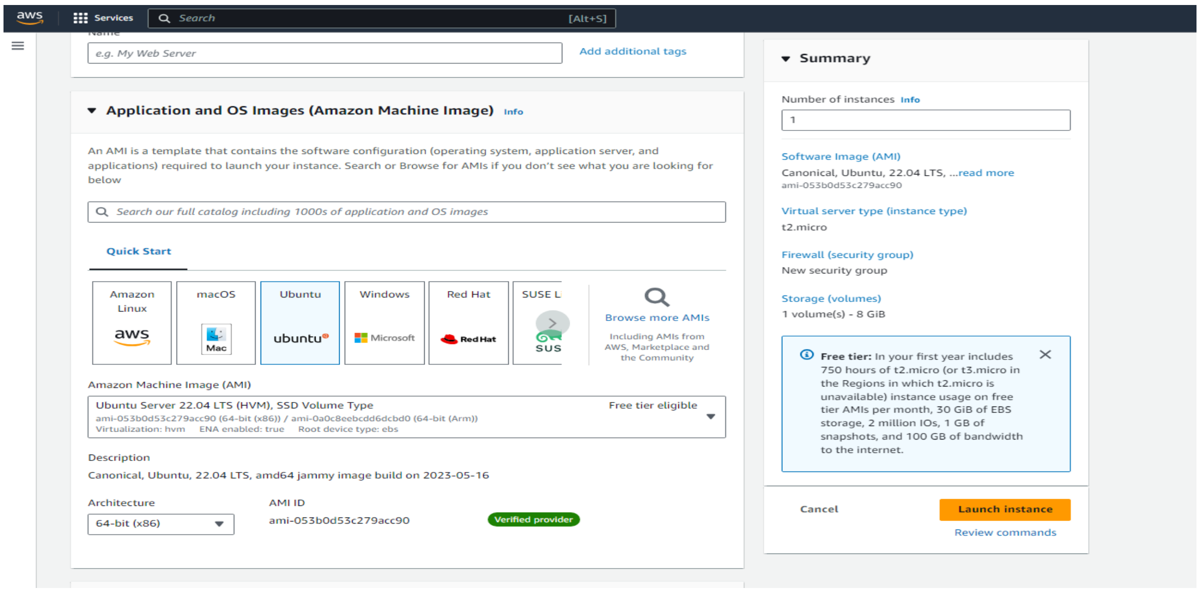
First we need to set up 3 servers instance in amazon EC2 one as Docker Master and other two servers as workers with ubuntu 22.04 LTS as AMI we create 3 instances at a time.


Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Amazon Machine Image (AMI)
Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-053b0d53c279acc90 (64-bit (x86)) / ami-0a0c8eebcdd6dcbd0 (64-bit (Arm))
Virtualization: hvm    ENA enabled: true    Root device type: ebs

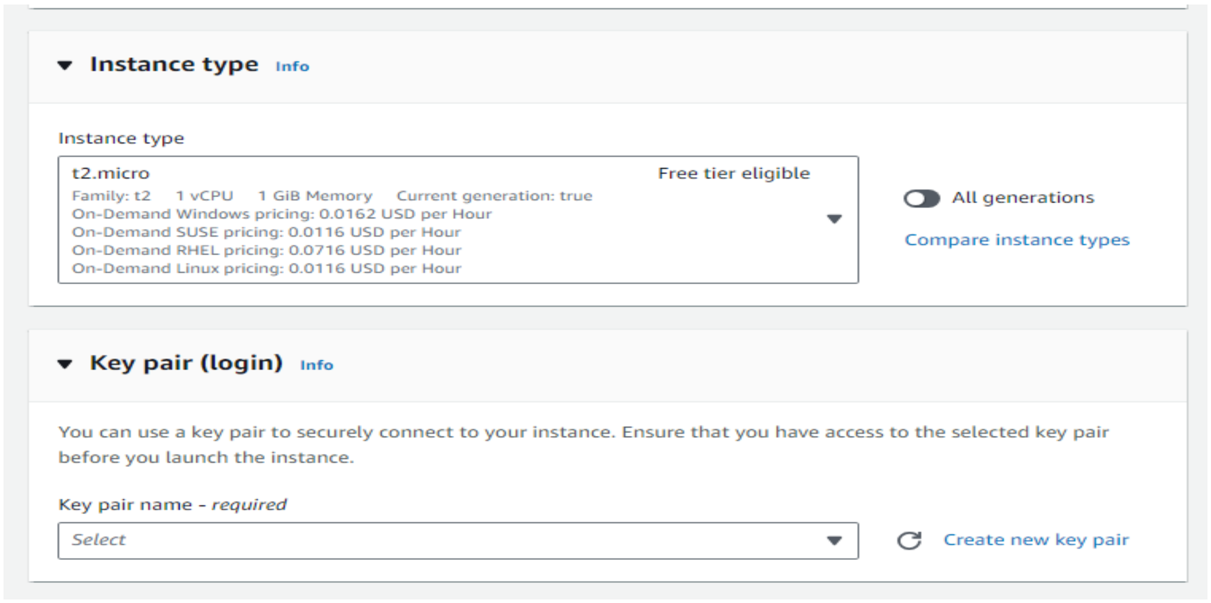
Description
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-05-16

Architecture: 64-bit (x86)
AMI ID: ami-053b0d53c279acc90
Verified provider

Summary
Number of instances: 1
Software Image (AMI): Canonical, Ubuntu, 22.04 LTS, ...read more ami-053b0d53c279acc90
Virtual server type (instance type): t2.micro
Firewall (security group): New security group
Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

With t2.micro as instance type. We need add key pair login if existed or create new key pair a pem file will be downloaded in our system.


Instance type

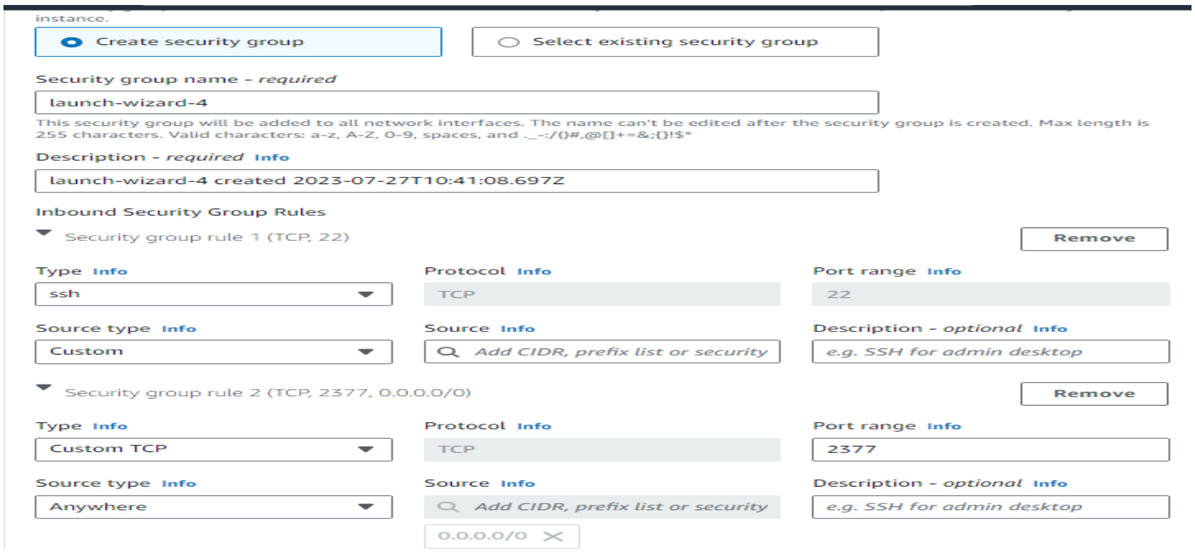
Instance type
t2.micro — Free tier eligible
Family: t2    1 vCPU    1 GiB Memory    Current generation: true
On-Demand Windows pricing: 0.0162 USD per Hour
On-Demand SUSE pricing: 0.0116 USD per Hour
On-Demand RHEL pricing: 0.0716 USD per Hour
On-Demand Linux pricing: 0.0116 USD per Hour

Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*
*Select*

We need to add TCP 2377 port for cluster Management. TCP 7946 and UDP7946 to be added for access. UDP 4789 to have worker server access for deployment. In the below screenshots we can see ports that we added in security group:


instance.

Create security group    Select existing security group

Security group name - *required*

launch-wizard-4

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* Info

launch-wizard-4 created 2023-07-27T10:41:08.697Z

Inbound Security Group Rules

Security group rule 1 (TCP, 22)

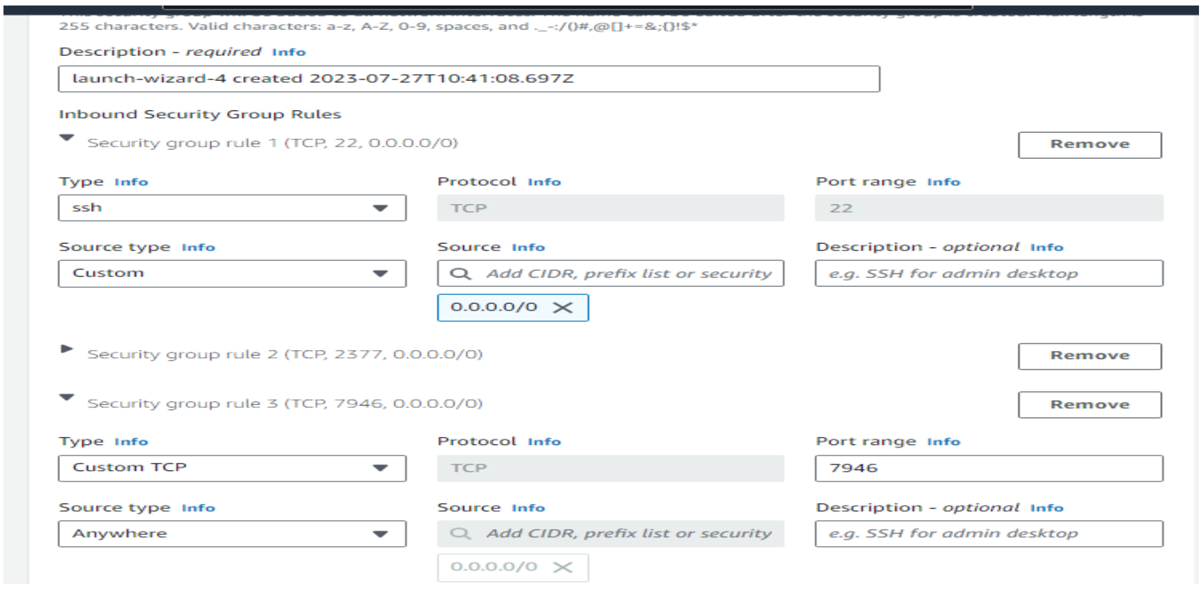
| Type | Protocol | Port range |
|---|---|---|
| ssh | TCP | 22 |

| Source type | Source | Description - *optional* |
|---|---|---|
| Custom | Add CIDR, prefix list or security | e.g. SSH for admin desktop |

Security group rule 2 (TCP, 2377, 0.0.0.0/0)

| Type | Protocol | Port range |
|---|---|---|
| Custom TCP | TCP | 2377 |

| Source type | Source | Description - *optional* |
|---|---|---|
| Anywhere | Add CIDR, prefix list or security<br>0.0.0.0/0 | e.g. SSH for admin desktop |


255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* Info

launch-wizard-4 created 2023-07-27T10:41:08.697Z

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

| Type | Protocol | Port range |
|---|---|---|
| ssh | TCP | 22 |

| Source type | Source | Description - *optional* |
|---|---|---|
| Custom | Add CIDR, prefix list or security<br>0.0.0.0/0 | e.g. SSH for admin desktop |

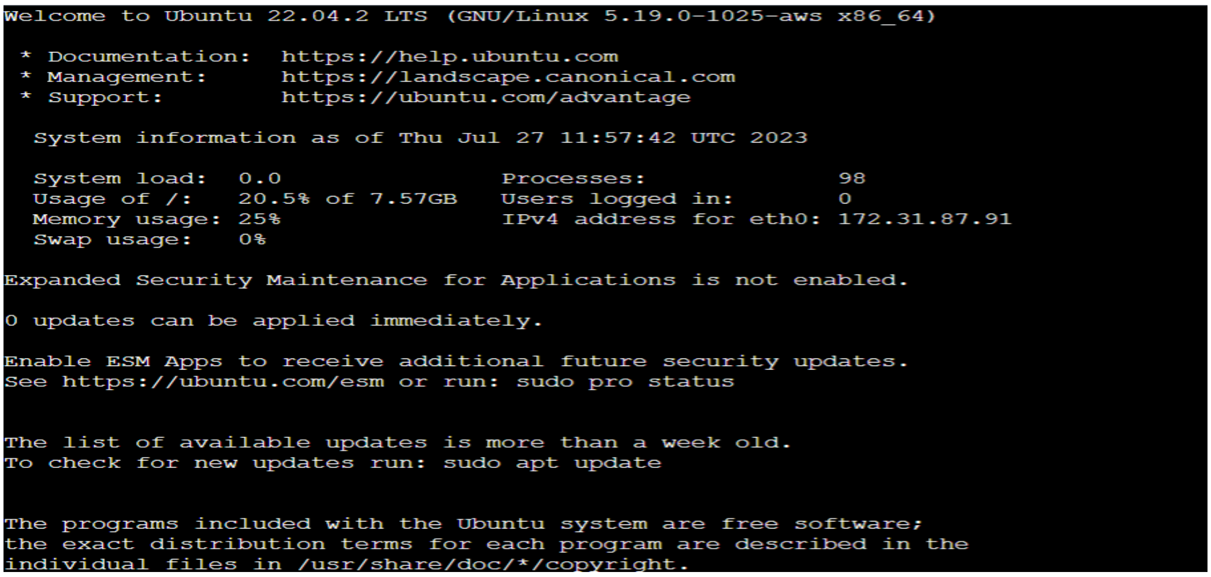
Security group rule 2 (TCP, 2377, 0.0.0.0/0)

Security group rule 3 (TCP, 7946, 0.0.0.0/0)

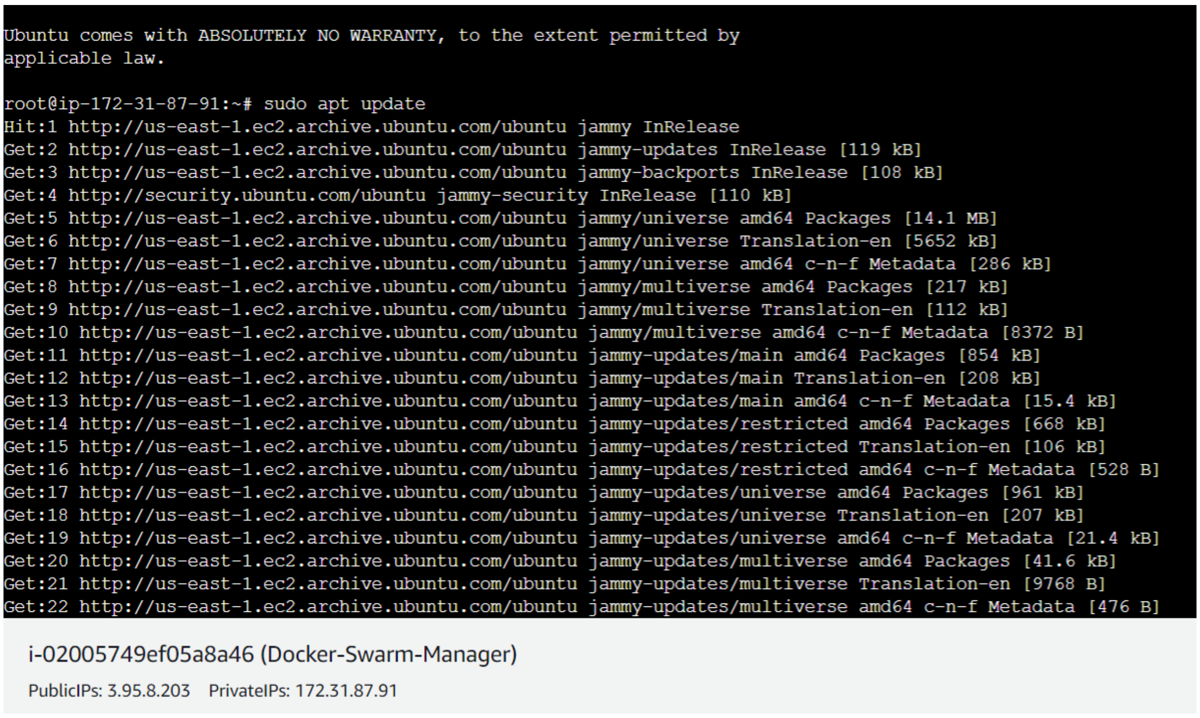
| Type | Protocol | Port range |
|---|---|---|
| Custom TCP | TCP | 7946 |

| Source type | Source | Description - *optional* |
|---|---|---|
| Anywhere | Add CIDR, prefix list or security<br>0.0.0.0/0 | e.g. SSH for admin desktop |

After that we connect to master instance we use to run command sudo apt update in ubuntu user.


```
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Jul 27 11:57:42 UTC 2023

  System load:  0.0               Processes:             98
  Usage of /:   20.5% of 7.57GB   Users logged in:       0
  Memory usage: 25%               IPv4 address for eth0: 172.31.87.91
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

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.
```

Here we can see the packages are being updated.

