



Course: DevOps Laboratory

Code: BIT26VS01

Name: Amar Vaijinath Chavan

PRN: 124B2F001

Assignment 3: Study of Jenkins Architecture and Plugins with Installation and Configuration.

Aim: To install Jenkins on an AWS EC2 Master node, configure Java on both Master and Worker nodes, and establish a secure connection between them using SSH keys.

Objectives:

1. To understand the concept of Continuous Integration/Continuous Deployment (CI/CD).
2. To study Jenkins Controller-Agent (Master-Slave) architecture.
3. To install and configure Jenkins with essential plugins for Java/Maven projects.

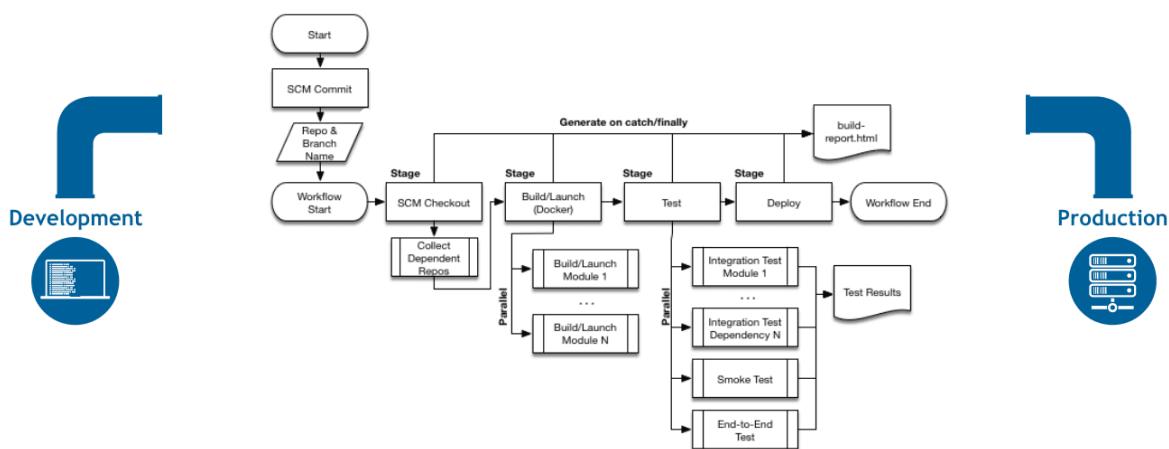
Prerequisites:

- AWS Account with two EC2 Instances (Ubuntu 24.04).
- Security Groups configured for Ports 8080 (Jenkins UI), 22 (SSH), and 8081 (Application).
- Java Development Kit (JDK 21) installed on both nodes.

Theory:

1. Jenkins Overview

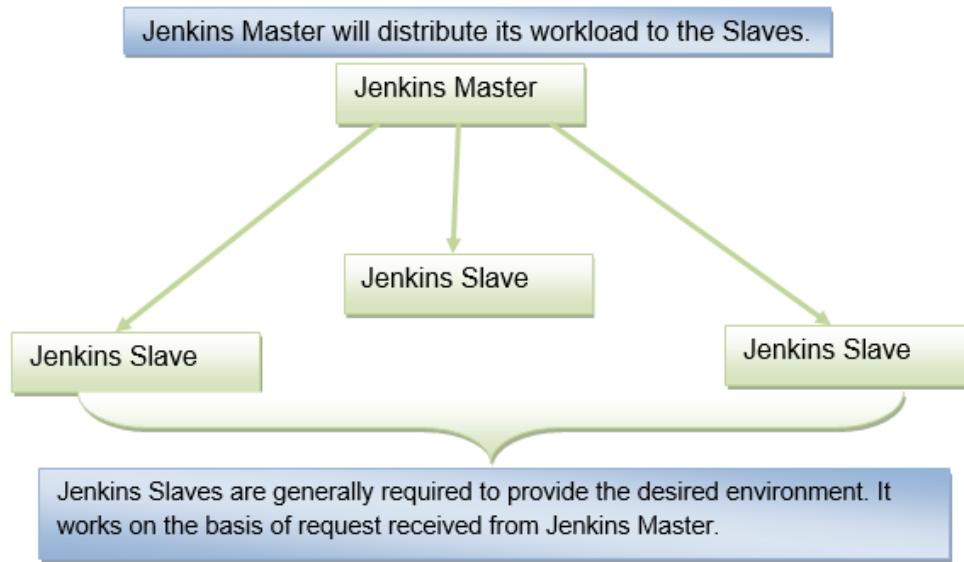
Jenkins is an open-source automation server that facilitates the technical aspects of Continuous Integration and Continuous Delivery. It automates the process of building, testing, and deploying software, allowing teams to focus on code quality.



2. Jenkins Architecture (Controller-Agent)

Jenkins follows a distributed architecture to handle high workloads:

- **Controller (Master):** The "Brain." It manages the configuration, scheduling builds, and monitoring agents.
- **Agent (Worker):** The "Muscle." It executes the actual build commands sent by the controller. This prevents the controller from becoming a bottleneck.



3. Jenkins Plugins

Plugins are the heart of Jenkins' extensibility. They allow integration with various tools:

- **Git Plugin:** Enables Jenkins to pull source code from GitHub.
- **Maven Integration:** Specifically designed for building Java-based projects.
- **SSH Build Agents:** Used to connect the Controller to remote Workers via SSH.
- **Pipeline Plugin:** Allows defining the CI/CD flow as code (Jenkinsfile).

