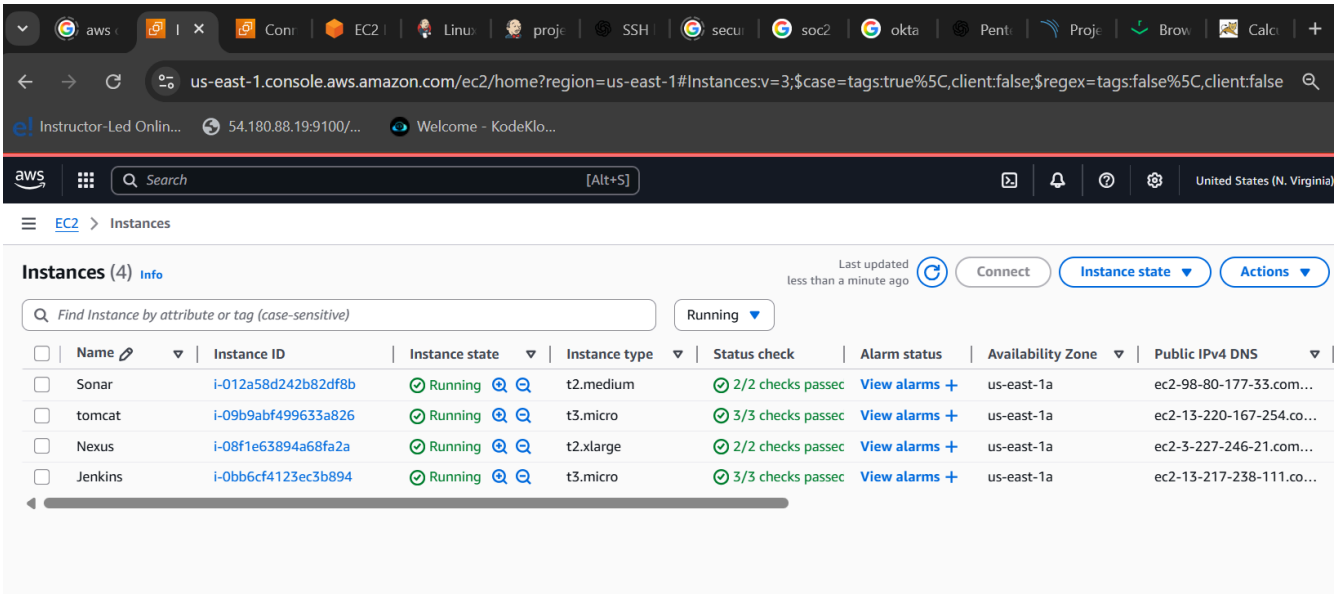
🚀 Integration Guide — GitHub → Jenkins → SonarQube → Nexus → Tomcat

---

🎯 Goal

On every push to GitHub:

1. Jenkins checks out code
2. SonarQube runs static analysis
3. Maven builds a WAR
4. Artifact is uploaded to Nexus (maven-releases)
5. Jenkins downloads the latest WAR and deploys it to Tomcat (Tomcat Manager)
   Everything is automated by a Jenkinsfile and triggered by GitHub webhook.



📚 Requirements (what to provision first)

- 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

---

1 — Create VMs & baseline OS setup (run on each VM)

Run as a sudo-capable user (example ubuntu).

sudo apt update && sudo apt upgrade -y

sudo apt install -y curl wget git unzip jq ufw

sudo timedatectl set-timezone Asia/Kolkata # optional

Open SSH in firewall:

sudo ufw allow OpenSSH

sudo ufw enable

Install Java 17 (agents, Sonar, Tomcat) and optionally Java 21 on Jenkins master:

Agents / Sonar / Tomcat

sudo apt install -y openjdk-17-jdk

java -version

Optional - Jenkins master

sudo apt install -y openjdk-21-jdk

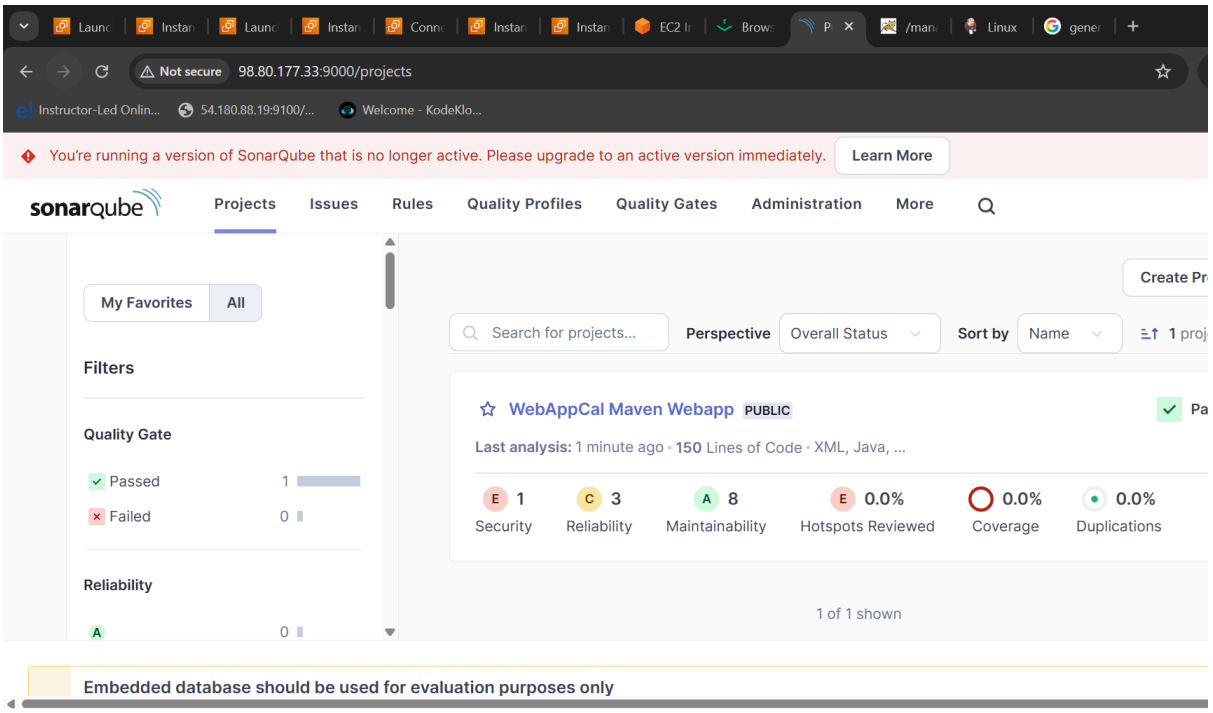
Create workspace dir for Jenkins agents:

sudo mkdir -p /home/ubuntu/jenkins

sudo chown ubuntu:ubuntu /home/ubuntu/jenkins

---

2 — Install & configure SonarQube (Sonar server)



(Prefer systemd install for production.)

1. **Install prerequisites**

sudo apt update

sudo apt install -y openjdk-17-jdk unzip

2. **Download & extract SonarQube**

cd /opt

sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.6.0.92116.zip

sudo unzip sonarqube-10.6.0.92116.zip

sudo mv sonarqube-10.6.0.92116 /opt/sonarqube

sudo useradd -r -s /bin/false sonar

sudo chown -R sonar:sonar /opt/sonarqube

3. **Create systemd service**

sudo tee /etc/systemd/system/sonarqube.service > /dev/null <<'EOF'

[Unit]

Description=SonarQube service

After=network.target

[Service]

Type=forking

User=sonar

Group=sonar

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start

ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

LimitNOFILE=65536

LimitNPROC=4096

[Install]

WantedBy=multi-user.target

EOF

sudo systemctl daemon-reload

sudo systemctl enable --now sonarqube

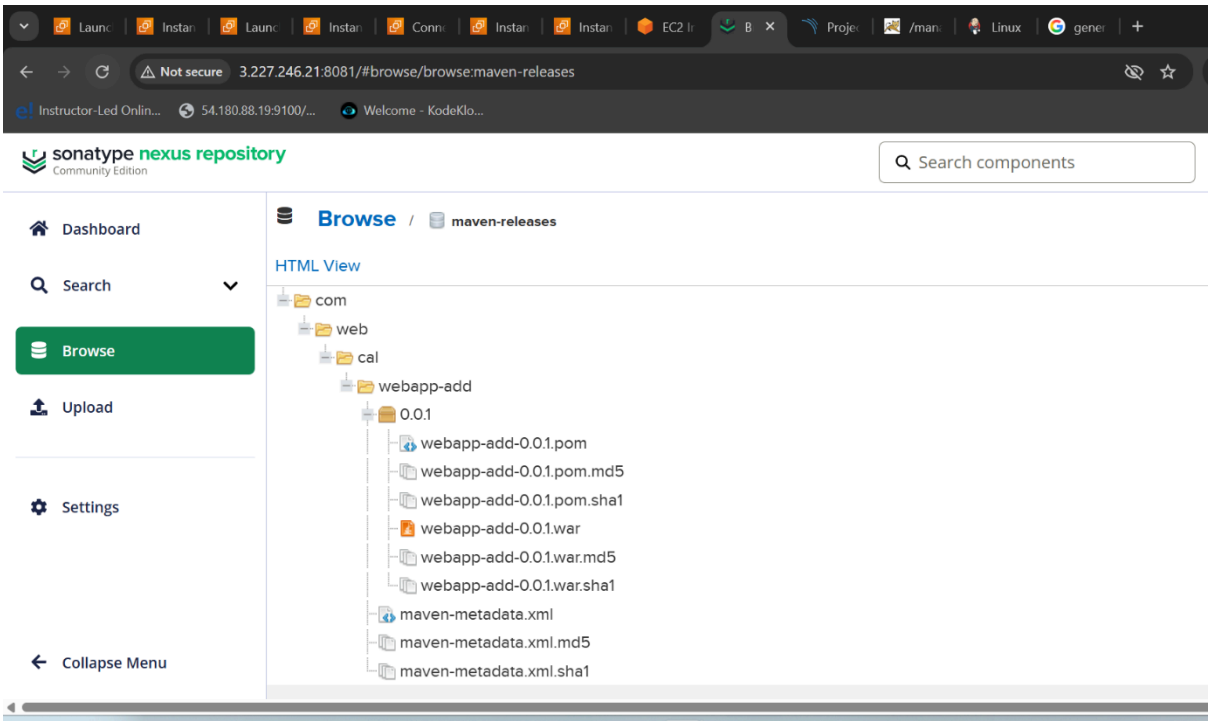
sudo systemctl status sonarqube

4. **Validate**
   Open http://SONAR_IP:9000. Default login admin/admin.
   Create a **token**: *My Account → Security → Generate Token* — copy it for Jenkins.

