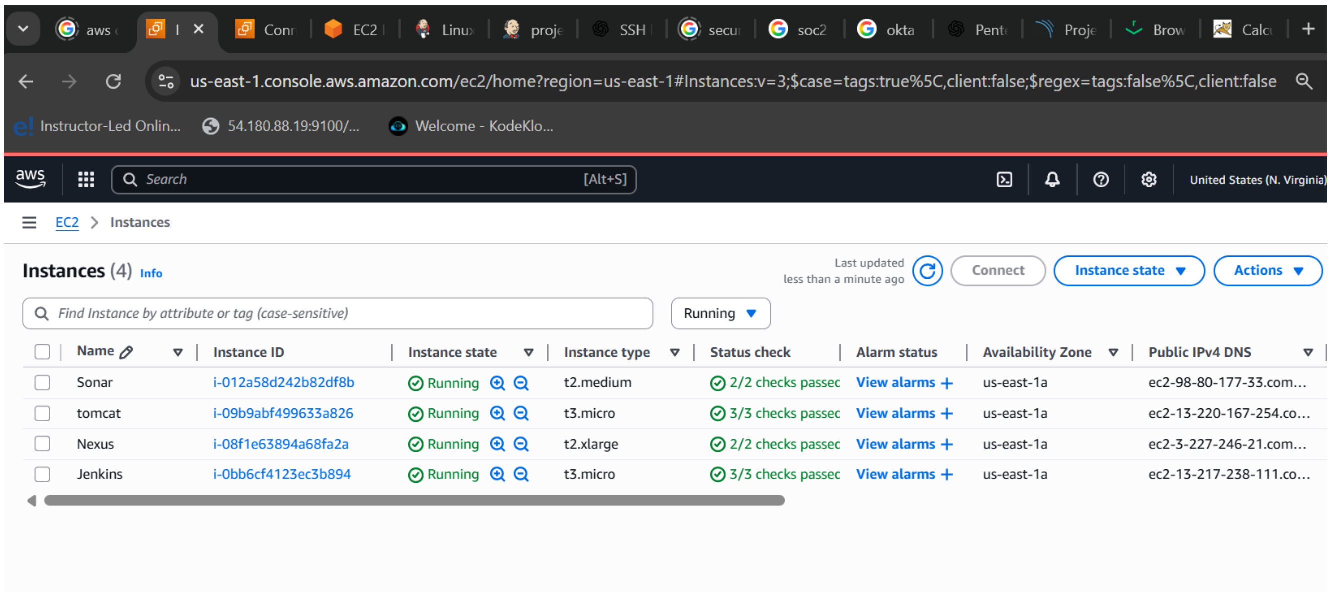
🚀 Integration Guide — GitHub → Jenkins → SonarQube → Nexus → Tomcat

---

🎯 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.



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:v=3;$case=tags:true%5C,client:false;$regex=tags:false%5C,client:false

EC2 > Instances

Instances (4) Info

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive) | Running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Sonar	i-012a58d242b82df8b	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1a	ec2-98-80-177-33.com...
tomcat	i-09b9abf499633a826	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1a	ec2-13-220-167-254.co...
Nexus	i-08f1e63894a68fa2a	Running	t2.xlarge	2/2 checks passec	View alarms +	us-east-1a	ec2-3-227-246-21.com...
Jenkins	i-0bb6cf4123ec3b894	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1a	ec2-13-217-238-111.co...

📚 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

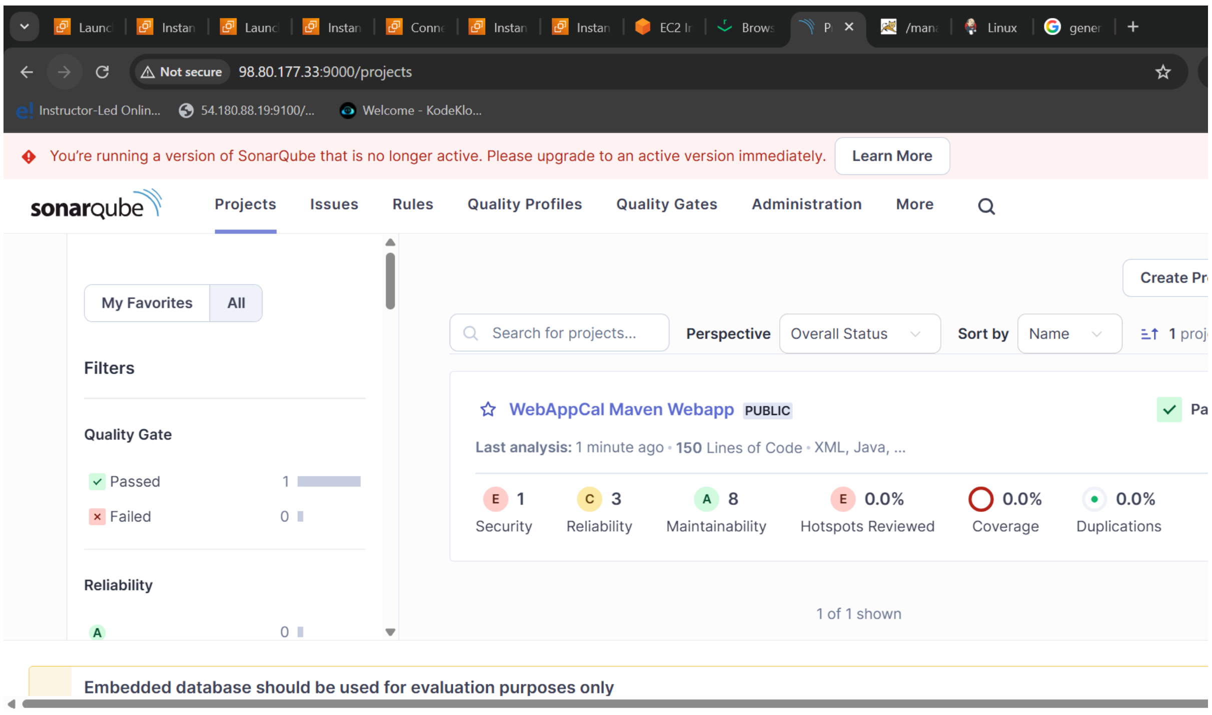
Create workspace dir for Jenkins agents:

sudo mkdir -p /home/ubuntu/jenkins

sudo chown ubuntu:ubuntu /home/ubuntu/jenkins

---

2 — Install & configure SonarQube (Sonar server)



You're running a version of SonarQube that is no longer active. Please upgrade to an active version immediately.

Filters

Quality Gate

Passed	1	
Failed	0	

Reliability

A	0

WebAppCal Maven Webapp PUBLIC Pa

Last analysis: 1 minute ago · 150 Lines of Code · XML, Java, ...

E 1	C 3	A 8	E 0.0%	0.0%	0.0%
Security	Reliability	Maintainability	Hotspots Reviewed	Coverage	Duplications

1 of 1 shown

Embedded database should be used for evaluation purposes only

(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

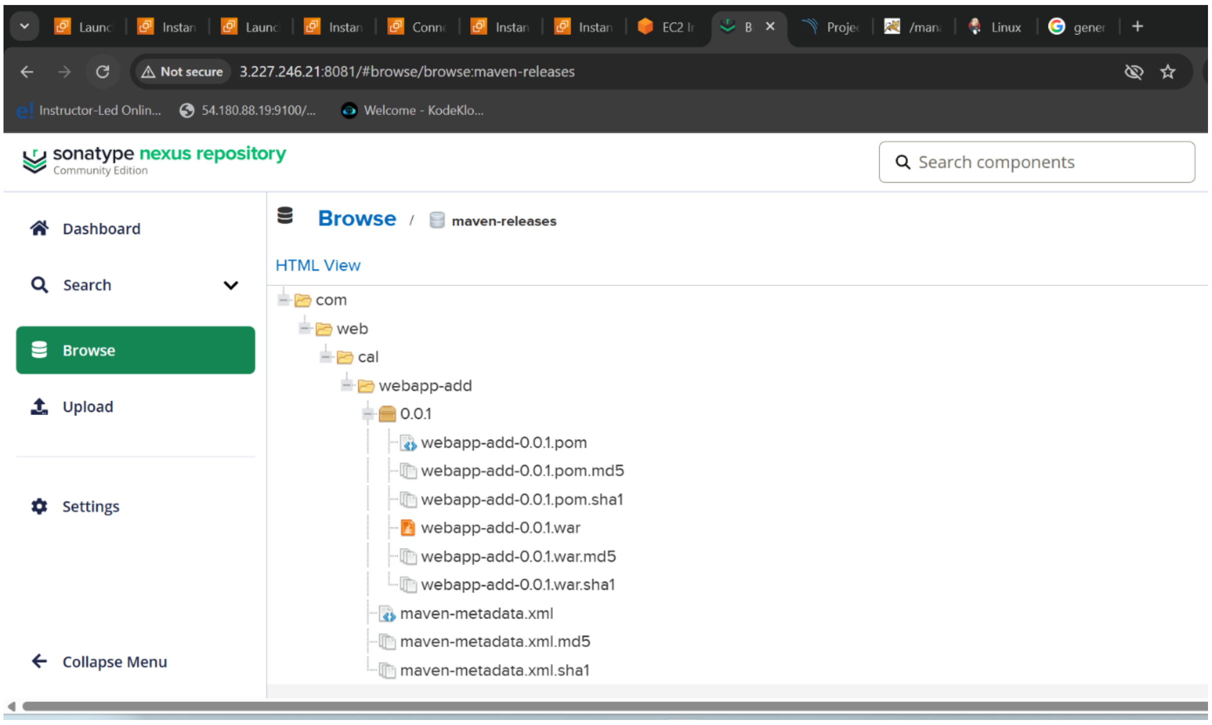
sudo systemctl status sonarqube

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)



```
com
  web
    cal
      webapp-add
        0.0.1
          webapp-add-0.0.1.pom
          webapp-add-0.0.1.pom.md5
          webapp-add-0.0.1.pom.sha1
          webapp-add-0.0.1.war
          webapp-add-0.0.1.war.md5
          webapp-add-0.0.1.war.sha1
        maven-metadata.xml
        maven-metadata.xml.md5
        maven-metadata.xml.sha1
```