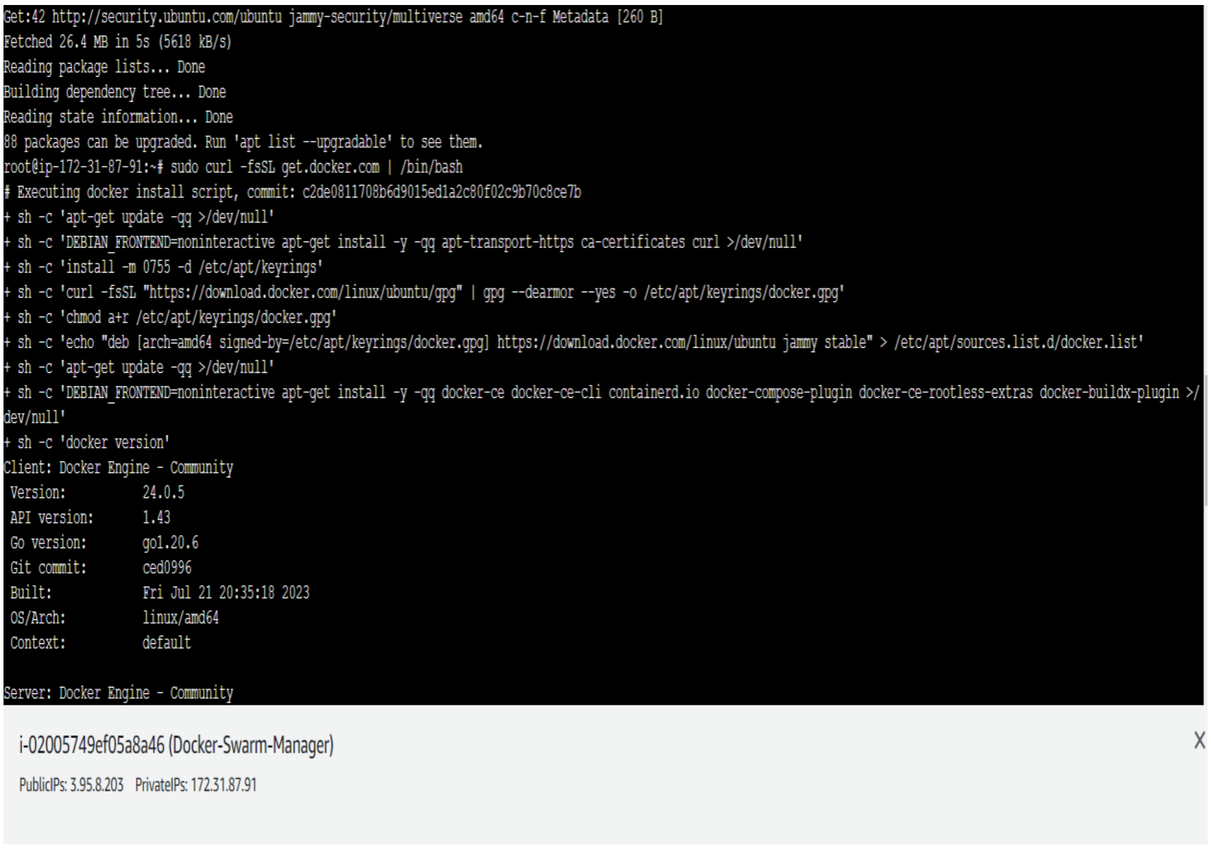


```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ip-172-31-87-91:~# sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [854 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [208 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [15.4 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [668 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [106 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [528 B]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [961 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [207 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [21.4 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [41.6 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [9768 B]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [476 B]
```

i-02005749ef05a8a46 (Docker-Swarm-Manager)

PublicIPs: 3.95.8.203    PrivateIPs: 172.31.87.91

After that we need to install docker in manager ubuntu user using command sudo curl -fsSL get.docker.com | /bin/bash

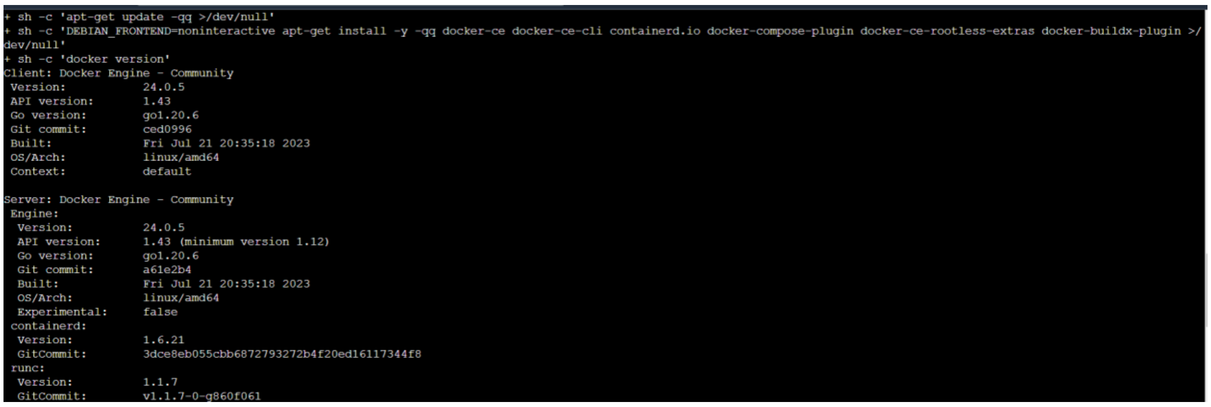


```
Get:42 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]
Fetched 26.4 MB in 5s (5618 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
88 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-87-91:~# sudo curl -fsSL get.docker.com | /bin/bash
# Executing docker install script, commit: c2de0811708b6d9015ed1a2c80f02c9b70c8ce7b
+ sh -c 'apt-get update -qq >/dev/null'
+ sh -c 'DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null'
+ sh -c 'install -m 0755 -d /etc/apt/keyrings'
+ sh -c 'curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | gpg --dearmor --yes -o /etc/apt/keyrings/docker.gpg'
+ sh -c 'chmod a+r /etc/apt/keyrings/docker.gpg'
+ sh -c 'echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu jammy stable" > /etc/apt/sources.list.d/docker.list'
+ sh -c 'apt-get update -qq >/dev/null'
+ sh -c 'DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-ce-rootless-extras docker-buildx-plugin >/
dev/null'
+ sh -c 'docker version'
Client: Docker Engine - Community
 Version:           24.0.5
 API version:       1.43
 Go version:        go1.20.6
 Git commit:        ced0996
 Built:             Fri Jul 21 20:35:18 2023
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
```

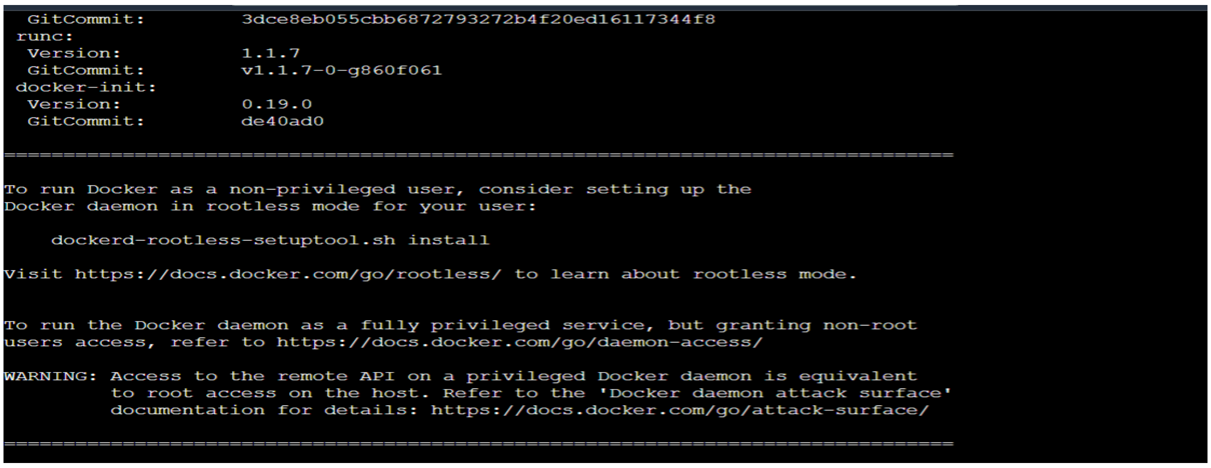
i-02005749ef05a8a46 (Docker-Swarm-Manager)

PublicIPs: 3.95.8.203    PrivateIPs: 172.31.87.91

We can see docker script is being ready and we need to wait for some time installation of docker.

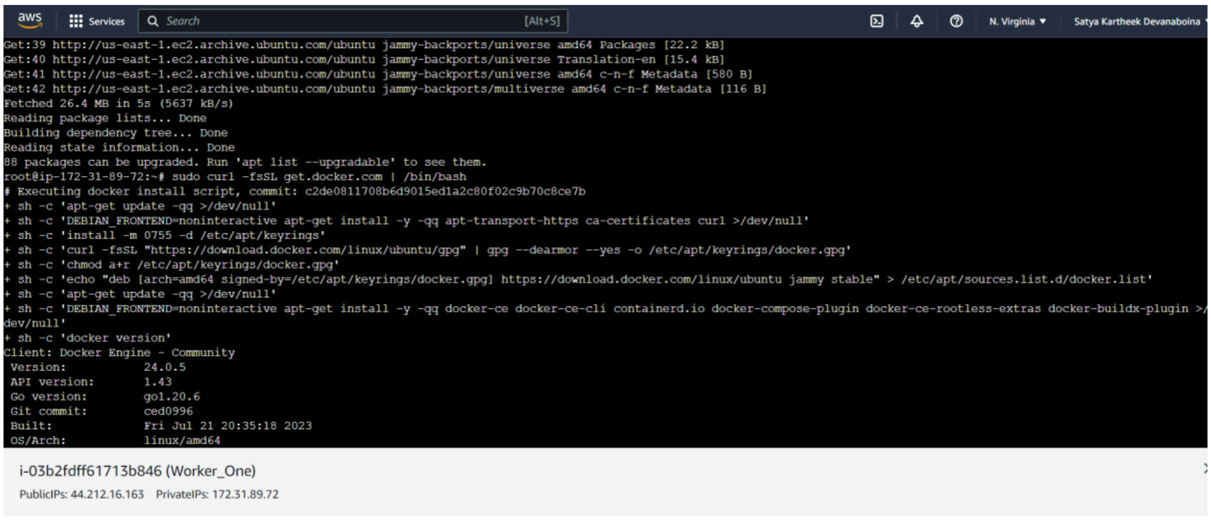


```
+ sh -c 'apt-get update -qq >/dev/null'
+ sh -c 'DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-ce-rootless-extras docker-buildx-plugin >/
dev/null'
+ sh -c 'docker version'
Client: Docker Engine - Community
 Version:           24.0.5
 API version:       1.43
 Go version:        go1.20.6
 Git commit:        ced0996
 Built:             Fri Jul 21 20:35:18 2023
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          24.0.5
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.6
  Git commit:       a61e2b4
  Built:            Fri Jul 21 20:35:18 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.21
  GitCommit:        3dce8eb055cbb6872793272b4f20ed16117344f8
 runc:
  Version:          1.1.7
  GitCommit:        v1.1.7-0-g860f061
```



```
  GitCommit:        3dce8eb055cbb6872793272b4f20ed16117344f8
 runc:
  Version:          1.1.7
  GitCommit:        v1.1.7-0-g860f061
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0

================================================================================

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

    dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.


To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
         to root access on the host. Refer to the 'Docker daemon attack surface'
         documentation for details: https://docs.docker.com/go/attack-surface/

================================================================================
```

We need to install docker in worker ubuntu user by connecting to ec2 instance in AWS console.

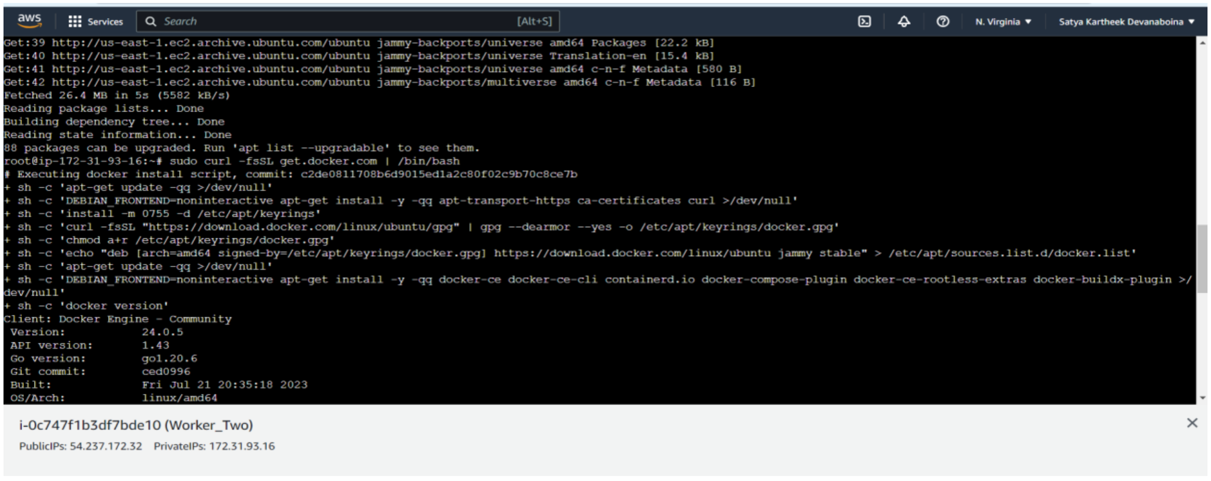


```
Get:39 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [22.2 kB]
Get:40 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [15.4 kB]
Get:41 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [580 B]
Get:42 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 26.4 MB in 5s (5637 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
88 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-89-72:~# sudo curl -fsSL get.docker.com | /bin/bash
# Executing docker install script, commit: c2de0811708b6d9015ed1a2c80f02c9b70c8ce7b
+ sh -c 'apt-get update -qq >/dev/null'
+ sh -c 'DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null'
+ sh -c 'install -m 0755 -d /etc/apt/keyrings'
+ sh -c 'curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | gpg --dearmor --yes -o /etc/apt/keyrings/docker.gpg'
+ sh -c 'chmod a+r /etc/apt/keyrings/docker.gpg'
+ sh -c 'echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu jammy stable" > /etc/apt/sources.list.d/docker.list'
+ sh -c 'apt-get update -qq >/dev/null'
+ sh -c 'DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-ce-rootless-extras docker-buildx-plugin >/
dev/null'
+ sh -c 'docker version'
Client: Docker Engine - Community
 Version:           24.0.5
 API version:       1.43
 Go version:        go1.20.6
 Git commit:        ced0996
 Built:             Fri Jul 21 20:35:18 2023
 OS/Arch:           linux/amd64
```

i-03b2fdff61713b846 (Worker_One)

PublicIPs: 44.212.16.163 PrivateIPs: 172.31.89.72

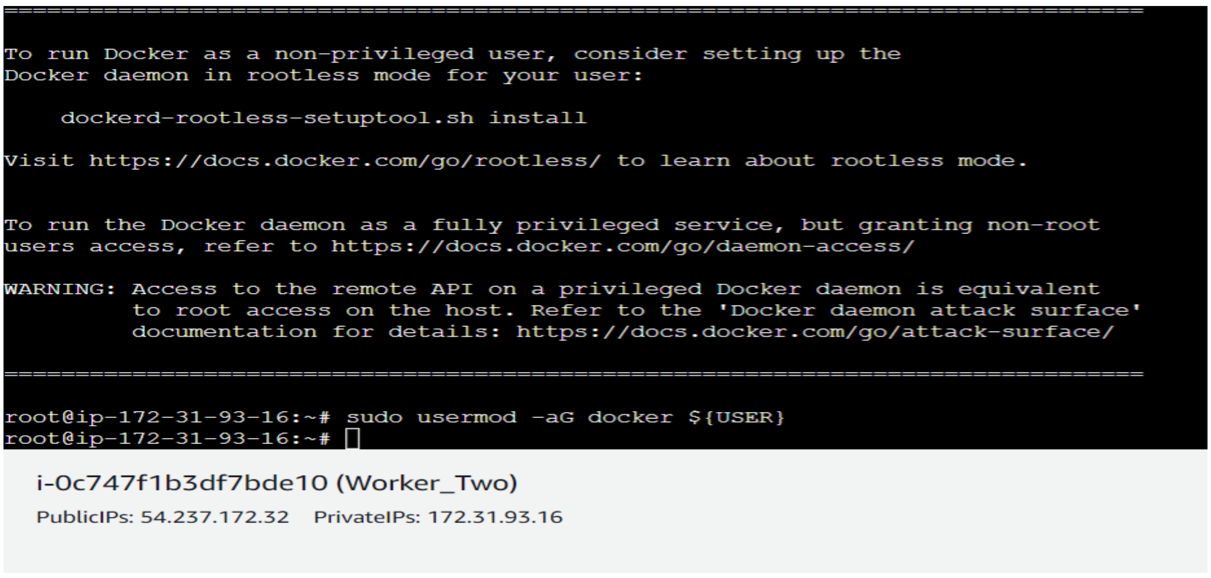
Here we can see that docker installed in worker user also.



```
Get:39 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [22.2 kB]
Get:40 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [15.4 kB]
Get:41 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [580 B]
Get:42 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 26.4 MB in 5s (5582 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
88 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-93-16:~# sudo curl -fsSL get.docker.com | /bin/bash
# Executing docker install script, commit: c2de0811708b6d9015ed1a2c80f02c9b70c8ce7b
+ sh -c 'apt-get update -qq >/dev/null'
+ sh -c 'DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-https ca-certificates curl >/dev/null'
+ sh -c 'install -m 0755 -d /etc/apt/keyrings'
+ sh -c 'curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | gpg --dearmor --yes -o /etc/apt/keyrings/docker.gpg'
+ sh -c 'chmod a+r /etc/apt/keyrings/docker.gpg'
+ sh -c 'echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu jammy stable" > /etc/apt/sources.list.d/docker.list'
+ sh -c 'apt-get update -qq >/dev/null'
+ sh -c 'DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-ce-rootless-extras docker-buildx-plugin >/
dev/null'
+ sh -c 'docker version'
Client: Docker Engine - Community
 Version:           24.0.5
 API version:       1.43
 Go version:        go1.20.6
 Git commit:        ced0996
 Built:             Fri Jul 21 20:35:18 2023
 OS/Arch:           linux/amd64
```

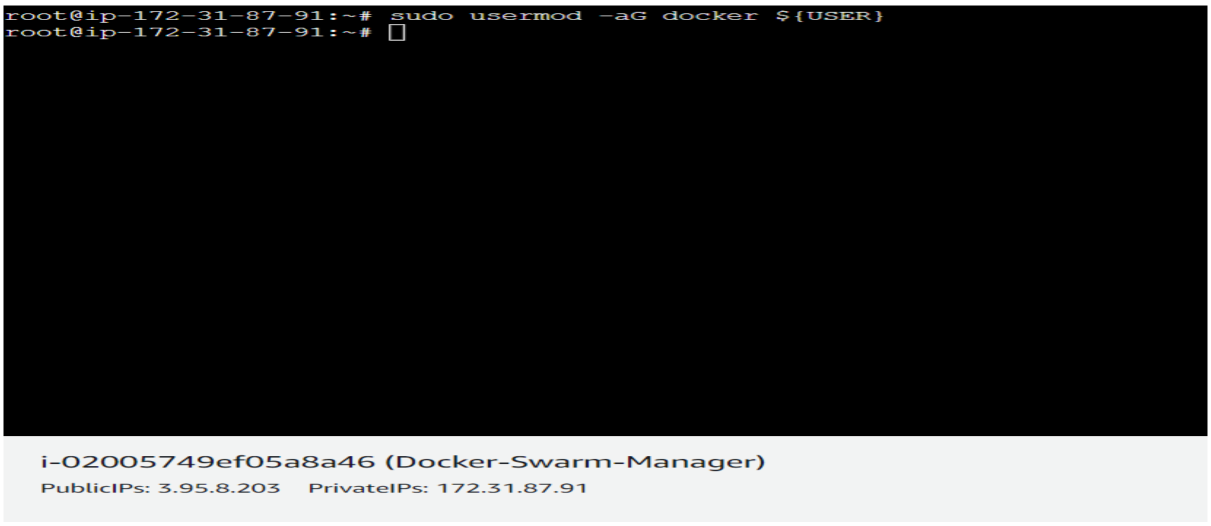
i-0c747f1b3df7bde10 (Worker_Two)
PublicIPs: 54.237.172.32    PrivateIPs: 172.31.93.16

