# JENKINS -02

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

**Project Report: Jenkins Distributed Architecture, Automation, and Disaster Recovery**

**1. Objective**

The goal of this project was to establish a robust, scalable CI/CD environment. This involved configuring a **Jenkins Master-Slave architecture** via SSH to offload build execution, implementing **Automation Triggers** (GitHub Webhooks and Polling) for hands-free builds, and creating a **Disaster Recovery Strategy** using both manual shell scripts and the ThinBackup plugin to ensure data persistence and portability.

**2. Prerequisites**

* **Infrastructure:** 4 EC2 Instances (Amazon Linux 2023) – Master, Slave-01, Slave-02, and a Backup/Recovery Server.
* **Access:** SSH access to all instances and Security Groups allowing Port 8080 (Jenkins) and Port 22 (SSH).
* **Software:** Java 17 (Amazon Corretto) installed on all nodes.
* **SCM:** A GitHub repository for testing CI/CD triggers.

**4. Errors Encountered & Fixes**

| **Error / Issue** | **Root Cause** | **Resolution** |
| --- | --- | --- |
| **Agent Offline (Permission Denied)** | Jenkins tried to use /root as the Remote Root Directory, which lacked permissions. | Created /home/ec2-user/jenkins, changed ownership to ec2-user, and updated the node config. |
| **Host Key Verification Failed** | The Master did not recognize the Slave's SSH fingerprint. | Used ssh-keyscan -H <Slave-IP> >> known\_hosts and fixed permissions to 600. |
| **Webhook 403 / No Trigger** | Jenkins Security/CSRF or incorrect Payload URL. | Verified the URL ended in /github-webhook/ and ensured the "Active" status in GitHub settings. |
| **Backup Transfer Permission** | The backup file in /opt was not readable by the ec2-user for SCP. | Applied chmod 600 or 644 to the backup file to allow the transfer to the new server. |

**5. Conclusion**

The project successfully demonstrates a production-ready Jenkins environment. By offloading builds to **Slaves**, the Master remains responsive. The integration of **Webhooks** ensures a true DevOps "Push-to-Deploy" workflow. Finally, the **Restore** process confirms that the system is resilient against server failure, allowing for a full recovery of the CI/CD pipeline in minutes

1.Configure 2 slave machines in Jenkins master.

**EC2 & Initial Setup**

1. Launched three EC2 instances:
   * jenkins-master
   * slave-01
   * slave-02

**Slave-01 OS Configuration**

Logged in to slave-01 as root.

Installed Java:

yum install java-17-amazon-corretto -y

Verified Java installation completed.

**SSH Configuration on Slave-01**

Generated SSH key pair:

ssh-keygen

Navigated to SSH directory:

cd ~/.ssh

Added public key to authorized keys:

cat id\_rsa.pub >> authorized\_keys

Fixed permissions:

chmod 700 authorized\_keys

**Jenkins Master SSH Trust**

Logged in to jenkins-master.

Added slave host key:

ssh-keyscan -H 13.61.10.105 >> /var/lib/jenkins/.ssh/known\_hosts

Set correct permissions:

chown jenkins:jenkins /var/lib/jenkins/.ssh/known\_hosts

chmod 700 /var/lib/jenkins/.ssh

chmod 600 /var/lib/jenkins/.ssh/known\_hosts

**Jenkins Node Creation (slave-01)**

Opened Jenkins dashboard.

Navigated to **Manage Jenkins → Nodes**.

Clicked **New Node**.

Entered node name: slave-01.

Selected **Permanent Agent** and created the node.

**Node Configuration (slave-01)**

Set **Number of executors** to 2.

Initially observed agent offline due to root directory permission issue.

Logged in to slave-01 and created Jenkins workspace:

mkdir -p /home/ec2-user/jenkins

chown -R ec2-user:ec2-user /home/ec2-user/jenkins

Updated **Remote root directory** in Jenkins to:

/home/ec2-user/jenkins

Added **Label**:

java

Set **Usage** to:

Only build jobs with label expressions matching this node

**SSH Launch Configuration**

Selected **Launch agents via SSH**.

Entered **Host**: 13.61.10.105.

Selected SSH **credentials (ec2-user)**.

Chose **Known hosts file Verification Strategy**.

Saved configuration.

Verified SSH authentication successful in node logs.

Confirmed slave-01 status changed to **online**.

**Job Creation & Execution**

Clicked **New Item**.

Entered job name: first\_job.

Selected **Freestyle project**.

Enabled **Restrict where this project can be run**.

Entered label:

java

Configured **Source Code Management → Git**.

Entered repository URL:

https://github.com/betawins/Techie\_horizon\_Login\_app.git

Saved the job configuration.

Ran the job.

Verified build executed **remotely on slave-01**.

Confirmed **Git clone successful**.

Build finished with **SUCCESS**.

**Final Node Verification**

Navigated to **Manage Jenkins → Nodes**.

Verified:

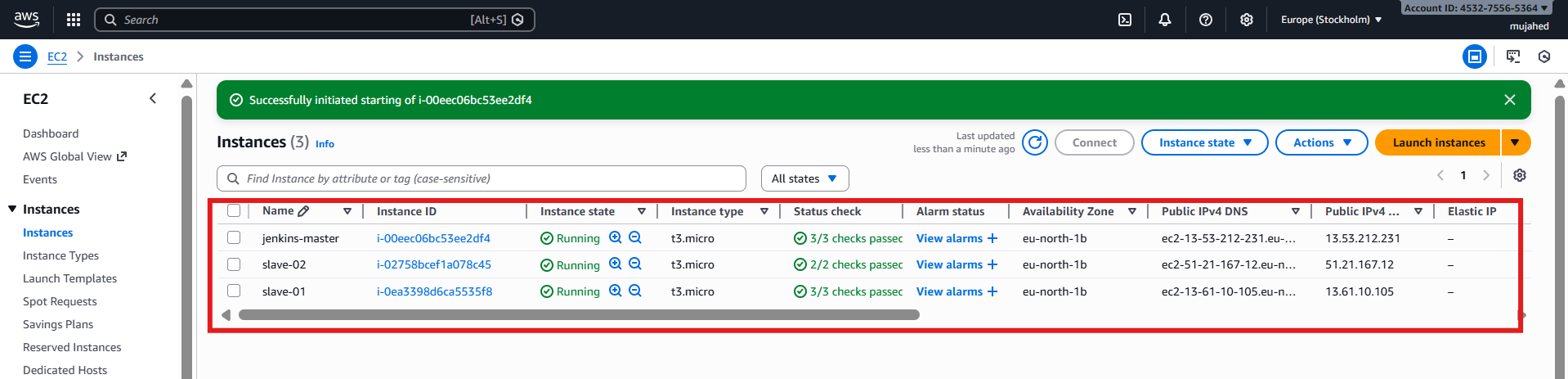
* + Built-in node → online
  + slave-01 → online with disk & response metrics visible

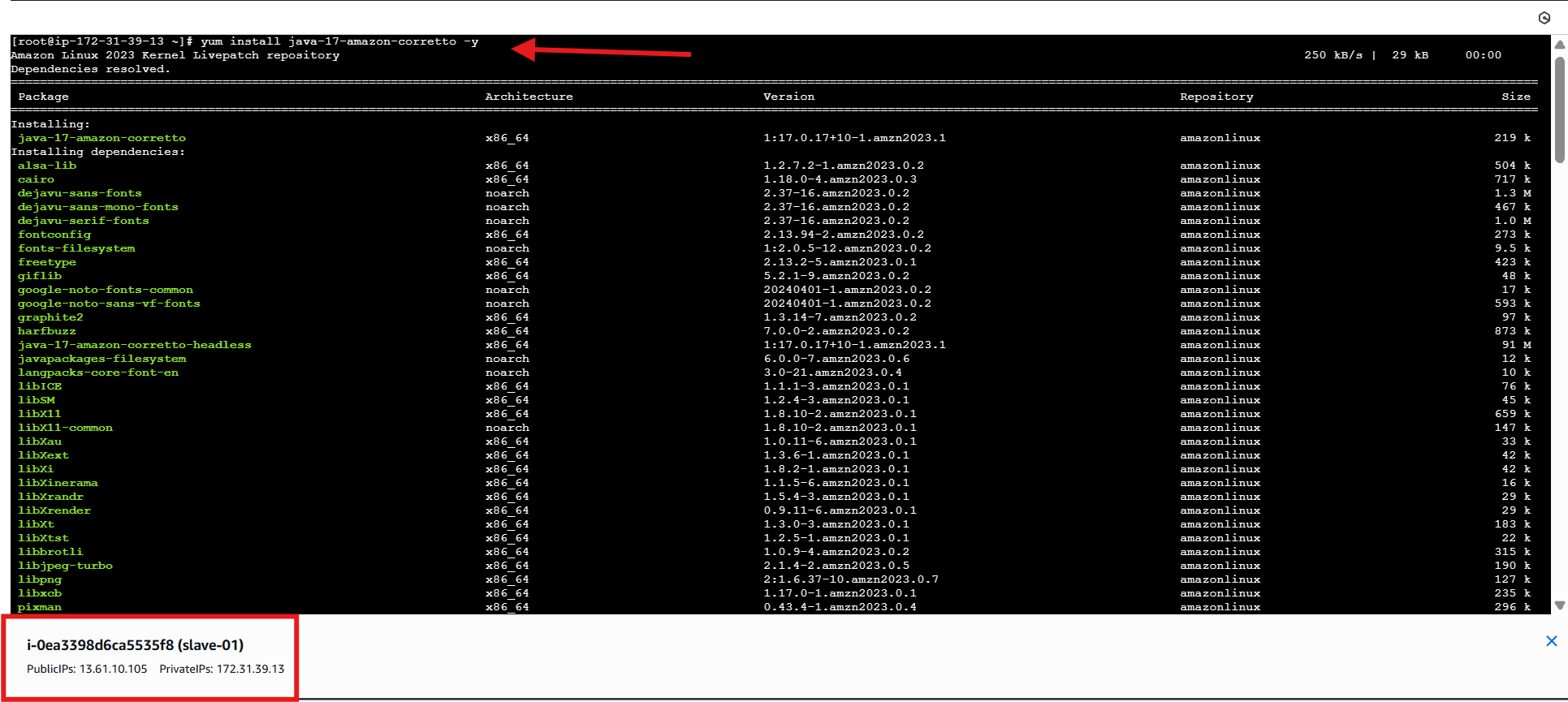
**Slave-02 Configuration**

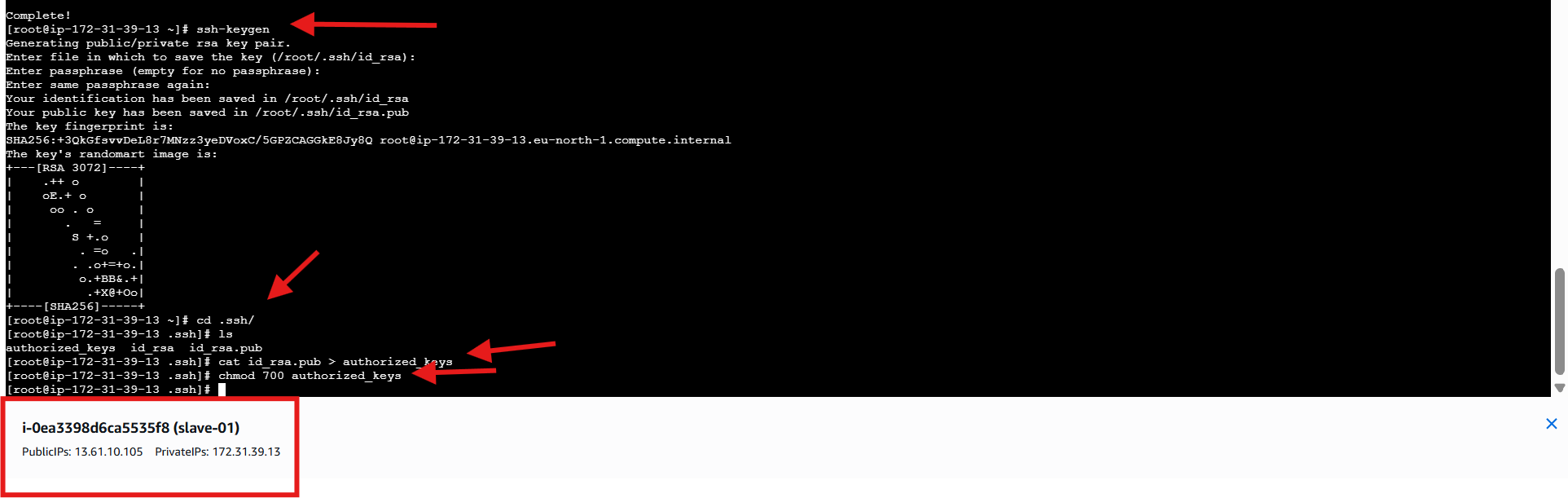
Repeated **the same configuration steps** (Java install, SSH setup, workspace creation, Jenkins node setup) for **slave-02**.

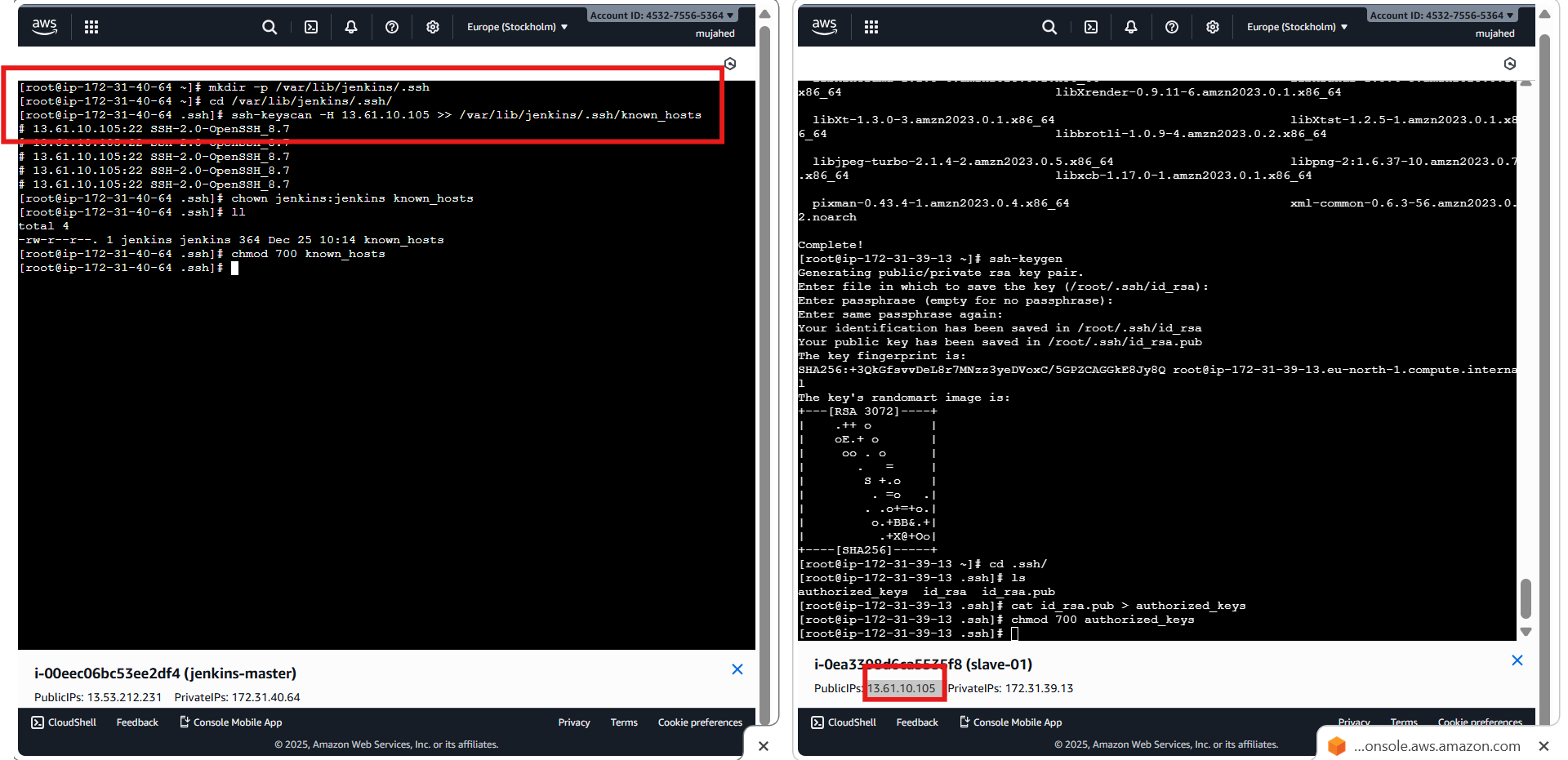
Added slave-02 successfully to Jenkins.

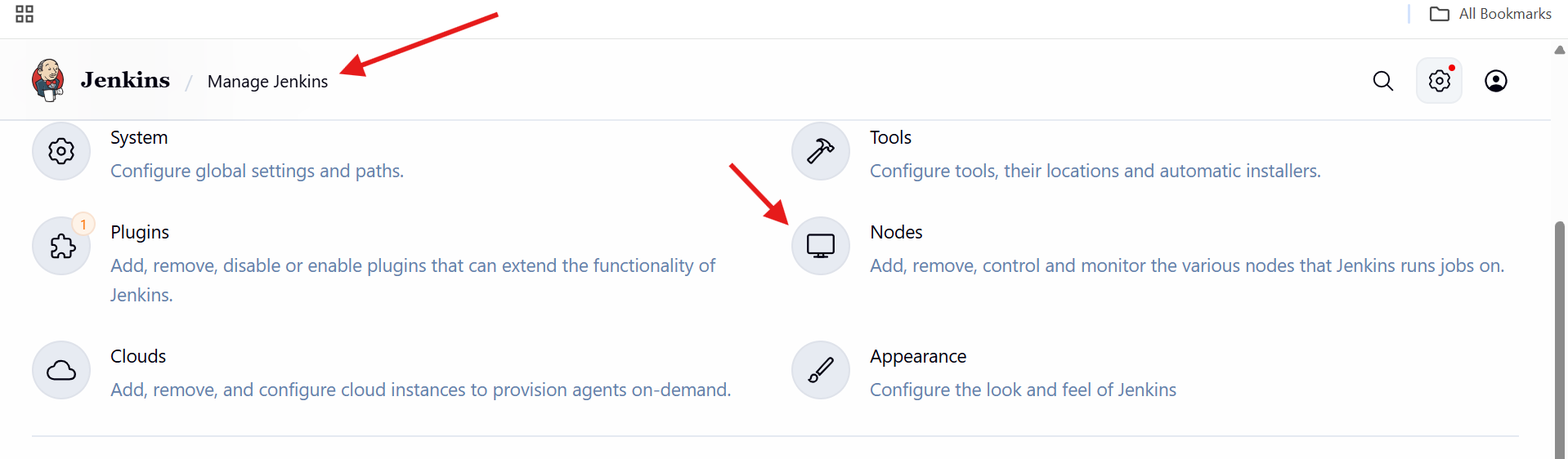
Verified both **slave-01 and slave-02** are **online and available for builds**.

