**Jenkins-02**

1. **Configure 2 slave machines in Jenkins master.**

Jenkins Master–Slave Configuration

Steps to be done on slave

Step-1 : Create an ec2 –slave-01

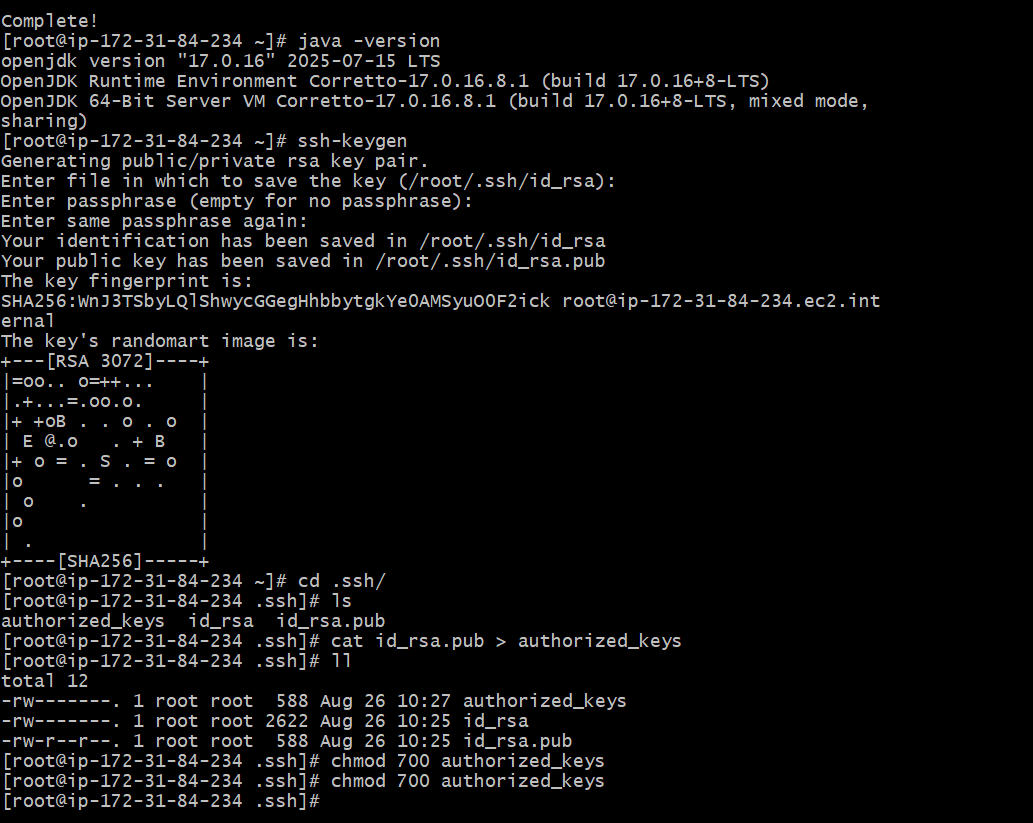
Login to slave machine..

switch to root user

sudo dnf install -y java-17-amazon-corretto

Create ssh-keygen

cat id\_rsa.pub > authorized\_keys

chmod 700 authorized\_keys 

Steps to be done on master machine:

Login to Master machine

switch to root user.

Prepare Jenkins SSH directory

mkdir -p /var/lib/jenkins/.ssh

cd /var/lib/jenkins/.ssh

Add slave’s SSH key to known\_hosts

ssh-keyscan -H SLAVE-NODE-IP-OR-HOSTNAME>>/var/lib/jenkins/.ssh/known\_hosts

# ssh-keyscan -H 172.31.38.42 >>/var/lib/jenkins/.ssh/known\_hosts

Fix permissions

chown jenkins:jenkins known\_hosts

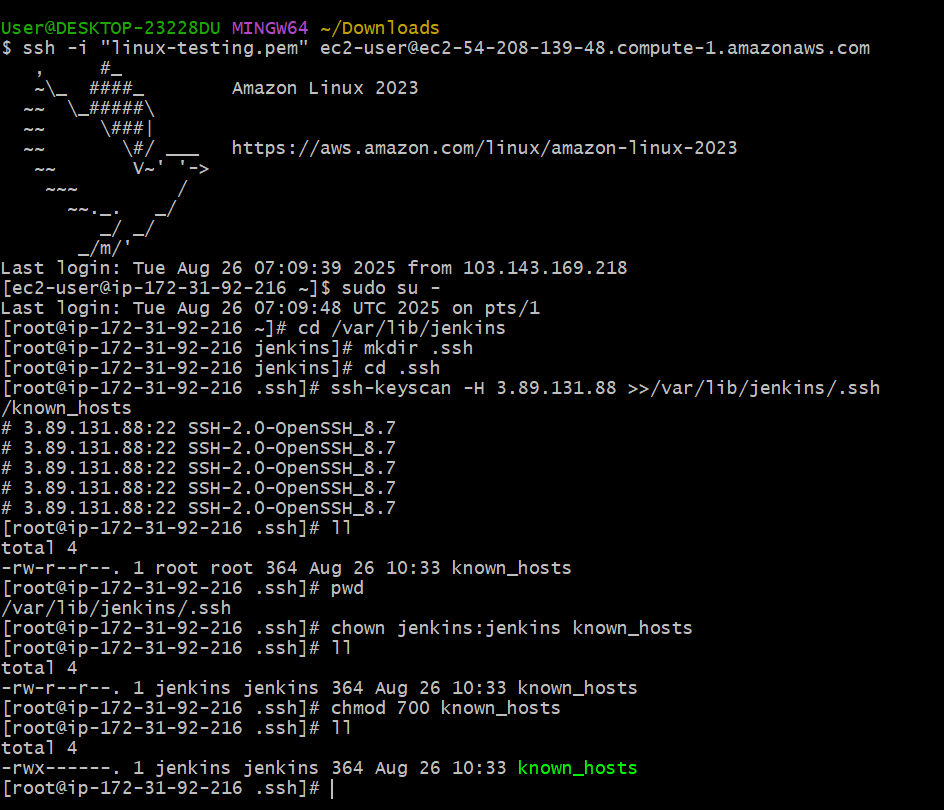
#we need to change the owner as we ran ssh-keyscan command using “root”

user.

# default user of Jenkins will be “jenkins”

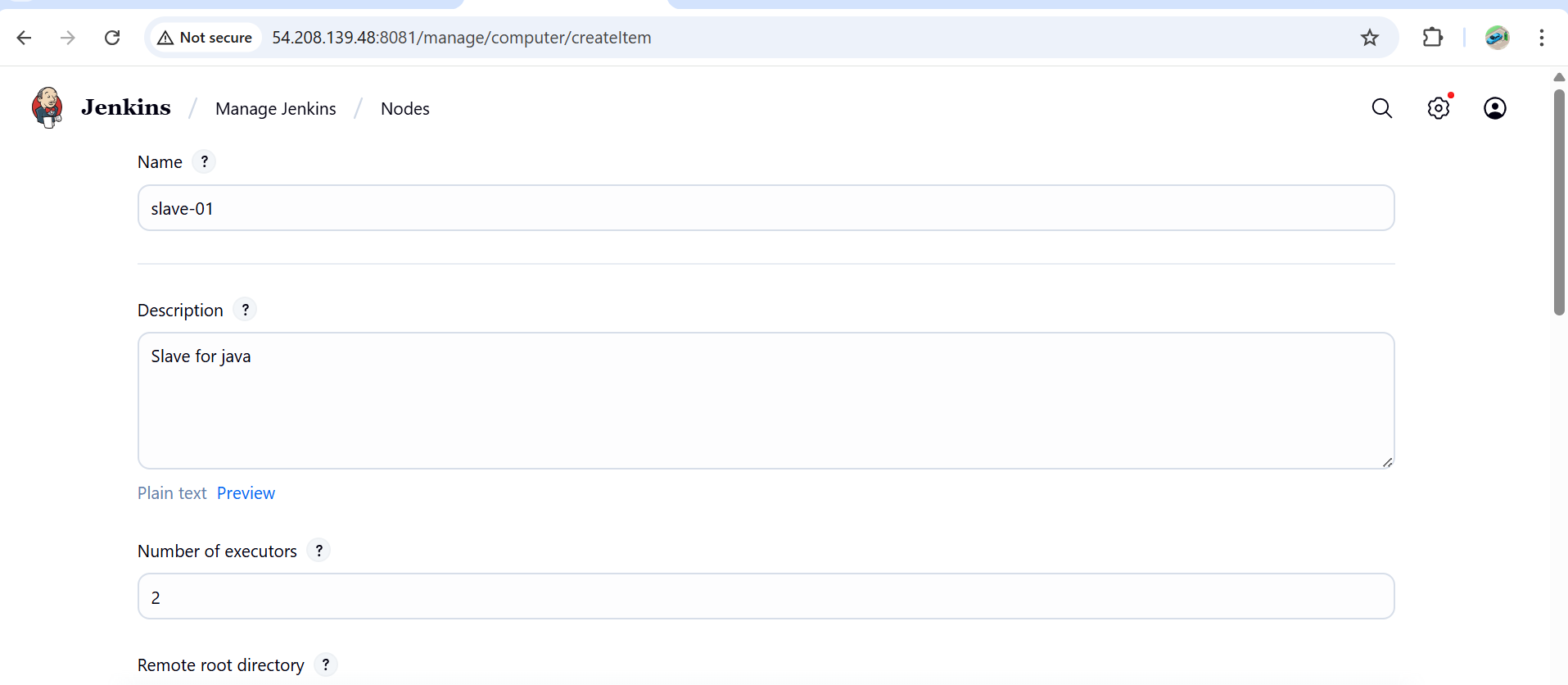
chmod 700 known\_hosts

# Done !!

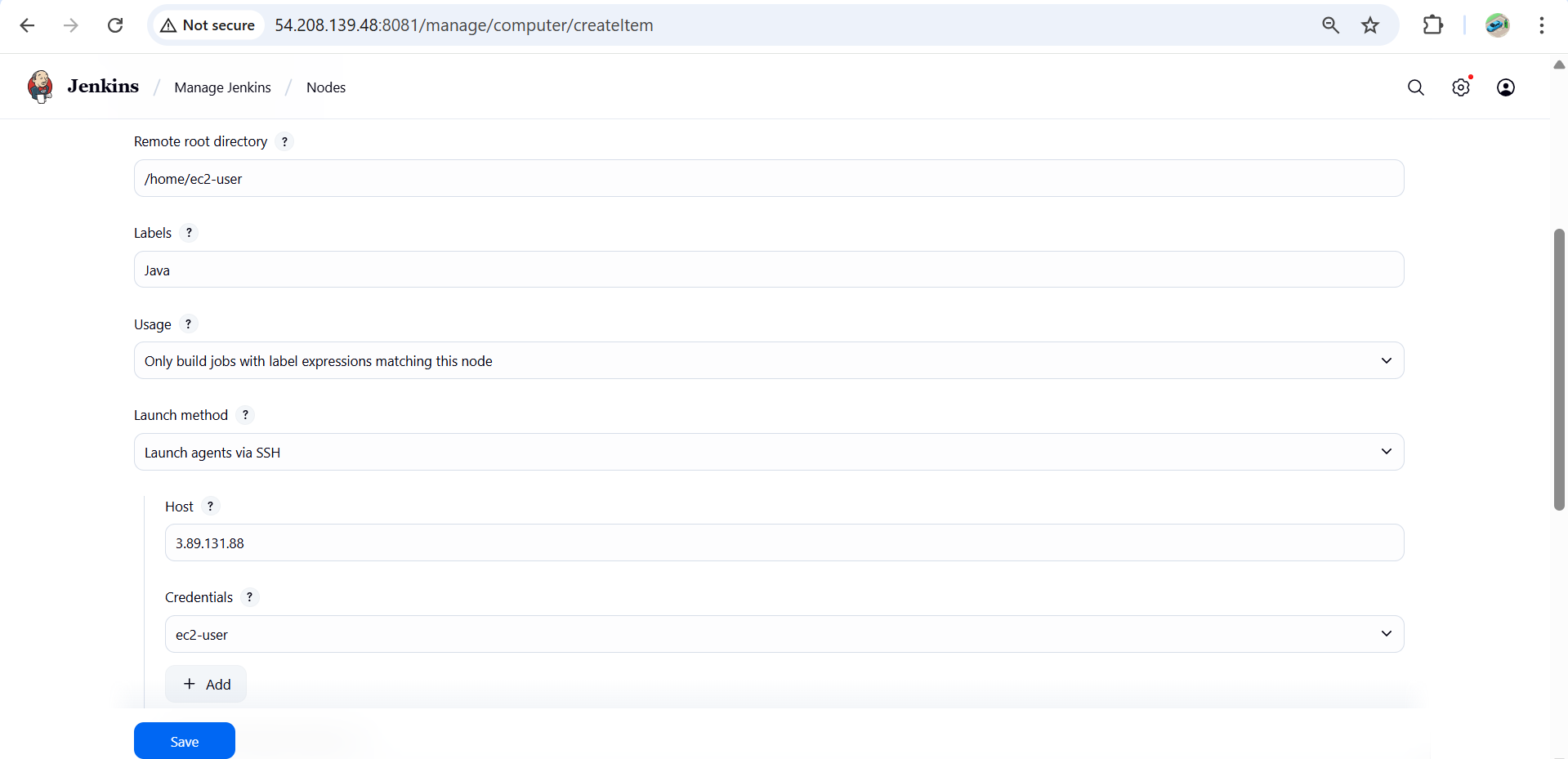


**Steps on Jenkins GUI**

1. Go to **Manage Jenkins → Manage Nodes and Clouds → New Node**
   * Give it a name (e.g., slave-1)
   * Select **Permanent Agent**