Docker is also installed in worker two system server also we need to add user in all 3 systems by using command sudo usermod -aG docker ${USER}



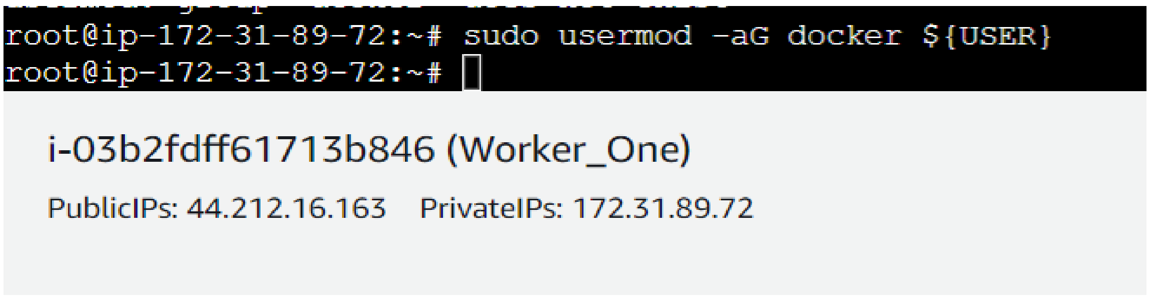
```
================================================================================

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

    dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.


To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
         to root access on the host. Refer to the 'Docker daemon attack surface'
         documentation for details: https://docs.docker.com/go/attack-surface/

================================================================================

root@ip-172-31-93-16:~# sudo usermod -aG docker ${USER}
root@ip-172-31-93-16:~# 
```

i-0c747f1b3df7bde10 (Worker_Two)
PublicIPs: 54.237.172.32    PrivateIPs: 172.31.93.16



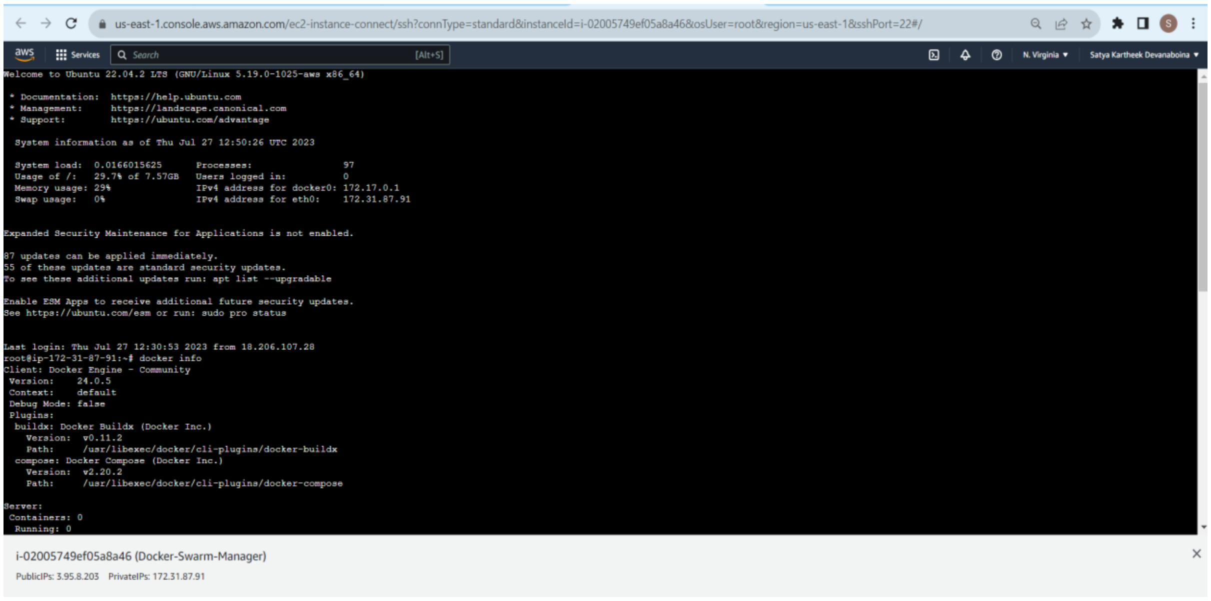
```
root@ip-172-31-87-91:~# sudo usermod -aG docker ${USER}
root@ip-172-31-87-91:~# 
```

i-02005749ef05a8a46 (Docker-Swarm-Manager)
PublicIPs: 3.95.8.203    PrivateIPs: 172.31.87.91


```
root@ip-172-31-89-72:~# sudo usermod -aG docker ${USER}
root@ip-172-31-89-72:~# 
```
i-03b2fdff61713b846 (Worker_One)

PublicIPs: 44.212.16.163    PrivateIPs: 172.31.89.72

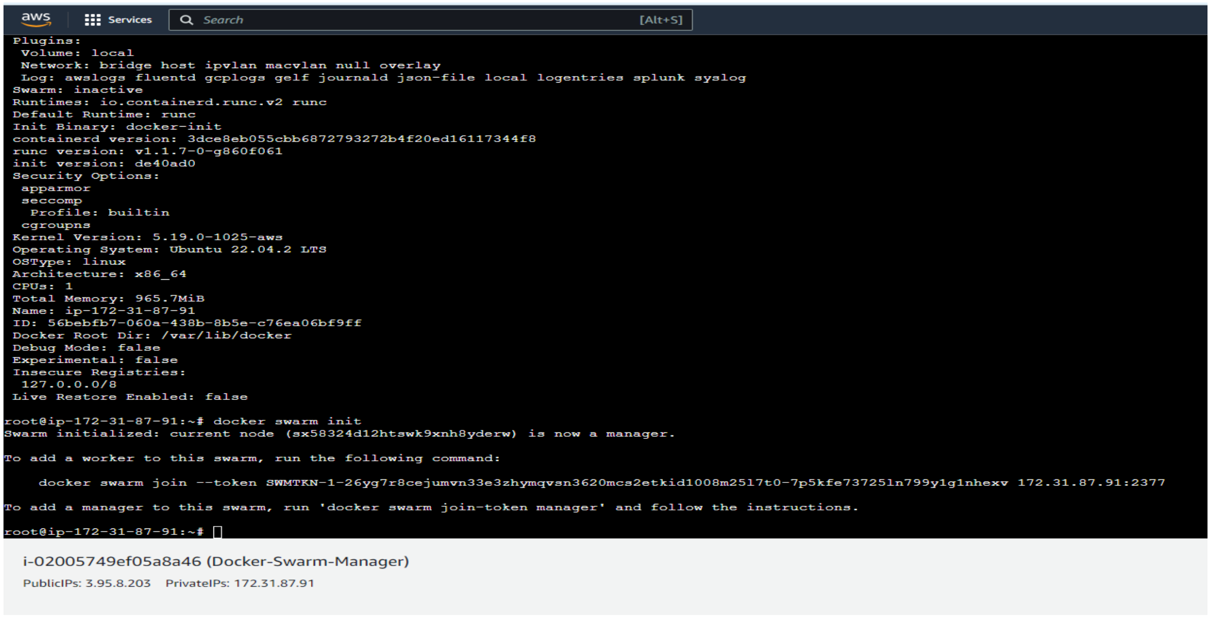
We can directly using command docker info after installation docker in ubuntu user so that version of docker will be displayed as we can see in this screenshot:


us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-02005749ef05a8a46&osUser=root&region=us-east-1&sshPort=22#/

```
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Jul 27 12:50:26 UTC 2023

  System load:  0.0166015625      Processes:                97
  Usage of /:   29.7% of 7.57GB   Users logged in:          0
  Memory usage: 29%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for eth0:    172.31.87.91


Expanded Security Maintenance for Applications is not enabled.

87 updates can be applied immediately.
55 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

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


Last login: Thu Jul 27 12:30:53 2023 from 18.206.107.28
root@ip-172-31-87-91:~# docker info
Client: Docker Engine - Community
 Version:    24.0.5
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.11.2
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.20.2
    Path:     /usr/libexec/docker/cli-plugins/docker-compose

Server:
 Containers: 0
  Running: 0
```
i-02005749ef05a8a46 (Docker-Swarm-Manager)

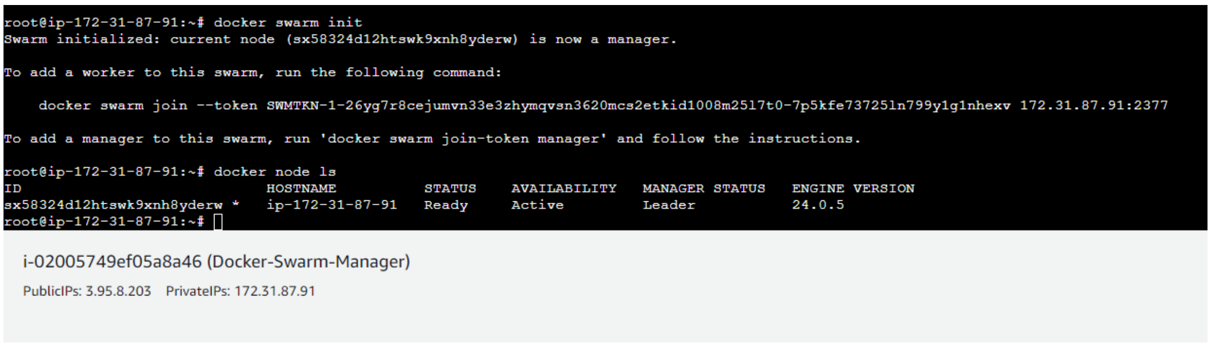
PublicIPs: 3.95.8.203    PrivateIPs: 172.31.87.91

We use docker swarm command in manager ubuntu server system the command  is docker swarm init. To add worker in the system use the command displayed in screenshot in worker 1 and worker 2 systems.


```
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 3dce8eb055cbb6872793272b4f20ed16117344f8
 runc version: v1.1.7-0-g860f061
 init version: de40ad0
 Security Options:
  apparmor
  seccomp
   Profile: builtin
  cgroupns
 Kernel Version: 5.19.0-1025-aws
 Operating System: Ubuntu 22.04.2 LTS
 OSType: linux
 Architecture: x86_64
 CPUs: 1
 Total Memory: 965.7MiB
 Name: ip-172-31-87-91
 ID: 56bebfb7-060a-438b-8b5e-c76ea06bf9ff
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

root@ip-172-31-87-91:~# docker swarm init
Swarm initialized: current node (sx58324d12htswk9xnh8yderw) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-26yg7r8cejumvn33e3zhymqvsn3620mcs2etkid1008m2517t0-7p5kfe737251n799y1g1nhexv 172.31.87.91:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@ip-172-31-87-91:~# 
```
i-02005749ef05a8a46 (Docker-Swarm-Manager)

PublicIPs: 3.95.8.203    PrivateIPs: 172.31.87.91

We use command docker node ls to display active user status of system as we can see in the screenshot below:

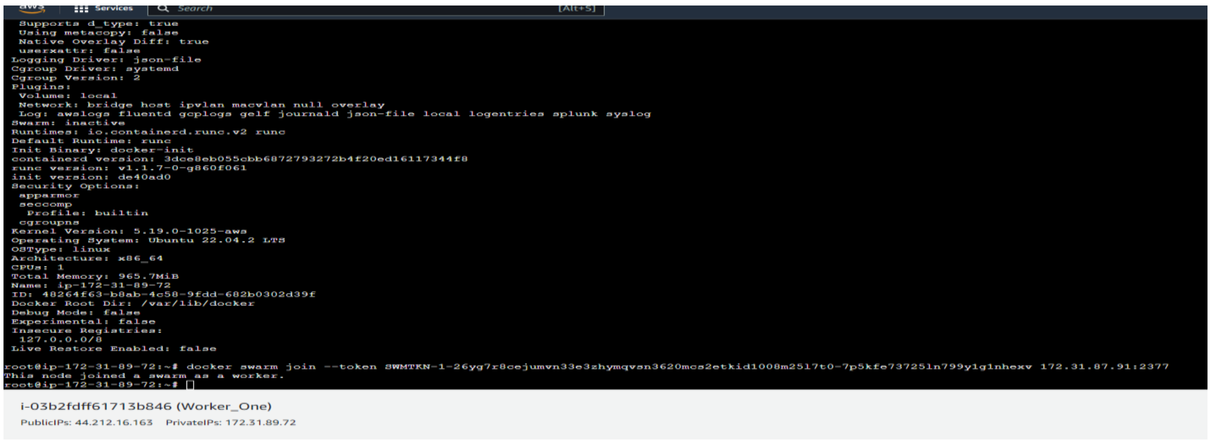


```
root@ip-172-31-87-91:~# docker swarm init
Swarm initialized: current node (sx58324d12htswk9xnh8yderw) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-26yg7r8cejumvn33e3zhymqvsn3620mcs2etkid1008m2517t0-7p5kfe737251n799y1g1nhexv 172.31.87.91:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@ip-172-31-87-91:~# docker node ls
ID                            HOSTNAME          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
sx58324d12htswk9xnh8yderw *   ip-172-31-87-91   Ready     Active         Leader           24.0.5
root@ip-172-31-87-91:~# 
```

i-02005749ef05a8a46 (Docker-Swarm-Manager)

PublicIPs: 3.95.8.203 PrivateIPs: 172.31.87.91

We used command provided in docker manager user system copied it and paste it in worker system to add user. As we can see it in below screenshot:

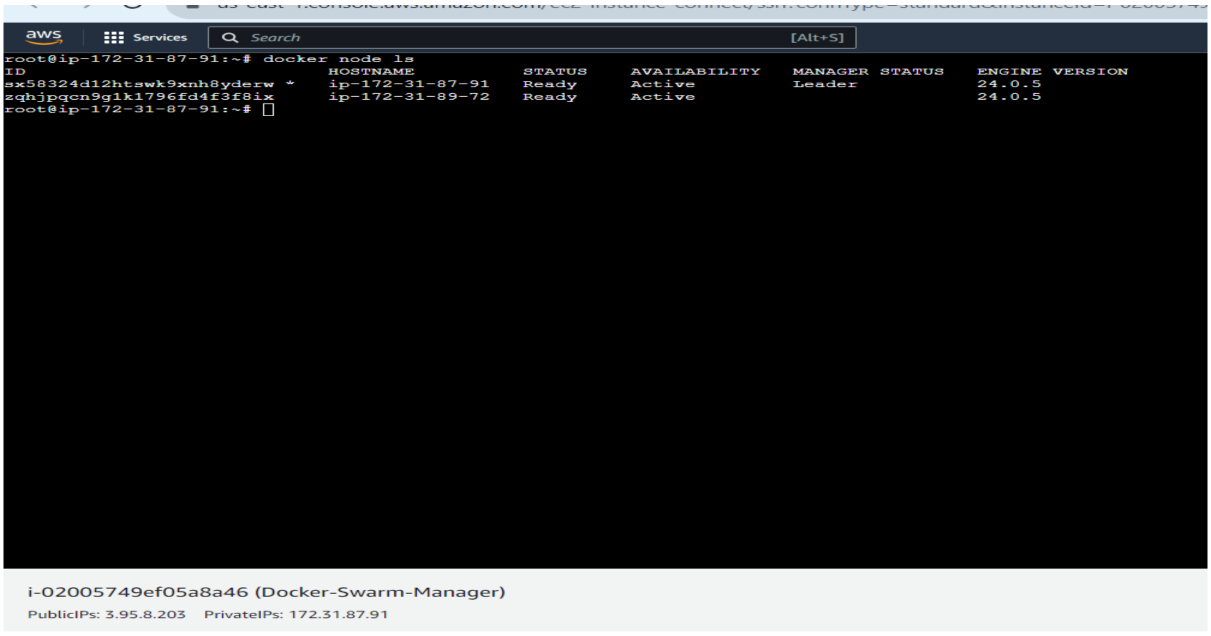


```
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 2
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 3dce8eb055cbb6872793272b4f20ed16117344f8
 runc version: v1.1.7-0-g860f061
 init version: de40ad0
 Security Options:
  apparmor
  seccomp
   Profile: builtin
  cgroupns
 Kernel Version: 5.19.0-1025-aws
 Operating System: Ubuntu 22.04.2 LTS
 OSType: linux
 Architecture: x86_64
 CPUs: 1
 Total Memory: 965.7MiB
 Name: ip-172-31-89-72
 ID: 48264f63-b8ab-4c58-9fdd-682b0302d39f
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false

root@ip-172-31-89-72:~# docker swarm join --token SWMTKN-1-26yg7r8cejumvn33e3zhymqvsn3620mcs2etkid1008m2517t0-7p5kfe737251n799y1g1nhexv 172.31.87.91:2377
This node joined a swarm as a worker.
root@ip-172-31-89-72:~# 
```

i-03b2fdff61713b846 (Worker_One)

PublicIPs: 44.212.16.163 PrivateIPs: 172.31.89.72

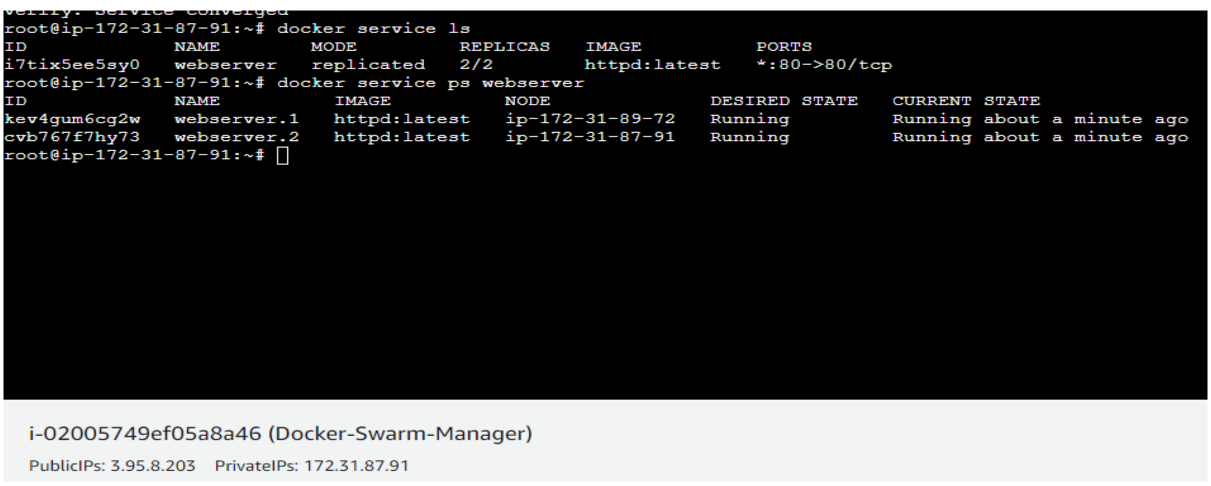
When we run docker node ls command in docker manager ubuntu server we can see worker is added.



```
root@ip-172-31-87-91:~# docker node ls
ID                            HOSTNAME          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
sx58324d12htswk9xnh8yderw *   ip-172-31-87-91   Ready     Active         Leader           24.0.5
zqhjpqcn9g1k1796fd4f3f8ix     ip-172-31-89-72   Ready     Active                          24.0.5
root@ip-172-31-87-91:~# 
```

i-02005749ef05a8a46 (Docker-Swarm-Manager)

