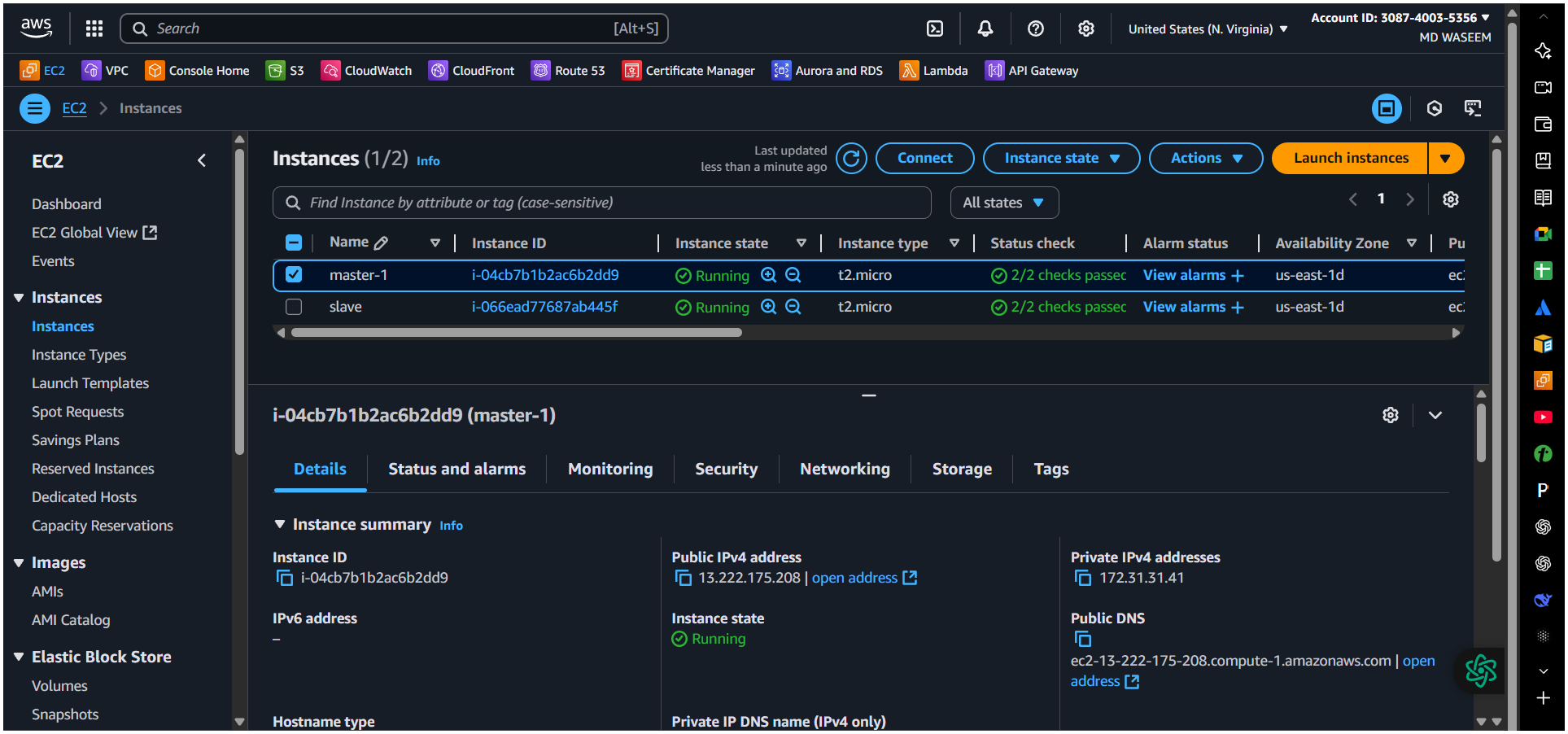
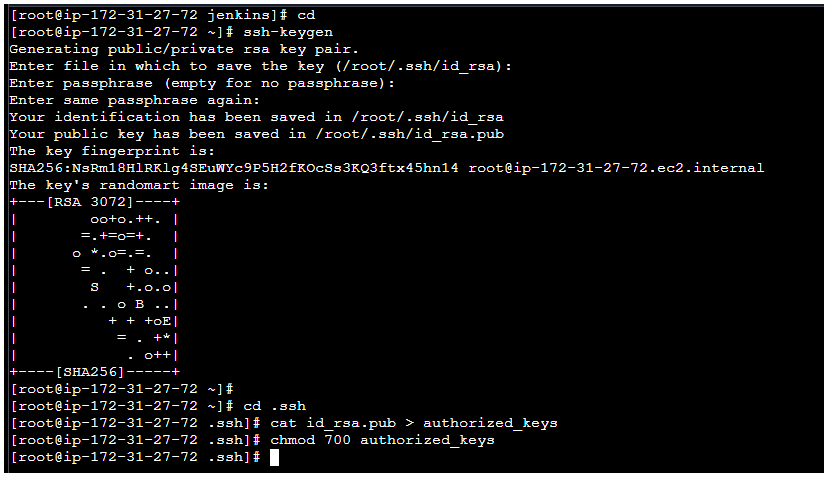
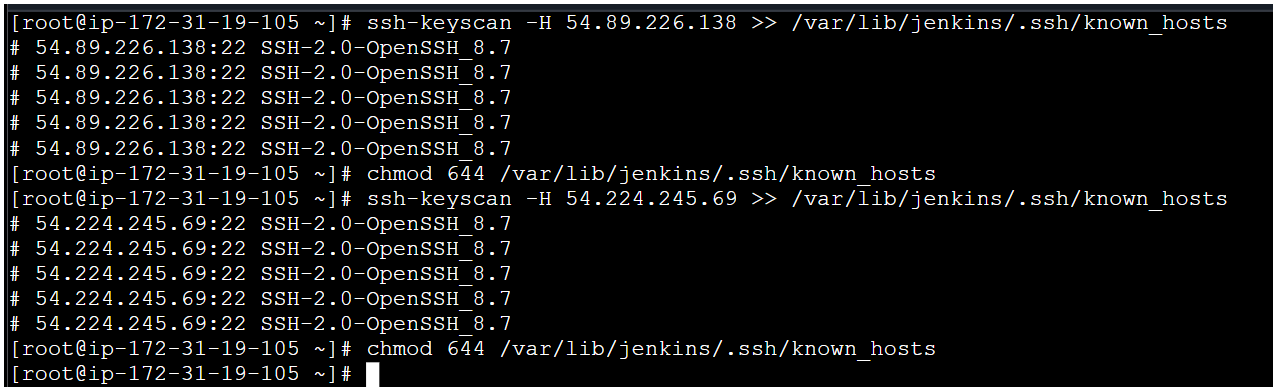
1. Configure 2 slave machines in Jenkins master.



