	Pimpri Chinchwad Education Trust's **Pimpri Chinchwad College of Engineering (PCCoE)** (An Autonomous Institute) Affiliated to Savitribai Phule Pune University(SPPU) ISO 21001:2018 Certified by TUV	

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.

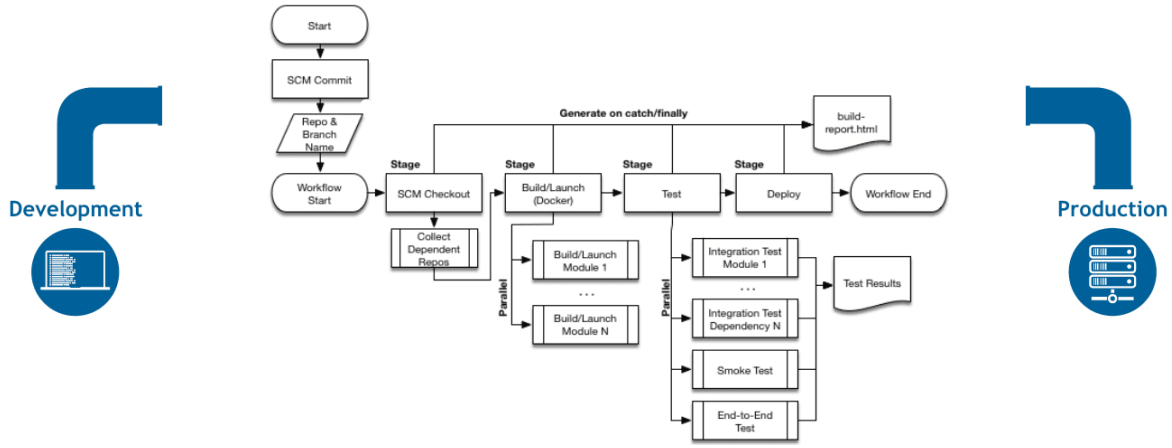
Prerequisites:

- 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

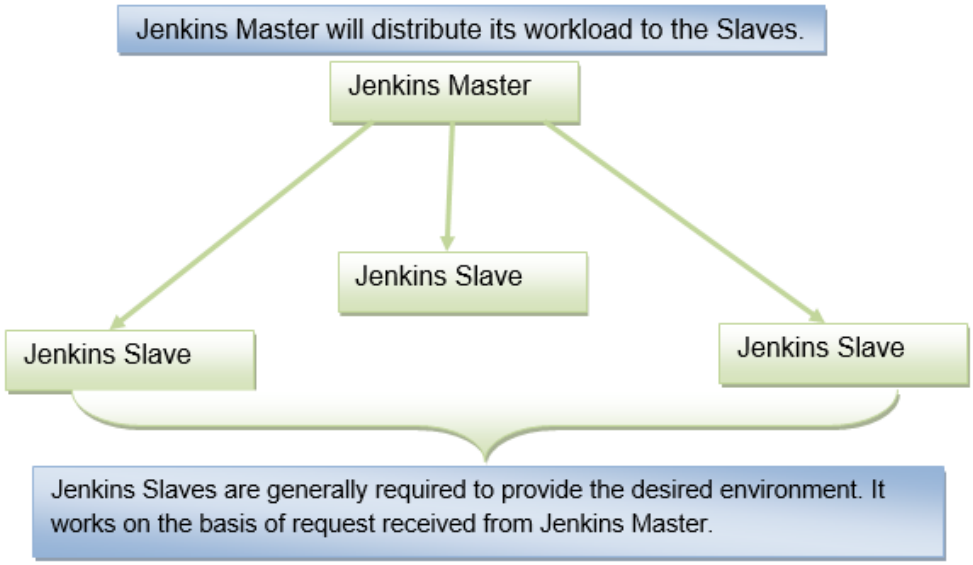
Jenkins is an open-source automation server that facilitates the technical aspects of Continuous Integration and Continuous Delivery. It automates the process of building, testing, and deploying software, allowing teams to focus on code quality.



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:

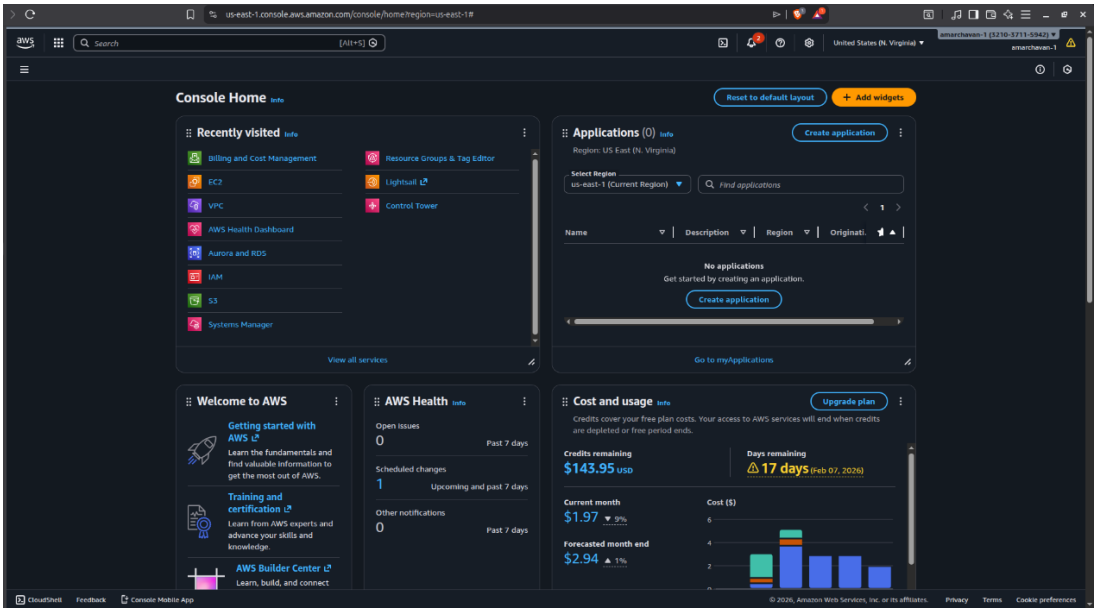
Step 1: AWS EC2 Instance Creation

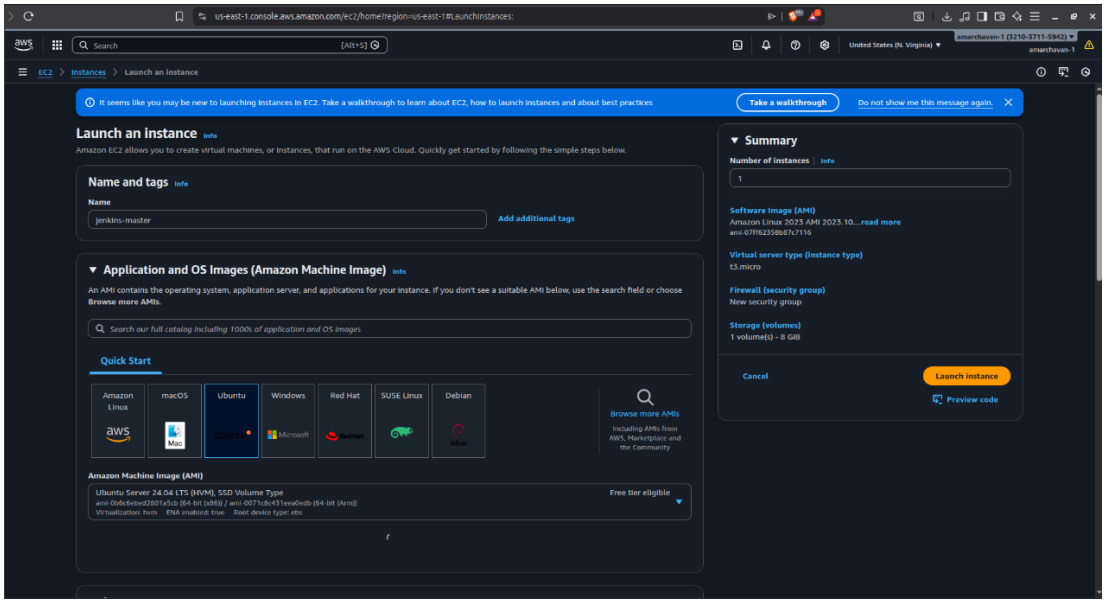
Before installing Jenkins, we must provision the virtual servers. Repeat these steps for both the **Master** and **Worker** nodes.