Notes: Tune JVM memory in production (/opt/sonarqube/conf).

---

3 — Install & configure Nexus OSS (Nexus server)



1. **Install Java 17**

sudo apt update

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

2. **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

3. **Create systemd service**

sudo tee /etc/systemd/system/nexus.service > /dev/null <<'EOF'

[Unit]

Description=nexus service

After=network.target

[Service]

Type=forking

User=nexus

ExecStart=/opt/nexus/bin/nexus start

ExecStop=/opt/nexus/bin/nexus stop

Restart=on-abort

[Install]

WantedBy=multi-user.target

EOF

sudo systemctl daemon-reload

sudo systemctl enable --now nexus

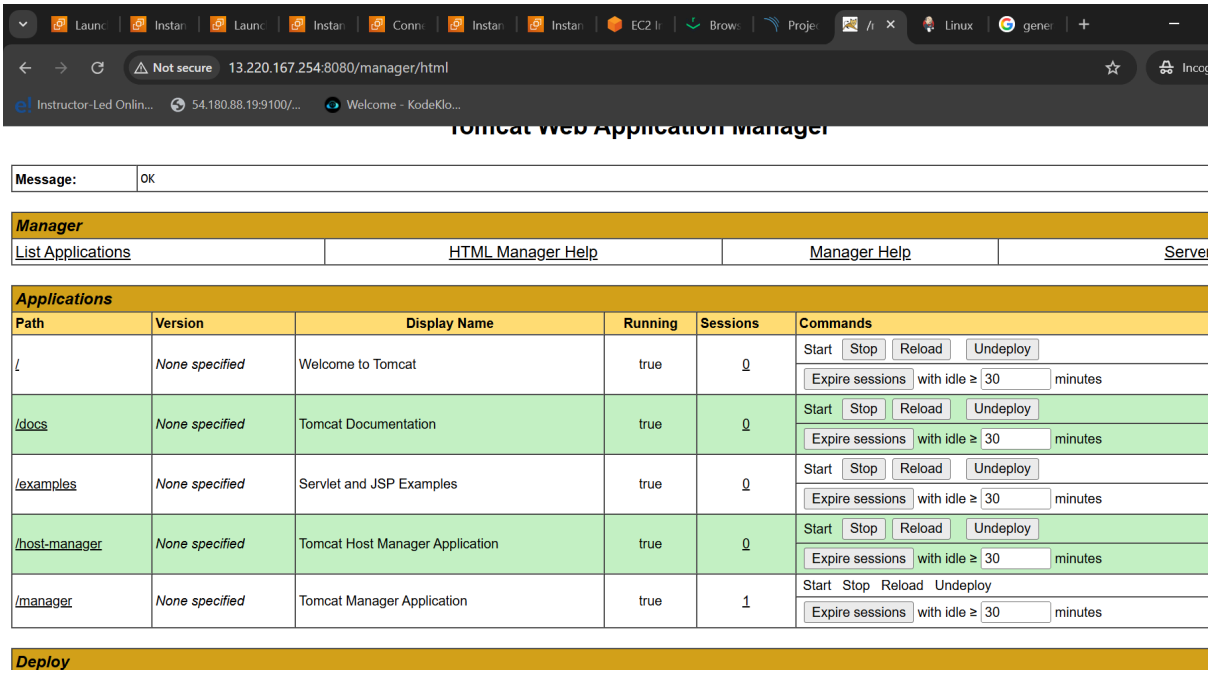
sudo systemctl status nexus

4. **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.

Important: If Nexus binds to 127.0.0.1, set application-host=0.0.0.0 in Nexus config (then restart).

---

4 — Install & configure Tomcat (Tomcat server)



1. **Install Tomcat**

sudo apt update

sudo apt install -y tomcat9 tomcat9-admin

2. **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

3. **Validate**
   From an agent:

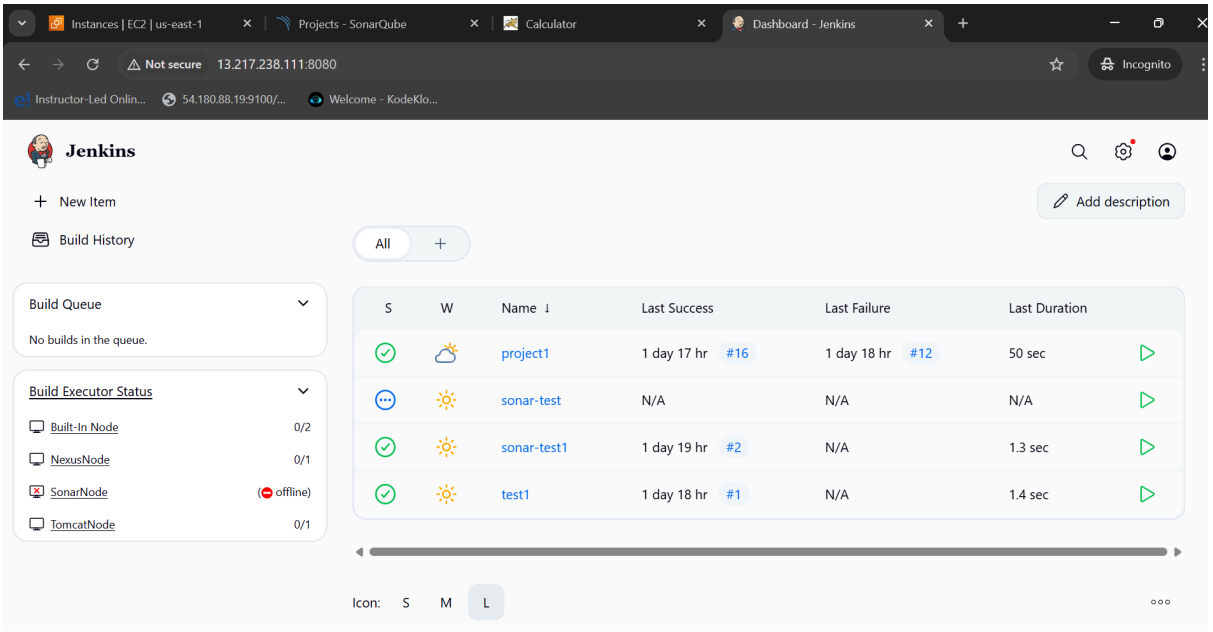
curl -u admin:admin123 http://TOMCAT_IP:8080/manager/text/list

Expect an OK response including contexts

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

---

5 — Install Jenkins master (exact commands + UI steps)



(Do this on Jenkins master VM.)

1. **Install Java (if not)**

sudo apt update

sudo apt install -y openjdk-21-jdk # optional; JDK17 works too

2. **Install Jenkins**

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list

sudo apt update

sudo apt install -y jenkins

sudo systemctl enable --now jenkins

sudo systemctl status jenkins

3. **Initial unlock & install suggested plugins**

- Browse to http://JENKINS_IP:8080
- Paste initial admin password:

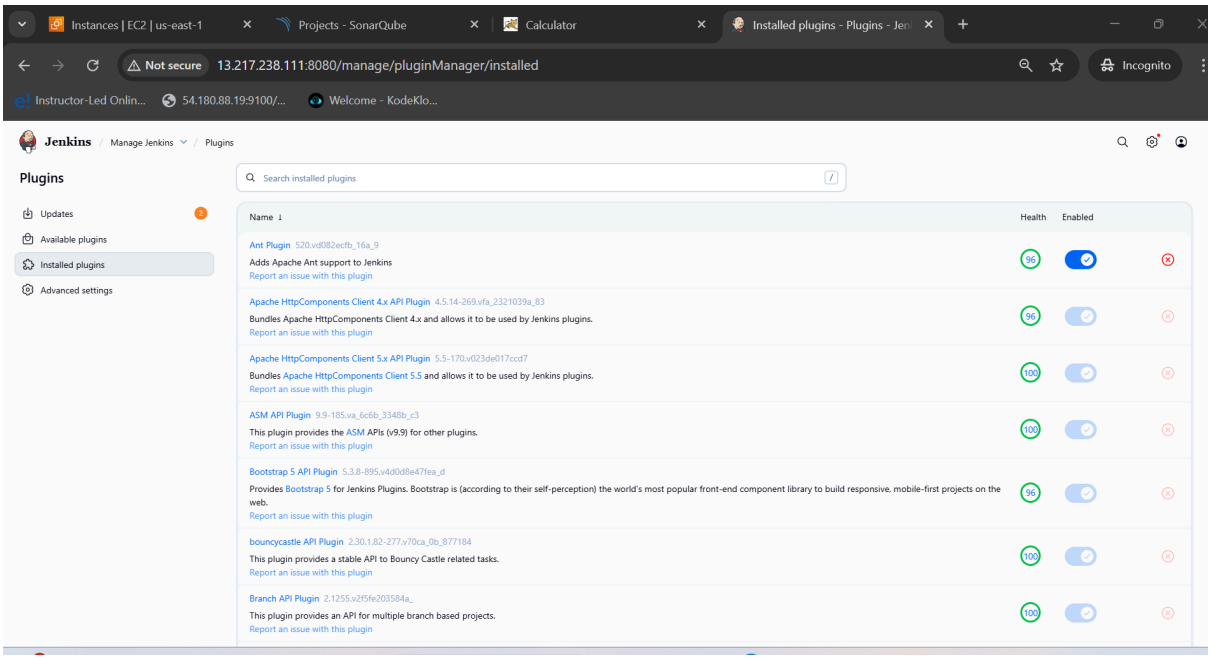
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

- On first run choose **Install suggested plugins**.

---

6 — Jenkins: Global configuration (plugins, tools, credentials, SSH key)

Do the following in Jenkins UI; copy/paste where possible.