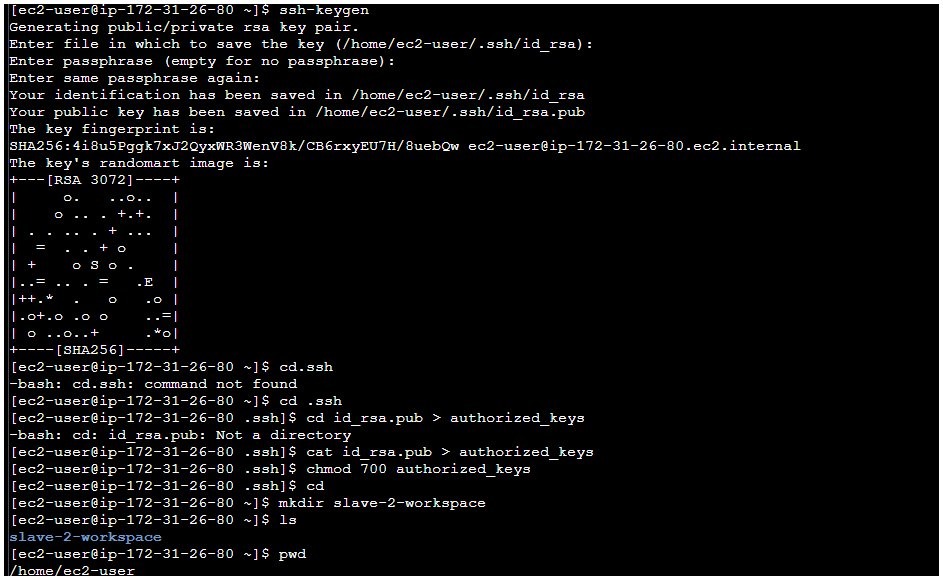
🡪slave-1



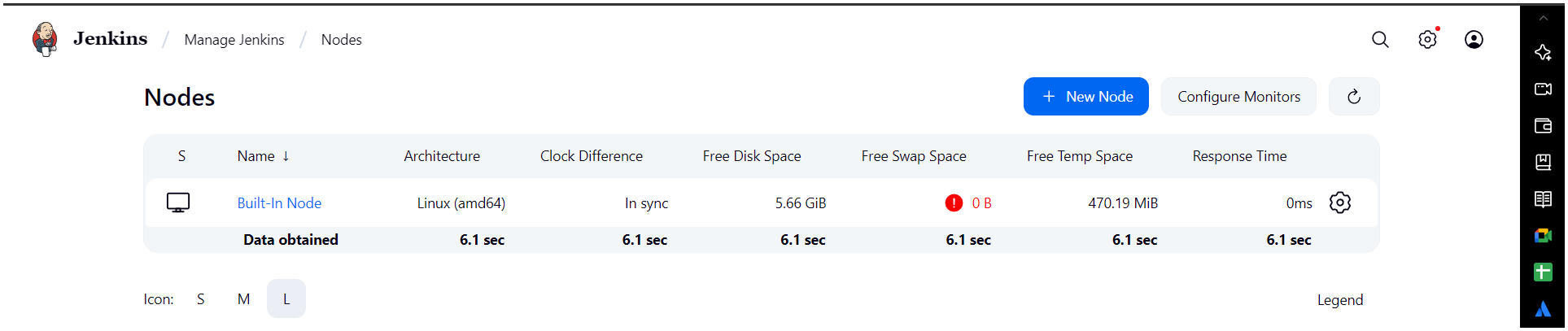
🡪master

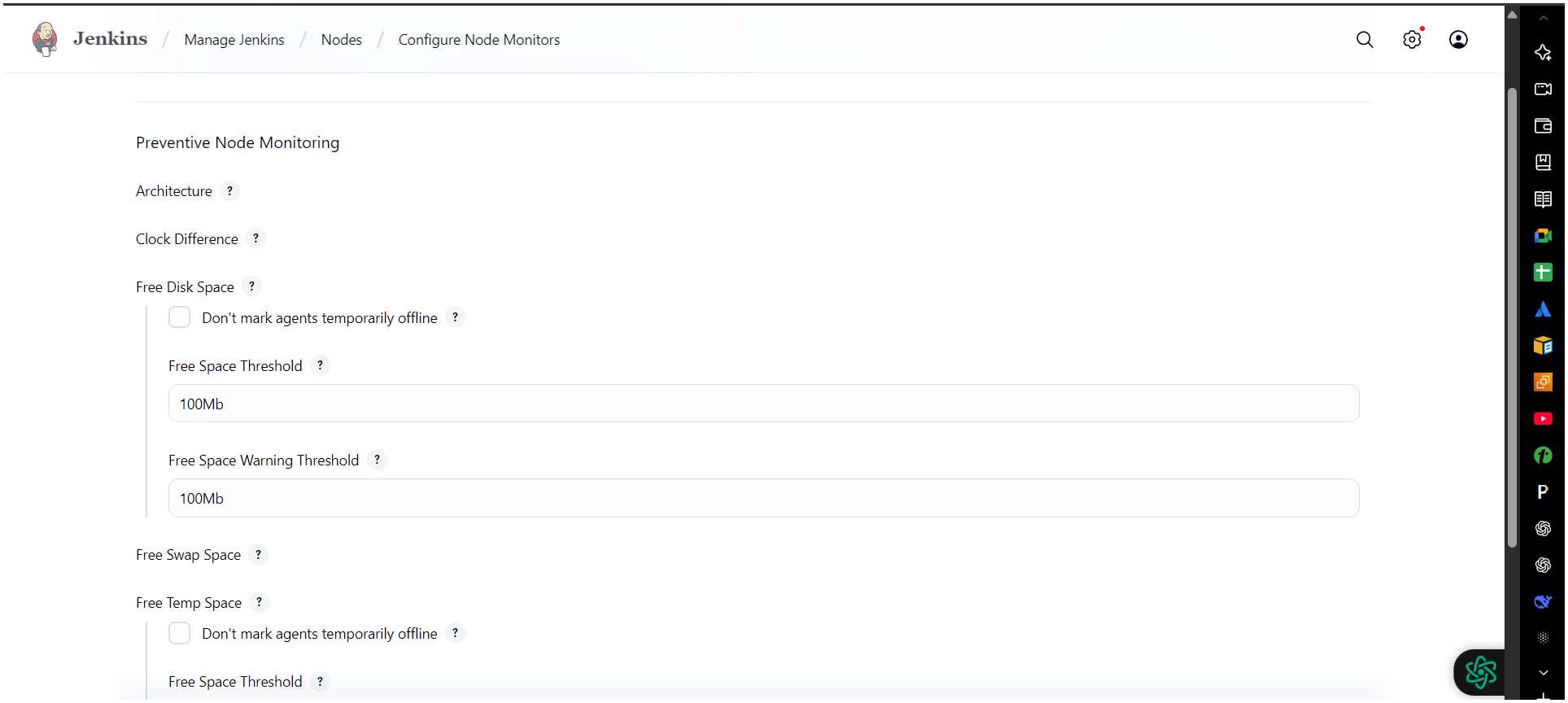


🡪slave-2

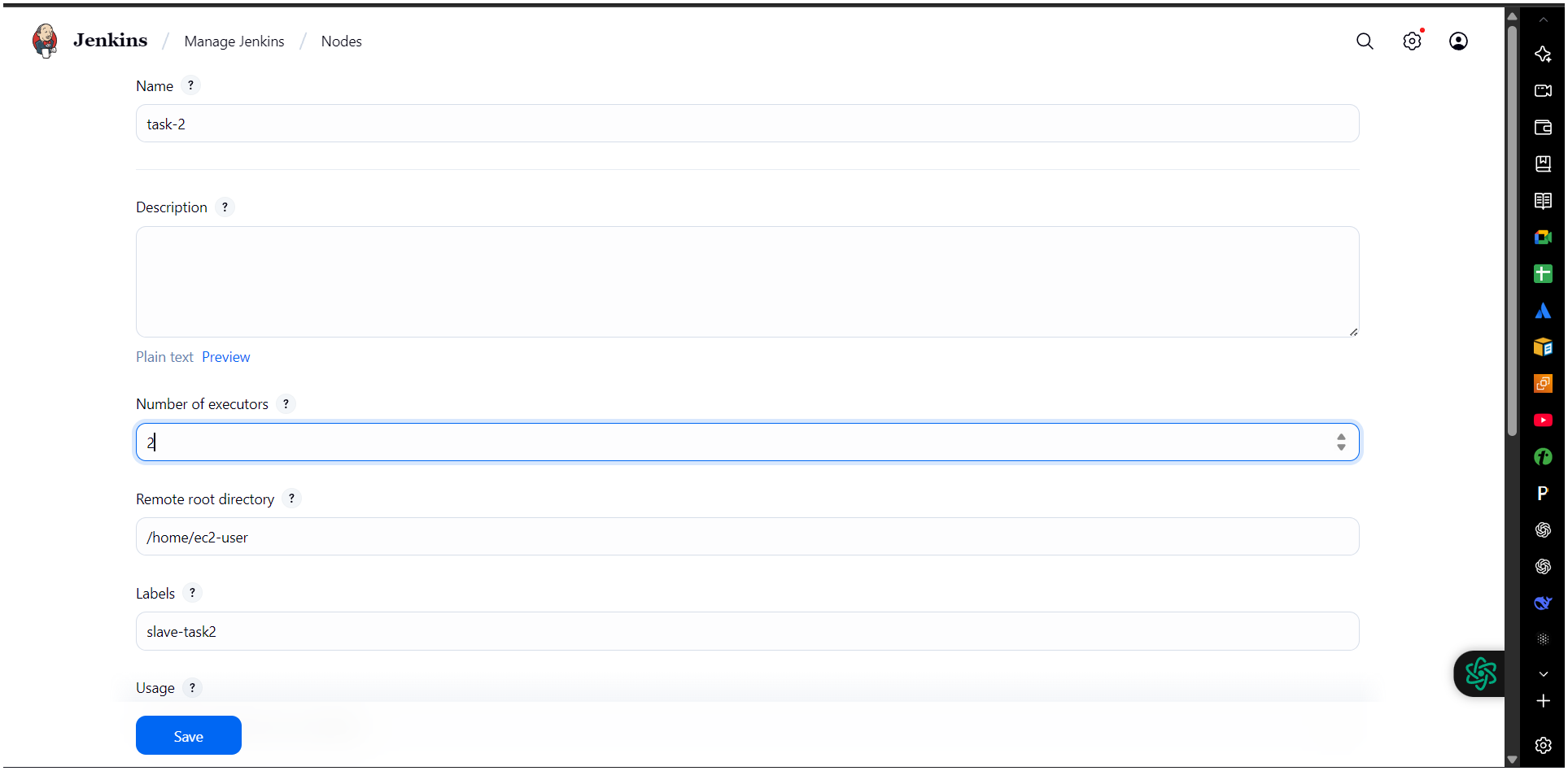


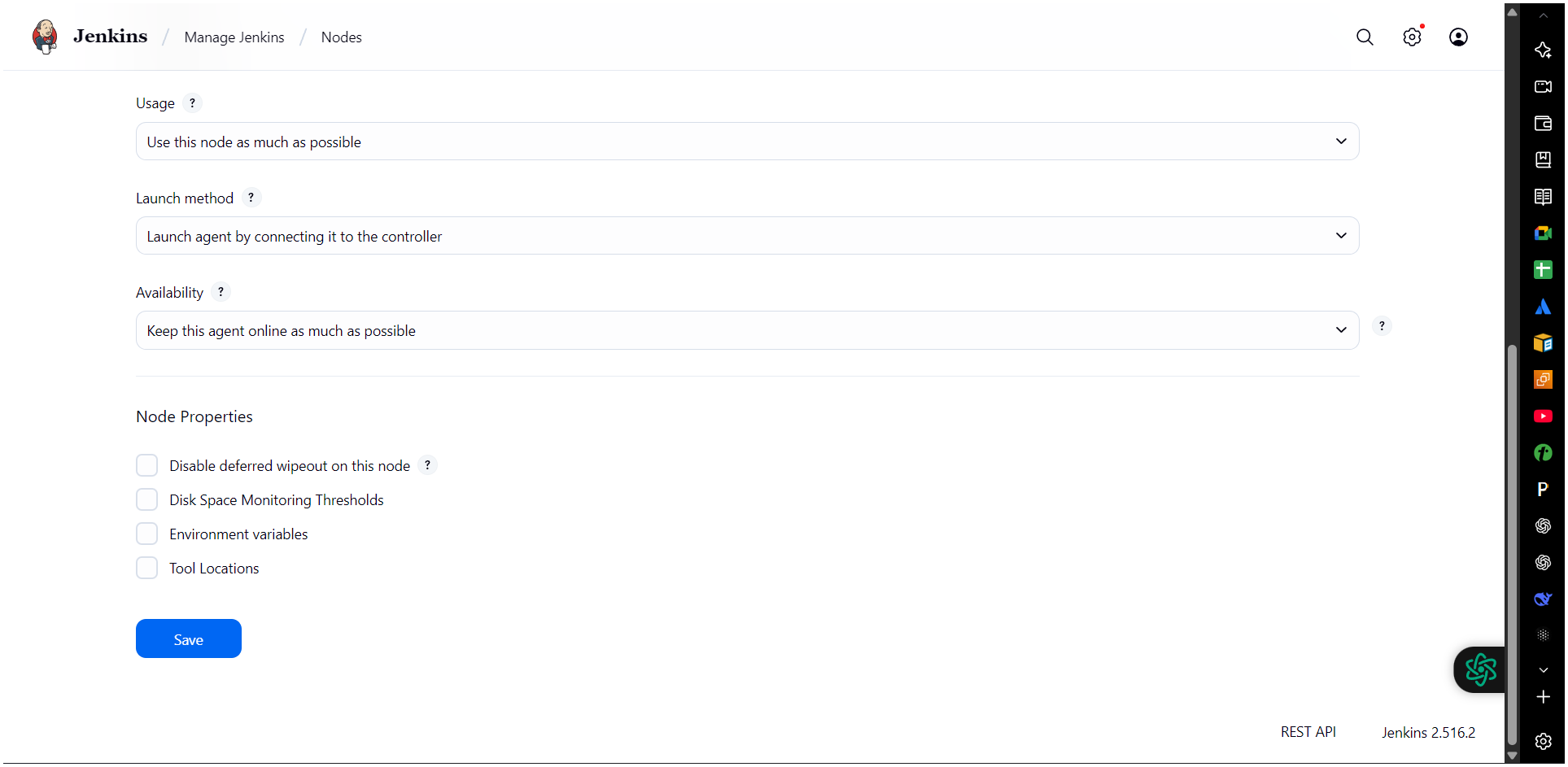
🡪In master Jenkin’s Build In-Node error 🡪configure Monitors🡪Replace 1gb to 100mb(Free Disk Space ) 🡪Apply 🡪Save.