Practical Procedure / Steps:

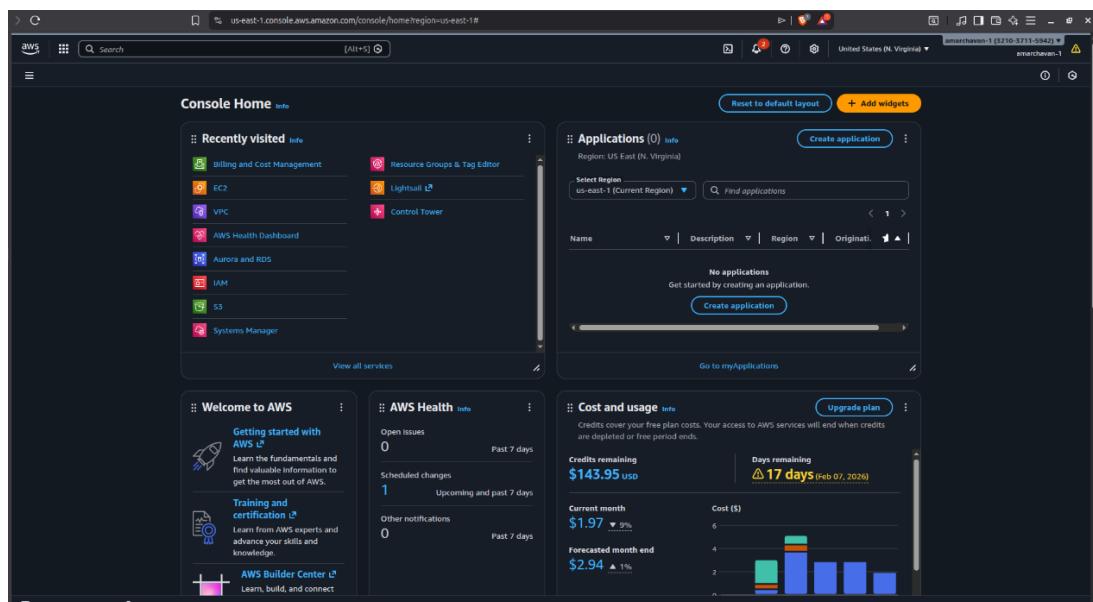
Step 1: AWS EC2 Instance Creation

Before installing Jenkins, we must provision the virtual servers. Repeat these steps for both the **Master** and **Worker** nodes.

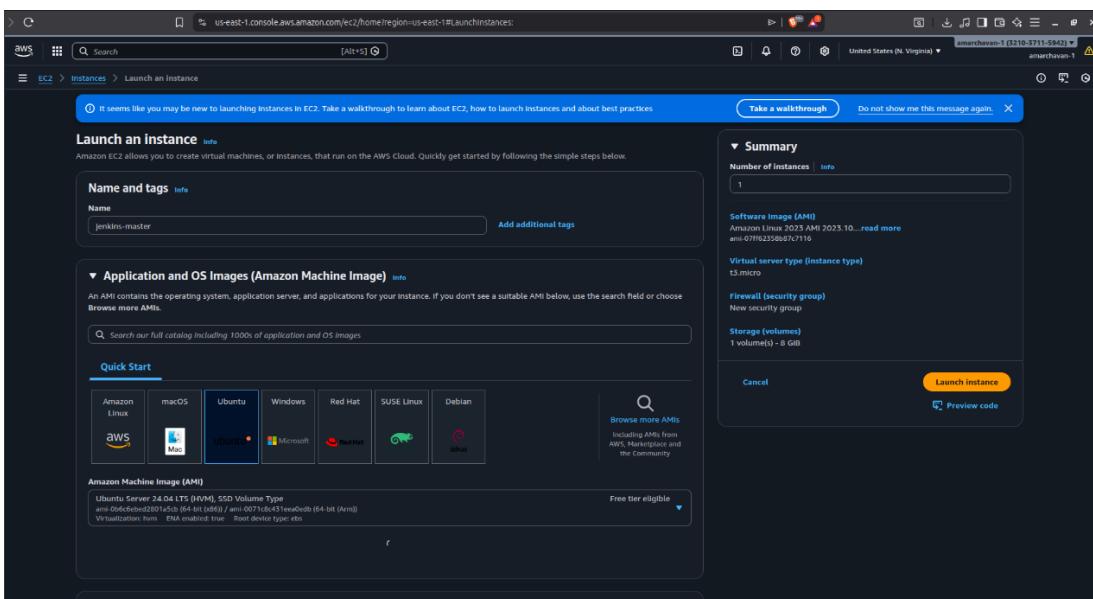
1. **Sign in to AWS Console:** Navigate to the EC2 Dashboard.
2. **Launch Instance:** Click on the "Launch Instance" button.
3. **Name and Tags:** Name the first instance: Jenkins-Master.
Name the second instance: Jenkins-Worker.
4. **Application and OS Image:** Select **Ubuntu 24.04 LTS** (Free Tier eligible).
5. **Instance Type:** Select **t2.micro** (1 vCPU, 1 GiB Memory). [or free eligible]

6. **Key Pair:** Select an existing key pair or create a new one (e.g., devops-lab.pem) to enable SSH access.
7. **Network Settings (Security Groups):** * Create a Security Group named Jenkins-SG.
 - o **Inbound Rules for Master:**
 - Type: **SSH** | Port: **22** | Source: **My IP**
 - Type: **Custom TCP** | Port: **8080** | Source: **Anywhere (0.0.0.0/0)**
 - o **Inbound Rules for Worker:**
 - Type: **SSH** | Port: **22** | Source: **My IP**
 - Type: **Custom TCP** | Port: **8081** | Source: **Anywhere (0.0.0.0/0)** (For your app)
8. **Launch:** Click "Launch Instance" and wait for the status to change to Running.

