Deploying a Java Web Calculator using CI/CD Pipeline Documentation

1. Overview

This document explains how to set up a complete **CI/CD pipeline** for a Java Web calculator using **GitHub, Jenkins, SonarQube, Nexus, and Tomcat**.

The process automates:

- Code integration from GitHub
- Static analysis using SonarQube
- Build and packaging using Maven
- Artifact upload to Nexus
- Deployment to Tomcat

CI/CD Architecture: GitHub → Jenkins → SonarQube → Nexus → Tomcat

2. Prerequisites

Before you begin, ensure the following are available:

- A **GitHub** repository (public or private) containing the Java webapp (example: https://github.com/you/your-app)
- Four VMs (Ubuntu 22.04 LTS recommended) or cloud instances:
    - **Jenkins master** (orchestration & UI)
    - **SonarQube server** (analysis) — can double as build agent
    - **Nexus server** (artifact repo) — 3.227.246.21 used in examples
    - **Tomcat server** (application runtime) — 13.220.167.254 in examples
- Two Jenkins **agents** (workers) — we'll use the Sonar server as **SonarNode** (build + analysis) and Tomcat server as **TomcatNode** (deploy)
- Network: Jenkins must be able to reach Sonar, Nexus and Tomcat; agents must reach Nexus and Tomcat.
- Accounts and keys:

- SSH access between Jenkins master → agents (key auth)
- GitHub Personal Access Token (PAT) with repo and admin:repo_hook
- Nexus user for deploy
- Tomcat manager user (with manager-script role)
- Sonar token

3. Jenkins Setup (Detailed)

Step 1: Install Java

sudo apt update && sudo apt install -y openjdk-17-jdk

Step 2: Install Jenkins

sudo apt install fontconfig openjdk-21-jre

java -version

sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.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

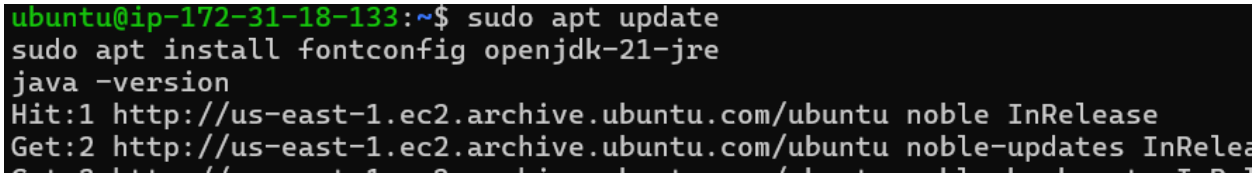
sudo apt update

sudo apt install jenkins

sudo systemctl enable jenkins

sudo systemctl start jenkins

sudo systemctl status jenkins



```
ubuntu@ip-172-31-18-133:~$ sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.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
--2025-11-04 04:41:02--  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.38.133, 2a04:4e42:78::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|146.75.38.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/etc/apt/keyrings/jenkins-keyring.asc'

/etc/apt/keyrings/jenkins-key 100%[=================================================>]   3.10K  --.-
```

```
ubuntu@ip-172-31-18-133:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-s
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-18-133:~$ sudo systemctl start jenkins
ubuntu@ip-172-31-18-133:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-11-04 04:41:37 UTC; 35s ago
   Main PID: 3963 (java)
      Tasks: 42 (limit: 1008)
     Memory: 315.8M (peak: 316.2M)
```

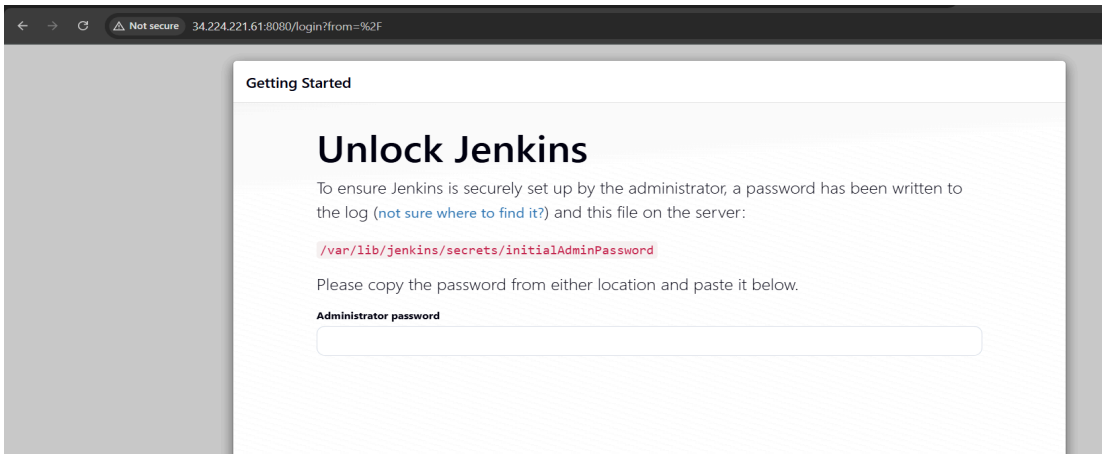
```
ubuntu@ip-172-31-18-133:~$ sudo hostnamectl set-hostname Jenkins
ubuntu@ip-172-31-18-133:~$ sudo init 6

Broadcast message from root@ip-172-31-18-133 on pts/1 (Tue 2025-11-04 04:46:01 UTC):

The system will reboot now!
```

Step 3: Access Jenkins

Open in browser → http://<jenkins-ip>:8080

Get unlock password:

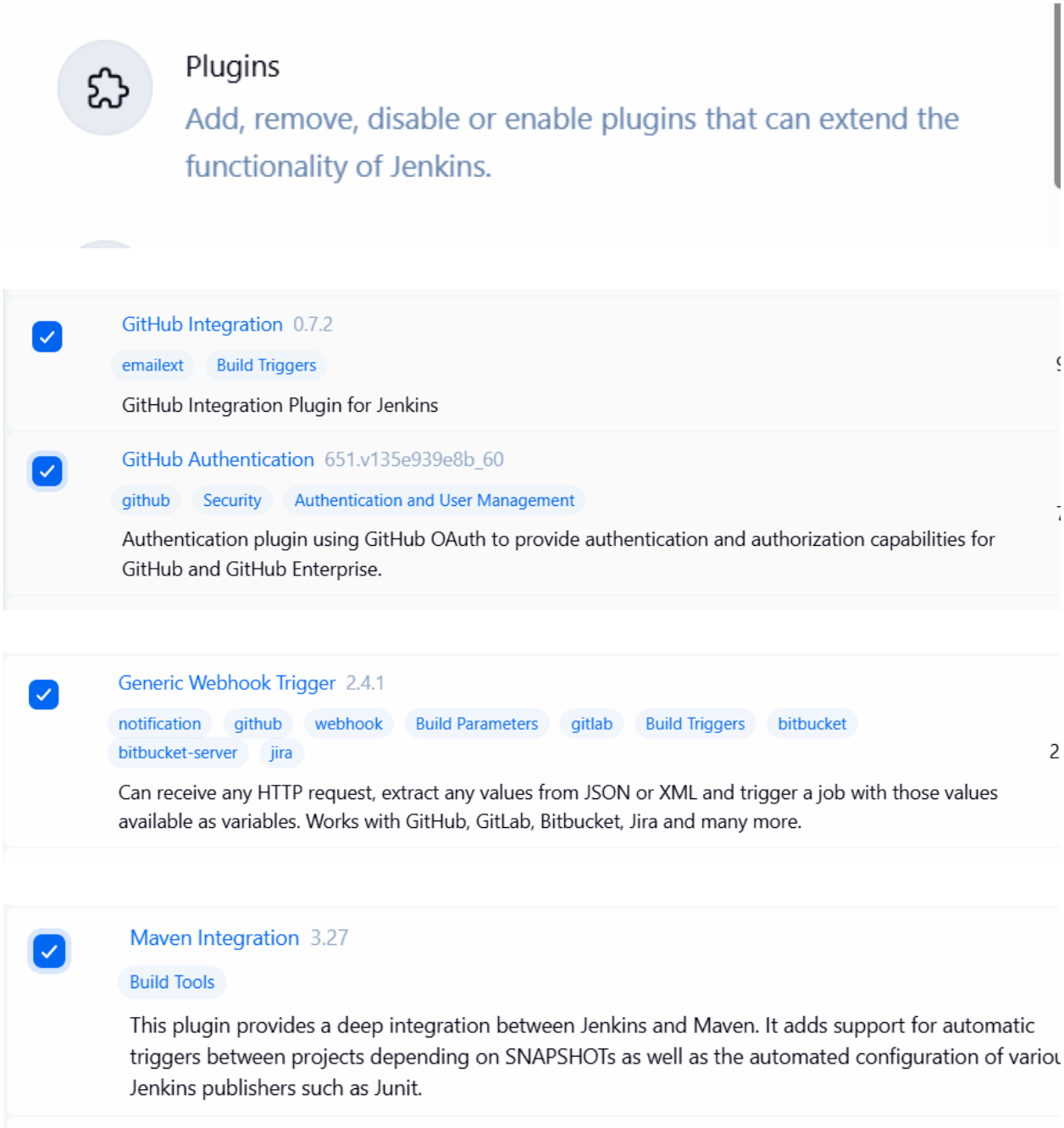
sudo cat /var/lib/jenkins/secrets/initialAdminPassword



```
ubuntu@Jenkins:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
0a8d7f48882d453bb973bbeb418dd9bd
ubuntu@Jenkins:~$
```

Step 4: Install Plugins

Click **“Plugins”** after the login.



Pipeline Maven Integration 1567.vb_2c3a_2116860

pipeline Maven

This plugin provides integration with Pipeline, configures maven environment to use within a pipeline job by calling sh mvn or bat mvn. The selected maven installation will be configured and prepended to the path.

SonarQube Scanner 2.18

External Site/Tool Integrations Build Reports

This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.

Sonar Quality Gates 352.vdcdb_d7994fb_6

Library plugins (for use by other plugins) analysis Other Post-Build Actions

Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")

Quality Gates 2.5

Fails the build whenever the Quality Gates criteria in the Sonar analysis aren't met (the project Quality Gates status is different than "Passed")

Warning: This plugin version may not be safe to use. Please review the following security notices:

- Credentials transmitted in plain text

Artifactory 4.0.8

pipeline

This plugin allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, and then have them linked to the build job that created them. The plugin includes a vast collection of features, including a rich pipeline API library and release management for Maven and Gradle builds with Staging and Promotion.

Deploy to container 1.17

Nexus Artifact Uploader 2.14

Artifact Uploaders

This plugin to upload the artifact to Nexus Repository.

This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

Artifact Deployer 1.3

Artifact Uploaders

This plugin makes it possible to deploy artifacts from workspace to output directories.

This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

SSH 158.ve2a_e90fb_7319

Build Wrappers

This plugin executes shell commands remotely using SSH protocol.

Warning: This plugin version may not be safe to use. Please review the following security notices:

- CSRF vulnerability and missing permission checks allow capturing credentials
- Missing permission check allows enumerating credentials IDs

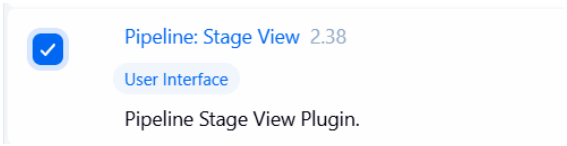
SSH Agent 386.v36cc0c7582f0

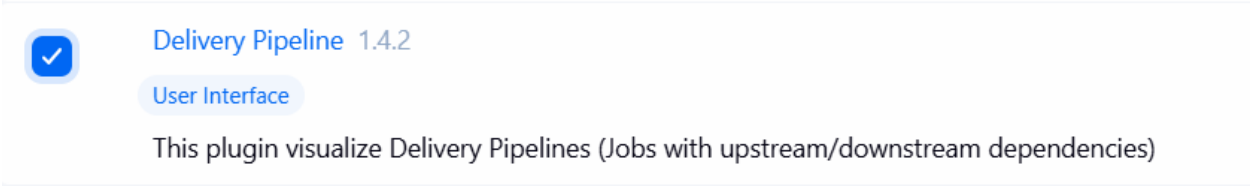
This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.