PublicIPs: 3.95.8.203 PrivateIPs: 172.31.87.91

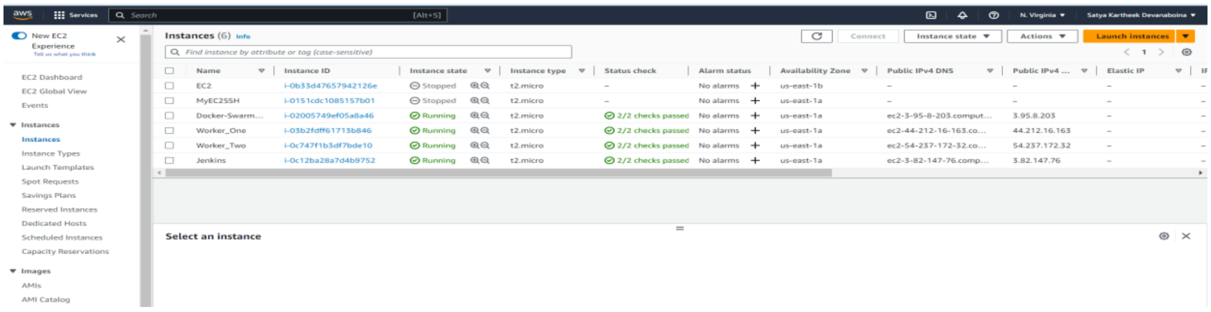


```
verify: Service converged
root@ip-172-31-87-91:~# docker service ls
ID             NAME        MODE         REPLICAS   IMAGE          PORTS
i7tix5ee5sy0   webserver   replicated   2/2        httpd:latest   *:80->80/tcp
root@ip-172-31-87-91:~# docker service ps webserver
ID             NAME          IMAGE          NODE             DESIRED STATE   CURRENT STATE
kev4gum6cg2w   webserver.1   httpd:latest   ip-172-31-89-72  Running         Running about a minute ago
cvb767f7hy73   webserver.2   httpd:latest   ip-172-31-87-91  Running         Running about a minute ago
root@ip-172-31-87-91:~# 
```

i-02005749ef05a8a46 (Docker-Swarm-Manager)

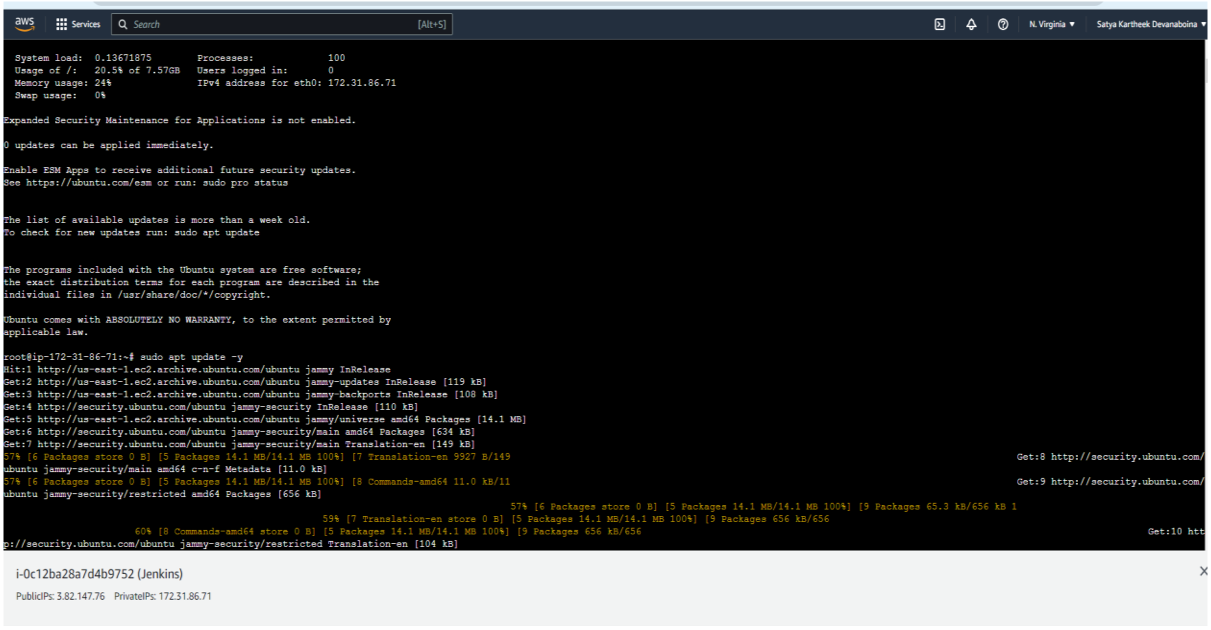
PublicIPs: 3.95.8.203    PrivateIPs: 172.31.87.91

We need to create Jenkins server instance as well as we can in screenshot below:



| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS | Public IPv4 ... | Elastic IP |
|---|---|---|---|---|---|---|---|---|---|
| EC2 | i-0b33d47657942126e | Stopped | t2.micro | – | No alarms | us-east-1b | – | – | – |
| MyEC2SSH | i-0151cdc1085157b01 | Stopped | t2.micro | – | No alarms | us-east-1a | – | – | – |
| Docker-Swarm... | i-02005749ef05a8a46 | Running | t2.micro | 2/2 checks passed | No alarms | us-east-1a | ec2-3-95-8-203.comput... | 3.95.8.203 | – |
| Worker_One | i-03b2fdff61713b846 | Running | t2.micro | 2/2 checks passed | No alarms | us-east-1a | ec2-44-212-16-163.co... | 44.212.16.163 | – |
| Worker_Two | i-0c747f1b3df7bde10 | Running | t2.micro | 2/2 checks passed | No alarms | us-east-1a | ec2-54-237-172-32.co... | 54.237.172.32 | – |
| Jenkins | i-0c12ba28a7d4b9752 | Running | t2.micro | 2/2 checks passed | No alarms | us-east-1a | ec2-3-82-147-76.comp... | 3.82.147.76 | – |

Use command sudo apt update -y to perform updates.



```
  System load:  0.13671875       Processes:             100
  Usage of /:   20.5% of 7.57GB  Users logged in:       0
  Memory usage: 24%              IPv4 address for eth0: 172.31.86.71
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

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.

root@ip-172-31-86-71:~# sudo apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [634 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [149 kB]
57% [6 Packages store 0 B] [5 Packages 14.1 MB/14.1 MB 100%] [7 Translation-en 9927 B/149                                   Get:8 http://security.ubuntu.com/
ubuntu jammy-security/main amd64 c-n-f Metadata [11.0 kB]
57% [6 Packages store 0 B] [5 Packages 14.1 MB/14.1 MB 100%] [8 Commands-amd64 11.0 kB/11                                   Get:9 http://security.ubuntu.com/
ubuntu jammy-security/restricted amd64 Packages [656 kB]
                                                                                    57% [6 Packages store 0 B] [5 Packages 14.1 MB/14.1 MB 100%] [9 Packages 65.3 kB/656 kB 1
                                                    59% [7 Translation-en store 0 B] [5 Packages 14.1 MB/14.1 MB 100%] [9 Packages 656 kB/656
                      60% [8 Commands-amd64 store 0 B] [5 Packages 14.1 MB/14.1 MB 100%] [9 Packages 656 kB/656                        Get:10 htt
p://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [104 kB]
```

i-0c12ba28a7d4b9752 (Jenkins)

PublicIPs: 3.82.147.76    PrivateIPs: 172.31.86.71

We need to install java Jenkins in Jenkins server sysem.

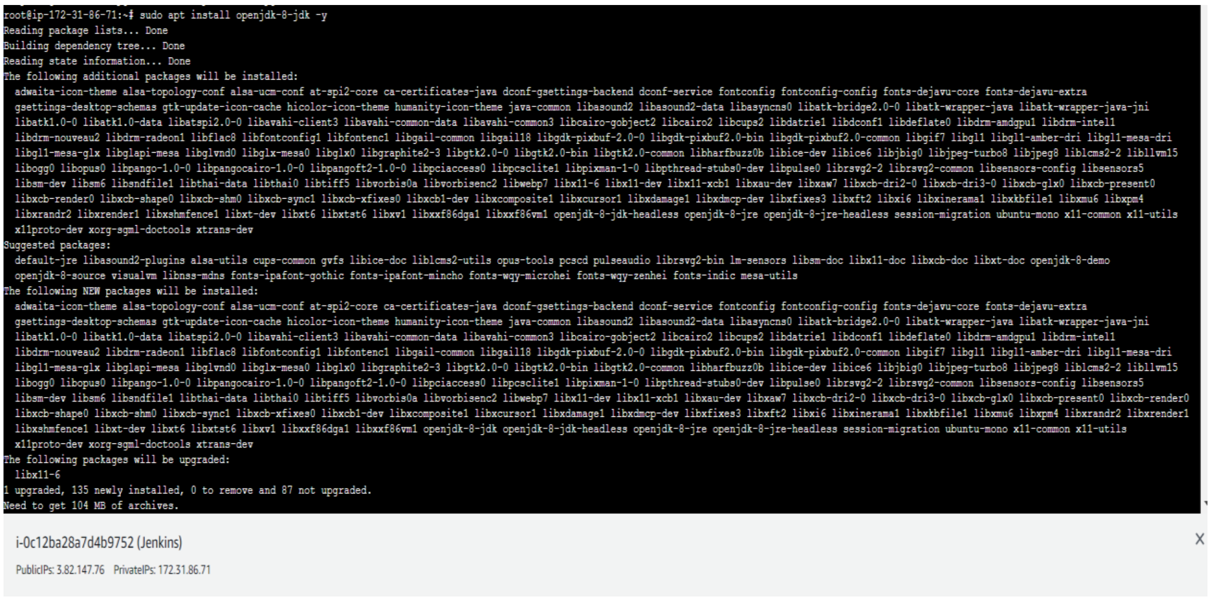
. Depending on which Java version you want to install, Java 8 or 11, run one of the following commands:

To install OpenJDK 8, run:

sudo apt install openjdk-8-jdk -y

To install OpenJDK 11, run:

sudo apt install openjdk-11-jdk -y



```
root@ip-172-31-86-71:~# sudo apt install openjdk-8-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  adwaita-icon-theme alsa-topology-conf alsa-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig fontconfig-config fonts-dejavu-core fonts-dejavu-extra
  gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humanity-icon-theme java-common libasound2 libasound2-data libasyncns0 libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni
  libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcairo-gobject2 libcairo2 libcups2 libdatrie1 libdconf1 libdeflate0 libdrm-amdgpu1 libdrm-intel1
  libdrm-nouveau2 libdrm-radeon1 libflac8 libfontconfig1 libfontenc1 libgail-common libgail18 libgdk-pixbuf-2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgif7 libgl1 libgl1-amber-dri libgl1-mesa-dri
  libgl1-mesa-glx libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libgtk2.0-0 libgtk2.0-bin libgtk2.0-common libharfbuzz0b libice-dev libice6 libjbig0 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm15
  libogg0 libopus0 libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpciaccess0 libpcsclite1 libpixman-1-0 libpthread-stubs0-dev libpulse0 librsvg2-2 librsvg2-common libsensors-config libsensors5
  libsm-dev libsm6 libsndfile1 libthai-data libthai0 libtiff5 libvorbis0a libvorbisenc2 libwebp7 libx11-6 libx11-dev libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0
  libxcb-render0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcomposite1 libxcursor1 libxdamage1 libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4
  libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless session-migration ubuntu-mono x11-common x11-utils
  x11proto-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
  default-jre libasound2-plugins alsa-utils cups-common gvfs libice-doc liblcms2-utils opus-tools pcscd pulseaudio librsvg2-bin lm-sensors libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-8-demo
  openjdk-8-source visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei fonts-indic mesa-utils
The following NEW packages will be installed:
  adwaita-icon-theme alsa-topology-conf alsa-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig fontconfig-config fonts-dejavu-core fonts-dejavu-extra
  gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humanity-icon-theme java-common libasound2 libasound2-data libasyncns0 libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni
  libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcairo-gobject2 libcairo2 libcups2 libdatrie1 libdconf1 libdeflate0 libdrm-amdgpu1 libdrm-intel1
  libdrm-nouveau2 libdrm-radeon1 libflac8 libfontconfig1 libfontenc1 libgail-common libgail18 libgdk-pixbuf-2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgif7 libgl1 libgl1-amber-dri libgl1-mesa-dri
  libgl1-mesa-glx libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libgtk2.0-0 libgtk2.0-bin libgtk2.0-common libharfbuzz0b libice-dev libice6 libjbig0 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm15
  libogg0 libopus0 libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpciaccess0 libpcsclite1 libpixman-1-0 libpthread-stubs0-dev libpulse0 librsvg2-2 librsvg2-common libsensors-config libsensors5
  libsm-dev libsm6 libsndfile1 libthai-data libthai0 libtiff5 libvorbis0a libvorbisenc2 libwebp7 libx11-dev libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-render0
  libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcomposite1 libxcursor1 libxdamage1 libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1
  libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 openjdk-8-jdk openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless session-migration ubuntu-mono x11-common x11-utils
  x11proto-dev xorg-sgml-doctools xtrans-dev
The following packages will be upgraded:
  libx11-6
1 upgraded, 135 newly installed, 0 to remove and 87 not upgraded.
Need to get 104 MB of archives.
```

i-0c12ba28a7d4b9752 (Jenkins)

PublicIPs: 3.82.147.76 PrivateIPs: 172.31.86.71

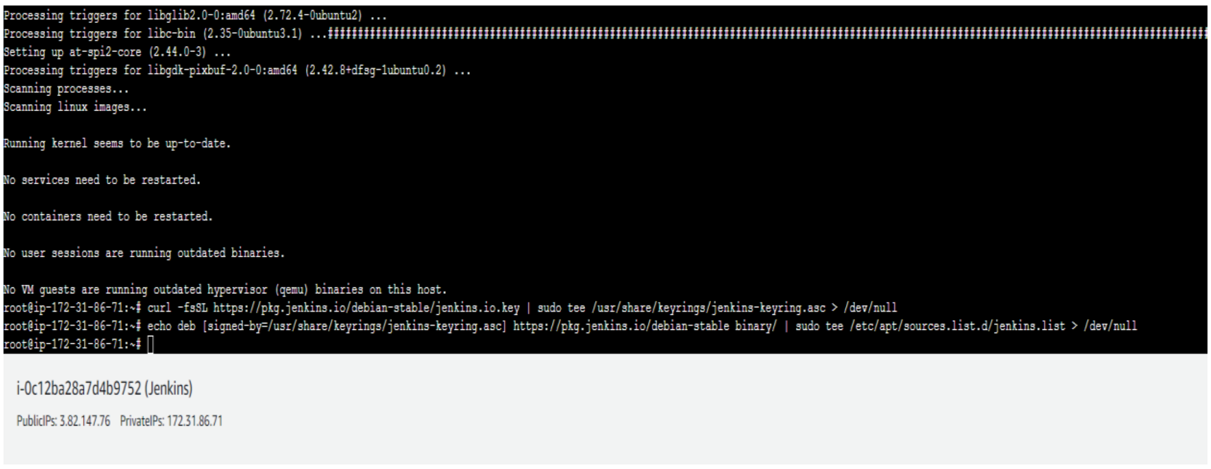
Follow the steps below to add the Jenkins repository to your Ubuntu system.