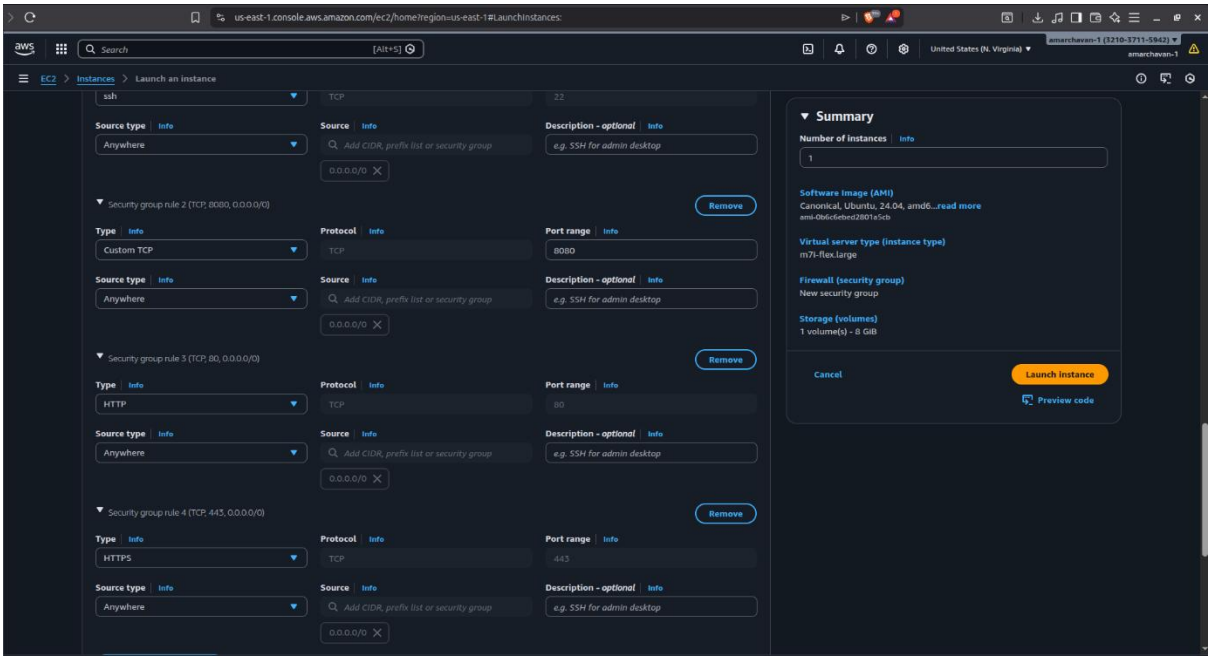
1. **Sign in to AWS Console:** Navigate to the EC2 Dashboard.
2. **Launch Instance:** Click on the "Launch Instance" button.
3. **Name and Tags:** Name the first instance: Jenkins-Master.
   Name the second instance: Jenkins-Worker.
4. **Application and OS Image:** Select **Ubuntu 24.04 LTS** (Free Tier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

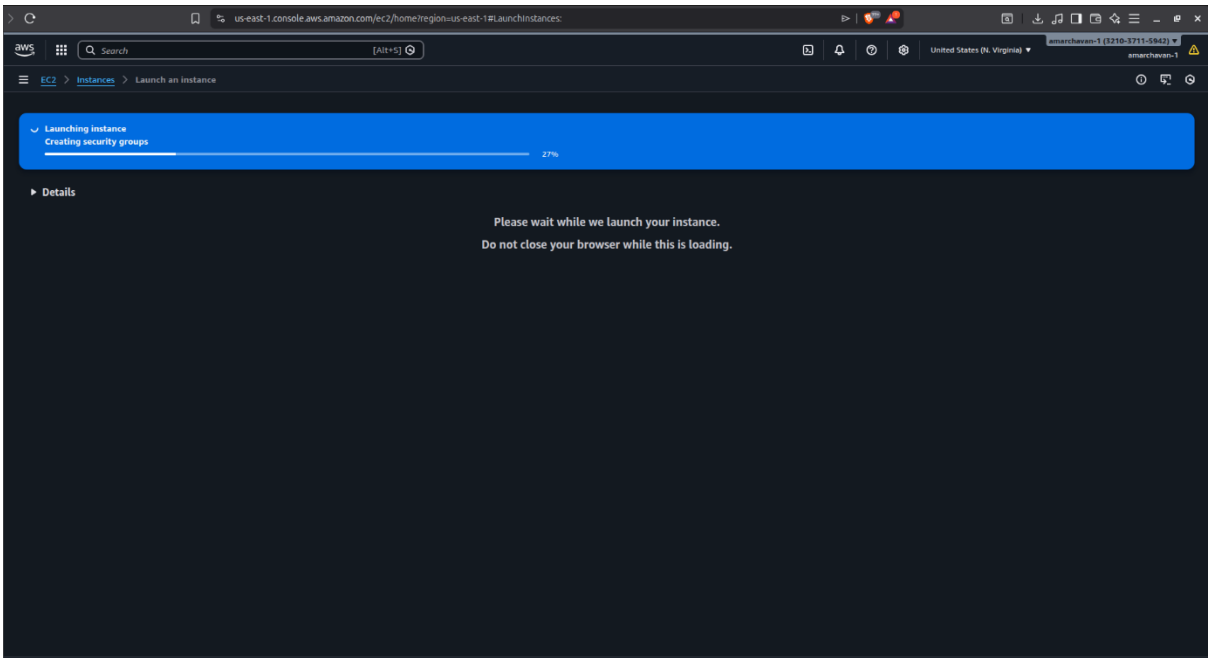
6. **Key Pair:** Select an existing key pair or create a new one (e.g., devops-lab.pem) to enable SSH access.
7. **Network Settings (Security Groups):** * Create a Security Group named Jenkins-SG.
    - **Inbound Rules for Master:**
        - Type: **SSH** | Port: **22** | Source: **My IP**
        - Type: **Custom TCP** | Port: **8080** | Source: **Anywhere (0.0.0.0/0)**
    - **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