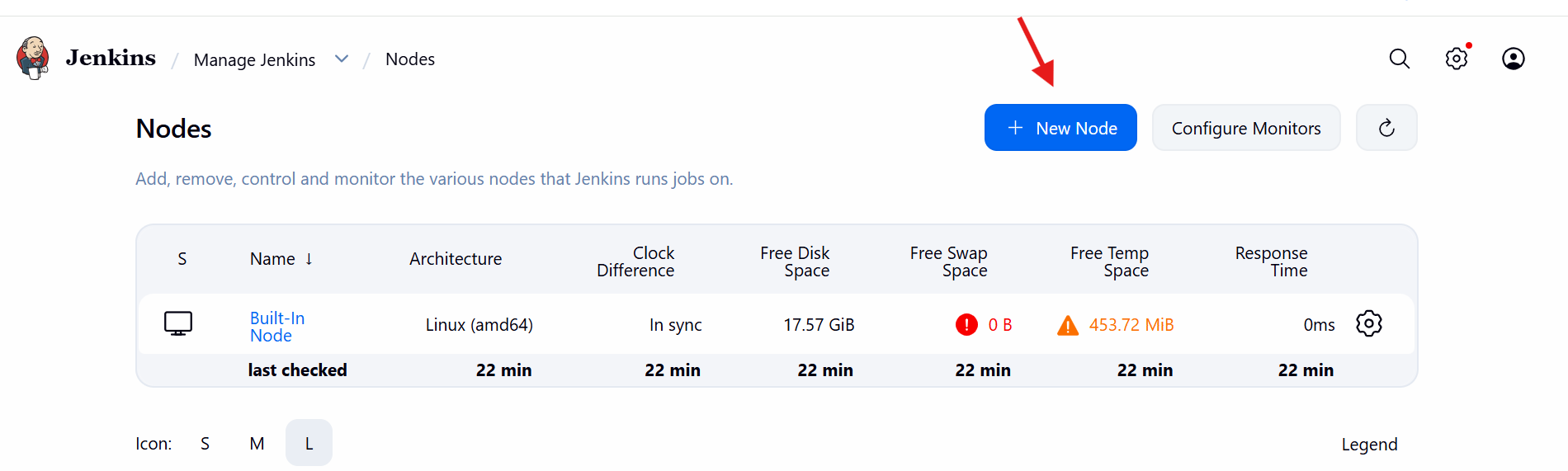


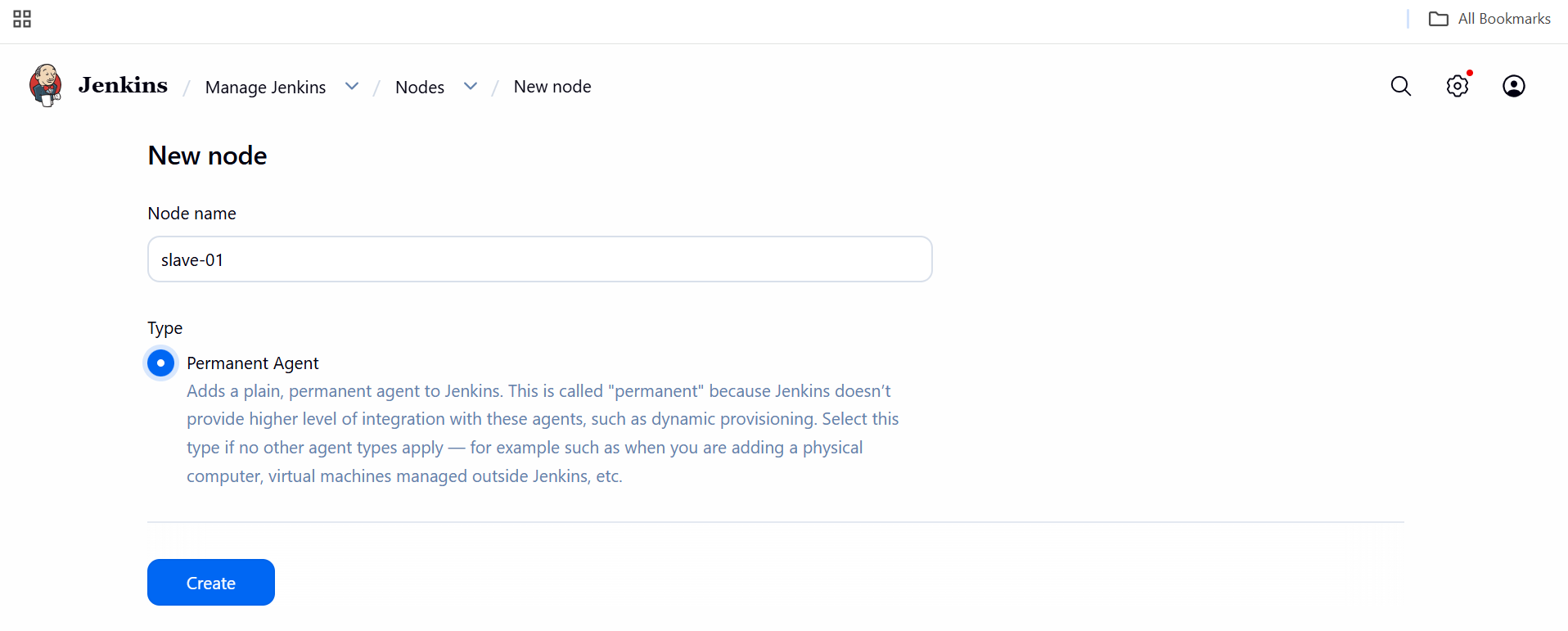


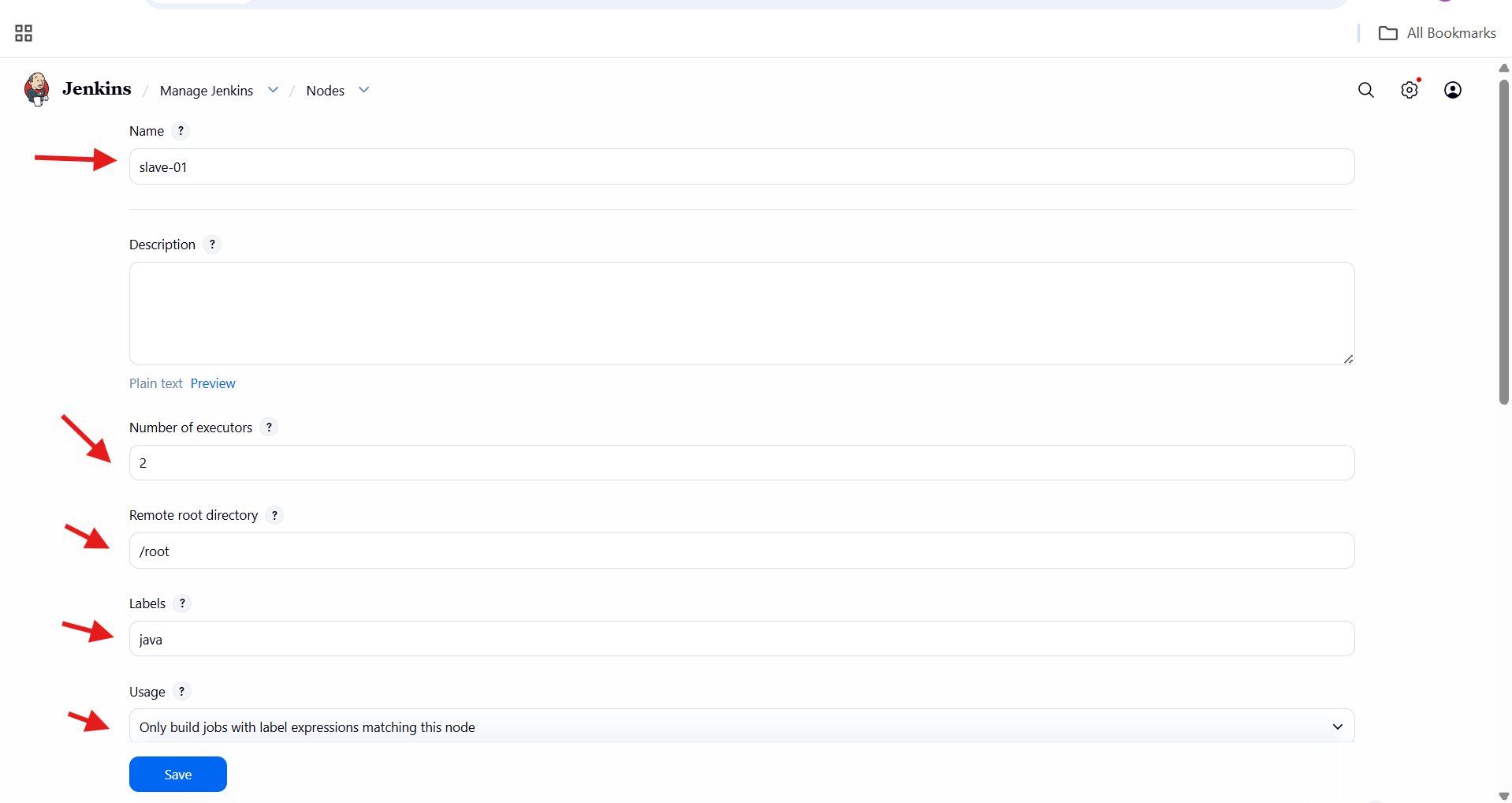


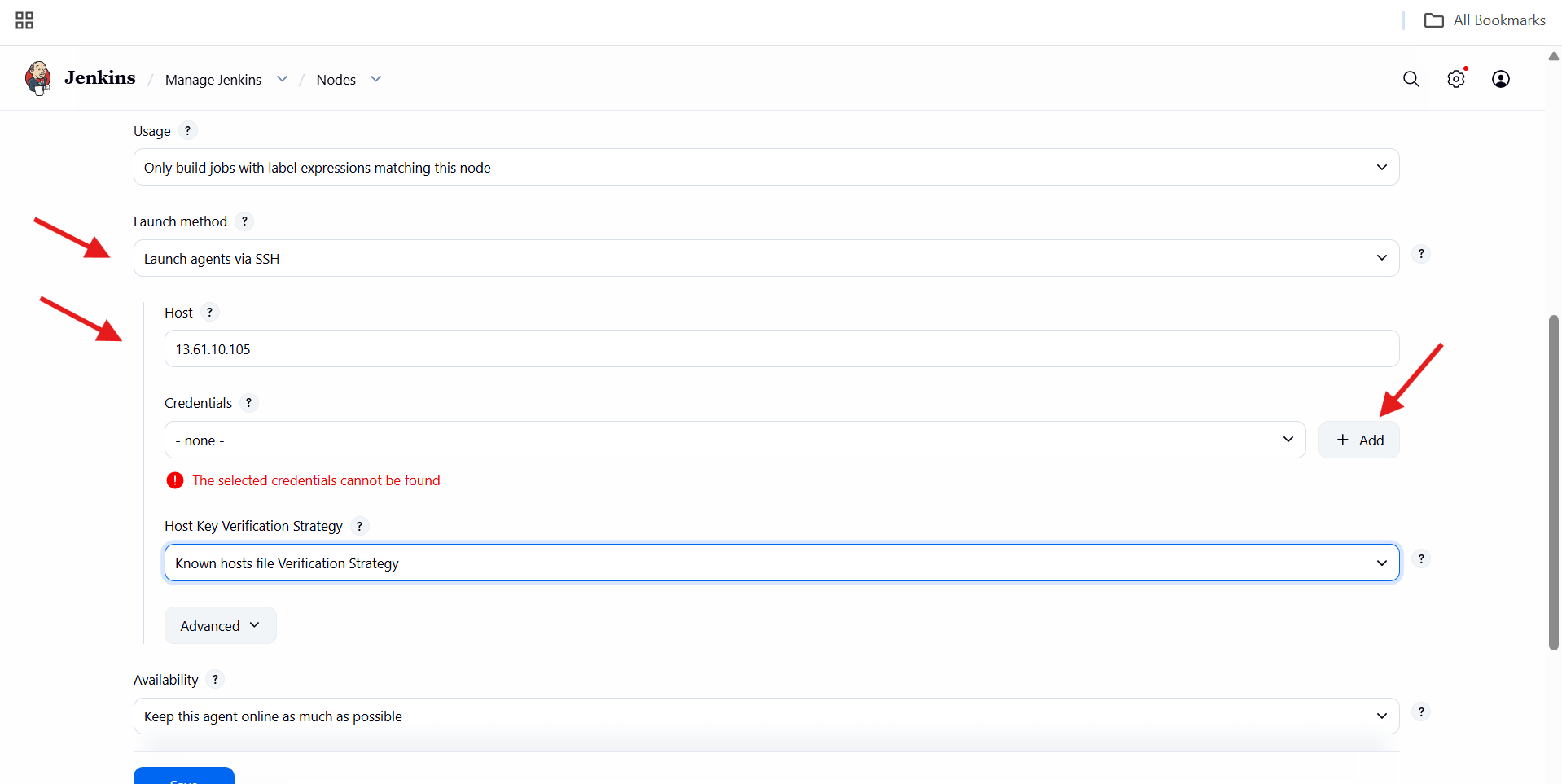


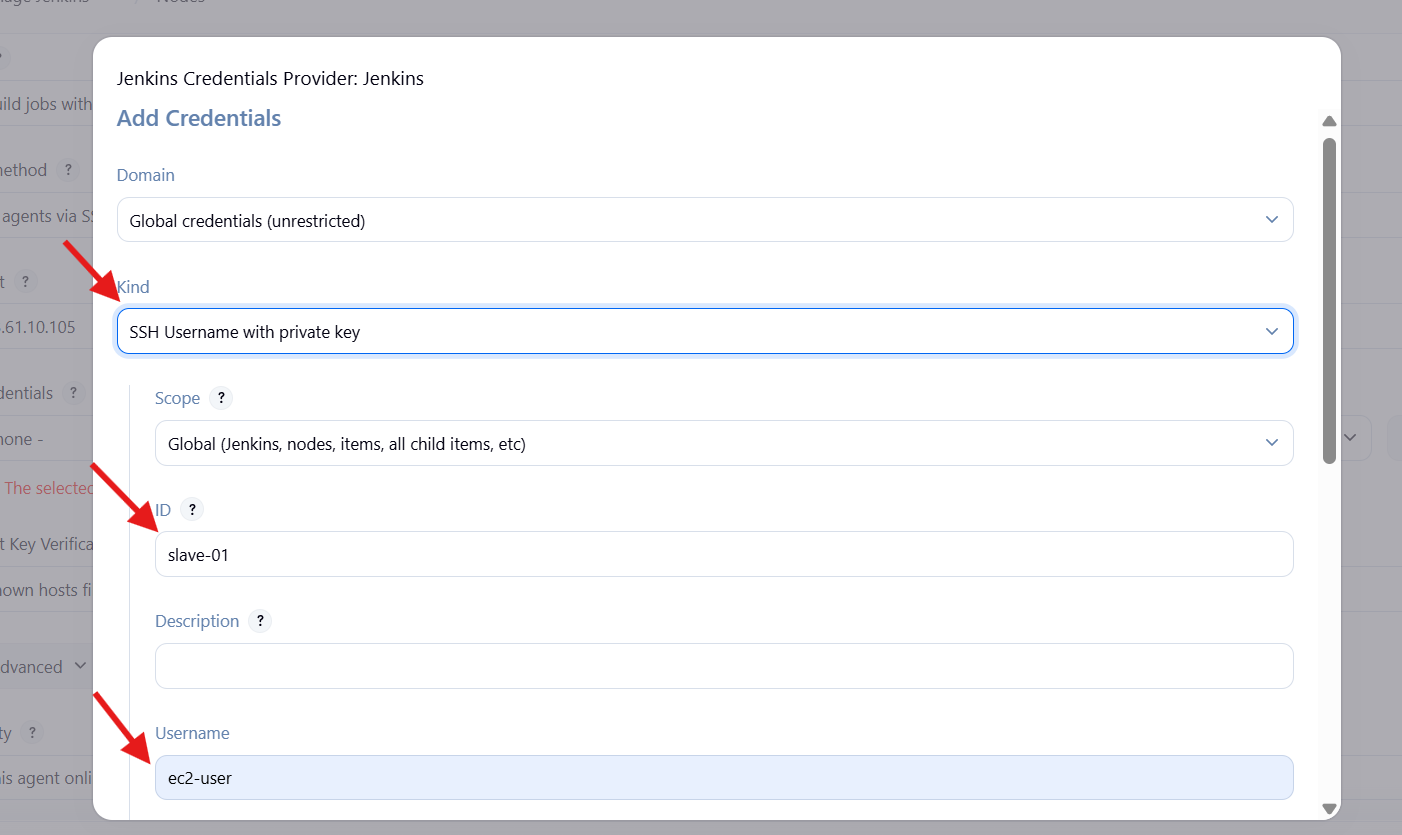


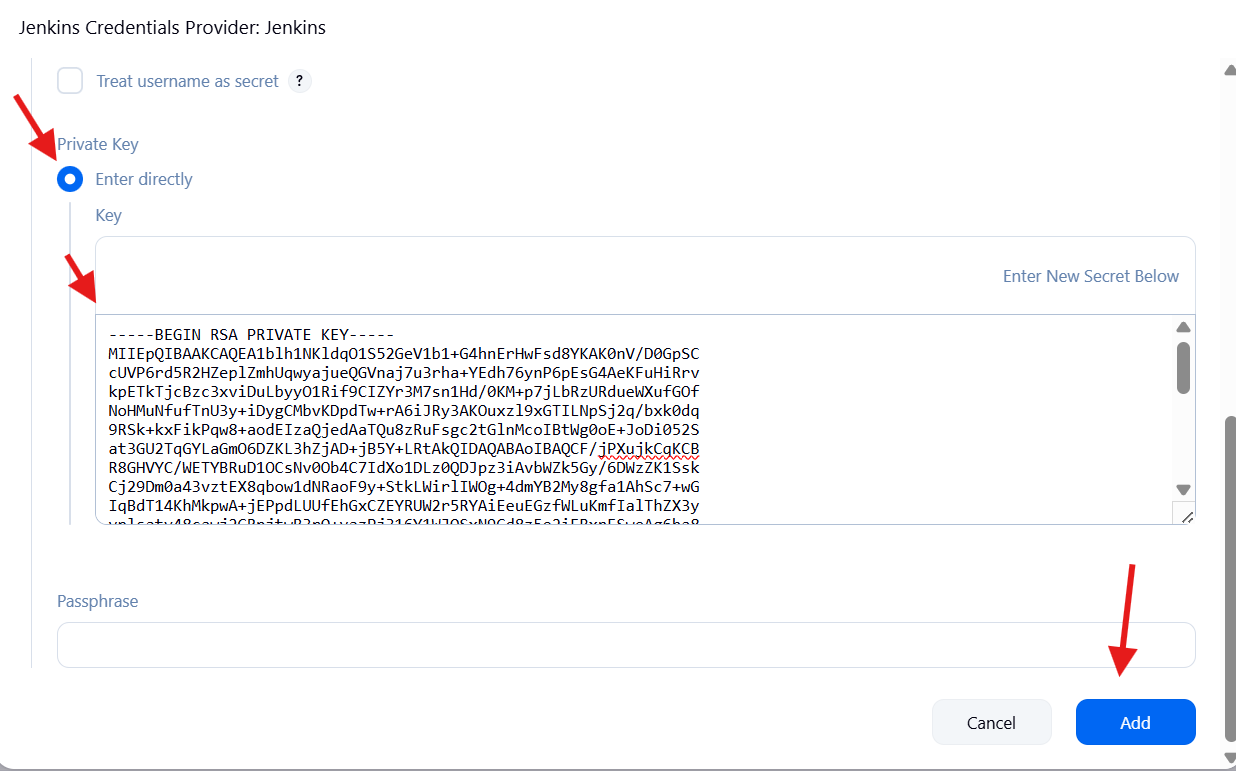


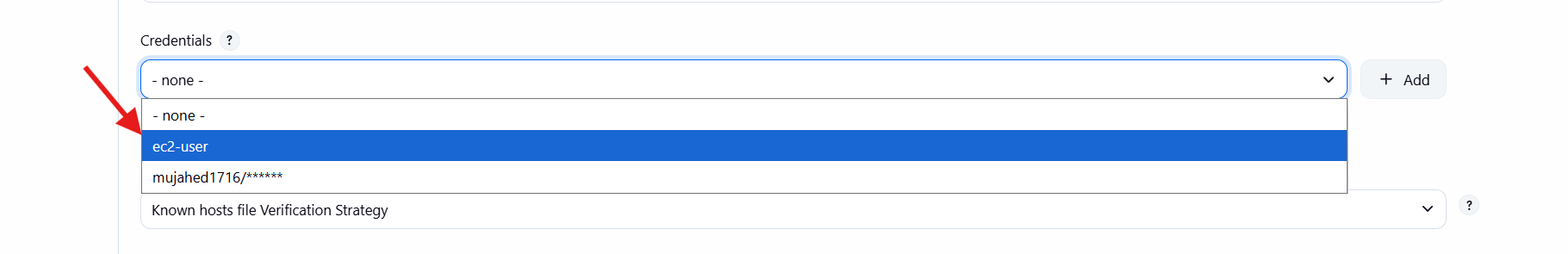


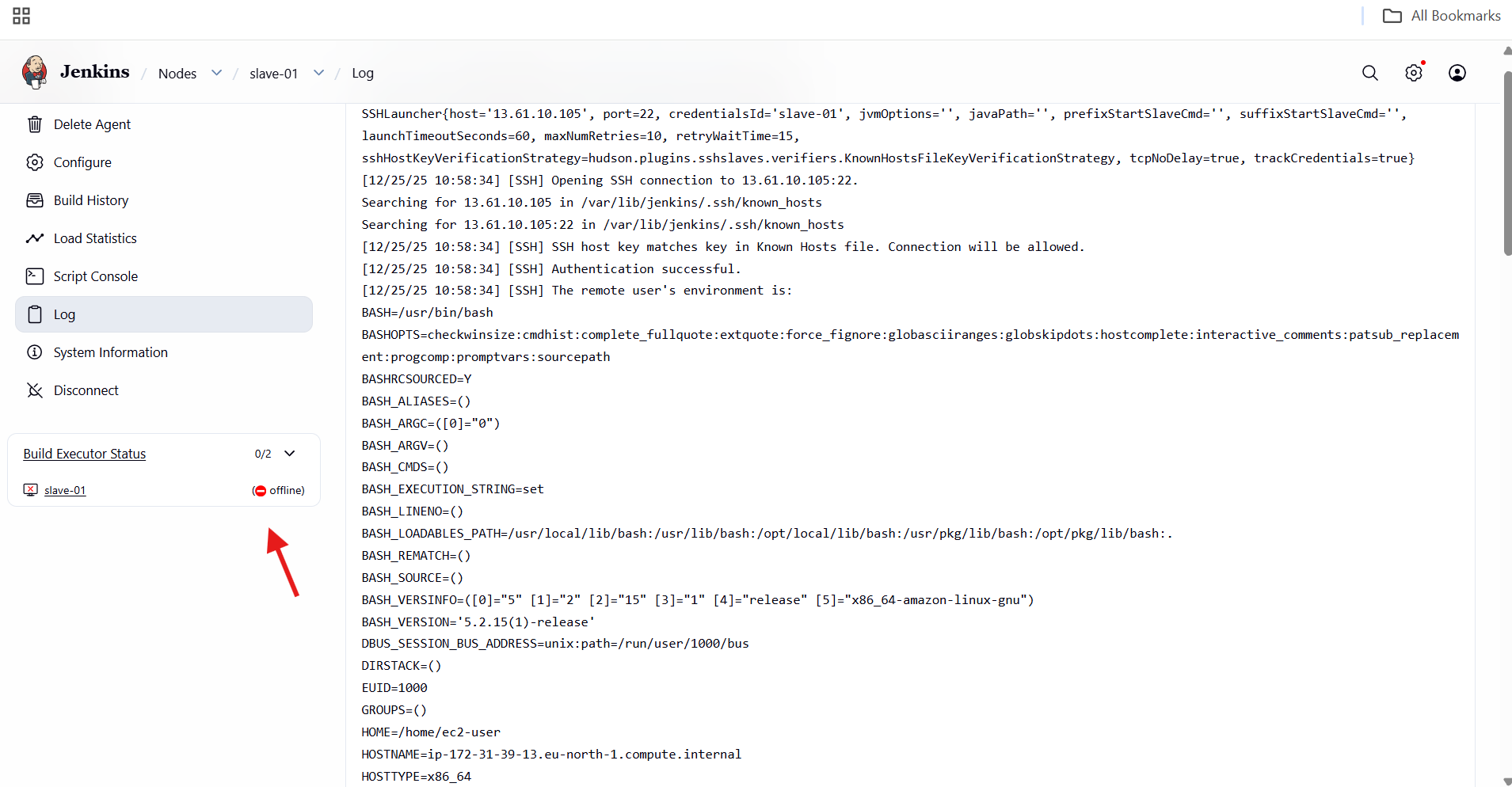


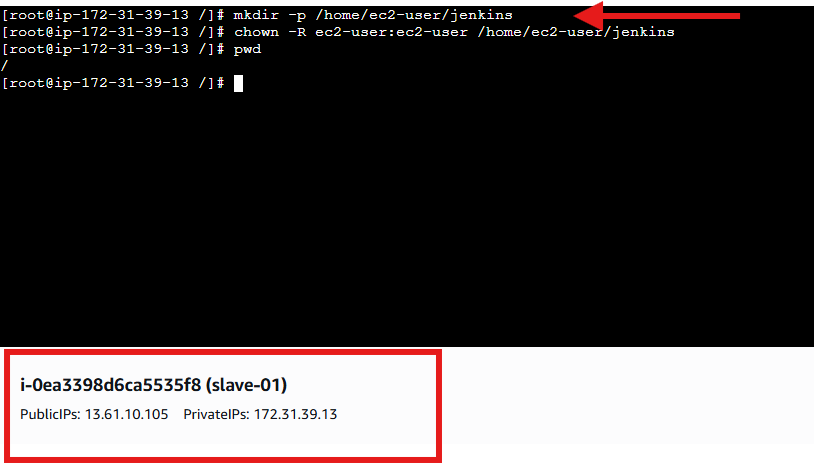


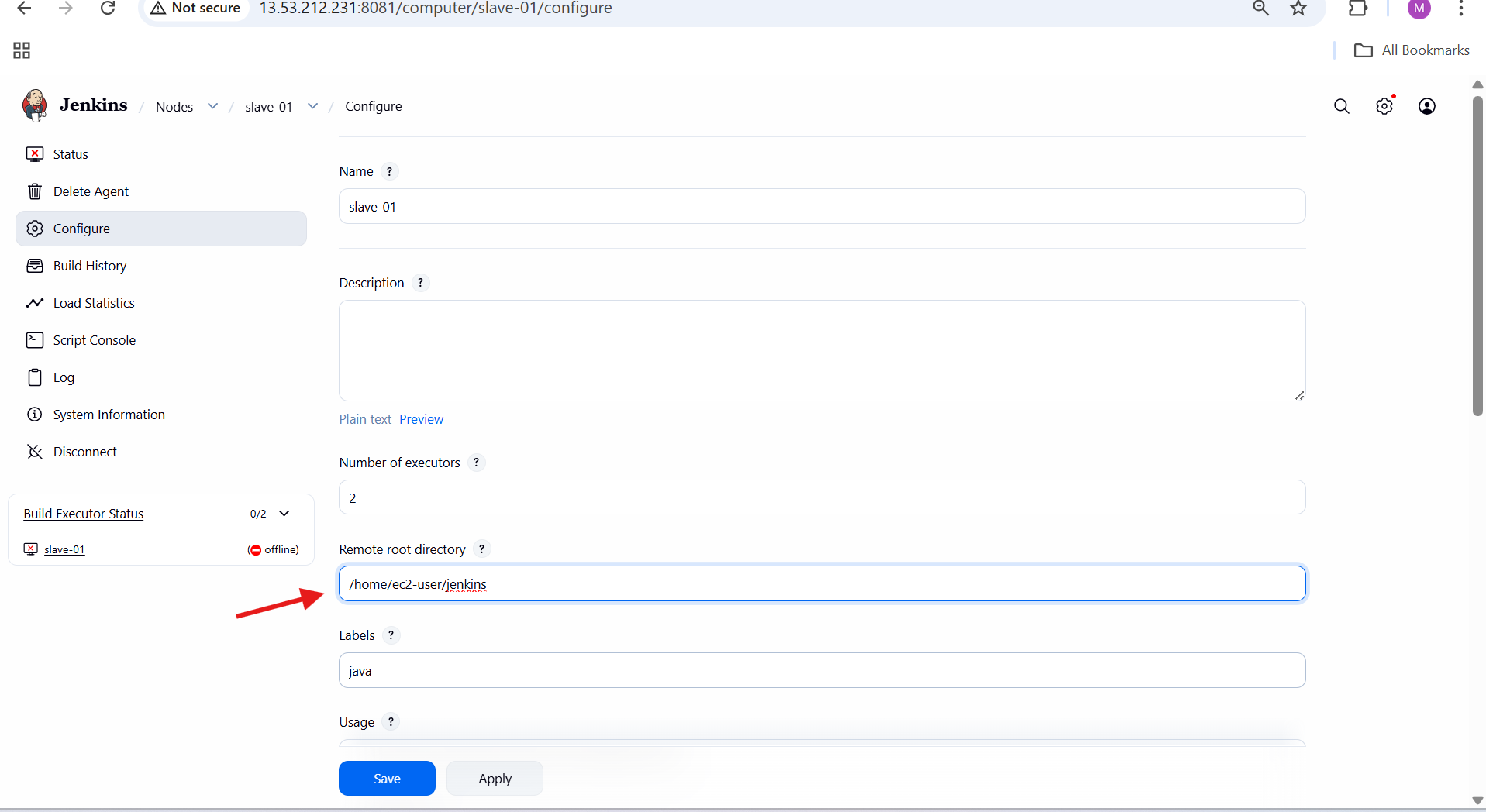


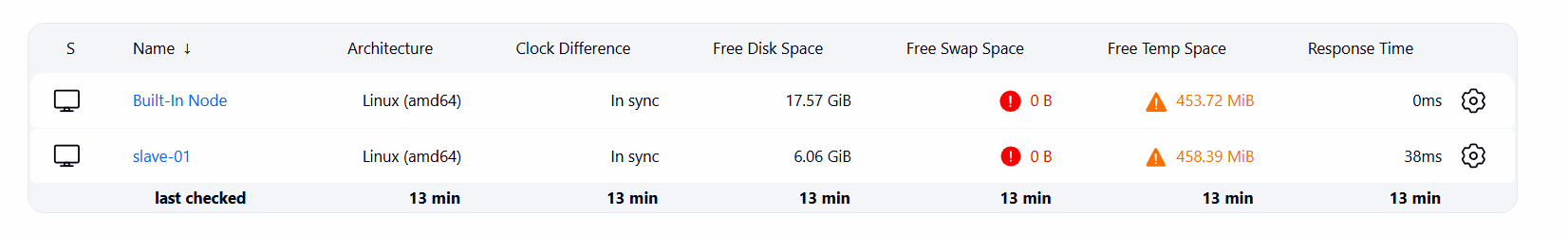


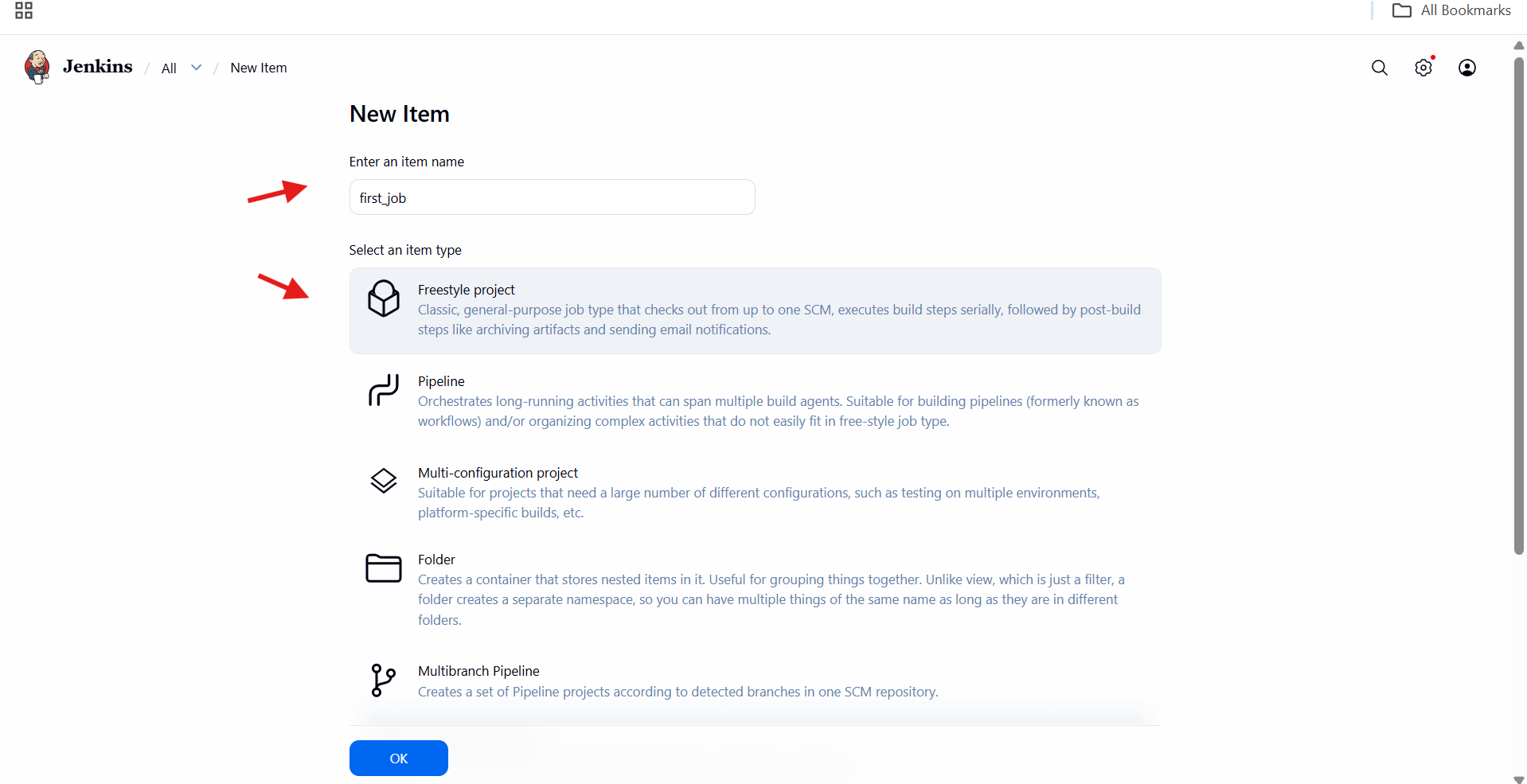


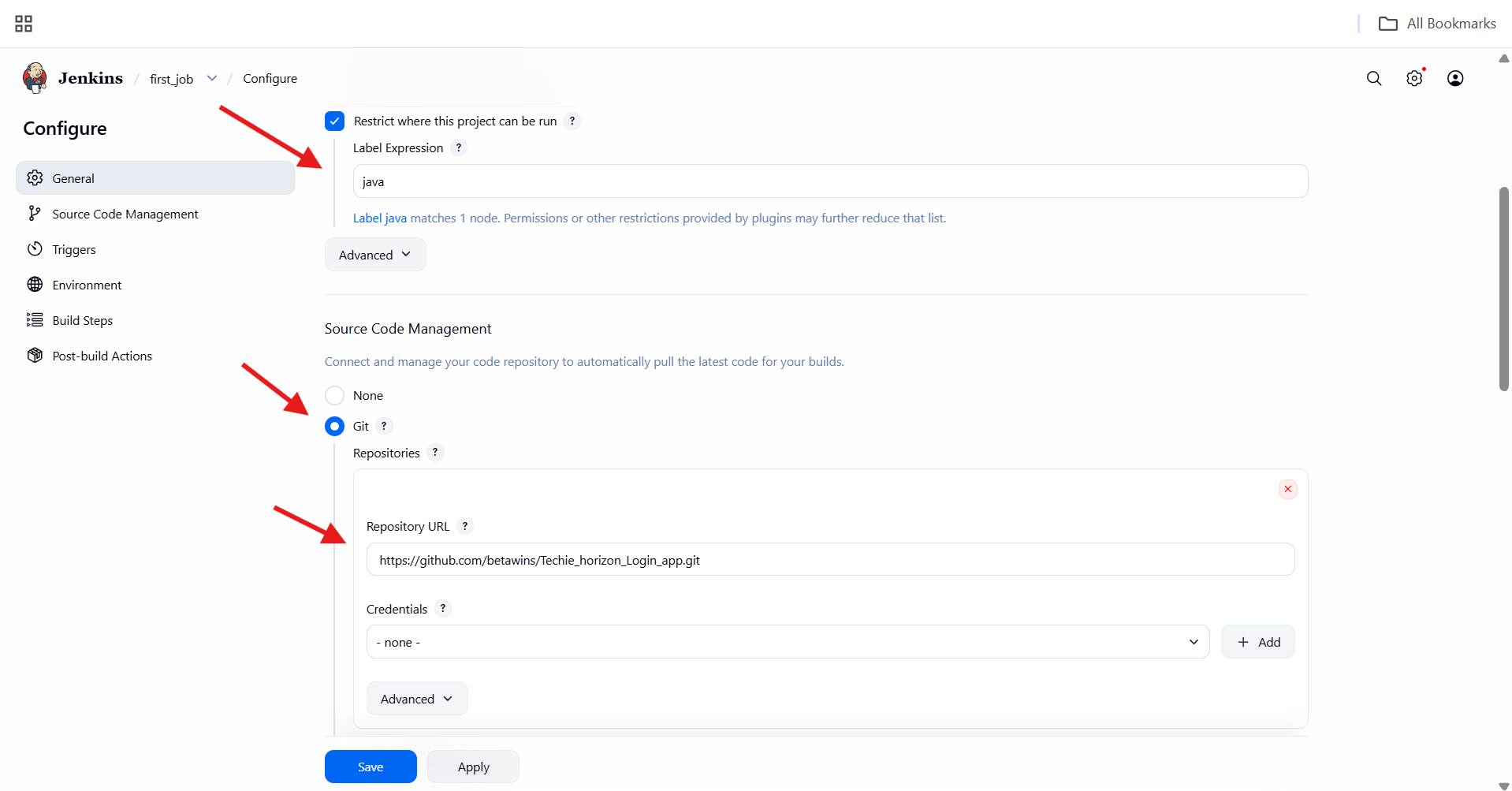


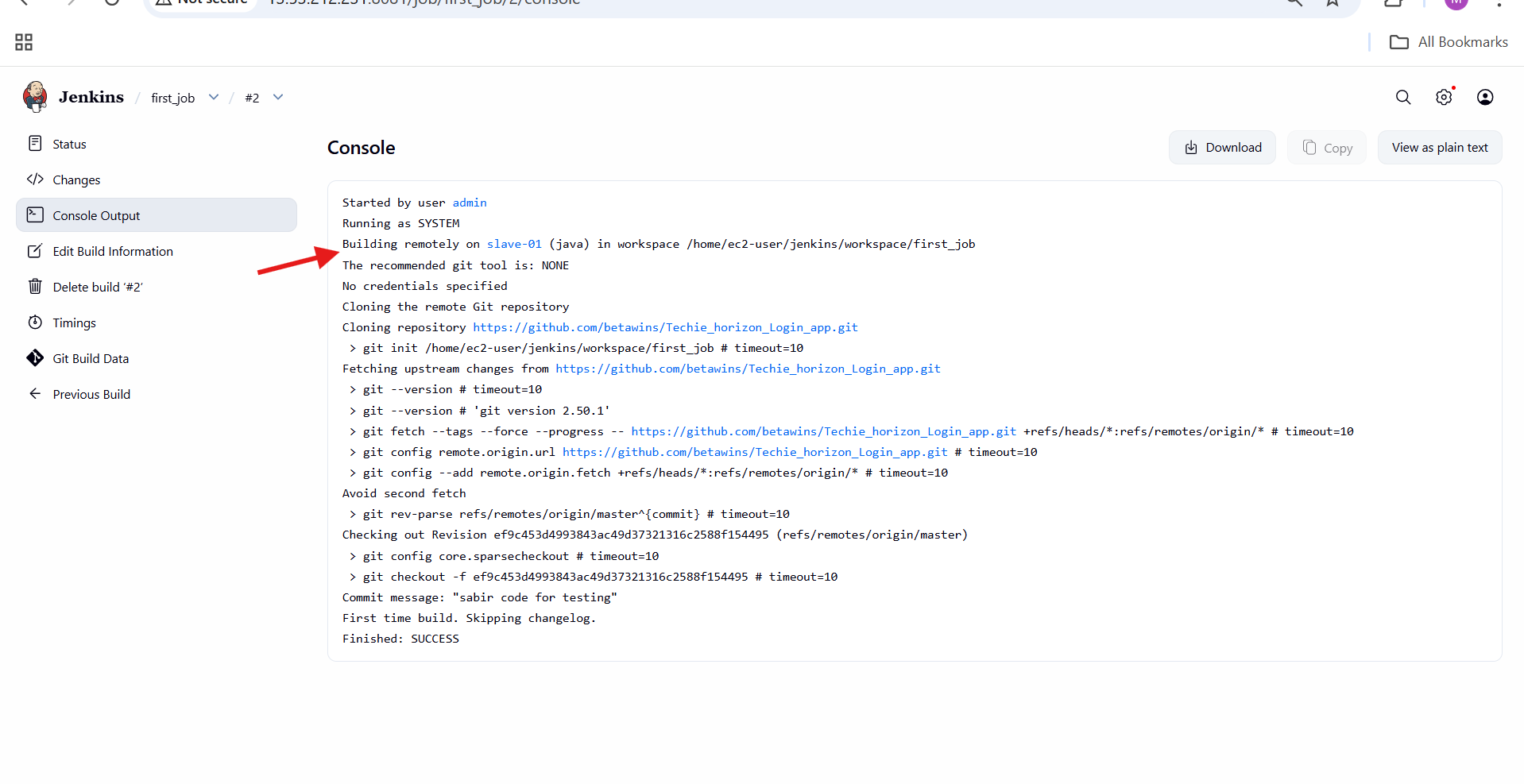


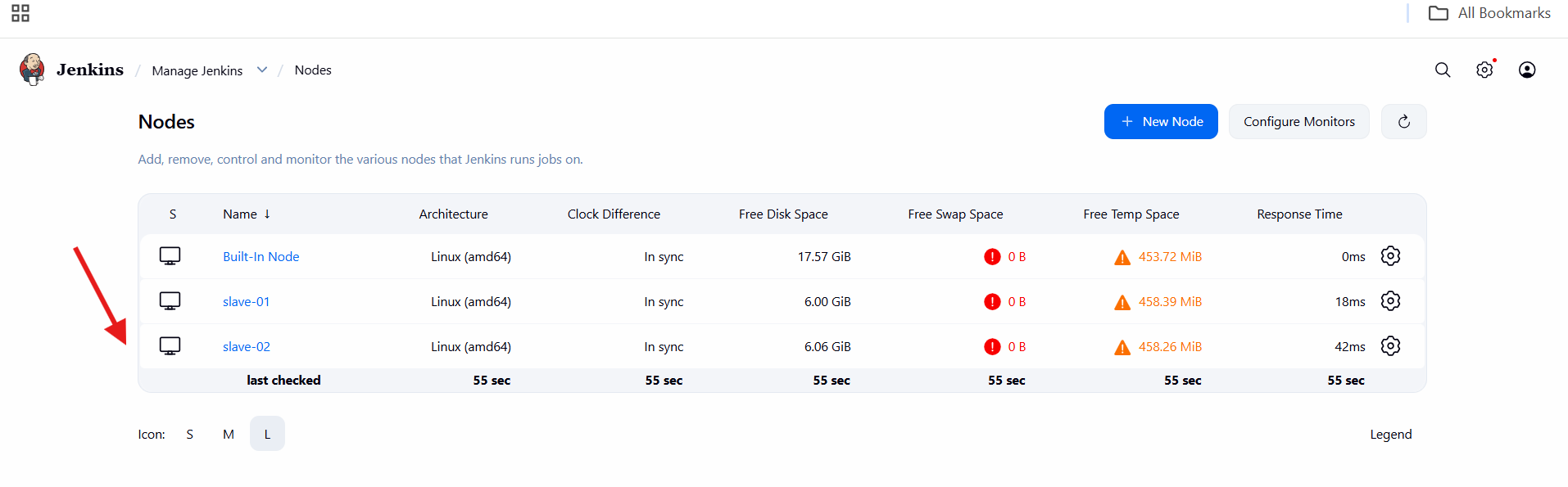












2.Configure webhooks to Jenkins job.

**Jenkins Job Configuration**

Opened Jenkins and selected the job **webhook**.