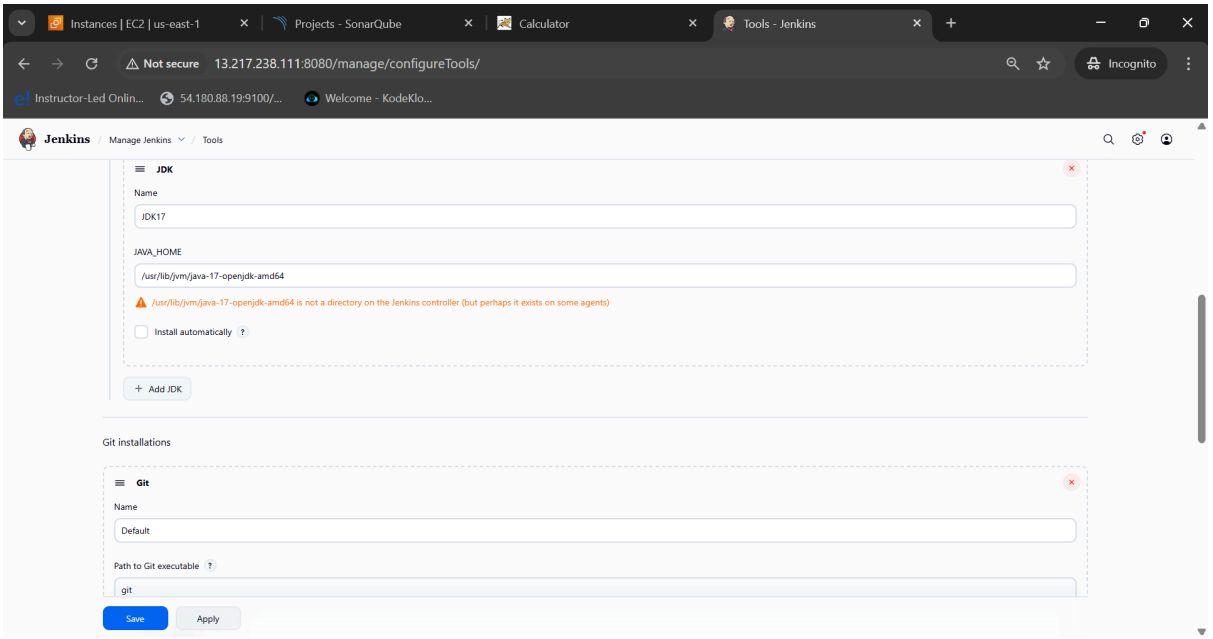
6A — Install required plugins

Manage Jenkins → *Plugin Manager* → *Available* → install:

- **Git Plugin**
- **GitHub Integration Plugin**
- **Pipeline**
- **Pipeline: Multibranch**
- **Maven Integration plugin**
- **SonarQube Scanner for Jenkins**
- **SSH Slaves / SSH Agents**
- **Credentials Binding**

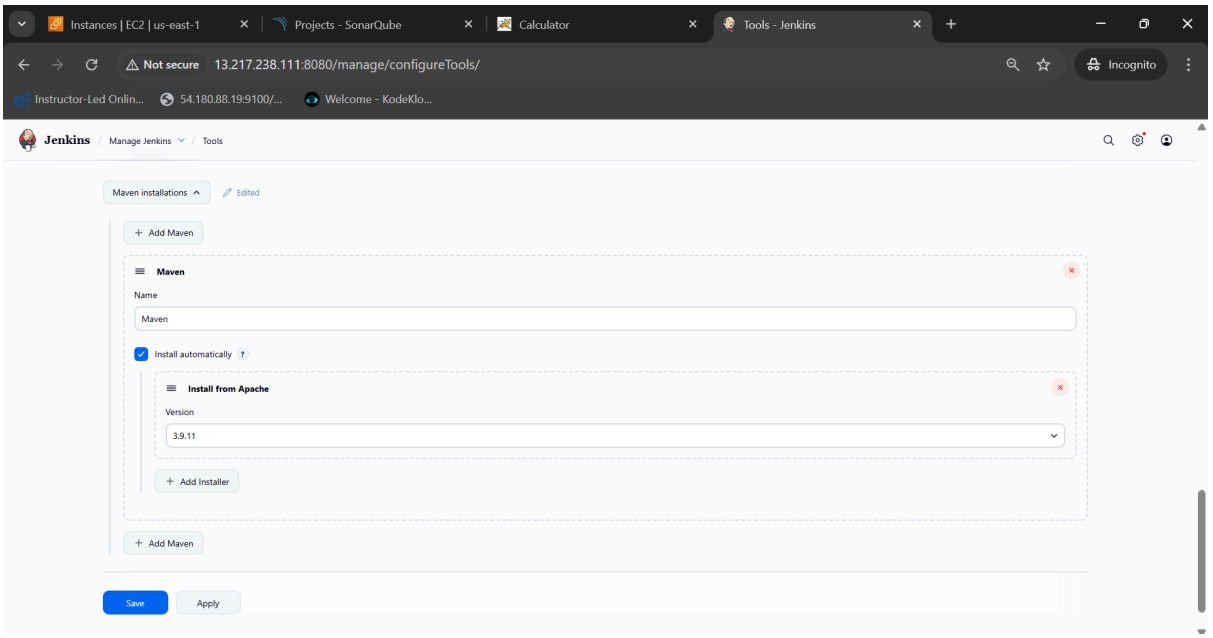
Restart Jenkins if required.