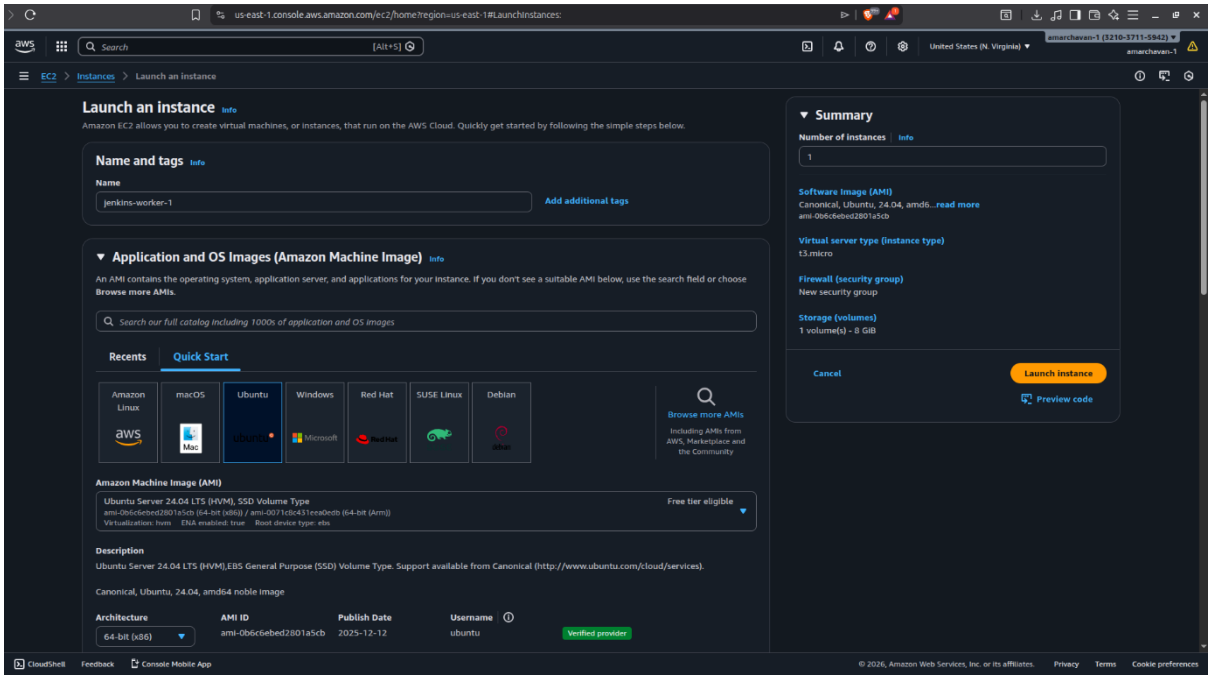


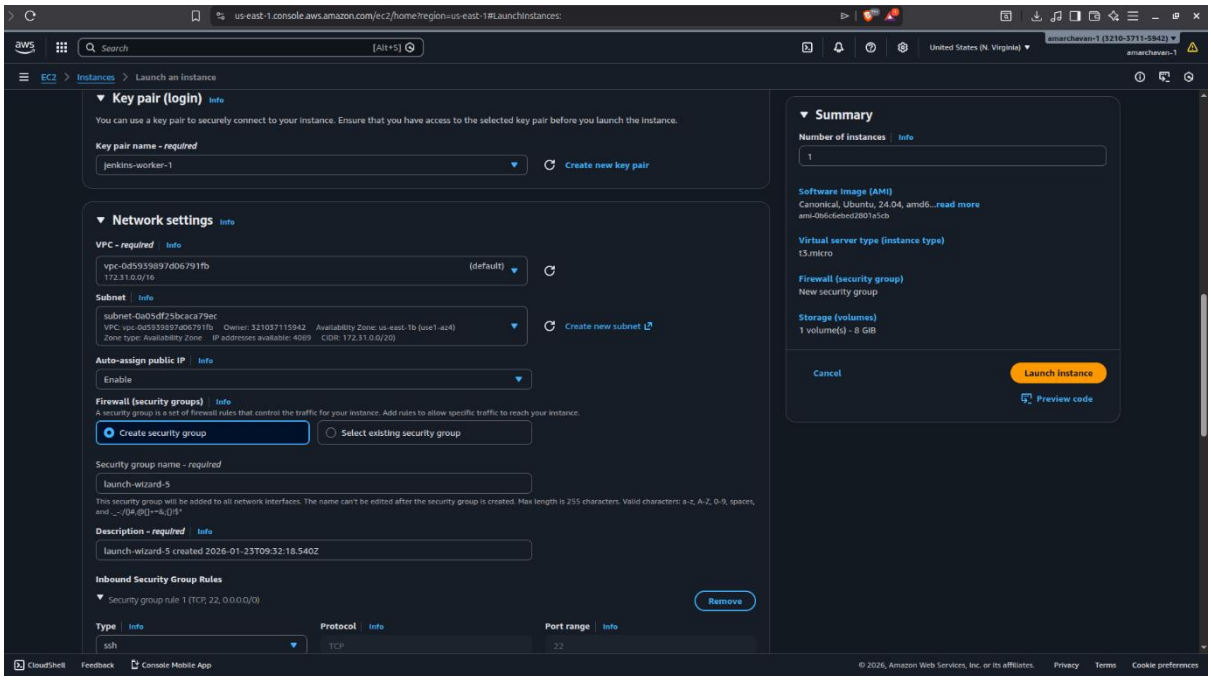


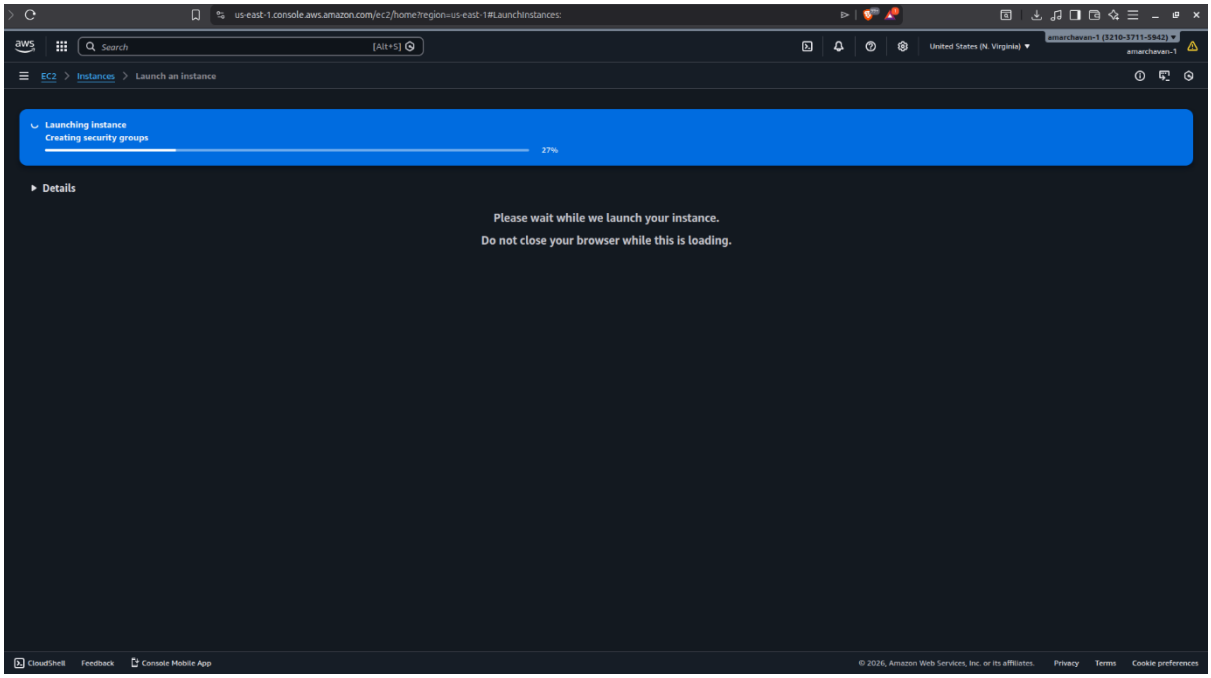

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
Search
[Alt+S]
United States (N. Virginia)
amarchavan-1 (3210-3711-5942)
amarchavan-1
EC2 > Instances > Launch an instance
ssh
TCP
22
Source type | Info
Anywhere
Source | Info
Add CIDR, prefix list or security group
Description - optional | Info
e.g. SSH for admin desktop
0.0.0.0/0
Security group rule 2 (TCP, 8080, 0.0.0.0/0)
Remove
Type | Info
Custom TCP
Protocol | Info
TCP
Port range | Info
8080
Source type | Info
Anywhere
Source | Info
Add CIDR, prefix list or security group
Description - optional | Info
e.g. SSH for admin desktop
0.0.0.0/0
Security group rule 3 (TCP, 80, 0.0.0.0/0)
Remove
Type | Info
HTTP
Protocol | Info
TCP
Port range | Info
80
Source type | Info
Anywhere
Source | Info
Add CIDR, prefix list or security group
Description - optional | Info
e.g. SSH for admin desktop
0.0.0.0/0
Security group rule 4 (TCP, 443, 0.0.0.0/0)
Remove
Type | Info
HTTPS
Protocol | Info
TCP
Port range | Info
443
Source type | Info
Anywhere
Source | Info
Add CIDR, prefix list or security group
Description - optional | Info
e.g. SSH for admin desktop
0.0.0.0/0
Summary
Number of instances | Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0b6c6ebed2801a5cb
Virtual server type (instance type)
m7i-flex.large
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Cancel
Launch instance
Preview code


us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
Search
[Alt+S]
United States (N. Virginia)
amarchavan-1 (3210-3711-5942)
amarchavan-1
EC2 > Instances > Launch an instance
Launching instance
Creating security groups
27%
Details
Please wait while we launch your instance.
Do not close your browser while this is loading.

Jenkins-Worker (Slave) Creation EC2 Screenshots







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.