Clicked **Configure**.

Enabled **Restrict where this project can be run**.

Entered **Label Expression** as:

master

**Source Code Management Setup**

Selected **Git** under Source Code Management.

Entered **Repository URL**:

https://github.com/mujahed1716/devops-hub.git

Left **Credentials** as none (public repository).

**Trigger Configuration**

Navigated to **Triggers** section.

Enabled:

GitHub hook trigger for GITScm polling

**Build Step Configuration**

Scrolled to **Build Steps**.

Selected **Execute shell**.

Added command:

sleep 15

Clicked **Save**.

**GitHub Webhook Creation**

Opened GitHub repository:

mujahed1716/devops-hub

Navigated to:

Settings → Webhooks → Add webhook

Entered **Payload URL**:

http://13.53.212.231:8081/github-webhook/

Selected **Content type**:

application/json

Kept **Secret** empty.

Selected:

Send me everything

Enabled **Active**.

Clicked **Add webhook**.

Verified webhook added successfully.

**Repository Change**

Opened **README.md** in the repository.

Edited the file.

Committed changes to the **master** branch.

**Jenkins Build Execution**

GitHub webhook triggered Jenkins automatically.

Jenkins job started without manual build.

Build executed on **Built-in (master) node**.

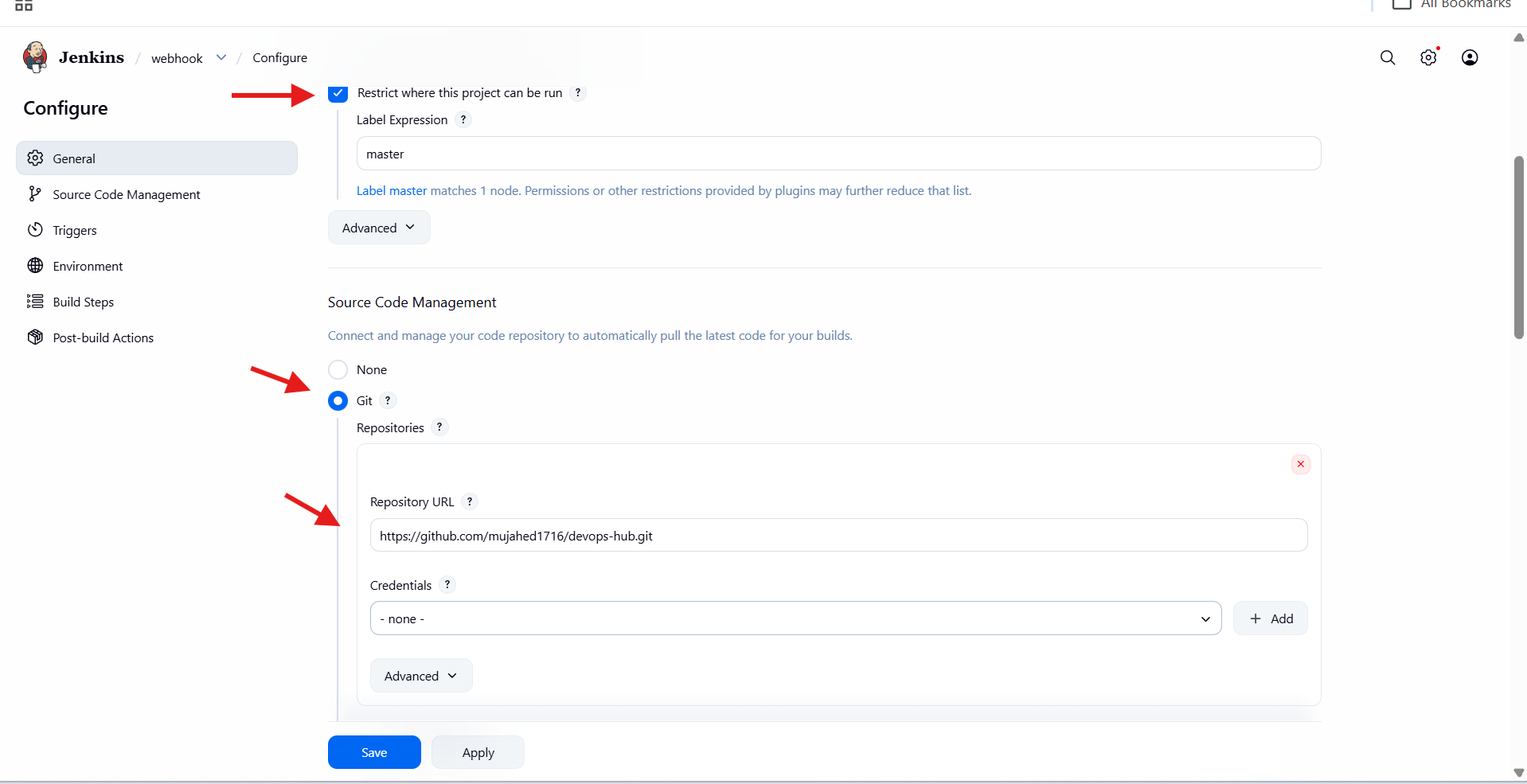
Jenkins cloned the repository successfully.

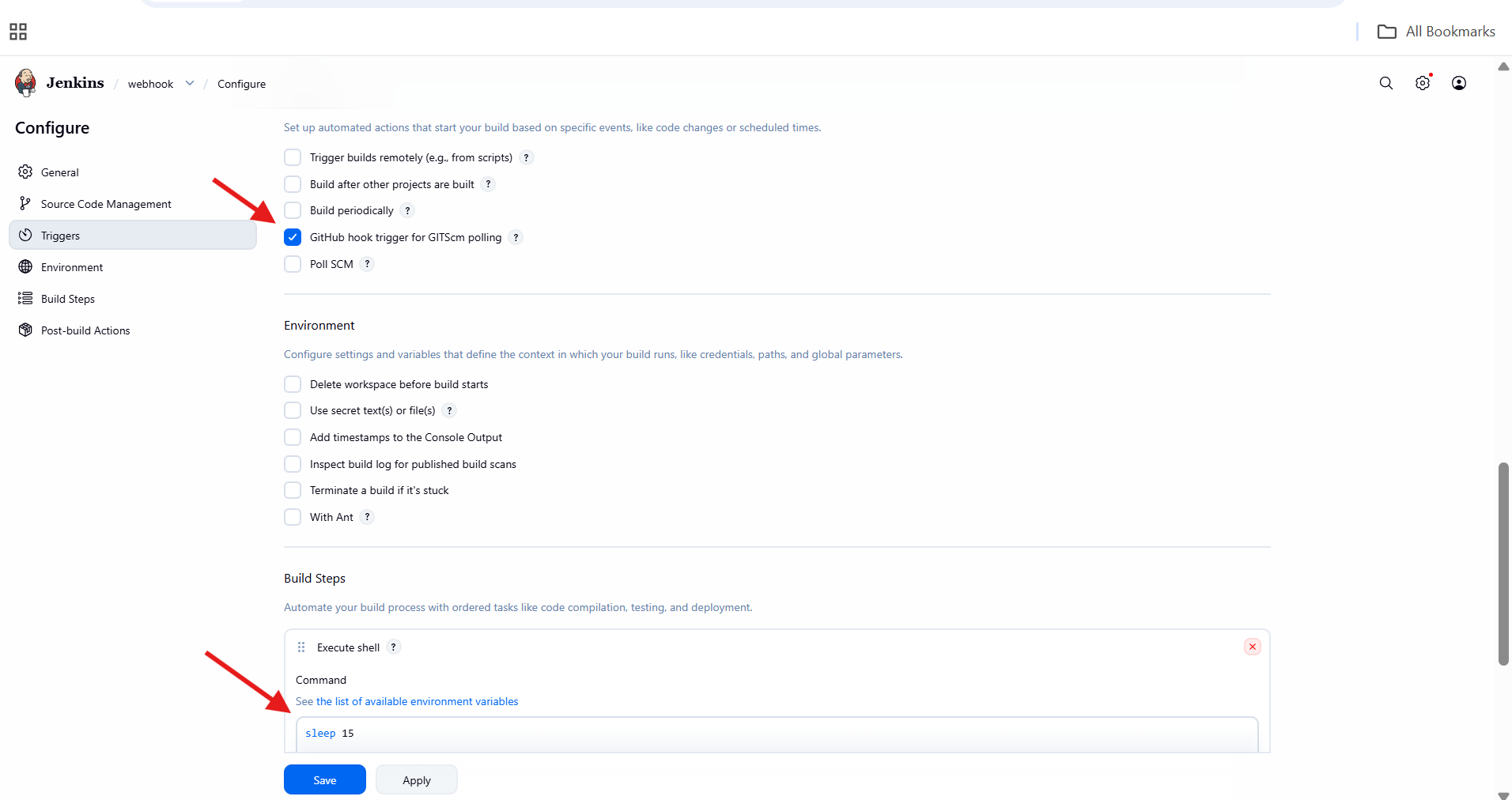
Shell command executed.

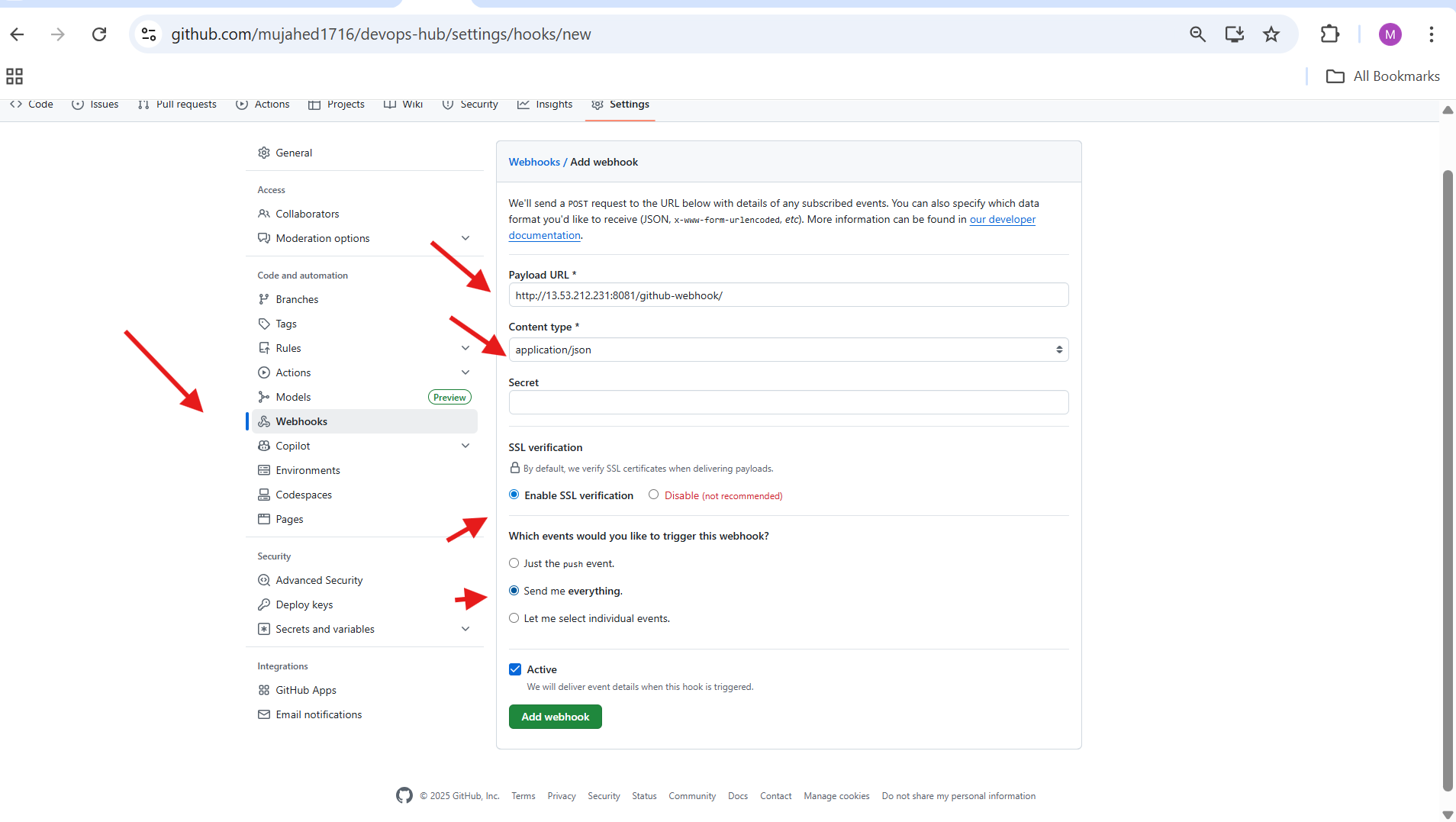
Console output showed:

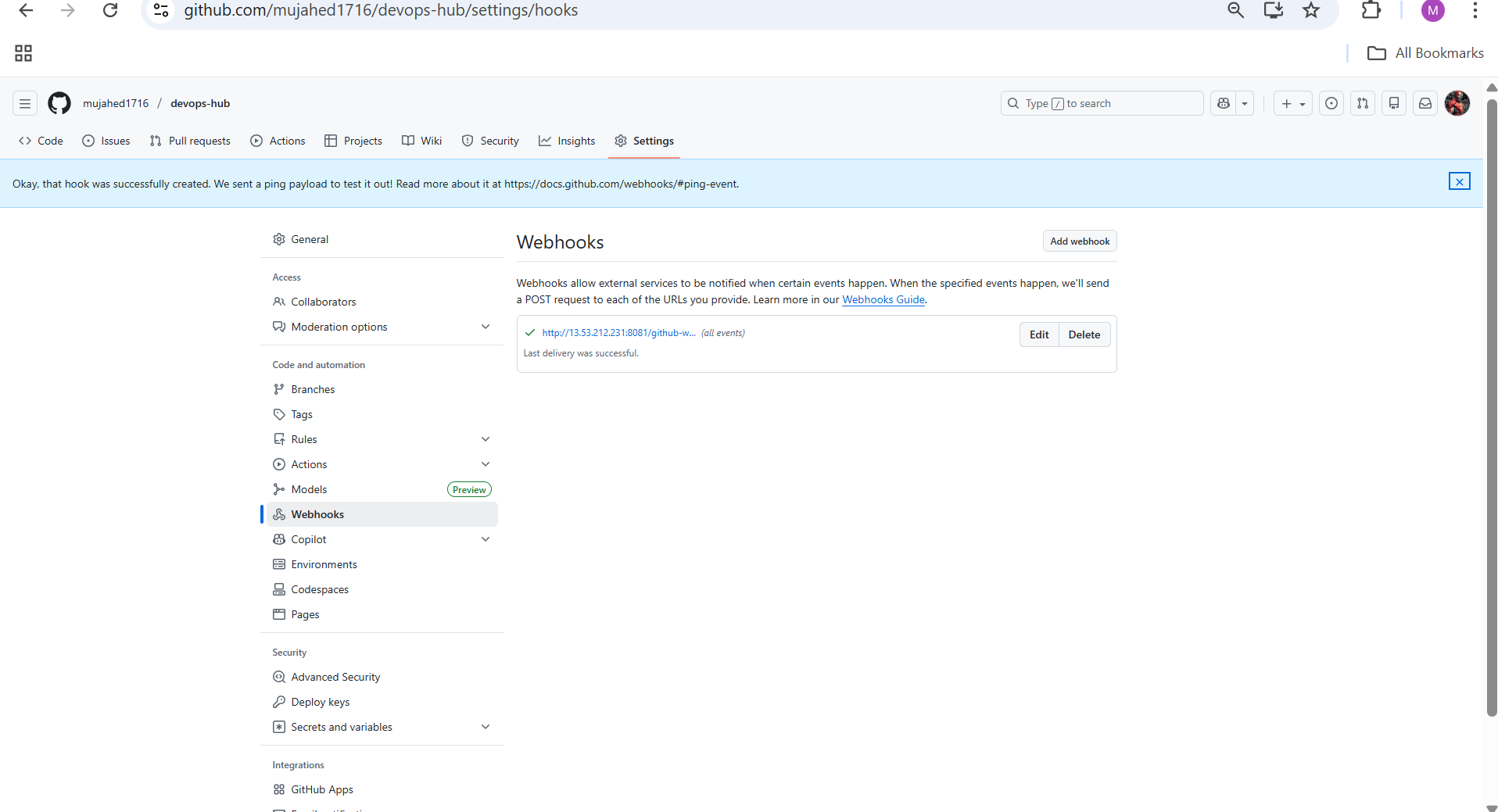
job triggered successfully

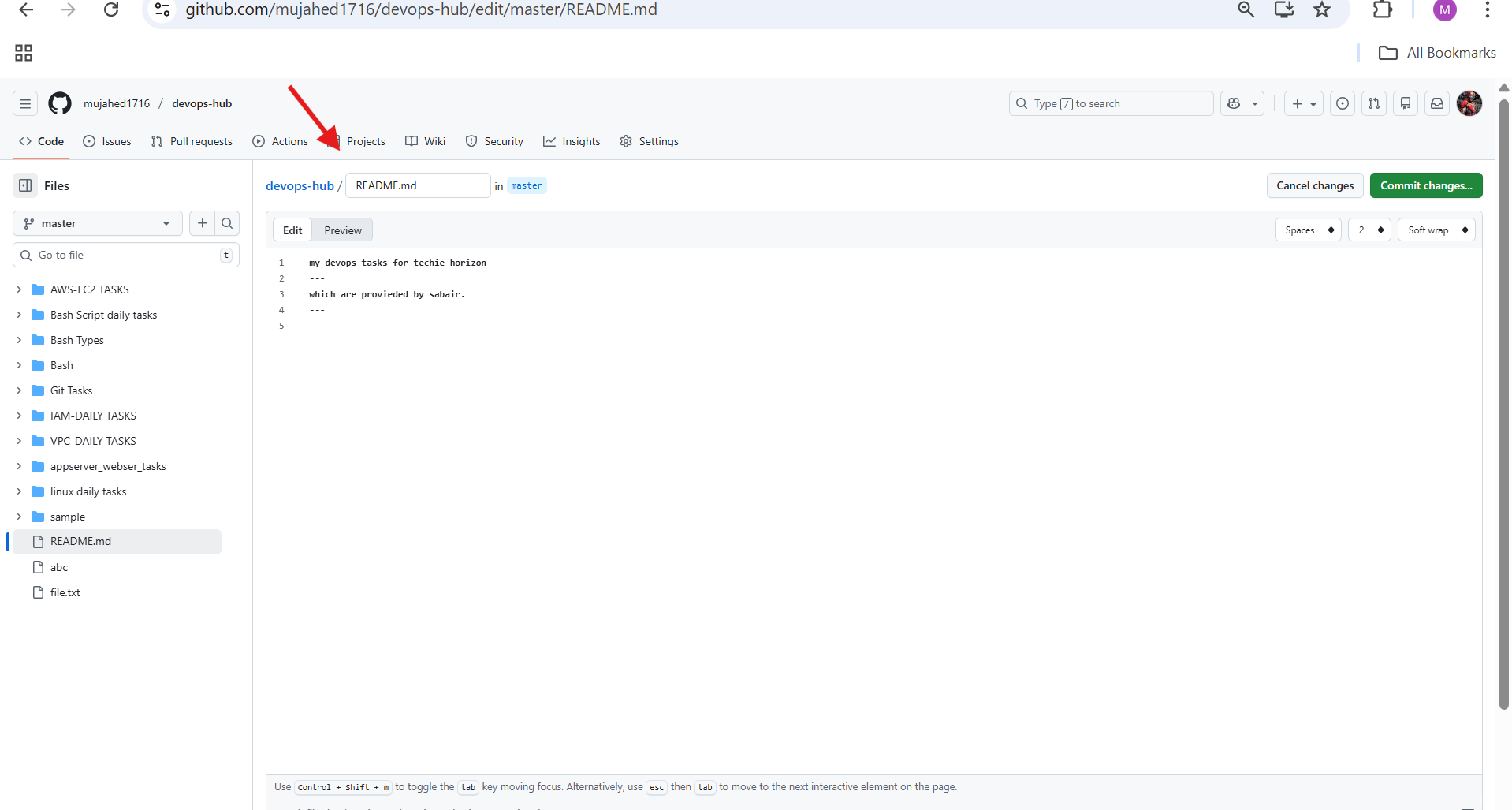
Build finished with **SUCCESS** status.

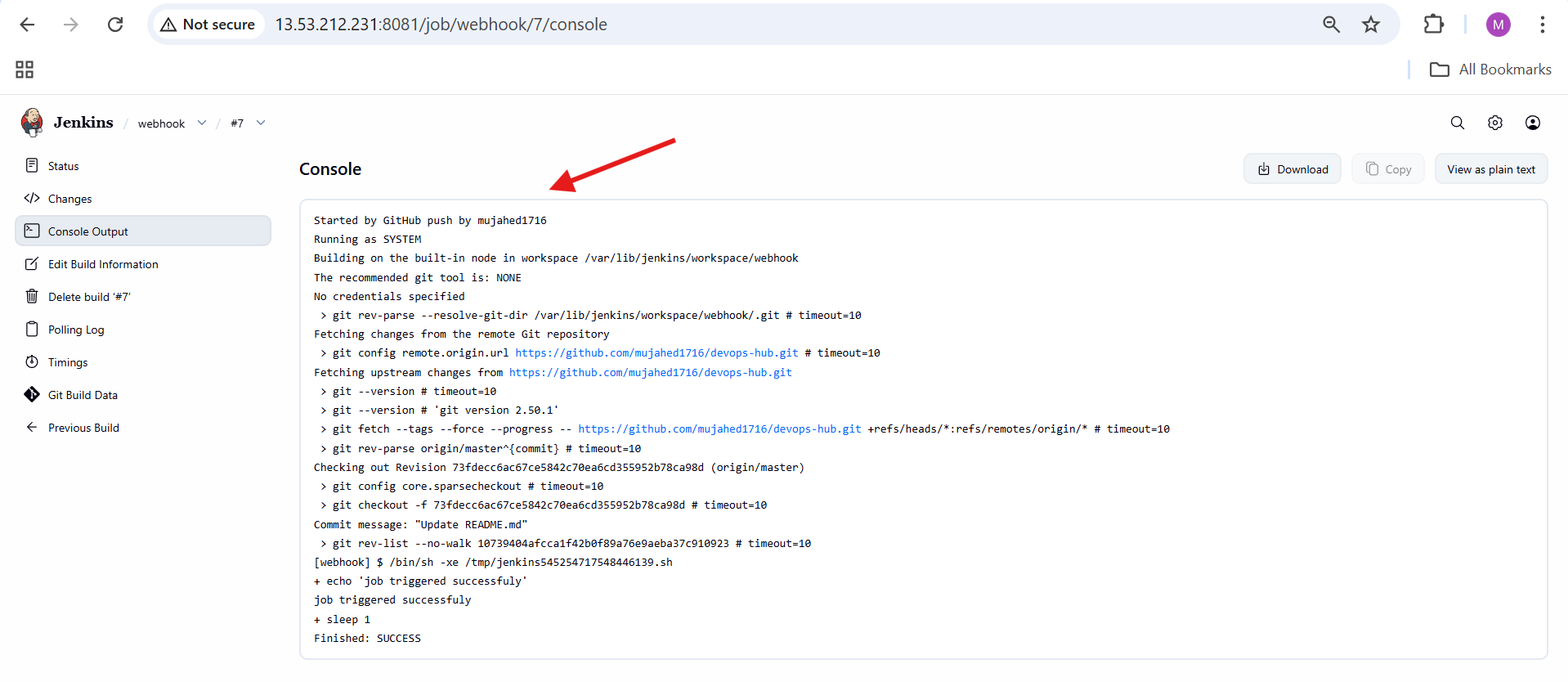












3.Configure poll scm and build periodical options in Jenkins job.

**Steps Performed – Jenkins Periodic Build (first\_job)**

**Job General Configuration**

Opened Jenkins and selected **first\_job**.

Clicked **Configure**.

Added **Description**:

added build periodically

Enabled **Restrict where this project can be run**.

Set **Label Expression** to:

master

**Source Code Management**

Navigated to **Source Code Management**.

Selected **Git**.

Entered **Repository URL**:

https://github.com/mujahed1716/devops-hub.git

Left **Credentials** as none.

Set **Branch Specifier** to:

\*/master

**Trigger – Build Periodically**

Opened **Triggers** section.

Enabled **Build periodically**.

Entered schedule:

\* \* \* \* \*

Saved the job configuration.

**Jenkins Built Execution**

Jenkins automatically triggered builds based on the schedule.

Multiple builds appeared in **Build History**.

Selected a build and opened **Console Output**.

Verified message:

Started by timer

Jenkins fetched the latest code from GitHub.

Build completed with **SUCCESS** status.

**SCM Trigger Verification**

Returned to **Triggers** configuration.

Enabled **Poll SCM**.

Entered schedule:

\* \* \* \* \*

Saved configuration.

**GitHub Repository Change**

Opened GitHub repository **devops-hub**.

Edited **README.md**.

Added content to the file.

Committed changes to **master** branch.

**Jenkins SCM-Based Build**

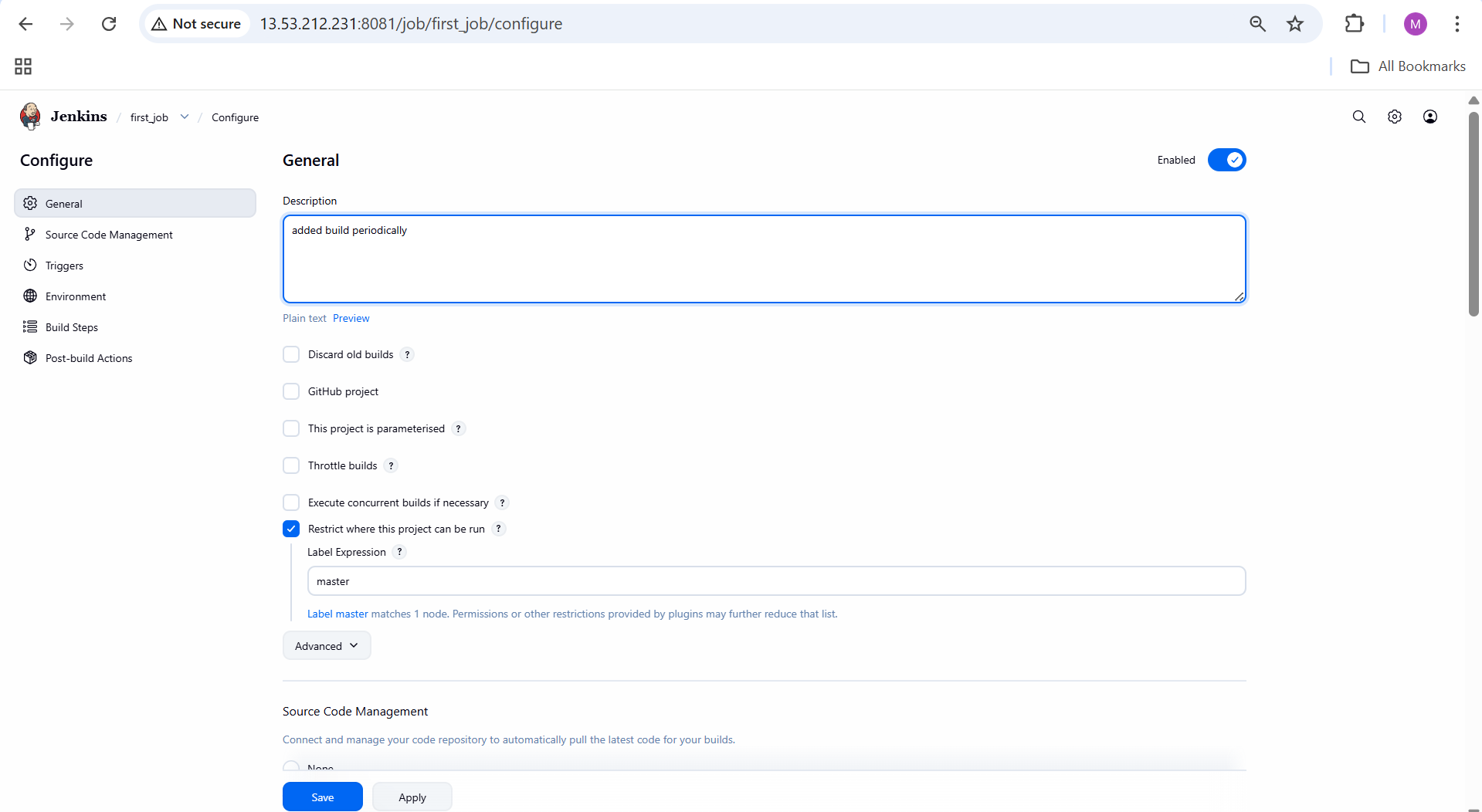
Jenkins detected SCM change.

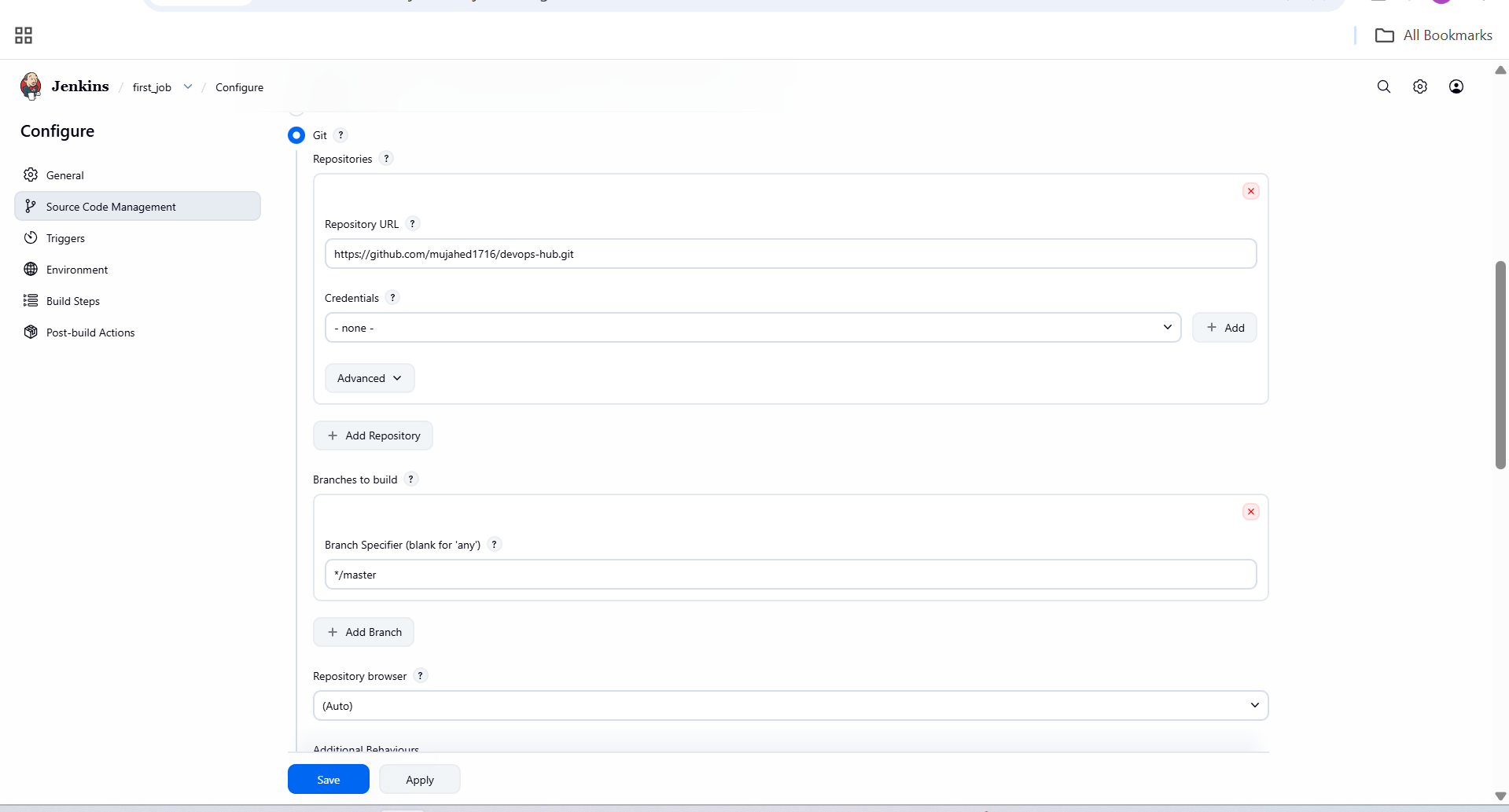
New build appeared with status **Started by an SCM change**.