Jenkins-Master Creation EC2 Screenshots



The screenshot shows the AWS Console Home page. On the left, there's a sidebar with 'Recently visited' links: Billing and Cost Management, EC2, VPC, AWS Health Dashboard, Aurora and RDS, IAM, S3, and Systems Manager. The main area has three sections: 'Applications (0)' which says 'No applications. Get started by creating an application.', 'Welcome to AWS' with sections for Getting started with AWS Lambda, Training and certification, and AWS Builder Center, and 'AWS Health' which shows 0 open issues and 1 scheduled change. On the right, there's a 'Cost and usage' section with a bar chart showing credits remaining (\$143.95 USD), current month cost (\$1.97), and forecasted month end cost (\$2.94). A note at the bottom says 'Credits cover your free plan costs. Your access to AWS services will end when credits are depleted or free period ends.'



The screenshot shows the 'Launch an instance' wizard. It starts with a welcome message: 'It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices.' Below this, there are sections for 'Name and tags', 'Application and OS Images (Amazon Machine Image)', and 'Amazon Machine Image (AMI)'. In the 'Name and tags' section, 'jenkins-master' is entered. In the 'Application and OS Images (Amazon Machine Image)' section, 'Amazon Linux 2023.10...' is selected. The 'Amazon Machine Image (AMI)' section shows 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' as the selected AMI. On the right, there's a summary section with fields for 'Number of instances' (1), 'Software Image (AMI)' (Amazon Linux 2023.10...), 'Virtual server type (instance type)' (t3.micro), 'Firewall (security group)' (New security group), and 'Storage (volumes)' (1 volume(s) ~ 8 GiB). At the bottom right are 'Launch instance' and 'Preview code' buttons.

The screenshot shows the AWS EC2 Instances launch configuration. On the left, there's a sidebar with a search bar and navigation links for 'EC2 > Instances > Launch an instance'. The main area is titled 'Launch an instance' with a sub-section 'Configure instance details'. A 'Security group' dropdown is set to 'sg-12345678'. Below it, four security group rules are listed:

- Security group rule 1 (TCP, 22, 0.0.0.0/0): Type: Custom TCP, Protocol: TCP, Port range: 22, Source: Anywhere, Description: e.g. SSH for admin desktop.
- Security group rule 2 (TCP, 8080, 0.0.0.0/0): Type: Custom TCP, Protocol: TCP, Port range: 8080, Source: Anywhere, Description: e.g. SSH for admin desktop.
- Security group rule 3 (TCP, 80, 0.0.0.0/0): Type: HTTP, Protocol: TCP, Port range: 80, Source: Anywhere, Description: e.g. SSH for admin desktop.
- Security group rule 4 (TCP, 443, 0.0.0.0/0): Type: HTTPS, Protocol: TCP, Port range: 443, Source: Anywhere, Description: e.g. SSH for admin desktop.

To the right, a 'Summary' section shows:

- Number of instances: 1
- Software Image (AMI): Canonical, Ubuntu, 24.04, amd64... (read more)
- Virtual server type (instance type): m7c4xlarge
- Firewall (security group): New security group
- Storage (volumes): 1 volume(s) - 8 GiB

Buttons at the bottom include 'Cancel', 'Launch instance' (highlighted in orange), and 'Preview code'.

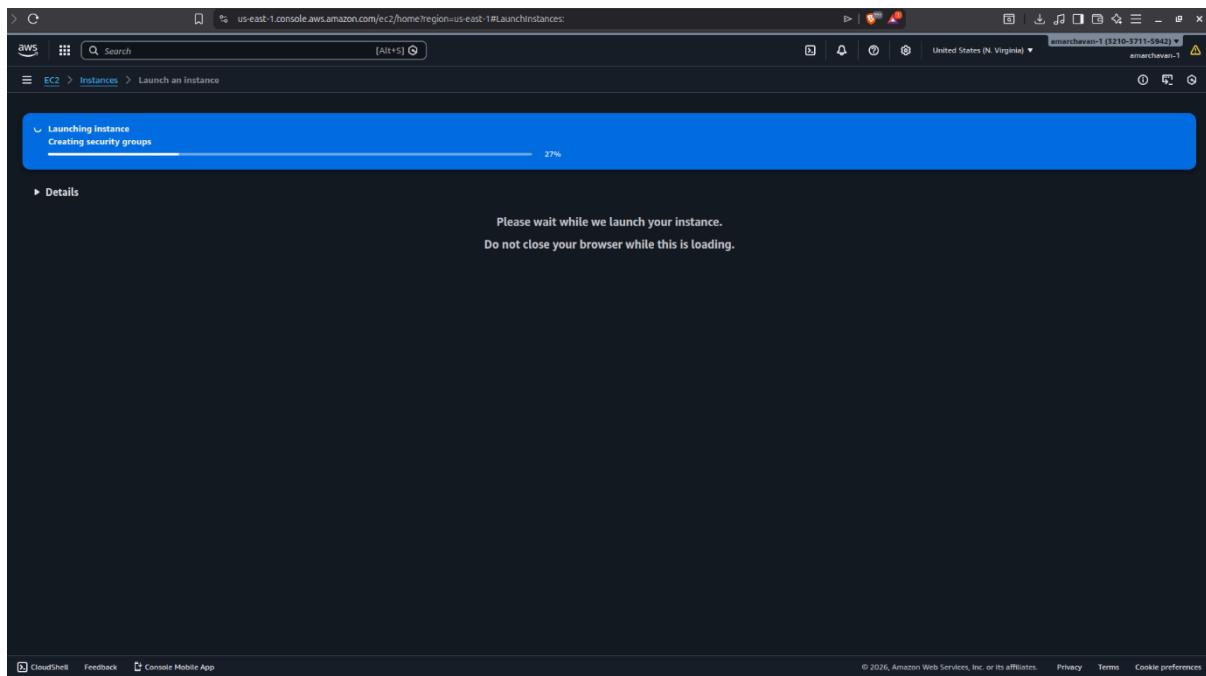
The screenshot shows the 'Details' tab of the launch progress. At the top, a blue header bar indicates 'Creating security groups' and shows a progress bar at 27%. The main content area has a dark background with white text:

Please wait while we launch your instance.
Do not close your browser while this is loading.