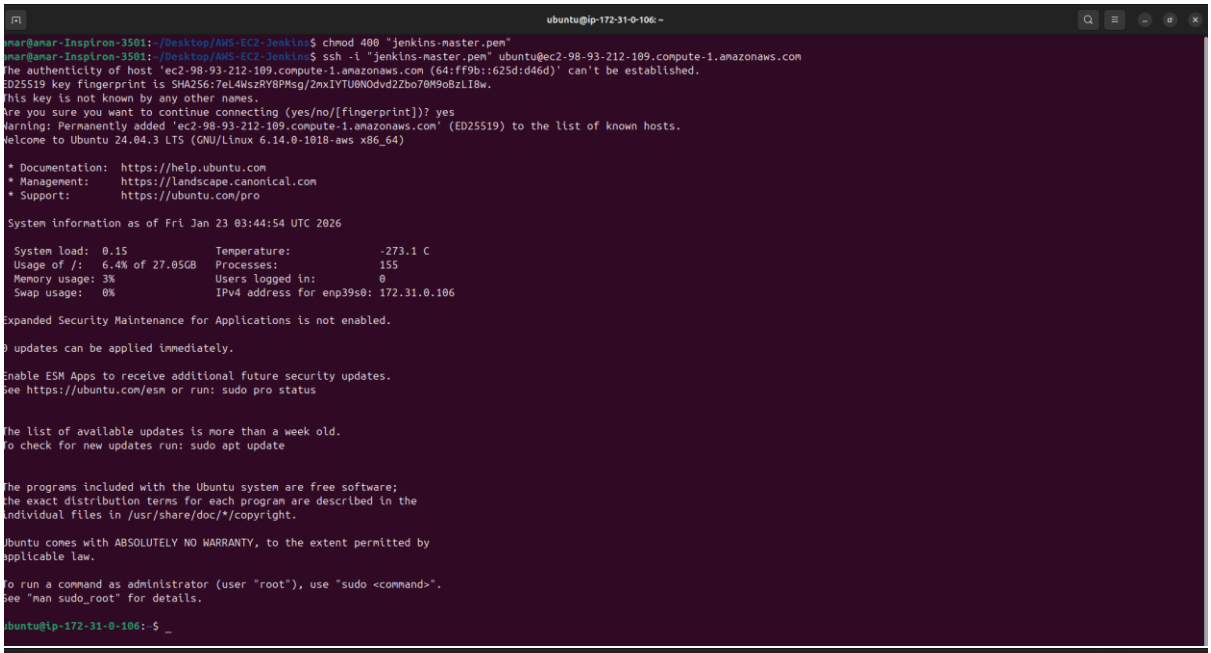
   sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
   https://pkg.jenkins.io/debian-stable/jenkins.io-2026.key
   echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \
   https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
   /etc/apt/sources.list.d/jenkins.list > /dev/null

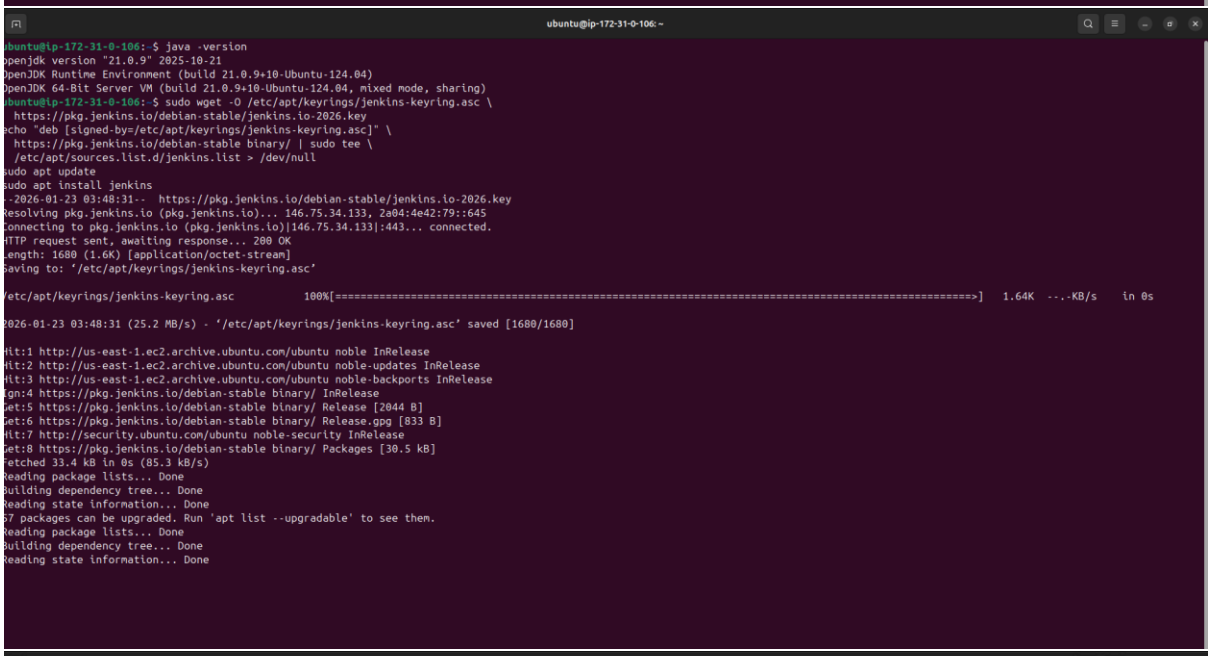
   sudo apt update

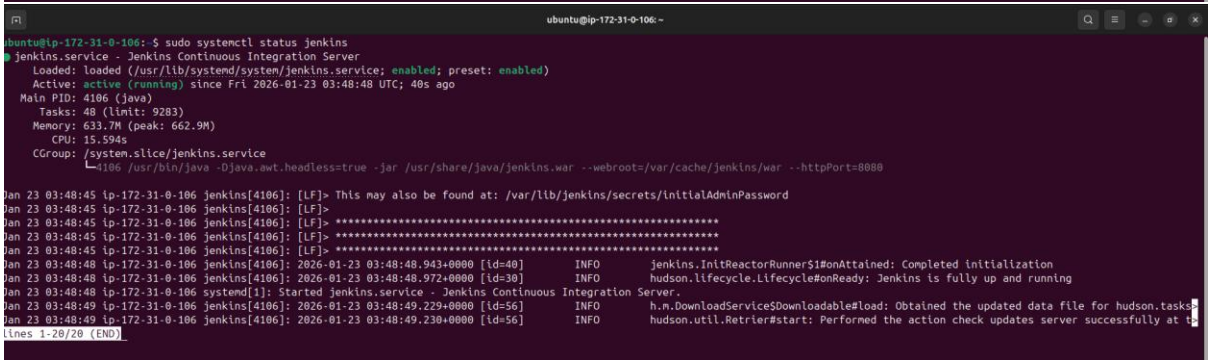
   sudo apt install Jenkins

3. **Check Status:**
   sudo systemctl status Jenkins

Screenshots of Installation on the Mater Node







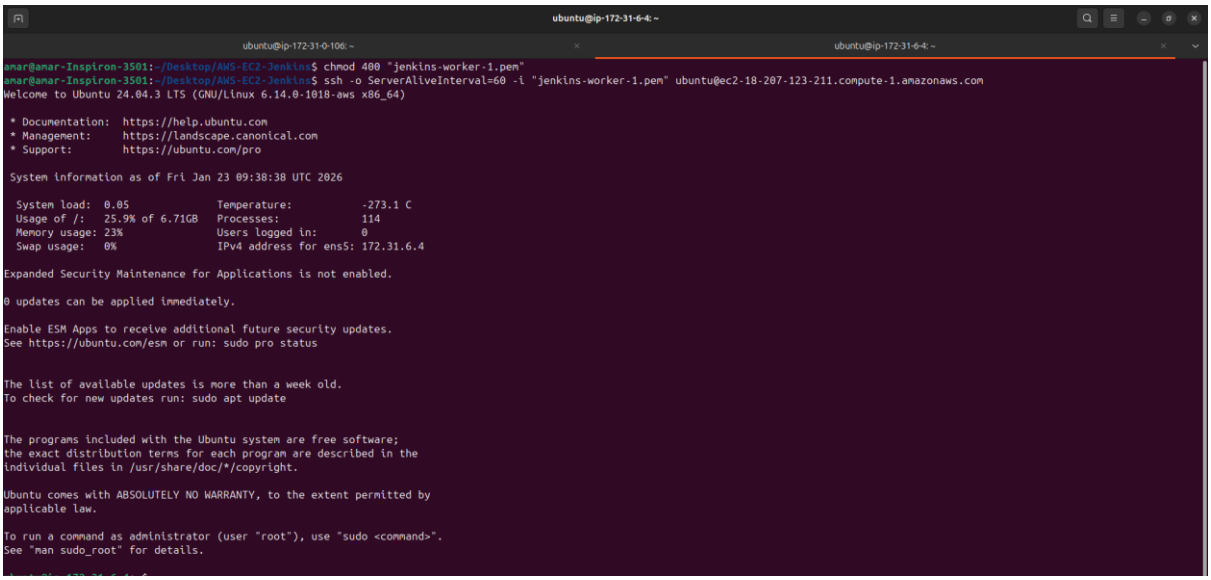
Step 3: Configuration on Worker Node

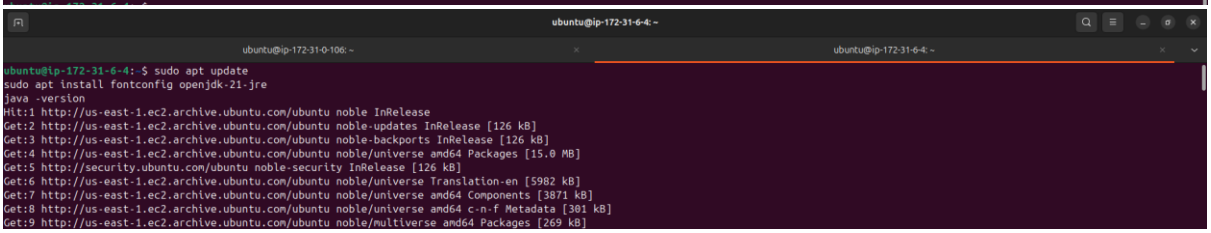
Connect to the **Worker** via SSH and only install the Java runtime:

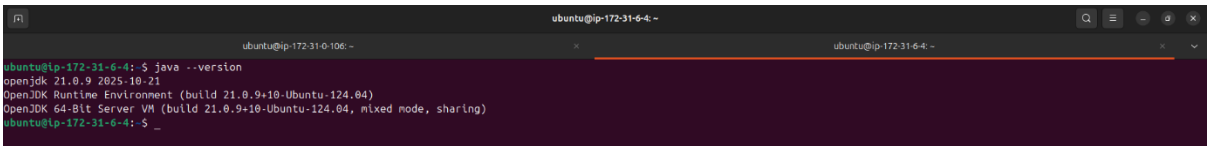
Install Java:

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

Screenshots of Configuration on Worker Node



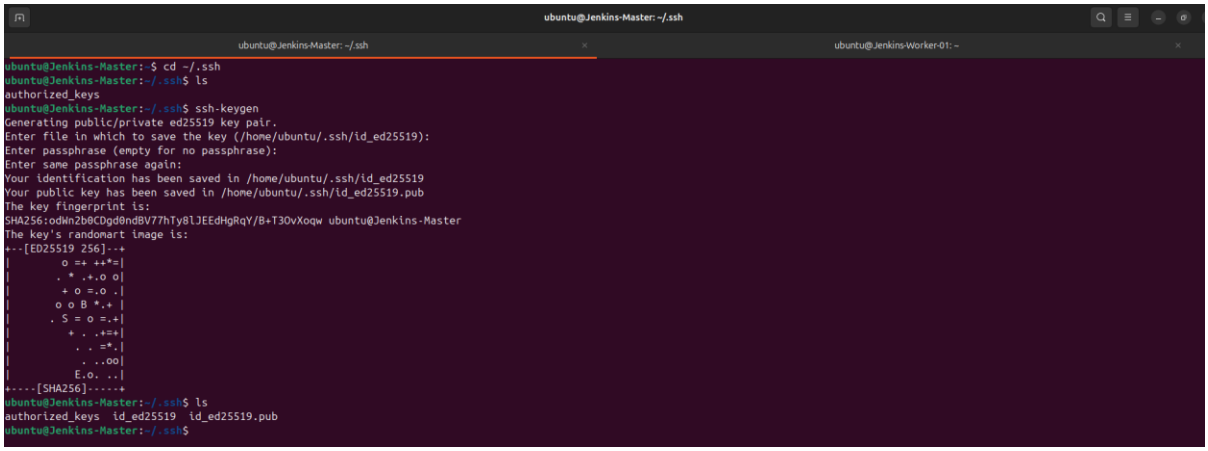




Step 4: Establishing Master-Worker Connectivity (SSH Handshake)

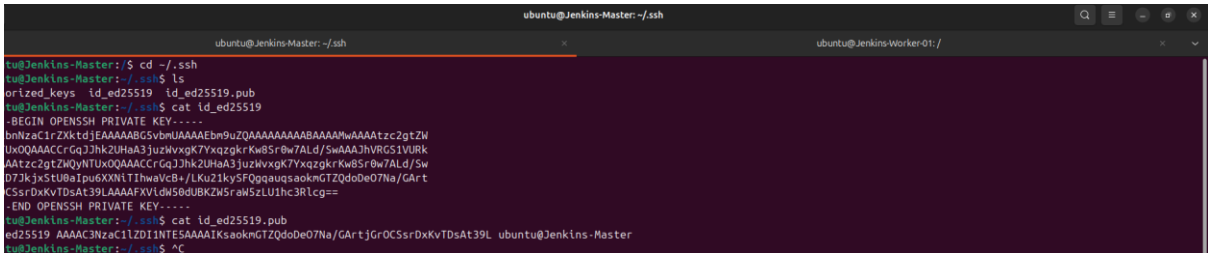
This step connects the " Master Node" to the " Worker Node."

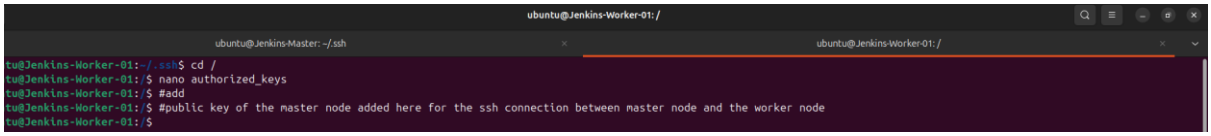
1. **Generate SSH Keys on Master:**
   Shh -keygen