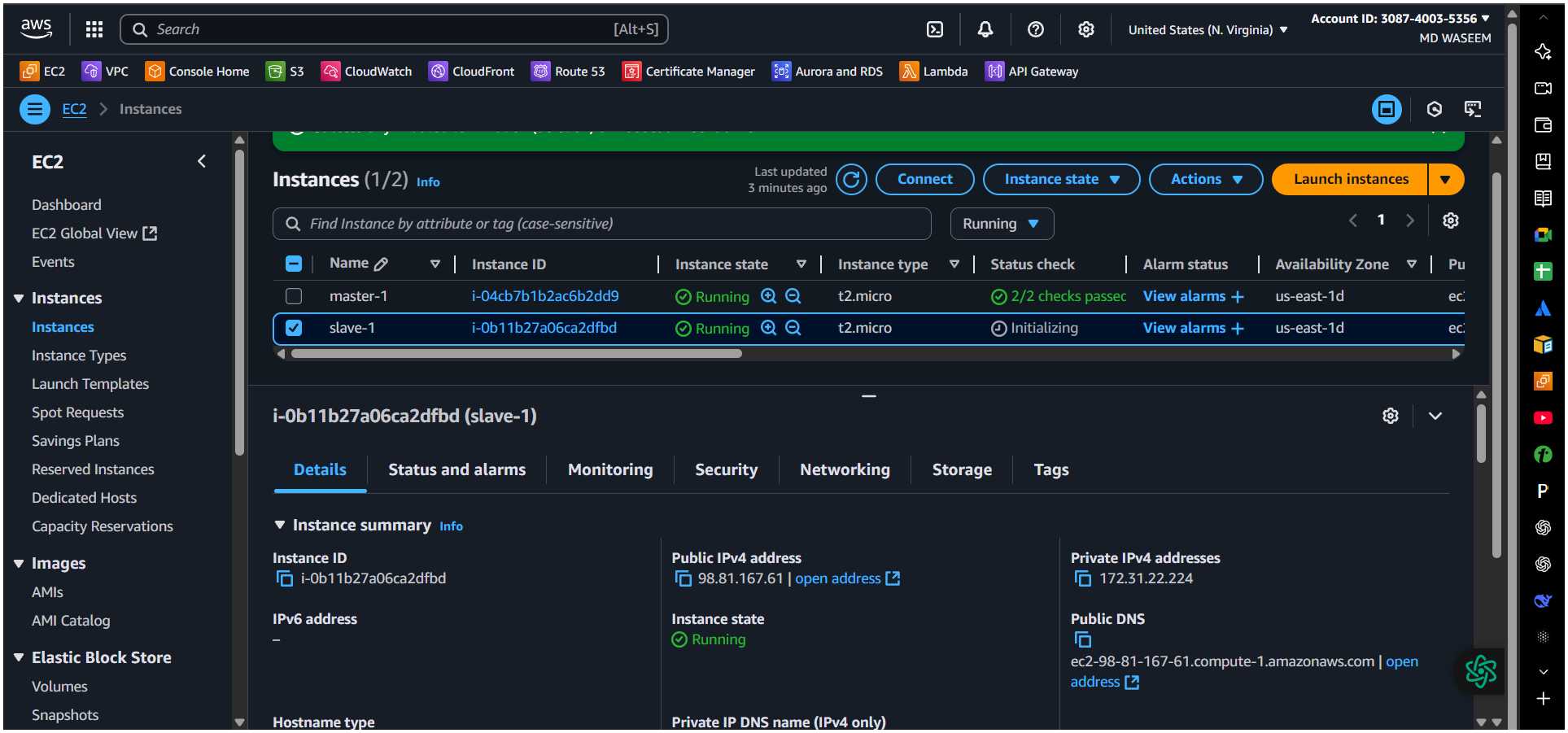


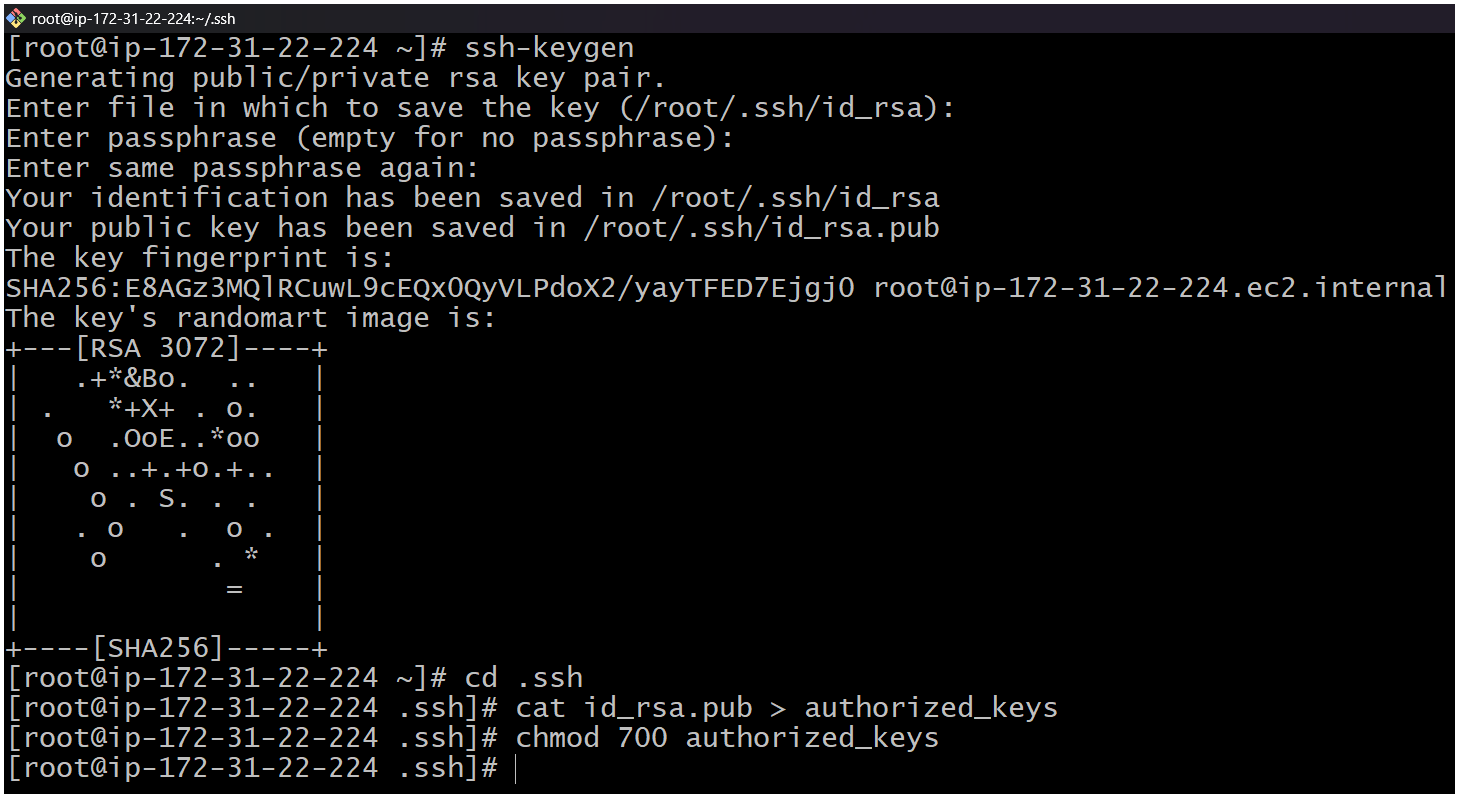






* Steps on Master Machine  
  1. Login to master machine.  
  2.Switch to root user.  
  3. Create Jenkins SSH directory:  
  mkdir -p /var/lib/jenkins/.ssh  
  cd /var/lib/jenkins/.ssh  
  . ssh-keyscan -H SLAVE-NODE-PUBLIC-IP >> /var/lib/jenkins/.ssh/known\_hosts  
  . chown jenkins:jenkins /var/lib/jenkins/.ssh/known\_hosts  
  . chmod 644 /var/lib/jenkins/.ssh/known\_hosts  
  . Create the node in Jenkins GUI  
  . Manage Jenkins → Nodes → New Node  
  . Name: slave2 → Type: Permanent Agent  
  . Remote root directory: /home/ec2-user  
  . This path is taken from slave machine  
  . Slave ec2 → cat /etc/passwd  
  . chown ec2-user:ec2-user slave02-workspace  
  . chmod 777 slave02-workspace  
  . Labels: slave2 java (as you like)