Jenkins-Worker (Slave) Creation EC2 Screenshots

This screenshot shows the 'Launch an instance' wizard on the AWS EC2 console. The user has selected the 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' AMI. The 'Name' field contains 'jenkins-worker-1'. The 'Software image (AMI)' section shows 'Canonical, Ubuntu, 24.04, amd64' with a link to 'Read more'. The 'Virtual server type (instance type)' is set to 't3.micro'. Under 'Storage (volumes)', there is one volume of 8 GiB. The 'Launch instance' button is highlighted in orange at the bottom right.

This screenshot shows the continuation of the 'Launch an instance' wizard. The 'Key pair (login)' section requires a key pair, which is currently 'jenkins-worker-1'. The 'Network settings' section includes configuration for a VPC (subnet: 'subnet-005df75bc0ca79fb'), auto-assigning a public IP, and creating a new security group named 'launch-wizard-5'. The 'Inbound Security Group Rules' table shows a single rule allowing SSH traffic (TCP port 22). The 'Launch instance' button is again highlighted in orange.



Step 2: Installation on Master Node

Once the instances are running, connect to the **Master** via SSH and execute:

1. Install Java (Prerequisite):

Update the Debian apt repositories, install OpenJDK 21, and check the installation using the following commands:

```
sudo apt update  
sudo apt install fontconfig openjdk-21-jre  
java -version
```

2. Install Jenkins:

A LTS (Long-Term Support) release is chosen every 12 weeks from the stream of regular releases as the stable release for that time period. It can be installed from the debian-stable apt repository.

```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \  
https://pkg.jenkins.io/debian-stable/jenkins.io-2026.key  
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \  
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
sudo apt update
```

```
sudo apt install Jenkins
```

3. Check Status:

```
sudo systemctl status Jenkins
```

Screenshots of Installation on the Mater Node

Step 3: Configuration on Worker Node

Connect to the **Worker** via SSH and only install the Java runtime:

Install Java:

```
sudo apt update  
sudo apt install fontconfig openjdk-21-jre  
java -version
```

Screenshots of Configuration on Worker Node

```
ubuntu@ip-172-31-0-106:~  
amar@amar-Inspiron-3501:~/Desktop/AWS-EC2-Jenkins$ chmod 400 "jenkins-worker-1.pem"  
amar@amar-Inspiron-3501:~/Desktop/AWS-EC2-Jenkins$ ssh -o ServerAliveInterval=60 -i "jenkins-worker-1.pem" ubuntu@ec2-18-207-123-211.compute-1.amazonaws.com  
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1018-aws x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/pro  
  
System information as of Fri Jan 23 09:38:38 UTC 2026  
  
System load: 0.05 Temperature: -273.1 °C  
Usage of /: 25.9% of 6.71GB Processes: 114  
Memory usage: 23% Users logged in: 0  
Swap usage: 0% IPv4 address for ens5: 172.31.6.4  
  
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.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.
```

```
ubuntu@ip-172-31-0-106:~  
ubuntu@ip-172-31-6-4:~  
ubuntu@ip-172-31-0-106:~  
sudo apt update  
sudo apt install fontconfig openjdk-21-jre  
java -version  
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease  
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]  
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]  
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [126 kB]  
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]  
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]  
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]  
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [391 kB]  
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
```

```
ubuntu@ip-172-31-0-106:~  
ubuntu@ip-172-31-6-4:~  
ubuntu@ip-172-31-0-106:~  
$ java --version  
openjdk 21.0.9 2025-10-21  
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)  
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)  
ubuntu@ip-172-31-6-4:~
```

Step 4: Establishing Master-Worker Connectivity (SSH Handshake)

This step connects the "Master Node" to the "Worker Node."

1. Generate SSH Keys on Master:

Shh -keygen

```

ubuntu@Jenkins-Master: ~/.ssh
ubuntu@Jenkins-Master: $ cd ~/.ssh
ubuntu@Jenkins-Master: ~/ssh$ ls
authorized_keys
ubuntu@Jenkins-Master: ~/ssh$ 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:odWn2b0C0dg0nDBV77hTyB1JEEdHgRqY/B+T3DvXoqw ubuntu@Jenkins-Master
The key's randomart image is:
+-- [ED25519 256] --
|   o += +*=|
| . * .+o o|
| + o =.o ..|
| o o B *+.+|
| . S = o =.+|
| + . .+=|
| . . =*..|
| . .oo|
|       E.o. ...|
+--- [SHA256] ----+
ubuntu@Jenkins-Master: ~/ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
ubuntu@Jenkins-Master: ~/ssh$
```

2. Authorize Master on Worker:

- Copy the content of /var/lib/jenkins/.ssh/id_rsa.pub (Master).
- On the Worker node, open ~/.ssh/authorized_keys and paste the key

```

ubuntu@Jenkins-Master: ~/.ssh
ubuntu@Jenkins-Master: $ cd ~/.ssh
ubuntu@Jenkins-Master: ~/ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
ubuntu@Jenkins-Master: ~/ssh$ cat id_ed25519
-BEGIN OPENSSH PRIVATE KEY-----
bnNzac1rZktdjEAAAABG5vbhJAAAEBm9uzQAAAAAAAABAAAAAMwAAAAtzc2gtZW
UxQQAACCrGqJhk2UhA3Ju2WxgK7YxzqzgkKw85r0w7ALd/SwAAJhVRGS1URK
AA7zc2gtzQyNTUX00AACrGqJhk2UhA3Ju2Wxgk7Yxzgkkrkw85r0w7ALd/Sw
D7JkxStu0IpwXNn1IhwavCb+/LKU2IkysFQgqaugsaoKnGTZd0d0e07Na/GArt
CSsrDvxvDsAt39LAAAAXFXVtdw50dUBKzW5r0w5zLU1hc3Rlcg==
-END OPENSSH PRIVATE KEY-----
ubuntu@Jenkins-Master: ~/ssh$ cat id_ed25519.pub
ed25519 AAAAC3zaI1ZDI1NTESAAAIIKsaokmGTQd0e07Na/GArtjGr0C5srDxKvTDsAt39L  ubuntu@Jenkins-Master
ubuntu@Jenkins-Master: ~/ssh$
```

