

# Java Web Application CI/CD Pipeline Documentation

## 1. Overview

This document explains how to set up a complete **CI/CD pipeline** for a Java Web Application 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:

- GitHub repository containing your Java web application source code.
- Jenkins master server (Ubuntu 22.04 LTS).
- SonarQube, Nexus, and Tomcat servers (Ubuntu 22.04 LTS).
- SSH connectivity between Jenkins master and all agent servers.
- GitHub Personal Access Token (PAT), SonarQube token, Nexus credentials, and Tomcat Manager credentials.

## 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 apt update
sudo apt install fontconfig openjdk-21-jre
java -version
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
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
```

```
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: 215.8M (peak: 216.2M)
```

```
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):

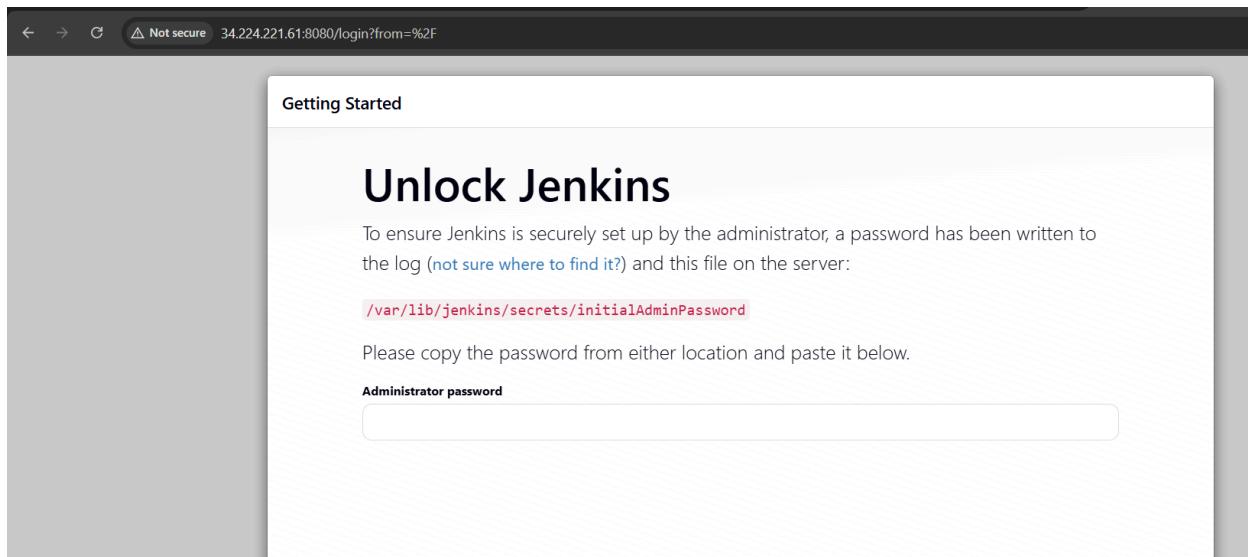
The system will reboot now!
```

## 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.

A screenshot of the Jenkins 'Plugins' management interface. At the top, there's a circular icon with a puzzle piece symbol and the word 'Plugins'. Below it, a descriptive text reads: 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins.' Two plugins are listed: 'GitHub Integration 0.7.2' and 'GitHub Authentication 651.v135e939e8b\_60'. Both have a checked checkbox icon to their left. Under each plugin, there are tabs for 'emailext', 'Build Triggers', 'github', 'Security', and 'Authentication and User Management'. Descriptions for each plugin are provided below their respective tabs.

## Generic Webhook Trigger 2.4.1

notification    github    webhook    Build Parameters    gitlab    Build Triggers    bitbucket  
bitbucket-server    jira

2

Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.

## Maven Integration 3.27

Build Tools

This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.

## 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

Pipeline: Stage View 2.38  
User Interface  
Pipeline Stage View Plugin.

Delivery Pipeline 1.4.2  
User Interface  
This plugin visualize Delivery Pipelines (Jobs with upstream/downstream dependencies)

## Step 5: Configure Global Tools

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

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



Tools

Configure tools, their locations and automatic installers.

