbb52a36a62
```

We just need to copy and paste it should login.

# 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**



We are now successfully logged into Jenkins.

# Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

Select the option Install suggested plugins, these will be installed.

Getting Started

## Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** Ionicons API
✓ Timestampers	Workspace Cleanup	Ant	Gradle	Folders
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline Graph View	OWASP Markup Formatter
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication	** ASM API
LDAP	Email Extension	Mailer	Dark Theme	** JSON Path API
				** Struts
				** Pipeline: Step API
				** Token Macro
				Build Timeout
				** bouncycastle API
				** Credentials
				** Plain Credentials
				** Variant
				** SSH Credentials
				Credentials Binding
				** SCH API
				** Pipeline: API
				** commons-lang3 v3.x Jenkins API
				Timestampers
				** Caffeine API
				** Script Security
				** JavaBeans Activation Framework (JAF) API
				** JAXB
				** SnakeYAML API
				** - required dependency

Plugins are now installed and we are now creating user in Jenkins, this is the admin user.

---

## Getting Started

---

# Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

We can now configure the Jenkins url, we can leave it as default. Click on Save and finish

## Getting Started

---

# Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins Master is now successfully installed.

## Getting Started

# Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

We will now move on Slave configuration.

### **Slave Node/Worker node configuration.**

On slave node, we only need to have java installed, we will be installing Java on ubuntu machine.

We have successfully logged in to the Ubuntu machine.

```
0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-23-162:~$ uptime
16:42:20 up 26 min, 1 user, load average: 0.00, 0.00, 0.00
```

Referring to the link we have downloaded and installed Java

```
sudo apt update
sudo apt install fontconfig openjdk-17-jre
java -version
openjdk version "17.0.13" 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Debian-2)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Debian-2, mixed mode, sharing)
```

Java is now successfully installed.

```
ubuntu@ip-172-31-23-162:~$ java --version
openjdk 17.0.14 2025-01-21
OpenJDK Runtime Environment (build 17.0.14+7-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 17.0.14+7-Ubuntu-124.04, mixed mode, sharing)
ubuntu@ip-172-31-23-162:~$ |
```

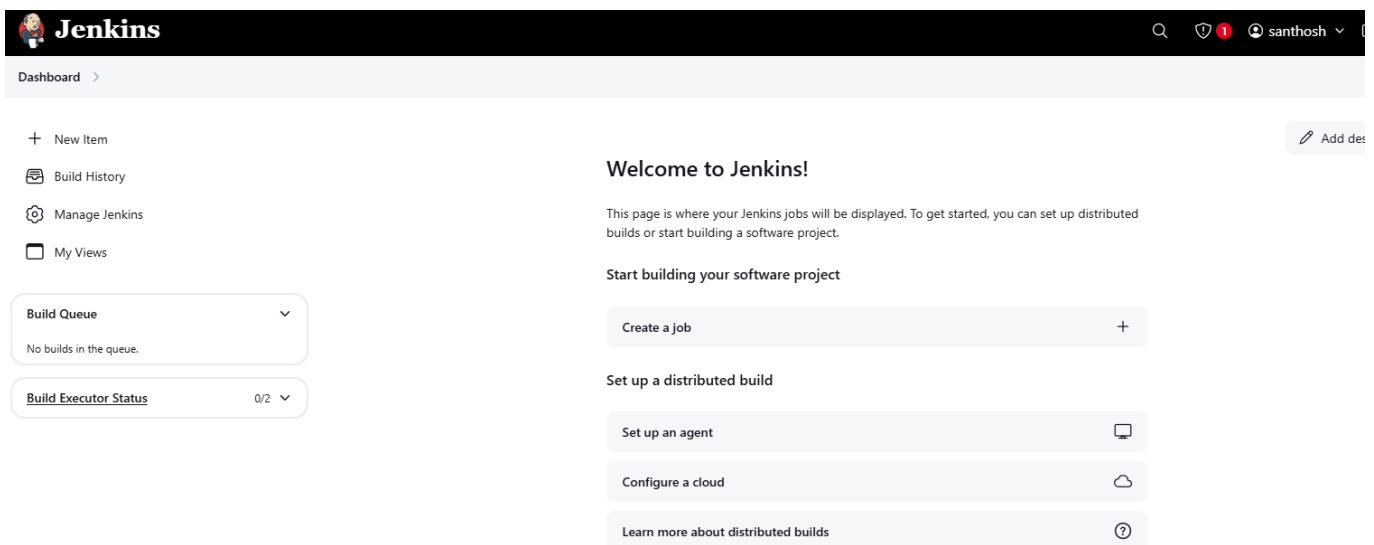
We need to create remote root directory.

We can go to home directory of ubuntu user and create new directory. I have created directory **jenkins**