Publish Over SSH 390.vb_f56e7405751

Artifact Uploaders Build Tools

Send build artifacts over SSH

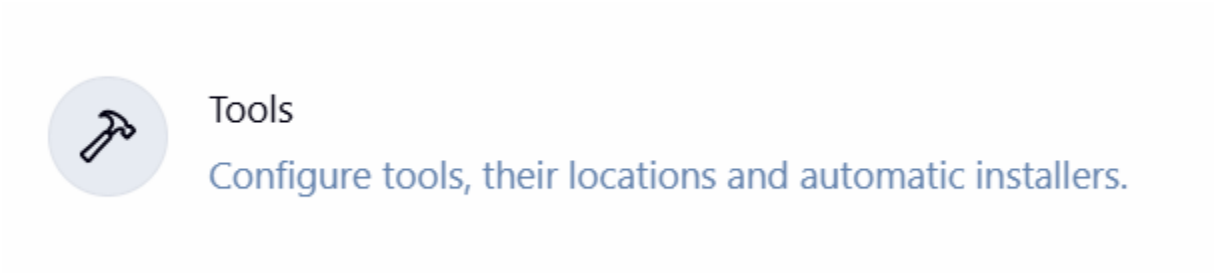


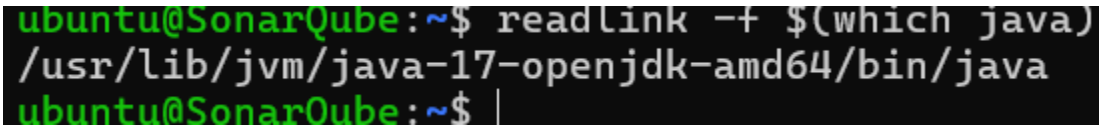


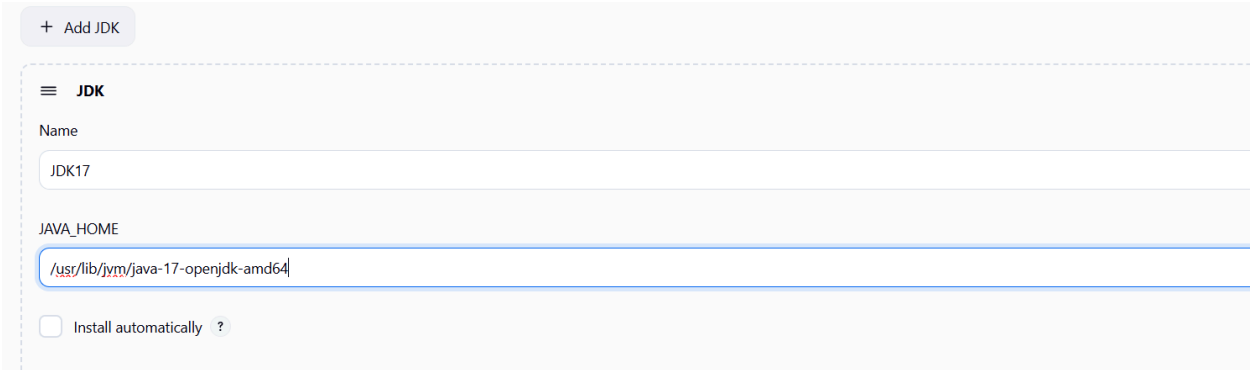
Step 5: Configure Global Tools

Go to **Manage Jenkins → Global Tool Configuration**

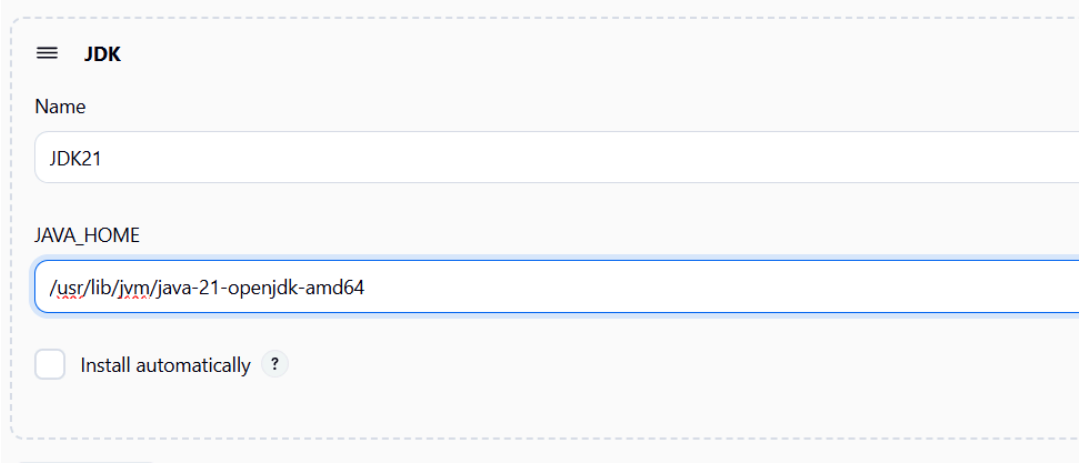
- Add JDK (Name: JDK17)
- Add JDK (Name: JDK21)
- Add Maven (Name: Maven)



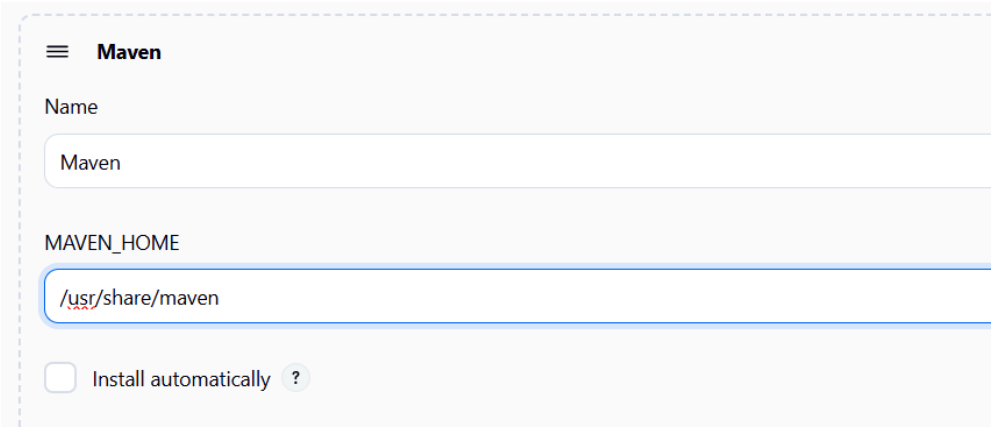




```
ubuntu@Jenkins:~$ readlink -f $(which java)
/usr/lib/jvm/java-21-openjdk-amd64/bin/java
ubuntu@Jenkins:~$
```




```
ubuntu@SonarQube:~$  readlink -f $(which mvn)
/usr/share/maven/bin/mvn
ubuntu@SonarQube:~$
```



Step 6: Create SSH Keys for Agents

Sudo su jenkins

ssh-keygen

Copy the public key into each agent server under:

~/.ssh/authorized_keys

Try to connect to the servers from the jenkins server:

Ssh ubuntu@publicip-of-the-agent-server

```
ubuntu@Jenkins:~$ sudo su jenkins
jenkins@Jenkins:/home/ubuntu$ cd
jenkins@Jenkins:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_ed25519):
Created directory '/var/lib/jenkins/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_ed25519
Your public key has been saved in /var/lib/jenkins/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:Em+Lsswn1vilUO7YQt9a24lFEeIiwbxNBhTcoNYaIDg jenkins@Jenkins
The key's randomart image is:
+--[ED25519 256]--+
|+  =*=  . .      |
|E. o+.+. . .     |
| .+ o=o . .      |
| . 8...+  .      |
```

SonarQube:

```
  GNU nano 7.2                                  authorized_keys *
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDaRu0y1AEdXFKMcoYSjQ+QRSQhgDVEDyqP6m3PCpHCu3wdxtzTNsNDe1gP5L4sZYjNjISQ8SHlkvrSJjd>

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGvAupKMk3BO4B/iH+zY/PmVPtyjzGAShPSVRcL7OBOz jenkins@Jenkins
```

```
jenkins@Jenkins:~$ ssh ubuntu@18.208.131.176
The authenticity of host '18.208.131.176 (18.208.131.176)' can't be established.
ED25519 key fingerprint is SHA256:wgB1/LMd7ycwrO39wSHHfBJCMOVbyAzKXoiYNMhh7DU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '18.208.131.176' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)
```

```
ubuntu@SonarQube:~$ exit
logout
Connection to 18.208.131.176 closed.
jenkins@Jenkins:~$
```

Tomcat:

```
jenkins@Jenkins:~$ ssh ubuntu@98.91.17.31
The authenticity of host '98.91.17.31 (98.91.17.31)' can't be established.
ED25519 key fingerprint is SHA256:D/IAnan7cjP0pmgEbu/tPZlwIgR27Rpk6MfqS0nZD5o.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '98.91.17.31' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
```

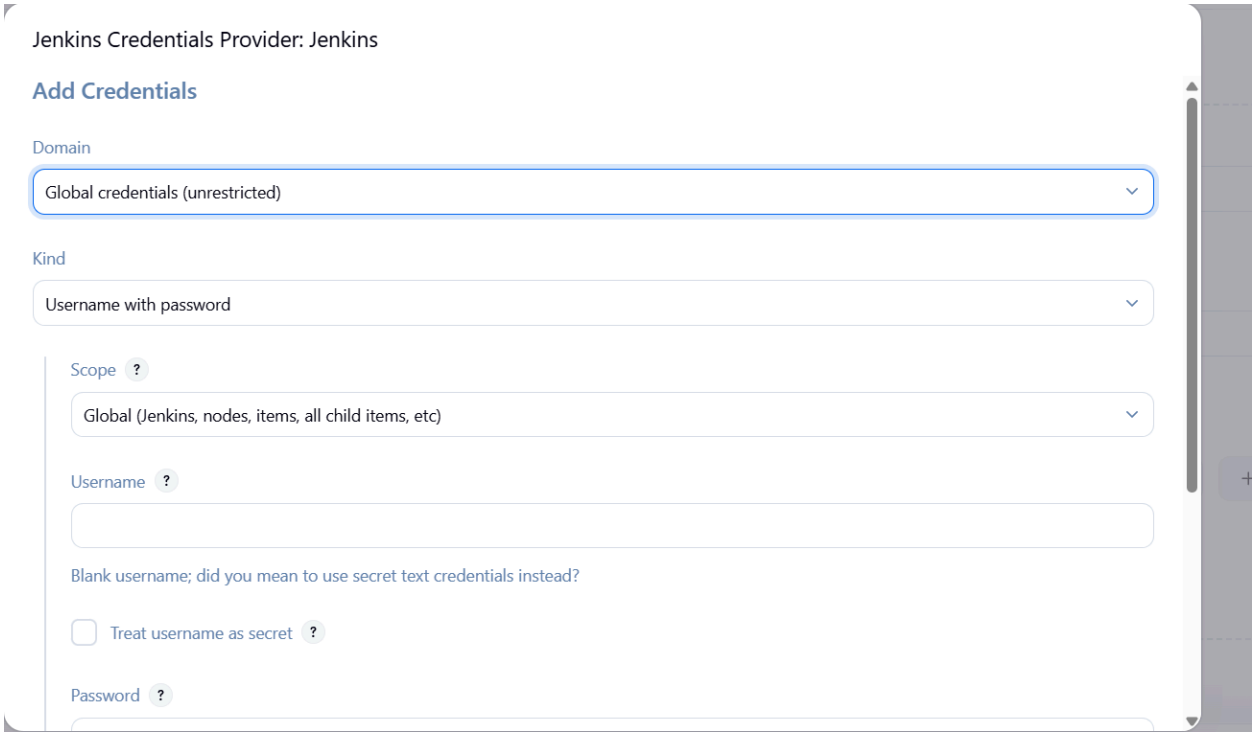
```
ubuntu@ip-172-31-23-240:~$ exit
logout
Connection to 98.91.17.31 closed.
jenkins@Jenkins:~$
```

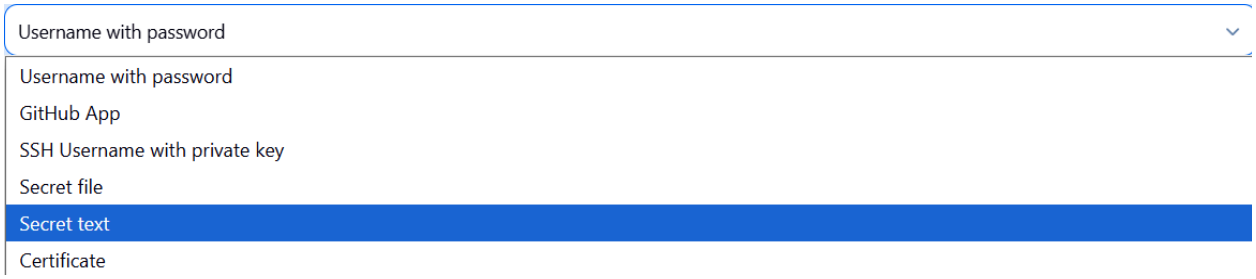
Step 7: Add Credentials in Jenkins

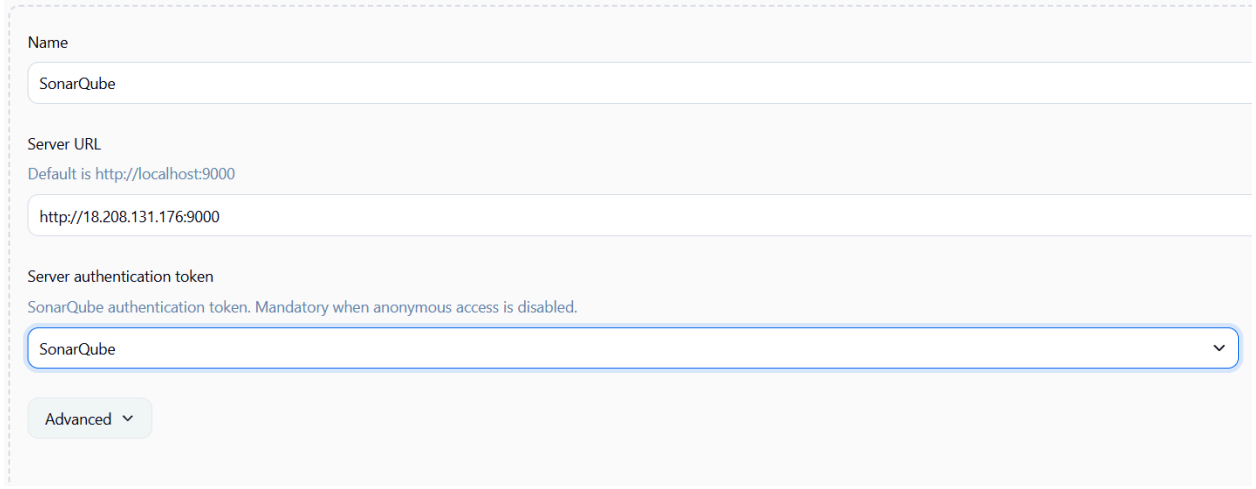
Go to **Manage Jenkins → Credentials → System → Global → Add Credentials**