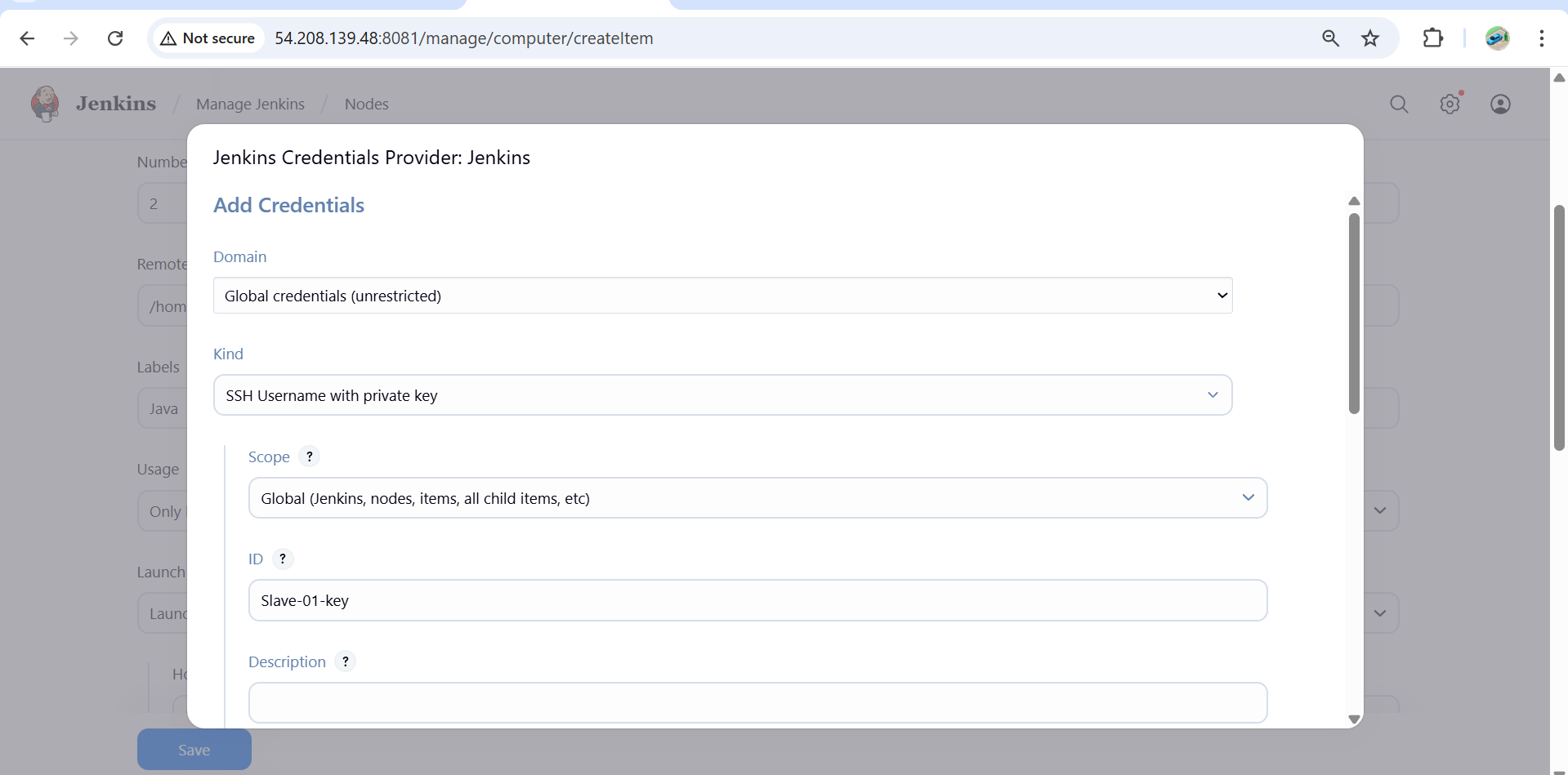


1. **Configure Node**
   * **Remote root directory**: /home/ec2-user (or another working dir on slave)
   * **Labels**: (optional, e.g., Java)
   * **Launch method**: *Launch agent via SSH*
   * **Host**: *Public IP of Slave*



**Add Credentials**

* + Kind: *SSH Username with private key*
  + Username: ec2-user
  + Private key: Paste your .pem key (from AWS)
  + Save

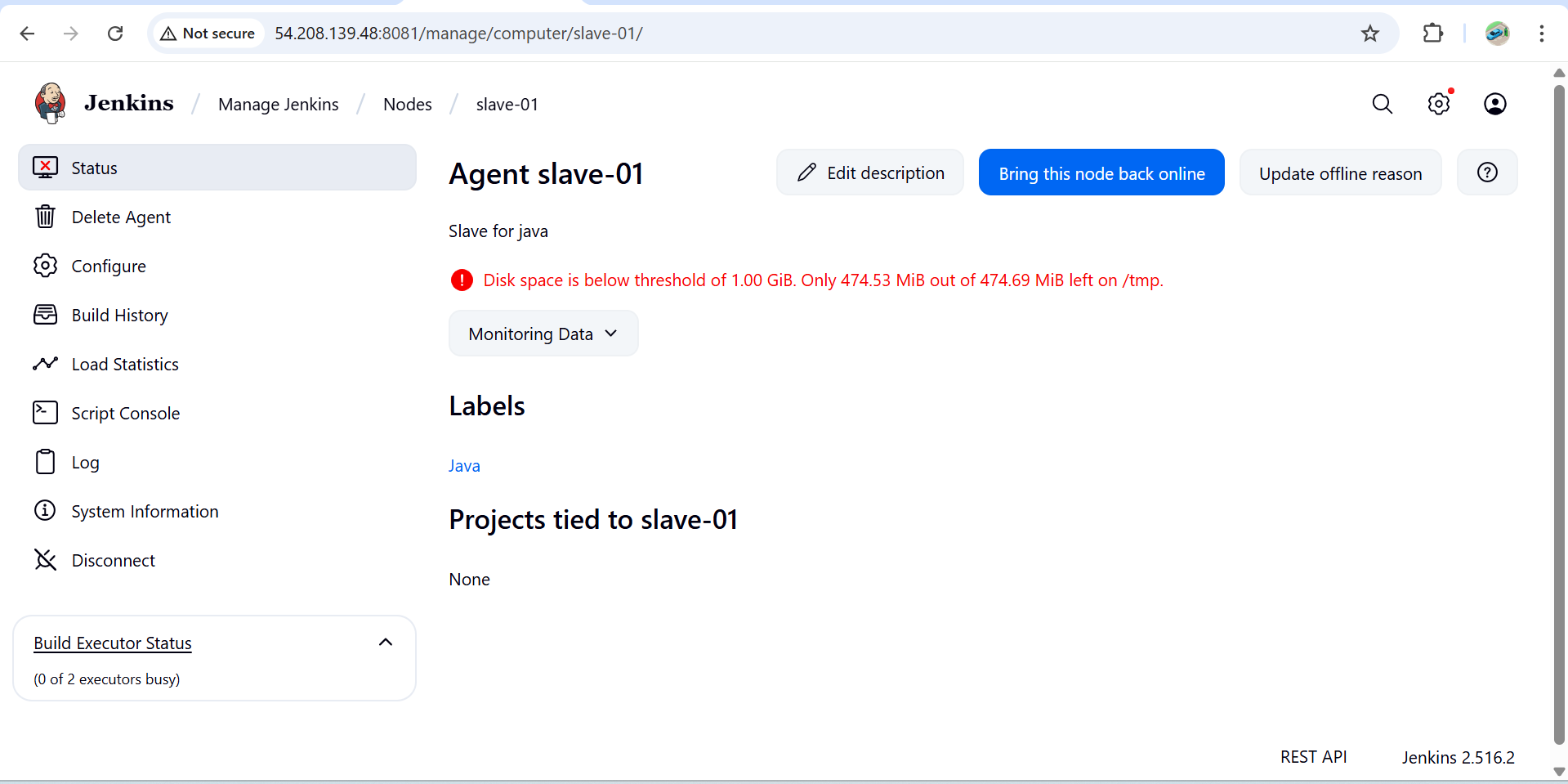


1. **Save & Test Connection**
   * Jenkins master should now connect to the slave machine via SSH.
   * If successful, the node status will show as **online**.

Your **slave-01 agent is offline**

* This means the **/tmp partition on the slave machine is almost full**. Jenkins automatically puts the node offline when it detects free space is lower than the threshold.

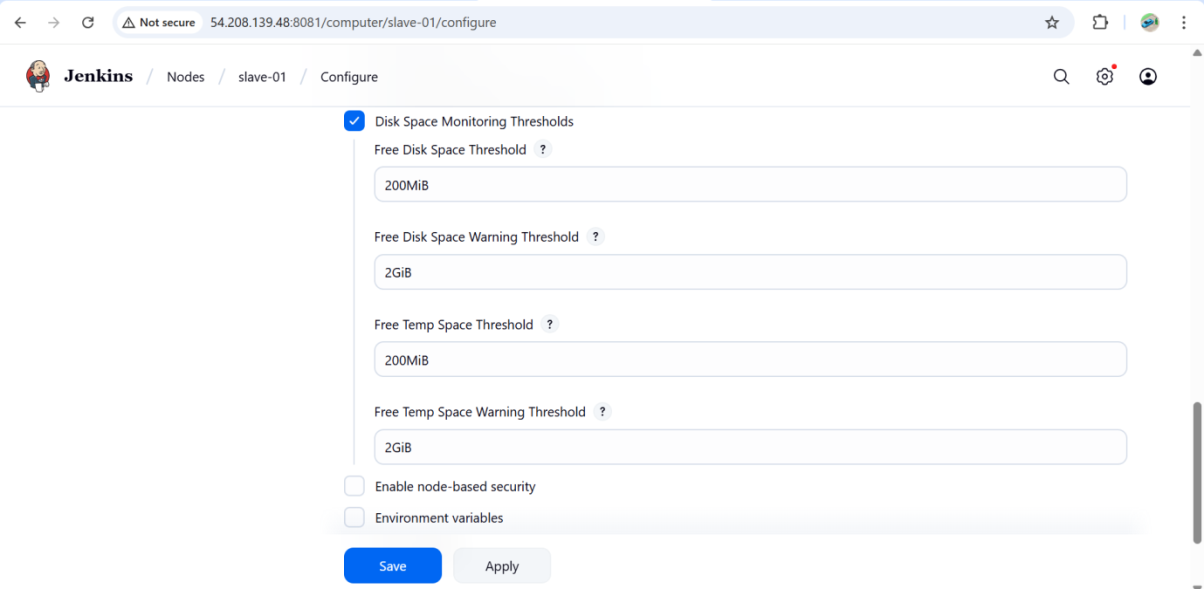
.