1. Start by importing the GPG key. The GPG key verifies package integrity but there is no output. Run:

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

2. Add the Jenkins software repository to the source list and provide the authentication 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



```
Processing triggers for libglib2.0-0:amd64 (2.72.4-0ubuntu2) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...################################################################################################################
Setting up at-spi2-core (2.44.0-3) ...
Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.8+dfsg-1ubuntu0.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-86-71:~# curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
root@ip-172-31-86-71:~# 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
root@ip-172-31-86-71:~# 
```

i-0c12ba28a7d4b9752 (Jenkins)

PublicIPs: 3.82.147.76 PrivateIPs: 172.31.86.71

After setting up the prerequisites, follow the steps below to install Jenkins on Ubuntu:

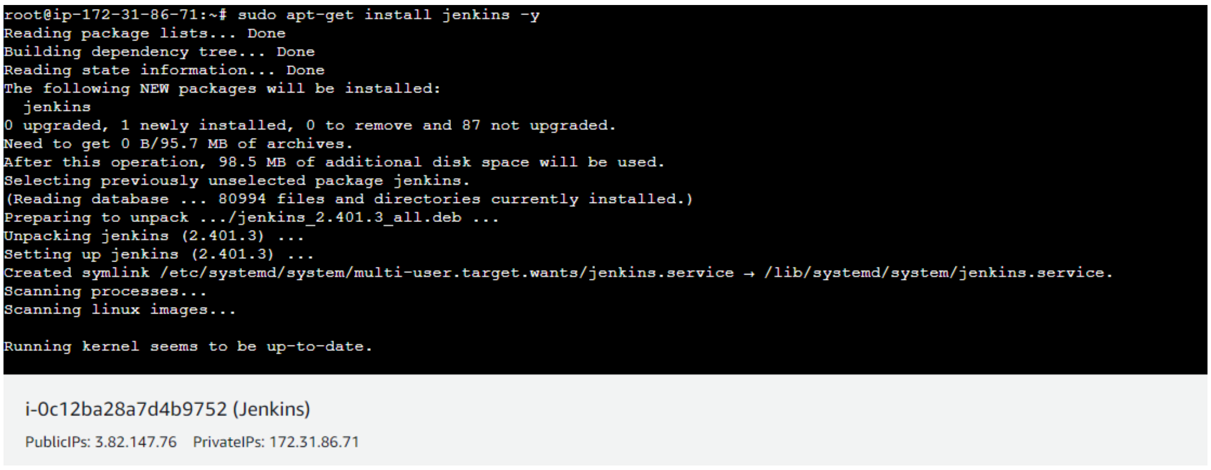
1. Update the system repository one more time. Updating refreshes the cache and makes the system aware of the new Jenkins repository.

sudo apt update

2. Install Jenkins by running:

sudo apt install jenkins -y

Wait for the download and installation to complete.



```
root@ip-172-31-86-71:~# sudo apt-get install jenkins -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  jenkins
0 upgraded, 1 newly installed, 0 to remove and 87 not upgraded.
Need to get 0 B/95.7 MB of archives.
After this operation, 98.5 MB of additional disk space will be used.
Selecting previously unselected package jenkins.
(Reading database ... 80994 files and directories currently installed.)
Preparing to unpack .../jenkins_2.401.3_all.deb ...
Unpacking jenkins (2.401.3) ...
Setting up jenkins (2.401.3) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
```

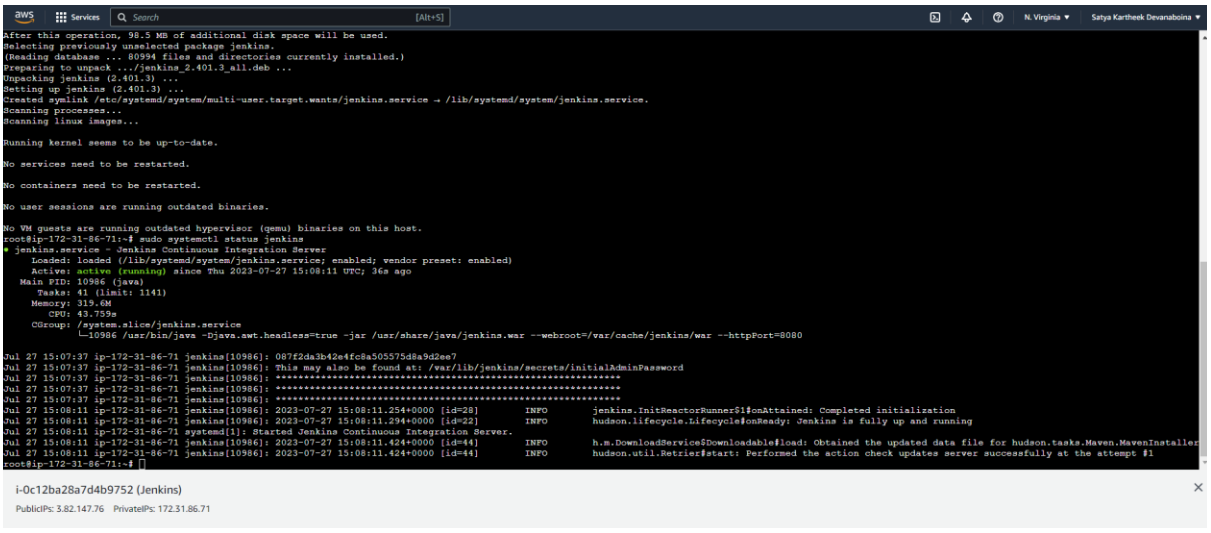
i-0c12ba28a7d4b9752 (Jenkins)

PublicIPs: 3.82.147.76 PrivateIPs: 172.31.86.71

To check if Jenkins is installed and running, run the following command:

sudo systemctl status jenkins

A bright green entry labelled active (running) should appear in the output, indicating that the service is running.



```
After this operation, 98.5 MB of additional disk space will be used.
Selecting previously unselected package jenkins.
(Reading database ... 80994 files and directories currently installed.)
Preparing to unpack .../jenkins_2.401.3_all.deb ...
Unpacking jenkins (2.401.3) ...
Setting up jenkins (2.401.3) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-86-71:~# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-07-27 15:08:11 UTC; 36s ago
   Main PID: 10986 (java)
      Tasks: 41 (limit: 1141)
     Memory: 319.6M
        CPU: 43.759s
     CGroup: /system.slice/jenkins.service
             └─10986 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jul 27 15:07:37 ip-172-31-86-71 jenkins[10986]: 087f2da3b42e4fc8a505575d8a9d2ee7
Jul 27 15:07:37 ip-172-31-86-71 jenkins[10986]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jul 27 15:07:37 ip-172-31-86-71 jenkins[10986]: *************************************************************
Jul 27 15:07:37 ip-172-31-86-71 jenkins[10986]: *************************************************************
Jul 27 15:07:37 ip-172-31-86-71 jenkins[10986]: *************************************************************
Jul 27 15:08:11 ip-172-31-86-71 jenkins[10986]: 2023-07-27 15:08:11.254+0000 [id=28]        INFO        jenkins.InitReactorRunner$1#onAttained: Completed initialization
Jul 27 15:08:11 ip-172-31-86-71 jenkins[10986]: 2023-07-27 15:08:11.294+0000 [id=22]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Jul 27 15:08:11 ip-172-31-86-71 systemd[1]: Started Jenkins Continuous Integration Server.
Jul 27 15:08:11 ip-172-31-86-71 jenkins[10986]: 2023-07-27 15:08:11.424+0000 [id=44]        INFO        h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Jul 27 15:08:11 ip-172-31-86-71 jenkins[10986]: 2023-07-27 15:08:11.424+0000 [id=44]        INFO        hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
root@ip-172-31-86-71:~# 
```

i-0c12ba28a7d4b9752 (Jenkins)

PublicIPs: 3.82.147.76 PrivateIPs: 172.31.86.71

Allow Jenkins to communicate by setting up the default UFW firewall.

1. Open port 8080 by running the following commands:

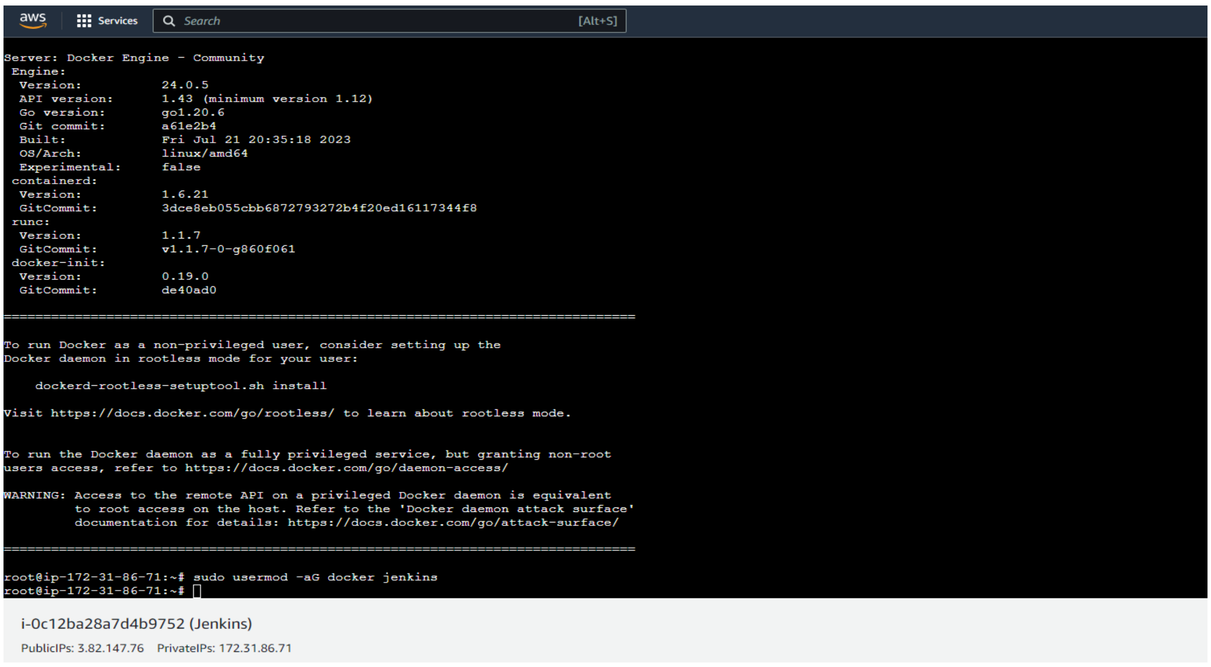
sudo ufw allow 8080

sudo ufw status

If you're using a different firewall application, follow its specific instructions to allow traffic on port 8080.

2. If you haven't configured the UFW firewall yet, it displays as inactive. Enable UFW by running:

```
sudo ufw enable
```



```
Server: Docker Engine - Community
 Engine:
  Version:          24.0.5
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.6
  Git commit:       a61e2b4
  Built:            Fri Jul 21 20:35:18 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.21
  GitCommit:        3dce8eb055cbb6872793272b4f20ed16117344f8
 runc:
  Version:          1.1.7
  GitCommit:        v1.1.7-0-g860f061
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0

================================================================================

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

    dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.


To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
         to root access on the host. Refer to the 'Docker daemon attack surface'
         documentation for details: https://docs.docker.com/go/attack-surface/

================================================================================

root@ip-172-31-86-71:~# sudo usermod -aG docker jenkins
root@ip-172-31-86-71:~# 
```

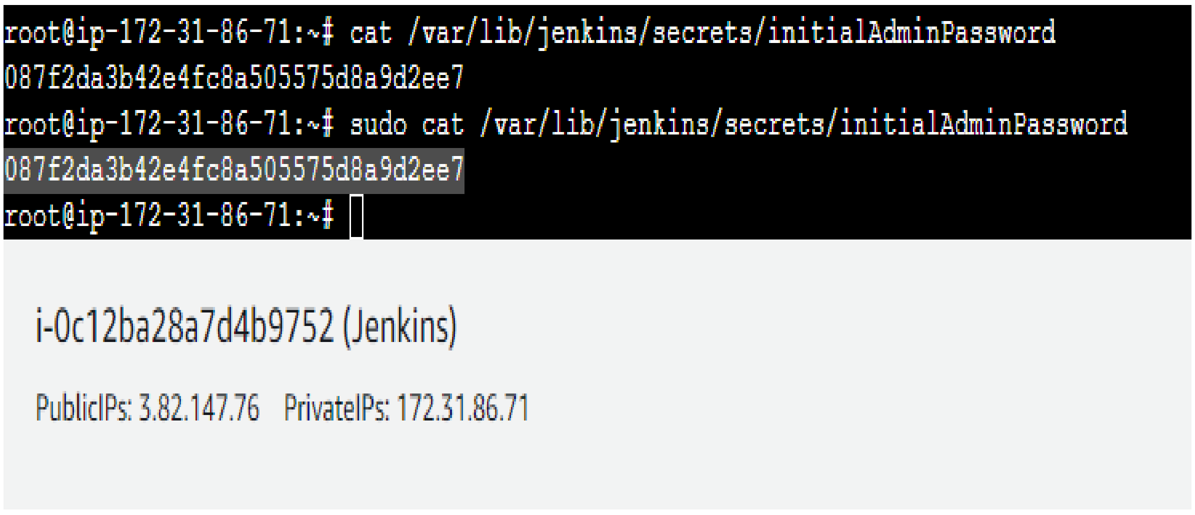
i-0c12ba28a7d4b9752 (Jenkins)

PublicIPs: 3.82.147.76  PrivateIPs: 172.31.86.71

We use public ip of Jenkins server to access it we need admin password. A page opens prompting you to Unlock Jenkins. Obtain the required administrator password in the next step.

Obtain the default Jenkins unlock password by opening the terminal and running the following command:

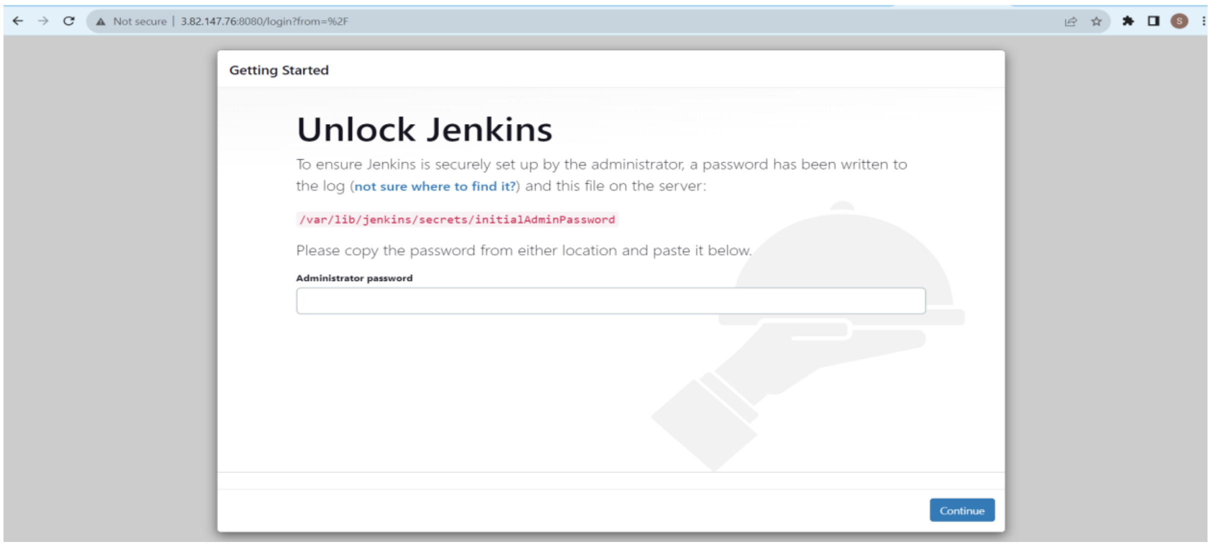
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



```
root@ip-172-31-86-71:~# cat /var/lib/jenkins/secrets/initialAdminPassword
087f2da3b42e4fc8a505575d8a9d2ee7
root@ip-172-31-86-71:~# sudo cat /var/lib/jenkins/secrets/initialAdminPassword
087f2da3b42e4fc8a505575d8a9d2ee7
root@ip-172-31-86-71:~# 
```

i-0c12ba28a7d4b9752 (Jenkins)

PublicIPs: 3.82.147.76  PrivateIPs: 172.31.86.71



## Unlock Jenkins

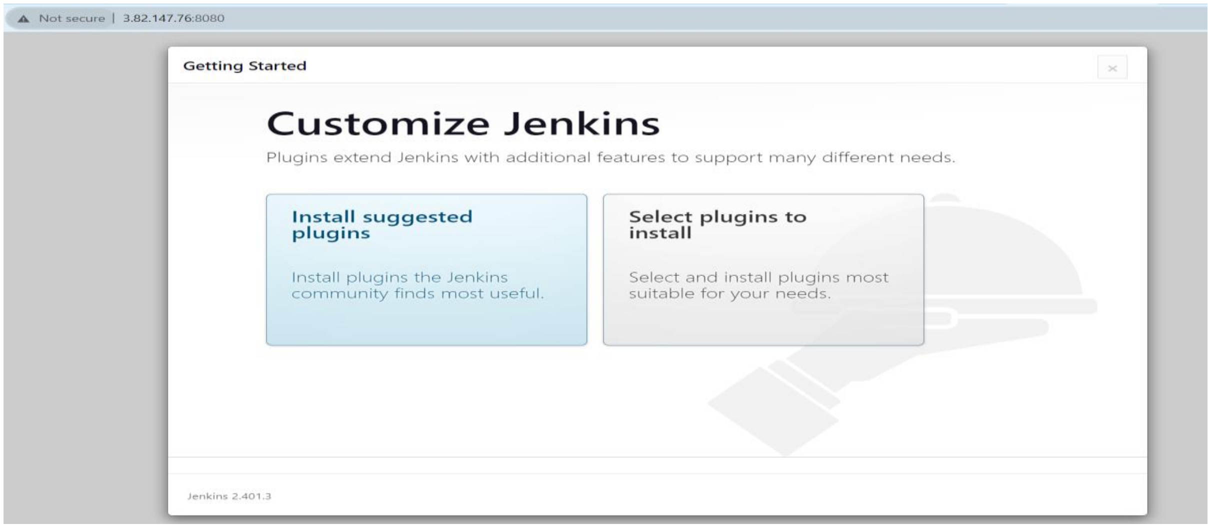
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

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

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

**Administrator password**

We need to install suggested plugins it will take some time to install after that we need to create account by giving user name password confirm password email id then save and continue.



## Customize Jenkins

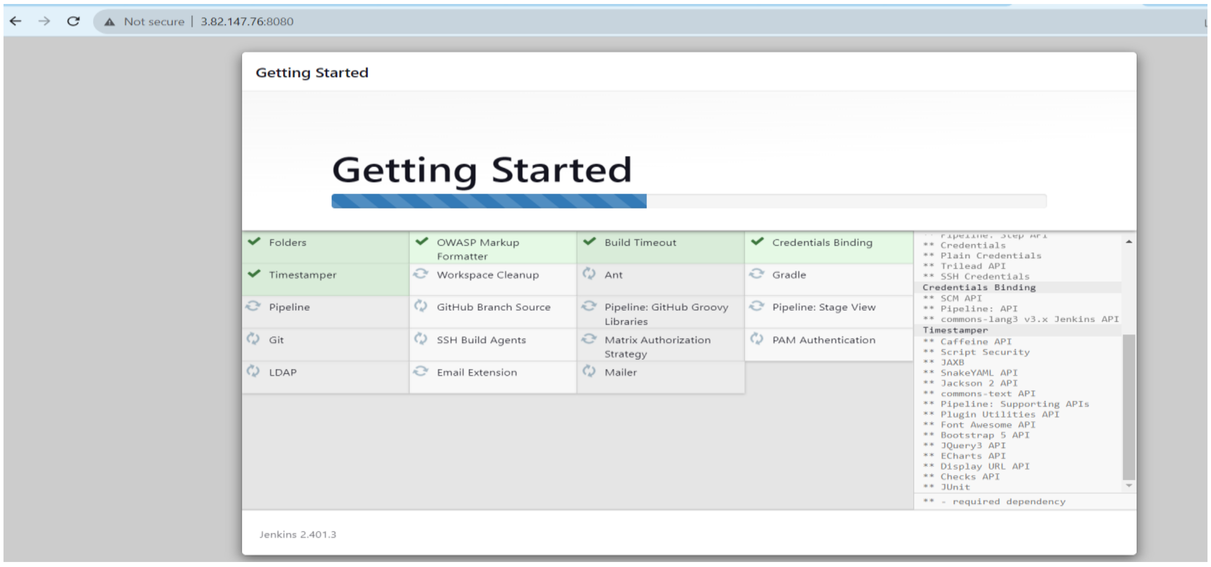
Plugins extend Jenkins with additional features to support many different needs.

**Install suggested plugins**

Install plugins the Jenkins community finds most useful.

