AUTOMATION DEPLOYMENT USING JENKINS FREE STYLE AND PIPELINE JOBS:

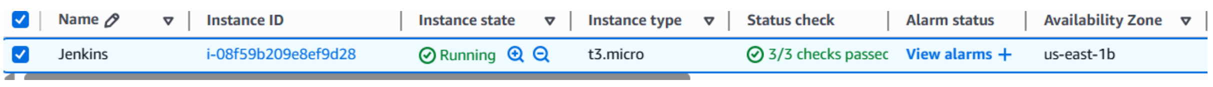
TOOLS: **GIT, GITHUB, JENKINS, MAVEN, NEXUS, SONAR & TOMCAT.**

DESCRIPTION: WRITE A JENKINS PIPELINE TO GET THE SOURCE CODE FROM GITHUB TO CI SERVER AND BUILD THE CODE USING MAVEN AND STORING THE ARTIFACT's ON NEXUS AND WILL BE ABLE TO ROLL BACK ONCE IT FAILED. SCAN THE SOURCE CODE USING SONARQUBE TO CHECK THE BUGS AND CODE SMELLS. DEPLOY THE WEB APPLICATION ON A APPLICATION SERVER LIKE TOMCAT.

JENKINS Server Setup:

Step 1:

- Launch an EC2- Instance, Instance type is t3.micro with NewEc2KeyPair.pem file and Security group of MY-SG allowing inbound rules of
- SSH 22 from anywhere IP4
- TCP Port 8080-8081 from anywhere IP4
- TCP Port 9000 from anywhere IP4
- Storage volume 8gb - gp3



✓	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
✓	Jenkins	i-08f59b209e8ef9d28	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1b

Step 2:

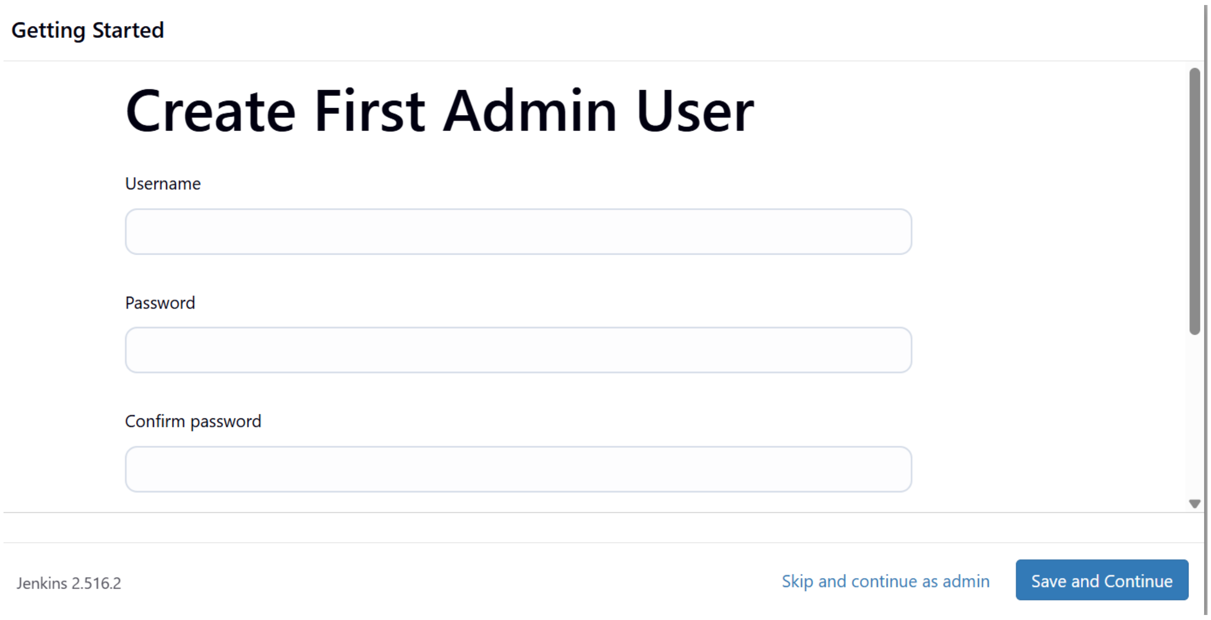
- Now connect to the instances via CMD
- ssh -i "NewEc2KeyPair.pem" ec2-user@ec2-54-159-98-78.compute-1.amazonaws.com

Step 3:

- Change the hostname of the sever just to differentiate which server you are working on
- Commands: hostnamectl set-hostname "Jenkins"
- Now we will install the GIT, JENKINS, JAVA-17. Git is used to get the code to our server, Jenkins for Automating the pipeline, Java-17 is dependencies for Jenkins after installing these tools
- Now start the Jenkins service using: systemctl start Jenkins command

Step 4: [launching the Jenkins]

- Open the Jenkins Instance in AWS console, copy the Public IP and paste it in the browser with port 8080 Example: Publicip:8080
- In the Jenkins server do cat /var/lib/Jenkins/....
- Install the suggested Plugins and create the admin user