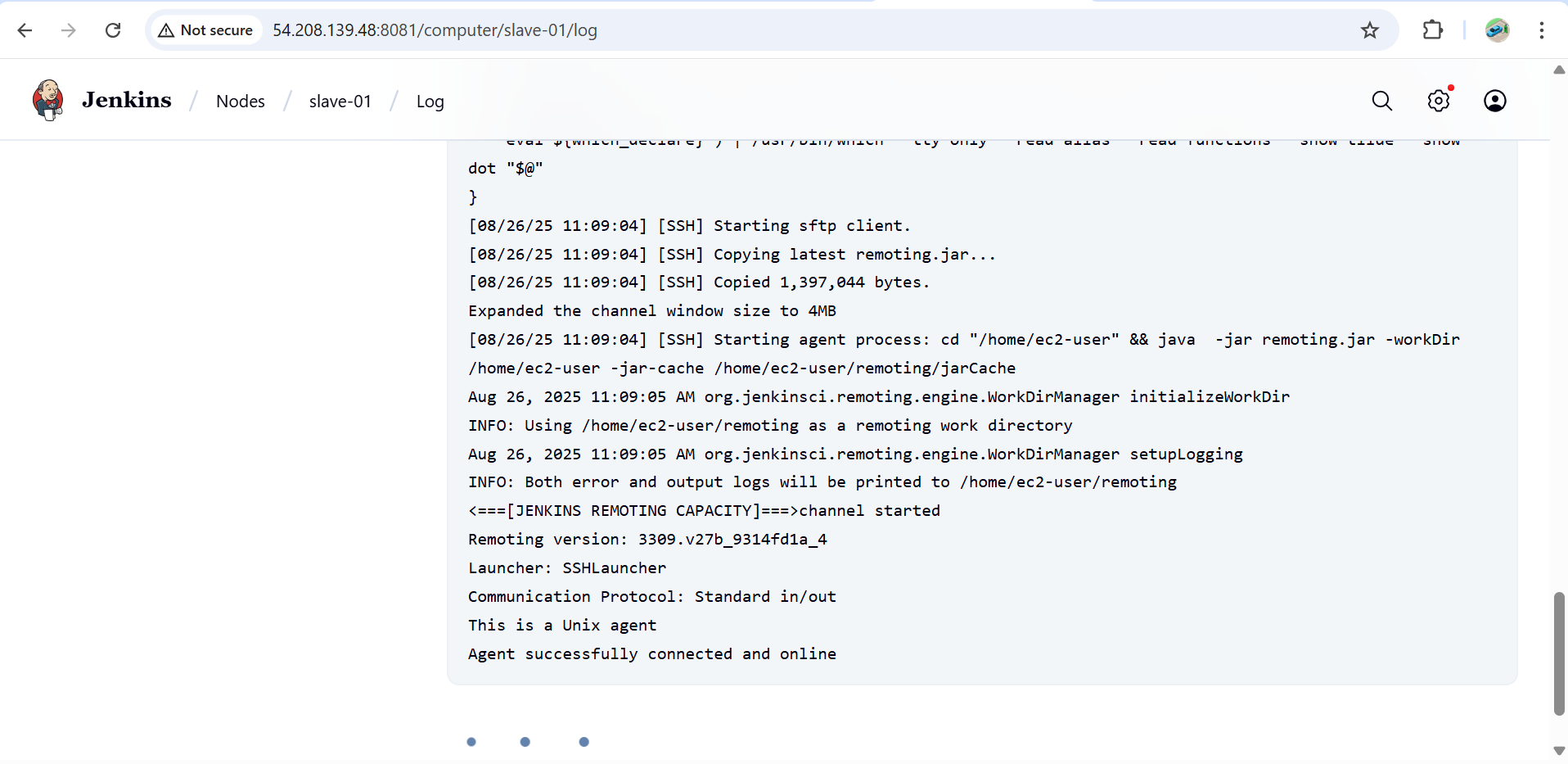


Lower Jenkins Disk Threshold

1. Uncheck **Disk Space Monitoring Thresholds**, OR
2. Set:
   * Free Disk Space Threshold = 200MiB
   * Free Temp Space Threshold = 200MiB

Click **Save** → Then **Bring this node back online**





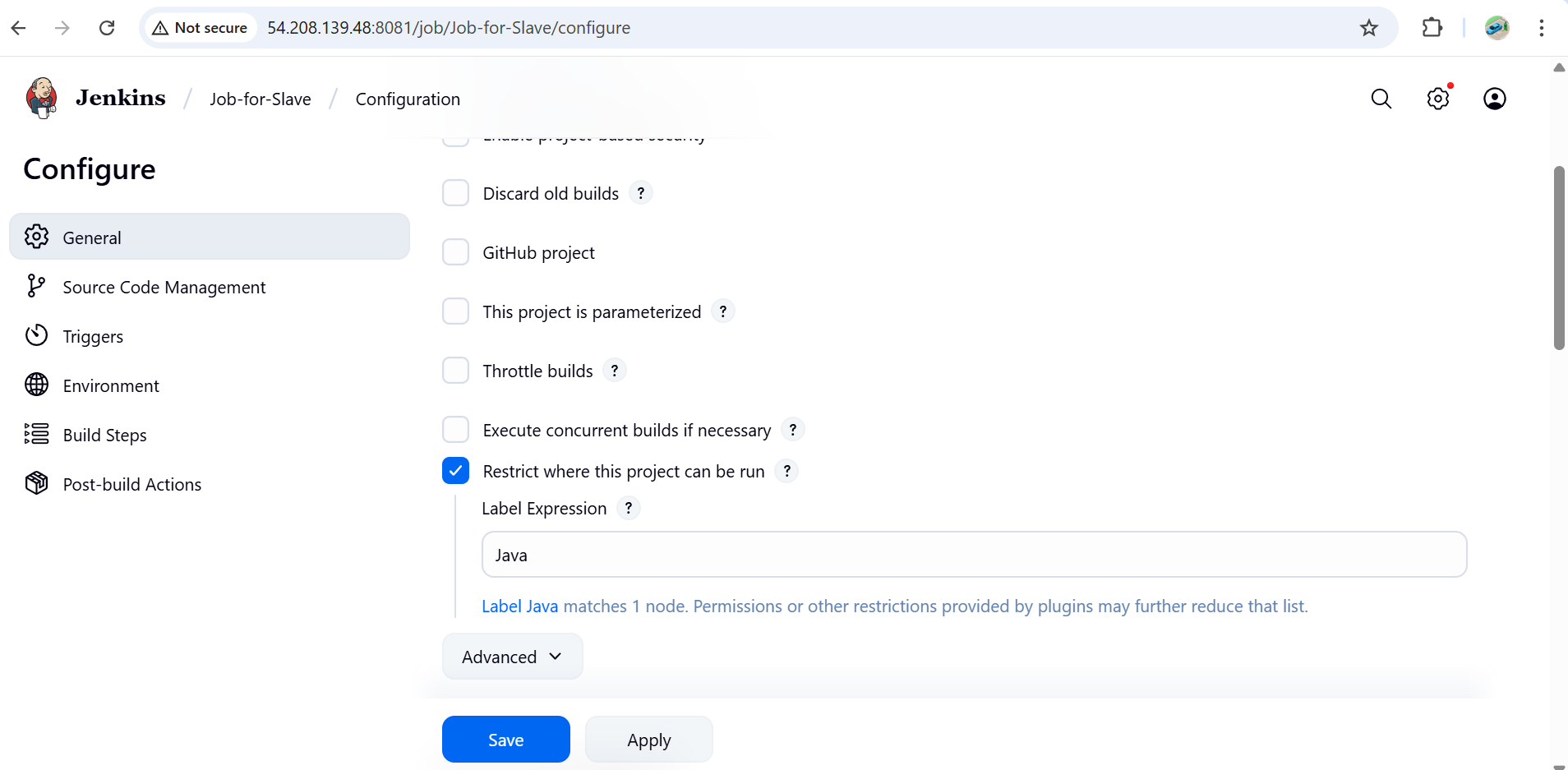
Let’s test it with a job tied to the label java (the label you assigned to that node).

### ****1. Create a New Freestyle Job and Check****

1. Go to Jenkins → **New Item**.
2. Enter a job name (e.g., Job-for-slave).
3. Select **Freestyle project** → Click **OK**.

### ****2. Configure Job to Run on Slave****

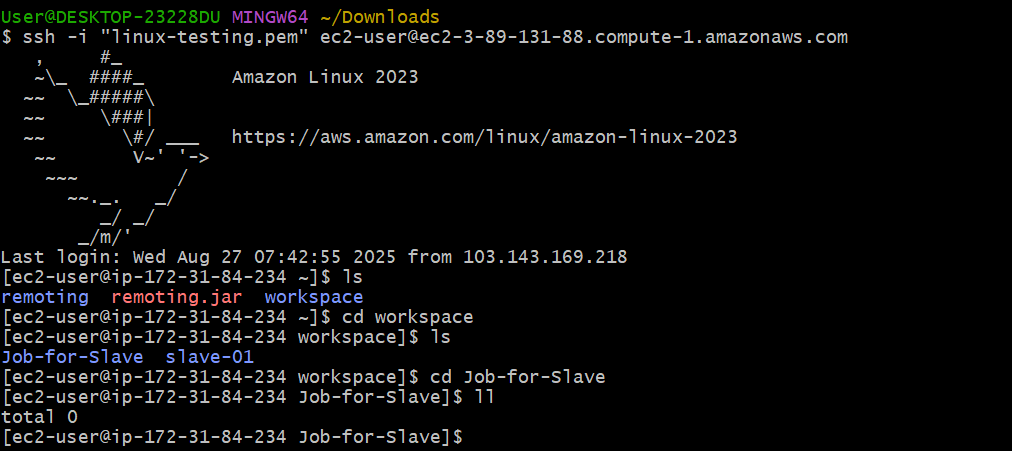
1. In job configuration:
   * Scroll to **General** section.
   * Check **Restrict where this project can be run**.
   * In **Label Expression**, enter: java
   * (This ensures the job runs only on nodes with the label java → your slave-01).



### ****4. Save & Build****

* Save the job.
* Click **Build Now**.