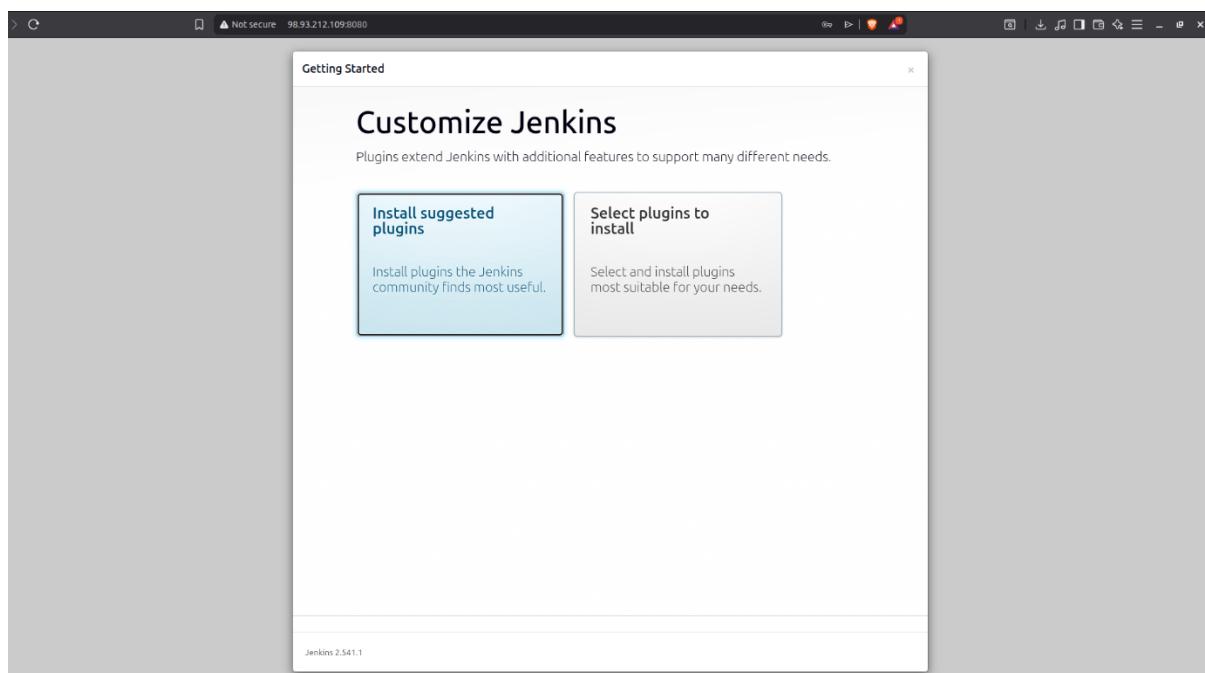
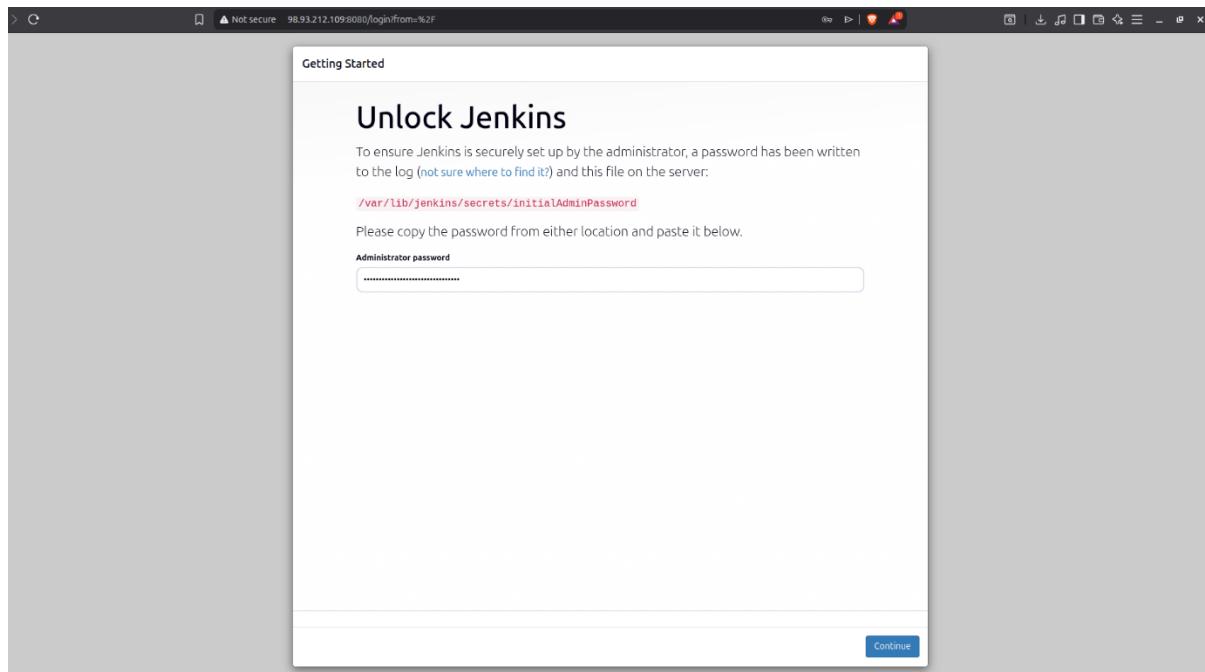
```

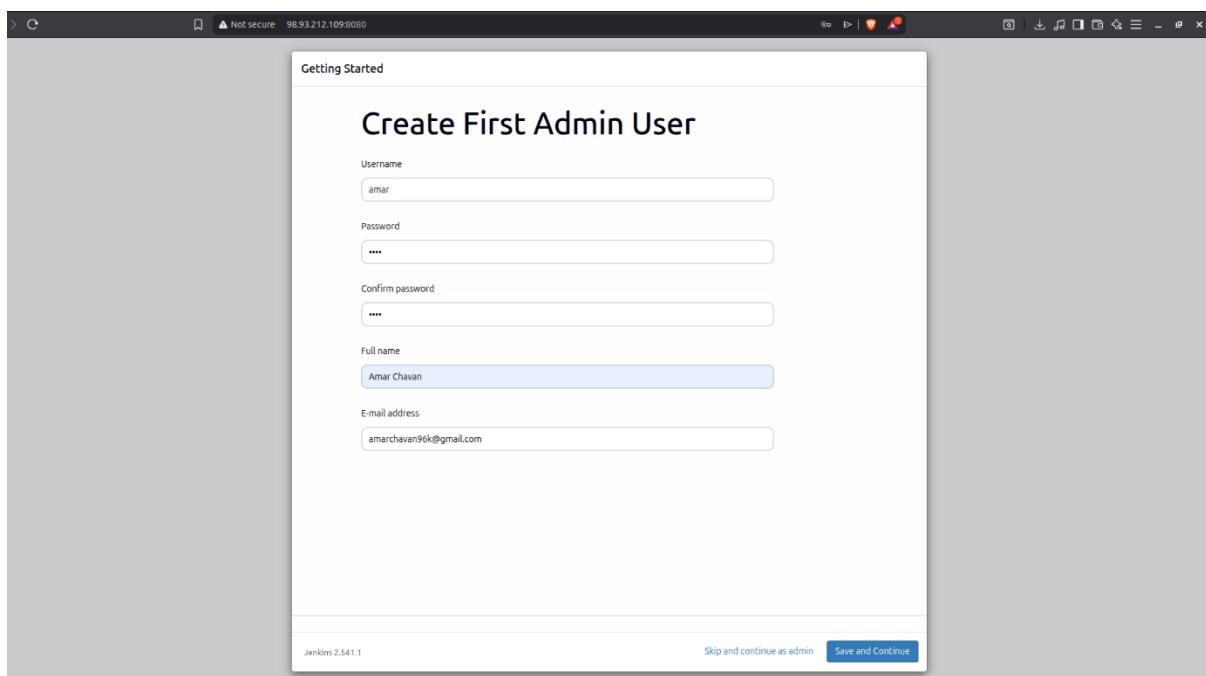
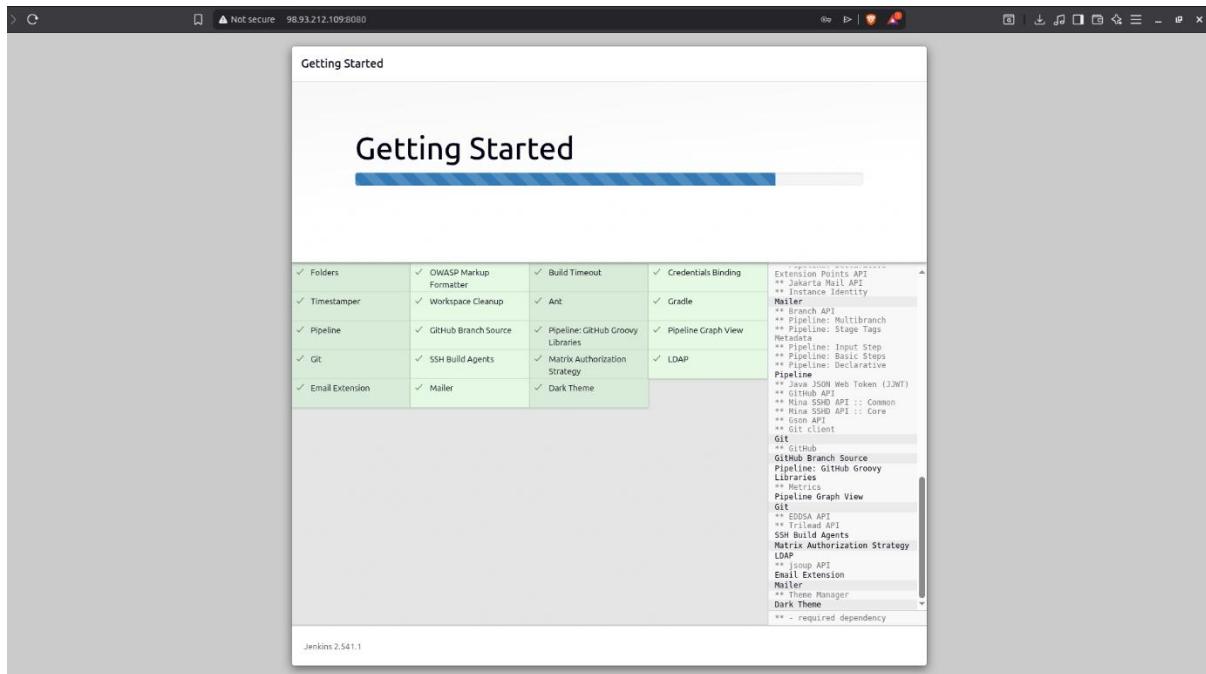
ubuntu@Jenkins-Worker-01: ~
ubuntu@Jenkins-Worker-01: ~/.ssh
ubuntu@Jenkins-Worker-01: $ cd /
ubuntu@Jenkins-Worker-01: $ nano authorized_keys
ubuntu@Jenkins-Worker-01: $ #add
ubuntu@Jenkins-Worker-01: $ #public key of the master node added here for the ssh connection between master node and the worker node
ubuntu@Jenkins-Worker-01: $
```