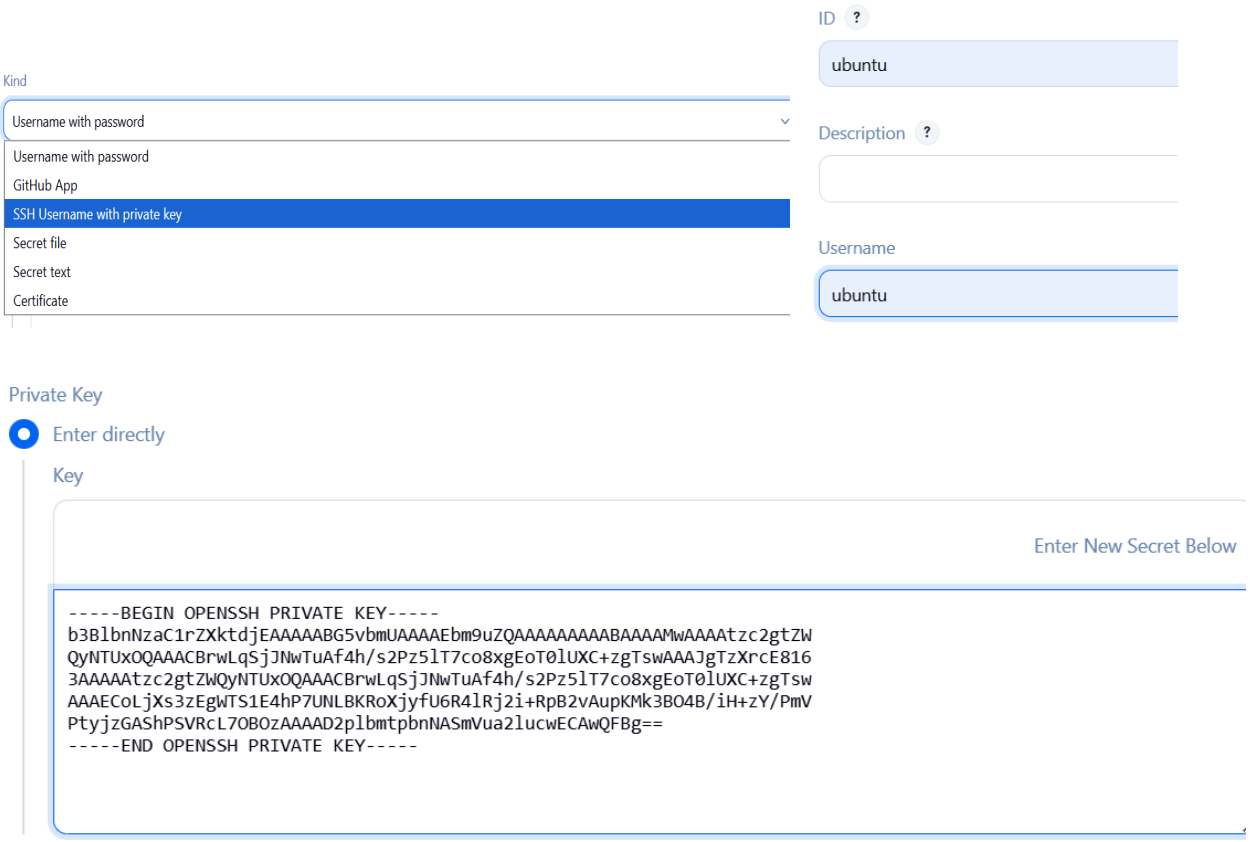
Create the following:

- ssh-ubuntu → SSH Username with private key
- nexus-creds → Username & password for Nexus
- tomcat-manager → Tomcat manager credentials
- github-token → GitHub PAT


Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
Username with password
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
Blank username; did you mean to use secret text credentials instead?
Treat username as secret
Password


Username with password
Username with password
GitHub App
SSH Username with private key
Secret file
Secret text
Certificate


Name
SonarQube
Server URL
Default is http://localhost:9000
http://18.208.131.176:9000
Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
SonarQube
Advanced

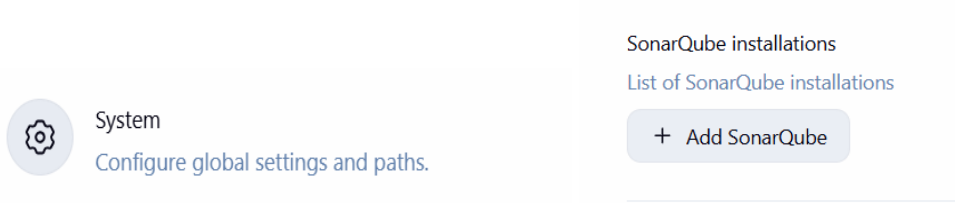


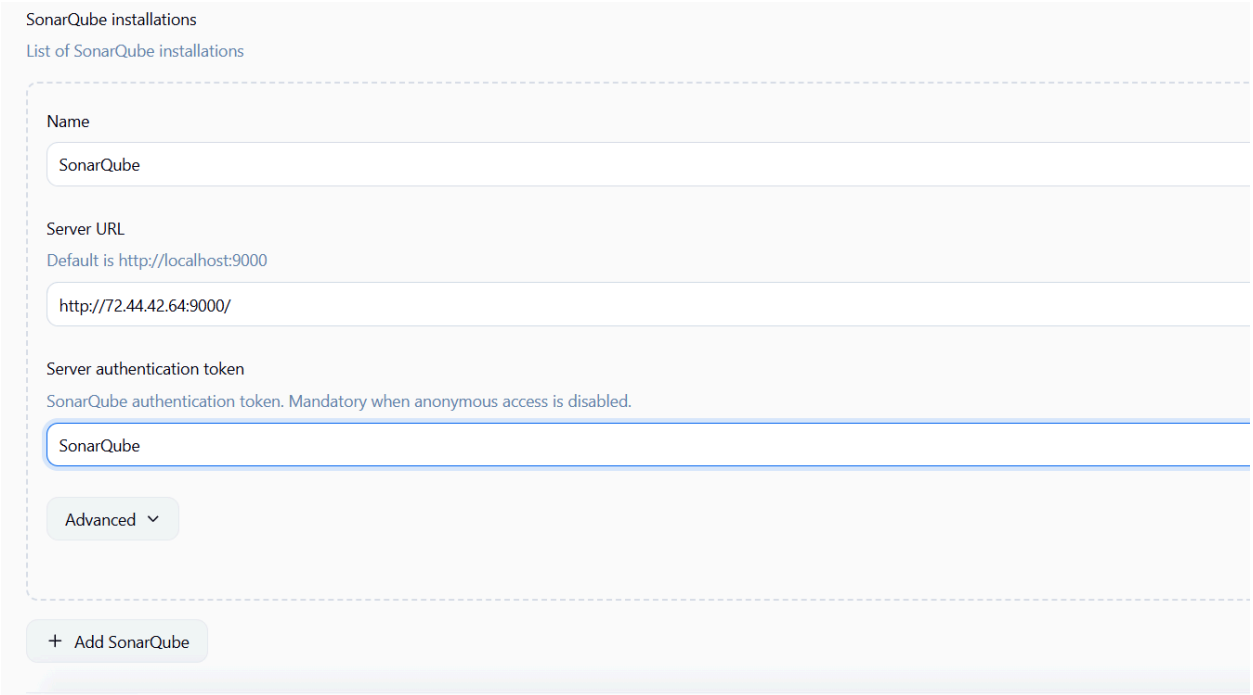
Step 8: Configure SonarQube in Jenkins

Go to **Manage Jenkins → Configure System → SonarQube servers**

Add:

- Name: SonarQube
- Server URL: http://<sonar-ip>:9000
- Token: paste from SonarQube UI



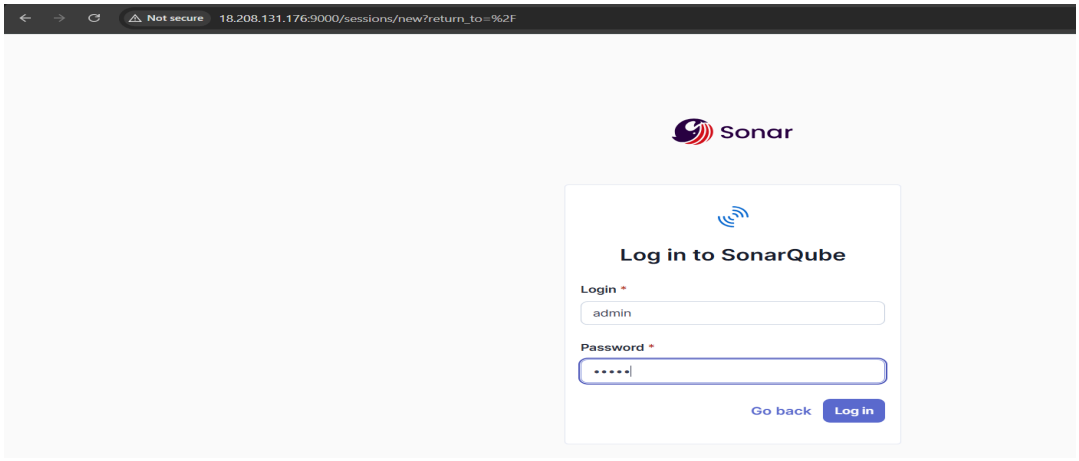


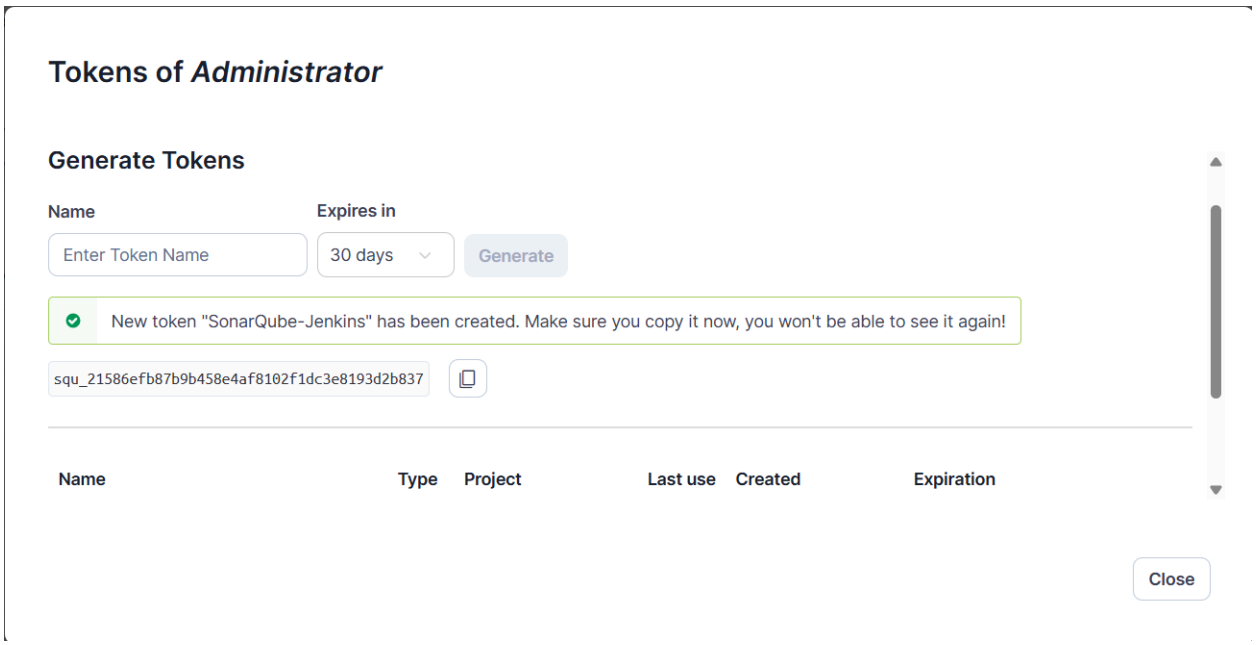
4. SonarQube Setup (Summary)

- Install SonarQube with Java 17.
- Create a jenkins directory so that jenkins can use that as the workspace.
- Extract the package.
- Access SonarQube at http://<sonar-ip>:9000.
- Login with admin/admin, create a new token, and use it in Jenkins.

```
ubuntu@ip-172-31-30-72:~$ sudo apt update
Hit: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 [12
```

```
ubuntu@SonarQube:~$ mkdir jenkins
ubuntu@SonarQube:~$ ls
jenkins  sonarqube-24.12.0.100206  sonarqube-24.12.0.100206.zip
ubuntu@SonarQube:~$ ls -a
.   .bash_history  .bashrc  .profile  .sudo_as_admin_successful  sonarqube-24.12.0.100206
..  .bash_logout   .cache   .ssh      jenkins                    sonarqube-24.12.0.100206.zip
ubuntu@SonarQube:~$ cd .ssh/
ubuntu@SonarQube:~/.ssh$ ls
authorized_keys
ubuntu@SonarQube:~/.ssh$
```