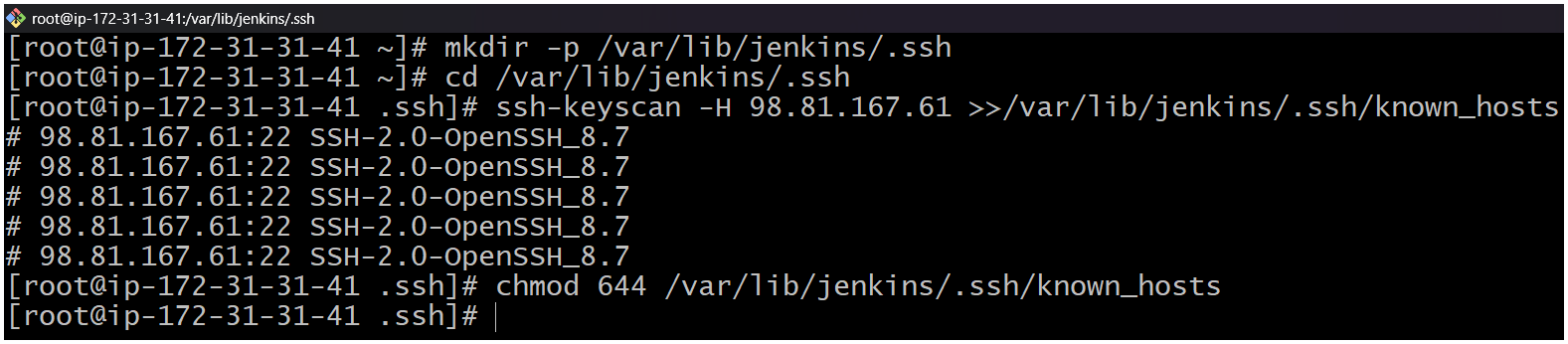




1- Create or launch two instances slave01-ec2 and slave02-ec2  
2- Install java and git on both ec2  
3- Sudo yum install -y git  
4- sudo dnf install -y java-17-amazon-corretto  
5- Copy the key gen → ssh-keygen

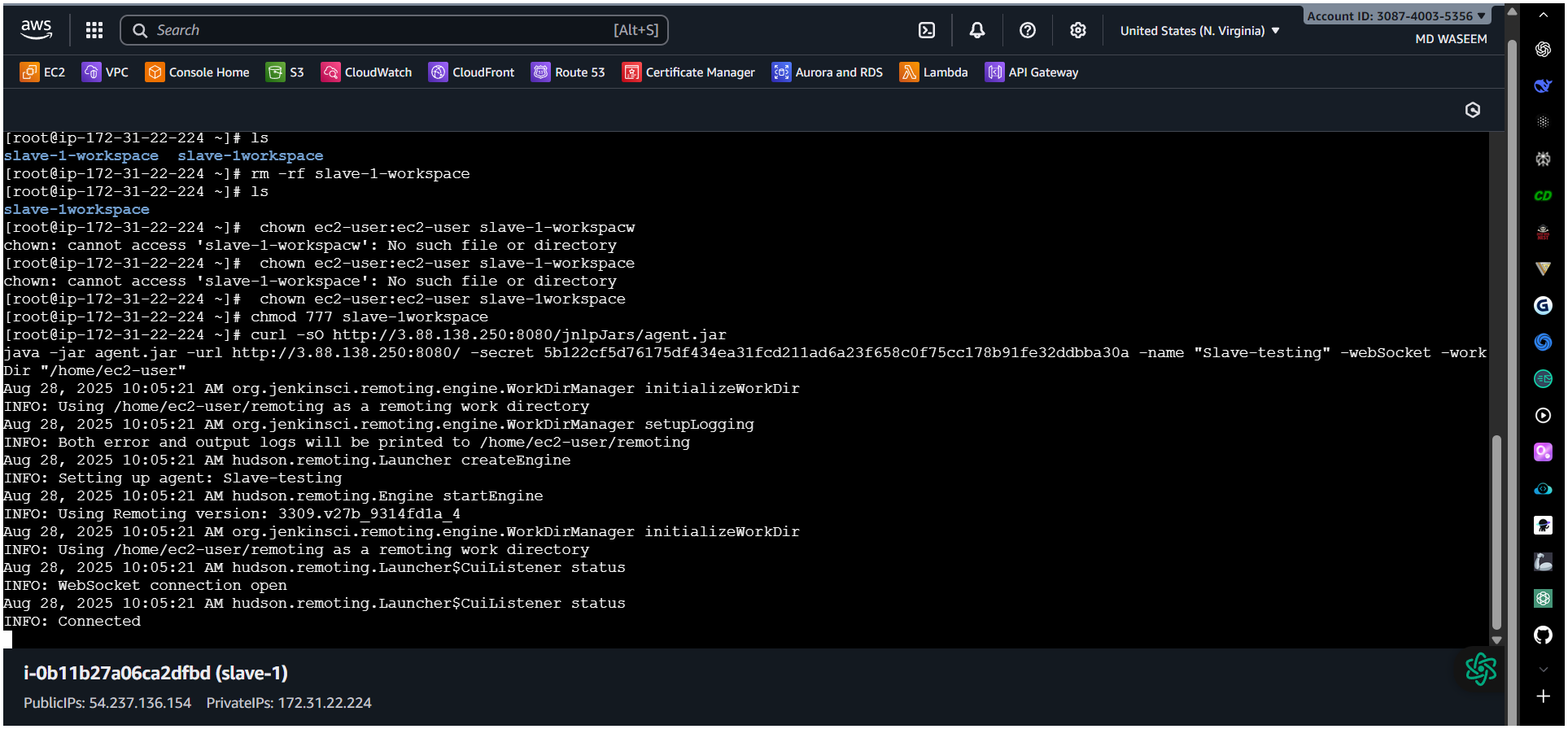
6- Cd .ssh  
7- cat id\_rsa.pub > authorized\_keys  
8- chmod 700 authorized\_keys

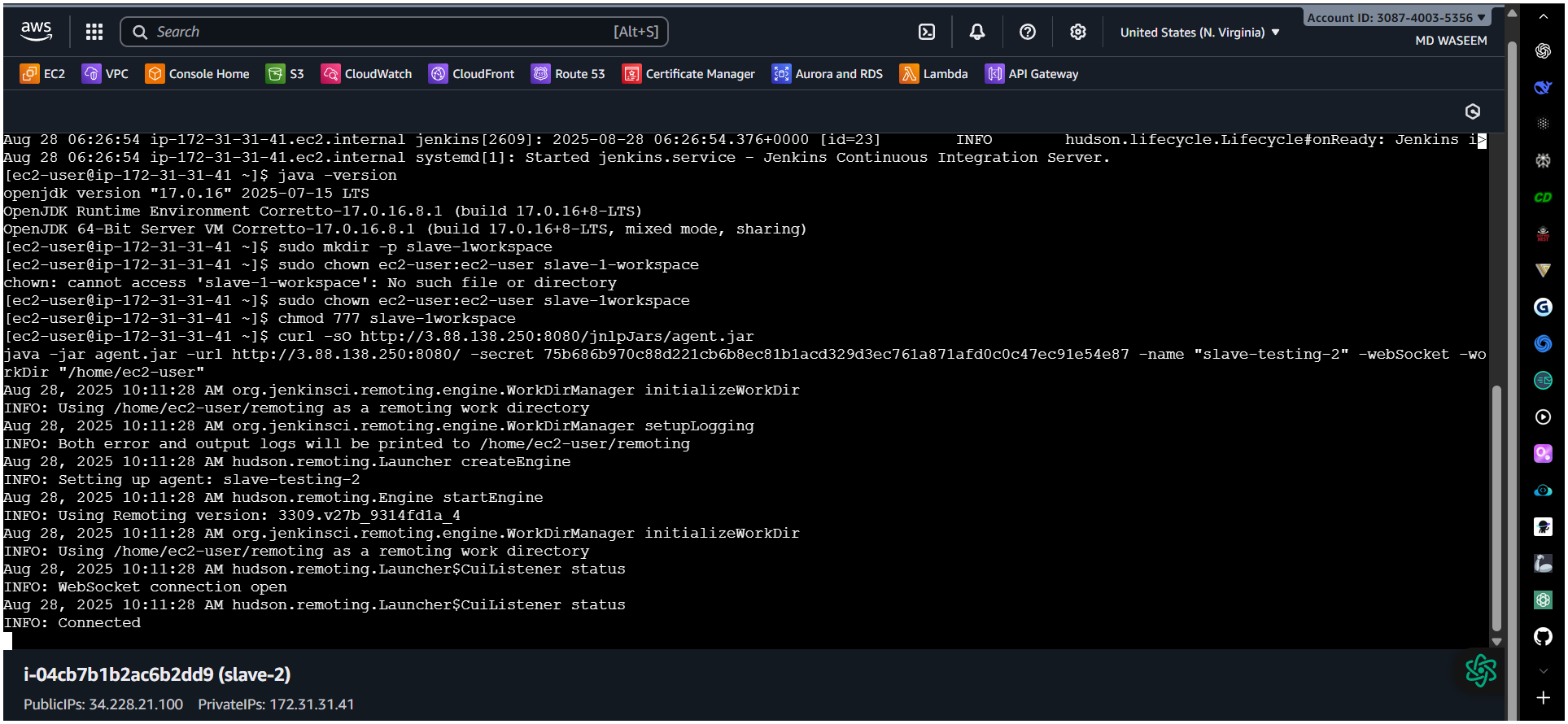
| Step | Steps on Master Machine Command | What it Does | Why it is Needed |
| --- | --- | --- | --- |
| a | *Login to master machine*Example: ssh ec2-user@<MASTER-IP> | Opens a terminal session to the Jenkins master server. | To perform all configurations on the master node where Jenkins is running. |
| b | sudo -i | Switches to the root user. | Root privileges are needed to modify system directories like /var/lib/jenkins/. |
| c | mkdir -p /var/lib/jenkins/.sshcd /var/lib/jenkins/.ssh | Creates the .ssh directory for the Jenkins user if it does not exist (-p avoids errors if it already exists) and then moves into that directory. | Jenkins uses this folder to store SSH-related files such as keys and known\_hosts for secure communication. |
| d | ssh-keyscan -H SLAVE-NODE-PUBLIC-IP >> /var/lib/jenkins/.ssh/known\_hosts | Fetches the SSH public key fingerprint of the slave machine and appends it to the known\_hosts file. -H hashes the hostname for security. | Prevents SSH from asking: *“Are you sure you want to continue connecting?”* when Jenkins master connects to the slave, enabling automatic and secure communication. |
| e | chown jenkins:jenkins /var/lib/jenkins/.ssh/known\_hosts | Changes the ownership of the known\_hosts file to the Jenkins user and group. | Jenkins runs as the jenkins user. It must have permission to read the file to establish an SSH connection. |
| f | chmod 644 /var/lib/jenkins/.ssh/known\_hosts | Sets file permissions: Owner can read/write; Group can read; Others can read. | Ensures the file is secure (not writable by others) but still readable by Jenkins for SSH authentication. |