```
ubuntu@ip-172-31-23-162:~$ cd /home/ubuntu/
ubuntu@ip-172-31-23-162:~$ mkdir jenkins
```

```
ubuntu@ip-172-31-23-162:~$ cd /home/ubuntu/
ubuntu@ip-172-31-23-162:~$ mkdir jenkins
```

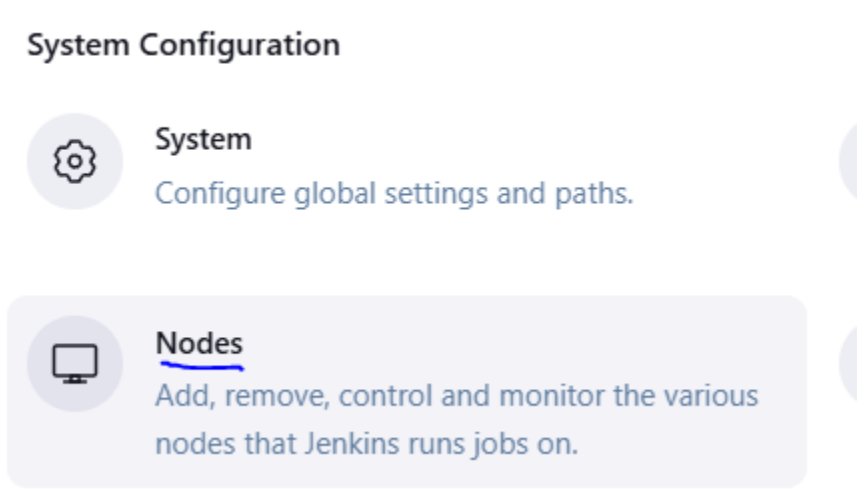
We need to login to Jenkins with the credential that we initially created.

The screenshot shows the Jenkins web interface. At the top is a black header with the Jenkins logo and name on the left, and search, notifications, and user profile icons on the right. Below the header is a light blue navigation bar with a 'Dashboard' link. The main content area is divided into two columns. The left column contains a sidebar with links: '+ New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are two status boxes: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing '0/2'. The right column features a 'Welcome to Jenkins!' message, followed by instructions on how to get started. It includes a 'Start building your software project' section with a 'Create a job' button, and a 'Set up a distributed build' section with buttons for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'.

Click on Manage Jenkins



Click on Nodes





Here we can see our main master node showing details of drive information



Click on New node. This is on top right hand corner

## Nodes

[+ New Node](#)[Configure Monitor](#)

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	<a href="#">Built-In Node</a>	Linux (amd64)	In sync	15.22 GiB	 0 B	15.22 GiB	0n
	last checked	9 min 58 sec	9 min 58 sec	9 min 58 sec	9 min 58 sec	9 min 58 sec	9 min 58 s

We are giving name **ubuntuworkernode** – Click create

## New node

Node name

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.

Create

We can update the number of executors here, I have given 5, this is the maximum number of build that this node can perform. This can vary depending on load and type of build.

Name ?

Description ?

Plain text [Preview](#)

Number of executors ?

We had created directory for the jenkins we can update the directory name.

**/home/ubuntu/jenkins**

Remote root directory ?

/home/ubuntu/jenkins

An agent needs to have a directory dedicated to Jenkins. Specify the path to this directory on the agent. It is best to use an absolute path, such as /var/jenkins or c:\jenkins . This should be a path local to the agent machine. There is no need for this path to be visible from the controller.

We need to label the node, in case of multiple nodes we can specify the labels to that it will be deployed on that node.

Labels ?

node1

Labels (or tags) are used to group multiple agents into one logical group.

We can specify the option usage **“Use this node as much as possible”**

Second option is **“Only build jobs with label expression matching this node”**

This will be selected in case of multi node and multi deployment environment since we have only 1 node we will be selecting use this node as much as possible.

Usage ?

Use this node as much as possible

Launch method ?

Launch agents via SSH

We can select Launch agents via SSH. This is launch method we will be selecting. This is secure way of having worker node.

We need to specify the host name, we will be specifying the host name of **SLAVE node** from our AWS Public IP

## Instance summary for i-0d26316f98cb5933a (Jenkinslave) [Info](#)



Updated less than a minute ago

### Instance ID


 i-0d26316f98cb5933a

### IPv6 address

### Public IPv4 address

 3.89.137.132 | [open address](#) 

### Instance state


Host 

3.89.137.132

We need to add the credentials, this is very important part . Click on Add button.

Credentials 

- none -

 Add

We need to select SSH username and private key

## Jenkins Credentials Provider: Jenkins

### Add Credentials

Domain

Global credentials (unrestricted)

Kind

SSH Username with private key

Scope 

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



Since we are using ssh connection, we need to setup public and private key.

We now go to our slave node of ubuntu and run the command **ssh-keygen**