5. Nexus Setup (Summary)

- **Install Java 17**

sudo apt update

sudo apt install -y openjdk-17-jdk wget tar

- **Download & extract Nexus**

cd /opt

wget https://download.sonatype.com/nexus/3/nexus-3.85.0-03-linux-x86_64.tar.gz

sudo tar -xzf nexus-latest.tar.gz

sudo mv nexus-3* nexus

sudo useradd -r -s /bin/false nexus

sudo chown -R nexus:nexus /opt/nexus /opt/sonatype-work

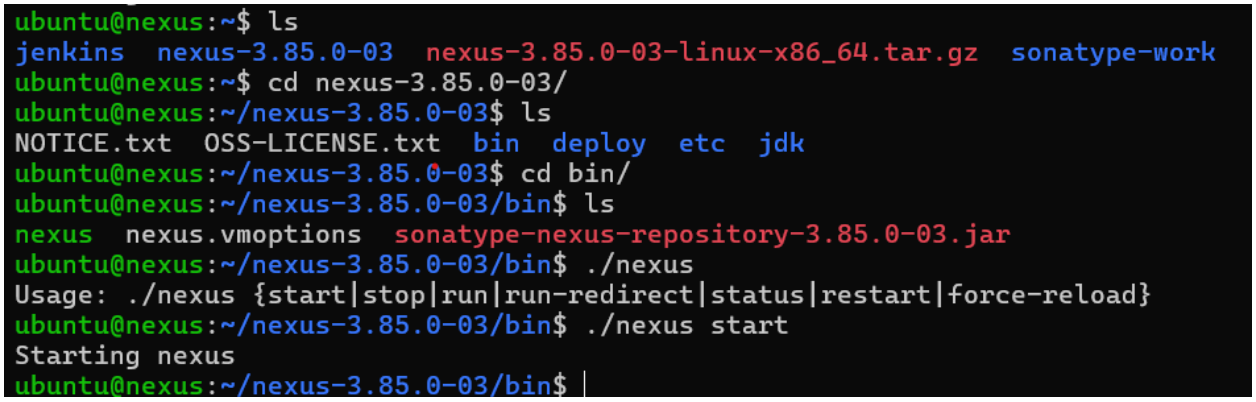
echo 'run_as_user="nexus"' | sudo tee /opt/nexus/bin/nexus.rc

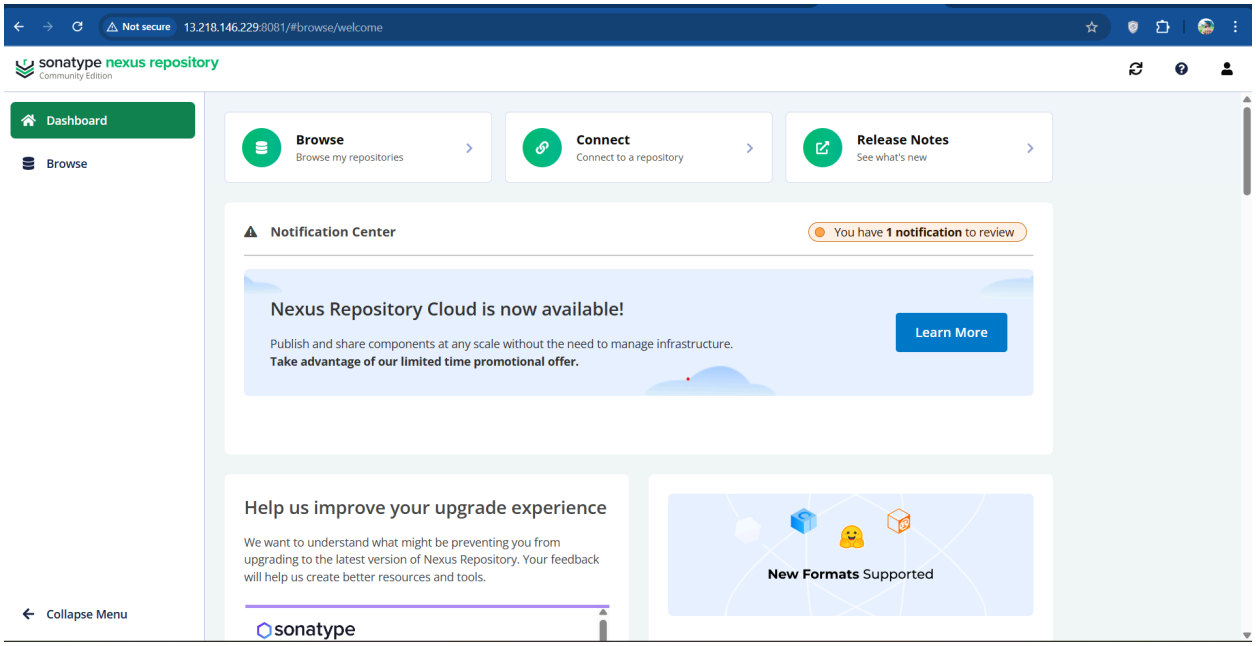
sudo systemctl daemon-reload

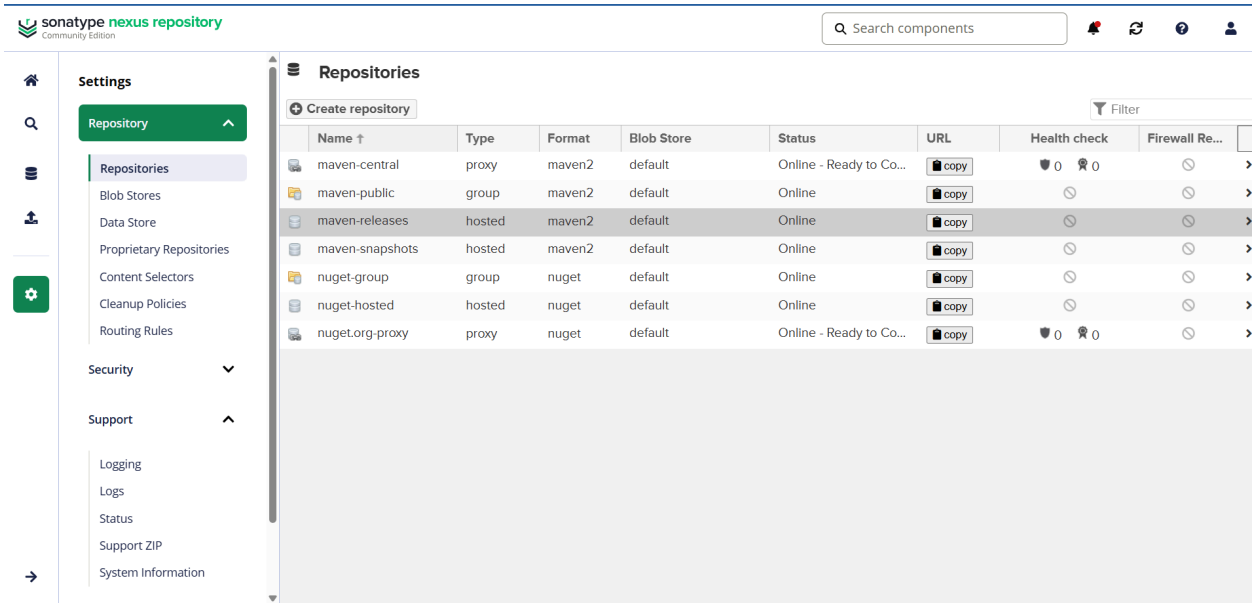
sudo systemctl enable --now nexus

sudo systemctl status nexus

- **Validate & create repository**
  Open http://NEXUS_IP:8081. Initial admin password: /opt/sonatype-work/nexus3/admin.password.
  Create hosted Maven repo maven-releases: **Administration → Repositories → Create repository → maven2 (hosted)** → name maven-releases







6. Tomcat Setup (Summary)

- **Install Tomcat**

sudo apt update

sudo apt install -y tomcat9 tomcat9-admin

- **Create Tomcat manager user**
  Append to /etc/tomcat9/tomcat-users.xml:

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<user username="admin" password="admin123" roles="manager-gui,manager-script"/>

Restart Tomcat:

sudo systemctl restart tomcat9

sudo systemctl status tomcat9

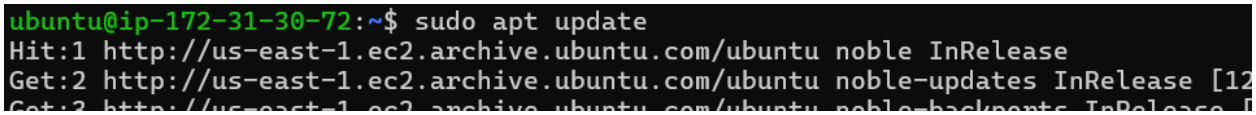
- **Validate**
  From an agent:

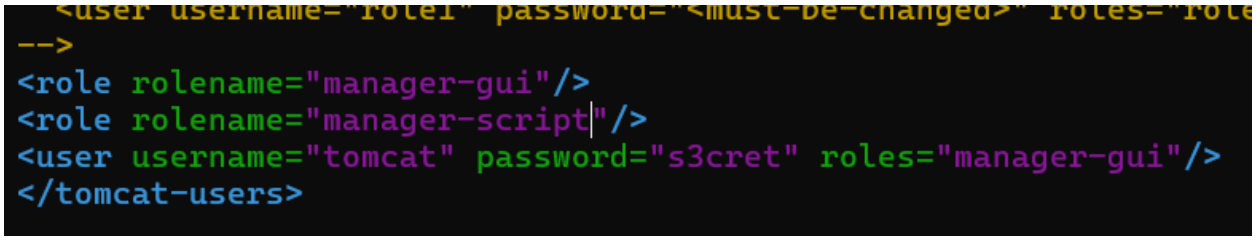
curl -u admin:admin123 http://TOMCAT_IP:8080/manager/text/list

Expect an OK response including contexts

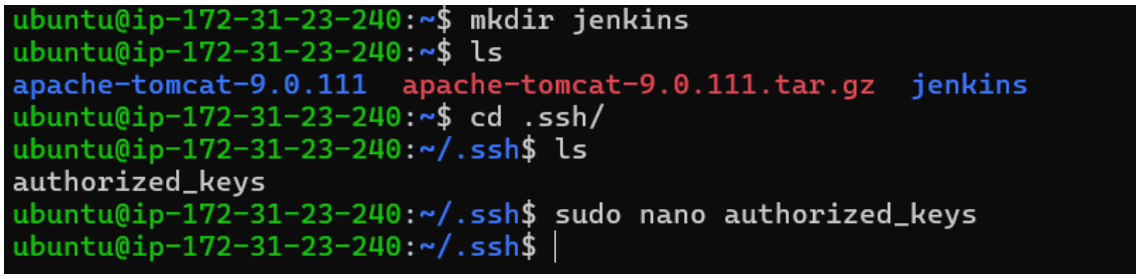
If connection fails, open port 8080 in the firewall / cloud security group.

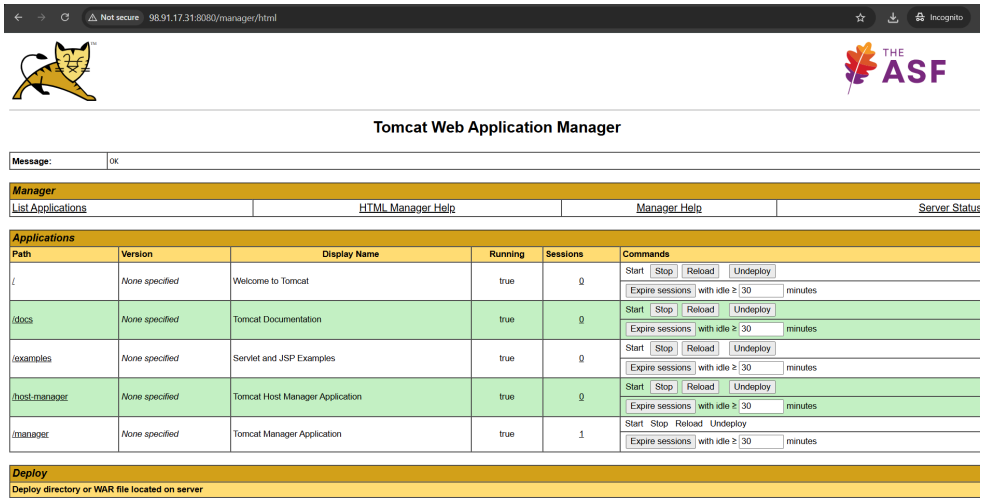
- Create a jenkins directory so that jenkins can use that as the workspace.






```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
</tomcat-users>
```



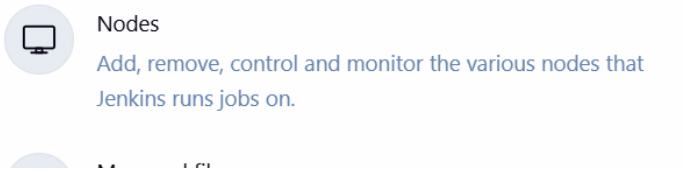


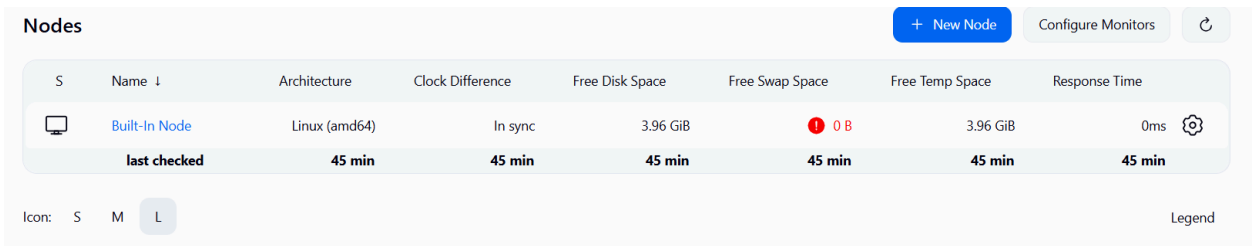
7.Jenkins Nodes configuration

Go to **Manage Jenkins → Configure Nodes → New Nodes**

Add SonarQube:

- Name: SonarQube
- Type: Permanent Agent
- Remote root directory: /home/ubuntu/jenkins
- Labels: sonarqube
- Launch method: Launch agents via SSH
- Host: sonar-Public ip
- Credentials: ubuntu or your own id name
- Host key Verification strategy: Non verifying Verification Strategy.