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

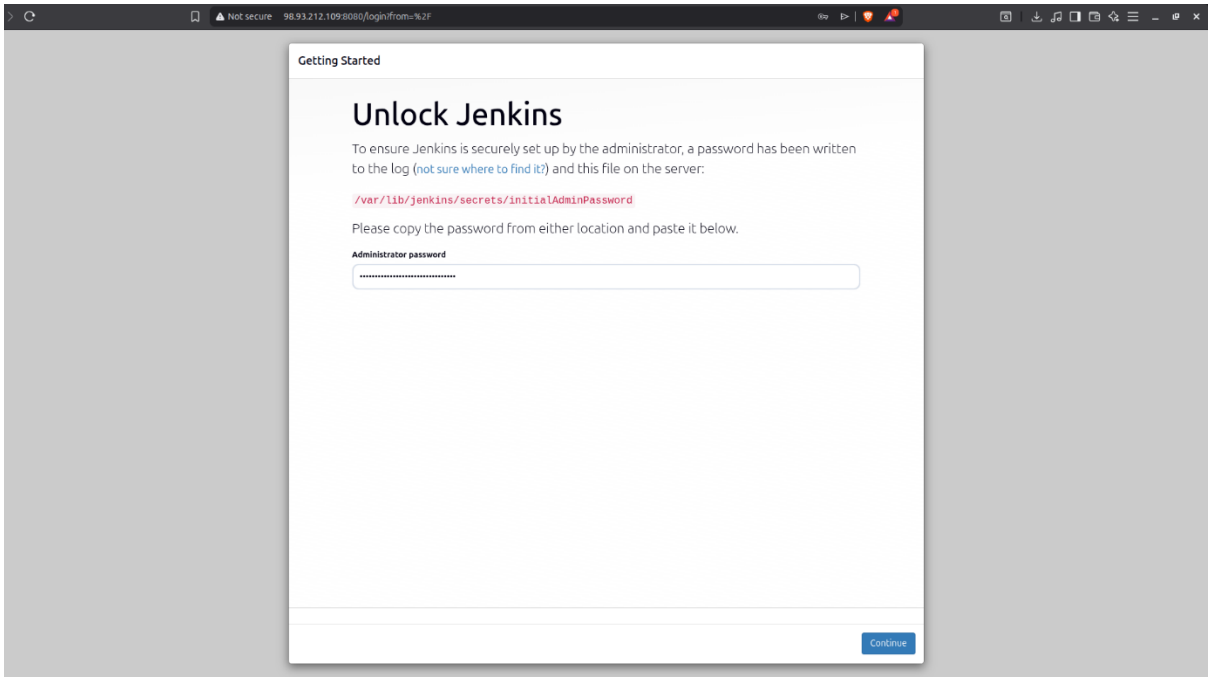


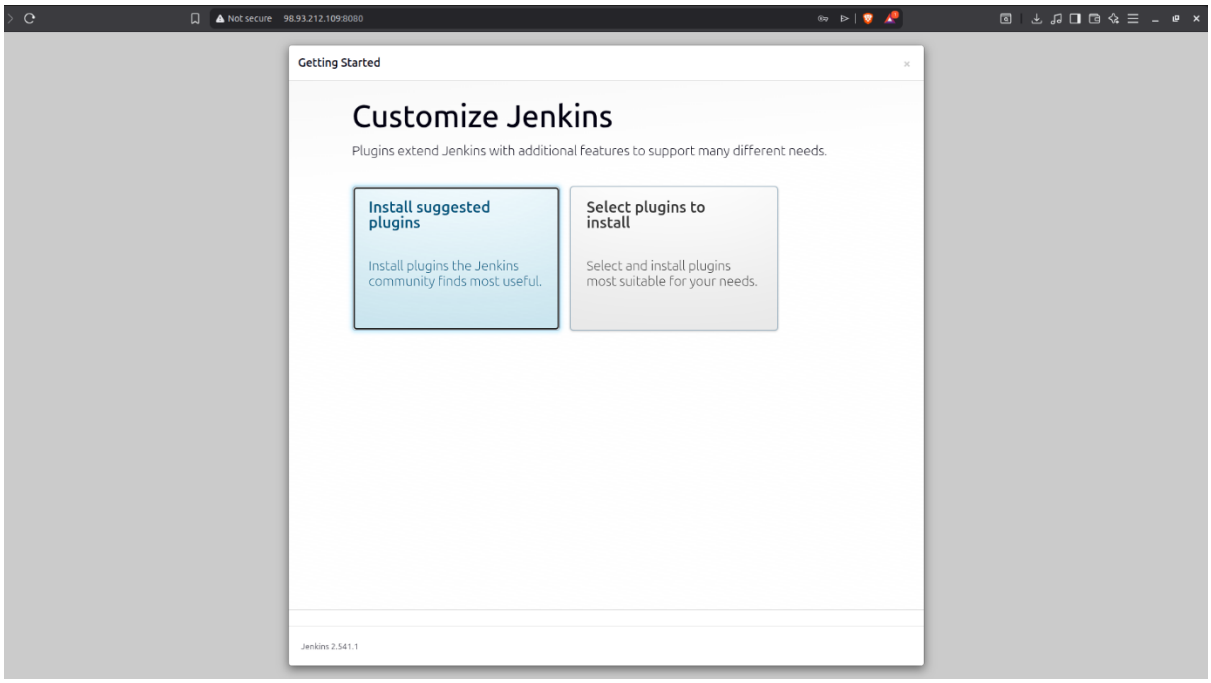


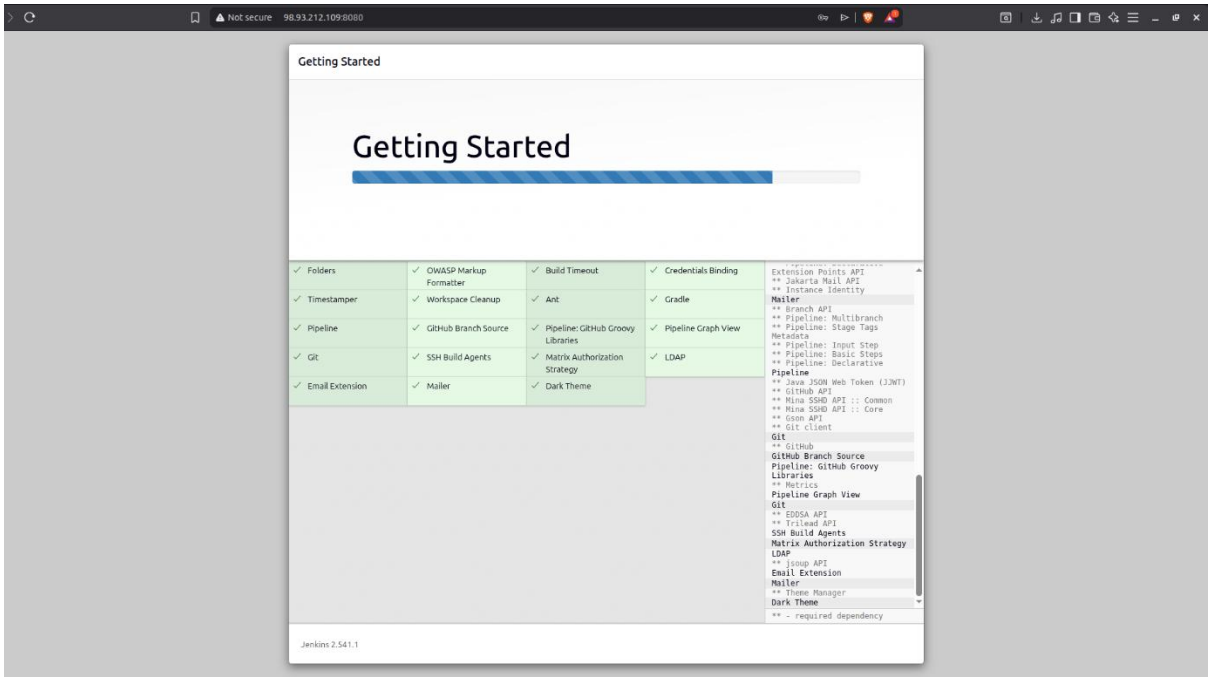
Step 5: Jenkins UI Setup & Node Registration

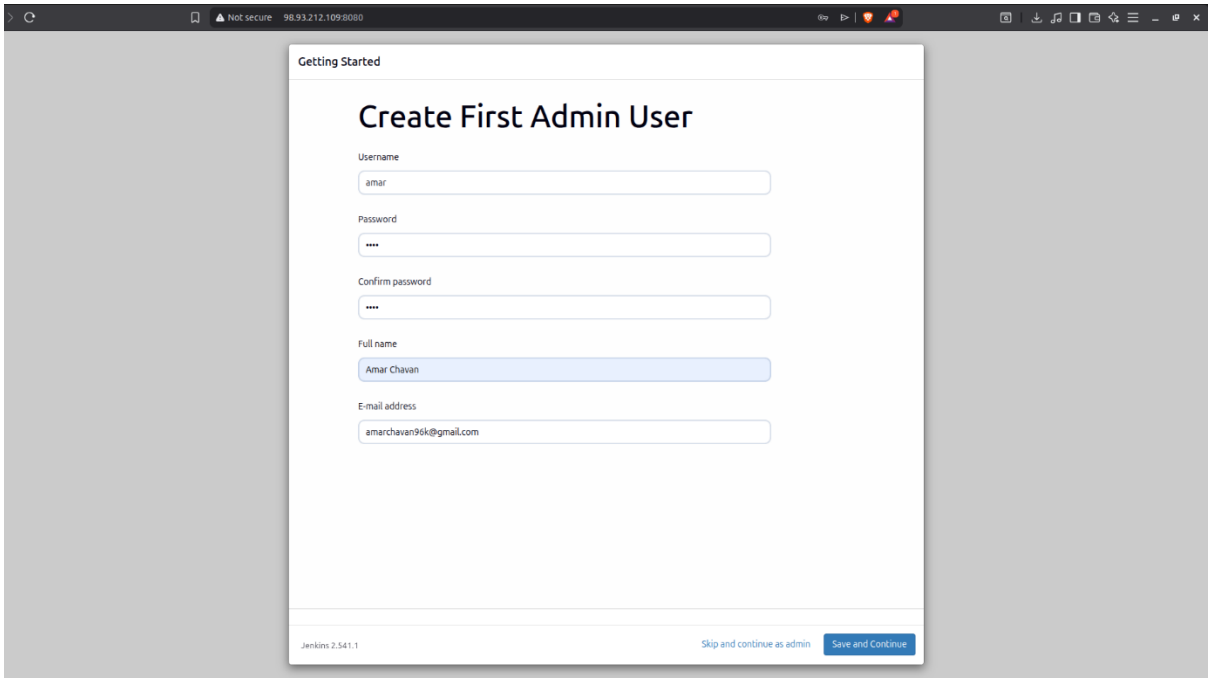
1. **Access Jenkins:** Open your browser and navigate to http://<Master-Public-IP>:8080.
2. **Unlock Jenkins:** Retrieve the initial admin password from the Master terminal: sudo cat /var/lib/jenkins/secrets/initialAdminPassword
3. **Install Plugins:** Select "Install Suggested Plugins."