Getting Started

Create First Admin User

Username

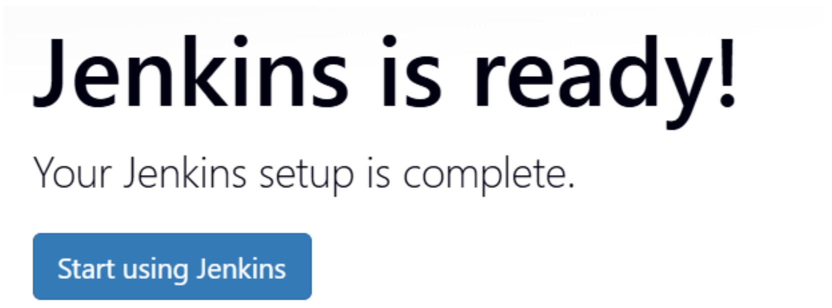
Password

Confirm password

Jenkins 2.516.2

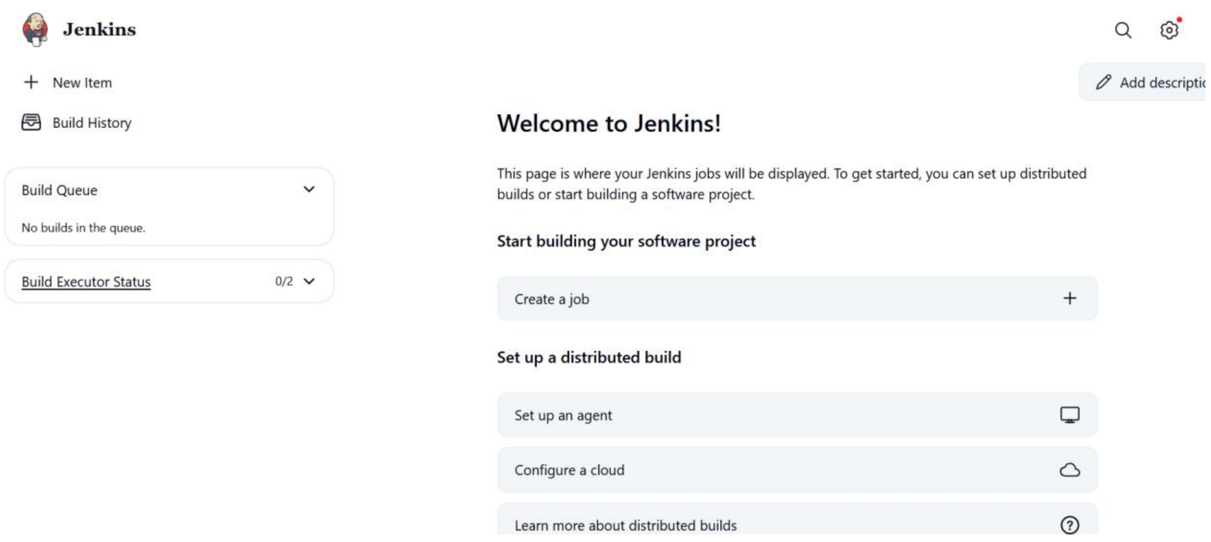
Step 5:

- Jenkins Set up is completed


Jenkins is ready!

Your Jenkins setup is complete.

Creating the Pipeline Job:


Jenkins

New Item

Build History

Add descriptio

Build Queue

No builds in the queue.

Build Executor Status 0/2

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

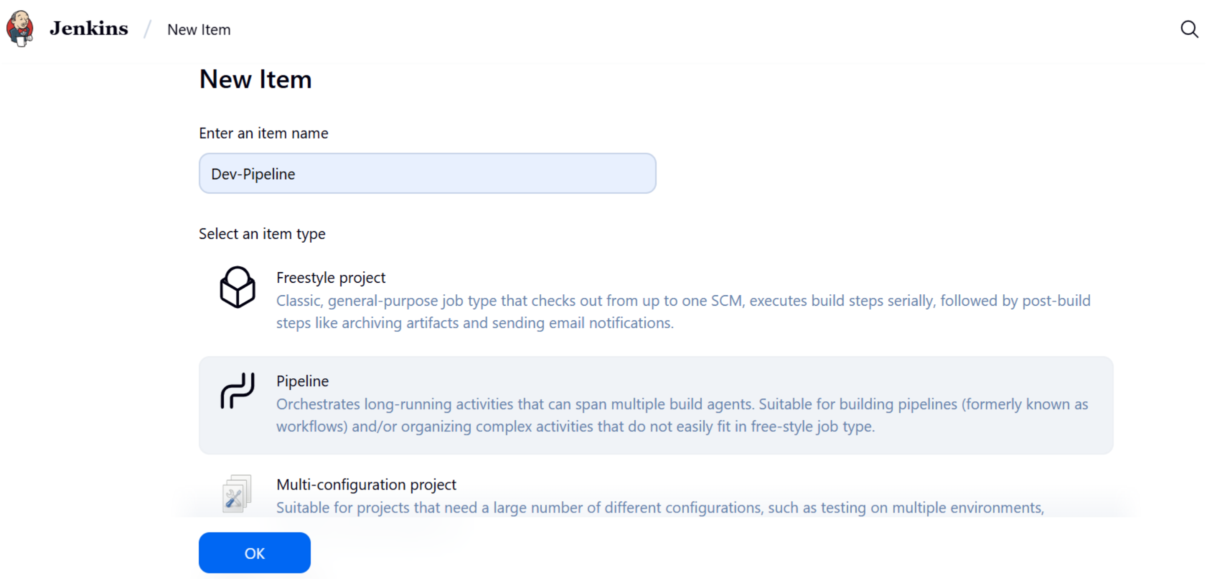
Set up an agent

Configure a cloud

Learn more about distributed builds

Step 1:

- Click on New item
- Enter an item name and Select Pipeline Job


Jenkins / New Item

New Item

Enter an item name

Dev-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.

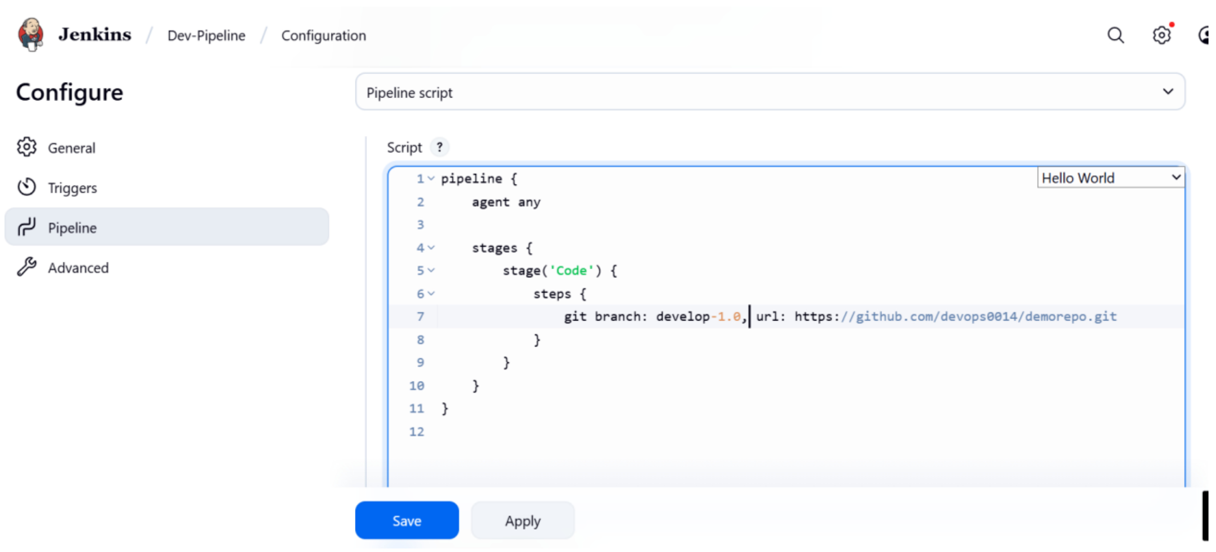
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments,

OK

Step 2:

- We need to configure the pipeline job
- First, we create pipeline script for our job
- Pipeline job syntax is PASSS
    - P = pipeline
    - A = agent any
    - S = stages
    - S = stage
    - S =steps



```
pipeline {
    agent any

    stages {
        stage('Code') {
            steps {
                git branch: develop-1.0, url: https://github.com/devops0014/demorepo.git
            }
        }
    }
}
```

Pipeline Script:

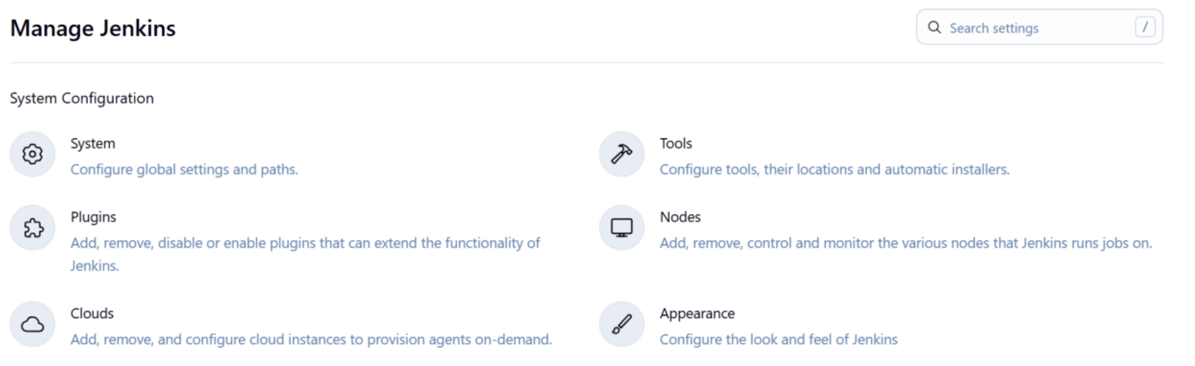
- git branch = branch name of the github repo
- url = github repo url

Code:

```
pipeline {
  agent any
  stages {
    stage(‘Code’) {
      steps {
        git branch: 'develop-1.0', url: 'https://github.com/devops0014/demorepo.git'
}
      }
   }
}
```

Configuring Build Tool: MAVEN

- Click on settings button = Manage Jenkins
- Open the tools section here we will configure the build tools



Manage Jenkins

Search settings

System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

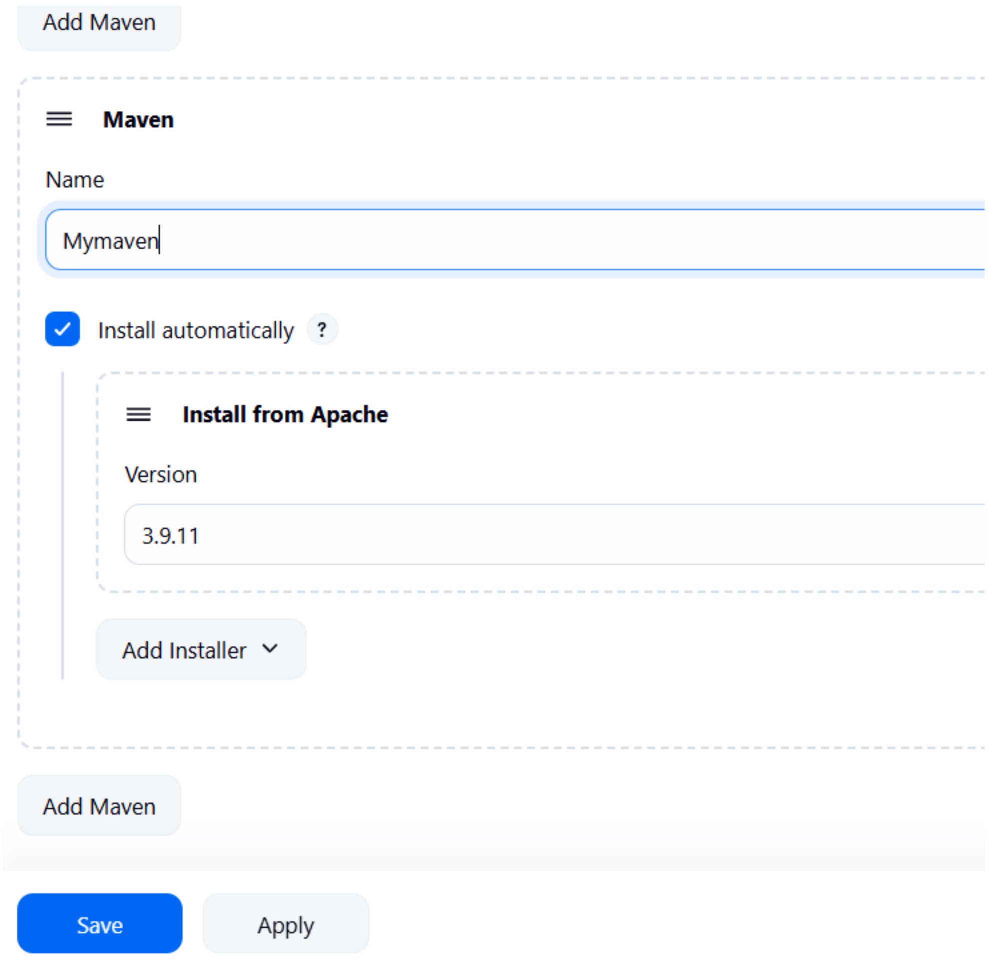
Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

Appearance
Configure the look and feel of Jenkins

- Click on Add maven
- Provide name as “Mymaven”
- Select the latest version and save
- Go to the Pipeline Job
- Open the pipeline script and add build script as mentioned as below

Code:

```
pipeline {
  agent any
  tools {
    maven 'Mymaven'
  }
  stages {
    stage('Code') {
      steps {
        git branch: 'develop-1.0', url: 'https://github.com/devops0014/demorepo.git'


      }
    }
    stage('Build') {
      steps {
        sh 'mvn clean package'
      }
    }
  }
}
```



Add Maven

Maven

Name

Mymaven

☑ Install automatically

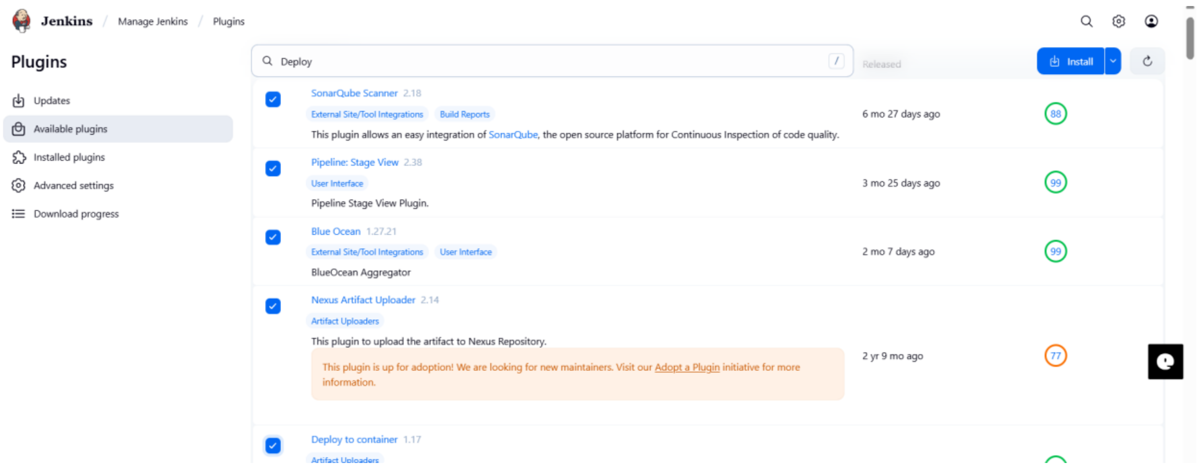
Install from Apache

Version

3.9.11

Installing The Plugins:

- Open Manage Jenkins
- Select Plugins and Available Plugins
- Pipeline Stage View and Blue Ocean Plugin for Pipeline job UI
- Sonar Scanner plugin for SonarQube
- Nexus Artifact Uploader for Nexus
- Deploy to container for Tomcat

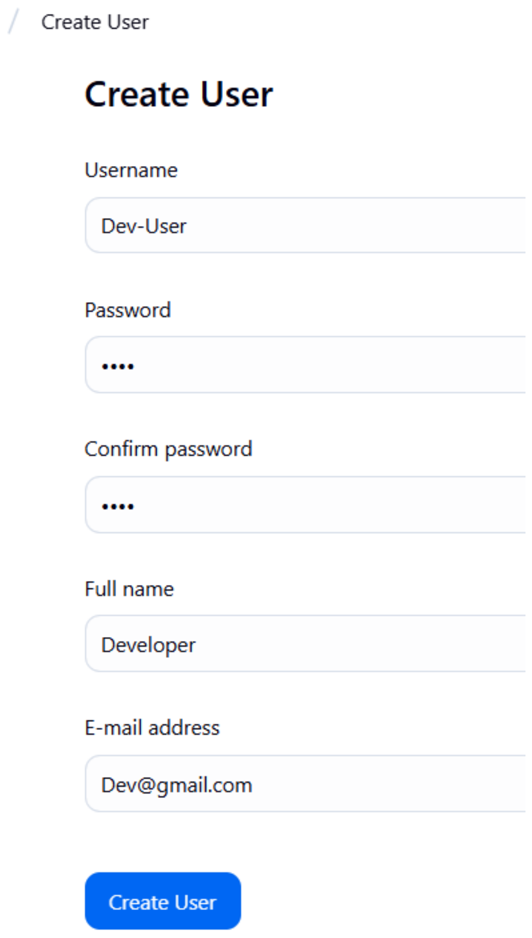


	Plugin	Released	
✓	SonarQube Scanner 2.18 External Site/Tool Integrations · Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	6 mo 27 days ago	88
✓	Pipeline: Stage View 2.38 User Interface Pipeline Stage View Plugin.	3 mo 25 days ago	99
✓	Blue Ocean 1.27.21 External Site/Tool Integrations · User Interface BlueOcean Aggregator	2 mo 7 days ago	99
✓	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.	2 yr 9 mo ago	77
✓	Deploy to container 1.17 Artifact Uploaders		

Creating a User:

- Manage Jenkins and open the users
- Fill the required details and save
- Open the security and select project-based authorization strategy
- Add the user which we created earlier
- Provide the only Read option to the user
- Add the admin user and provide administrator access
- Now go to the pipeline job and enable-project based security option
- Add the user, give job access as read, view and build

Screenshot: 1

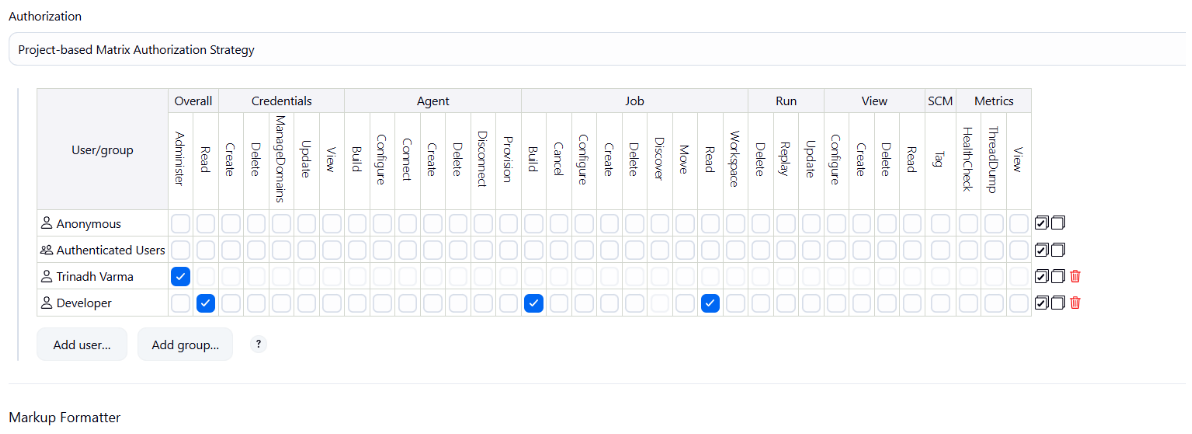


/ Create User

Create User

Username	Dev-User
Password	••••
Confirm password	••••
Full name	Developer
E-mail address	Dev@gmail.com

Screenshot: 2



Authorization

Project-based Matrix Authorization Strategy

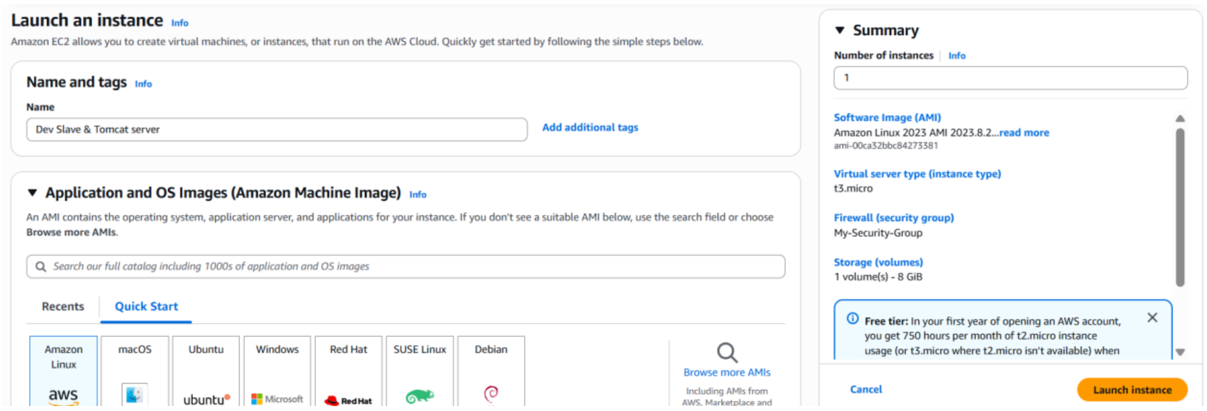
User/group	Overall		Credentials					Agent							Job									Run			View				SCM	Metrics		
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Configure	Create	Delete	Read	Tag	HealthCheck	ThreadDump	View
Anonymous																																		
Authenticated Users																																		
Trinadh Varma	✓																																	
Developer		✓													✓							✓												

Markup Formatter

Slave Server Set-up:

Step 1:

- Launch the New EC2- Instance with previous instance details but take EBS volume size as 28GB



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: Dev Slave & Tomcat server

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose Browse more AMIs.