```
ubuntu@SonarQube:~$ readlink -f $(which java)
/usr/lib/jvm/java-17-openjdk-amd64/bin/java
ubuntu@SonarQube:~$ |
```

+ Add JDK

☰ JDK

Name

JDK17

JAVA\_HOME

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

Install automatically ?

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

☰ JDK

Name

JDK21

JAVA\_HOME

/usr/lib/jvm/java-21-openjdk-amd64

Install automatically ?

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

☰ Maven

Name

Maven

MAVEN\_HOME

/usr/share/maven

Install automatically ?

## 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+Lsswn1vilU07YQt9a24lFEeIiwbxBhTcoNYaIDg jenkins@Jenkins
The key's randomart image is:
+--[ED25519 256]--+
|+ == . .
|E. o+. .
| .+ o=o .
| . o...+ .
```

**SonarQube:**

```
GNU nano 7.2
authorized_keys *
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDaRu0y1AEdXFKMcoYSjQ+QRSGhgDVEDyqP6m3PCpHCu3wdxztTNsNDe1gP5L4sZYjNjISQ8SHlkvrSJjd>
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGvAupKMk3B04B/iH+zY/PmVPtyjzGASHPSVRcL70B0z 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/LMd7ycwr039wSHHfBJCM0VbyAzKXoiYNMhh7DU.
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 ▾

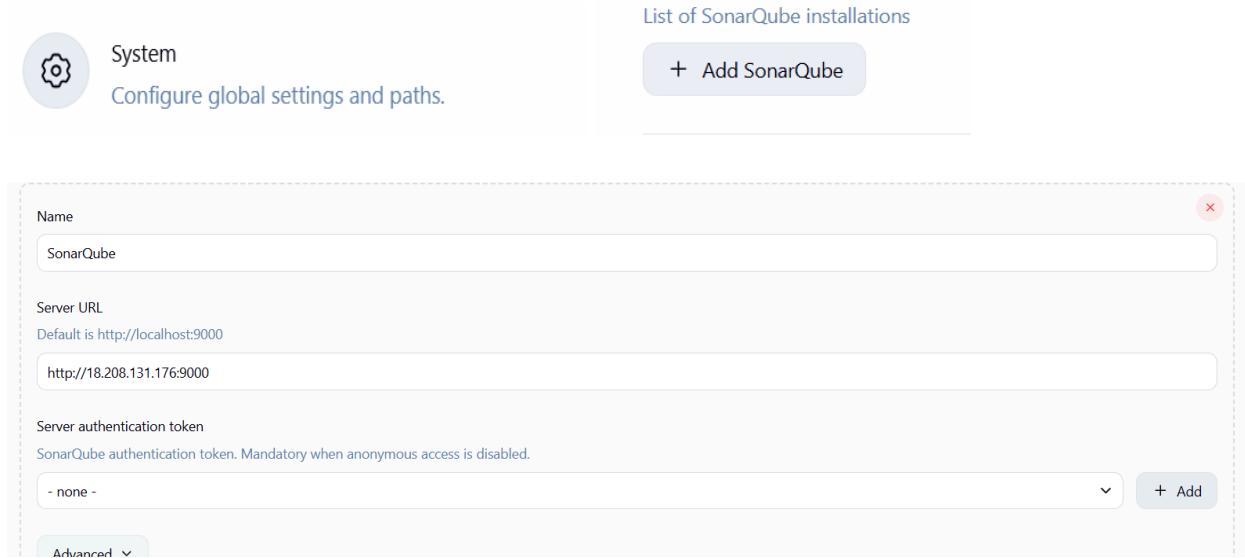