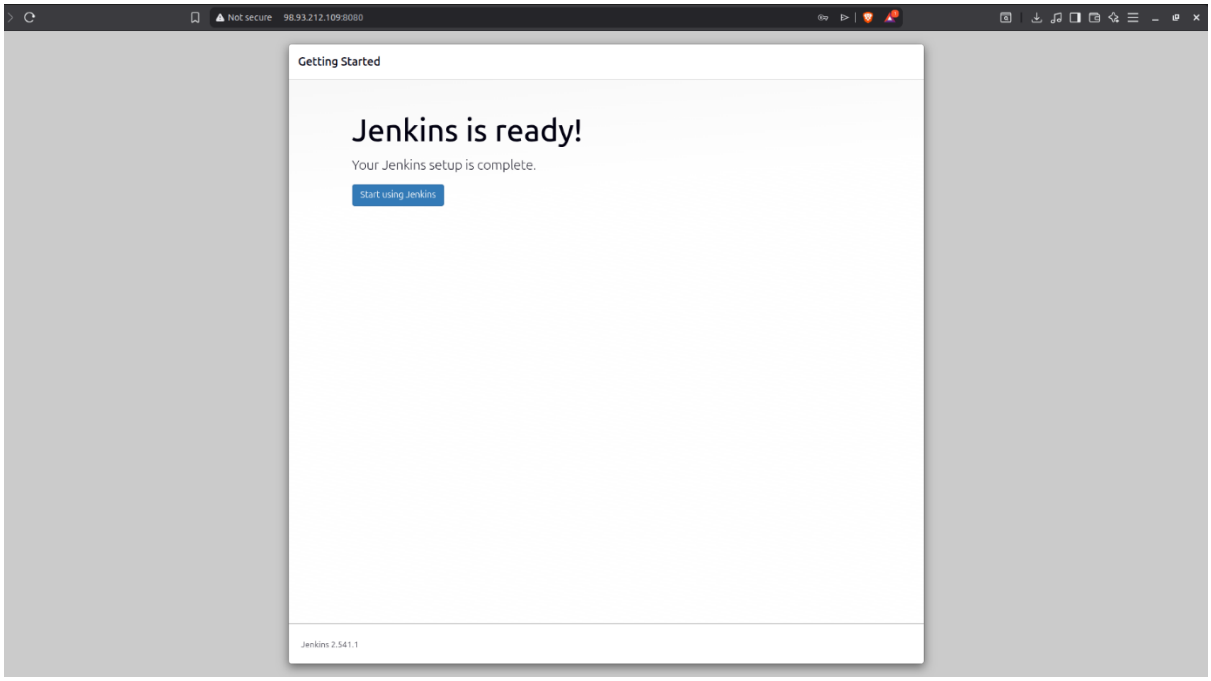
Screenshots of Jenkins UI Setup

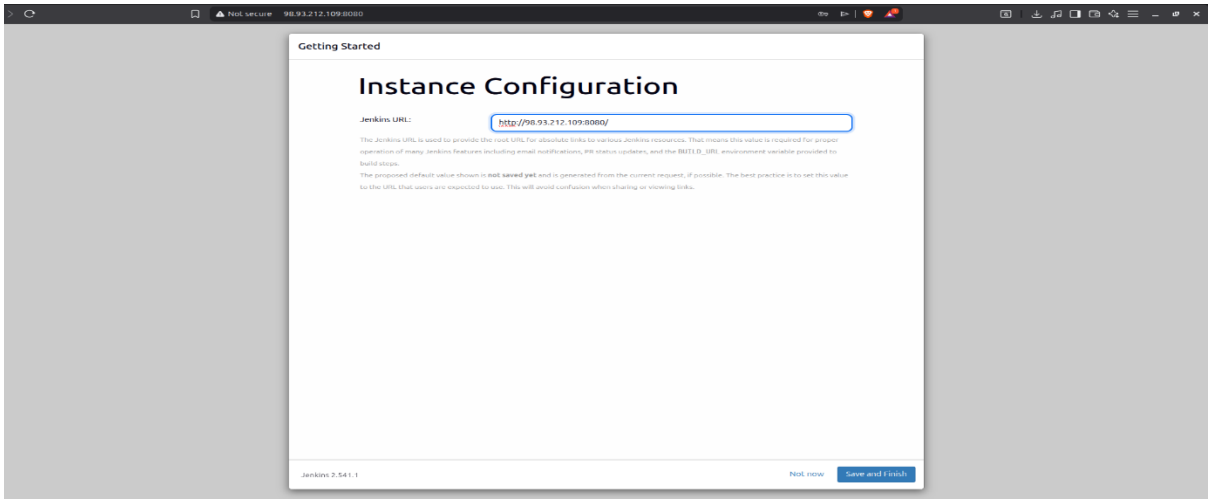


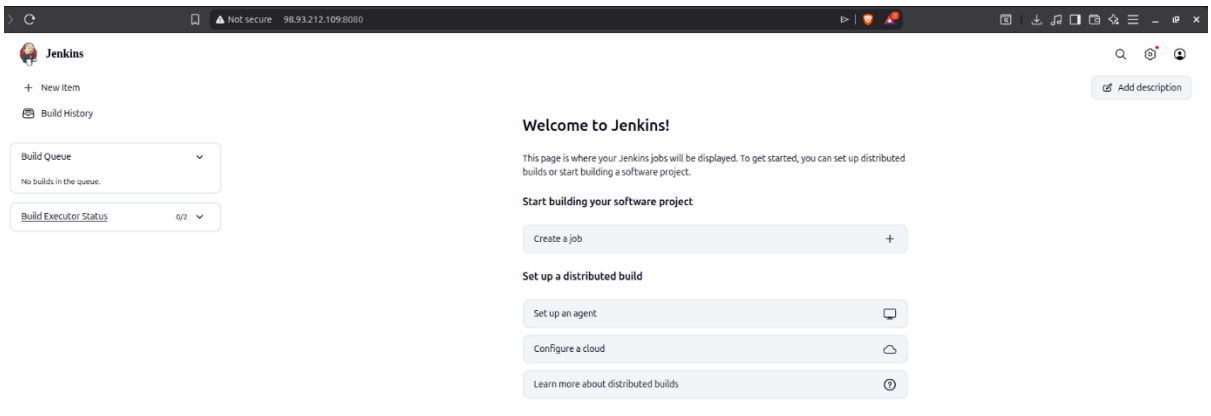



Not secure 98.93.212.109:8080
Getting Started
Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline Graph View
Git
SSH Build Agents
Matrix Authorization Strategy
LDAP
Email Extension
Mailer
Dark Theme
Jenkins 2.541.1


Not secure 98.93.212.109:8080
Getting Started
Create First Admin User
Username
amar
Password
Confirm password
Full name
Amar Chavan
E-mail address
amarchavan96k@gmail.com
Jenkins 2.541.1
Skip and continue as admin
Save and Continue

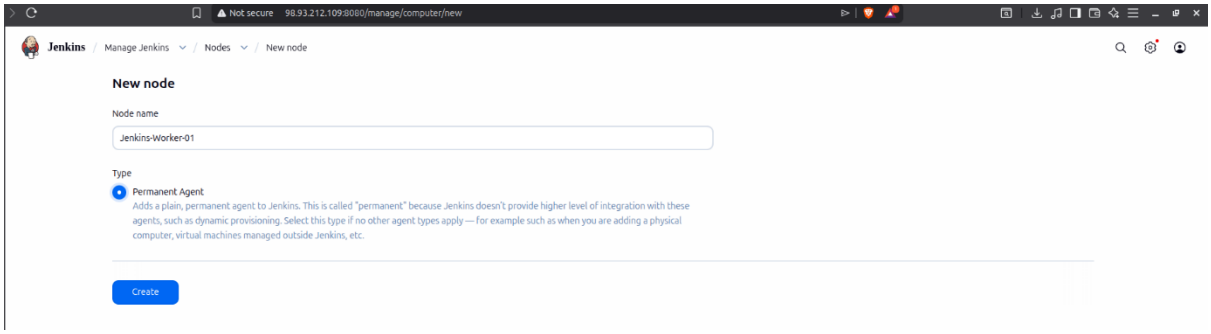

Not secure 98.93.212.109:8080
Getting Started
Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins
Jenkins 2.541.1