1. **Install Java 17**

sudo apt update

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

2. **Download & extract Nexus**

cd /opt

sudo wget https://download.sonatype.com/nexus/3/latest-unix.tar.gz -O nexus-latest.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
```

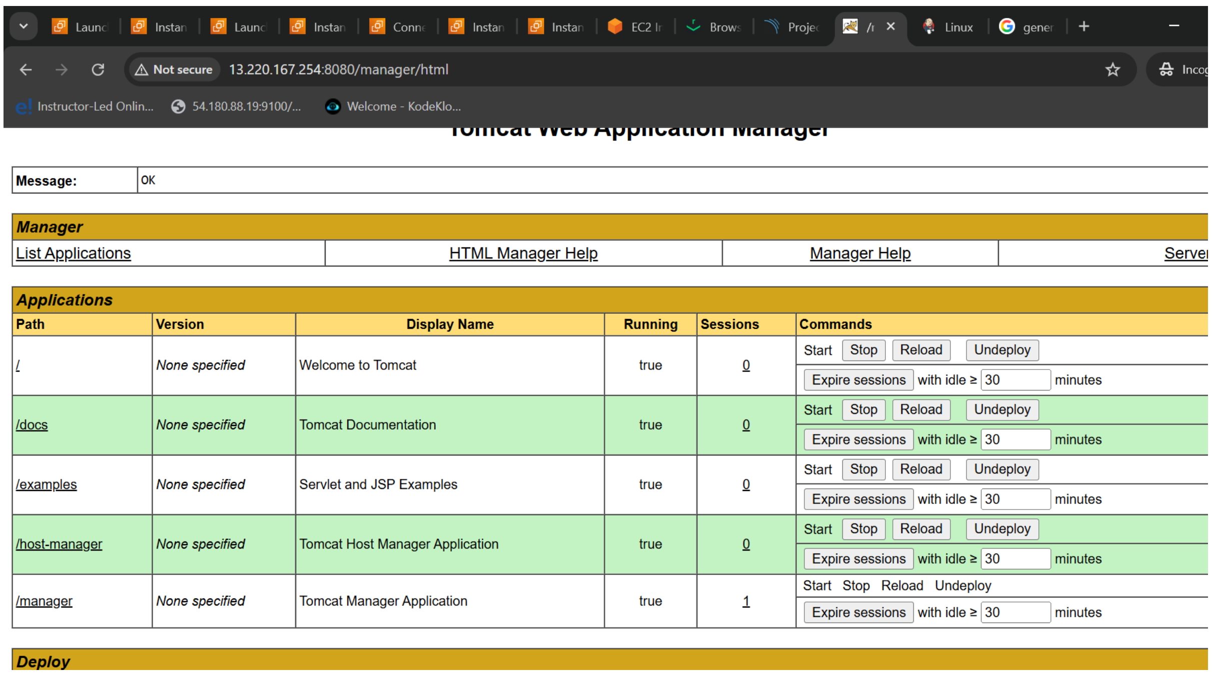
sudo systemctl status nexus

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)


Not secure
13.220.167.254:8080/manager/html
Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	*None specified*	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	*None specified*	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	*None specified*	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	*None specified*	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	*None specified*	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

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:

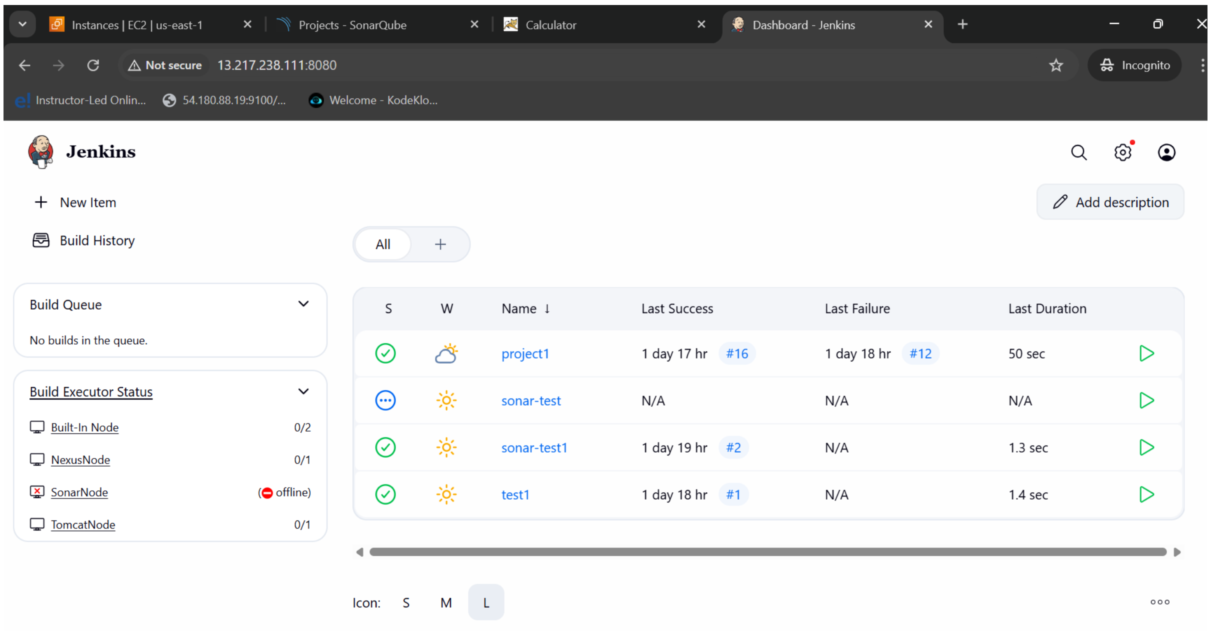
curl -u admin:admin123 http://TOMCAT_IP:8080/manager/text/list