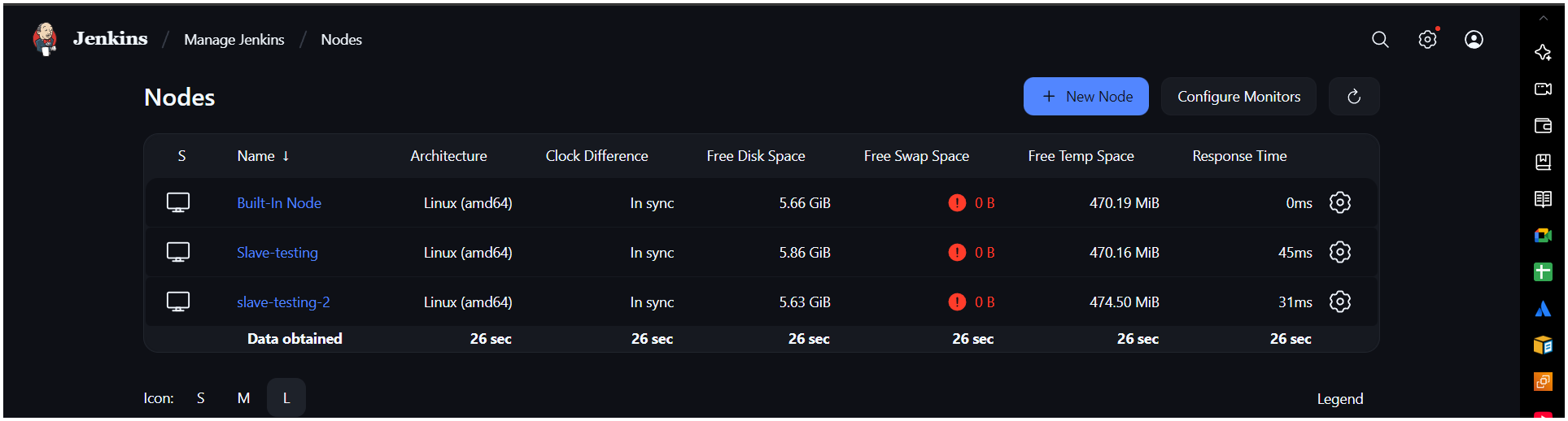
🡪Steps on Master Machine  
  
a. Login to master machine.  
b. Switch to root user.  
c. Create Jenkins SSH directory:  
mkdir -p /var/lib/jenkins/.ssh  
cd /var/lib/jenkins/.ssh  
d. ssh-keyscan -H SLAVE-NODE-PUBLIC-IP >> /var/lib/jenkins/.ssh/known\_hosts  
e. chown jenkins:jenkins /var/lib/jenkins/.ssh/known\_hosts  
f. chmod 644 /var/lib/jenkins/.ssh/known\_hosts  
  


🡪 Create the node in Jenkins GUI.  
  
1. Manage Jenkins (master) → Nodes → New Node  
2. Name: slave2 → Type: Permanent Agent  
3. Remote root directory: /home/ec2-user  
a. This path is taken from slave machine  
b. Slave ec2 → cat /etc/passwd  
c. chown ec2-user:ec2-user slave02-workspace  
d. chmod 777 slave02-workspace  
4. Labels: slave2 java (as you like)

Usage: “Only build jobs with label expressions” (optional)







6. Launch method: Launch agents via SSH  
○ Host: <NEW\_AGENT\_IP\_OR\_DNS>  
○ Credentials: create/select “SSH Username with private key”  
■ Username: ec2-user  
■ Private key: paste content of /var/lib/jenkins/.ssh/id\_rsa   
from the master  
○ Host Key Verification Strategy: Known hosts file (recommended)

2) Configure webhooks to Jenkins job.

✅ What is a Webhook in Jenkins?

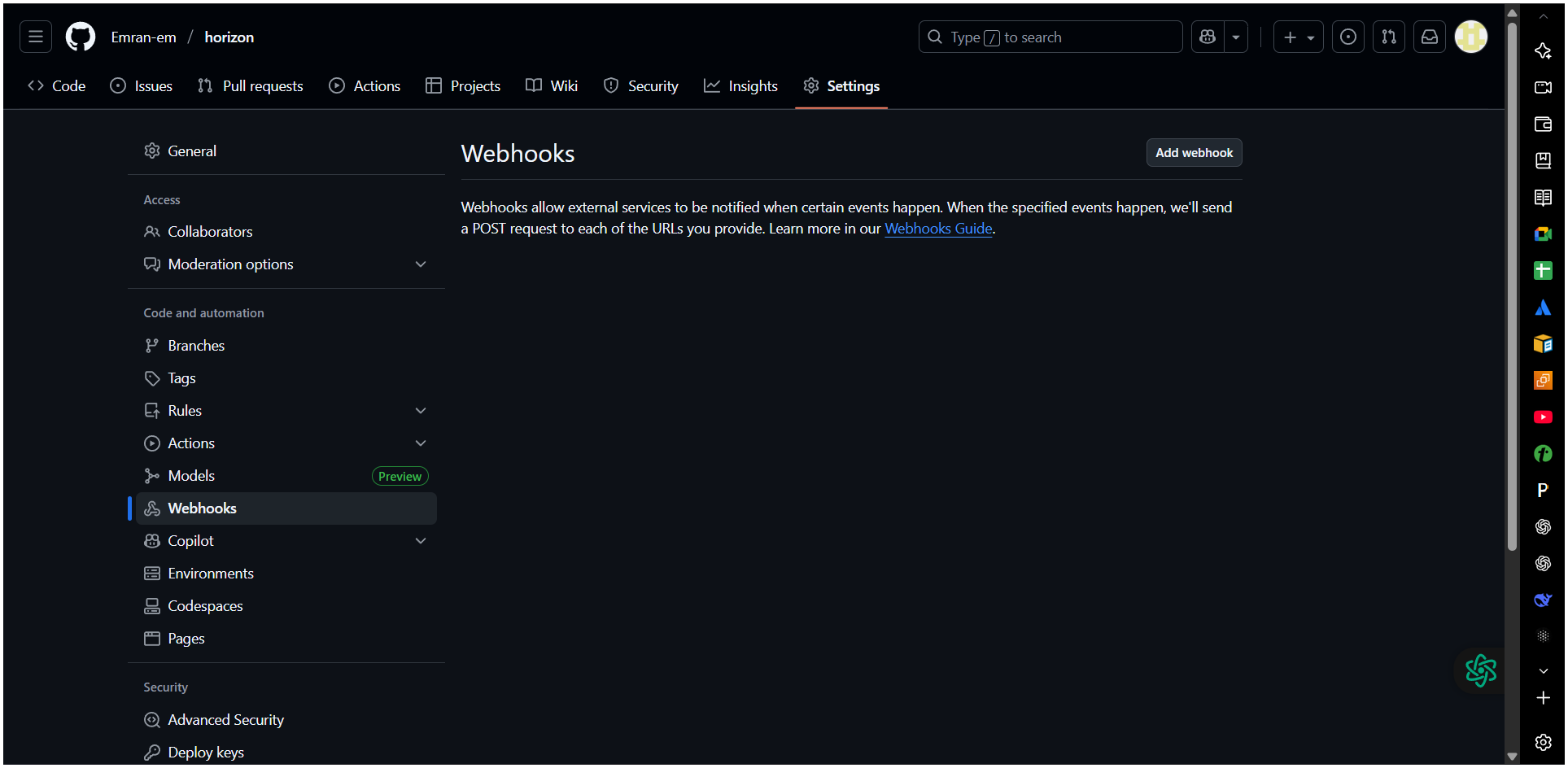
A webhook is an HTTP callback that sends real-time data from your repository (e.g., GitHub) to Jenkins whenever a specific event occurs (like a push, PR merge, or tag creation).  
Instead of Jenkins polling the repository every few minutes, the webhook notifies Jenkins immediately.