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux | Debian

Browse more AMIs — Including AMIs from AWS, Marketplace and

Summary

Number of instances Info: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.8.2...read more ami-00ca32bbc84273381

Virtual server type (instance type): t3.micro

Firewall (security group): My-Security-Group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when

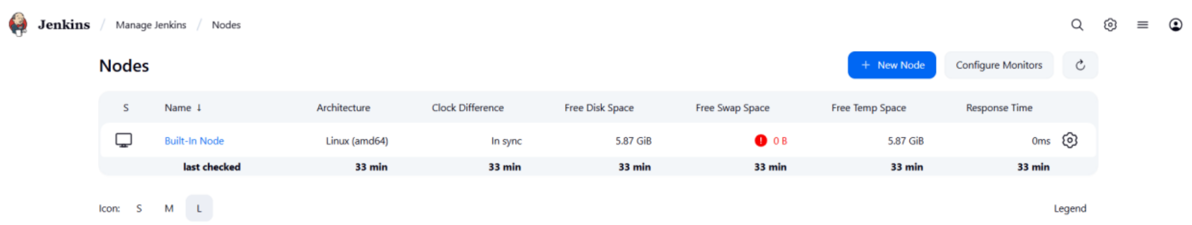
Cancel | Launch instance

Step 2:

- Connect to the server via CMD / Powershell
- Change the hostname – hostnamectl set-hostname “Slave”
- Install Java and git using below commands
- Java Command: sudo yum install java-17-amazon-corretto -y
- Git Command: sudo yum install git -y

Step 3:

- Manage Jenkins → Nodes
- Create a New Node



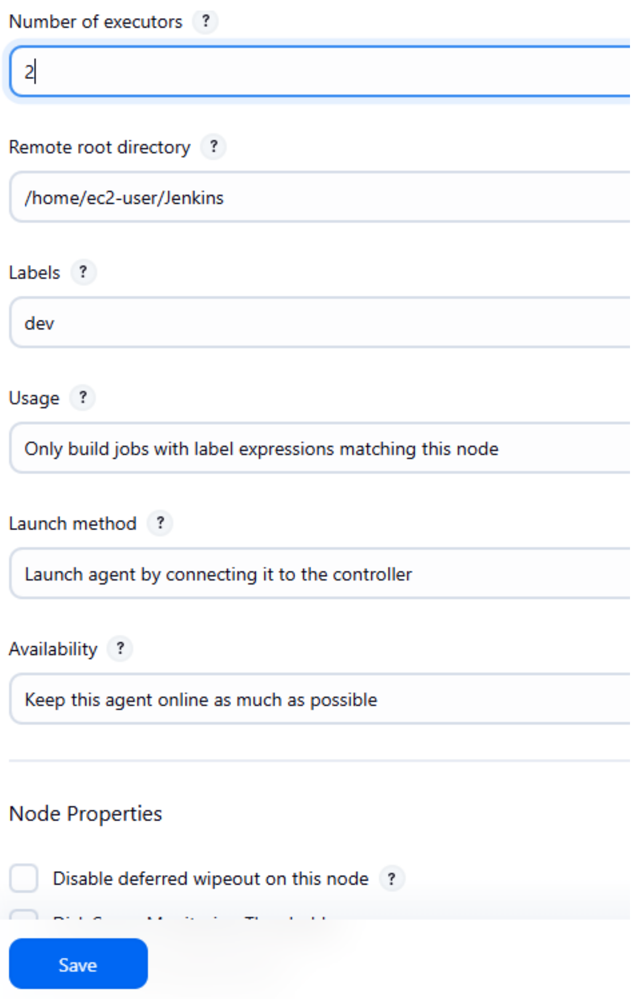
Jenkins / Manage Jenkins / Nodes

Nodes — + New Node | 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	5.87 GiB	● 0 B	5.87 GiB	0ms
	last checked	33 min	33 min	33 min	33 min	33 min	33 min

Icon: S M L — Legend

- Config the details as below



Number of executors ?

2

Remote root directory ?

/home/ec2-user/Jenkins

Labels ?

dev

Usage ?

Only build jobs with label expressions matching this node

Launch method ?

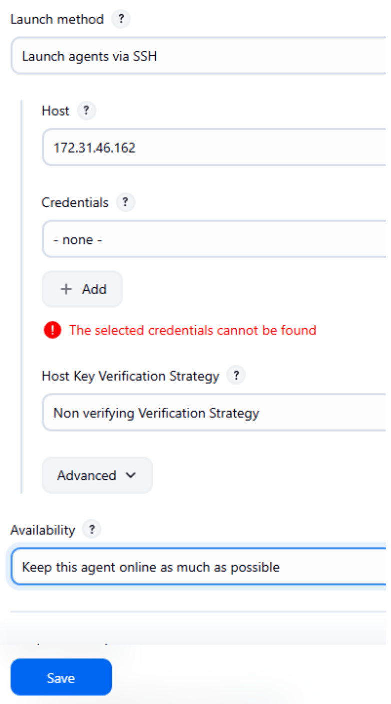
Launch agent by connecting it to the controller

Availability ?

Keep this agent online as much as possible

Node Properties

☐ Disable deferred wipeout on this node ?



Launch method ?

Launch agents via SSH

Host ?

172.31.46.162

Credentials ?