```
ubuntu@ip-172-31-23-162:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519): |
```

Press Enter key, keep the default value.

Press Enter key again.

```
ubuntu@ip-172-31-23-162:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:YGMbHsJtLzg6vr7uWsdS2mvDO5wtJsINAejyePkS7so ubuntu@ip-172-31-23-162
The key's randomart image is:
--[ED25519 256]--+
|
|o . .
|o o X
|.o B B
|.o..+ + S
|..=* .
|.oO=+o
|o++=@..
|oEOBo=
|
+----[SHA256]-----+
```

So our keys are saved in /home/ubuntu.ssh

cd /home/ubuntu/.ssh/

```
+----[SHA256]-----+
ubuntu@ip-172-31-23-162:~$ cd /home/ubuntu/.ssh/
ubuntu@ip-172-31-23-162:~/.ssh$ ls
authorized_keys id_ed25519 id_ed25519.pub
```

Now we need to copy the private key to the Jenkins portal.

Use the cat command

```
ubuntu@ip-172-31-23-162:~/.ssh$ cat id_ed25519
```

We need to update the details in private key

Select below options in the credential section.

Username should be the username which used to login to slave node

In case if you are using redhat you need to specify root if you are using default redhat login.

Here I am giving ubuntu. Since I have logged in as ubuntu.

### Jenkins Credentials Provider: Jenkins

Scope ?

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

ID ?

jenkinslave

Description ?

Devopstest1

Username

ubuntu

Select the option enter directly and enter the key details. Click Add

Private Key

☒ Enter directly

Key

No Stored Value

Add

Copy and paste the key from ubuntu path where your private key is saved. Make sure there are no additional spaces.

☒ Enter directly

Key

```
AAAEB0QBefgahcXvs8+1tItXpw5j8Pe32khbjMoIjPgGT5QAm9Guyd3U3/YdY10XwVrZ74  
sed7wLke+7CucTwrhaQeAAAAF3VidW50dUBpcC0xNzItMzEtMjMtMTYyAQIDBAUG  
-----END OPENSSH PRIVATE KEY-----|
```

If passphrase is given, mention the passphrase, since we have not given any passphrase, I have left it blank.

☒ Enter directly

Key