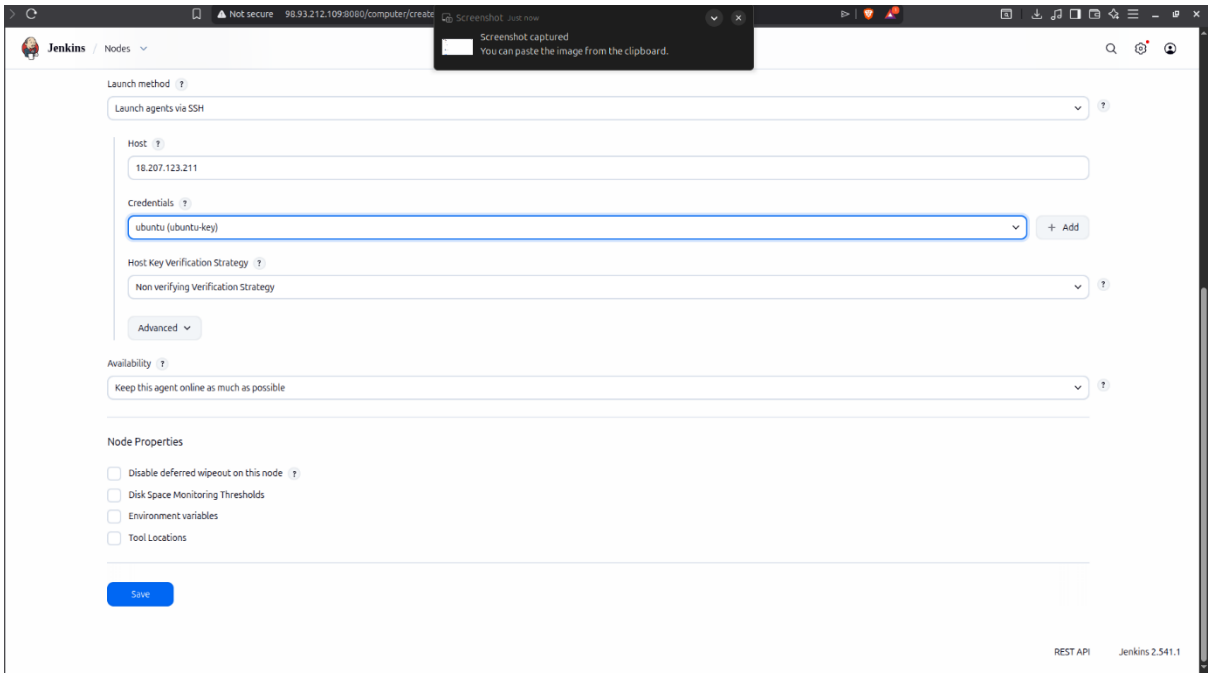

Not secure 98.93.212.109:8080
Getting Started
Instance Configuration
Jenkins URL:
http://98.93.212.109:8080/
Jenkins 2.541.1
Not now
Save and Finish

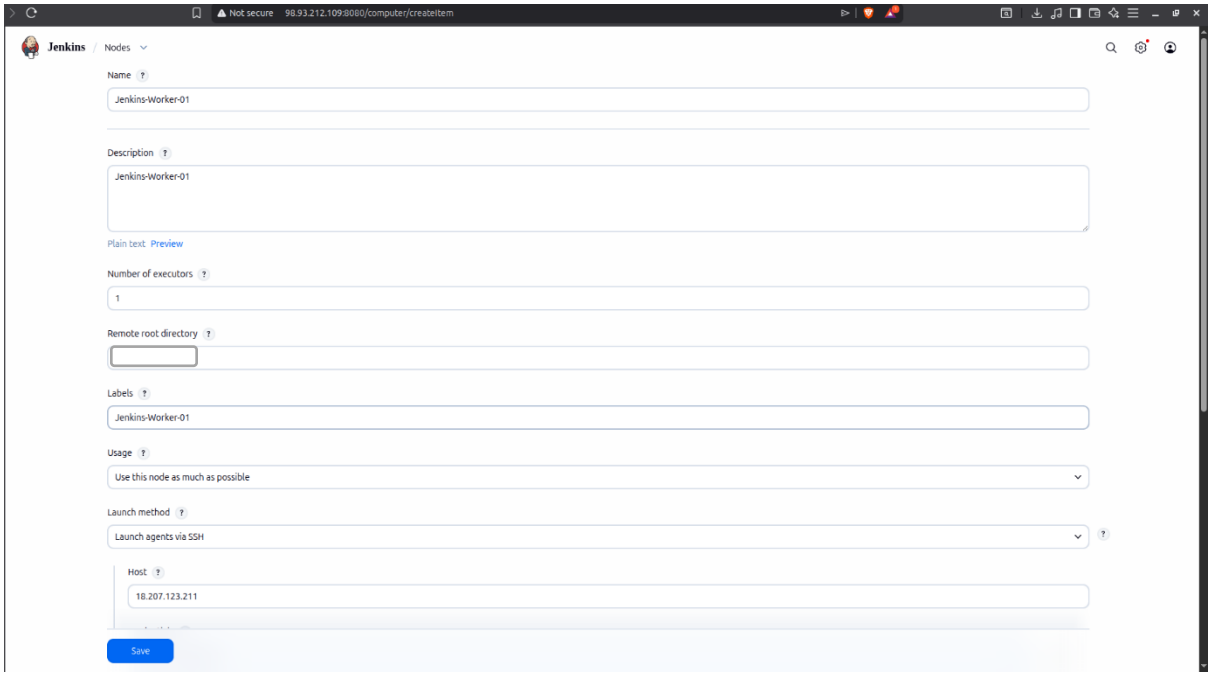

Not secure 98.93.212.109:8080
Jenkins
New Item
Build History
Build Queue
No builds in the queue.
Build Executor Status
0/2
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds

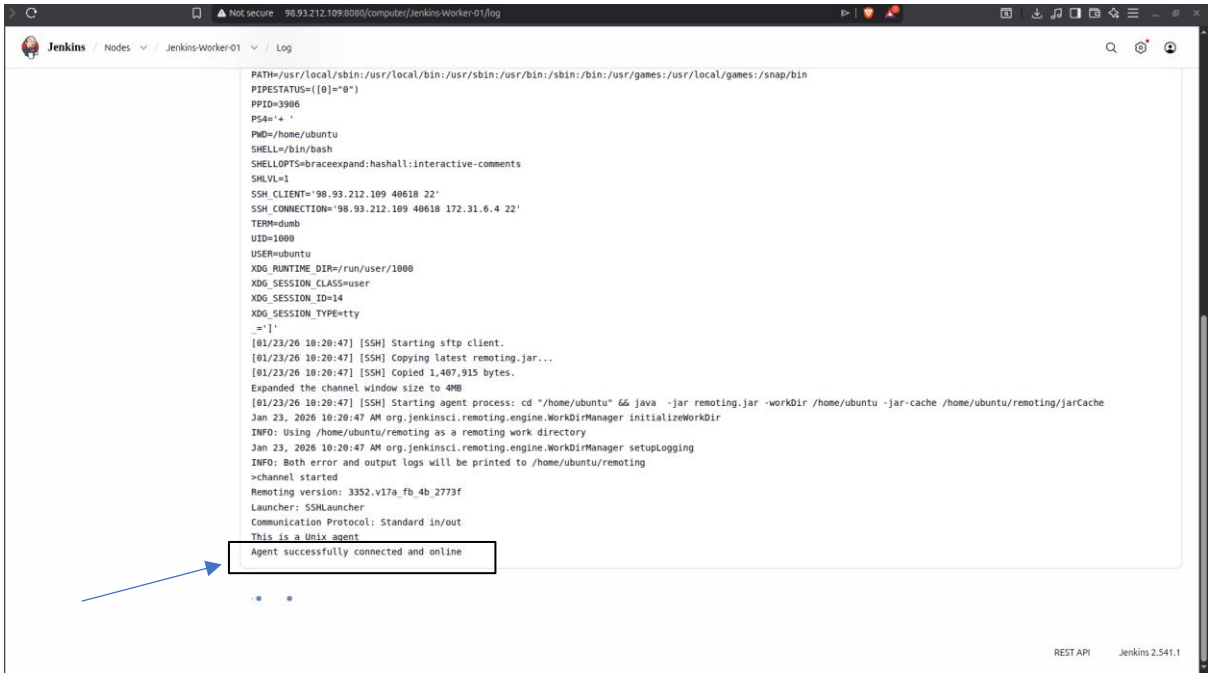
Navigate to **Manage Jenkins > Nodes > New Node**.

- **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









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.