**Select plugins to install**

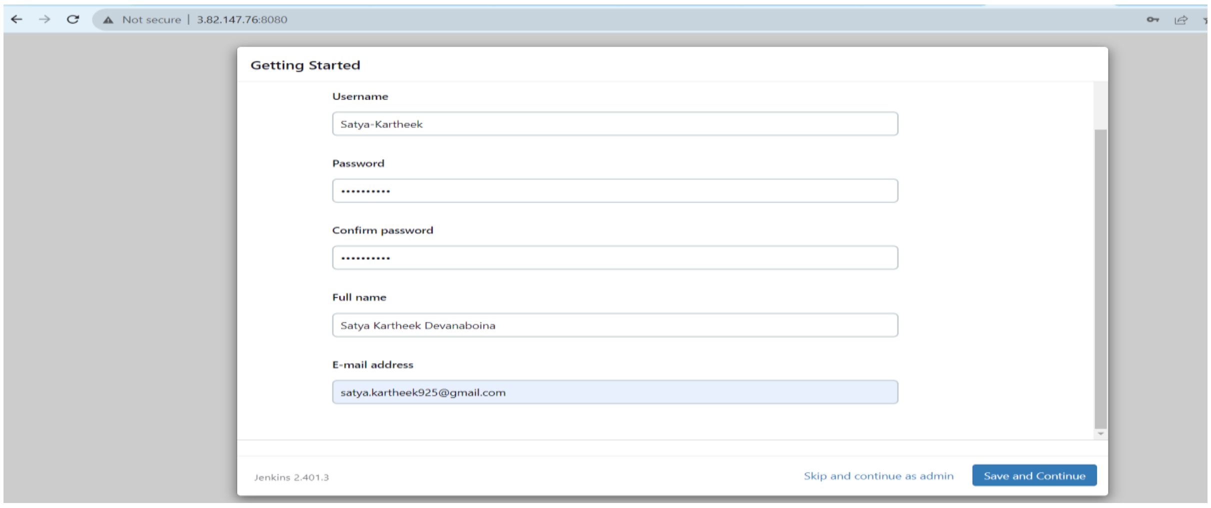
Select and install plugins most suitable for your needs.



## Getting Started

| | | | |
|---|---|---|---|
| ✓ Folders | ✓ OWASP Markup Formatter | ✓ Build Timeout | ✓ Credentials Binding |
| ✓ Timestamper | Workspace Cleanup | Ant | Gradle |
| Pipeline | GitHub Branch Source | Pipeline: GitHub Groovy Libraries | Pipeline: Stage View |
| Git | SSH Build Agents | Matrix Authorization Strategy | PAM Authentication |
| LDAP | Email Extension | Mailer | |

```
** Pipeline: Step API
** Credentials
** Plain Credentials
** Trilead API
** SSH Credentials
Credentials Binding
** SCM API
** Pipeline: API
** commons-lang3 v3.x Jenkins API
Timestamper
** Caffeine API
** Script Security
** JAXB
** SnakeYAML API
** Jackson 2 API
** commons-text API
** Pipeline: Supporting APIs
** Plugin Utilities API
** Font Awesome API
** Bootstrap 5 API
** JQuery3 API
** ECharts API
** Display URL API
** Checks API
** JUnit
** - required dependency
```


Not secure | 3.82.147.76:8080

Getting Started

Username: Satya-Kartheek

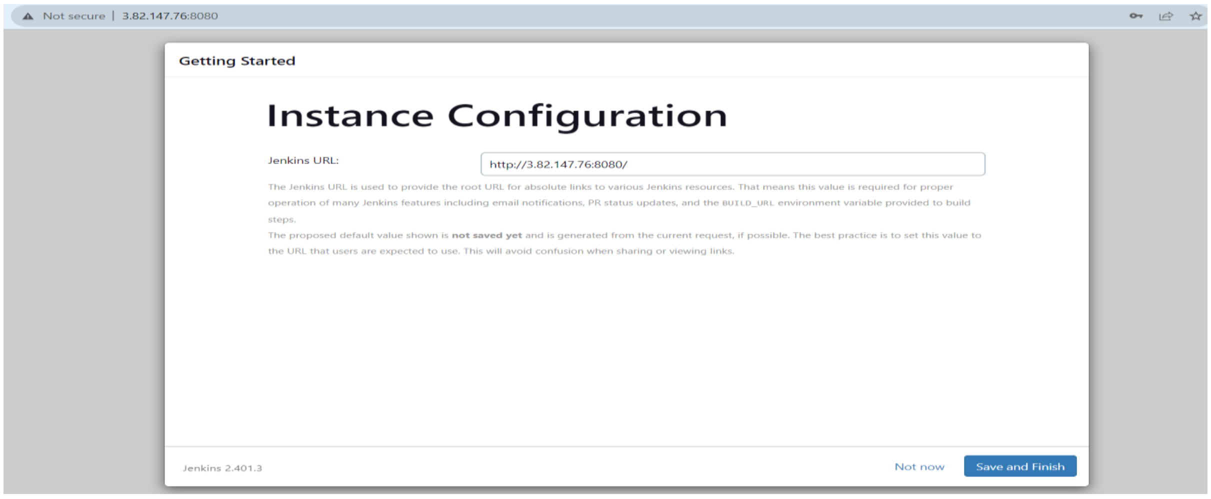
Password: ••••••••••

Confirm password: ••••••••••

Full name: Satya Kartheek Devanaboina

E-mail address: satya.kartheek925@gmail.com

Jenkins 2.401.3


Not secure | 3.82.147.76:8080

Getting Started

Instance Configuration

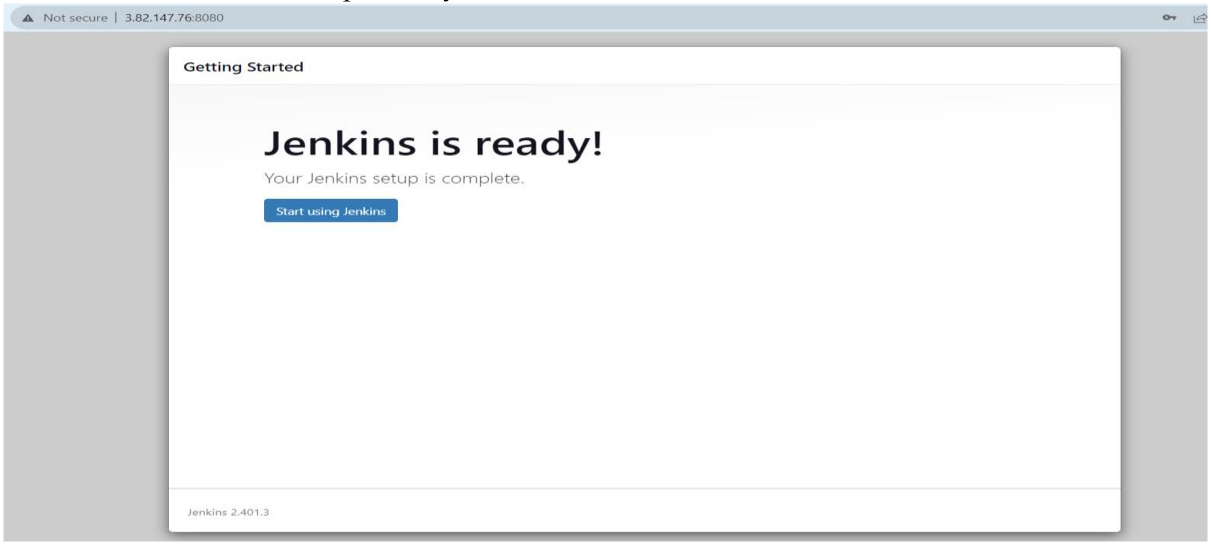
Jenkins URL: http://3.82.147.76:8080/

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 2.401.3

We can see that Jenkins set up is ready to use.


Not secure | 3.82.147.76:8080

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

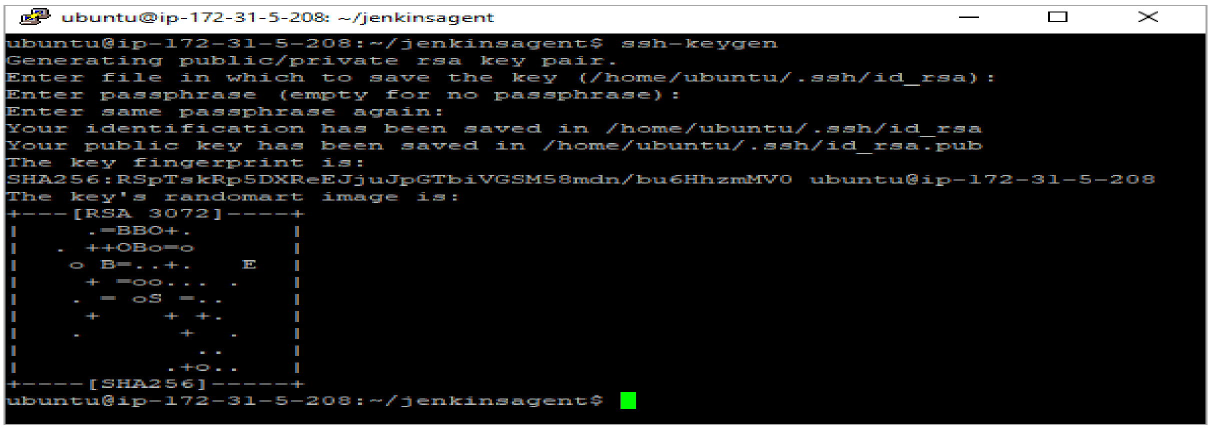
Jenkins 2.401.3

If Jenkins master is on Ubuntu/RHEL and slave nodes are also on Ubuntu/RHEL then you will need to copy the master's public key (~/.ssh/id_rsa.pub) to the slave nodes .ssh/authorized_keys file.

Step #1: Log in to the Jenkins MASTER machine and create a ssh key pair.

Use the following command to create a key pair:

# ssh-keygen
# cat .ssh/id_rsa.pub



```
ubuntu@ip-172-31-5-208: ~/jenkinsagent
ubuntu@ip-172-31-5-208:~/jenkinsagent$ 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:RSpTskRp5DXReEJjuJpGTbiVGSM58mdn/bu6HhzmMV0 ubuntu@ip-172-31-5-208
The key's randomart image is:
+---[RSA 3072]----+
|     .=BBO+.     |
|   . ++OBo=o     |
|    o B=..+.   E |
|     + =oo... .  |
|    . = oS =..   |
|     +    + +.   |
|    .      +  .  |
|             ..  |
|          .+o..  |
+----[SHA256]-----+
ubuntu@ip-172-31-5-208:~/jenkinsagent$
```

Copy the content and log in to the slave node. Add the copied content to authorized_keys.

# vi .ssh/authorized_keys

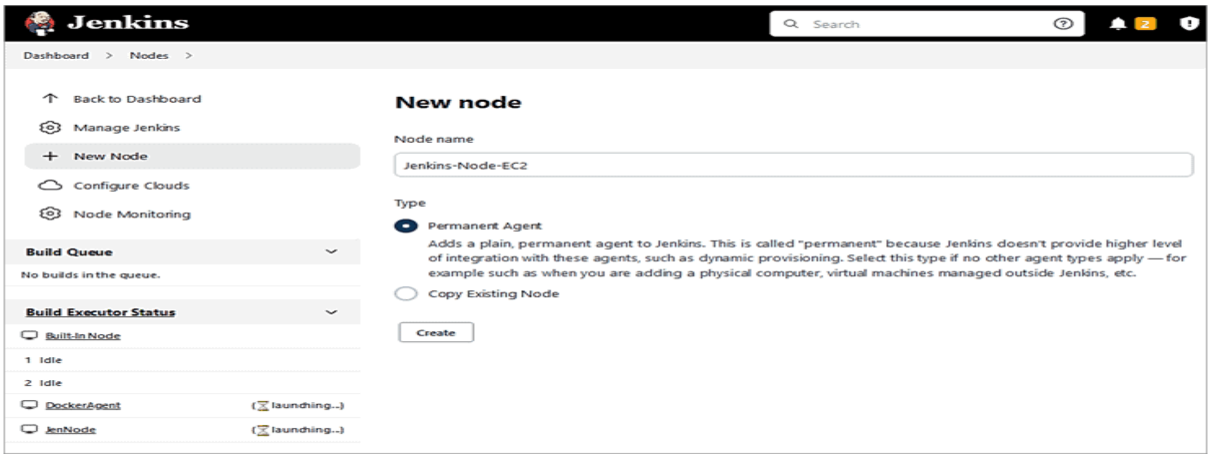
From the master machine ssh to the slave node using the below command. It will ask you to accept the ssh fingerprint, type yes and enter. You should be able to ssh into the slave node.

ssh userid@EC2VM-IP-Address

Step #2: Create a new Slave Node

Go to Manage Jenkins -> Manage nodes and clouds

Click on + New Node.



No builds in the queue.

Built-In Node
1 Idle
2 Idle
DockerAgent (launching...)
JenNode (launching...)

**New node**

Node name: Jenkins-Node-EC2

Type

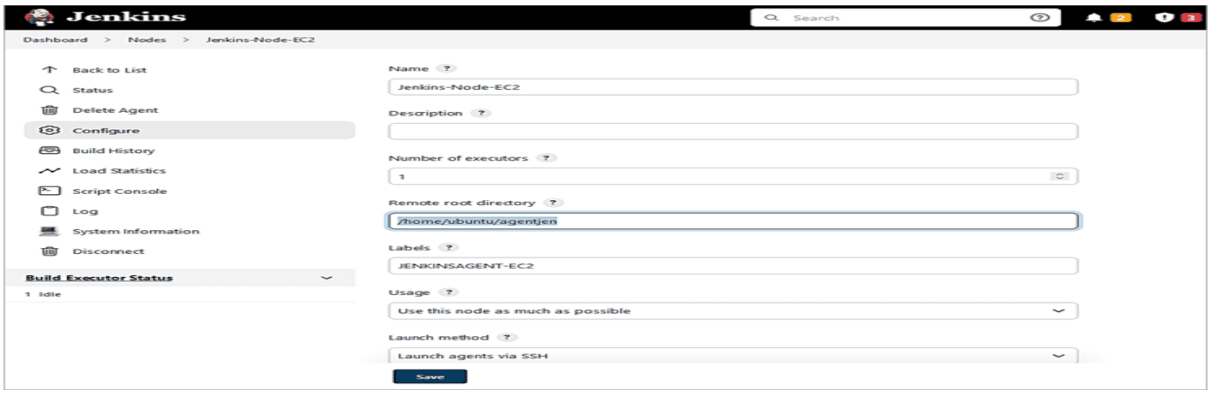
[✓ Permanent Agent]
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Copy Existing Node

Create

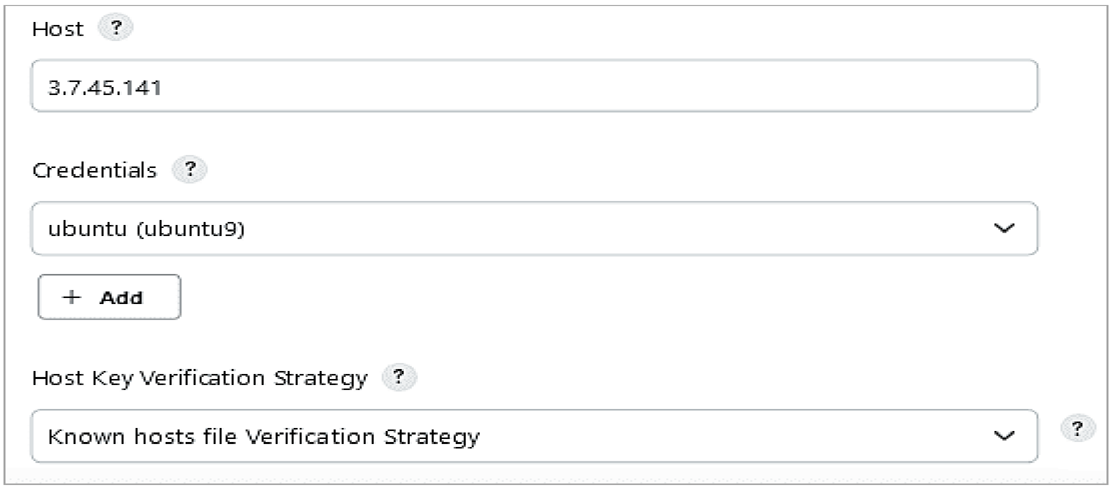
Click on Create.

Provide a Name and enter the Remote root directory of the slave machine.

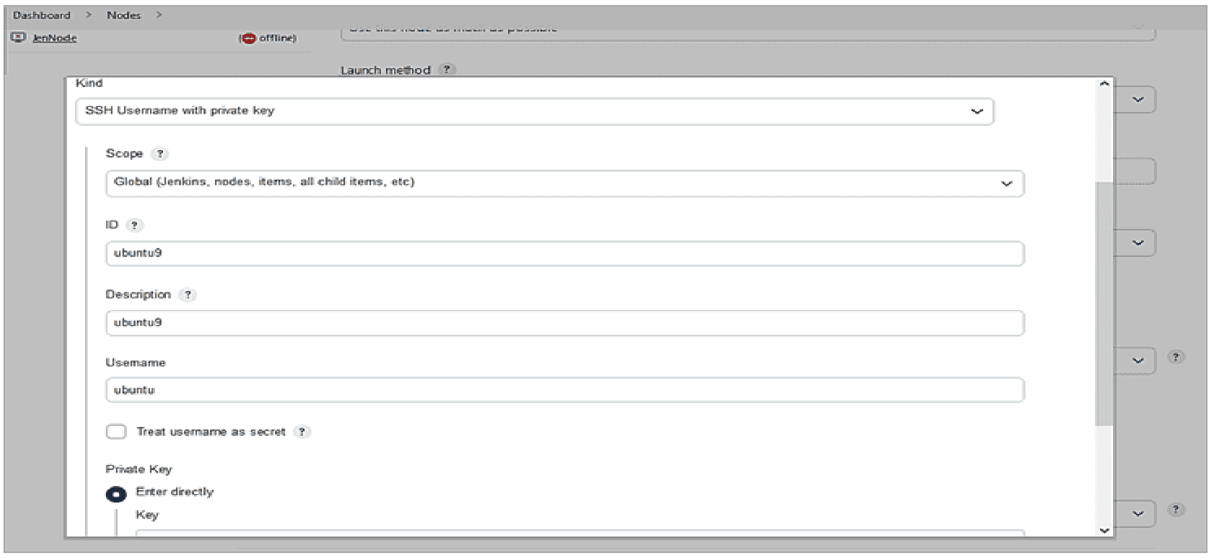


Name: Jenkins-Node-EC2
Description:
Number of executors: 1
Remote root directory: /home/ubuntu/agentjen
Labels: JENKINSAGENT-EC2
Usage: Use this node as much as possible
Launch method: Launch agents via SSH

Add the Host IP of the Slave machine (EC2 VM) and Add credentials.