```
AAAEB0QBefgahcXvs8+1tItXpw5j8Pe32khbjMoIjPgGT5QAm9Guyd3U3/YdY10XwVrZ74
sed7wLke+7CucTwrhaOeAAAAF3VidW50dUBpcC0xNzItMzEtMjMtMTYyAQIDBAUG
-----END OPENSSH PRIVATE KEY-----
```

Passphrase

Click Add button to add the credential.

Select the credential which we have configured.

Host ?

3.89.137.132

Credentials ?

ubuntu (Devopstest1)

Host key Verification strategy. Since we are using test environment we can use non verifying strategy.

**The Non-Verifying Host Key Verification Strategy in Jenkins is an approach where Jenkins does not perform any verification of the host key during the SSH connection. Essentially, this strategy bypasses the standard security checks, which can be useful for testing or in environments**

#### **Manually Provided Key Verification Strategy:**

- This strategy checks if the key provided by the remote host matches the key set by the user who configured the connection.

#### **Manually Trusted Key Verification Strategy:**

- This strategy checks if the remote key matches the key currently marked as trusted for this host.

- Depending on the configuration, the key will be automatically trusted for the first connection, or an authorized user will be asked to approve the key.

Since this is test environment we are using non Verifying Verification Strategy.

#### Host Key Verification Strategy ?

Controls how Jenkins verifies the SSH key presented by the remote host whilst connecting.

Non verifying Verification Strategy

Now we have added the key and created credential, we are now good to save.

#### Availability ?

Keep this agent online as much as possible

#### Node Properties

- ☐ Disable deferred wipeout on this node ?
- ☐ Disk Space Monitoring Thresholds
- ☐ Environment variables
- ☐ Tool Locations

Save

- We can now see the node is added however this is not yet completed and **donot** launch the node.

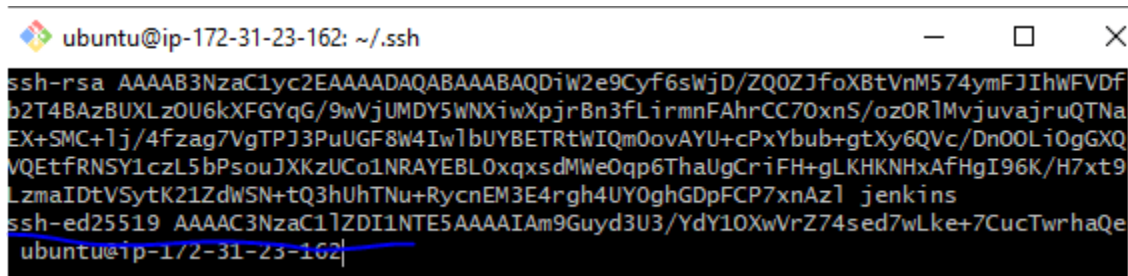
We now need to add the public key on the same node .

Cat the public key

```
cat id_ed25519.pub
```

use the vi editor to add the key

```
vi authorized_keys
```



A terminal window titled 'ubuntu@ip-172-31-23-162: ~/.ssh' with standard window controls. The terminal displays the contents of the 'authorized\_keys' file. It shows two entries: an 'ssh-rsa' key and an 'ssh-ed25519' key. The 'ssh-ed25519' key is preceded by a blue underline. The terminal text is as follows:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDiW2e9Cyf6swjD/ZQ0ZJfoXBtVnM574ymFJIhWFVDf
b2T4BAzBUXLz0U6kXFGYqG/9wVjUMDY5WNXiWxpjrBn3fLirmnFAhrCC70xnS/ozORlMvjuvajruQTNa
EX+SMC+lj/4fzag7VgTPJ3PuUGF8W4IwlbUYBETRtWlQm0ovAYU+cPxYbub+gtXy6QVc/Dn00Li0gGXQ
VQEtFRNSY1czL5bPsouJXKzUCo1NRAYEBL0xqxsdMWe0qp6ThaUgCriFH+gLKHKNHxAfHgI96K/H7xt9
LzmaIDtVSytK21ZdWSN+tQ3hUhtNu+RycnEM3E4rgh4UY0ghGDpFCP7xnAzl jenkins
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAm9Guyd3U3/YdY10XwVrZ74sed7wLke+7CucTwrhaQe
ubuntu@ip-172-31-23-162|
```

We need to click on Launch node and Now our slave node is online.

```

TERM=dumb
UID=1000
USER=ubuntu
XDG_RUNTIME_DIR=/run/user/1000
XDG_SESSION_CLASS=user
XDG_SESSION_ID=26
XDG_SESSION_TYPE=ttty
_='']'
[02/26/25 18:55:57] [SSH] Starting sftp client.
[02/26/25 18:55:57] [SSH] Copying latest remoting.jar...
[02/26/25 18:55:57] [SSH] Copied 1,395,562 bytes.
Expanded the channel window size to 4MB
[02/26/25 18:55:57] [SSH] Starting agent process: cd "/home/ubuntu/jenkins" && java -jar remoting.jar -w
/home/ubuntu/jenkins/remoting/jarCache
Feb 26, 2025 6:55:57 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/jenkins/remoting as a remoting work directory
Feb 26, 2025 6:55:57 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/jenkins/remoting
<===[JENKINS REMOTING CAPACITY]===>channel started
Remoting version: 3283.v92c105e0f819
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
Agent successfully connected and online

```

when we click on nodes, we can see both are in sync

## Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space
	<a href="#">Built-In Node</a>	Linux (amd64)	In sync	15.35 GiB
	<a href="#">ubuntuworkernode</a>	Linux (amd64)	In sync	15.79 GiB
	<b>last checked</b>	<b>2 min 47 sec</b>	<b>2 min 47 sec</b>	<b>2 min 47 sec</b>

we will now run a small test job that we can run on server.