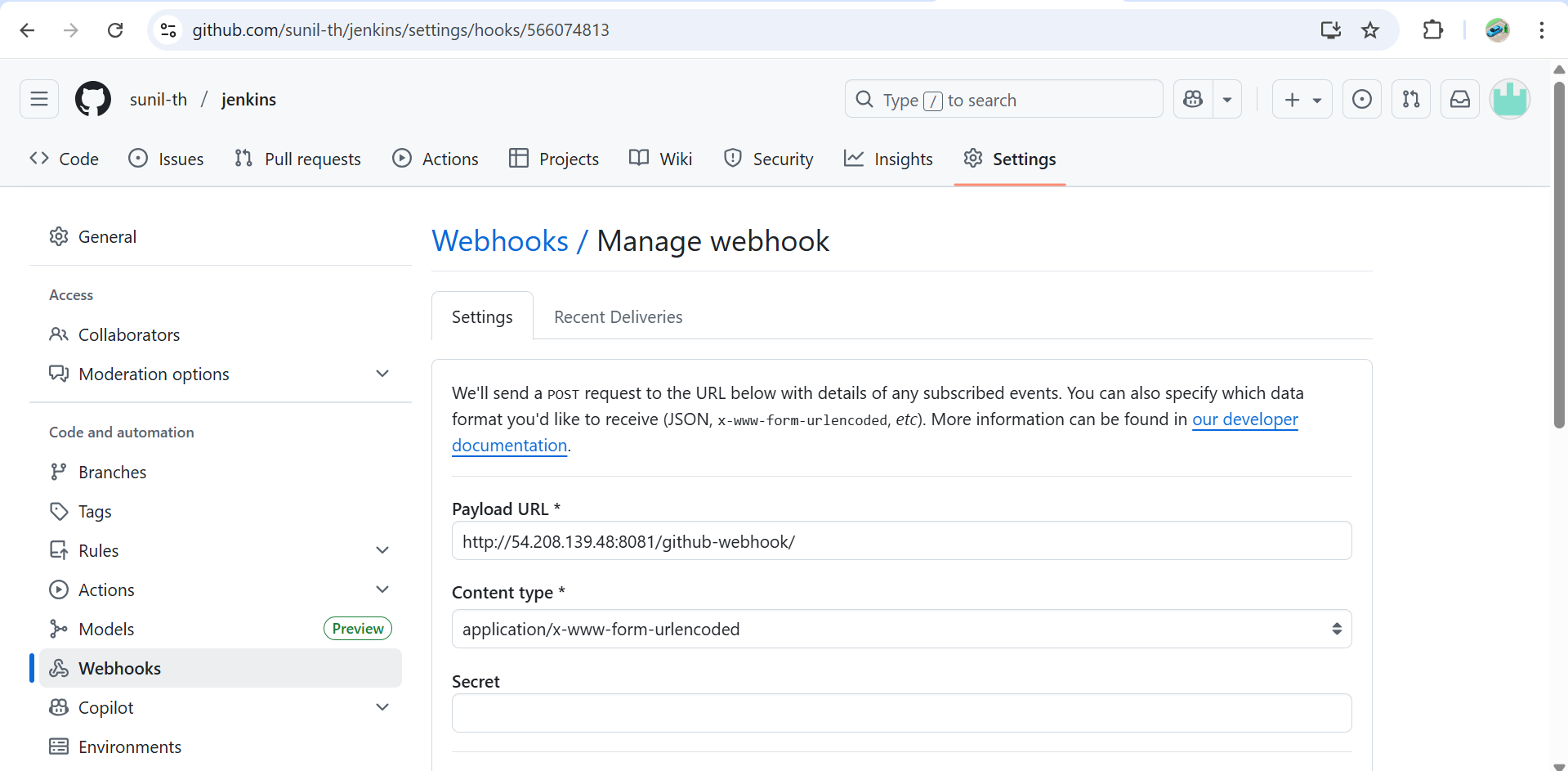


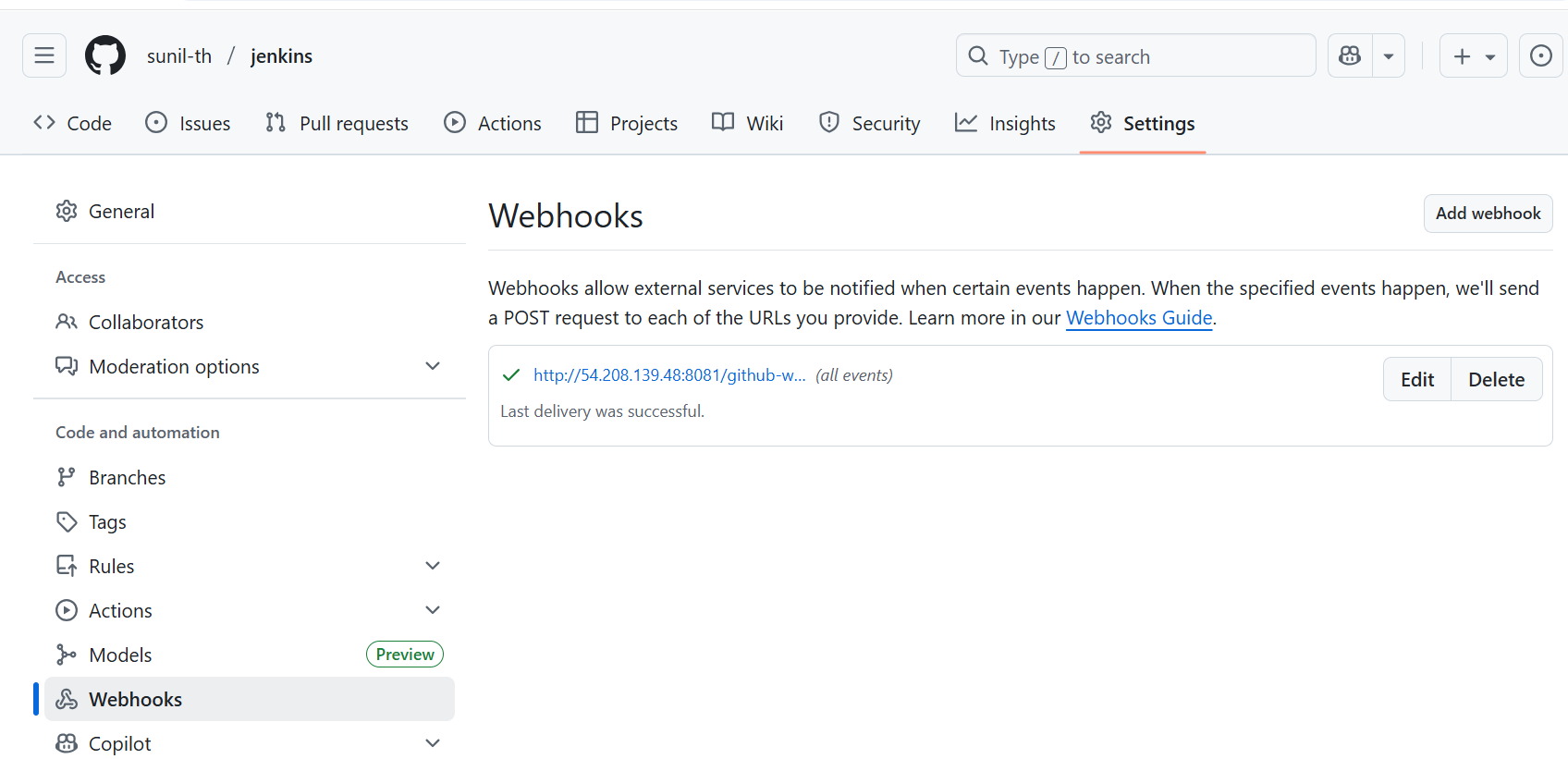


1. **Configure webhooks to Jenkins job.**

### **Configure GitHub Webhook**

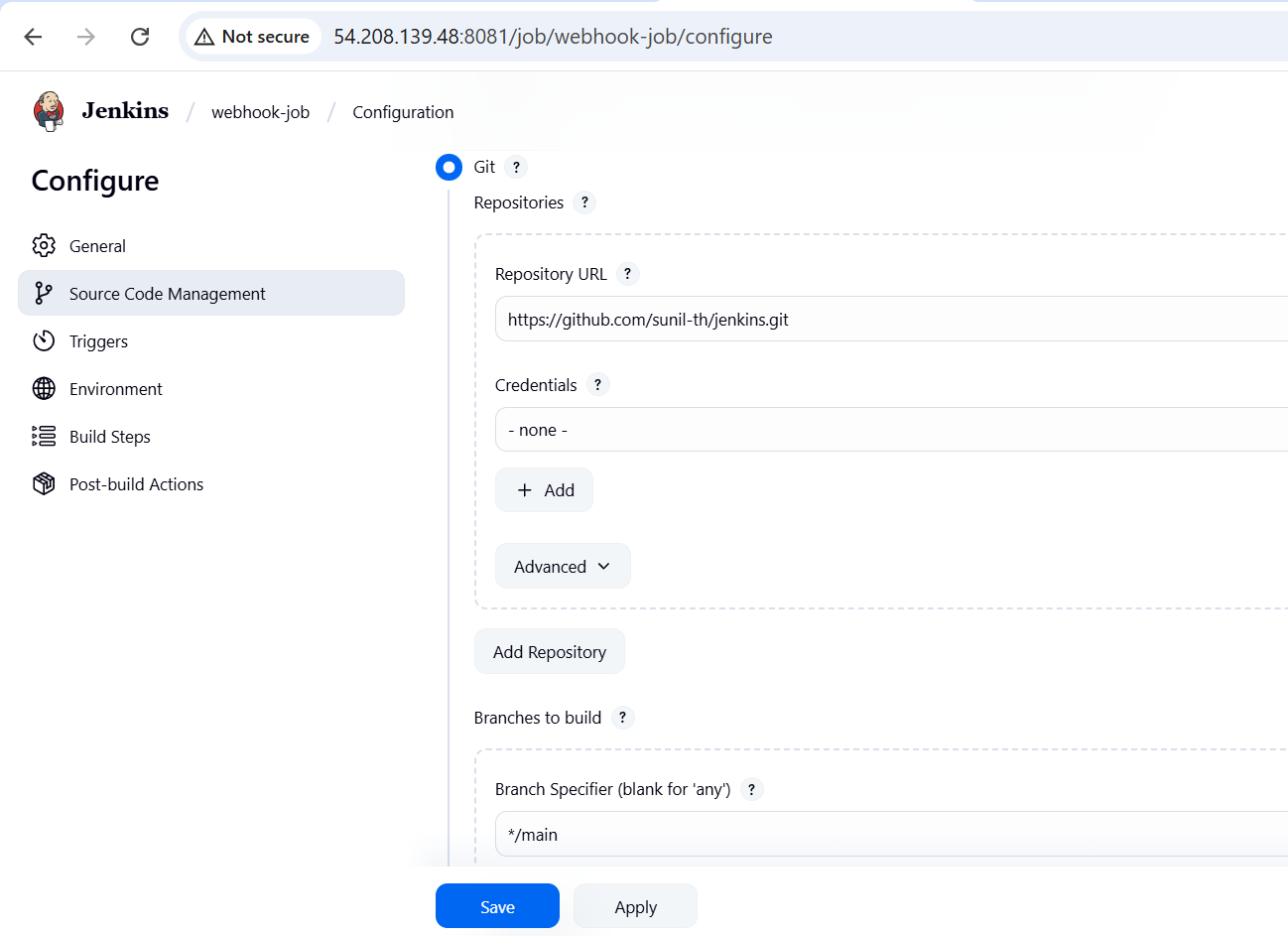
* Go to your GitHub repository → **Settings** → **Webhooks** → **Add webhook**.
* In the **Payload URL**, enter:
* http://<your-jenkins-server>:8080/github-webhook/
* Click **Add webhook**.
* Choose Send me **everything**.

****

****

### . ****Create / Configure Jenkins Job****

* Create a **Freestyle Project** –(webhook-job)
* Under **Source Code Management**, choose **Git** and add your repository URL.

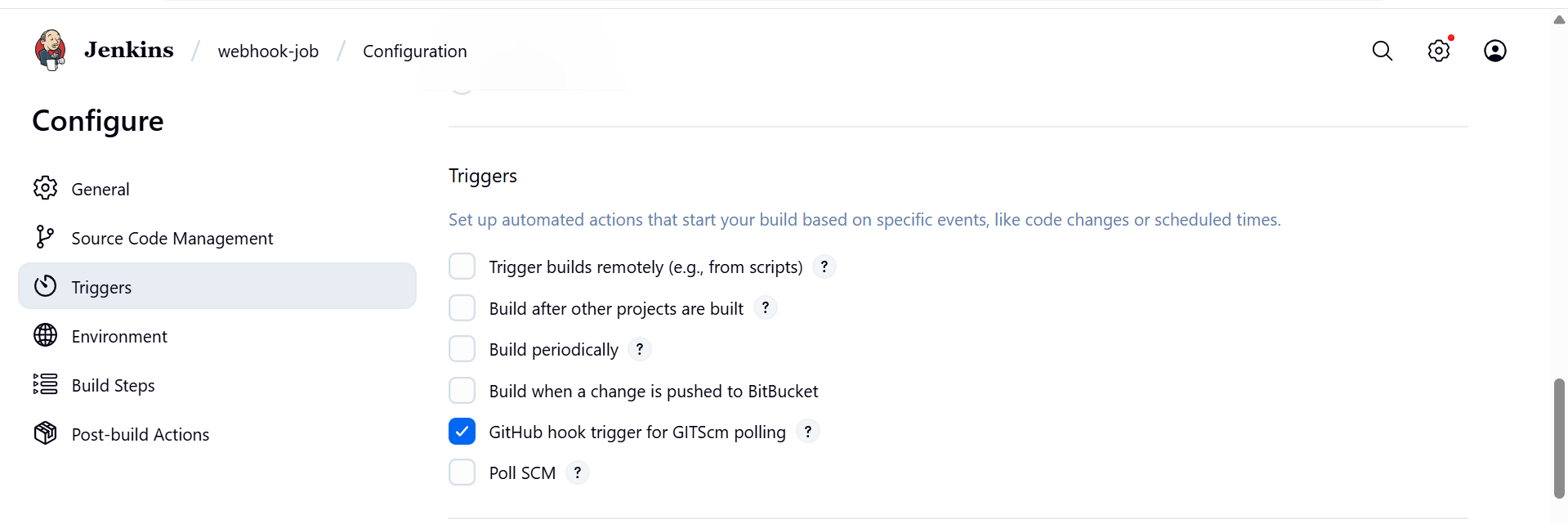


### 3. ****Enable Build Trigger****

* In your Jenkins job → **Configure** → Scroll to **Build Triggers**.
* Check:

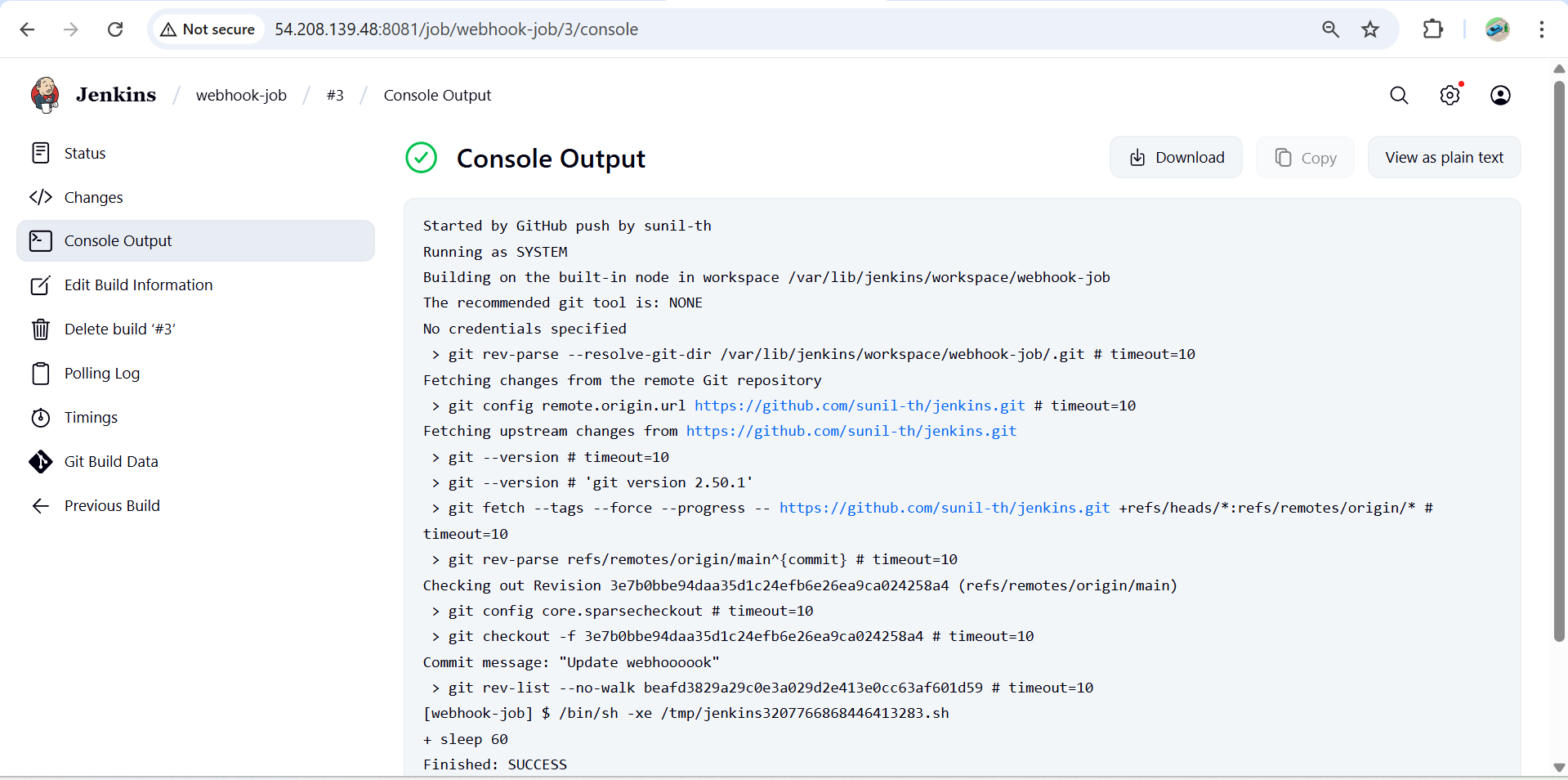
**GitHub hook trigger for GITScm polling**

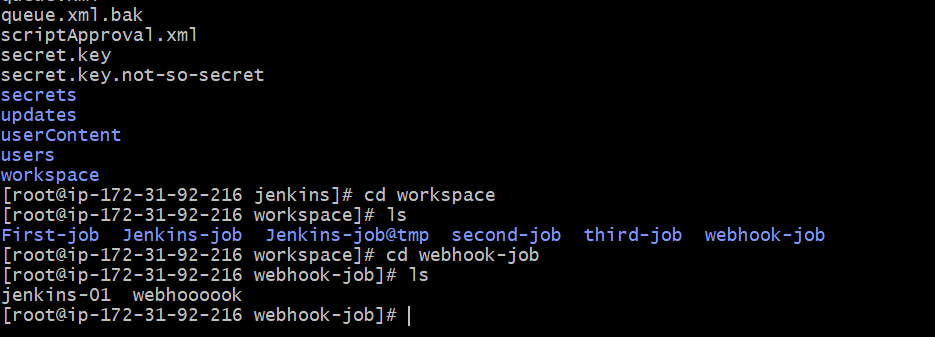
This tells Jenkins to listen for webhook events instead of polling GitHub periodically.



### ****Verify****

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





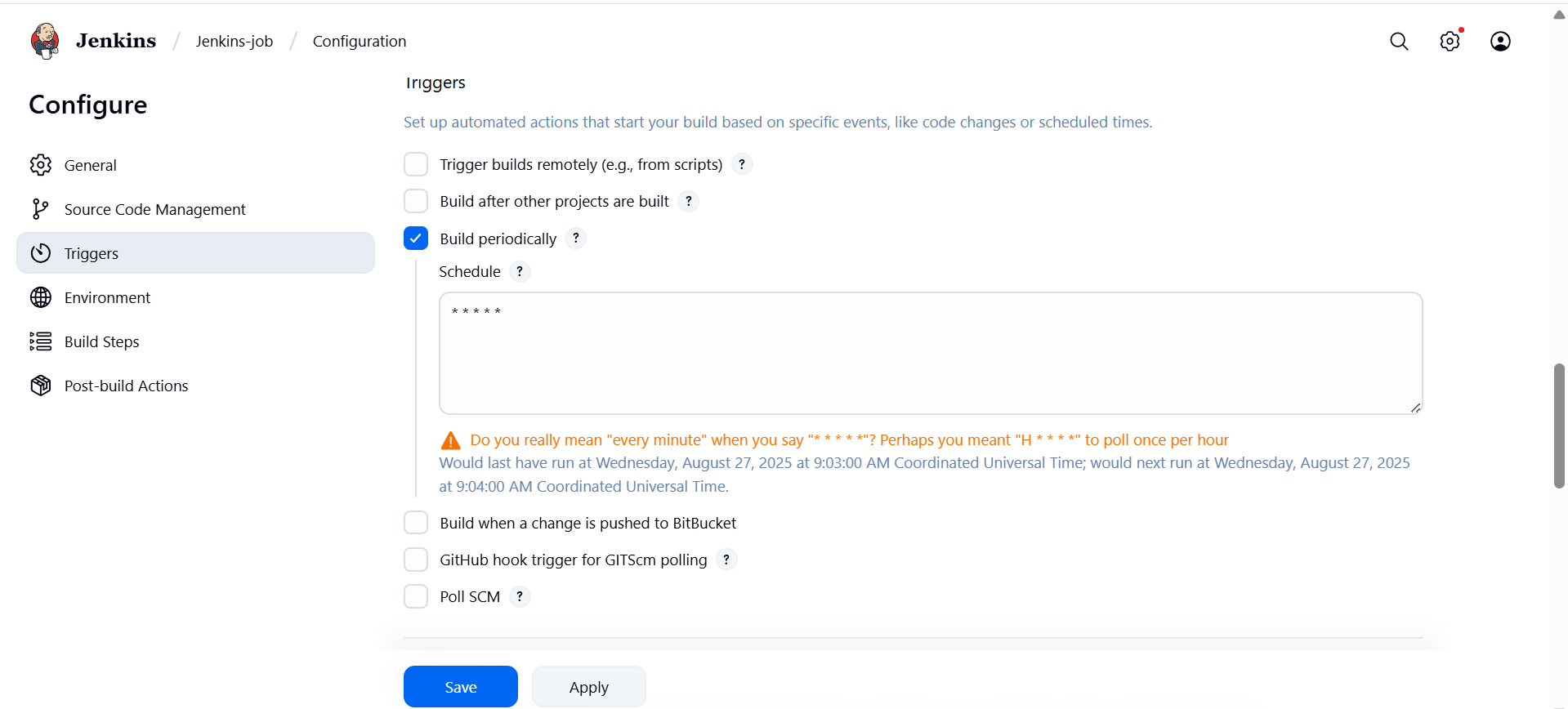
**Summary Flow:**  
GitHub → Webhook event → Jenkins /github-webhook/ endpoint → Job Trigger → Build runs

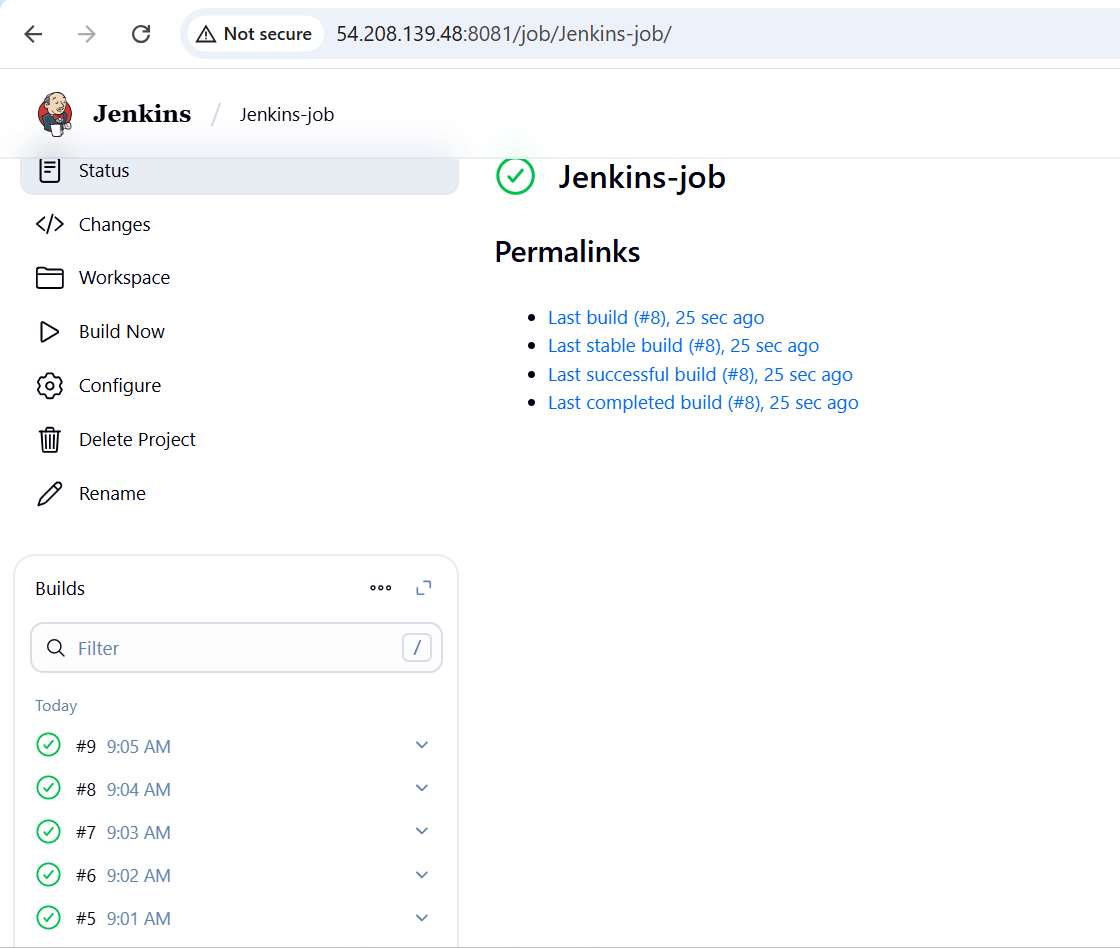
1. **Configure poll scm and build periodical options in Jenkins job.**

### *1.Build Periodically*

This option triggers builds **whether or not there are SourceCodeManagement ( scm) changes**.