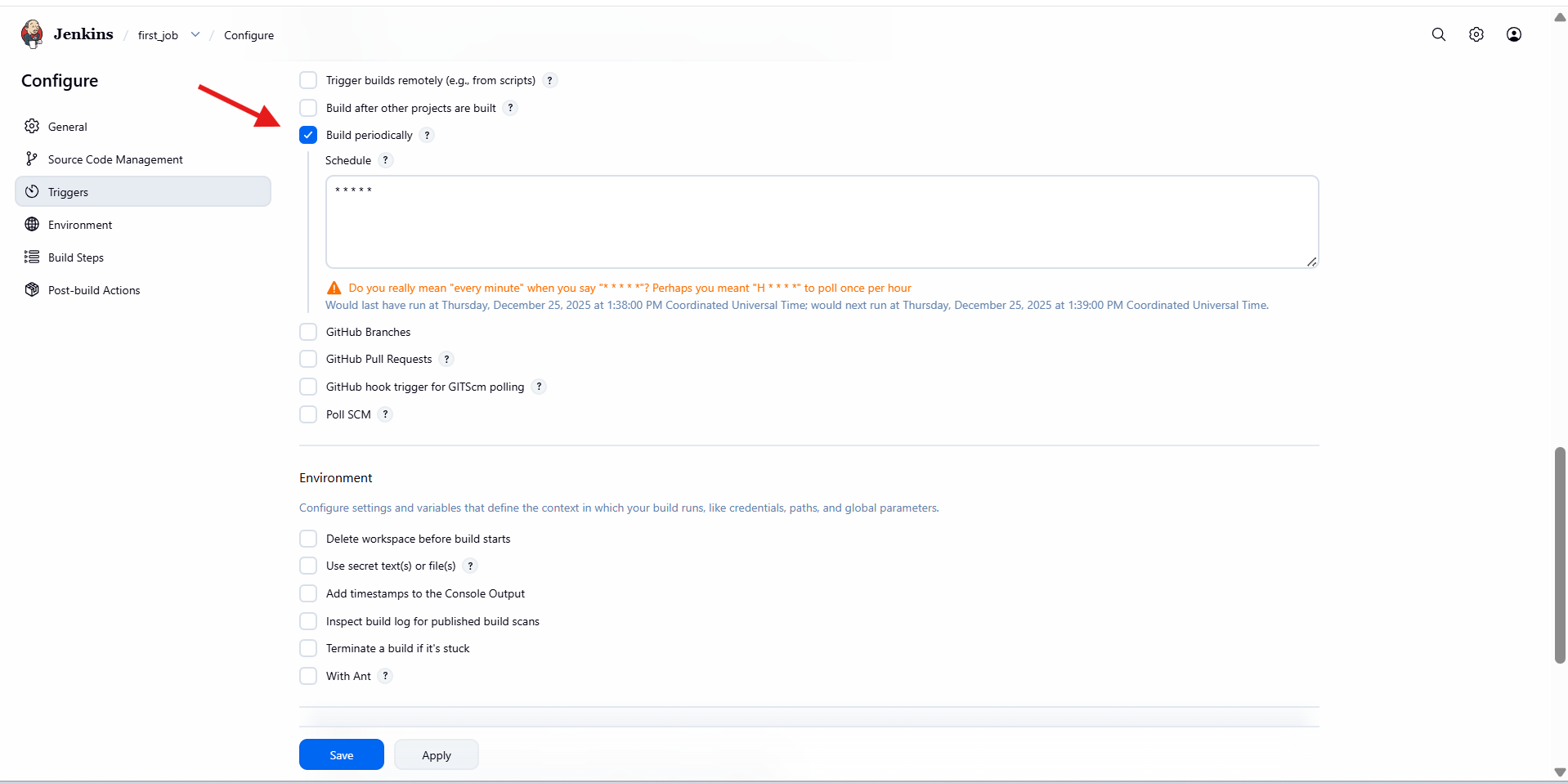
Opened Console Output.

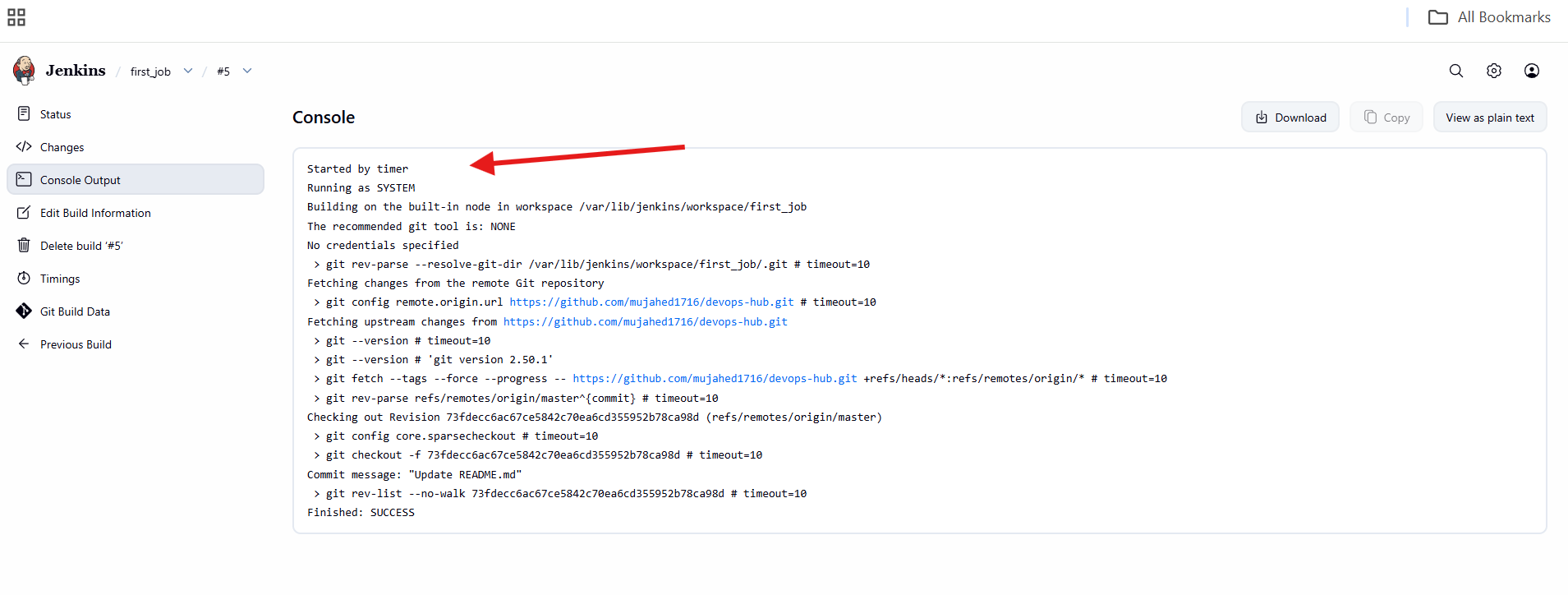
Verified Git fetch and checkout steps.

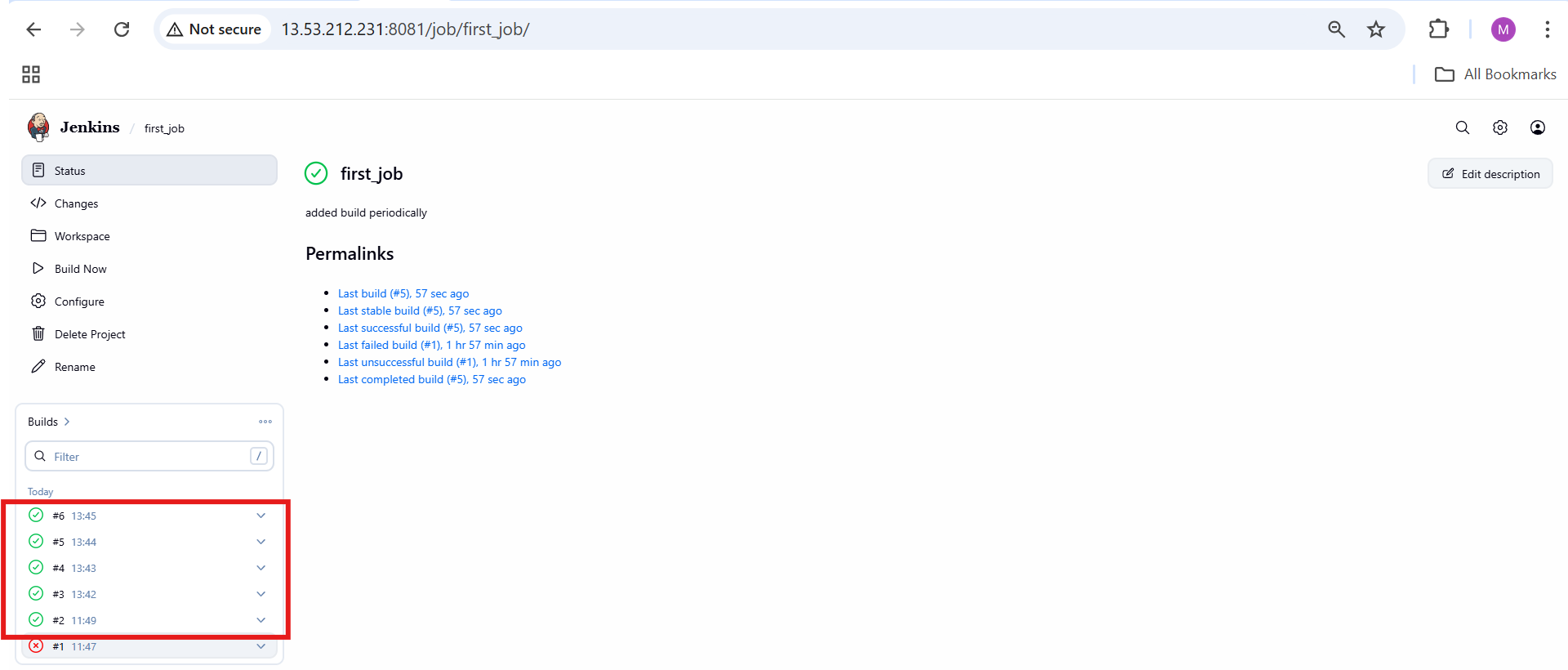
Build completed with **SUCCESS**.

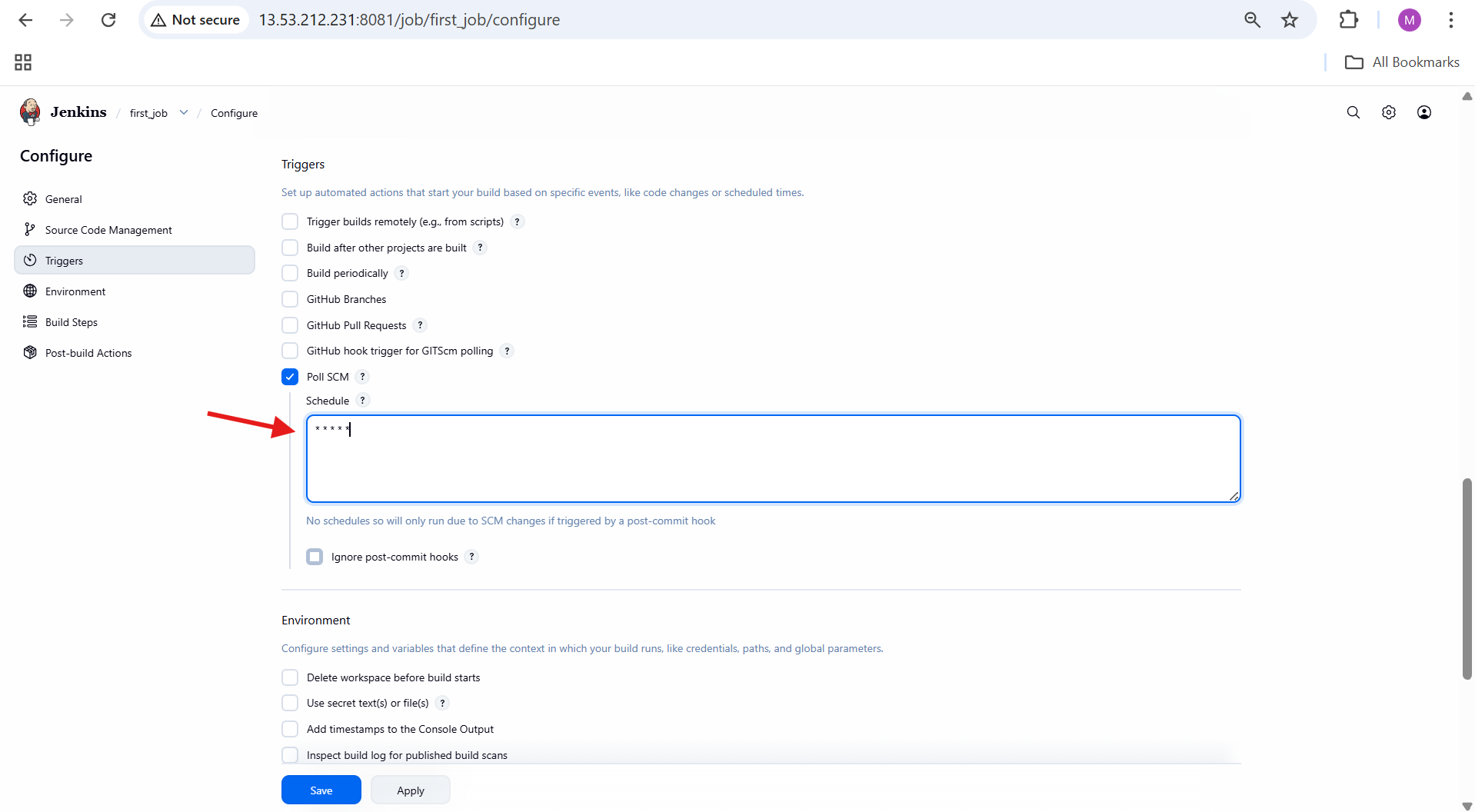


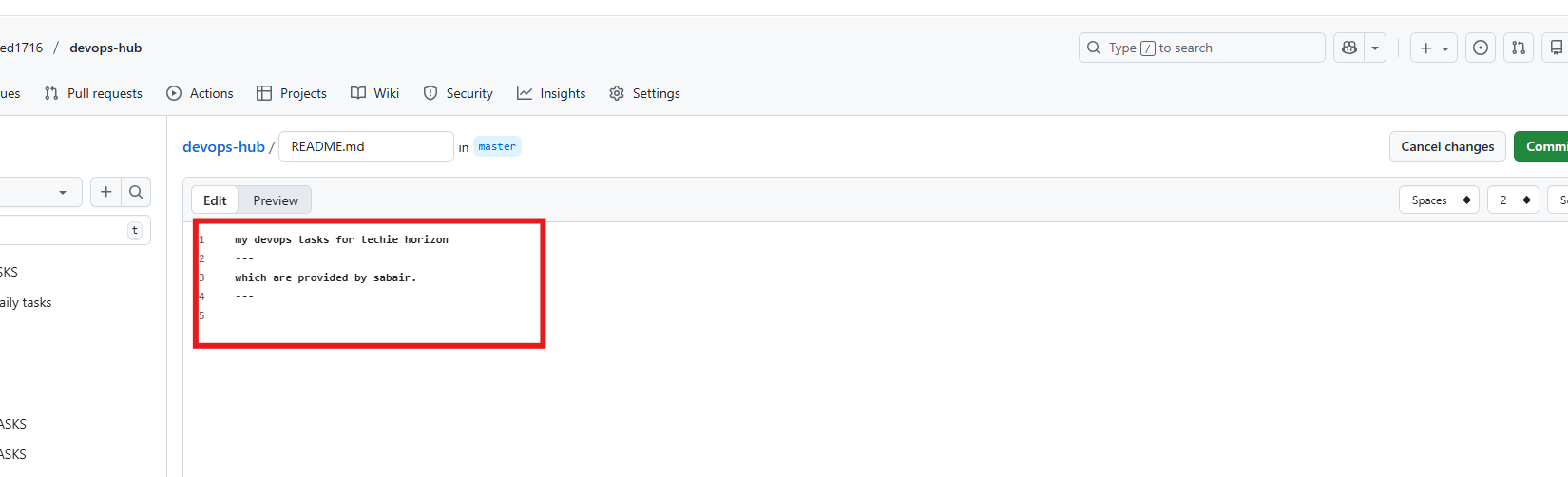


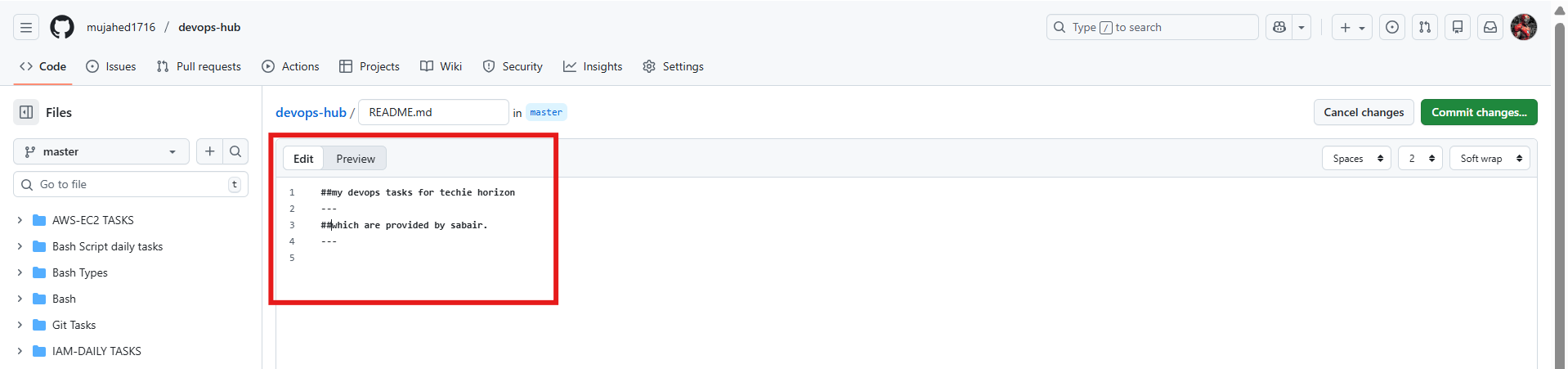
















4. Take backup of Jenkins server by using bash script.

**Steps Performed – Jenkins Backup Script**

Logged in to the server as root.