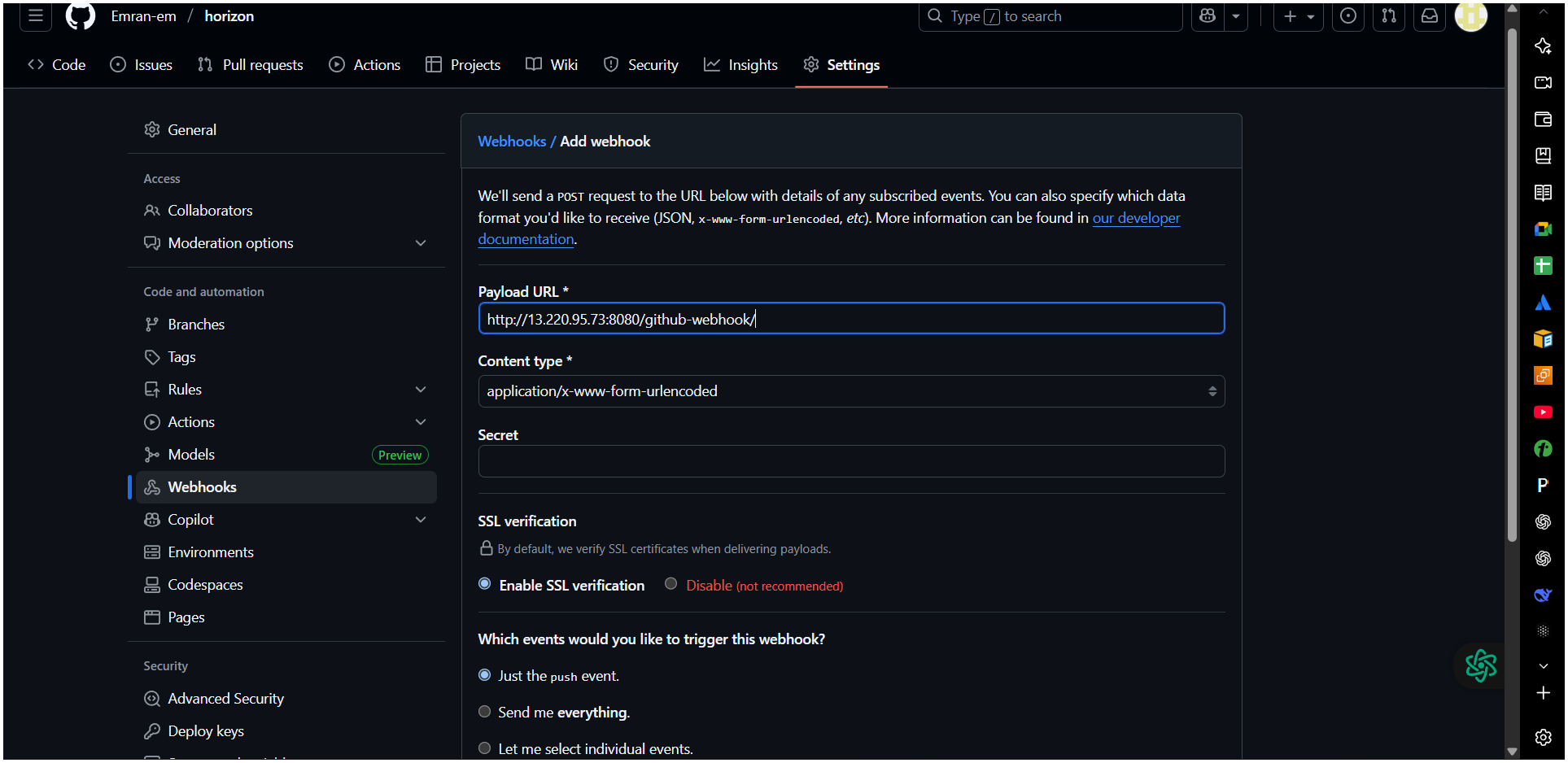
✅ Why Do We Need Webhooks?

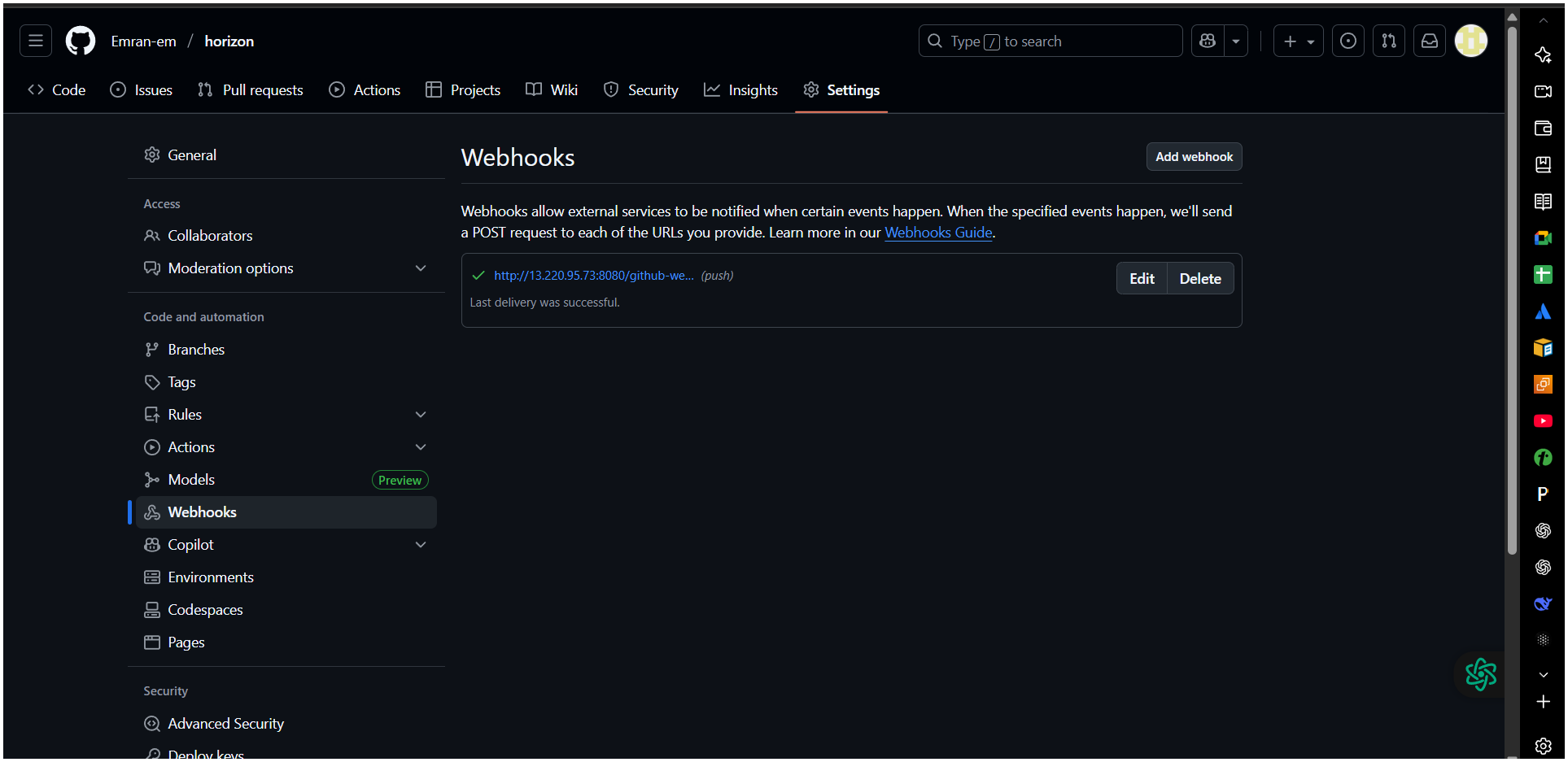
1. Instant Build Trigger – No waiting for polling; builds start as soon as changes are pushed.
2. Reduces Load – Jenkins doesn’t need to continuously check the repo for changes.
3. Efficient CI/CD – Improves deployment speed and automation.
4. Real-Time Feedback – Developers get instant feedback for their commits.

🡪GitHub → Webhook event → Jenkins /GitHub-webhook/ endpoint → Job   
Trigger → Build runs

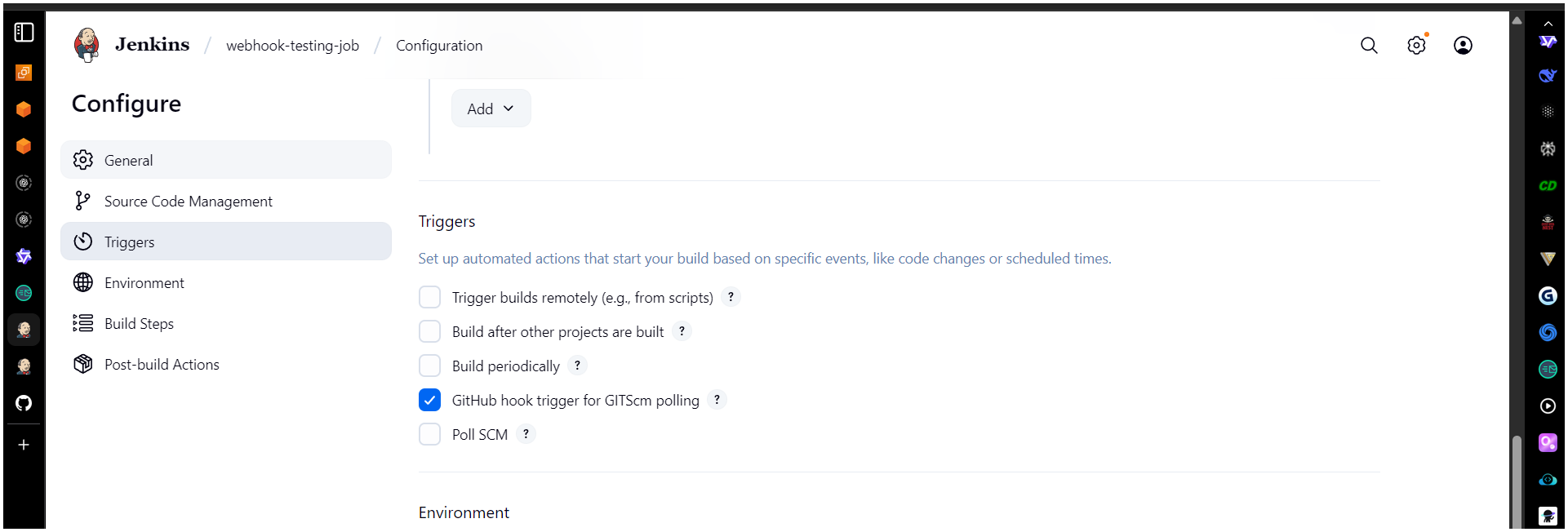
Create / Configure Jenkins Job  
🡪Create a Freestyle Project –(webhook-job)  
🡪Under Source Code Management, choose Git and add your   
repository URL