6B — Global Tool Configuration



Manage Jenkins → Global Tool Configuration

- **Add JDK**
  - Name: JDK17
  - (If you installed JDK on nodes, uncheck auto-install)



- **Add Maven**
    - Name: Maven
    - Choose auto-install or set path.

These names must match tools { jdk 'JDK17' ; maven 'Maven' } in the Jenkinsfile.

6C — Generate SSH key for agent SSH

Run on Jenkins master (as jenkins user OR root and then use file for credential):

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

sudo -u jenkins ssh-keygen -t rsa -b 4096 -f /var/lib/jenkins/.ssh/id_rsa_agent -N ""

sudo chown jenkins:jenkins /var/lib/jenkins/.ssh/id_rsa_agent*

sudo chmod 600 /var/lib/jenkins/.ssh/id_rsa_agent

sudo cat /var/lib/jenkins/.ssh/id_rsa_agent.pub

Copy the public key into each agent's /home/ubuntu/.ssh/authorized_keys:

on SonarNode & TomcatNode

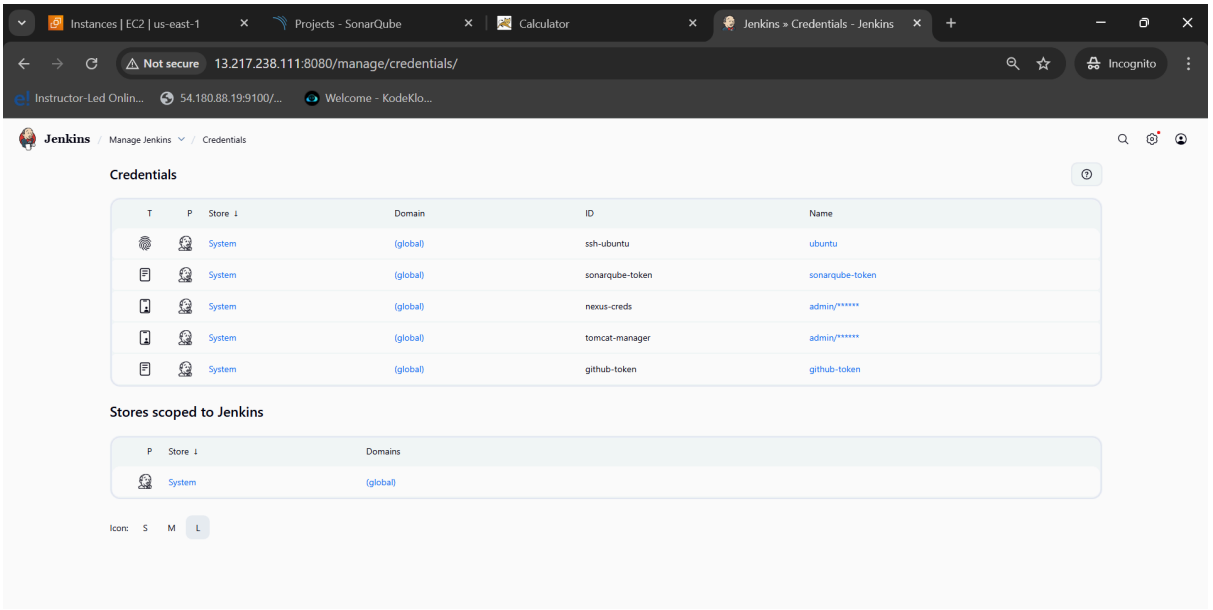
mkdir -p ~/.ssh

echo "<PUBLIC_KEY>" >> ~/.ssh/authorized_keys

chmod 700 ~/.ssh

chmod 600 ~/.ssh/authorized_keys

6D — Add credentials in Jenkins (exact IDs you will use)