New node

Node name

SonarQube

Type

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

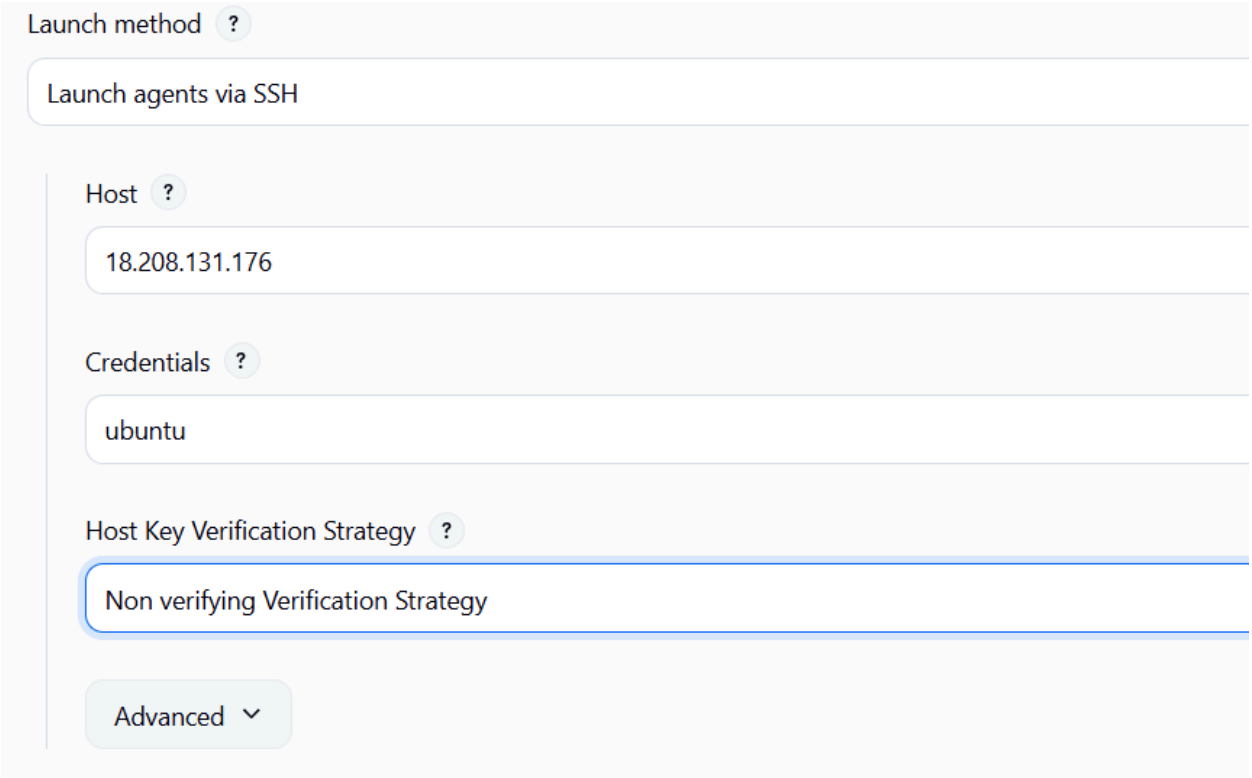
Remote root directory ?

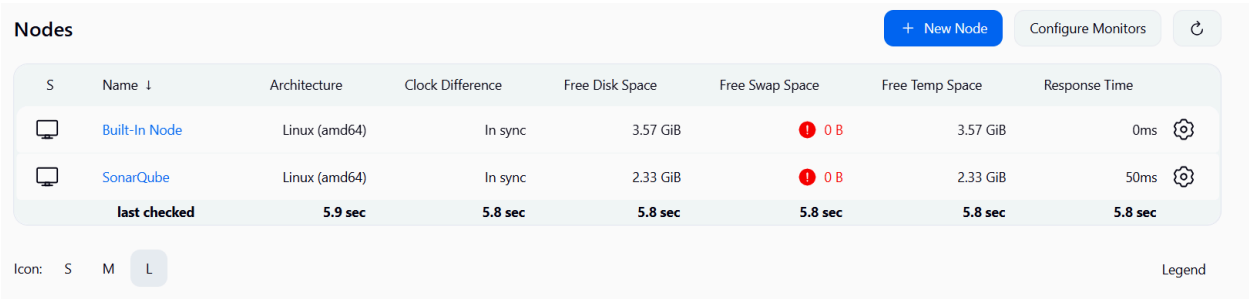
/home/ubuntu/jenkins

Remote directory is mandatory

Labels ?

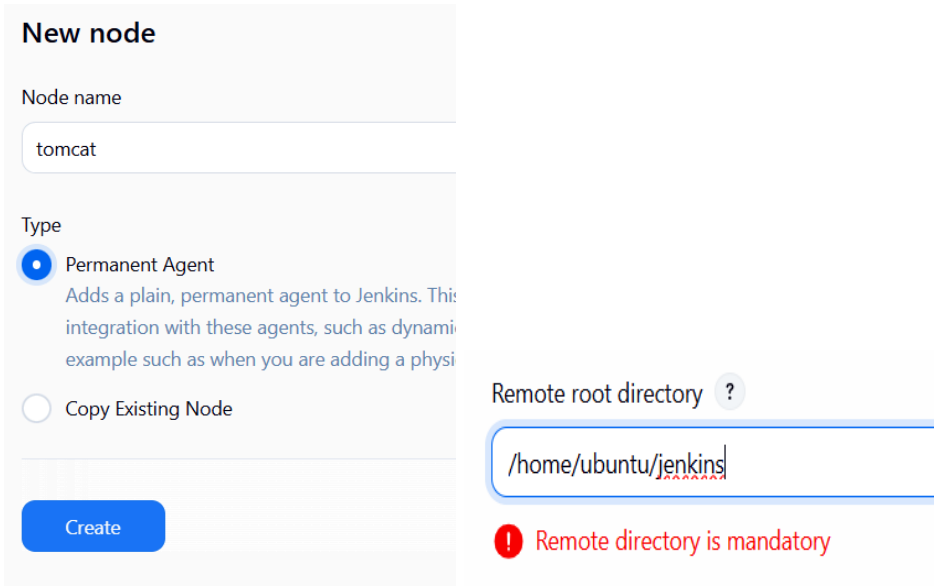
sonarqube

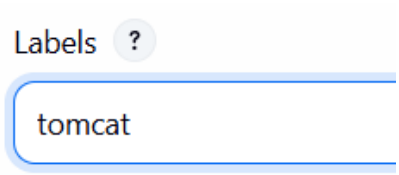


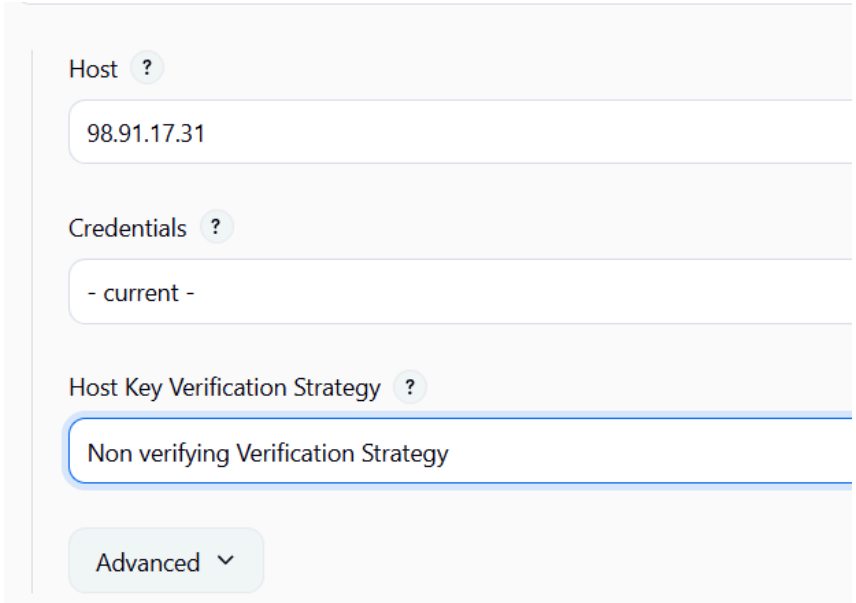


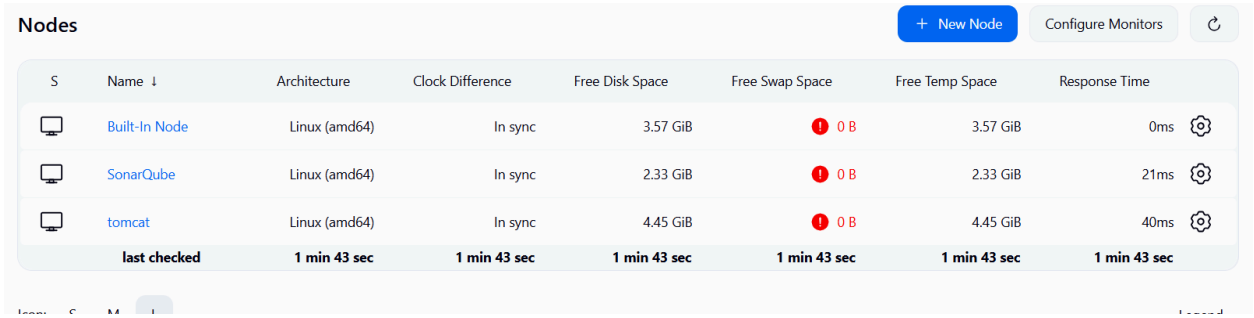
Add Tomcat:

- Name: tomcat
- Type: Permanent Agent
- Remote root directory: /home/ubuntu/jenkins
- Labels: tomcat
- Launch method: Launch agents via SSH
- Host: tomcat-Public ip
- Credentials: ubuntu or your own id name
- Host key Verification strategy: Non verifying Verification Strategy.


New node
Node name
tomcat
Type
Permanent Agent
Adds a plain, permanent agent to Jenkins. This
integration with these agents, such as dynami
example such as when you are adding a physi
Copy Existing Node
Create
Remote root directory
/home/ubuntu/jenkins
Remote directory is mandatory

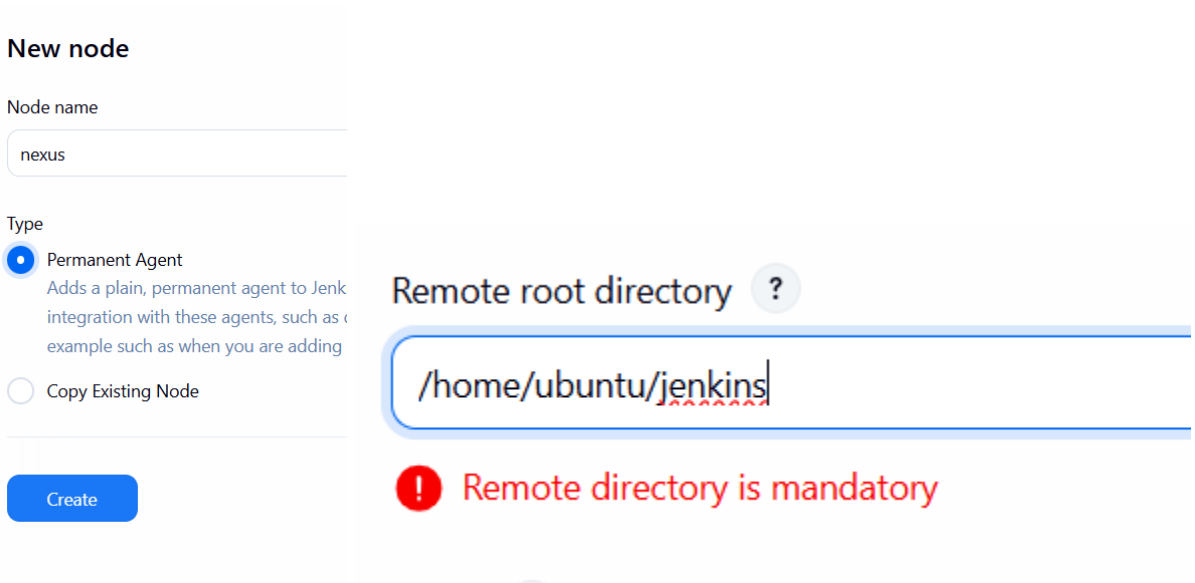

Labels
tomcat

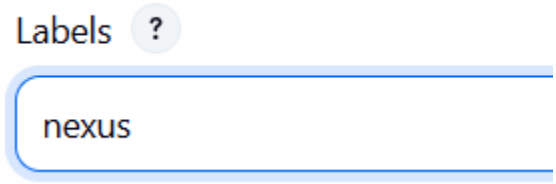

Host
98.91.17.31
Credentials
- current -
Host Key Verification Strategy
Non verifying Verification Strategy
Advanced