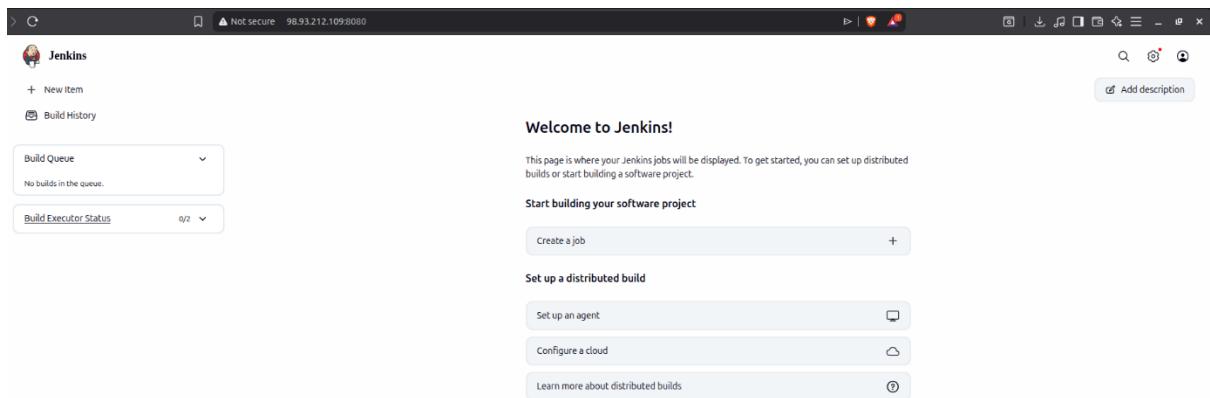
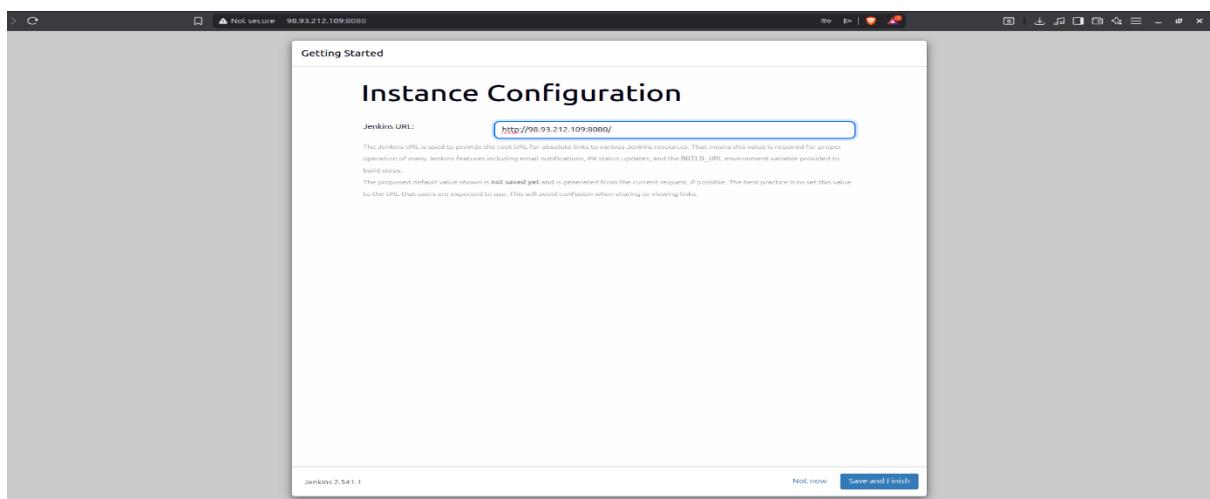
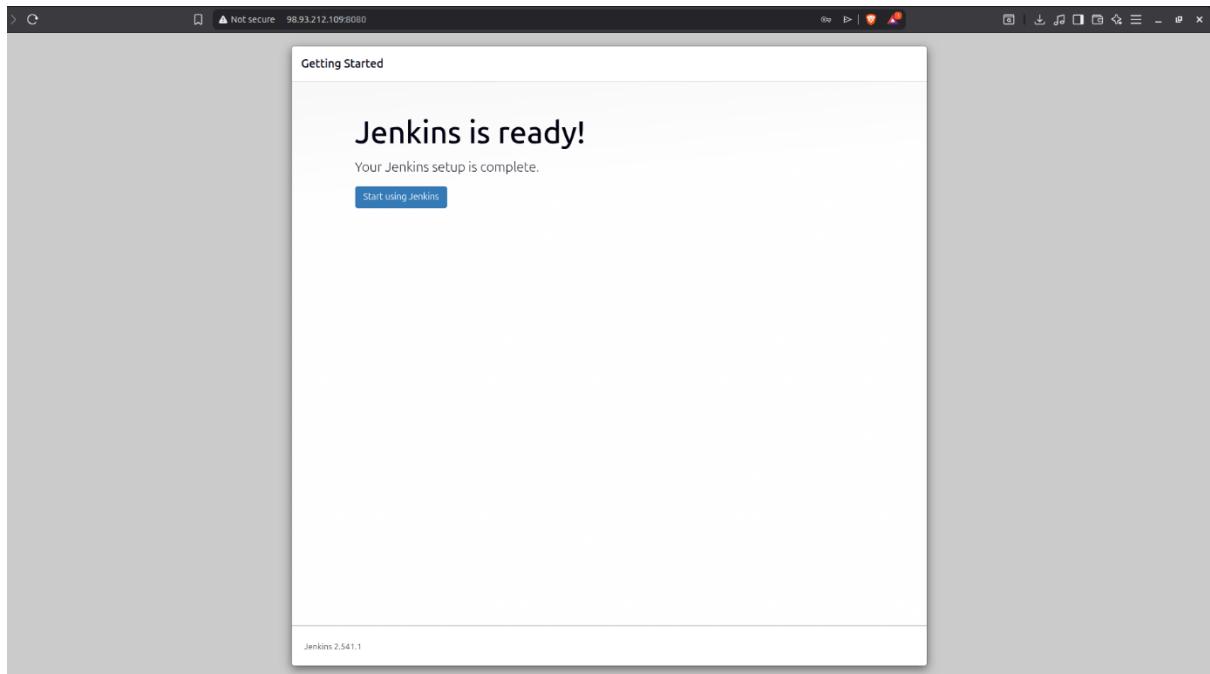
Step 5: Jenkins UI Setup & Node Registration