Manage Jenkins → Credentials → System → Global → Add Credentials

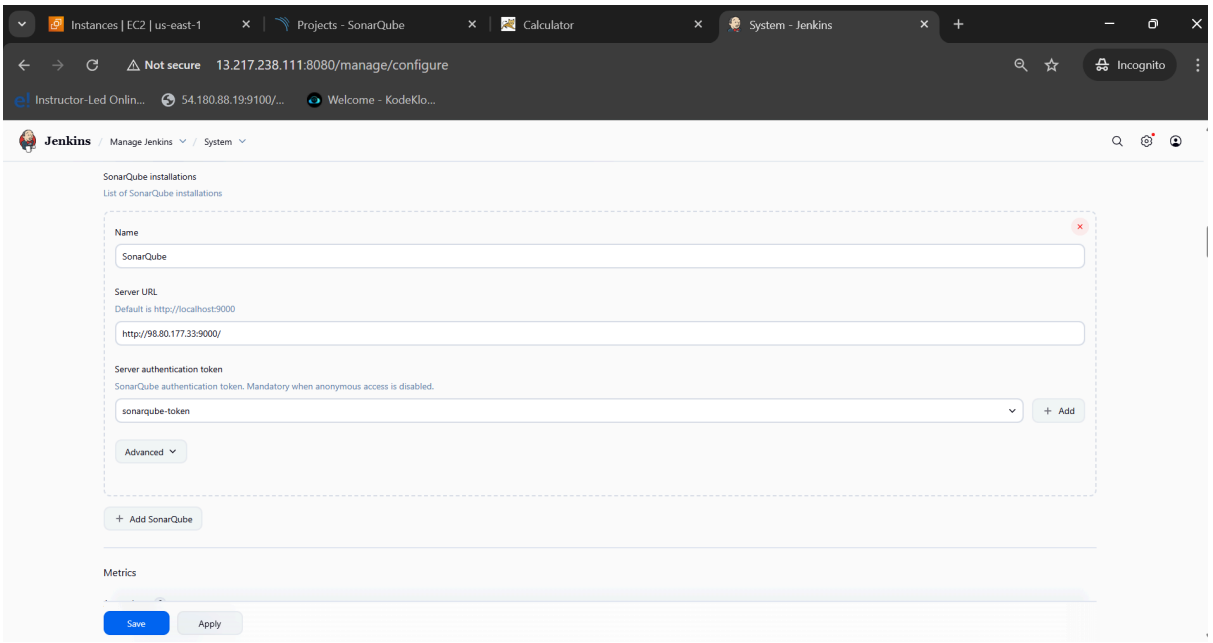
Create these credentials (IDs are used in pipeline/steps):

1. **SSH Username with private key**
    - Kind: SSH Username with private key
    - ID: ssh-ubuntu
    - Username: ubuntu
    - Private key: *Enter directly* → paste /var/lib/jenkins/.ssh/id_rsa_agent contents
    - Description: SSH key for ubuntu user on agents
2. **Nexus credentials**
    - Kind: Username with password
    - ID: nexus-creds
    - Username: admin
    - Password: admin123 (change)
3. **Tomcat manager**
    - Kind: Username with password

- ID: tomcat-manager
- Username: admin
- Password: admin123

4. **GitHub token**
   - Kind: Secret text
   - ID: github-token
   - Secret: Your GitHub PAT (scopes: repo, admin:repo_hook)
5. (Optional) **Sonar token** if you prefer storing it as credential instead of Configure System.

6E — Configure SonarQube Server in Jenkins

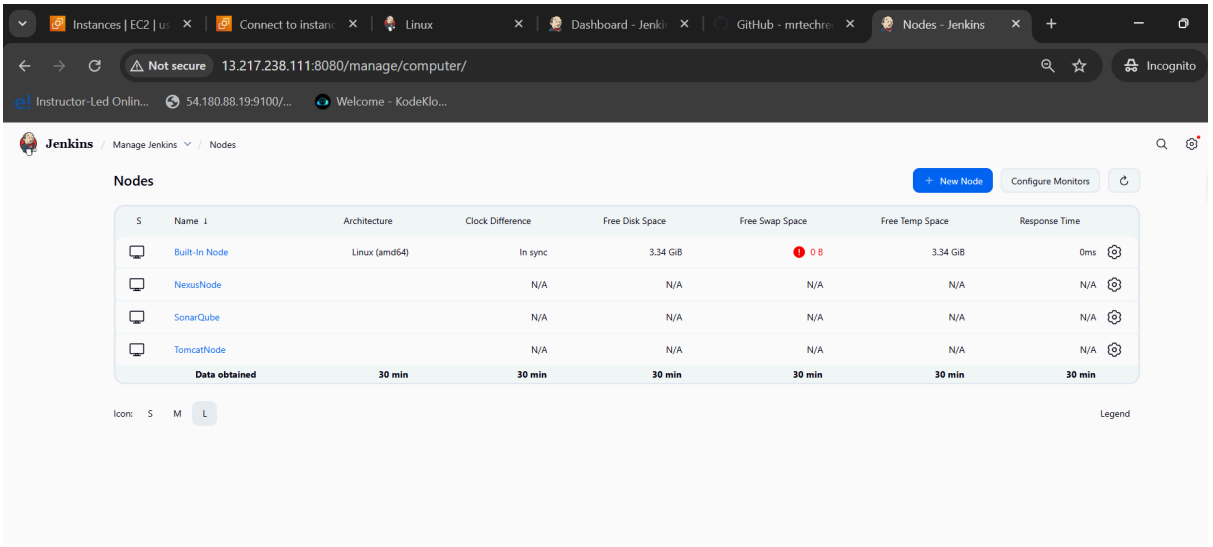


Manage Jenkins → Configure System → SonarQube servers

- Name: SonarQube
- Server URL: http://SONAR_IP:9000
- Server authentication token: paste token generated in Sonar UI
- Save

7 — Create Jenkins agents (workers / nodes)

Do in Jenkins UI:



Manage Jenkins → Manage Nodes and Clouds → New Node

Create **SonarNode**

- Name: SonarNode
- Type: Permanent Agent
- Remote root directory: /home/ubuntu/jenkins
- Labels: SonarNode
- Launch method: **Launch agents via SSH**
    - Host: Sonar VM IP
    - Credentials: ssh-ubuntu
    - Test connection → Save

Create **TomcatNode**

- Name: TomcatNode
- Remote root directory: /home/ubuntu/jenkins
- Labels: TomcatNode

- Launch via SSH → Host Tomcat VM IP, Creds ssh-ubuntu

Verify both nodes show **Online**.