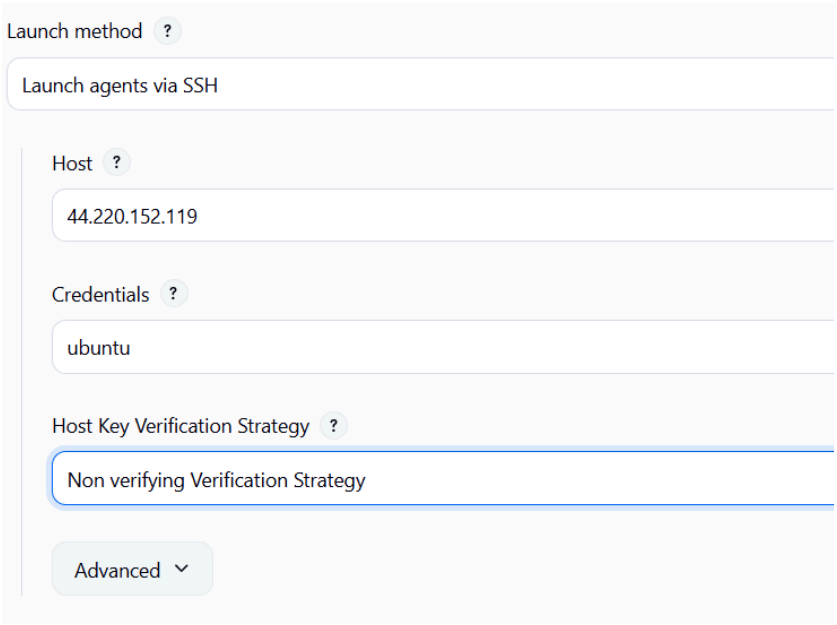


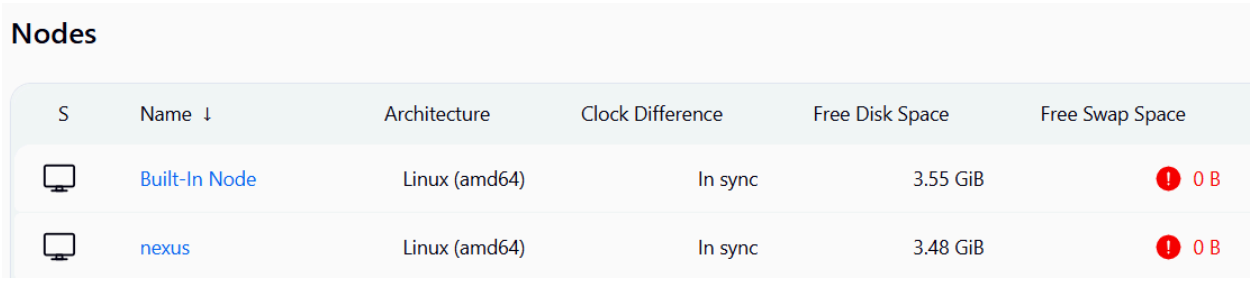
Add Nexus:

- Name: nexus
- Type: Permanent Agent
- Remote root directory: /home/ubuntu/jenkins
- Labels: nexus
- Launch method: Launch agents via SSH
- Host: nexus-Public ip
- Credentials: ubuntu or your own id name
- Host key Verification strategy: Non verifying Verification Strategy.

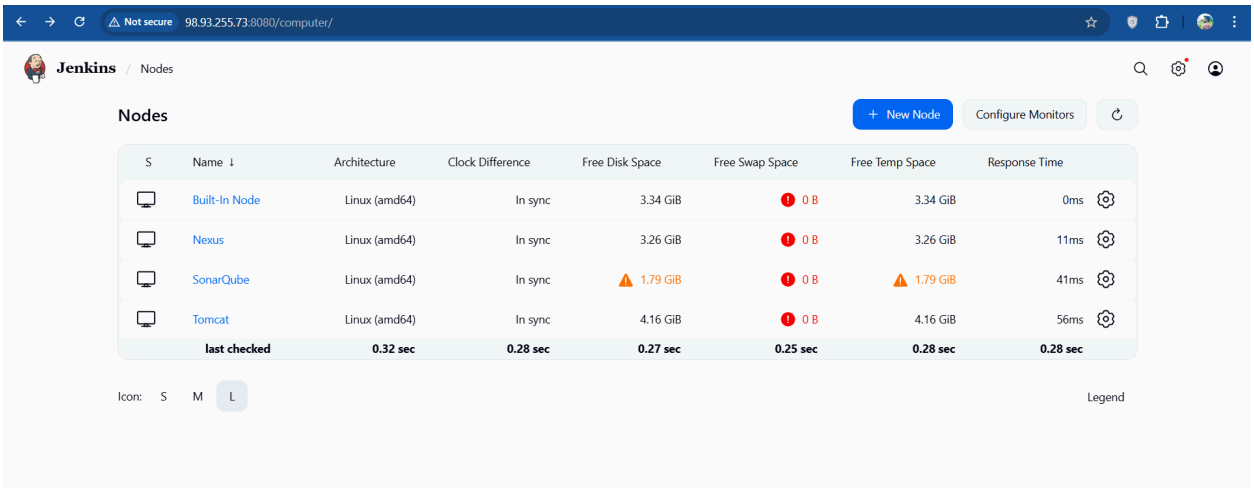








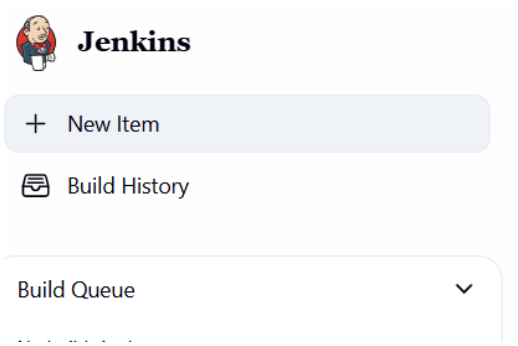
After building all the nodes this is how your Nodes should look line:

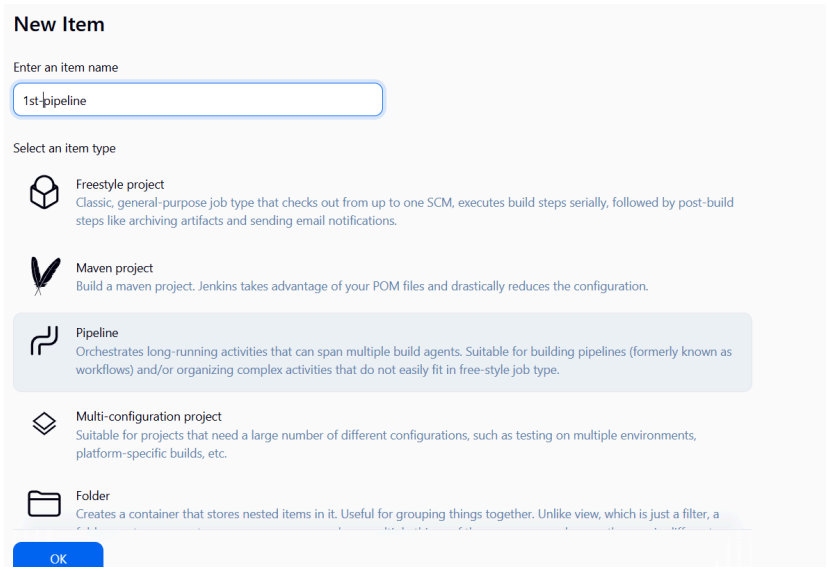


8. Jenkins Pipeline Configuration

In Jenkins:

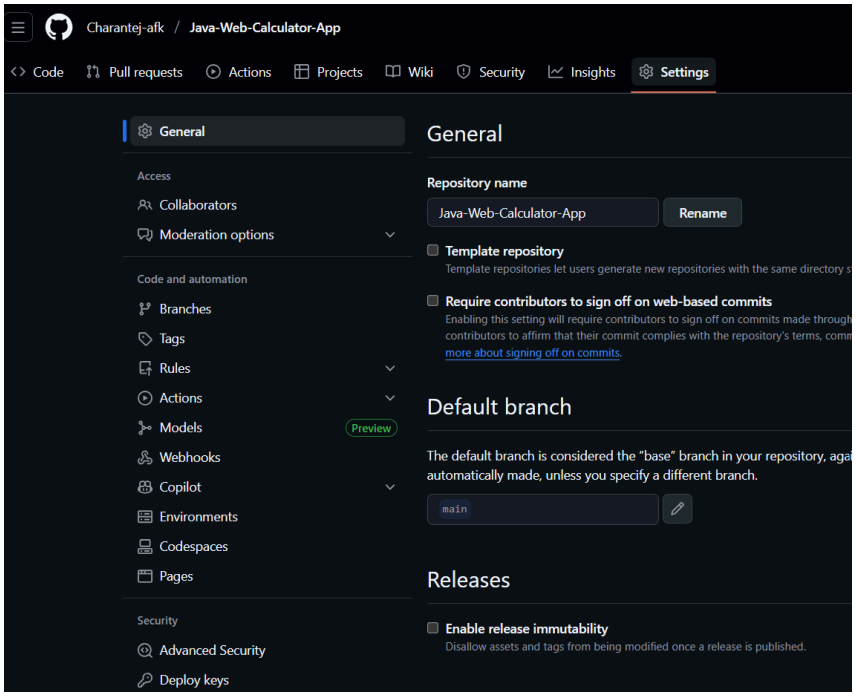
- Create a new item → **Pipeline**
- Definition: *Pipeline script from SCM*
- SCM: Git
- Repository URL: https://github.com/<your-repo>.git
- Branches: */main
- Script Path: Jenkinsfile