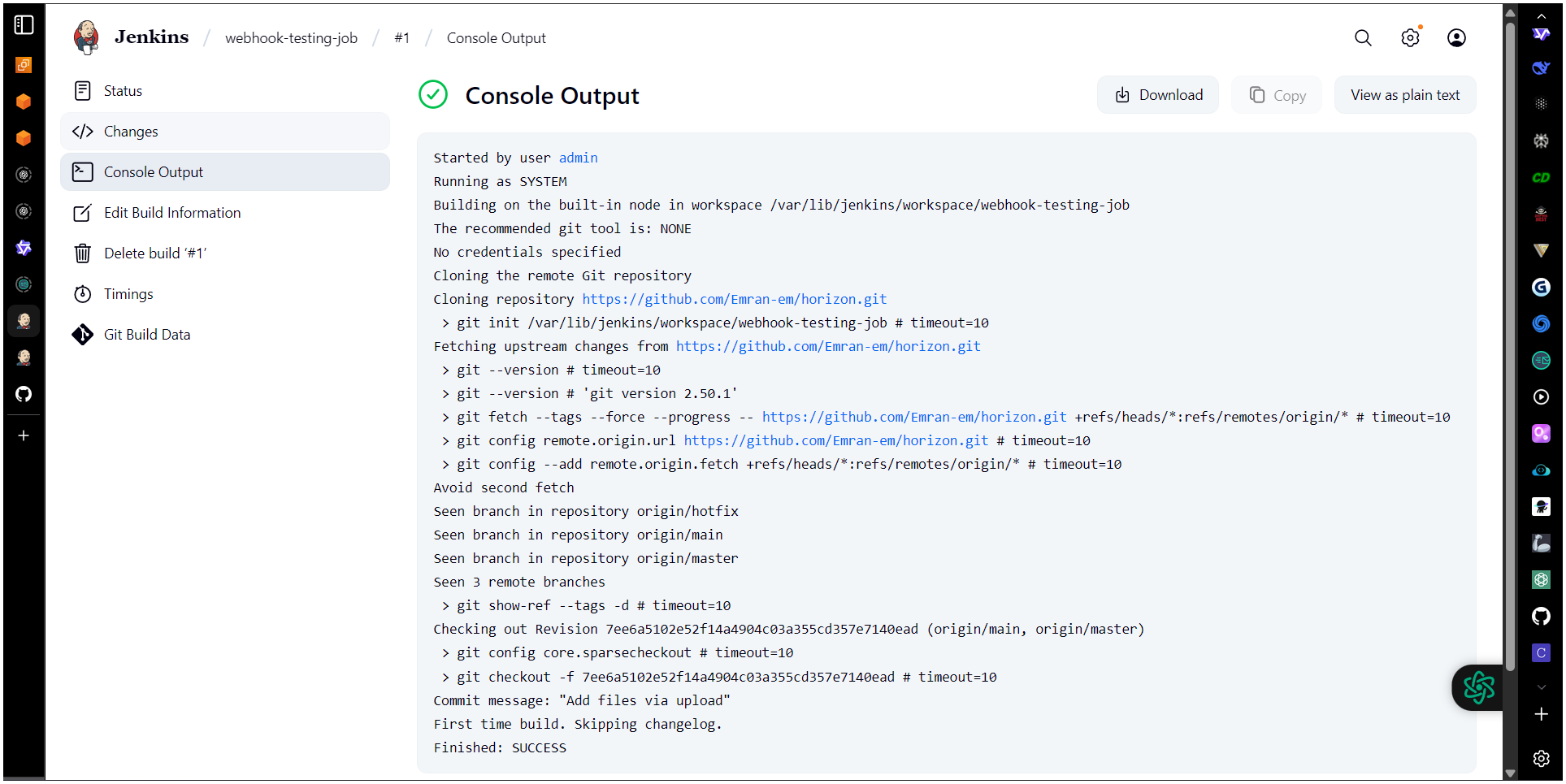


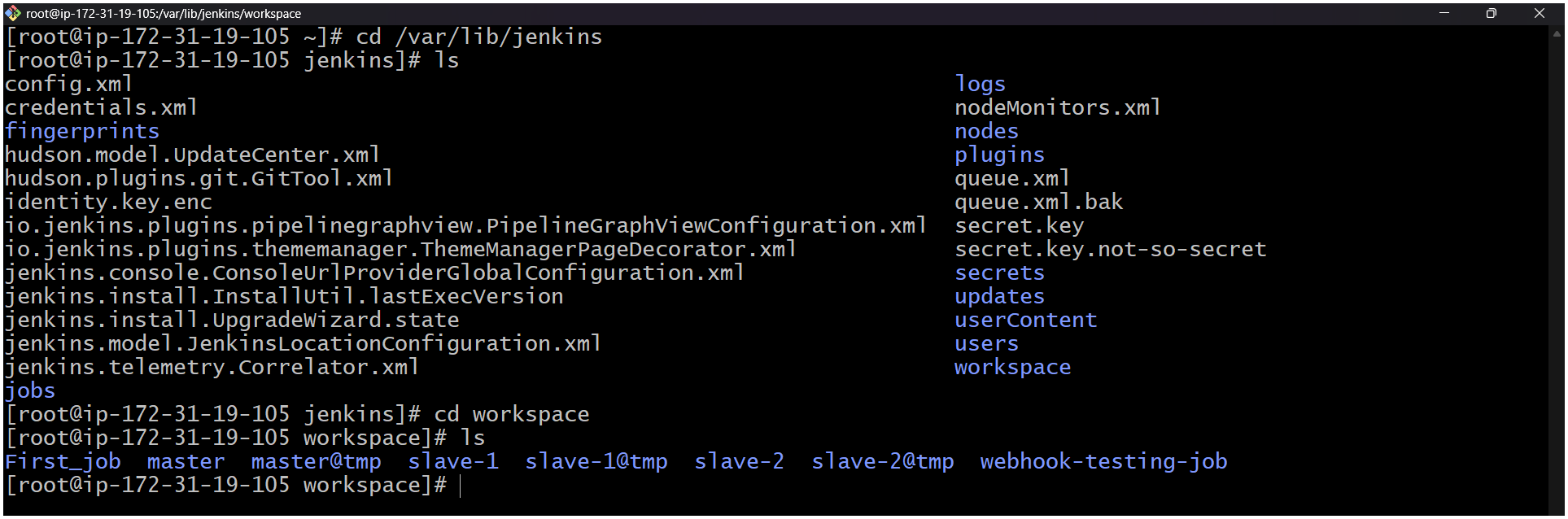




Enable Build Trigger  
🡪In your Jenkins job → Configure → Scroll to Build Triggers.  
🡪Check:  
 GitHub hook trigger for GITScm polling  
This tells Jenkins to listen for webhook events instead of polling GitHub   
periodically.



Verify  
🡪Push a commit to your GitHub repo.  
🡪Jenkins job should trigger automatically.  
🡪Check Jenkins logs to confirm webhook was received

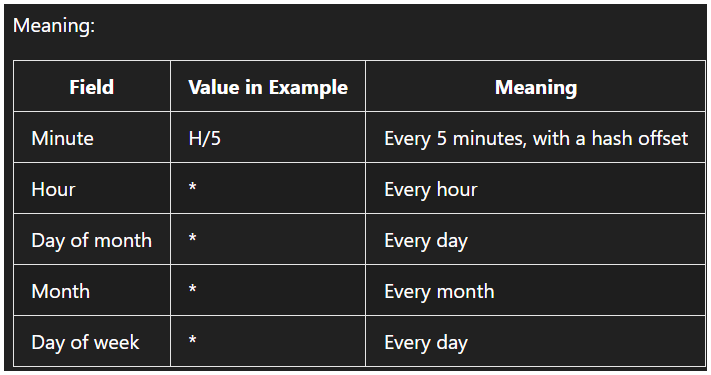


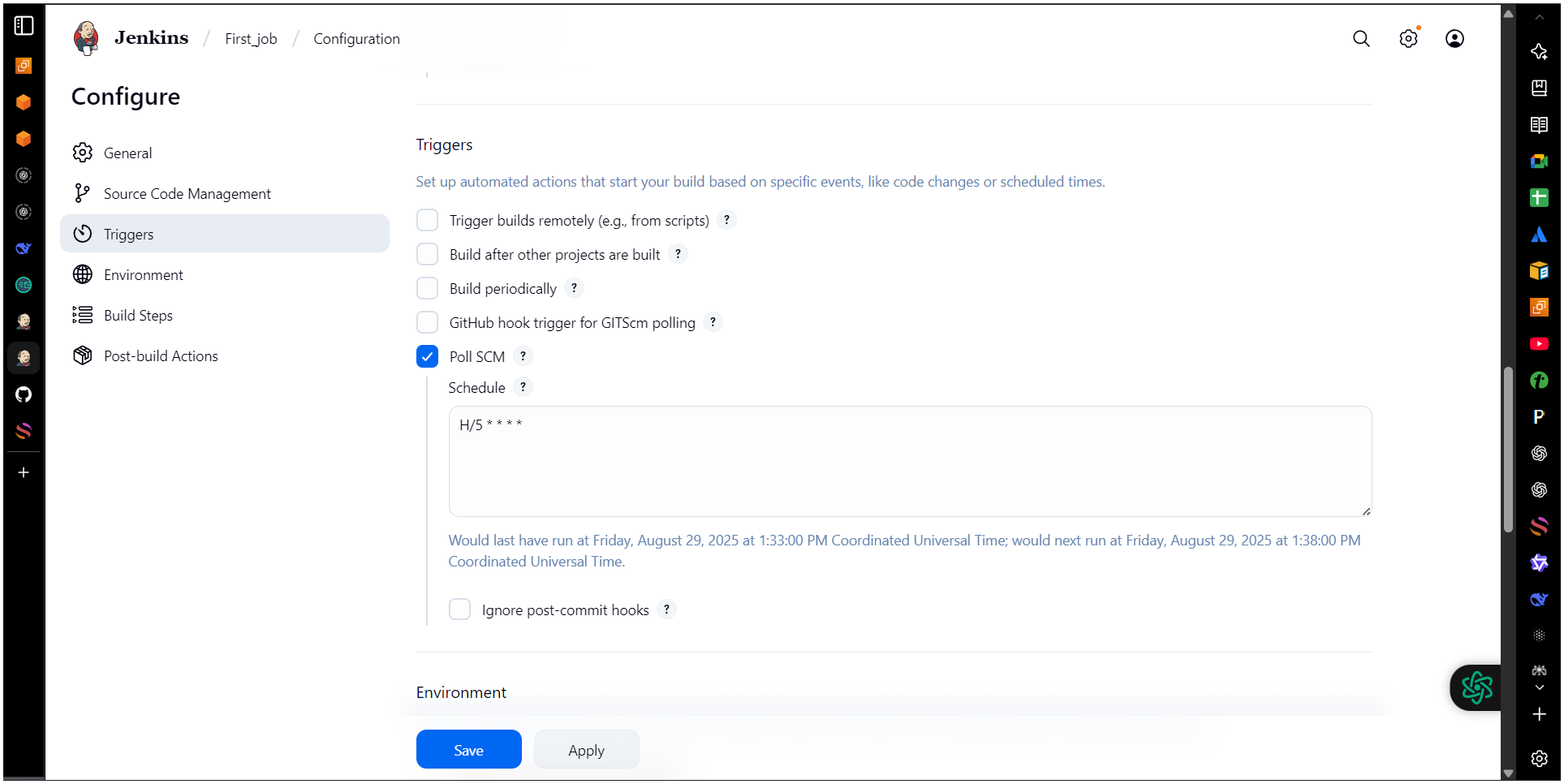
3) Configure poll scm and build periodical options in Jenkins job.