Click on Dashboard – Click on + sign to create new job

## Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can create a new job, view existing builds or start building a software project.

Start building your software project

Create a job

Type as testing jenkins- Select Free style project. Click ok

### NEW ITEM

Enter an item name

Testing jenkins |

Select an item type



**Freestyle project**

Classic, general-purpose job type that checks out from up to one SCM, executes build steps like archiving artifacts and sending email notifications.

Select Build steps select option Execute Shell.

Here we can execute commands.

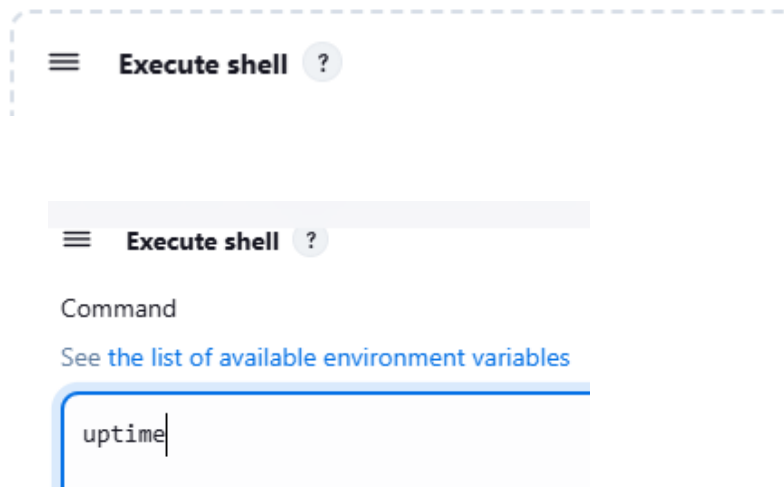
For this job we are selecting **uptime**

Click Save

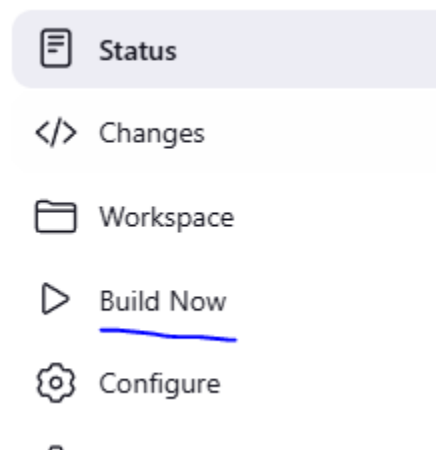


## Build Steps

Automate your build process with ordered tasks like code comp



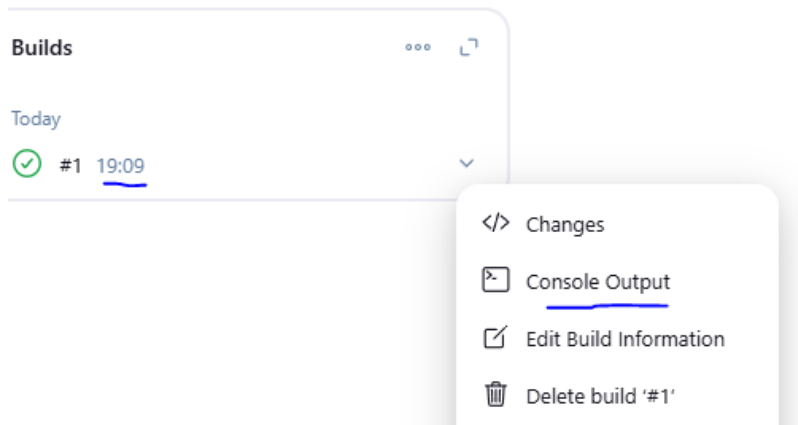
After clicking on Save, we can click **Build Now** option



We can now see our job is successful and node is working fine.

We can click on console output to see the status.

Console is now  
showing good and  
shows uptime of  
server



## Console Output

```
Started by user santhosh
Running as SYSTEM
Building on the built-in node in workspace /var/lib/jenkins/workspace/Testing jenkins
[Testing jenkins] $ /bin/sh -xe /tmp/jenkins14812687760815393334.sh
+ uptime
 19:09:35 up  2:53,  2 users,  load average: 0.00, 0.01, 0.00
Finished: SUCCESS
```

**END of Document**