1. **Access Jenkins:** Open your browser and navigate to `http://<Master-Public-IP>:8080`.
2. **Unlock Jenkins:** Retrieve the initial admin password from the Master terminal: `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
3. **Install Plugins:** Select "Install Suggested Plugins."

Screenshots of Jenkins UI Setup







Navigate to **Manage Jenkins > Nodes > New Node**.

- **Node Name:** Jenkins-Worker-01
- **Type:** Permanent Agent
- **Remote Root Directory:** /home/ubuntu (Create this folder on the worker first).
- **Launch Method:** Select "**Launch agents via SSH**".
- **Host:** <Worker-Private-IP>
- Click Save Button and check the logs of the Node

The screenshots illustrate the Jenkins 'New node' configuration interface. The top screenshot shows the initial setup with basic information like node name and type. The bottom screenshot shows the expanded configuration options, including host, credentials, and availability settings.

Jenkins / Nodes / Jenkins-Worker-01 / Log

```

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
PIPESTATUS={0}="0"
PID=3986
PS4=+
PWD=/home/ubuntu
SHELL=/bin/bash
SHELLOPTS=braceexpand:hashall:interactive-comments
SHLVL=1
SSH_CLIENT='98.93.212.109 40618 22'
SSH_CONNECTION='98.93.212.109 40618 172.31.6.4 22'
TERM=dumb
UID=1000
USER=ubuntu
XDG_RUNTIME_DIR=/run/user/1000
XDG_SESSION_CLASS=user
XDG_SESSION_ID=14
XDG_SESSION_TYPE=tty
=:1'
[01/23/26 10:20:47] [SSH] Starting sftp client.
[01/23/26 10:20:47] [SSH] Copying latest remoting.jar...
[01/23/26 10:20:47] [SSH] Copied 1,407,915 bytes.
Expanded the channel window size to 4MB
[01/23/26 10:20:47] [SSH] Starting agent process: cd "/home/ubuntu" && java -jar remoting.jar -workDir /home/ubuntu -jar-cache /home/ubuntu/remoting/jarCache
Jan 23, 2026 10:20:47 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/ubuntu/remoting as a remoting work directory
Jan 23, 2026 10:20:47 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/ubuntu/remoting
>channel started
Remoting version: 3352.v17a_fb_4b_2773f
Launcher: SSHLauncher
Communication Protocol: Standard in/out
This is a Unix agent
Agent successfully connected and online

```

Important Note on Networking: Although we have two EC2 instances, they are completely independent and "anonymous" to one another. To establish a Master-Worker connection, we use the **Private IP** of the Worker.

- **Why Private IP?** Using the Private IP ensures the communication stays within the AWS internal network (VPC). This is faster, more secure, and free of data transfer costs compared to using the Public IP.
- **The Challenge:** Since the Master doesn't "know" the Worker exists, we provide the Private IP as a direct pointer and the **Private SSH Key** as the digital "security clearance" to bridge the two servers.

- **Credentials:** 1. Click **Add > Jenkins**. 2. Kind: **SSH Username with private key**. 3. Username: `ubuntu` (or the default user of your OS). 4. Private Key: Select "Enter directly" and paste the content of the `id_rsa` file generated on the Master node.
- **Host Key Verification Strategy:** Select "**Non-verifying Verification Strategy**" (for lab environments) or "Known hosts file".

Conclusion

The successful completion of this assignment demonstrates the fundamental principles of **Distributed Computing** and **Continuous Integration** in a DevOps environment. Through this implementation, we can draw the following conclusions:

- **Scalability & Resource Optimization:** By offloading build and test executions to a Worker node, we ensure that the Jenkins Master remains responsive for administrative tasks and UI management. This prevents the "Single Point of Failure" regarding resource exhaustion.
- **Security through Private Networking:** Utilizing the **Worker's Private IP** for communication highlights a critical security best practice in Cloud Infrastructure. It ensures that internal CI/CD traffic remains within the VPC, reducing exposure to the public internet and eliminating unnecessary data transfer costs.
- **Authentication via SSH Keys:** The use of **RSA Key Pairs** for establishing the handshake between nodes provides a secure, password-less authentication method that is essential for automated systems, removing the need for manual human intervention during build triggers.
- **Architectural Understanding:** This setup provides a practical understanding of how independent cloud instances, which are "anonymous" to each other by default, can be unified into a single, cohesive automation cluster through proper networking and credential management.

Overall, this assignment provides a robust foundation for building complex, production-ready CI/CD pipelines where multiple worker nodes can be scaled horizontally to handle diverse operating systems and high-frequency deployment cycles.