**A. Poll SCM**

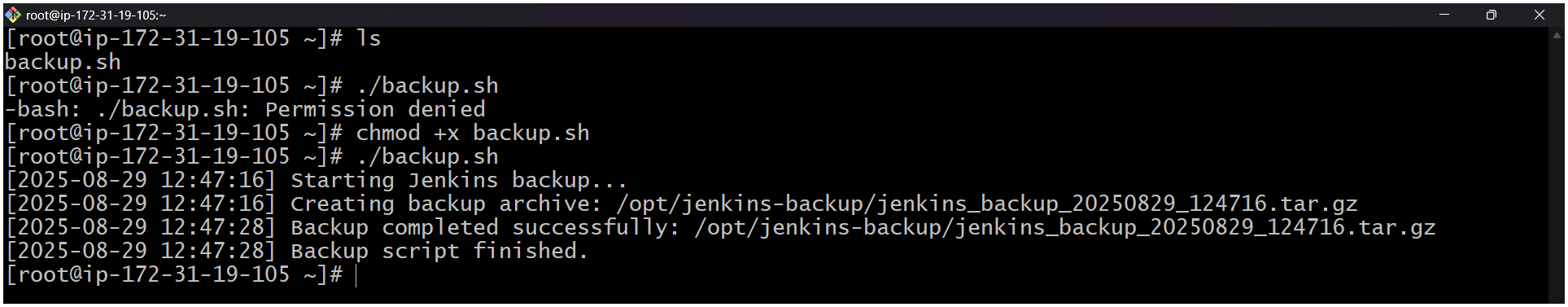
* Go to **Your Job** → **Configure**.
* Under **Build Triggers**, check **Poll SCM**.
* Enter a **schedule syntax**

H/5 \* \* \* \* # Checks every 5 minutes





4) Take backup of Jenkins server by using bash script.



#!/bin/bash

# =============================================

# Jenkins Backup Script

# Author: DevOps Engineer

# Purpose: Backup JENKINS\_HOME directory

# Output: Compressed tar.gz file with timestamp

# =============================================

# --- CONFIGURATION ---

JENKINS\_HOME="/var/lib/jenkins" # Default path; change if different

BACKUP\_DIR="/opt/jenkins-backup" # Where to store backups

DATE=$(date +"%Y%m%d\_%H%M%S")

BACKUP\_FILE="jenkins\_backup\_${DATE}.tar.gz"

# Optional: Retention (keep last N backups)

MAX\_BACKUPS=7

# Jenkins service name (used for safe mode if needed)

JENKINS\_SERVICE="jenkins"

# Enable safe backup mode? (Disable Jenkins temporarily)

SAFE\_MODE=false # Set to true for zero-downtime risk

# --- END CONFIGURATION ---

# Create backup directory if it doesn't exist

mkdir -p "$BACKUP\_DIR"

# Log function

log() {

echo "[$(date +'%Y-%m-%d %H:%M:%S')] $1"

}

# Check if JENKINS\_HOME exists

if [ ! -d "$JENKINS\_HOME" ]; then

log "ERROR: JENKINS\_HOME directory not found at $JENKINS\_HOME"

exit 1

fi

log "Starting Jenkins backup..."

# Optional: Stop Jenkins for consistency (safe mode)

if [ "$SAFE\_MODE" = true ]; then

log "Stopping Jenkins service to ensure data consistency..."

sudo systemctl stop "$JENKINS\_SERVICE"

if [ $? -ne 0 ]; then

log "ERROR: Failed to stop Jenkins service"

exit 1

fi

fi

# Create compressed backup

log "Creating backup archive: $BACKUP\_DIR/$BACKUP\_FILE"

tar -czf "$BACKUP\_DIR/$BACKUP\_FILE" -C "$(dirname "$JENKINS\_HOME")" "$(basename "$JENKINS\_HOME")"

# Check if tar succeeded

if [ $? -eq 0 ]; then

log "Backup completed successfully: $BACKUP\_DIR/$BACKUP\_FILE"

else

log "ERROR: Backup failed during archive creation"

# Restart Jenkins even if failed

if [ "$SAFE\_MODE" = true ]; then

sudo systemctl start "$JENKINS\_SERVICE"

fi

exit 1

fi

# Restart Jenkins after backup

if [ "$SAFE\_MODE" = true ]; then

log "Restarting Jenkins service..."

sudo systemctl start "$JENKINS\_SERVICE"

fi

# Enforce retention policy: Keep only last N backups

cd "$BACKUP\_DIR" || exit 1

ls -t jenkins\_backup\_\*.tar.gz | tail -n +$((MAX\_BACKUPS + 1)) | while read file; do

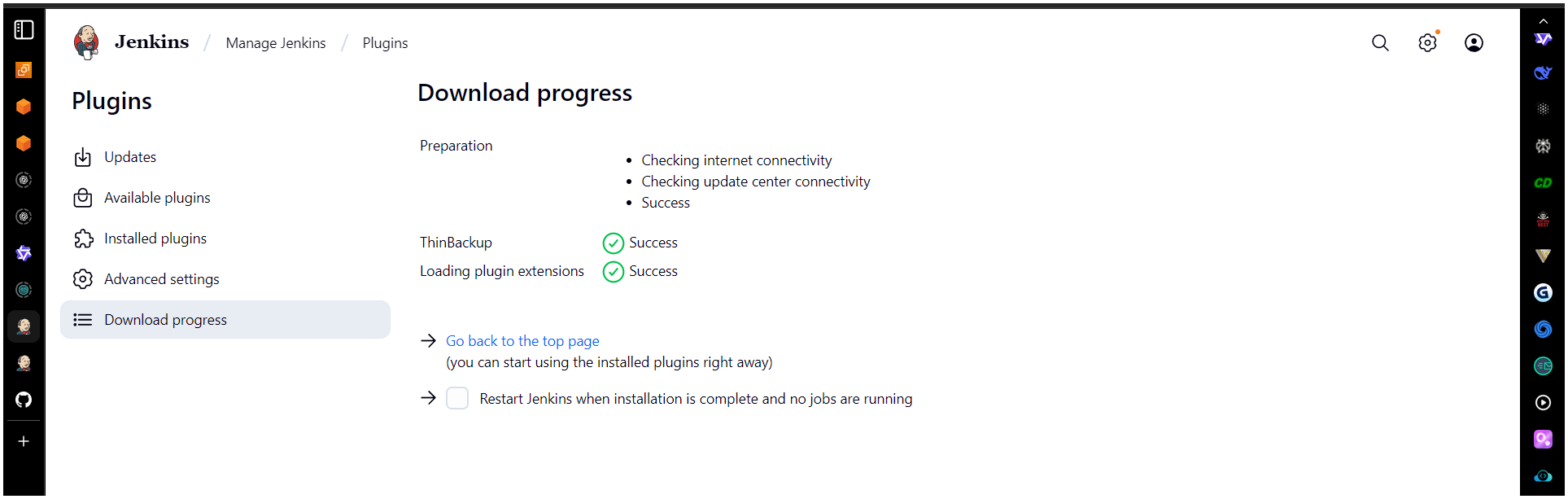
log "Removing old backup: $file"

rm -f "$file"

done

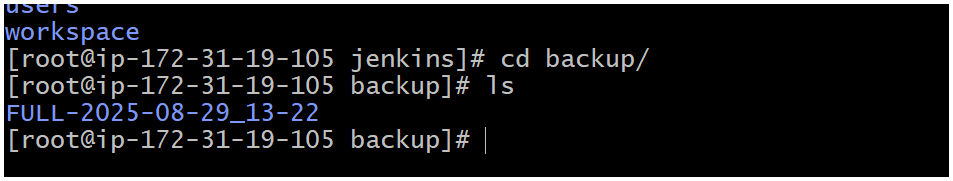
log "Backup script finished."  
  
  
**5) Take backup of Jenkins using rethin backup plugin.**  
🡪Step 1: Access Jenkins  
  
1. Open your Jenkins URL in a browser: http://<your-server-ip>:8080.  
2. Log in with an admin account.

🡪Step 2: Install ThinBackup Plugin  
  
1. Go to Manage Jenkins → Manage Plugins.  
2. Click the Available tab.  
3. Search for ThinBackup.

4. Check the box next to ThinBackup.  
5. Click Install without restart  
  
  
  
  
🡪Step 3: Configure ThinBackup  
  
1. Go to Manage Jenkins → System  
2. Click Configure.  
3. Set the following:Backup directory: Path where backups will be stored (e.g.,  
/var/Jenkins-backup).  
4. Click Save.

🡪Navigate to manage Jenkins 🡪scroll down (you’ll see) **ThinBackup**  
Step 4: Take Manual Backup

🡪 Cd /var/lib/Jenkins

🡪 ls  
🡪 cd backup/  


**6) Setup a new Jenkins server and dump the backup taken in task4.**