The screenshot shows the Jenkins Pipeline Configuration screen. On the left, under the 'Kind' section, 'SSH Username with private key' is selected. On the right, the 'ID' field is set to 'ubuntu', and the 'Username' field is also set to 'ubuntu'. Below this, the 'Private Key' section is expanded, showing the option 'Enter directly' selected. A large text area contains a long string of characters representing an SSH private key, starting with '-----BEGIN OPENSSH PRIVATE KEY-----' and ending with '-----END OPENSSH PRIVATE KEY-----'.

## 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

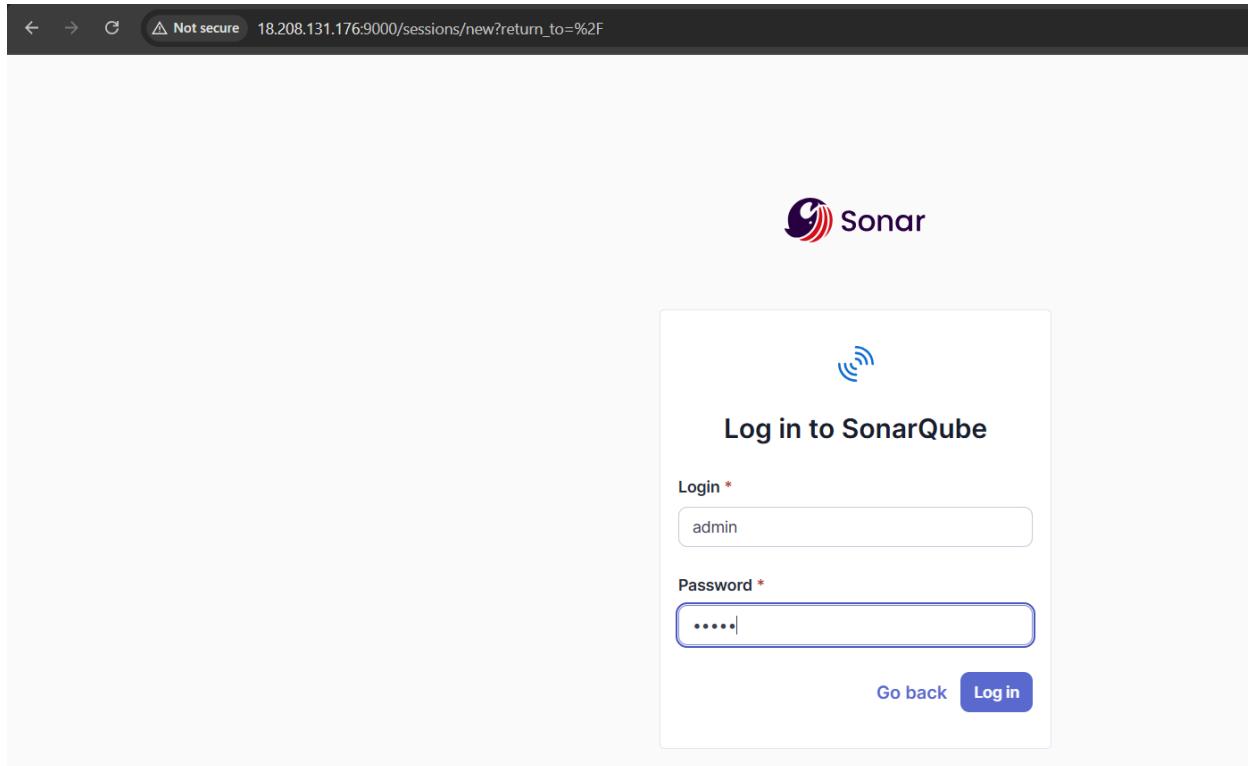


#### 4. SonarQube Setup (Summary)

- 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
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [
```

```
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$ |
```



### Tokens of Administrator

**Generate Tokens**

Name      Expires in  
       

✓ New token "SonarQube-Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

Name	Type	Project	Last use	Created	Expiration
squ_21586efb87b9b458e4af8102f1dc3e8193d2b837					

## 5. Nexus Setup (Summary)

- Install Nexus Repository OSS with Java 17.
- Start the service and access `http://<nexus-ip>:8081`.
- Login with admin credentials and create a hosted repository named `maven-releases`.
- Note the repository URL for your Maven settings.