# 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)



Instances | EC2 | us-east-1 — Projects - SonarQube — Calculator — Dashboard - Jenkins

Not secure 13.217.238.111:8080

Instructor-Led Onlin... 54.180.88.19:9100/... Welcome - KodeKlo...

Jenkins

+ New Item

Build History

Add description

All +

Build Queue

No builds in the queue.

Build Executor Status

Built-In Node	0/2
NexusNode	0/1
SonarNode	(offline)
TomcatNode	0/1

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	⛅	project1	1 day 17 hr #16	1 day 18 hr #12	50 sec
⋯	☀	sonar-test	N/A	N/A	N/A
✓	☀	sonar-test1	1 day 19 hr #2	N/A	1.3 sec
✓	☀	test1	1 day 18 hr #1	N/A	1.4 sec

Icon: S M L

(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:

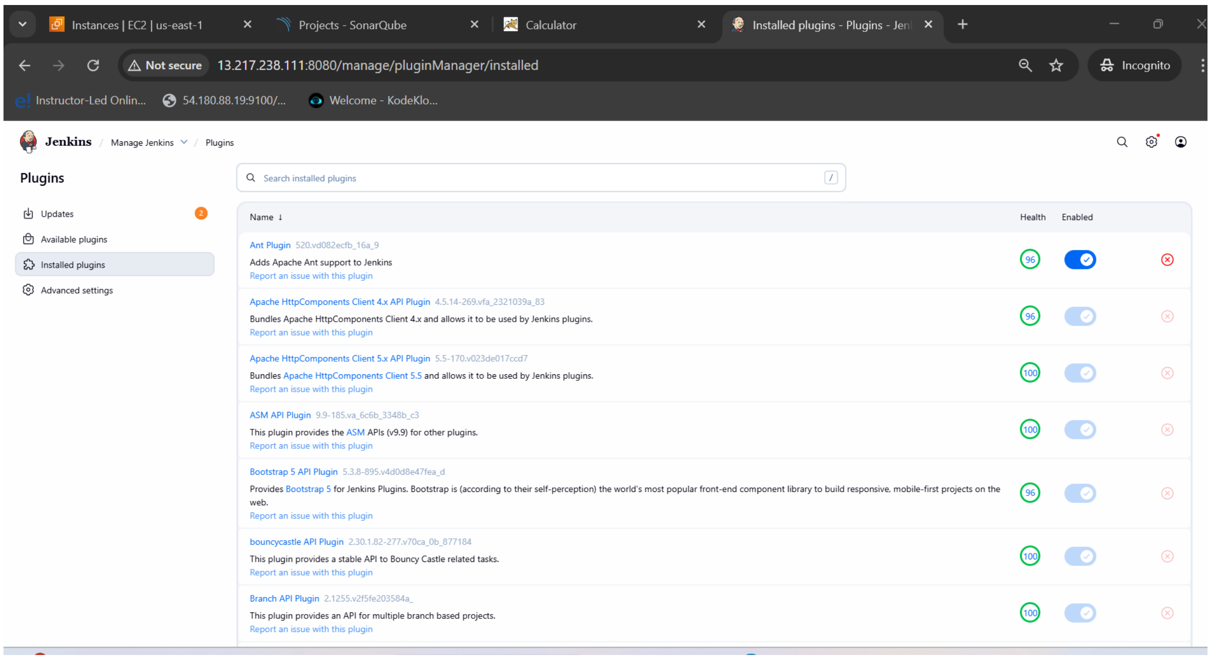
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

- 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.


13.217.238.111:8080/manage/pluginManager/installed

Jenkins / Manage Jenkins / Plugins

Plugins

- Updates 2
- Available plugins
- Installed plugins
- Advanced settings

Search installed plugins

Name ↓	Health	Enabled