Navigated to /opt directory:

cd /opt

Listed contents of /opt:

ls

Created a directory for scripts:

mkdir /opt/scripts

Moved into the scripts directory:

cd /opt/scripts

Created a backup script file:

vi jenkins\_backup.sh

Added Jenkins backup script content defining:

* + Jenkins home directory
  + Backup directory
  + Date variable
  + tar command with exclusions

Saved and exited the script file.

Provided execute permission to the script:

chmod +x jenkins\_backup.sh

Executed the backup script:

./jenkins\_backup.sh

Verified message displayed:

Online Jenkins backup completed

**Backup Verification**

Navigated back to /opt:

cd /opt

Listed directories:

ls

Entered Jenkins backup directory:

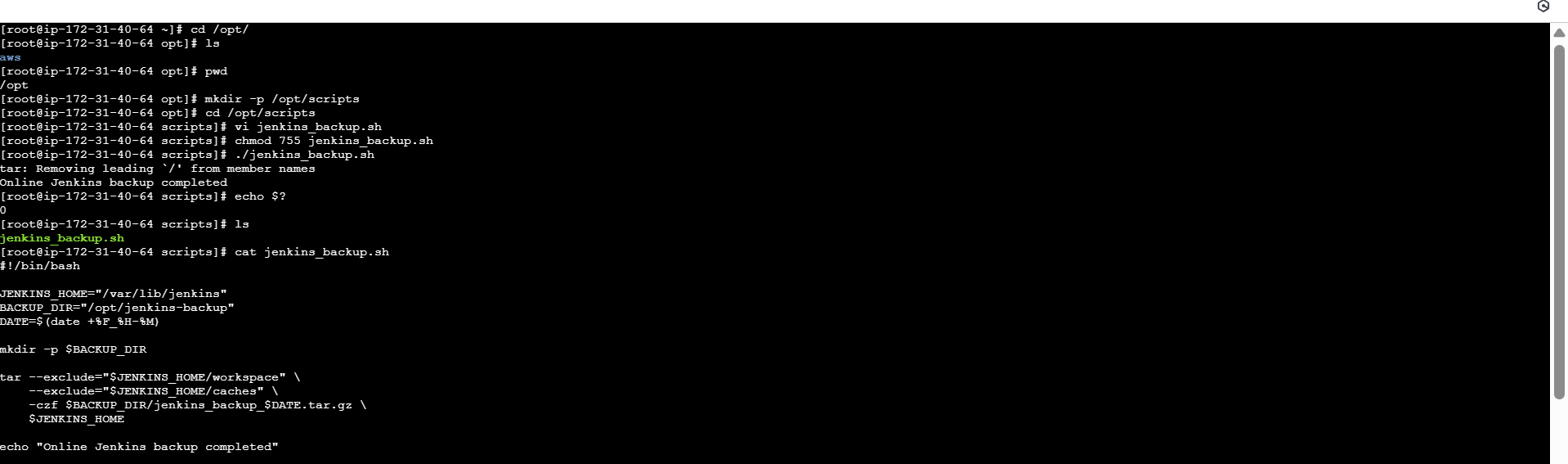
cd jenkins-backup

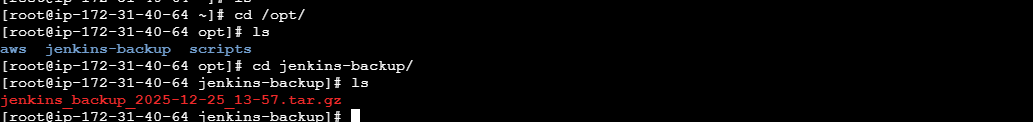
Listed backup files:

ls

Confirmed Jenkins backup archive created:

jenkins\_backup\_2025-12-25\_13-57.tar.gz





5. Take backup of Jenkins using rethin backup plugin.

**Steps Performed – Jenkins ThinBackup Plugin Backup**

**Install ThinBackup Plugin**

Opened Jenkins UI.

Navigated to:

Manage Jenkins → Plugins → Available plugins

Searched for **ThinBackup**.

Selected the **ThinBackup** plugin.

Clicked **Install**.

**Plugin Installation Verification**

Opened **Download progress** page.

Verified:

* + Internet connectivity check → Success
  + Update center connectivity → Success
  + ThinBackup plugin → Installed successfully

Clicked **Go back to the top page**.

**Access ThinBackup**

Navigated to:

Manage Jenkins → ThinBackup

Confirmed ThinBackup page loaded successfully.

Observed available options:

* **Backup now**
* **Restore**

**Prepare Backup Directory on Server**

Logged in to Jenkins server as root.

Created backup directory:

mkdir -p /opt/jenkins-backup

Changed ownership to Jenkins user:

chown -R jenkins:jenkins /opt/jenkins-backup

Set permissions:

chmod 755 /opt/jenkins-backup

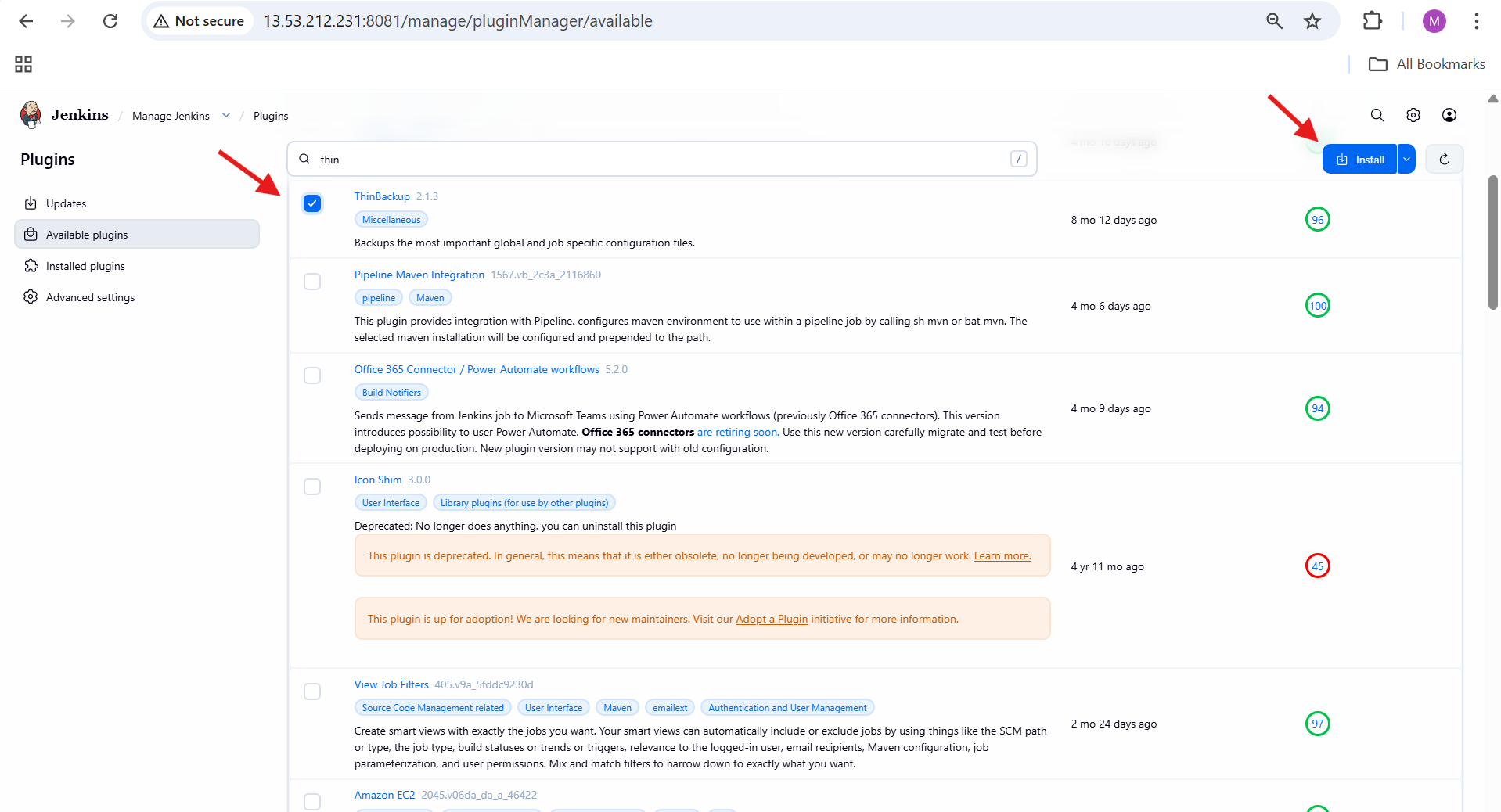
**Verify Backup File**

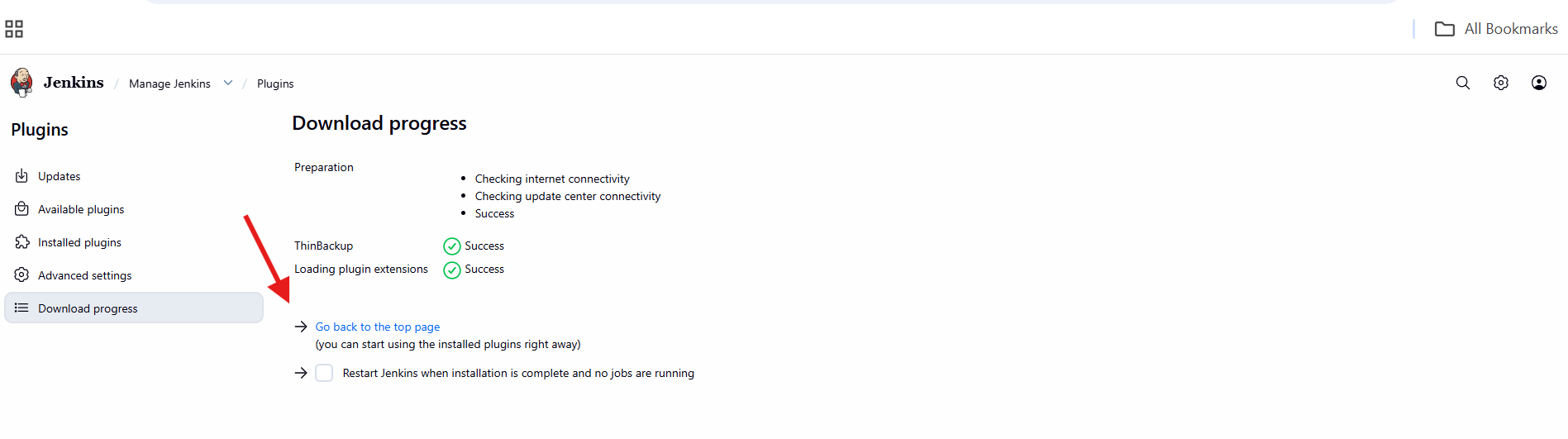
Listed contents of backup directory:

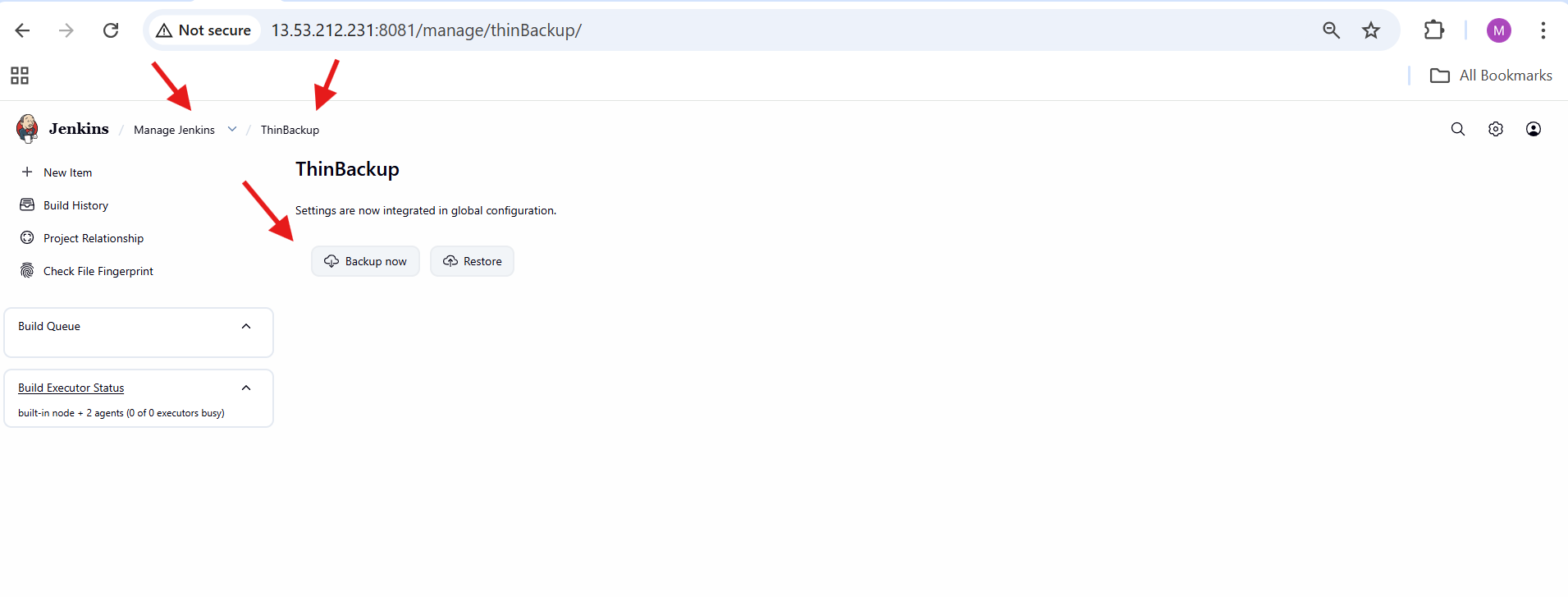
ls -ltr /opt/jenkins-backup

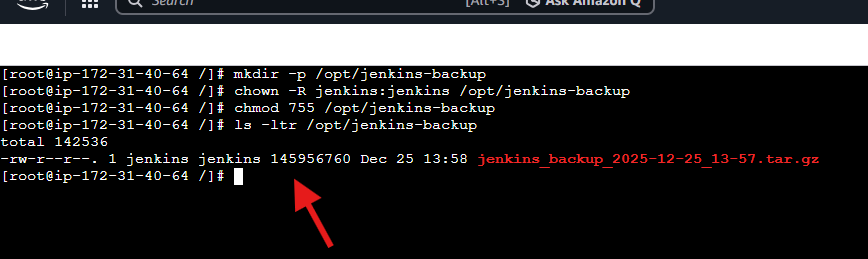
Verified backup archive exists:

jenkins\_backup\_2025-12-25\_13-57.tar.gz









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

**On Jenkins Master (Old Server)**

Logged in to **jenkins-master** EC2 instance.

Navigated to /opt:

cd /opt

Listed directories and verified jenkins-backup exists.

Moved into backup directory:

cd jenkins-backup

Verified Jenkins backup file is present:

jenkins\_backup\_2025-12-25\_13-57.tar.gz

**On Backup EC2 Instance (New Server)**

Logged in to **backup EC2 instance**.

Added Jenkins repository:

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

Imported Jenkins GPG key:

rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

Installed Java:

yum install java-17-amazon-corretto -y

Installed Jenkins:

yum install jenkins -y

Started Jenkins service:

systemctl start jenkins

Verified Jenkins service status:

systemctl status jenkins

**Access Jenkins on Backup Server**

Opened browser and accessed Jenkins:

http://56.228.32.222:8080

Confirmed Jenkins UI loaded successfully.

**Transfer Backup File to Backup Server**

Returned to **jenkins-master**.

Changed permission on backup file:

Created a file test.pem and copied pem key

chmod 600 test.pem

Copied backup file to backup server:

scp -i test.pem jenkins\_backup\_2025-12-25\_13-57.tar.gz ec2-user@56.228.32.222:/tmp

Accepted SSH host key when prompted.

Verified file transfer completed successfully.

**Restore Backup on Backup Server**

Logged in to **backup EC2 instance**.

Navigated to /tmp:

cd /tmp

Verified backup file exists:

jenkins\_backup\_2025-12-25\_13-57.tar.gz

Extracted backup into Jenkins home:

tar -xvf jenkins\_backup\_2025-12-25\_13-57.tar.gz -C /var/lib/jenkins

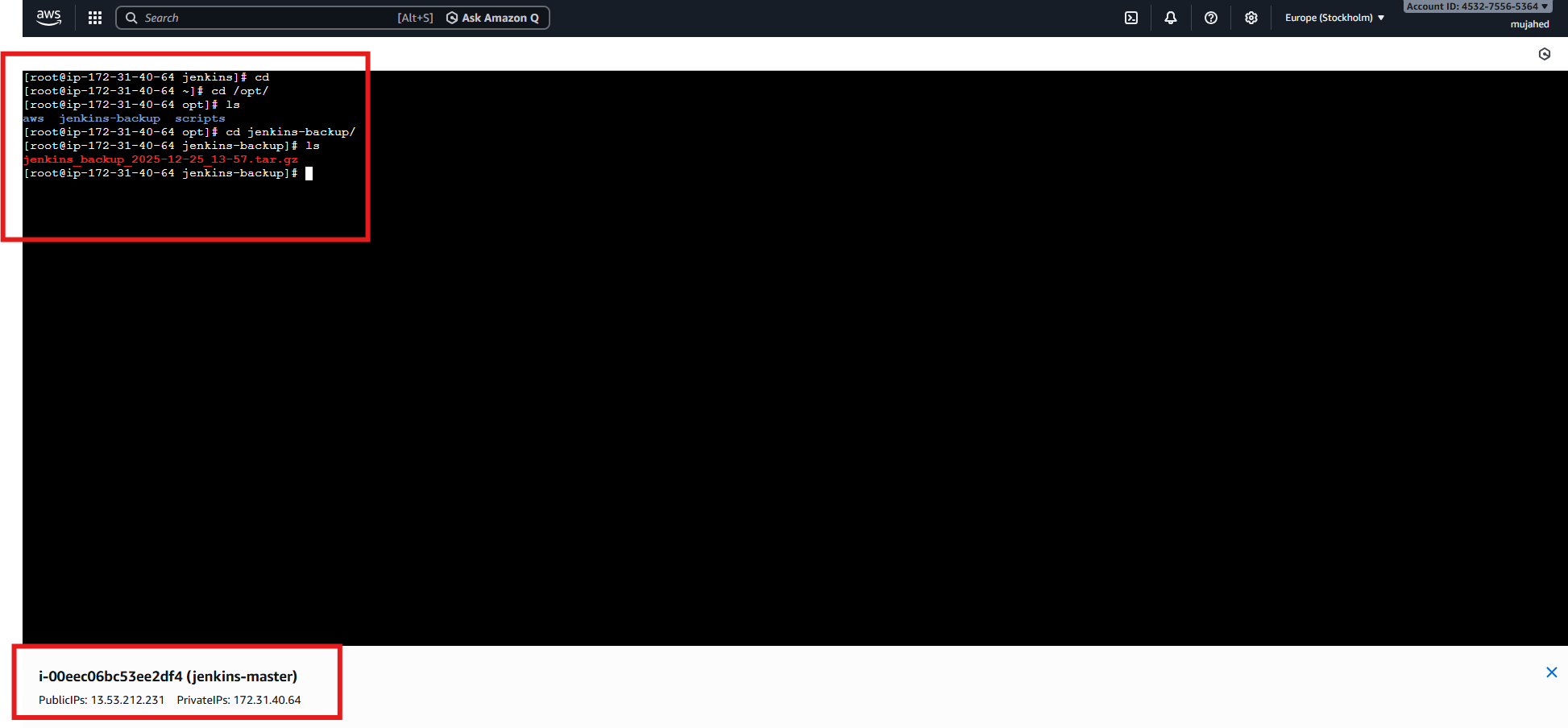
Verified Jenkins files restored under /var/lib/jenkins.