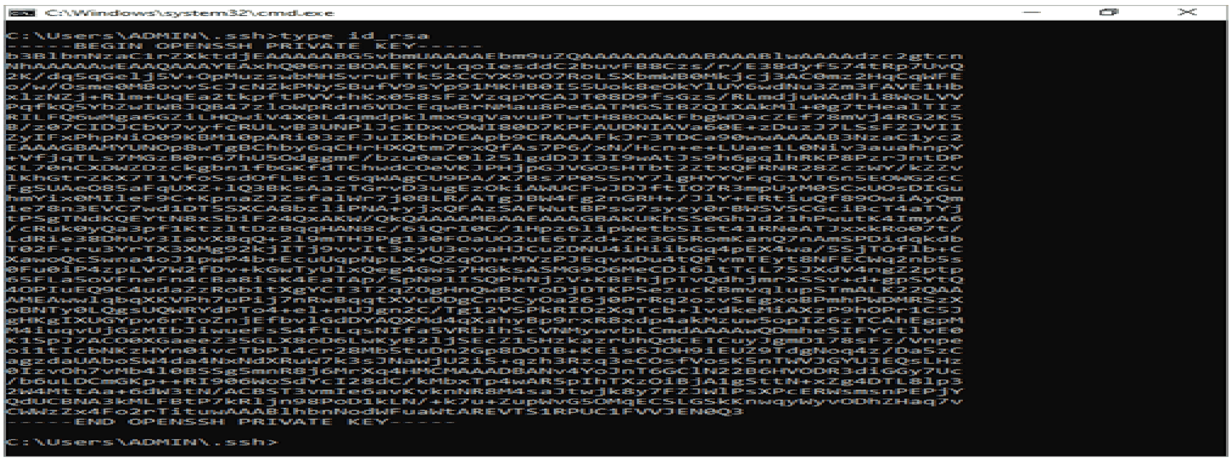
Host: 3.7.45.141
Credentials: ubuntu (ubuntu9)
Host Key Verification Strategy: Known hosts file Verification Strategy



Kind: SSH Username with private key
Scope: Global (Jenkins, nodes, items, all child items, etc)
ID: ubuntu9
Description: ubuntu9
Username: ubuntu
Treat username as secret
Private Key
Enter directly
Key

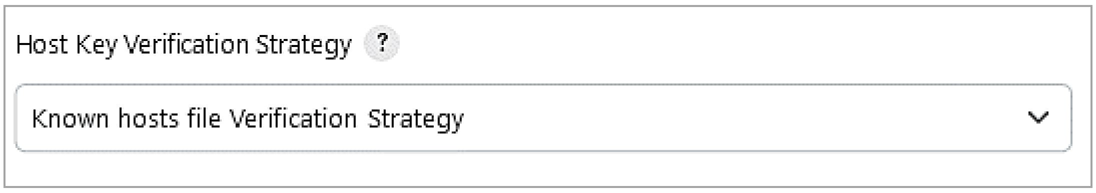
Username is ubuntu which is the login of the AWS Ec2 VM. In the private key field add the Jenkins master machine private key. You can find the private key in ~/.ssh/id_rsa file.

For e.g., if Jenkins Master is in Windows, here is the file in C:\users\<UserName>\.ssh\id_rsa



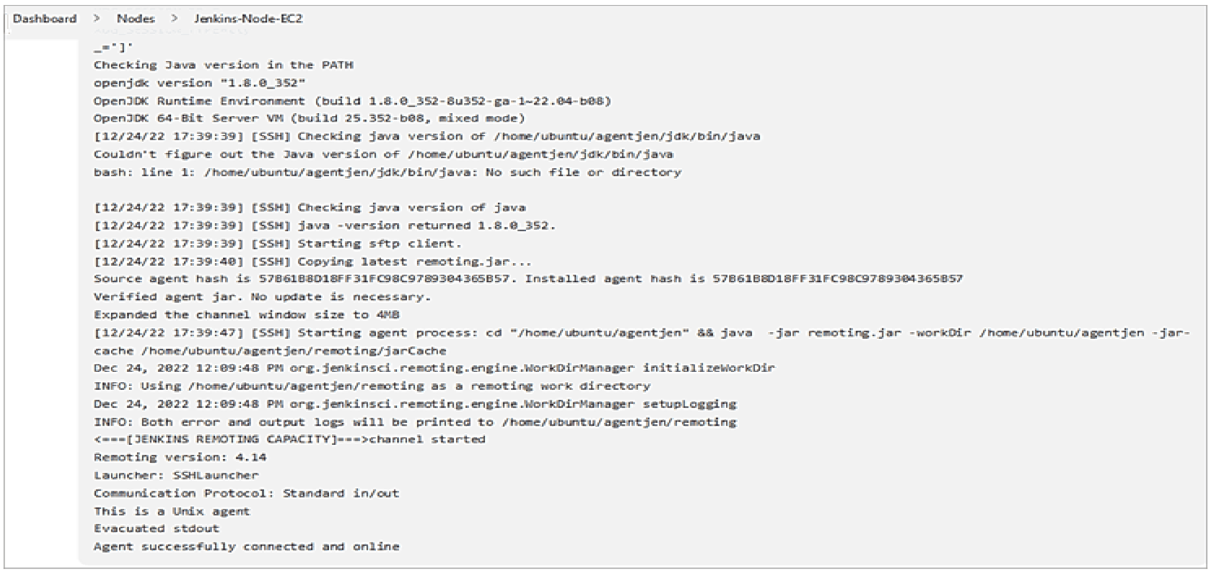
```
C:\Users\ADMIN\.ssh>type id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
[illegible]
-----END OPENSSH PRIVATE KEY-----

C:\Users\ADMIN\.ssh>
```

Next, add the Host Key Verification Strategy. Select the option Select Known hosts file verification strategy. Click on Save.



Host Key Verification Strategy: Known hosts file Verification Strategy

Launch the node and you should see a successfully connected message.



```
_="]'
Checking Java version in the PATH
openjdk version "1.8.0_352"
OpenJDK Runtime Environment (build 1.8.0_352-8u352-ga-1~22.04-b08)
OpenJDK 64-Bit Server VM (build 25.352-b08, mixed mode)
[12/24/22 17:39:39] [SSH] Checking java version of /home/ubuntu/agentjen/jdk/bin/java
Couldn't figure out the Java version of /home/ubuntu/agentjen/jdk/bin/java
bash: line 1: /home/ubuntu/agentjen/jdk/bin/java: No such file or directory

[12/24/22 17:39:39] [SSH] Checking java version of java
[12/24/22 17:39:39] [SSH] java -version returned 1.8.0_352.
[12/24/22 17:39:39] [SSH] Starting sftp client.
[12/24/22 17:39:40] [SSH] Copying latest remoting.jar...
Source agent hash is 57861B8D18FF31FC98C9789304365B57. Installed agent hash is 57861B8D18FF31FC98C9789304365B57
Verified agent jar. No update is necessary.
Expanded the channel window size to 4MB
[12/24/22 17:39:47] [SSH] Starting agent process: cd "/home/ubuntu/agentjen" && java  -jar remoting.jar -workDir /home/ubuntu/agentjen -jar-
cache /home/ubuntu/agentjen/remoting/jarCache
Dec 24, 2022 12:09:48 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/agentjen/remoting as a remoting work directory
Dec 24, 2022 12:09:48 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/agentjen/remoting
<===[JENKINS REMOTING CAPACITY]===>channel started
Remoting version: 4.14
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
Evacuated stdout
Agent successfully connected and online
```

## Configure Docker Container as Jenkins Build Slave

To configure the docker container as a build slave we will need a machine with Docker installed to perform docker build, launch containers, and push to Docker Hub.

Docker remote API should be configured and enabled for Jenkins to communicate with the machine with Docker installed. This should be done before configuring the agent.

I am using an AWS EC2 Ubuntu VM with Docker installed.

#1) Log in to the EC2 VM and open the docker service file /lib/systemd/system/docker.service

Search for ExecStart and replace that line with the following:

ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2376 -H unix:///var/run/docker.sock

#2) Reload and restart the docker service

$ sudo systemctl daemon-reload

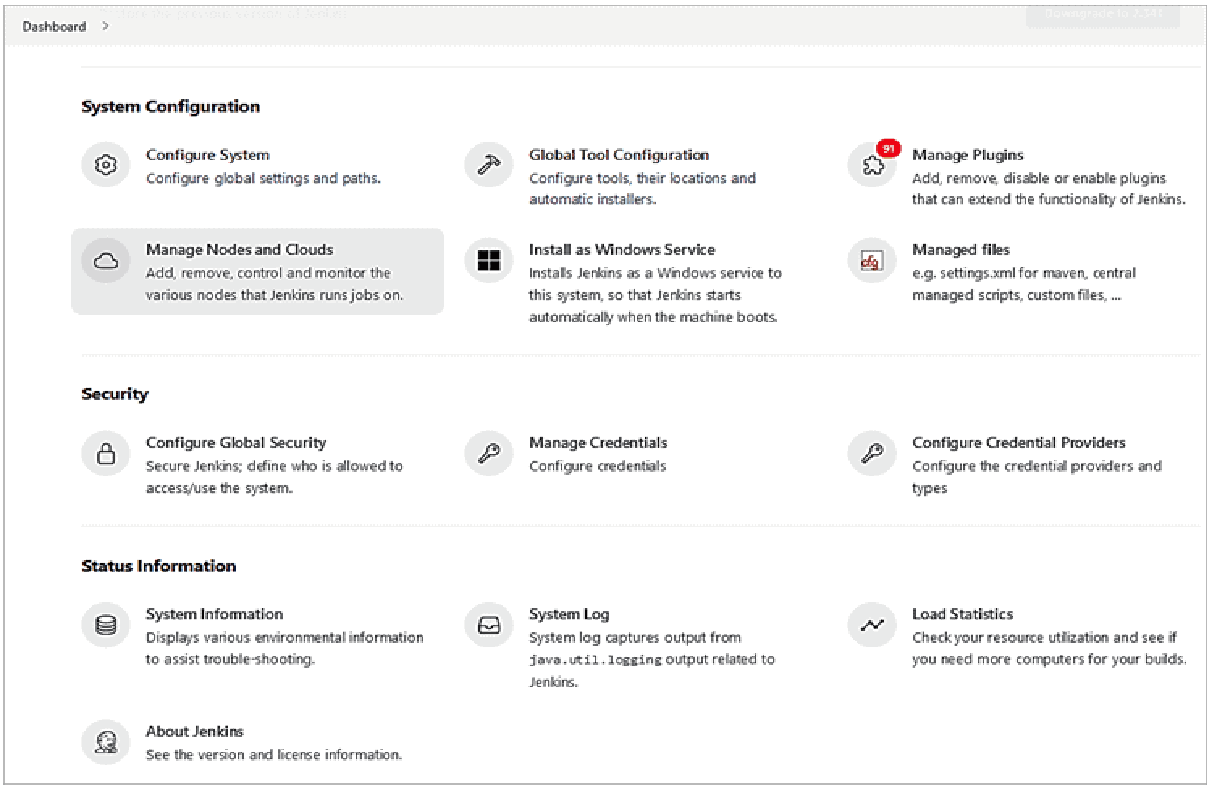
$ sudo service docker restart

#3) Check and Validate the API by executing the following curl commands

curl http://localhost:2376/version

curl http:// <EC2VM-IP>:2376/version

Step #1: Go to Manage Jenkins -> Manage Nodes and Cloud


Dashboard >

**System Configuration**

Configure System
Configure global settings and paths.

Global Tool Configuration
Configure tools, their locations and automatic installers.

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

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

Install as Windows Service
Installs Jenkins as a Windows service to this system, so that Jenkins starts automatically when the machine boots.

Managed files
e.g. settings.xml for maven, central managed scripts, custom files, ...

**Security**

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Manage Credentials
Configure credentials

Configure Credential Providers
Configure the credential providers and types

**Status Information**

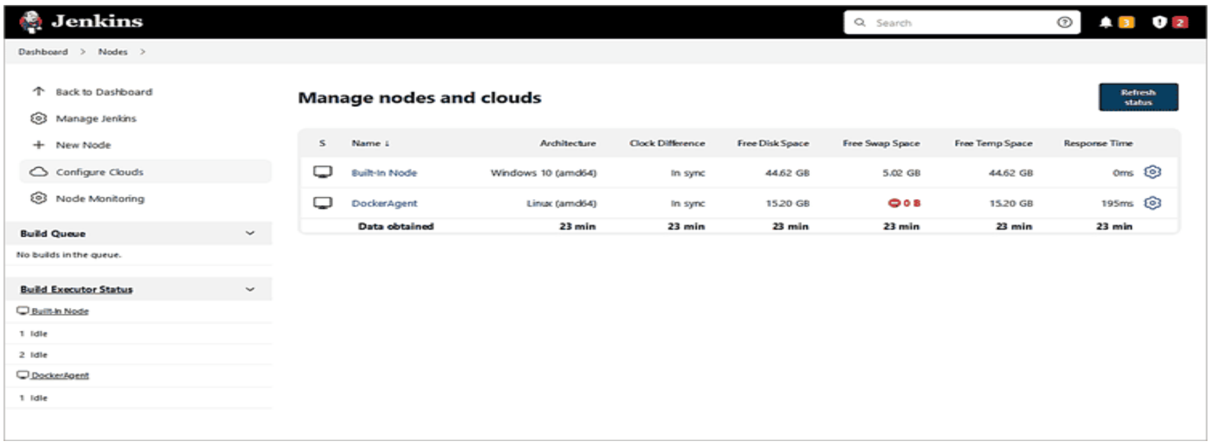
System Information
Displays various environmental information to assist trouble-shooting.

System Log
System log captures output from `java.util.logging` output related to Jenkins.

Load Statistics
Check your resource utilization and see if you need more computers for your builds.

About Jenkins
See the version and license information.

Step #2: Click on Configure Clouds in the left panel


Manage nodes and clouds

| S | Name ↓ | Architecture | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Space | Response Time |
|---|---|---|---|---|---|---|---|
|  | Built-In Node | Windows 10 (amd64) | In sync | 44.62 GB | 5.02 GB | 44.62 GB | 0ms |
|  | DockerAgent | Linux (amd64) | In sync | 15.20 GB | 0 B | 15.20 GB | 195ms |
|  | Data obtained | 23 min | 23 min | 23 min | 23 min | 23 min | 23 min |

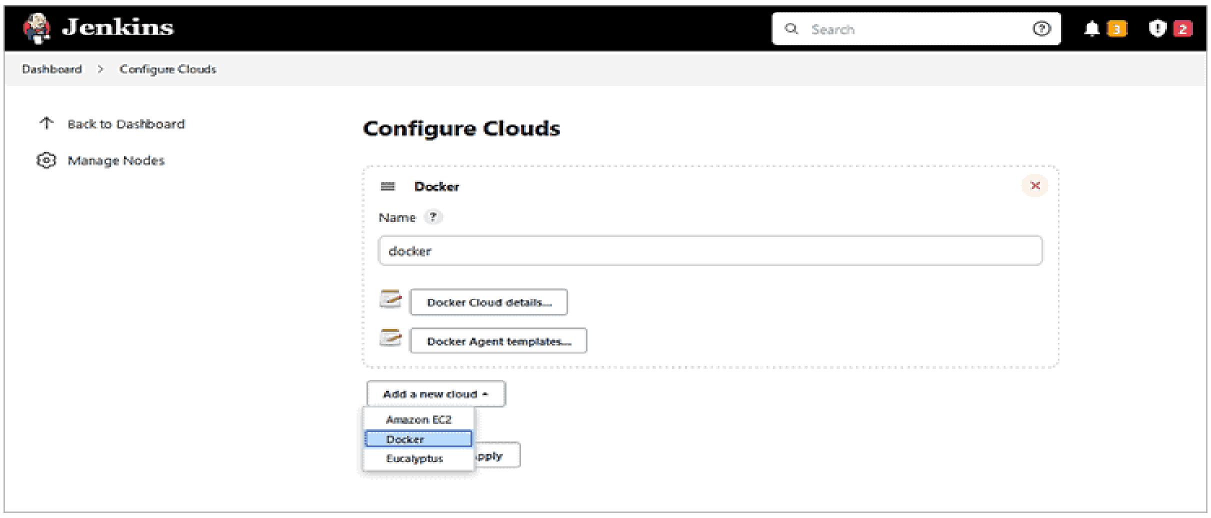
Build Queue

No builds in the queue.

Build Executor Status

Built-In Node
1 Idle
2 Idle
DockerAgent
1 Idle

Step #3: Click on Add a new cloud -> Docker

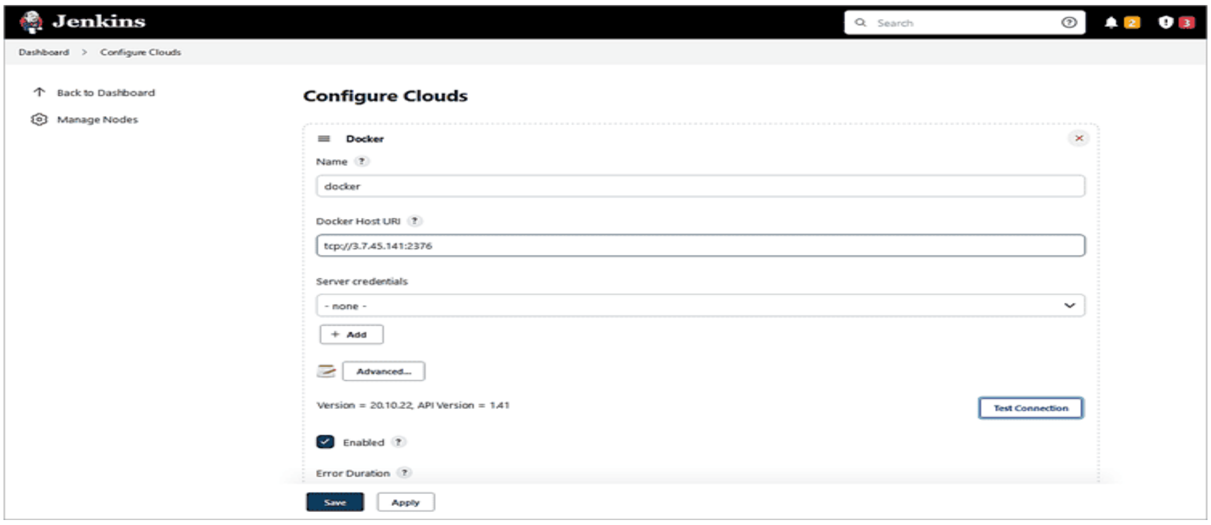

Configure Clouds

Docker

Name: docker

Click on Docker Cloud details

Step #4: Add Name and URI


Configure Clouds

Docker

Name: docker

Docker Host URI: tcp://3.7.45.141:2376

Server credentials: - none -

Version = 20.10.22, API Version = 1.41

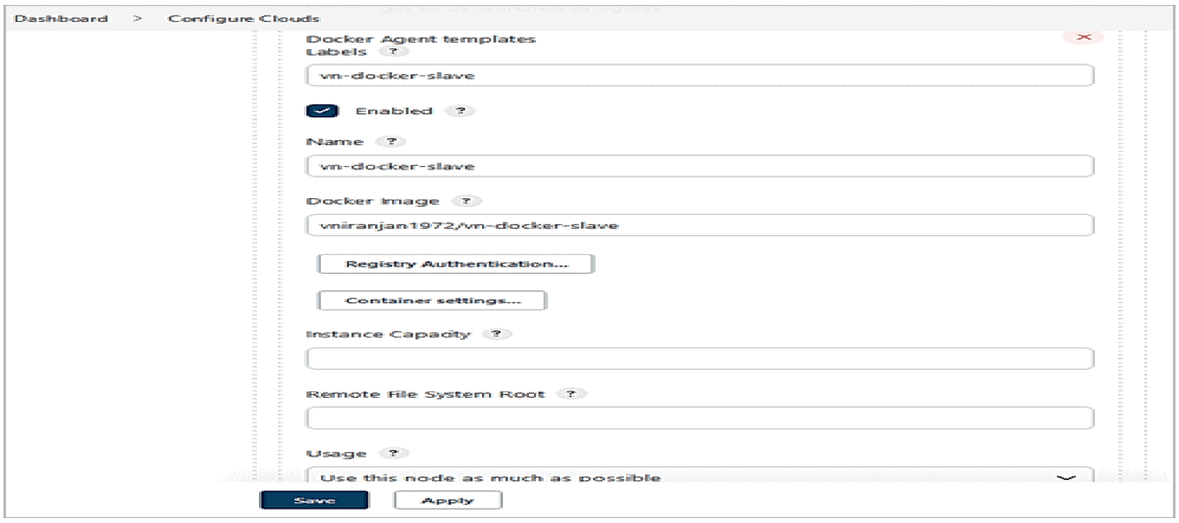
Enabled ✓

Error Duration

You can use the “Test connection” to test if Jenkins is able to connect to the Docker host.