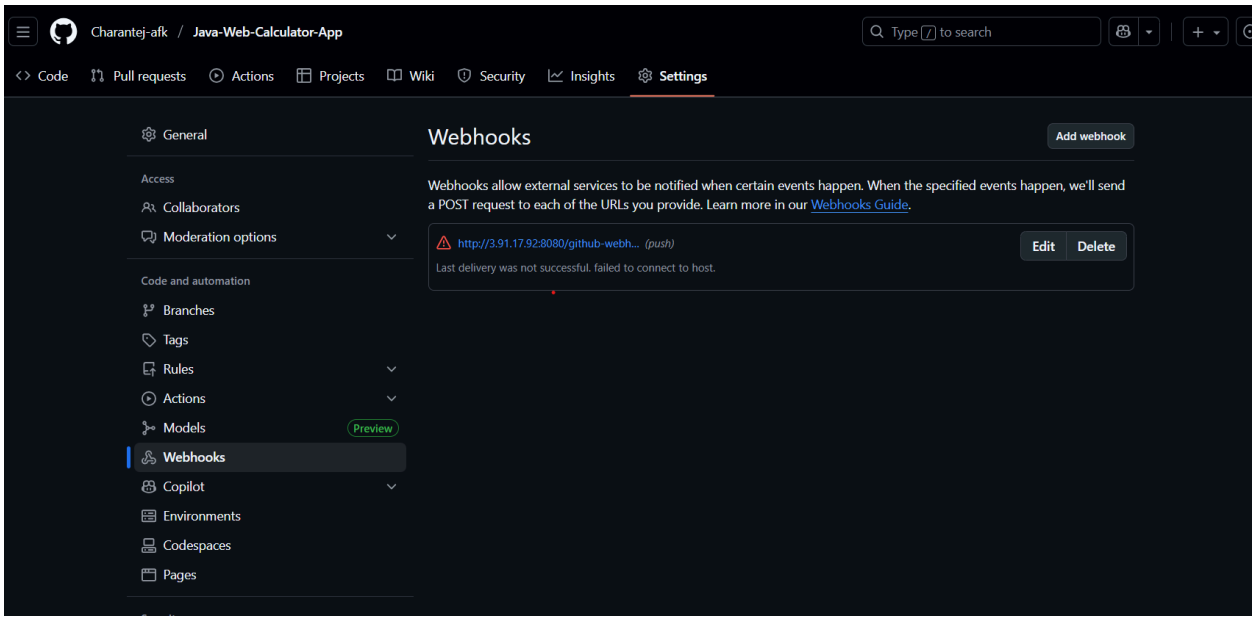


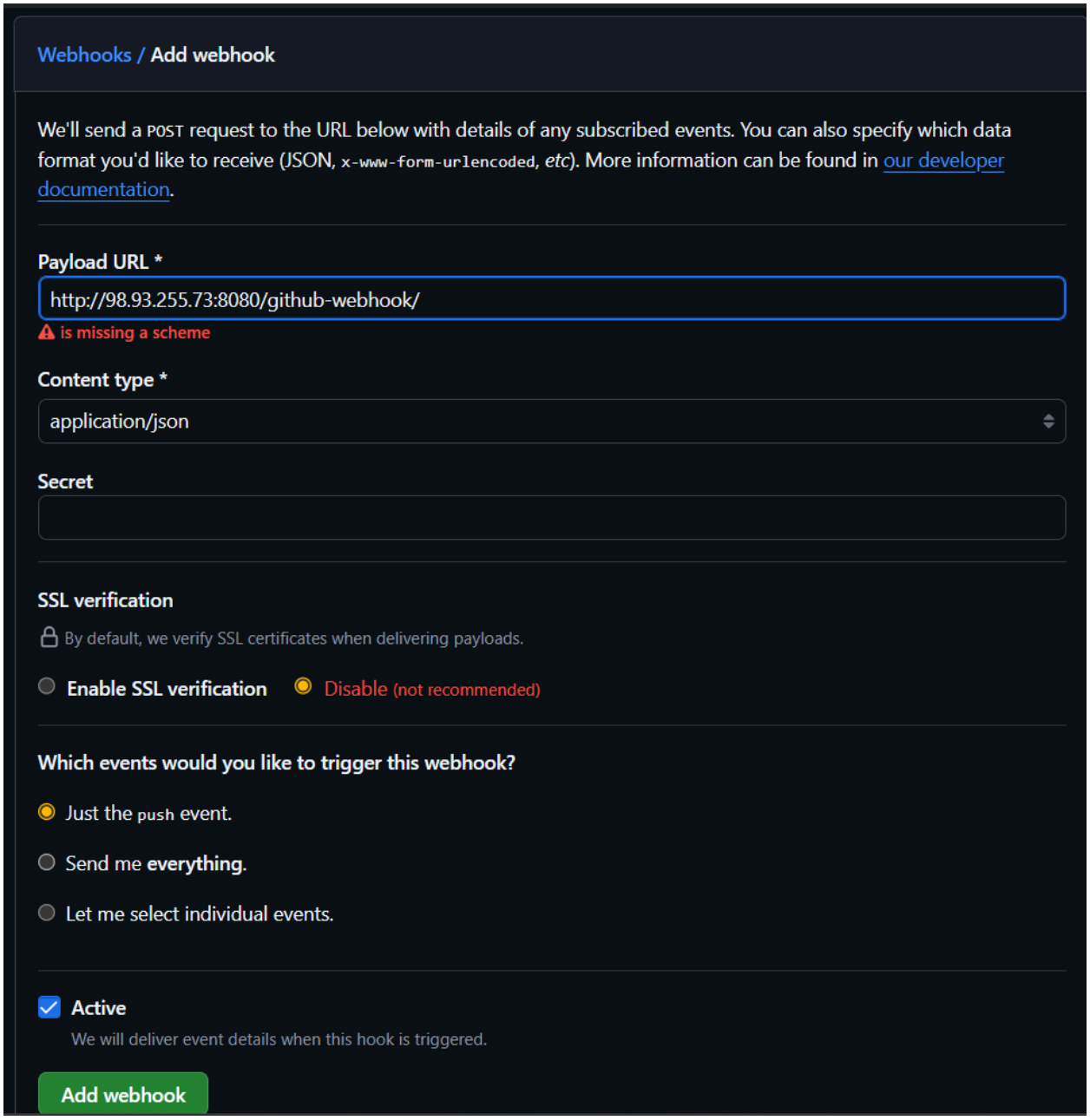


8. GitHub Webhook Integration

- Go to **GitHub → Repo → Settings → Webhooks → Add Webhook**
- Payload URL: http://<jenkins-public-ip>:8080/github-webhook/
- Content Type: application/json
- Event: **Push event**
- Save Webhook


Charantej-afk / Java-Web-Calculator-App
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Models
Preview
Webhooks
Copilot
Environments
Codespaces
Pages
Security
Advanced Security
Deploy keys
General
Repository name
Java-Web-Calculator-App
Rename
Template repository
Require contributors to sign off on web-based commits
more about signing off on commits.
Default branch
The default branch is considered the "base" branch in your repository, agai
automatically made, unless you specify a different branch.
main
Releases
Enable release immutability
Disallow assets and tags from being modified once a release is published.


Charantej-afk / Java-Web-Calculator-App
Type / to search
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Models
Preview
Webhooks
Copilot
Environments
Codespaces
Pages
Webhooks
Add webhook
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.
http://3.91.17.92:8080/github-webh... (push)
Last delivery was not successful. failed to connect to host.
Edit
Delete



9 — Project changes: POM snippets & Maven settings (on build agent)

Using git clone URL get the files into the Build server.

9A — Add distributionManagement to pom.xml

So mvn deploy pushes to the right repo.

<distributionManagement>

<repository>

```
    <id>maven-releases</id>
    <url>http://3.227.246.21:8081/repository/maven-releases/</url>
  </repository>
</distributionManagement>
```

9B — Add versions plugin in pom.xml

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>versions-maven-plugin</artifactId>
  <version>2.16.0</version>
</plugin>
```

9C — Create settings.xml on build agent (path used in pipeline)

Path used in Jenkinsfile: /home/jenkins/.m2/settings.xml (or use /home/ubuntu/.m2 if agent runs as ubuntu). Example:

```
<settings>
  <servers>
    <server>
      <id>maven-releases</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
  </servers>
</settings>
```

Commands (on SonarNode if it runs Maven):

```
sudo mkdir -p /home/jenkins/.m2
sudo tee /home/jenkins/.m2/settings.xml > /dev/null <<'XML'
<settings>
  <servers>
```

```
    <server>
      <id>maven-releases</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
  </servers>
</settings>
```

XML

```
sudo chown -R jenkins:jenkins /home/jenkins/.m2 || sudo chown -R ubuntu:ubuntu /home/jenkins/.m2
```

Important: pipeline uses --settings ${MVN_SETTINGS}, so this file must be readable by the OS user running the agent process.

Jenkinsfile (complete & final)

Put this file at repo root as Jenkinsfile. This file assumes credential IDs and node labels we created earlier.

```
pipeline {
    agent { label 'SonarQube' }

    tools {
        jdk 'JDK17'
        maven 'Maven'
    }

    environment {
        SONARQUBE_SERVER = 'SonarQube'
        MVN_SETTINGS = '/etc/maven/settings.xml'
        NEXUS_URL = 'http://18.226.34.227:8081'
```

```
        NEXUS_REPO = 'maven-releases'
        NEXUS_GROUP = 'com/web/cal'
        NEXUS_ARTIFACT = 'webapp-add'
        TOMCAT_URL = 'http://18.216.0.11:8080/manager/text'
    }

    stages {
        /* === Stage 1: Checkout Code === */
        stage('Checkout Code') {
            steps {
                echo '📦 Cloning source from GitHub...'
                checkout([$class: 'GitSCM',
                    branches: [[name: '*/main']],
                    userRemoteConfigs: [[url:
'https://github.com/mrtechreddy/Java-Web-Calculator-App.git']]
                ])
            }
        }

        /* === Stage 2: SonarQube Analysis === */
        stage('SonarQube Analysis') {
            steps {
                echo '🔍 Running SonarQube static analysis...'
                withSonarQubeEnv("${SONARQUBE_SERVER}") {
                    sh 'mvn clean verify sonar:sonar -DskipTests --settings ${MVN_SETTINGS}'
                }
            }
```

```
    }

    /* === Stage 3: Build Artifact === */
    stage('Build Artifact') {
      steps {
        echo '⚙️ Building WAR...'
        sh 'mvn clean package -DskipTests --settings ${MVN_SETTINGS}'
        sh 'echo ✅ Build Completed!'
        sh 'ls -lh target/*.war || true'
      }
    }

    /* === Stage 4: Upload Artifact to Nexus (via REST API) === */
    stage('Upload Artifact to Nexus') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'Nexus', usernameVariable:
'NEXUS_USR', passwordVariable: 'NEXUS_PSW')]) {
          sh '''#!/bin/bash
            set -e
            WAR_FILE=$(ls target/*.war | head -1)
            FILE_NAME=$(basename "$WAR_FILE")
            VERSION="0.0.${BUILD_NUMBER}"

            echo "📤 Uploading $FILE_NAME to Nexus as version $VERSION..."

            curl -v -u ${NEXUS_USR}:${NEXUS_PSW} --upload-file "$WAR_FILE" \
```

```
"${NEXUS_URL}/repository/${NEXUS_REPO}/${NEXUS_GROUP}/${NEXUS_ARTIFACT}/${VERSION}/${NEXUS_ARTIFACT}-${VERSION}.war"


                    echo "✅ Artifact uploaded successfully to Nexus!"
                '''
            }
        }
    }

    /* === Stage 5: Deploy to Tomcat === */
    stage('Deploy to Tomcat') {
        agent { label 'Tomcat' }
        steps {
            withCredentials([
                usernamePassword(credentialsId: 'Nexus', usernameVariable: 'NEXUS_USR', passwordVariable: 'NEXUS_PSW'),
                usernamePassword(credentialsId: 'Tomcat', usernameVariable: 'TOMCAT_USR', passwordVariable: 'TOMCAT_PSW')
            ]) {
                sh '''#!/bin/bash
                    set -e
                    cd /tmp; rm -f *.war

                    echo "🔍 Fetching latest WAR from Nexus..."
                    DOWNLOAD_URL=$(curl -s -u ${NEXUS_USR}:${NEXUS_PSW} \
```

```
"${NEXUS_URL}/service/rest/v1/search?repository=${NEXUS_REPO}&group=com.web.cal&name=webapp-add" \
                    | grep -oP '"downloadUrl"\\s*:\\s*"\\K[^"]+\\.war' | grep -vE '\\.md5|\\.sha1' | tail -1)

                if [[ -z "$DOWNLOAD_URL" ]]; then
                  echo "❌ No WAR found in Nexus!"
                  exit 1
                fi

                echo "⬇️ Downloading WAR: $DOWNLOAD_URL"
                curl -u ${NEXUS_USR}:${NEXUS_PSW} -O "$DOWNLOAD_URL"
                WAR_FILE=$(basename "$DOWNLOAD_URL")
                APP_NAME=$(echo "$WAR_FILE" | sed 's/-[0-9].*//')

                echo "🧹 Removing old deployment..."
                curl -u ${TOMCAT_USR}:${TOMCAT_PSW} "${TOMCAT_URL}/undeploy?path=/${APP_NAME}" || true

                echo "🚀 Deploying new WAR to Tomcat..."
                curl -u ${TOMCAT_USR}:${TOMCAT_PSW} --upload-file "$WAR_FILE" \
                  "${TOMCAT_URL}/deploy?path=/${APP_NAME}&update=true"

                echo "✅ Deployment successful! Application updated."
            '''
        }
    }
```