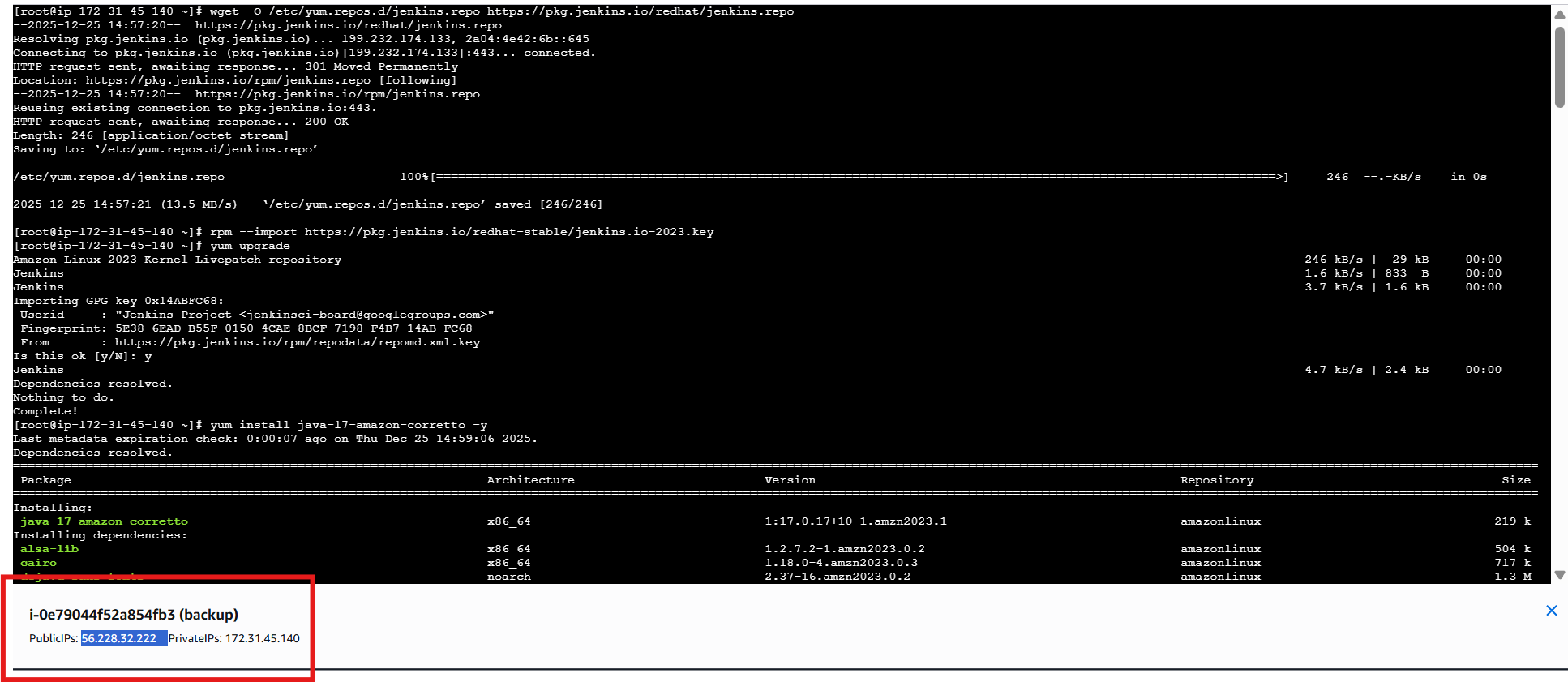
**Restart Jenkins**

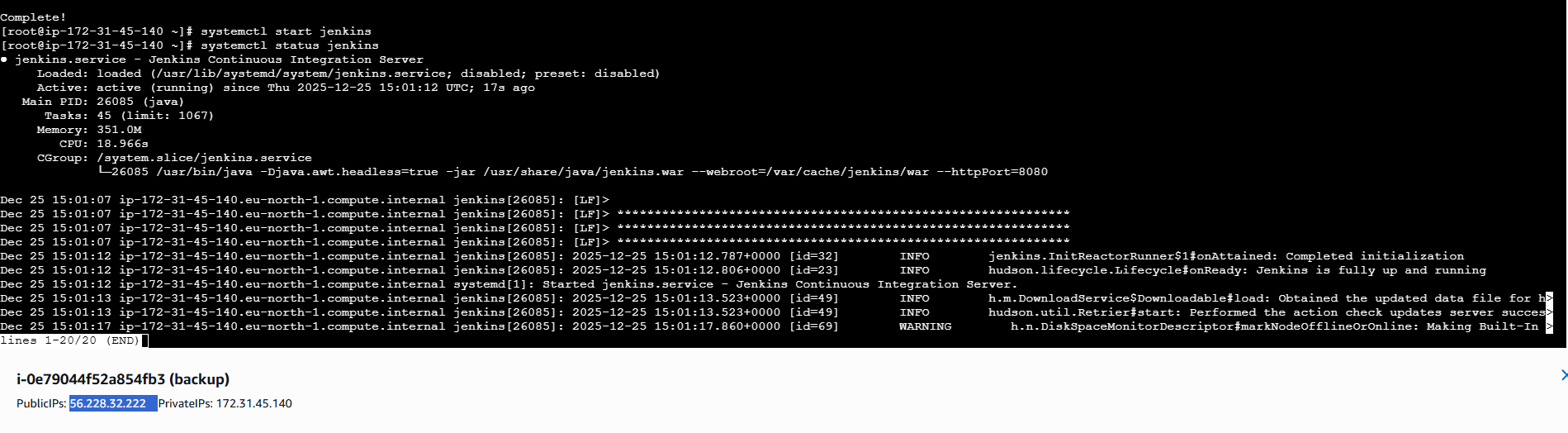
Restarted Jenkins using browser:

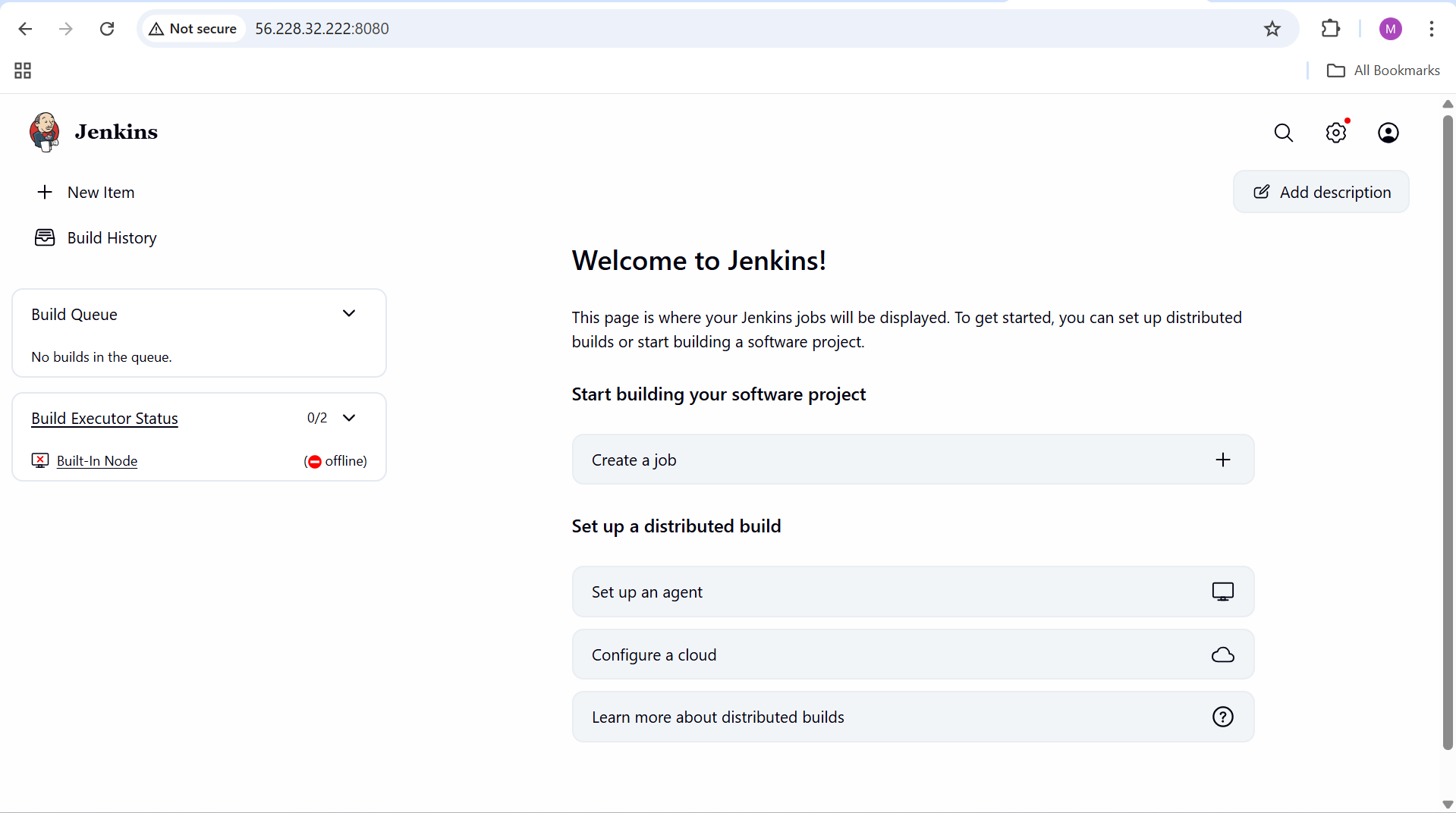
http://56.228.32.222:8080/restart

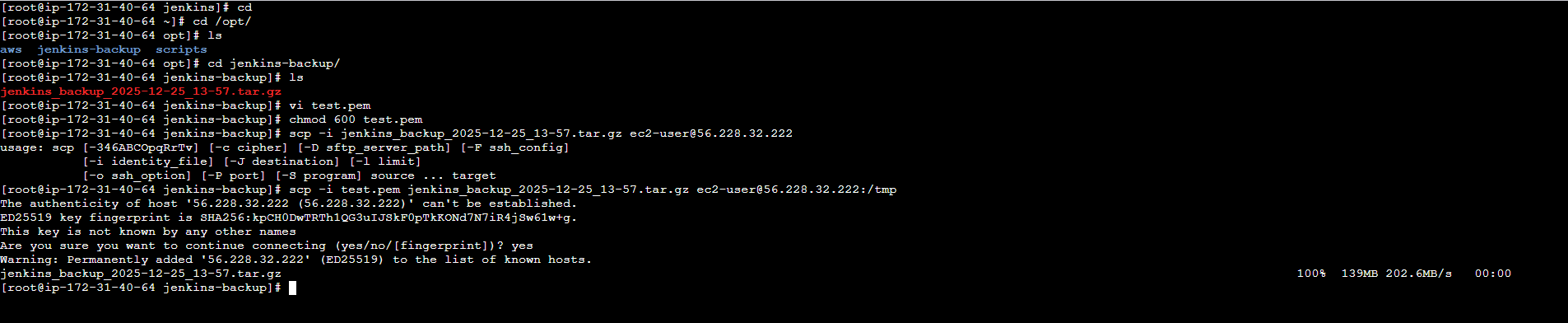
Jenkins restarted successfully.

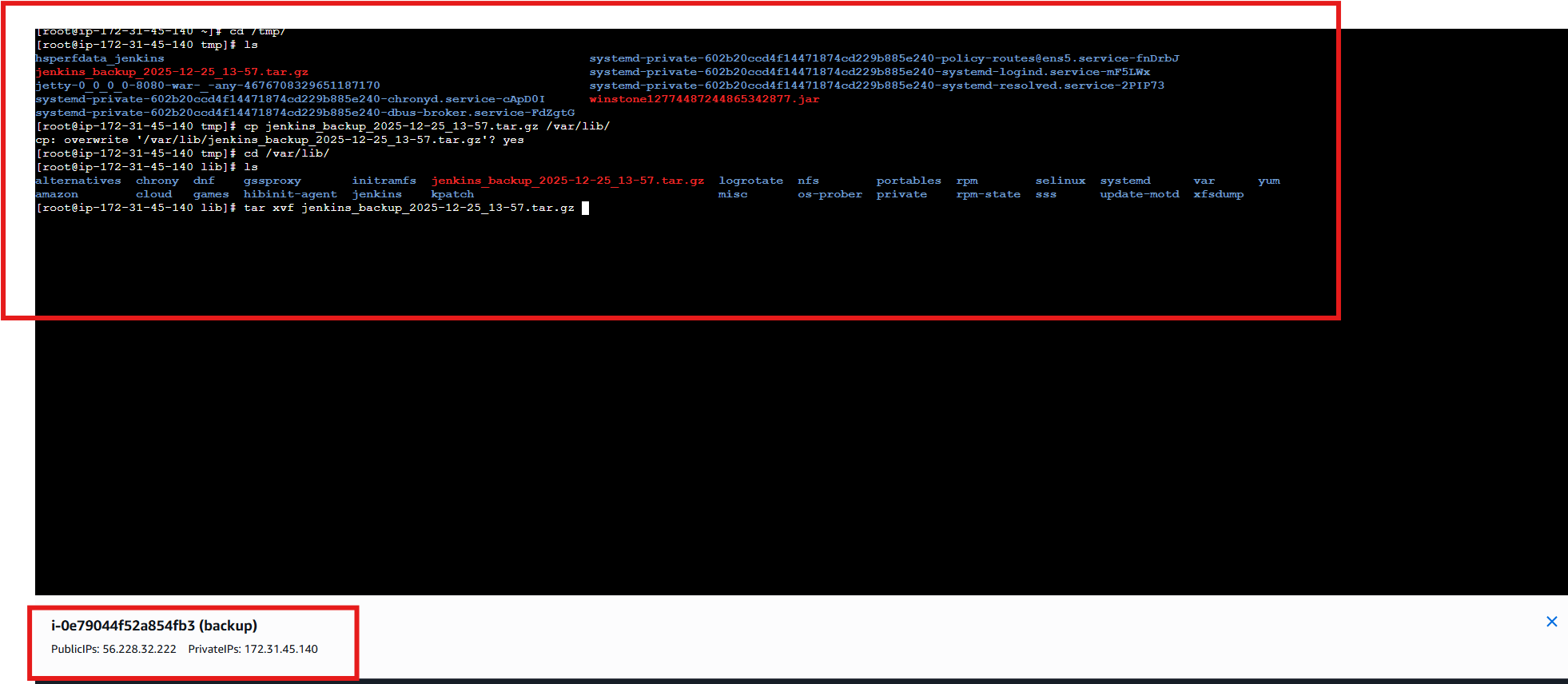


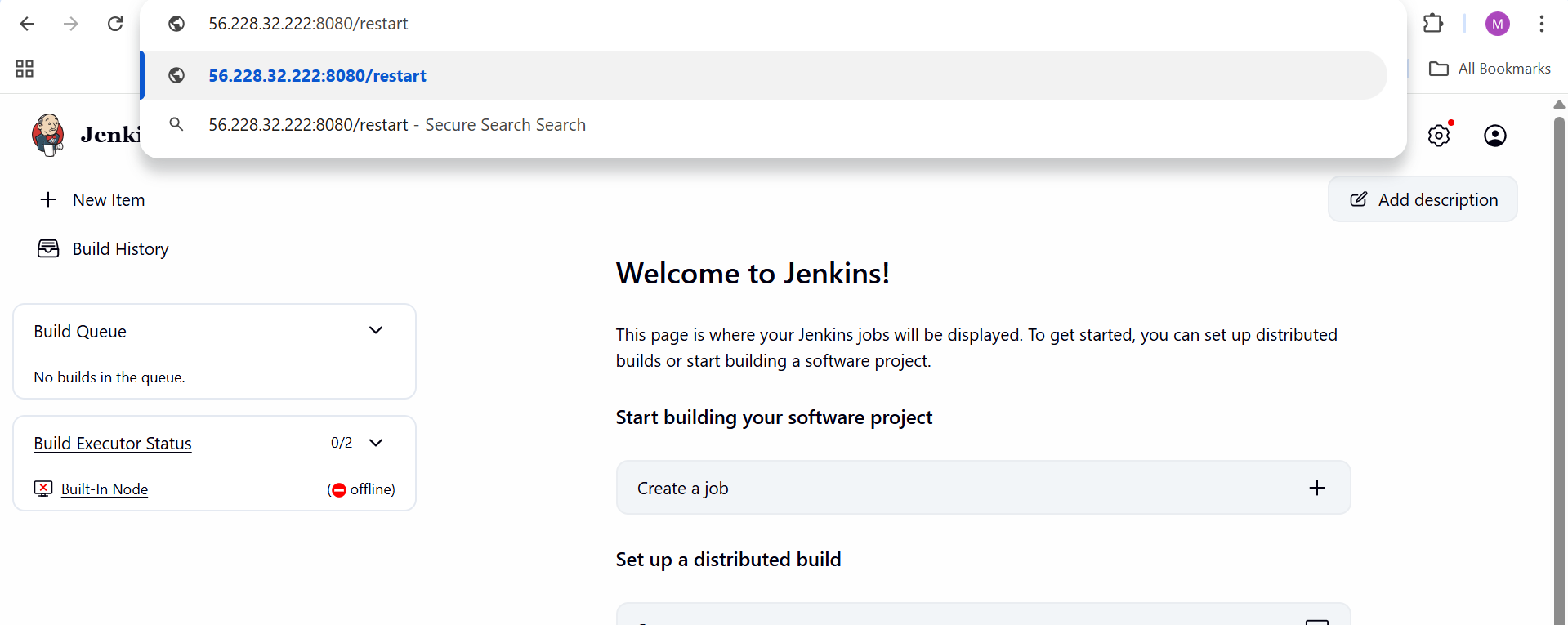












.