If nodes fail to connect: check that /home/ubuntu/jenkins exists and is owned by ubuntu:

sudo mkdir -p /home/ubuntu/jenkins

sudo chown ubuntu:ubuntu /home/ubuntu/jenkins

---

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

8A — 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>
```

8B — Add versions plugin in pom.xml

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

8C — 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.

---

9 — 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_ARTIFA
CT}/${VERSION}/${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.' }
}
```

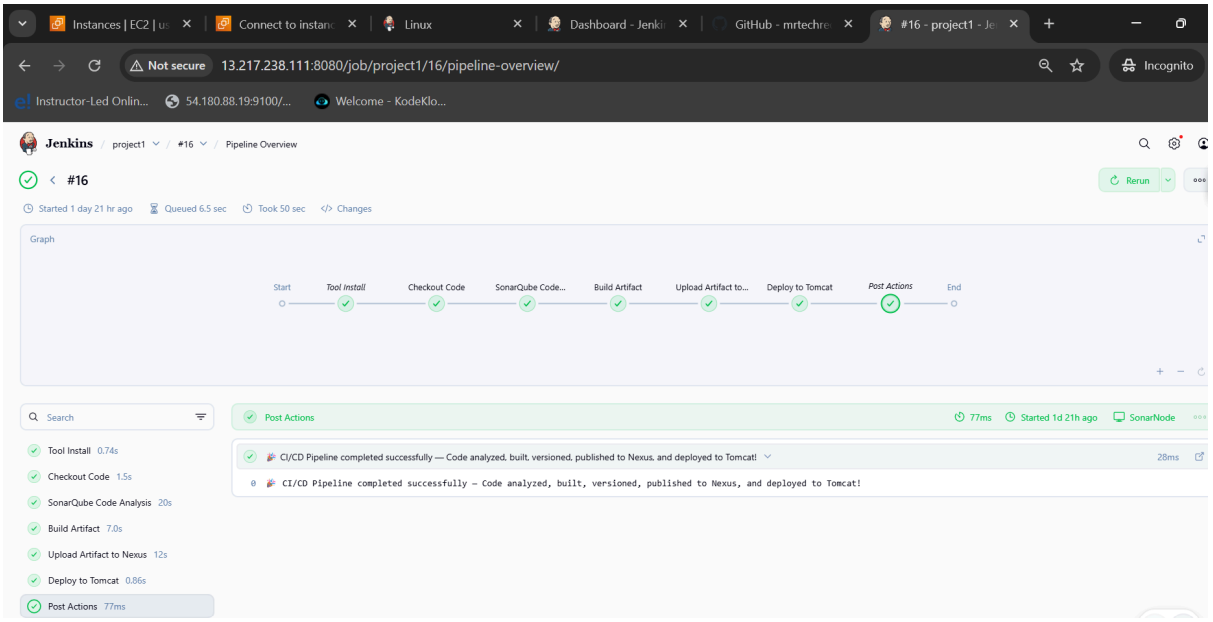
}

Notes

- NEW_VERSION="0.0.${BUILD_NUMBER}" ensures every deploy to maven-releases has unique version (avoids Nexus "cannot be updated" 400 error).
- APP_NAME derived from WAR file to deploy with matching context (prevents context name mismatch).

---

10 — Create Jenkins Pipeline job & GitHub webhook (UI steps)



1. **Create pipeline job**
    - Jenkins → New Item → project1 → Pipeline → OK
    - Pipeline section:
        - Definition: *Pipeline script from SCM*
        - SCM: Git

        - Repository URL: https://github.com/you/your-app.git
        - Branches to build: */main
        - Script path: Jenkinsfile
2. **Build trigger**
    - Check **GitHub hook trigger for GITScm polling**
3. **Add GitHub webhook**
    - GitHub repo → Settings → Webhooks → Add webhook
        - Payload URL: http://<JENKINS_PUBLIC_IP>:8080/github-webhook/
        - Content type: application/json
        - Events: *Push events* (optionally PRs)
    - Click **Add webhook**
4. **Test**: push a commit — Jenkins job should start automatically.

---

11 — Validation & manual checks (what to run & expected outputs)

Confirm Sonar token in Jenkins

- Manage Jenkins → Configure System → SonarQube servers → SonarQube configured.

Confirm Maven settings on agent

On SonarNode:

cat /home/jenkins/.m2/settings.xml

must contain <id>maven-releases</id> and correct user/password

Validate Nexus search API returns downloadUrl

curl -s -u admin:admin123 "http://3.227.246.21:8081/service/rest/v1/search?repository=maven-releases&group=com.web.cal&name=webapp-add" | jq .

items[].assets[].downloadUrl should include .war entries

Validate Tomcat deployment manually (from agent)

curl -u admin:admin123 http://13.220.167.254:8080/manager/text/list

OK or list of contexts

---

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>/