Ant Plugin 520.vd082ecfb_16a_9 Adds Apache Ant support to Jenkins Report an issue with this plugin	96	✓
Apache HttpComponents Client 4.x API Plugin 4.5.14-269.vfa_2321039a_83 Bundles Apache HttpComponents Client 4.x and allows it to be used by Jenkins plugins. Report an issue with this plugin	96	✓
Apache HttpComponents Client 5.x API Plugin 5.5-170.v023de017ccd7 Bundles Apache HttpComponents Client 5.5 and allows it to be used by Jenkins plugins. Report an issue with this plugin	100	✓
ASM API Plugin 9.9-185.va_6c6b_3348b_c3 This plugin provides the ASM APIs (v9.9) for other plugins. Report an issue with this plugin	100	✓
Bootstrap 5 API Plugin 5.3.8-895.v4d0d8e47fea_d Provides Bootstrap 5 for Jenkins Plugins. Bootstrap is (according to their self-perception) the world's most popular front-end component library to build responsive, mobile-first projects on the web. Report an issue with this plugin	96	✓
bouncycastle API Plugin 2.30.1.82-277.v70ca_0b_877184 This plugin provides a stable API to Bouncy Castle related tasks. Report an issue with this plugin	100	✓
Branch API Plugin 2.1255.v2f5fe203584a_ This plugin provides an API for multiple branch based projects. Report an issue with this plugin	100	✓

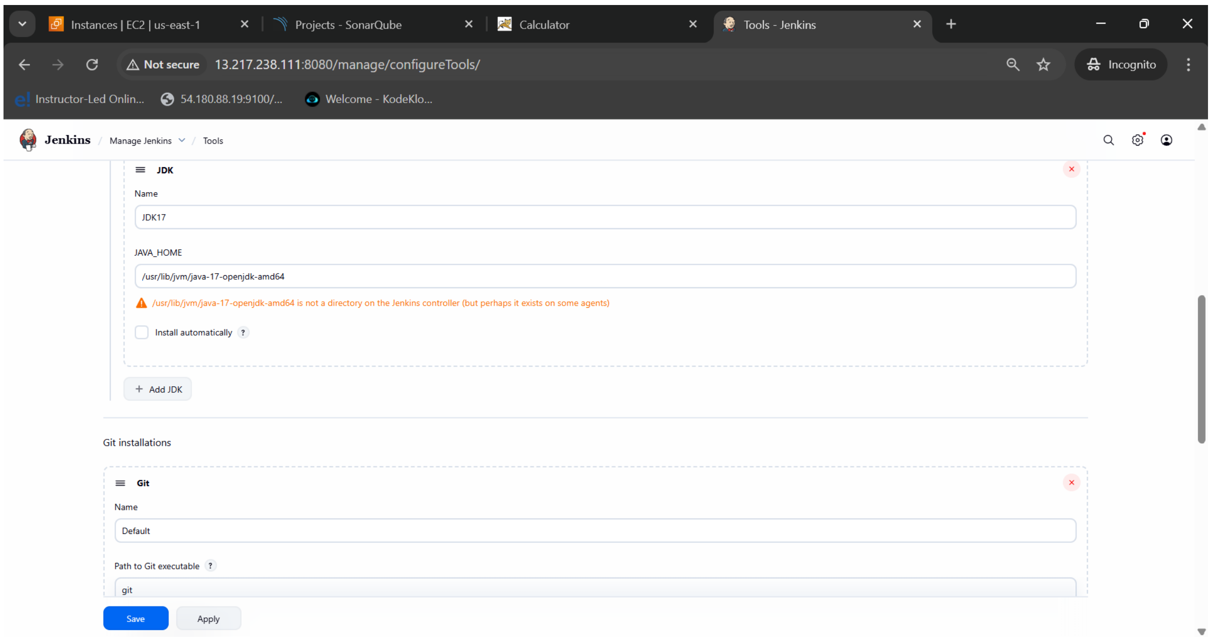
6A — Install required plugins

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



JDK

Name

JDK17

JAVA_HOME

/usr/lib/jvm/java-17-openjdk-amd64

/usr/lib/jvm/java-17-openjdk-amd64 is not a directory on the Jenkins controller (but perhaps it exists on some agents)

Install automatically

Add JDK

Git installations

Git

Name

Default

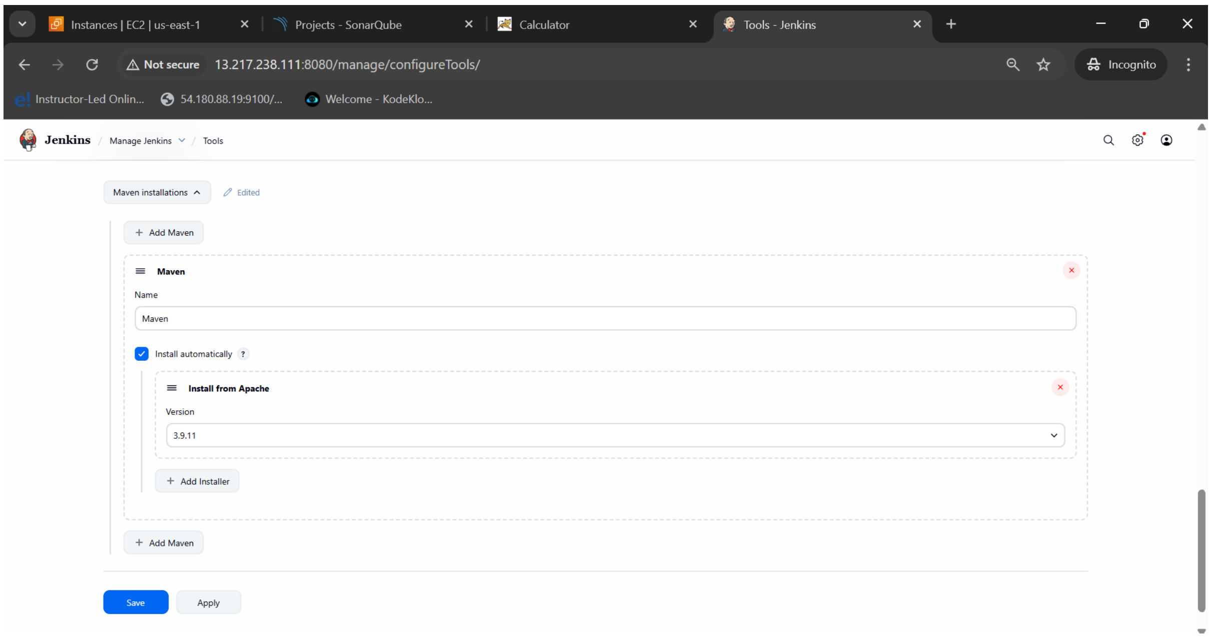
Path to Git executable

git

Save Apply

Manage Jenkins → Global Tool Configuration

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



Maven installations Edited

Add Maven

Maven

Name

Maven

Install automatically

Install from Apache

Version

3.9.11

Add Installer

Add Maven

Save Apply

- **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

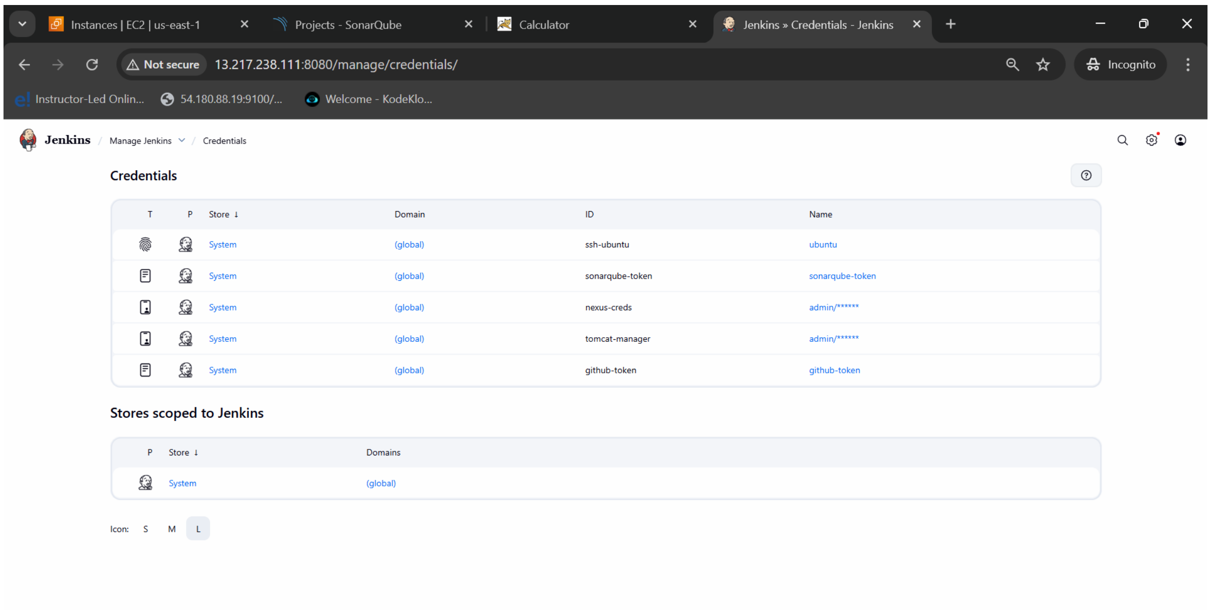
Run on Jenkins master (as jenkins user OR root and then use file for credential):