```
    }
  }

  post {
    success { echo '🎉 Pipeline completed successfully — Application live on Tomcat!' }
    failure { echo '❌ Pipeline failed — Check Jenkins logs.' }
  }
}
```

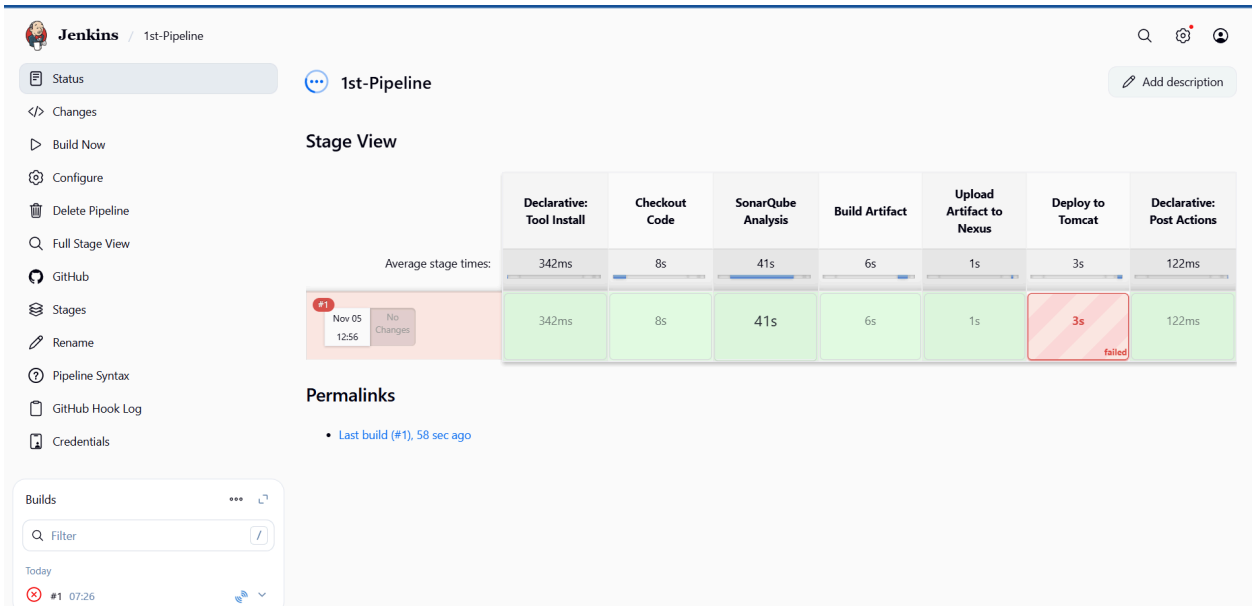
10 Validation

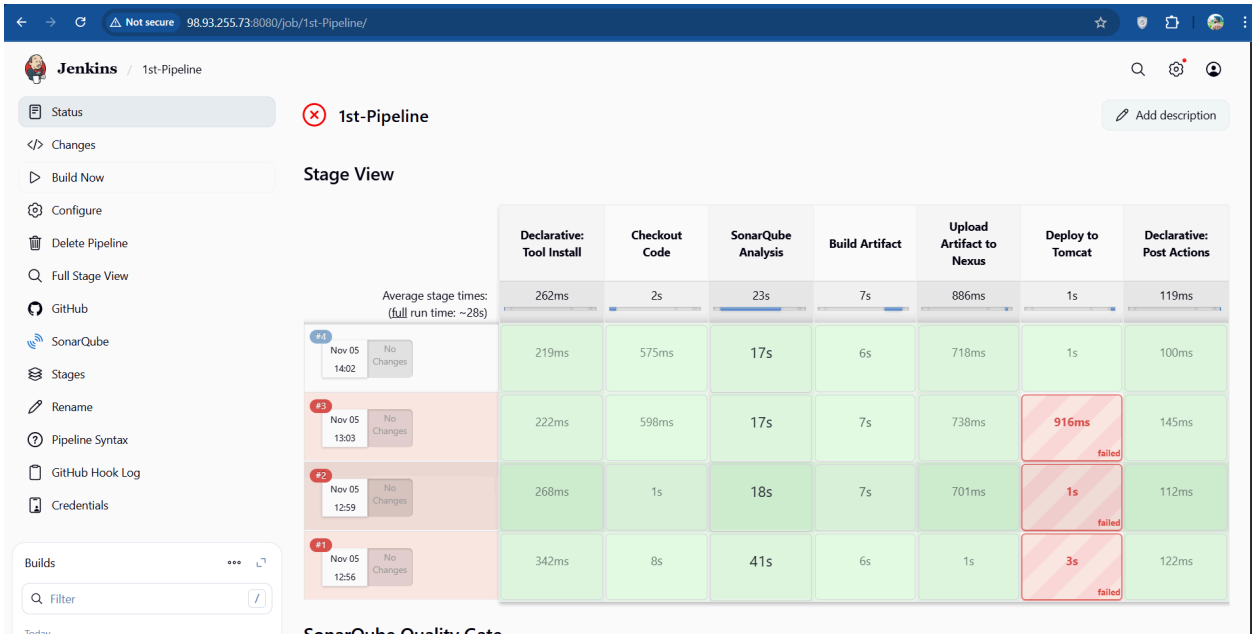
- Push code to the main branch — Jenkins should start automatically.
- Observe stages: Checkout → SonarQube → Build → Nexus Upload → Deploy to Tomcat.

Verify results:

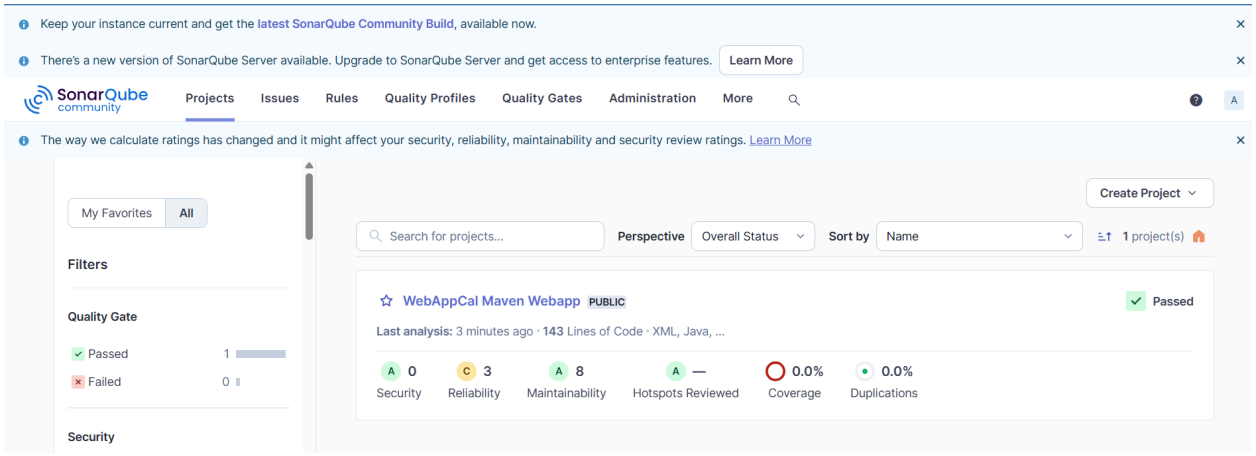
- SonarQube: http://<sonar-ip>:9000
- Nexus: http://<nexus-ip>:8081
- Tomcat: http://<tomcat-ip>:8080/<app-name>/

Jenkins Output:

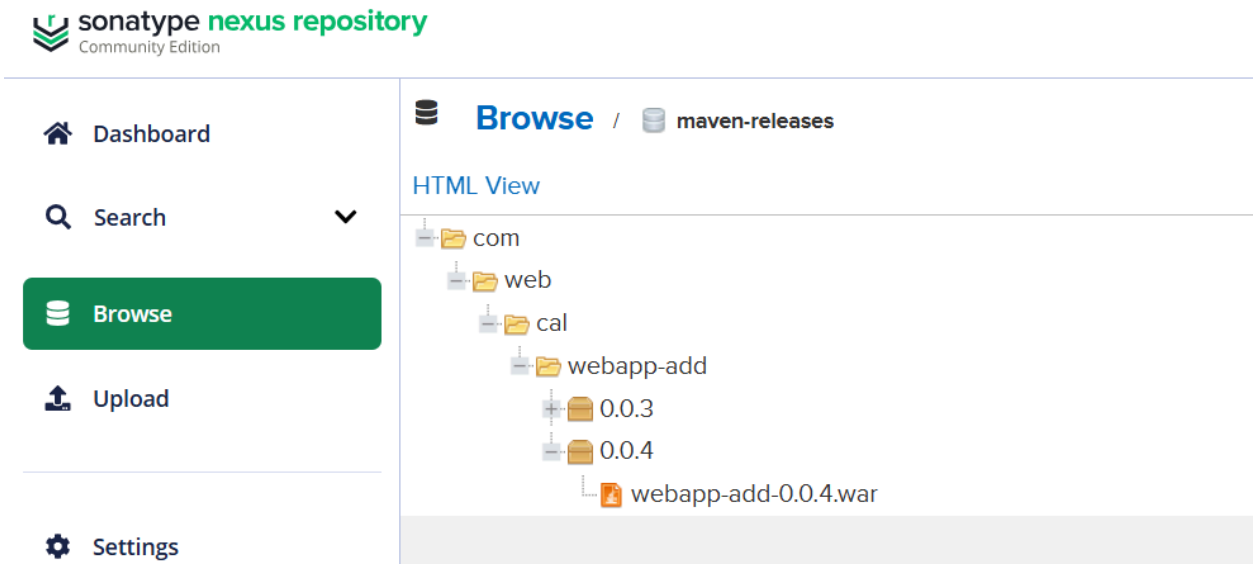




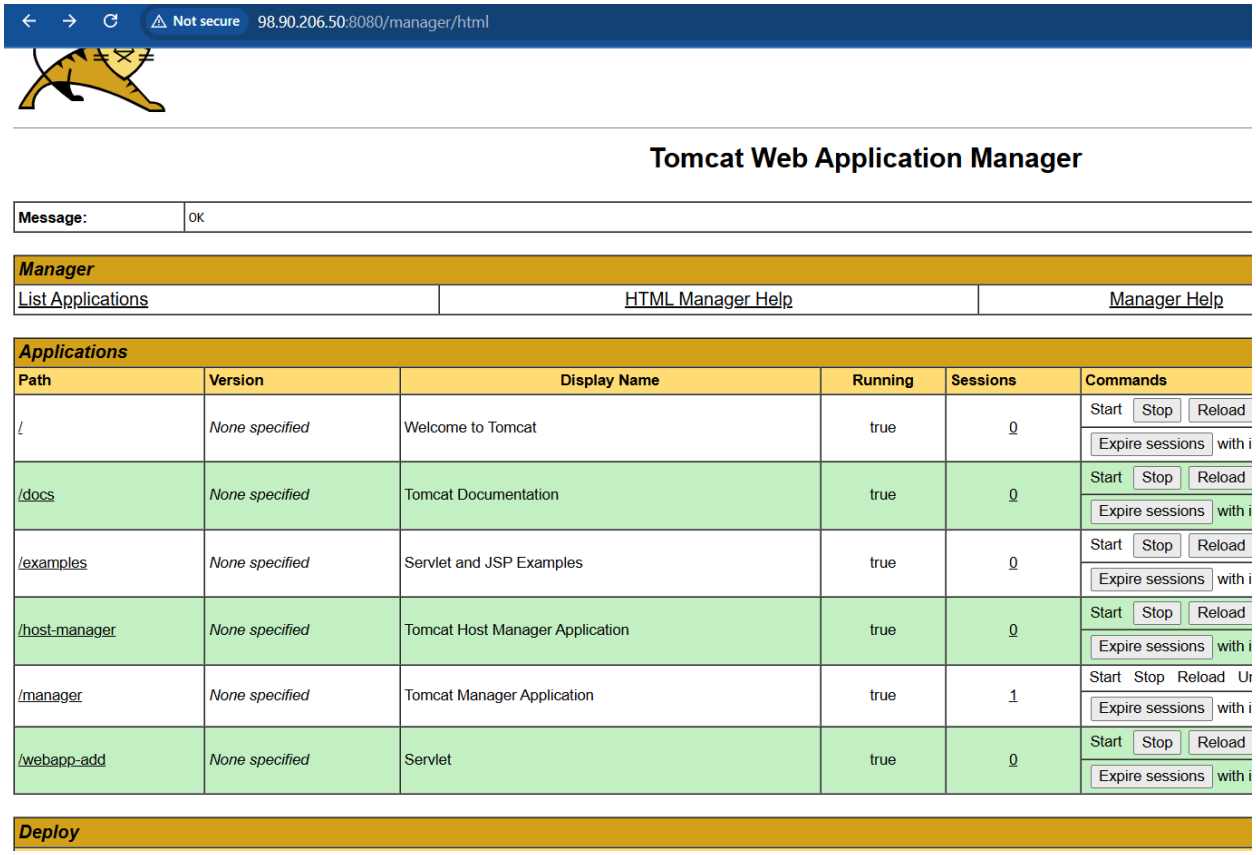
SonarQube Output:

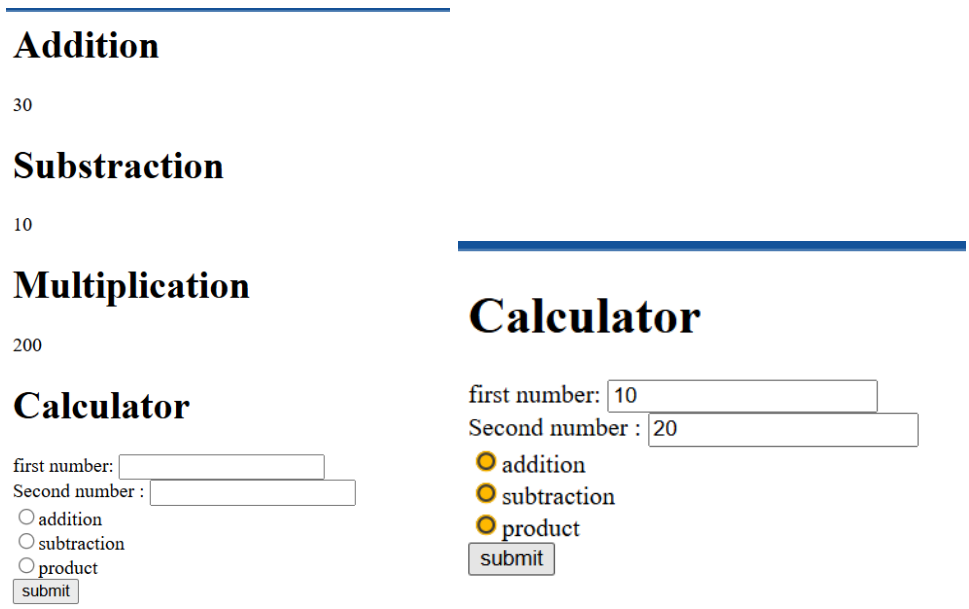


Nexus Output:



Tomcat Output:





12 — Troubleshooting (common errors we fixed during setup)

- **Tool type "jdk" does not have an install of "JDK21"**
  Ensure tools { jdk 'JDK17' } in Jenkinsfile or add JDK21 to Global Tool Config.
- **The JAVA_HOME environment variable is not defined correctly**
  Make sure agent has Java installed and tools section matches global tool name. Or set JAVA_HOME in environment in Jenkinsfile.
- **Nexus 401 Unauthorized during mvn deploy**
  Validate /home/jenkins/.m2/settings.xml server id and credentials and ensure pipeline uses --settings.
- **Nexus 400 cannot be updated**
  Don't redeploy the same release version; use unique versions or snapshots. We set 0.0.${BUILD_NUMBER}.
- **Downloaded file is .md5 or .sha1 or 404 bytes**
  Use REST JSON downloadUrl and filter .war only: grep -oP '"downloadUrl"\s*:\s*"\K[^"]+\.war' | grep -vE '\.md5|\.sha1'
- **Tomcat curl: (7) Failed to connect**
  Check Tomcat service, firewall, and cloud Security Group opening port 8080 to Jenkins agent.

- **Context starts but fails to run**
  Look at Tomcat logs: /var/log/tomcat9/catalina.out or Tomcat logs/ for stack traces; verify required libs and Java compatibility.

13 — Final checklist to hand to a colleague (copy-paste)

- GitHub repo accessible and contains Jenkinsfile (above)
- VMs provisioned: Jenkins, Sonar, Nexus, Tomcat
- Java installed (JDK17 on agents; JDK17/21 on master if desired)
- Sonar running at http://SONAR_IP:9000, token created
- Nexus running at http://NEXUS_IP:8081, repository maven-releases created
- Tomcat running at http://TOMCAT_IP:8080, manager-script user created
- Jenkins installed, plugins added, Global Tools configured (JDK17, Maven)
- SSH key generated on Jenkins master and public key added to agents
- Credentials added in Jenkins: ssh-ubuntu, nexus-creds, tomcat-manager, github-token
- Nodes created and online: SonarNode, TomcatNode
- settings.xml present on build agent (/home/jenkins/.m2/settings.xml)
- Jenkinsfile present in repo root
- GitHub webhook configured pointing to http://<jenkins>:8080/github-webhook/
- Successful pipeline run and app reachable at http://TOMCAT_IP:8080/<artifact-name>/

14. Optional improvements (future)

- Serve Angular app via **Nginx** instead of Tomcat for better static performance: build, upload to Nexus, and deploy to Nginx server (rsync or curl).
- Use Nexus npm repository and publish package versions (if you want npm package distribution).
- Add automated e2e tests (Protractor / Cypress) as a pipeline stage before deploy.
- Support environment-specific builds (staging/prod) by parameterizing Jenkinsfile.