* Go to your Jenkins job → **Configure**.
* Scroll down to **Build Triggers**.
* Check **Build periodically**.
* Enter a cron-style schedule in the **Schedule** box:

****

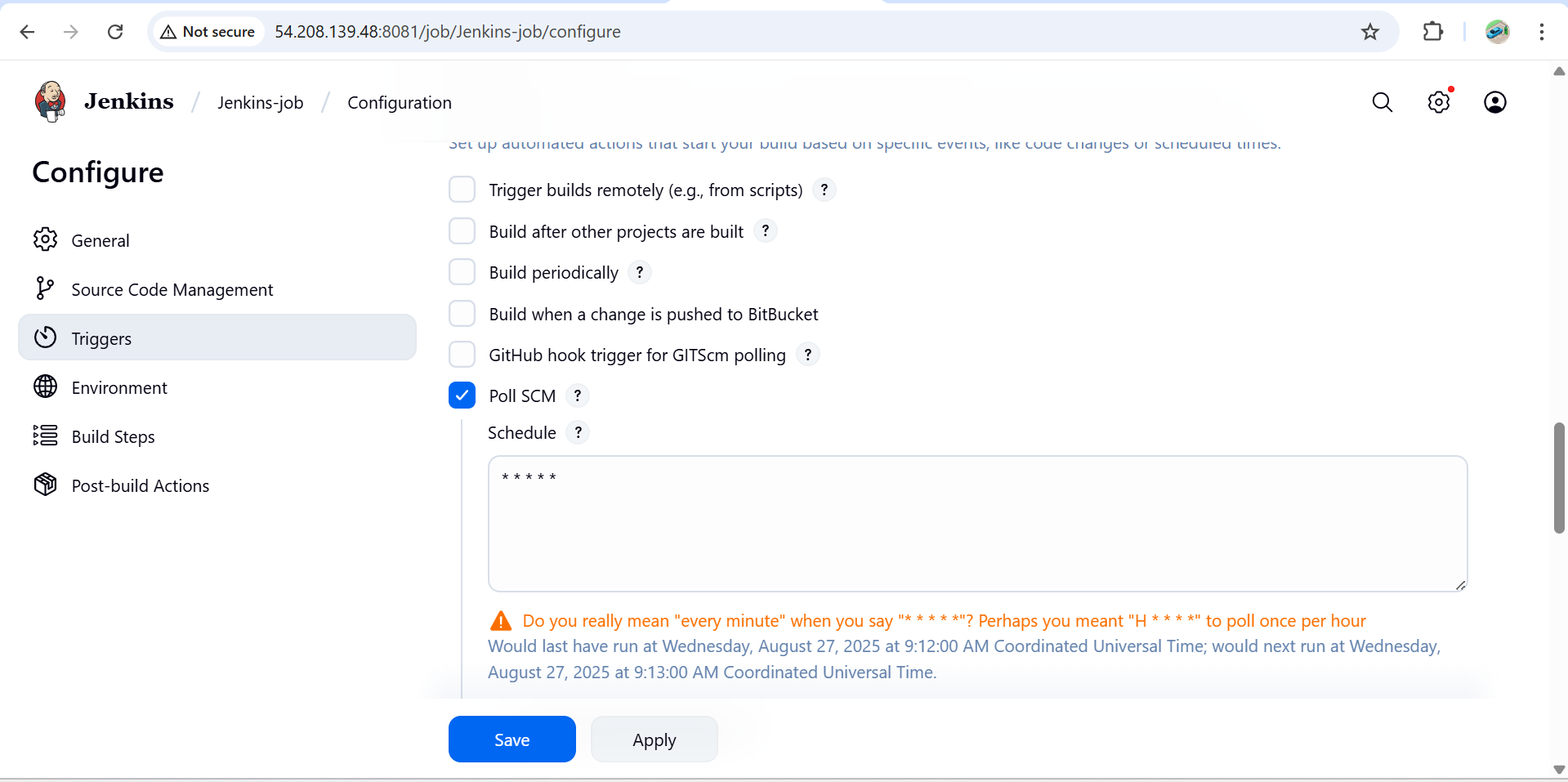
****

****

### 2. Poll SCM

This option tells Jenkins to check the source code repository (Git, SVN, etc.) at defined intervals. If it detects any change (new commit), it triggers a build.

* Go to your Jenkins job → **Configure**.
* Scroll down to **Build Triggers**.
* Check **Poll SCM**.
* In the **Schedule** box, use cron-style syntax:





**Poll SCM** → *Builds only when there are code changes.*

**Build periodically** → *Builds at scheduled times regardless of changes.*

1. **Take backup of Jenkins server by using bash script.**

Step 1: Create Backup Script

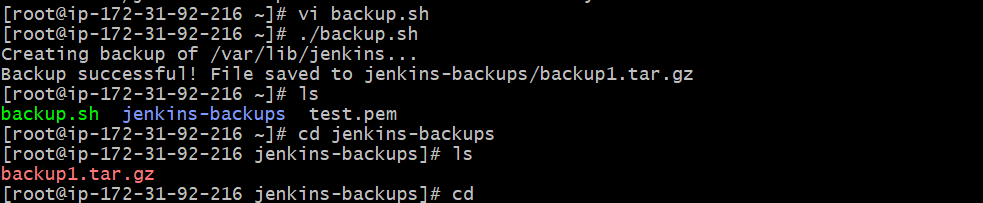
vi backup.sh

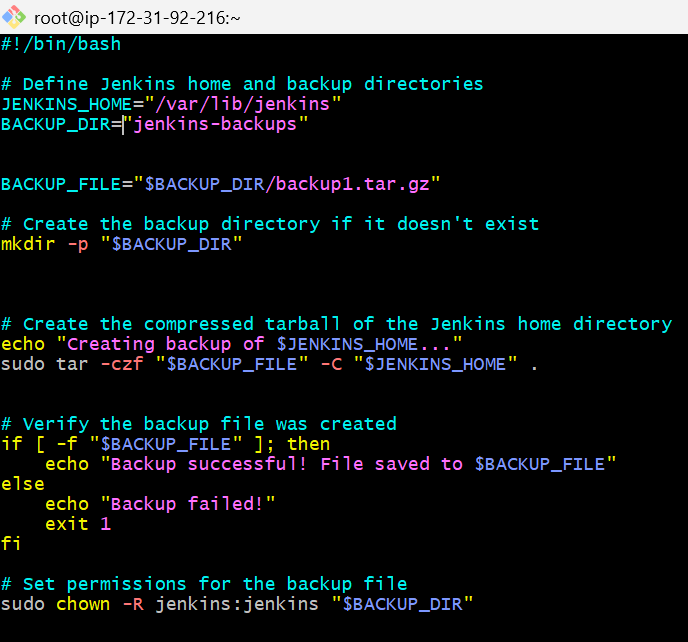
Make Script Executable

chmod +x backup.sh

Run Backup Script

./backup.sh



****

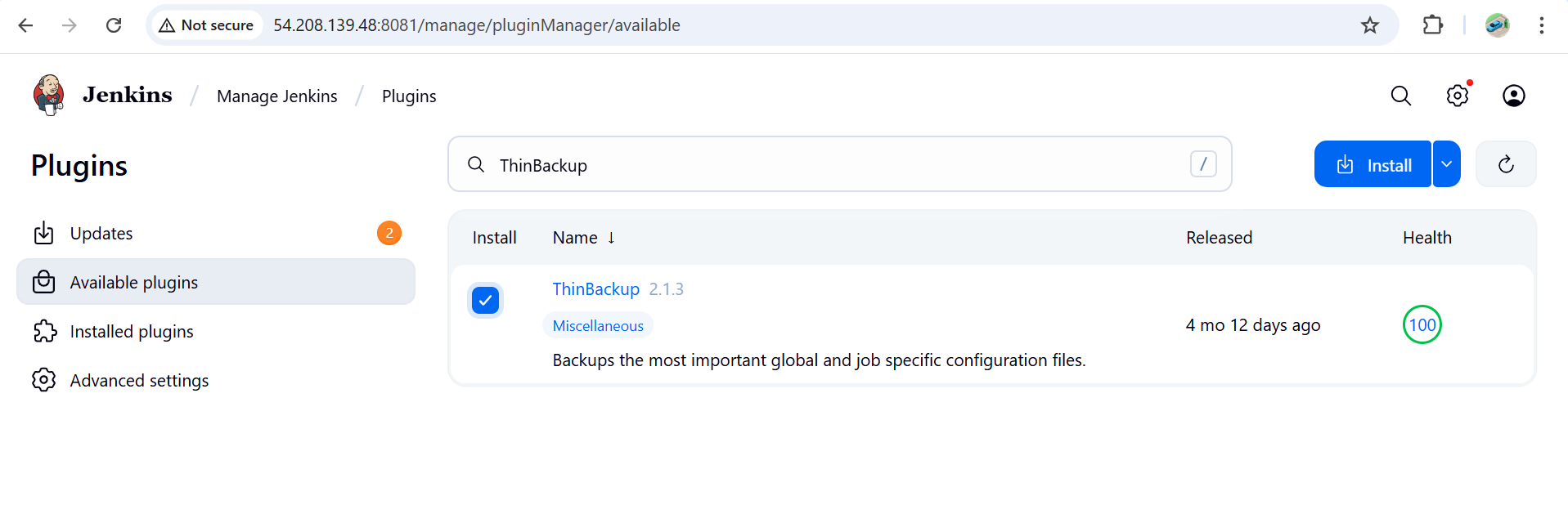
1. **Take backup of Jenkins using thin backup plugin.**

### ****Step 1: Access Jenkins****

1. Open your Jenkins URL in a browser: http://<your-server-ip>:8080.
2. Log in with an admin account.

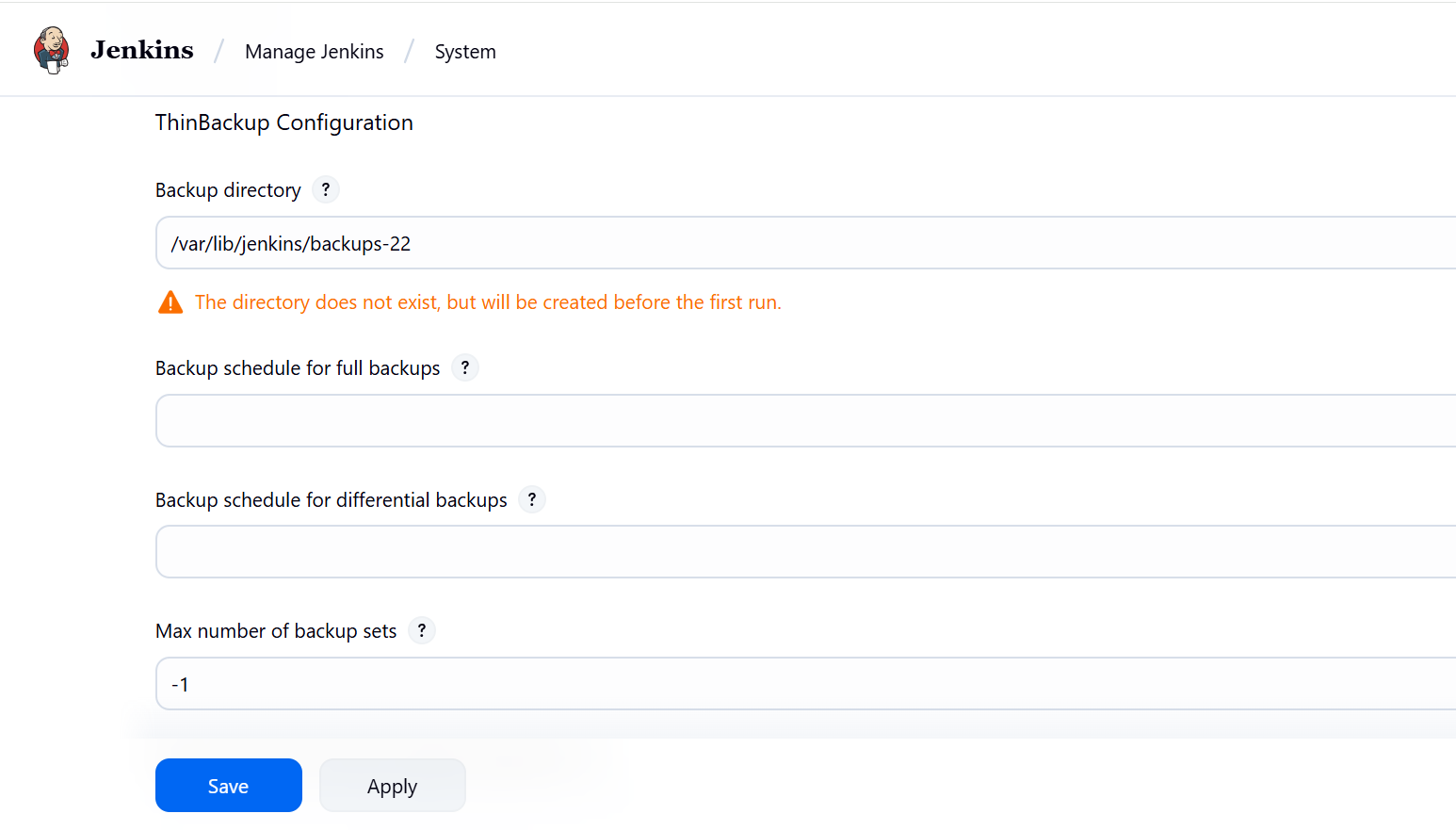
### ****Step 2: Install ThinBackup Plugin****