```
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)



Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	ssh-ubuntu	ubuntu
		System	(global)	sonarqube-token	sonarqube-token
		System	(global)	nexus-creds	admin/******
		System	(global)	tomcat-manager	admin/******
		System	(global)	github-token	github-token

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Icon: S M L

Manage Jenkins → Credentials → System → Global → Add Credentials

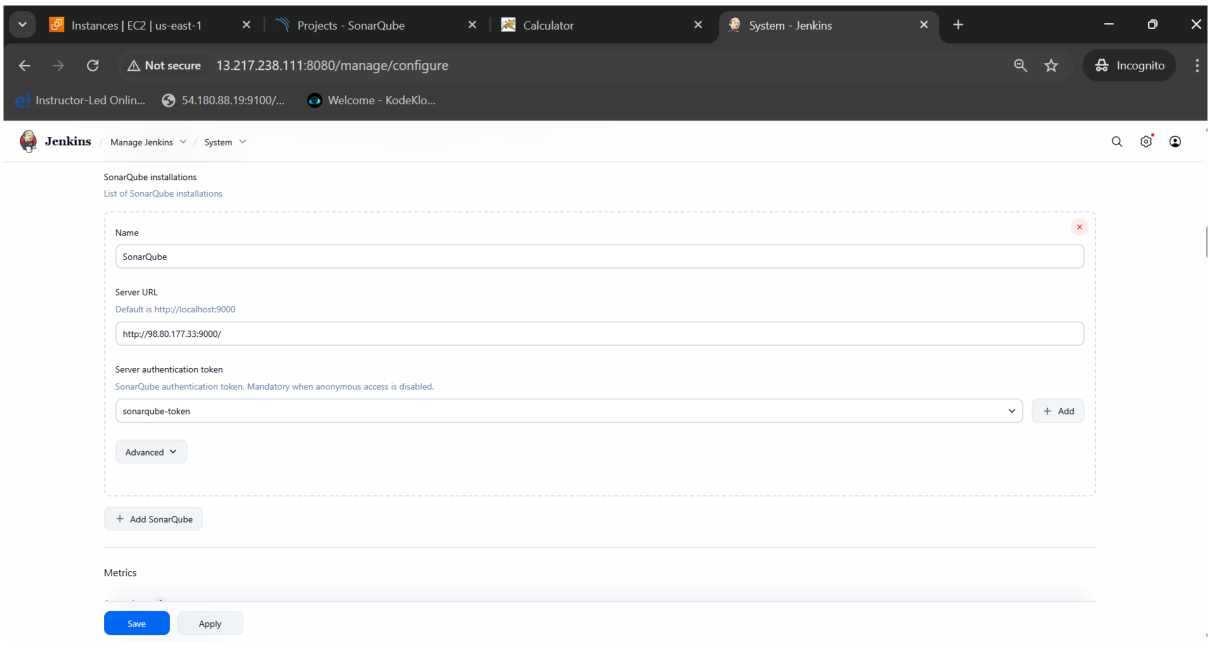
Create these credentials (IDs are used in pipeline/steps):

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



SonarQube installations
List of SonarQube installations

Name
SonarQube

Server URL
Default is http://localhost:9000
http://98.80.177.33:9000/

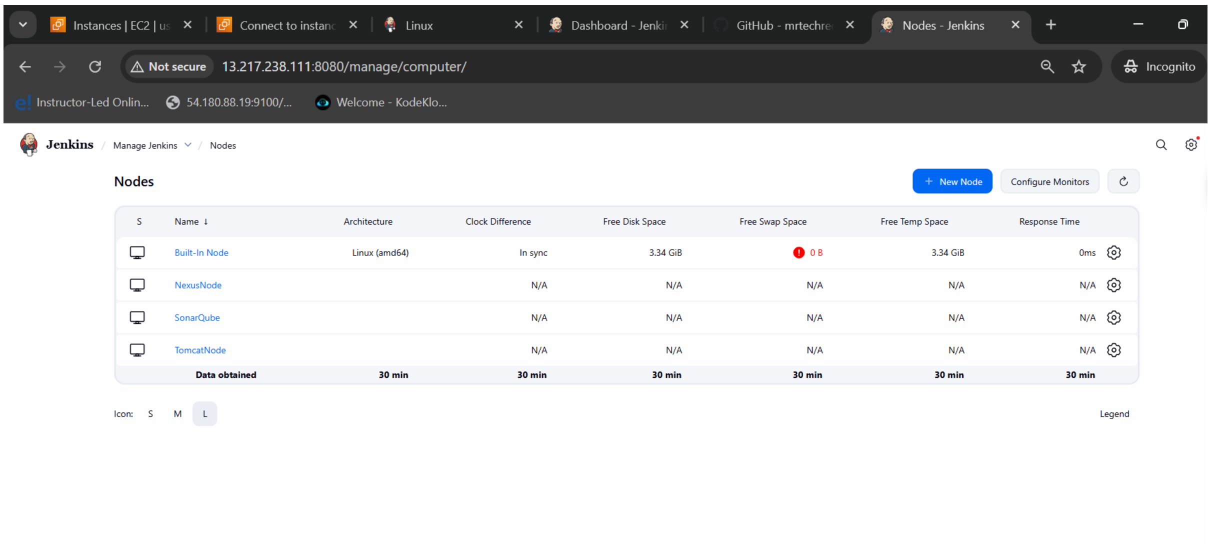
Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
sonarqube-token

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:


13.217.238.111:8080/manage/computer/

Jenkins / Manage Jenkins / Nodes

Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	3.34 GiB	0 B	3.34 GiB	0ms
	NexusNode		N/A	N/A	N/A	N/A	N/A
	SonarQube		N/A	N/A	N/A	N/A	N/A
	TomcatNode		N/A	N/A	N/A	N/A	N/A
	Data obtained	30 min	30 min	30 min	30 min	30 min	30 min

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 'SonarNode' }


  tools {
    jdk 'JDK17'
    maven 'Maven'
  }


  environment {
    SONARQUBE_SERVER = 'SonarQube'
    MVN_SETTINGS = '/home/jenkins/.m2/settings.xml'
    NEXUS_SEARCH_API = 'http://3.227.246.21:8081/service/rest/v1/search'
    NEXUS_REPO = 'maven-releases'
    NEXUS_GROUP = 'com.web.cal'
    NEXUS_ARTIFACT = 'webapp-add'
    TOMCAT_URL = 'http://13.220.167.254:8080/manager/text'
  }


  stages {
    stage('Checkout Code') {
      steps {
        echo '📦 Cloning source from GitHub...'
        checkout([$class: 'GitSCM',
          branches: [[name: '*/main']],
          userRemoteConfigs: [[url: 'https://github.com/you/your-app.git']]
        ])
      }
    }