## 6. Tomcat Setup (Summary)

- Install Tomcat9 and Tomcat9-admin:

```
sudo apt install -y tomcat9 tomcat9-admin
```

- Add the manager user to /etc/tomcat9/tomcat-users.xml.
- Restart Tomcat and test http://<tomcat-ip>:8080/manager.
- Create a jenkins directory so that jenkins can use that as the workspace.

```
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
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [1

<user username="role1" password="" roles="role1">
-->
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
</tomcat-users>
```

```
ubuntu@ip-172-31-23-240:~$ mkdir jenkins
ubuntu@ip-172-31-23-240:~$ ls
apache-tomcat-9.0.111  apache-tomcat-9.0.111.tar.gz  jenkins
ubuntu@ip-172-31-23-240:~$ cd .ssh/
ubuntu@ip-172-31-23-240:~/ssh$ ls
authorized_keys
ubuntu@ip-172-31-23-240:~/ssh$ sudo nano authorized_keys
ubuntu@ip-172-31-23-240:~/ssh$ |
```

The screenshot shows the Tomcat Web Application Manager interface. At the top, there's a navigation bar with links for 'Manager', 'List Applications', 'HTML Manager Help', 'Manager Help', and 'Server Status'. Below the navigation bar is a section titled 'Applications' with a table. The table has columns for 'Path', 'Version', 'Display Name', 'Running', 'Sessions', and 'Commands'. Under 'Display Name', it lists 'Welcome to Tomcat', 'Tomcat Documentation', 'Servlet and JSP Examples', 'Tomcat Host Manager Application', and 'Tomcat Manager Application'. Under 'Running', all entries are marked as 'true'. Under 'Sessions', the counts are 0, 0, 0, 0, and 1 respectively. Under 'Commands', each row contains buttons for 'Start', 'Stop', 'Reload', and 'Undeploy', along with a link to 'Expire sessions with idle ≥ 30 minutes'. At the bottom of the page, there's a 'Deploy' section with the sub-instruction 'Deploy directory or WAR file located on server'.

## 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.

The screenshot shows the Jenkins 'Nodes' configuration page. At the top, there's a summary section for 'Nodes' with a monitor icon, a brief description, and links for 'Add Node', 'Configure Monitors', and a refresh button. Below this is a table titled 'Nodes' with columns for S (Status), Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. It lists one node: 'Built-In Node' (Linux (amd64)), which was last checked 45 min ago and has 3.96 GiB free disk space, 0 B free swap space, 3.96 GiB free temp space, and a response time of 0ms. There are icons for each column header. At the bottom of the table are filters for 'Icon', 'S' (Status), 'M' (Monitors), and a 'Legend' link.

The screenshot shows the 'New node' creation dialog. It has a title 'New node'. The first field is 'Node name' with the value 'SonarQube'. The next section is 'Type' with a radio button selected for 'Permanent Agent'. A descriptive text explains that this adds a plain, permanent agent to Jenkins. At the bottom is a large blue 'Create' button.

Remote root directory ?

! Remote directory is mandatory

Labels ?

Launch method ?

Launch agents via SSH

Host ?

Credentials ?

Host Key Verification Strategy ?

Non verifying Verification Strategy

Advanced ▾

Nodes								<a href="#">+ New Node</a>	Configure Monitors		
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time				
	Built-In Node	Linux (amd64)	In sync	3.57 GiB	<span style="color: red;">! 0 B</span>	3.57 GiB	0ms				
	SonarQube	Linux (amd64)	In sync	2.33 GiB	<span style="color: red;">! 0 B</span>	2.33 GiB	50ms				
	last checked	5.9 sec	5.8 sec	5.8 sec	5.8 sec	5.8 sec	5.8 sec				

Icon: S M L Legend

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

Type

Permanent Agent  
Adds a plain, permanent agent to Jenkins. This is the most common way to integrate with these agents, such as dynamically adding a slave to a master. For example such as when you are adding a physical host to a cloud provider.