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**



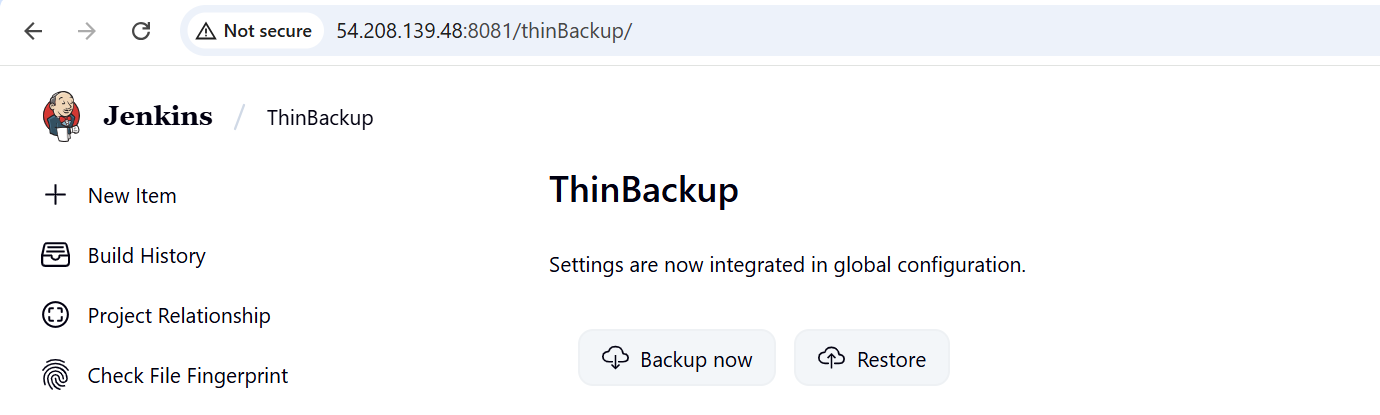
### ****Step 3: Configure ThinBackup****

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**.



### ****Step 4: Take Manual Backup****

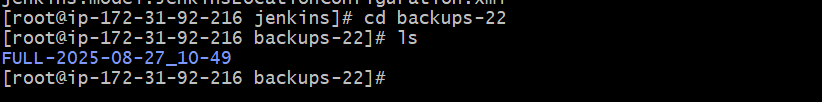
1. Go to **Manage Jenkins** → **ThinBackup**.
2. Click **Backup Now**.
3. The backup will be created in the folder you configured (/var/Jenkins/backups-22).



### ****Step 5: Verify Backup****

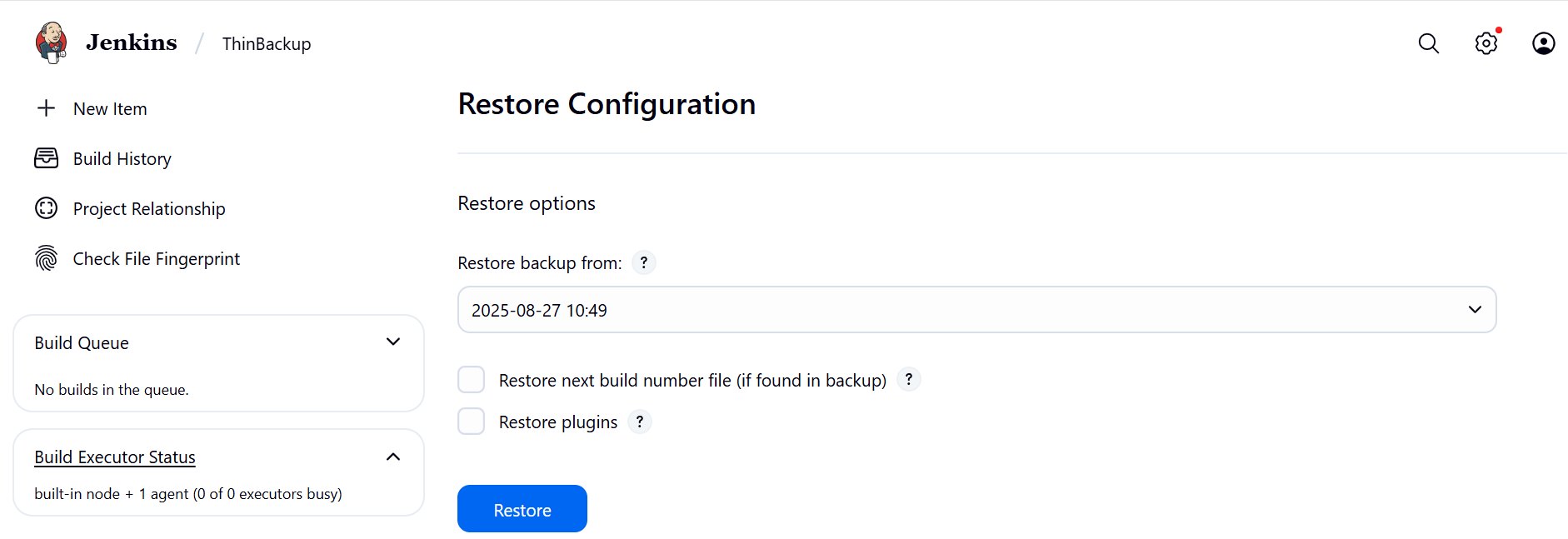
1. Navigate to the backup folder.





### ****Step 6: Restore Backup (Optional)****

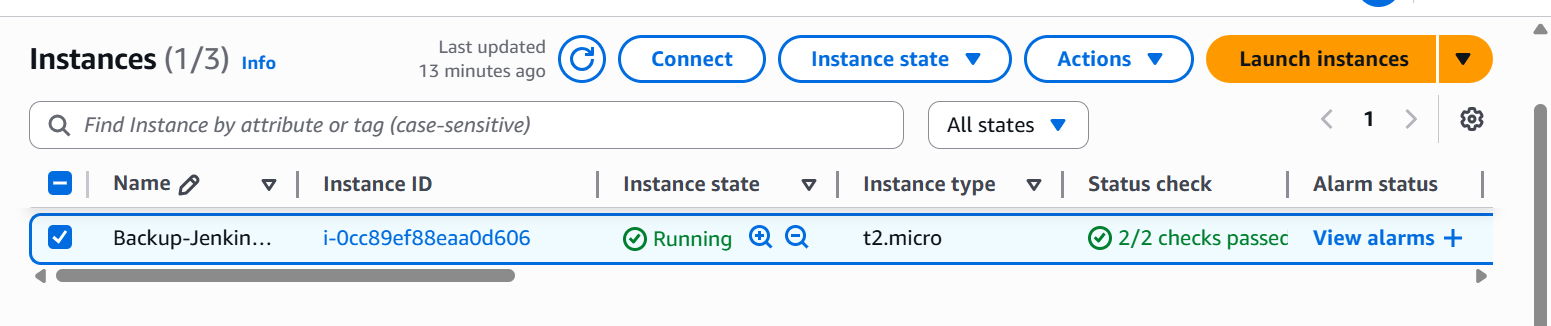
1. Go to **Manage Jenkins** → **ThinBackup**.
2. Click **Restore**.
3. Select the backup you want to restore.
4. Click **Restore** to bring Jenkins back to that state.



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

# Step 1: Launch New Server & Install Jenkins

1. **Launch a new EC2 instance**

****

1. Install Java (required for Jenkins)

sudo dnf install -y java-17-amazon-corretto

1. Add Jenkins Repository

**sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo**

**sudo rpm --import** <https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key>

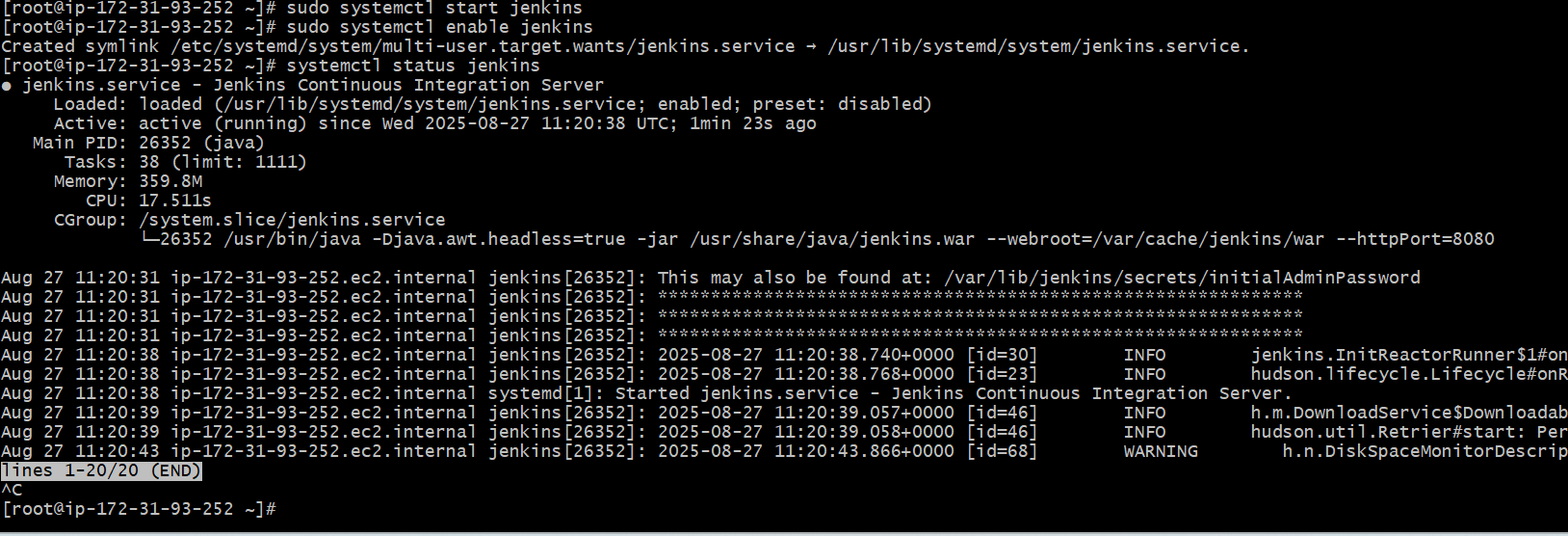
1. Install Jenkins

**sudo yum install jenkins –y**

**systemctl enable jenkins**

**systemctl start Jenkins**

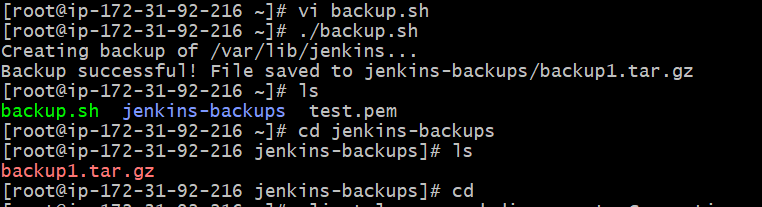
**systemctl status jenkins**

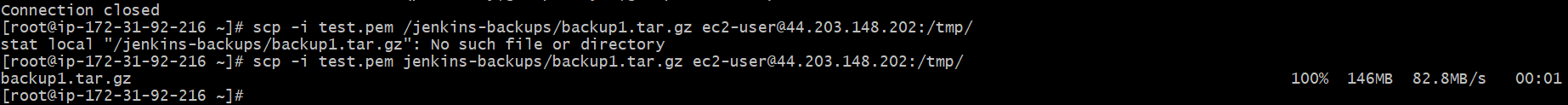


Step 2: Copy Your Backup to New Server

Login into your old main server you took backup

Create an test.pem file where you copy your pemkey of new-server and and paste in the oldserver