Step #5: Add Docker Agent templates

Click on Docker Agent templates



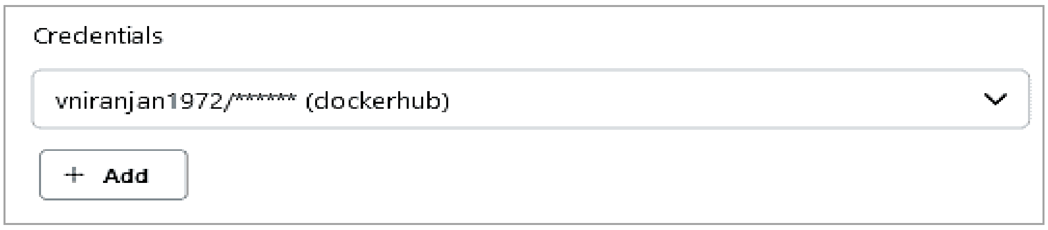
Dashboard > Configure Clouds

Docker Agent templates
Labels: vn-docker-slave
Enabled: ✓
Name: vn-docker-slave
Docker Image: vniranjan1972/vn-docker-slave
Registry Authentication...
Container settings...
Instance Capacity:
Remote File System Root:
Usage: Use this node as much as possible

Save Apply

Note the label provided. This will be provided later in the Jenkins job.

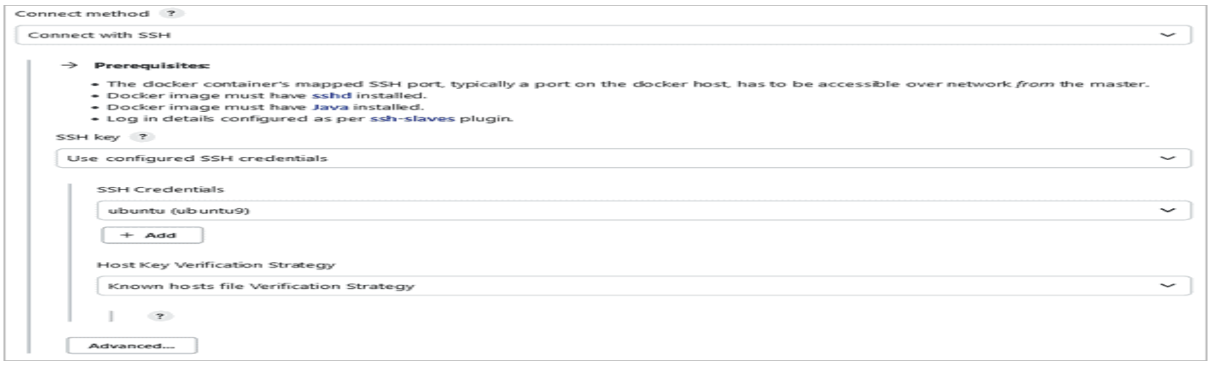
Click on Registry Authentication and enter Docker Hub credentials. Here I have taken sample user name as vniranjan1972.



Credentials: vniranjan1972/****** (dockerhub)

+ Add

Select the Connect method -> Connect with SSH.



Connect method: Connect with SSH

→ **Prerequisites:**
- The docker container's mapped SSH port, typically a port on the docker host, has to be accessible over network *from* the master.
- Docker image must have sshd installed.
- Docker image must have Java installed.
- Log in details configured as per ssh-slaves plugin.

SSH key: Use configured SSH credentials
SSH Credentials: ubuntu (ubuntu9)
+ Add
Host Key Verification Strategy: Known hosts file Verification Strategy

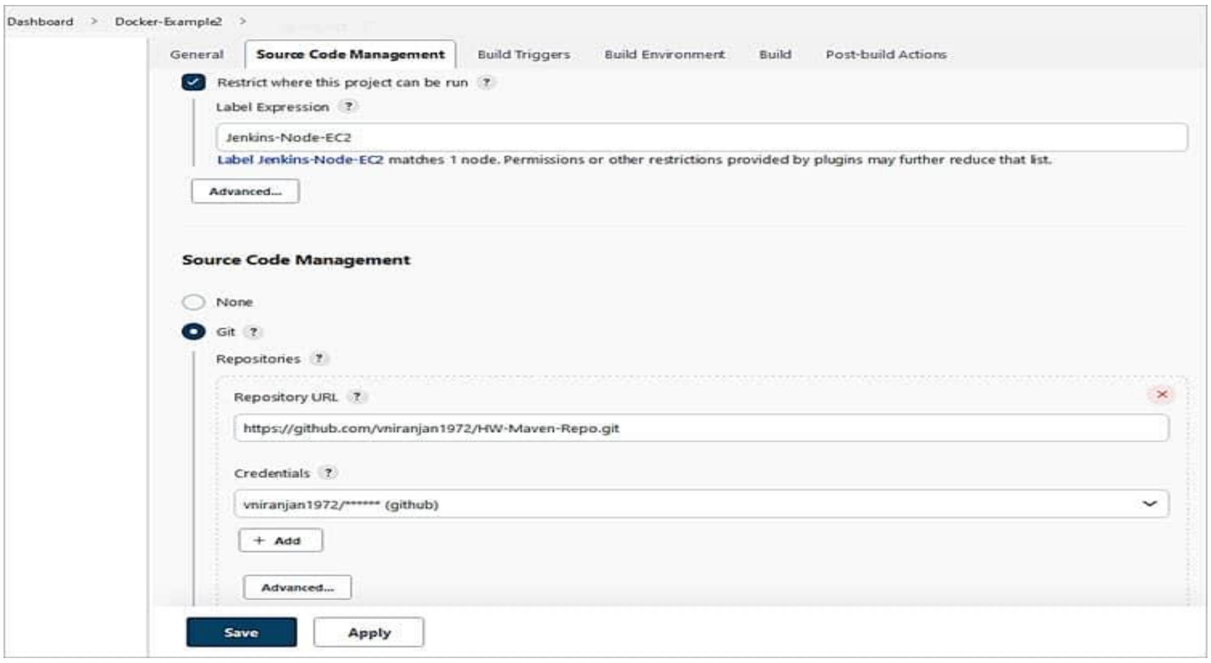
Advanced...

For the SSH key select use configured SSH credentials. This was configured in the earlier section

Click on Save.

## Create Jenkins job

Create a Jenkins freestyle job and enter the label expression for the project to run and configure the GitHub repo.



Restrict where this project can be run ✓

Label Expression: Jenkins-Node-EC2

Label Jenkins-Node-EC2 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.

Source Code Management

None

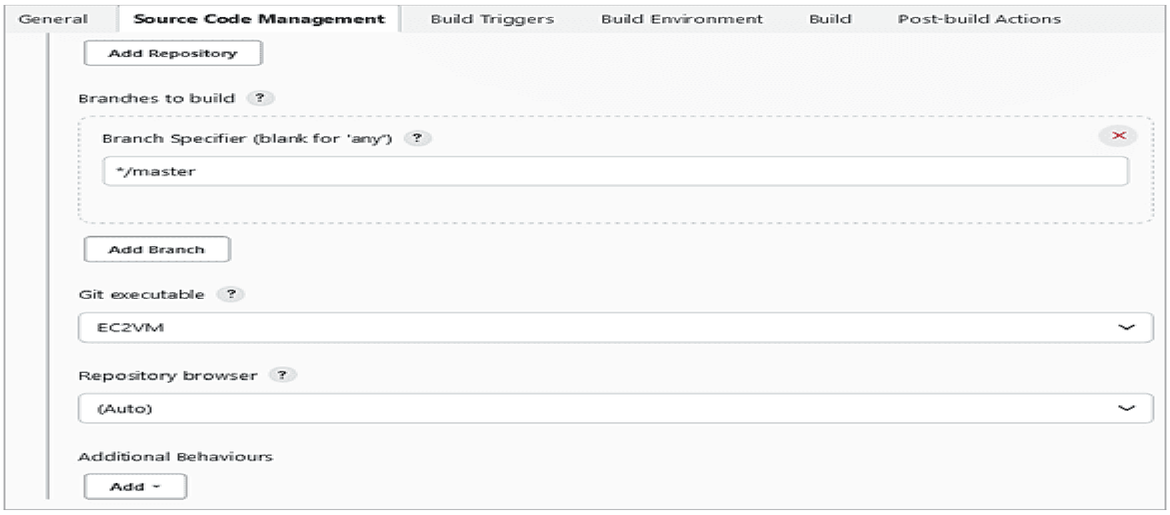
[✓ Git]

Repositories

Repository URL: https://github.com/vniranjan1972/HW-Maven-Repo.git

Credentials: vniranjan1972/****** (github)

Ensure the Git executables are configured under Manage Jenkins -> Global Tool Configuration



Branches to build

Branch Specifier (blank for 'any'): */master

Git executable: EC2VM

Repository browser: (Auto)

Additional Behaviours

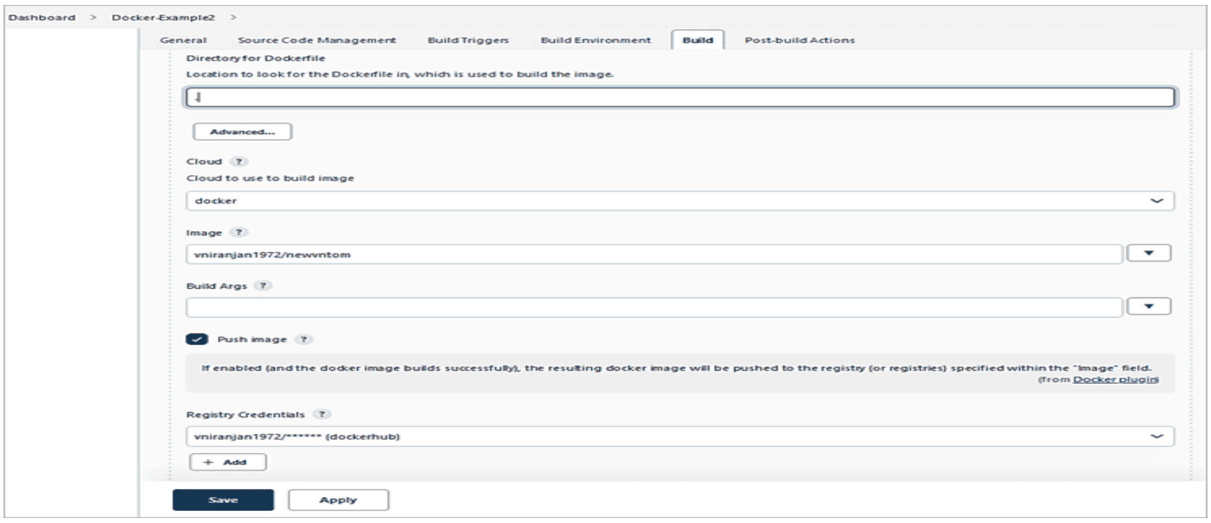
Next, add the build step Build/Publish Docker Image

Under location for Dockerfile enter DOT (.). The Dockerfile which contains commands to build the image is at the root of the GitHub repository

Under Cloud select the docker cloud that was added in the previous step

Enter the image name

Select the check box Push image and select the Registry credentials ( Docker hub credentials)


Dashboard > Docker-Example2 >

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

Directory for Dockerfile
Location to look for the Dockerfile in, which is used to build the image.
.

Cloud
Cloud to use to build image
docker

Image
vniranjan1972/newvntom

Build Args

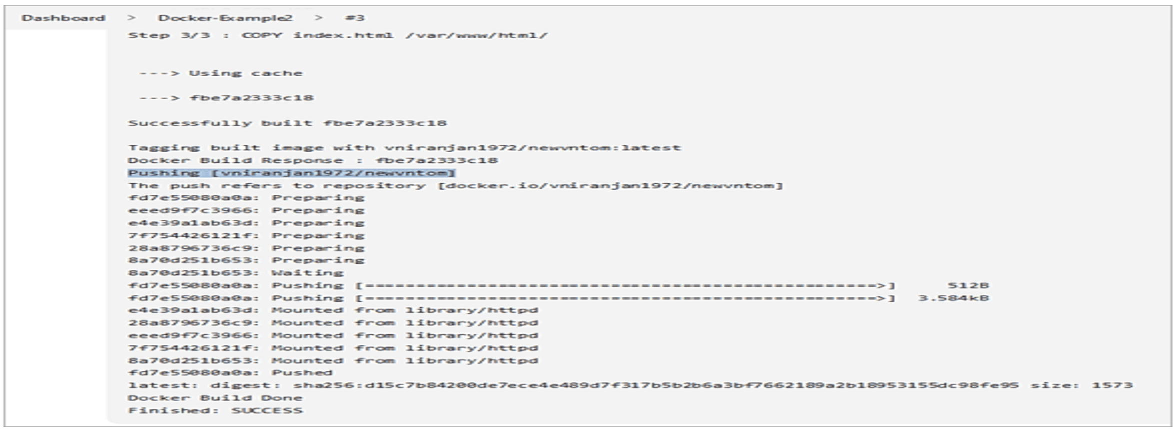
✓ Push image

If enabled (and the docker image builds successfully), the resulting docker image will be pushed to the registry (or registries) specified within the "Image" field.
(from Docker plugin)

Registry Credentials
vniranjan1972/****** (dockerhub)

Click on Save to trigger a build.

Console Output.


Dashboard > Docker-Example2 > #3

```
Step 3/3 : COPY index.html /var/www/html/

 ---> Using cache

 ---> fbe7a2333c18

Successfully built fbe7a2333c18

Tagging built image with vniranjan1972/newvntom:latest
Docker Build Response : fbe7a2333c18
Pushing [vniranjan1972/newvntom]
The push refers to repository [docker.io/vniranjan1972/newvntom]
fd7e55080a0a: Preparing
eeed9f7c3966: Preparing
e4e39a1ab63d: Preparing
7f754426121f: Preparing
28a8796736c9: Preparing
8a70d251b653: Preparing
8a70d251b653: Waiting
fd7e55080a0a: Pushing [==================================================>]     512B
fd7e55080a0a: Pushing [==================================================>]  3.584kB
e4e39a1ab63d: Mounted from library/httpd
28a8796736c9: Mounted from library/httpd
eeed9f7c3966: Mounted from library/httpd
7f754426121f: Mounted from library/httpd
8a70d251b653: Mounted from library/httpd
fd7e55080a0a: Pushed
latest: digest: sha256:d15c7b84200de7ece4e489d7f317b5b2b6a3bf7662189a2b18953155dc98fe95 size: 1573
Docker Build Done
Finished: SUCCESS
```

Once the image is pushed to Docker Hub please check with the registry.

## Jenkins And Docker Compose Integration

Docker Compose is used to run multiple containers using a single service. Docker compose files are written using YAML files and all the services (containers) can be started and stopped with the following commands

docker-compose up -d // To start in detached mode

docker-compose down

## Install docker-compose

Use the following command:

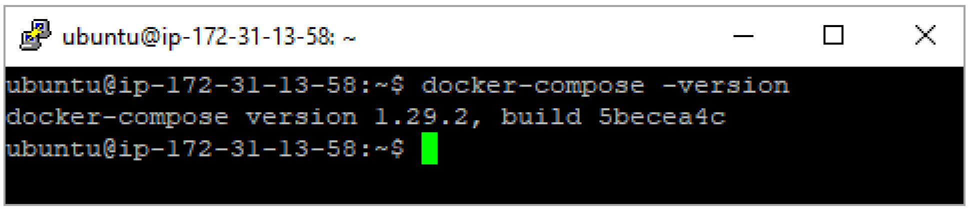
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

The command instructs the system to save the file in the /usr/local/bin/ directory, under the name docker-compose. Make the downloaded file executable by changing the file permissions with:

sudo chmod +x /usr/local/bin/docker-compose

Run the command to check the version of docker compose

$ docker–compose –version



```
ubuntu@ip-172-31-13-58:~$ docker-compose -version
docker-compose version 1.29.2, build 5becea4c
ubuntu@ip-172-31-13-58:~$
```

## Dockerfile and docker-compose.yml

Dockerfile

FROM centos

MAINTAINER Vasu Niranjan vniranjan72@outlook.com

RUN mkdir /opt/tomcat

WORKDIR /opt/tomcat

RUN curl -O https://downloads.apache.org/tomcat/tomcat-9/v9.0.71/bin/apache-tomcat-9.0.71.tar.gz

RUN tar -xvzf apache-tomcat-9.0.71.tar.gz

RUN mv apache-tomcat-9.0.71/* /opt/tomcat/.

RUN cd /etc/yum.repos.d/

RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*

RUN sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-*

RUN yum update -y

RUN yum install java -y

RUN java -version

WORKDIR /opt/tomcat/webapps

ADD ["target/HelloWorld-Maven.war", "/opt/tomcat/webapps"]

EXPOSE 8080

CMD ["/opt/tomcat/bin/catalina.sh", "run"]

docker-compose.yml file

version: ‘3.8’
services:
web:
build: .
ports:
– 9000:8080