Copy Existing Node

Remote root directory [?](#)

! Remote directory is mandatory

**Create**

### Labels [?](#)

Host [?](#)

Credentials [?](#)

- current -

Host Key Verification Strategy [?](#)

Advanced ▾

Nodes							
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
💻	Built-In Node	Linux (amd64)	In sync	3.57 GiB	0 B	3.57 GiB	0ms ⚙️
💻	SonarQube	Linux (amd64)	In sync	2.33 GiB	0 B	2.33 GiB	21ms ⚙️
💻	tomcat	Linux (amd64)	In sync	4.45 GiB	0 B	4.45 GiB	40ms ⚙️
	last checked	1 min 43 sec	1 min 43 sec	1 min 43 sec	1 min 43 sec	1 min 43 sec	1 min 43 sec

## 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.

## New node

Node name

Type

Permanent Agent

Adds a plain, permanent agent to Jenkins integration with these agents, such as an example such as when you are adding

Copy Existing Node

### Remote root directory ?

! Remote directory is mandatory

### Labels ?

Launch method ?

Launch agents via SSH

Host ?

44.220.152.119

Credentials ?

ubuntu

Host Key Verification Strategy ?

Non verifying Verification Strategy

Advanced ▾

Nodes					
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space
💻	Built-In Node	Linux (amd64)	In sync	3.55 GiB	! 0 B
💻	nexus	Linux (amd64)	In sync	3.48 GiB	! 0 B

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

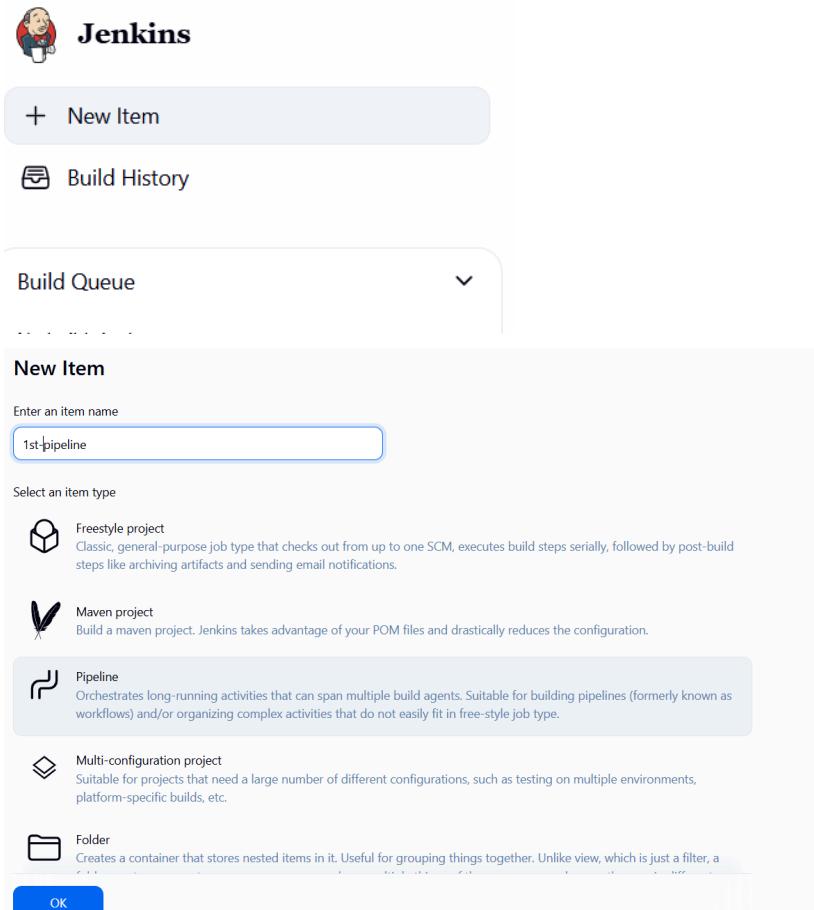
Nodes							
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
💻	Built-In Node	Linux (amd64)	In sync	3.55 GiB	! 0 B	3.55 GiB	0ms ⚙️
💻	nexus	Linux (amd64)	In sync	3.48 GiB	! 0 B	3.48 GiB	61ms ⚙️
💻	SonarQube	Linux (amd64)	In sync	2.32 GiB	! 0 B	2.32 GiB	46ms ⚙️
💻	tomcat	Linux (amd64)	In sync	4.45 GiB	! 0 B	4.45 GiB	44ms ⚙️
last checked		59 sec	59 sec	59 sec	59 sec	59 sec	59 sec