[root@ip-172-31-92-216 ~]# scp -i test.pem jenkins-backups/backup1.tar.gz [ec2-user@44.203.148.202:/tmp/](mailto:ec2-user@44.203.148.202:/tmp/)

test.pem is your new server pemkey

jenkins-backups - is the folder containing the backup tar file

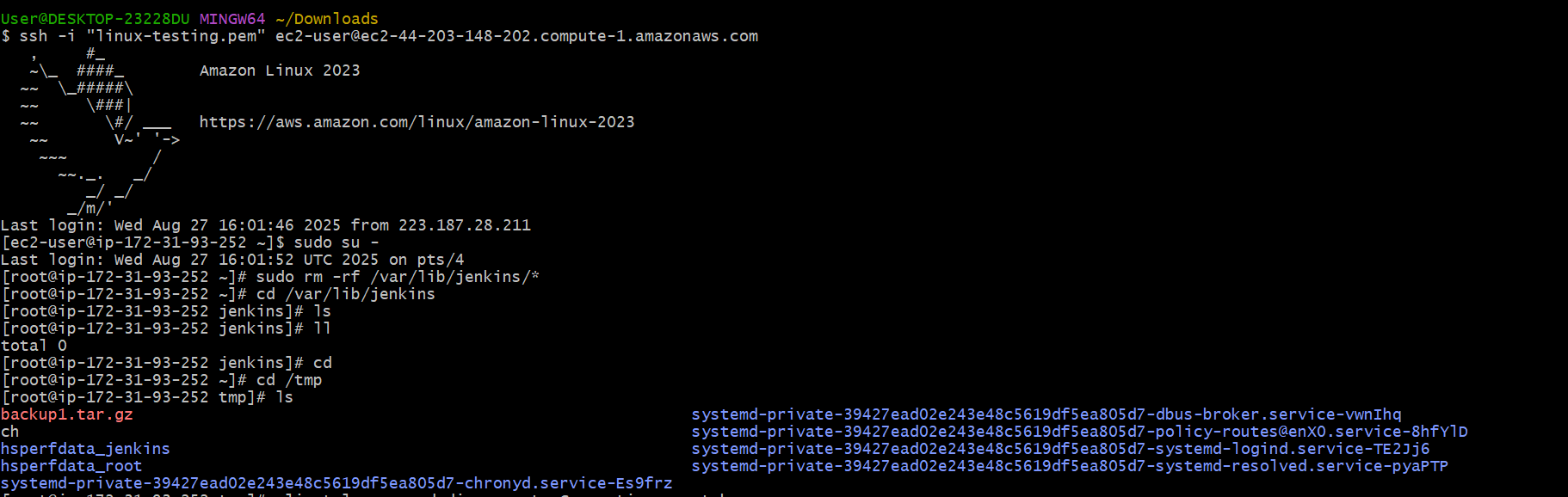
backup1.tar.gz -backup file

[ec2-user@44.203.148.202](mailto:ec2-user@44.203.148.202) - public ip of new server

/tmp – is the location in new server where you send your backup tar file

**Step 3:** Login in to new server and check the file in /tmp

backup1.tar.gz



Remove existing Jenkins data. Delete the contents of the new $JENKINS\_HOME to prepare for the restore.

sudo rm -rf /var/lib/jenkins/\*

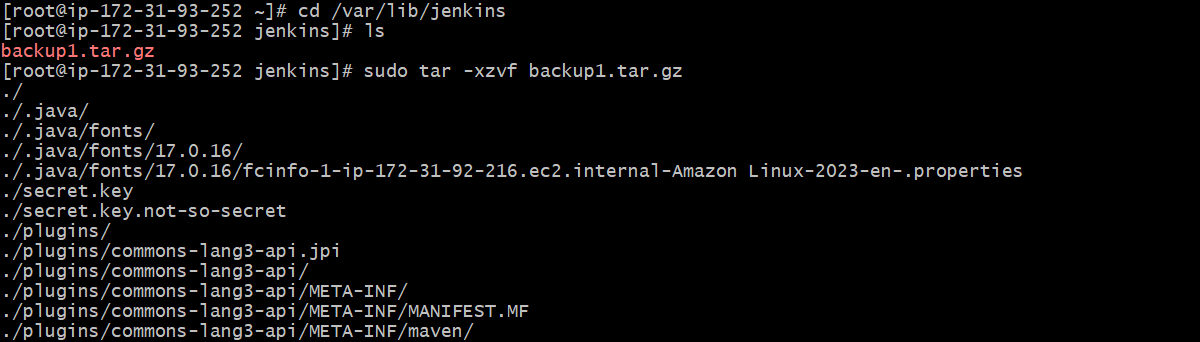
Copy the file from tmp to /var/lib/jenkins/

cp /tmp/backup1.tar.gz /var/lib/jenkins/

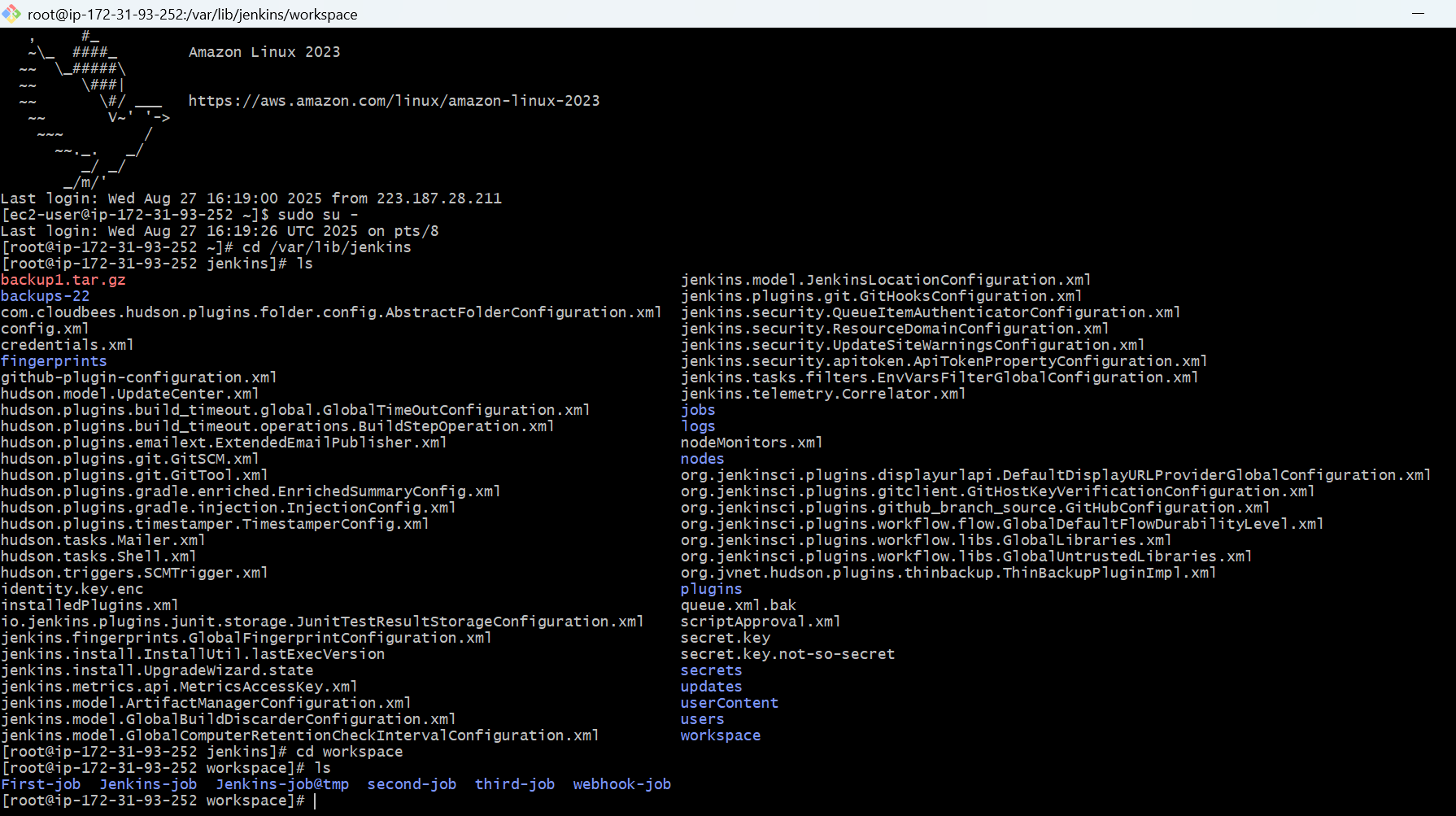
then extracts

cd /var/lib/jenkins

sudo tar -xzvf backup1.tar.gz



(Or you can directly extract from tmp by using sudo tar -xzf /tmp/backup1.tar.gz –C /var/lib/Jenkins)



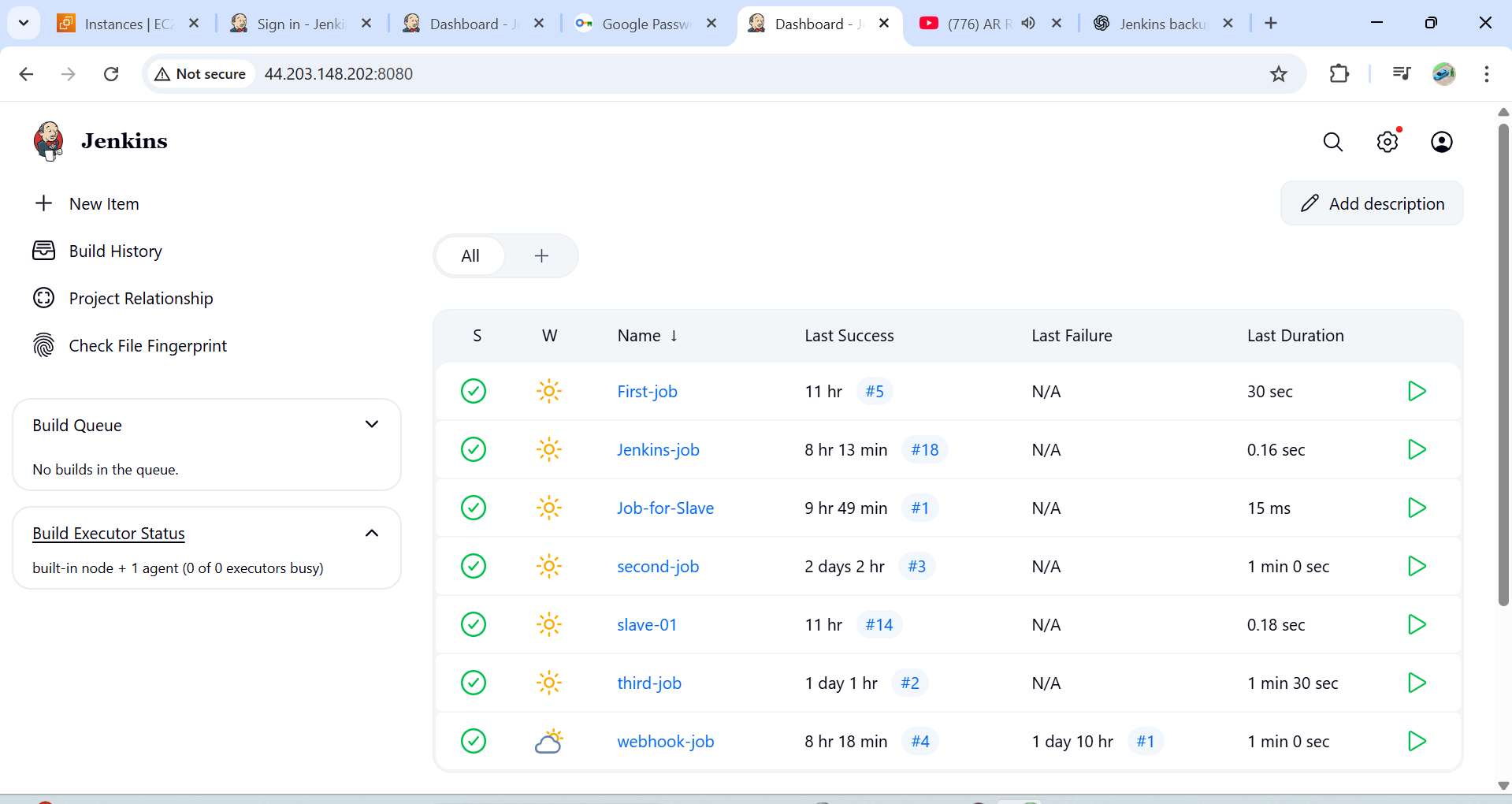
sudo systemctl start jenkins

sudo systemctl status Jenkins

Verify in browser

http://<EC2-Public-IP>:8080

new server by default running on port no-: 8080



Old server running on port –no. :- 8081