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

    stage('Build Artifact') {
      steps {
        echo '⚙️ Building WAR...'
        sh 'mvn package -DskipTests --settings ${MVN_SETTINGS}'
        sh 'ls -lh target/*.war || true'
      }
    }

    stage('Upload Artifact to Nexus') {
      steps {
        echo '⬆️ Uploading to Nexus (unique version)...'
        sh '''
          NEW_VERSION="0.0.${BUILD_NUMBER}"
          mvn versions:set -DnewVersion=${NEW_VERSION} --settings
${MVN_SETTINGS}
          mvn deploy -DskipTests --settings ${MVN_SETTINGS}
        '''
      }
    }

    stage('Deploy to Tomcat') {
      agent { label 'TomcatNode' }
      steps {
        withCredentials([
          usernamePassword(credentialsId: 'nexus-creds', usernameVariable:
'NEXUS_USR', passwordVariable: 'NEXUS_PSW'),
          usernamePassword(credentialsId: 'tomcat-manager', usernameVariable:
'TOMCAT_USR', passwordVariable: 'TOMCAT_PSW')
        ]) {
          sh '''
            set -e
            cd /tmp; rm -f *.war

            DOWNLOAD_URL=$(curl -s -u ${NEXUS_USR}:${NEXUS_PSW} \

"${NEXUS_SEARCH_API}?repository=${NEXUS_REPO}&group=${NEXUS_GROUP}
&name=${NEXUS_ARTIFACT}" \
                | grep -oP '"downloadUrl"\\s*:\\s*"\\K[^"]+\\.war' | grep -vE
'\\.md5|\\.sha1' | tail -1)

                [ -n "$DOWNLOAD_URL" ] || { echo "No WAR found"; exit 1; }

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

                curl -u ${TOMCAT_USR}:${TOMCAT_PSW} --upload-file
"$WAR_FILE" \
                      "${TOMCAT_URL}/deploy?path=/${APP_NAME}&update=true"
                  '''
          }
      }
  }
}

post {
  success { echo '🎉 Pipeline completed successfully!' }
  failure { echo '❌ Pipeline failed — check logs.' }
}
}
```

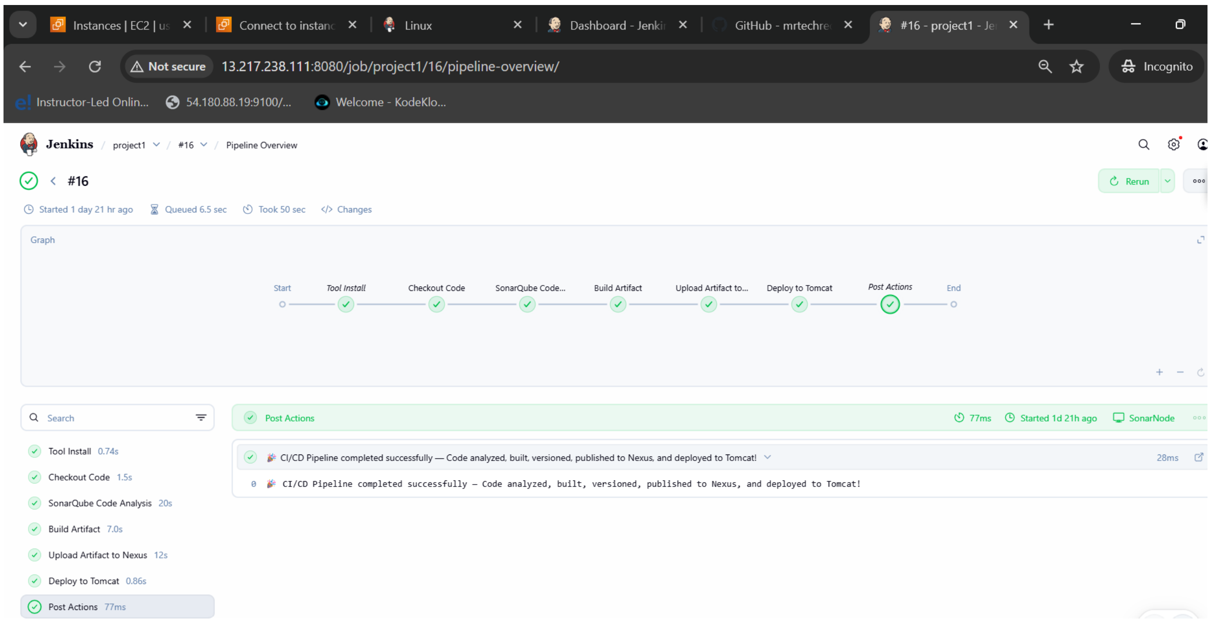
Notes

- 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)



Tool Install 0.74s
Checkout Code 1.5s
SonarQube Code Analysis 20s
Build Artifact 7.0s
Upload Artifact to Nexus 12s
Deploy to Tomcat 0.86s
Post Actions 77ms

Post Actions 77ms Started 1d 21h ago SonarNode

🎉 CI/CD Pipeline completed successfully — Code analyzed, built, versioned, published to Nexus, and deployed to Tomcat! 28ms

```
🎉 CI/CD Pipeline completed successfully – Code analyzed, built, versioned, published to Nexus, and deployed to Tomcat!
```

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>/