## 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



**New Item**

Enter an item name  
1st-pipeline

Select an item type

- Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a

**OK**

```

1< pipeline{
2   agent none
3<   stages{
4<     stage('sonar node'){
5       agent {label 'sonar'}
6<       steps{
7<         script{
8           //run the commands
9           echo 'hello from the sonar'
10          sh 'whoami'
11          sh 'hostname -i'
12        }
13      }
14    }
15

```

```

12           sh 'df -h'
13       }
14   }
15 }
16 v stage('nexus node'){
17     agent {label 'nexus'}
18 v     steps{
19 v         script{
20             //run the commands
21             echo 'hello from the nexus'
22             sh '''whoami
23                 hostname -i
24                 df-h'''
25         }
26     }

```

```

27     }
28 v     stage('tomcat node'){
29     agent {label 'tomcat'}
30 v     steps{
31 v         script{
32             //run the commands
33             echo 'hello from tomcat'
34             sh '''whoami
35                 hostname -i
36                 df -h'''
37         }
38     }
39 }
40 }
41 }

```

## 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

 [[SCREENSHOT: GitHub webhook setup page]]

## 9 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>/>

The screenshot shows the Jenkins Pipeline interface for the 'pipe' job. On the left, there's a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, Pipeline Syntax, and Credentials. Below that is a 'Builds' section with a 'Filter' input and a 'Today' dropdown showing builds #4 and #3. The main area is titled 'Stage View' and displays three stages: sonar-mvn, nexus, and tomcat. Each stage has a horizontal bar indicating average stage times: 1s for all. Below the bars is a grid of build results for each stage across four builds. Builds #4 and #3 show green '695ms' and '1s' respectively. Build #2 shows a red '1s' with a 'failed' status. Build #1 shows a red '1s' with a 'failed' status. The URL at the bottom is 18.234.35.70:8080/job/pipe.

The screenshot shows a terminal window on an Ubuntu system (SONAR-MVN) displaying the contents of the Jenkins workspace. The user runs 'ls' in the root directory, then navigates to the 'jenkins/workspace/pipe' directory and runs 'ls' again. This shows sub-directories 'caches', 'remoting', 'sonar', 'workspace', and 'tmp'. The user then runs 'cat sonar.txt' which outputs the file content: 'hi! from sonar' and 'hi! from sonar'. The URL at the bottom is 18.234.35.70:8080/job/pipe/build?delay=0sec.

```

ubuntu@SONAR-MVN:~/.jenkins$ ls
caches  remoting  workspace
ubuntu@SONAR-MVN:~/.jenkins$ cd workspace/
ubuntu@SONAR-MVN:~/jenkins/workspace$ ls
sonar  tmp
ubuntu@SONAR-MVN:~/jenkins/workspace$ cd pipe
ubuntu@SONAR-MVN:~/jenkins/workspace/pipe$ ls
sonar.txt
ubuntu@SONAR-MVN:~/jenkins/workspace/pipe$ cat sonar.txt
hi! from sonar
hi! from sonar
ubuntu@SONAR-MVN:~/jenkins/workspace/pipe$
```

## 10. Final Checklist

- [ ] GitHub repo and Jenkinsfile verified
- [ ] Jenkins configured with tools & credentials
- [ ] SonarQube, Nexus, and Tomcat running
- [ ] SSH connectivity verified
- [ ] GitHub webhook added and working
- [ ] Successful build and deployment confirmed.