- none -

The selected credentials cannot be found

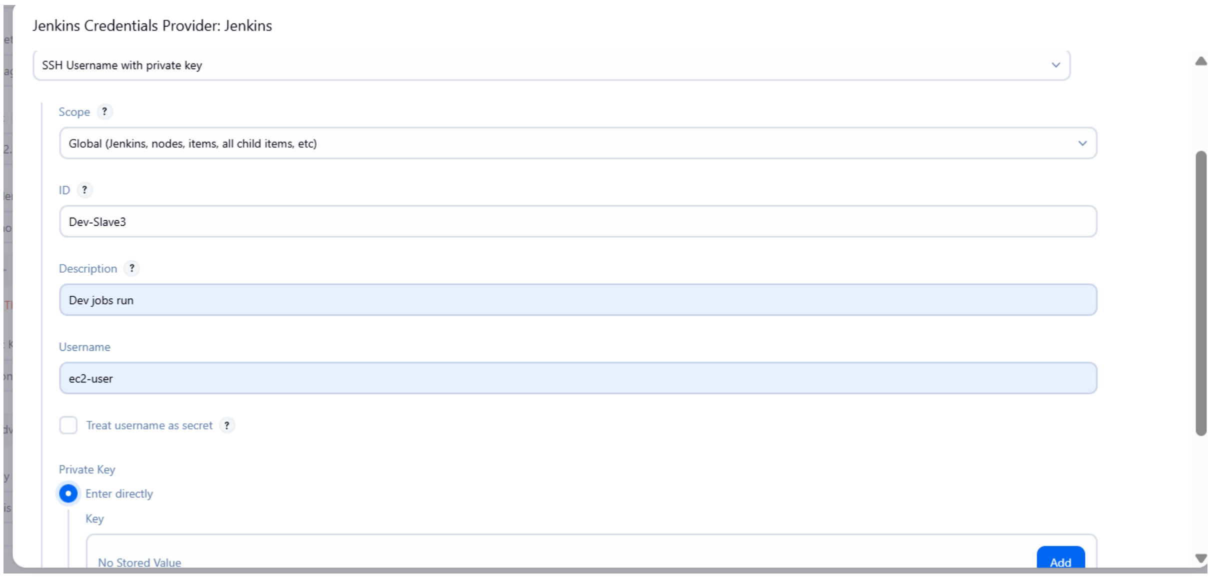
Host Key Verification Strategy ?

Non verifying Verification Strategy

Availability ?

Keep this agent online as much as possible

- Setting up Credentials for slave server



Jenkins Credentials Provider: Jenkins

SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID: Dev-Slave3

Description: Dev jobs run

Username: ec2-user

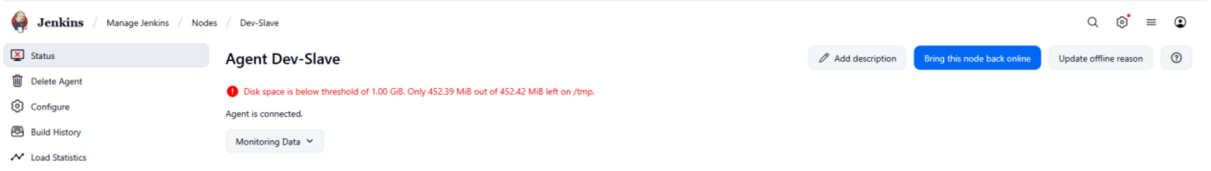
☐ Treat username as secret

Private Key
● Enter directly

Key: No Stored Value

Step 4:

- Node has been created but it was gone to offline due to disk space of issue of /tmp directory



Jenkins / Manage Jenkins / Nodes / Dev-Slave

Agent Dev-Slave

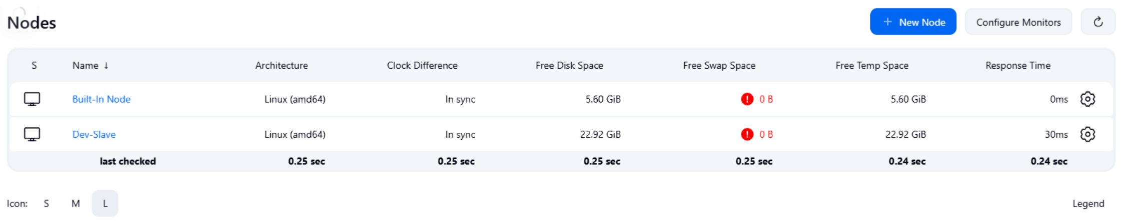
Disk space is below threshold of 1.00 GiB. Only 452.39 MiB out of 452.42 MiB left on /tmp.

Agent is connected.

- To bring Node online run the below commands in the slave server
- sudo mkdir -p /var/tmp_disk
- sudo chmod 1777 /var/tmp_disk
- sudo mount --bind /var/tmp_disk /tmp
- echo '/var/tmp_disk /tmp none bind 0 0' | sudo tee -a /etc/fstab
- sudo systemctl mask tmp.mount
- df -h /tmp

Step 5:

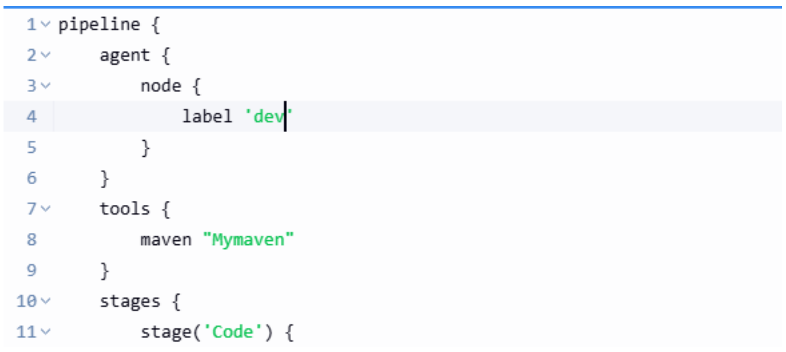
- After running the above commands
- Node is back to the online


Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
🖥	Built-In Node	Linux (amd64)	In sync	5.60 GiB	0 B	5.60 GiB	0ms
🖥	Dev-Slave	Linux (amd64)	In sync	22.92 GiB	0 B	22.92 GiB	30ms
	last checked	0.25 sec	0.25 sec	0.25 sec	0.25 sec	0.24 sec	0.24 sec

Step 6:

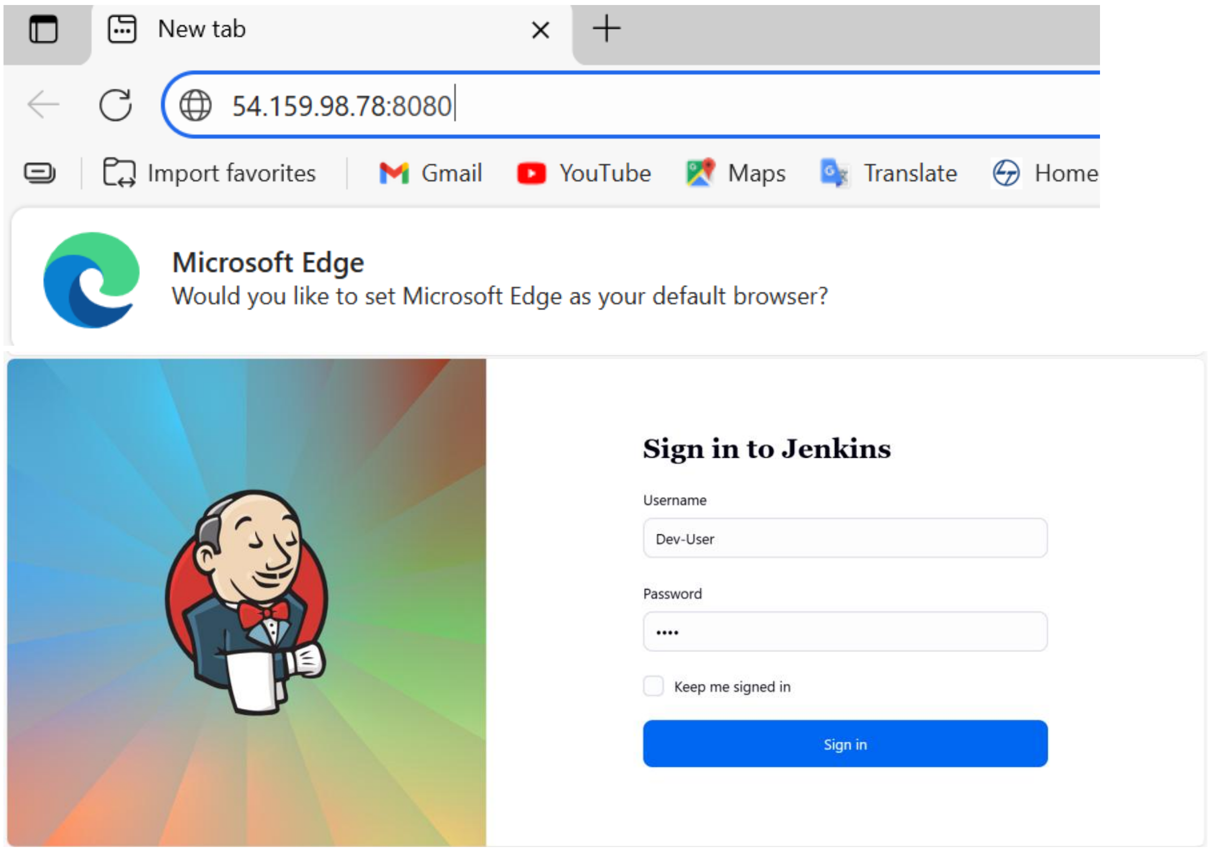
- Open the pipeline job
- In the pipeline job add node section
- Give label as dev matching it with Dev-Slave node


```
pipeline {
    agent {
        node {
            label 'dev'
        }
    }
    tools {
        maven "Mymaven"
    }
    stages {
        stage('Code') {
```

Logging into Jenkins as Developer user:

Step 1:

- Open a new search engine
- Provide Jenkins publicip:8080
- Log into the username and password


New tab

54.159.98.78:8080

Microsoft Edge
Would you like to set Microsoft Edge as your default browser?

Sign in to Jenkins

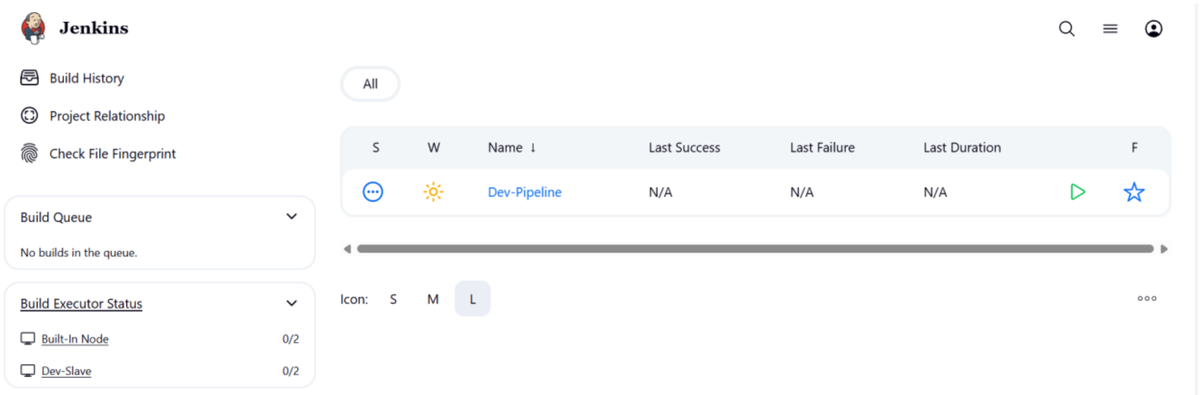
Username: Dev-User

Password: ••••

Keep me signed in

Step 2:

- Here the UI from developer account
- Open the dev-pipeline
- Click on build now



Build Queue

No builds in the queue.

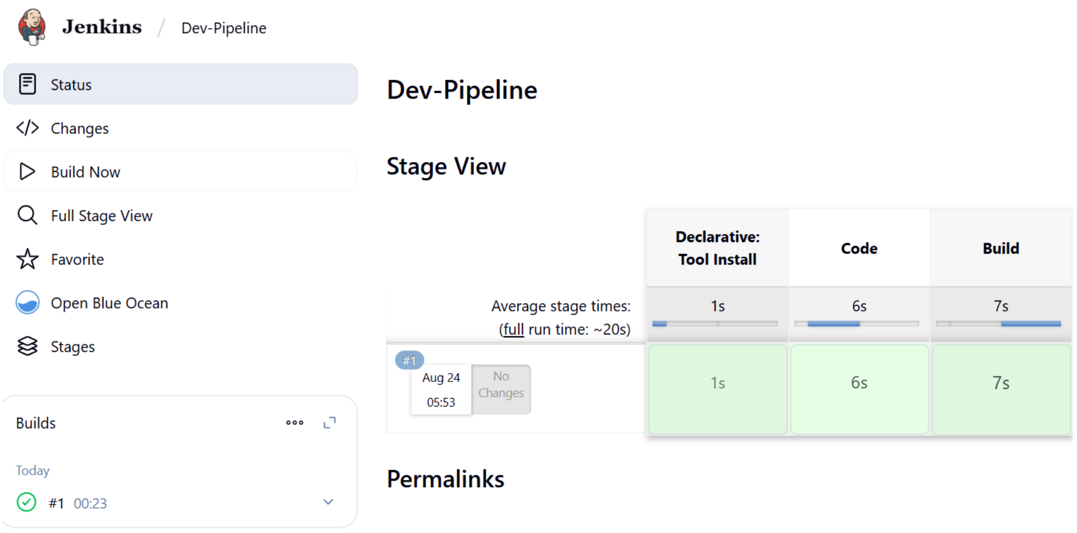
Build Executor Status

Built-In Node	0/2
Dev-Slave	0/2

S	W	Name ↓	Last Success	Last Failure	Last Duration	F
		Dev-Pipeline	N/A	N/A	N/A	

Step 3:

- Successfully Code and build is completed
- Stage view is coming because of we installed pipeline stage view plugin



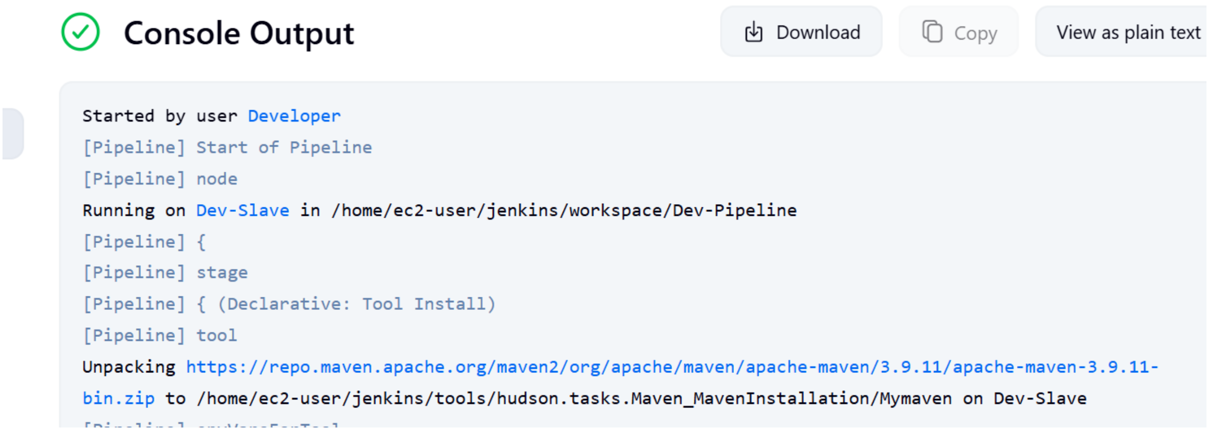
Dev-Pipeline

Stage View

	Declarative: Tool Install	Code	Build
Average stage times: (full run time: ~20s)	1s	6s	7s
#1 Aug 24 05:53 No Changes	1s	6s	7s

Permalinks

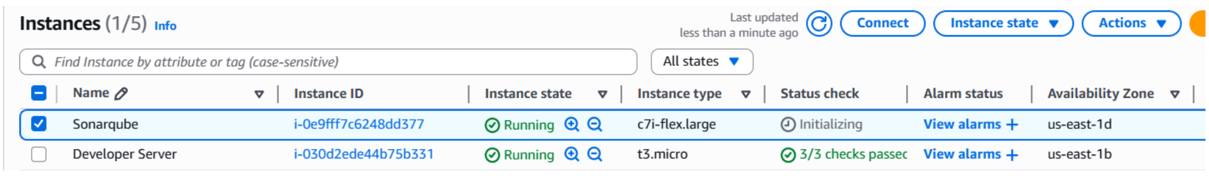
- From the developer account we are able to see the console output
- See the line number 4 as “Running on Dev—slave”
- The job is started by Developer and running on Dev-slave node



```
Started by user Developer
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Dev-Slave in /home/ec2-user/jenkins/workspace/Dev-Pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
Unpacking https://repo.maven.apache.org/maven2/org/apache/maven/apache-maven/3.9.11/apache-maven-3.9.11-
bin.zip to /home/ec2-user/jenkins/tools/hudson.tasks.Maven_MavenInstallation/Mymaven on Dev-Slave
```

Launching SonarQube Server:

- Launch an ec2-instance, Instance type as c7i-flex large
- Remaining config same
- EBS volume as 28 gb


Instances (1/5) Info

Last updated less than a minute ago

Connect | Instance state | Actions

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

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☑	Sonarqube	i-0e9fff7c6248dd377	Running	c7i-flex.large	Initializing	View alarms	us-east-1d
☐	Developer Server	i-030d2ede44b75b331	Running	t3.micro	3/3 checks passec	View alarms	us-east-1b

Step 1:

- Installing the SonarQube
- Run the below commands in SonarQube server

- cd /opt/
- wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.9.6.50800.zip
- unzip sonarqube-8.9.6.50800.zip
- yum install java-17-amazon-corretto -y
- useradd sonar
- chown sonar:sonar sonarqube-8.9.6.50800 -R
- chmod 777 sonarqube-8.9.6.50800 -R
- su – sonar

Step 2:

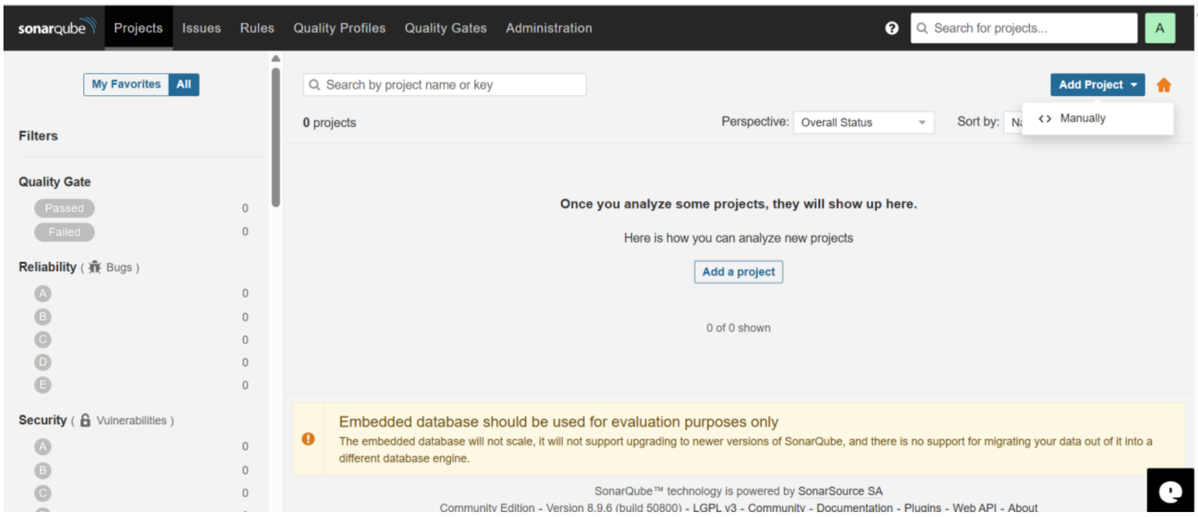
- Run the below commands manually
- cd /opt/sonarqube-8.9.6.50800/bin
- cd linux....
- Sh sonar.sh start

Step 3:

- Connect to the browser using sonarpublicip:9000 example: 3.80.198.164:9000
- Login as username: admin and Password: admin

Step 4:

- Create a project in SonarQube



Once you analyze some projects, they will show up here.

Here is how you can analyze new projects

0 of 0 shown

Embedded database should be used for evaluation purposes only

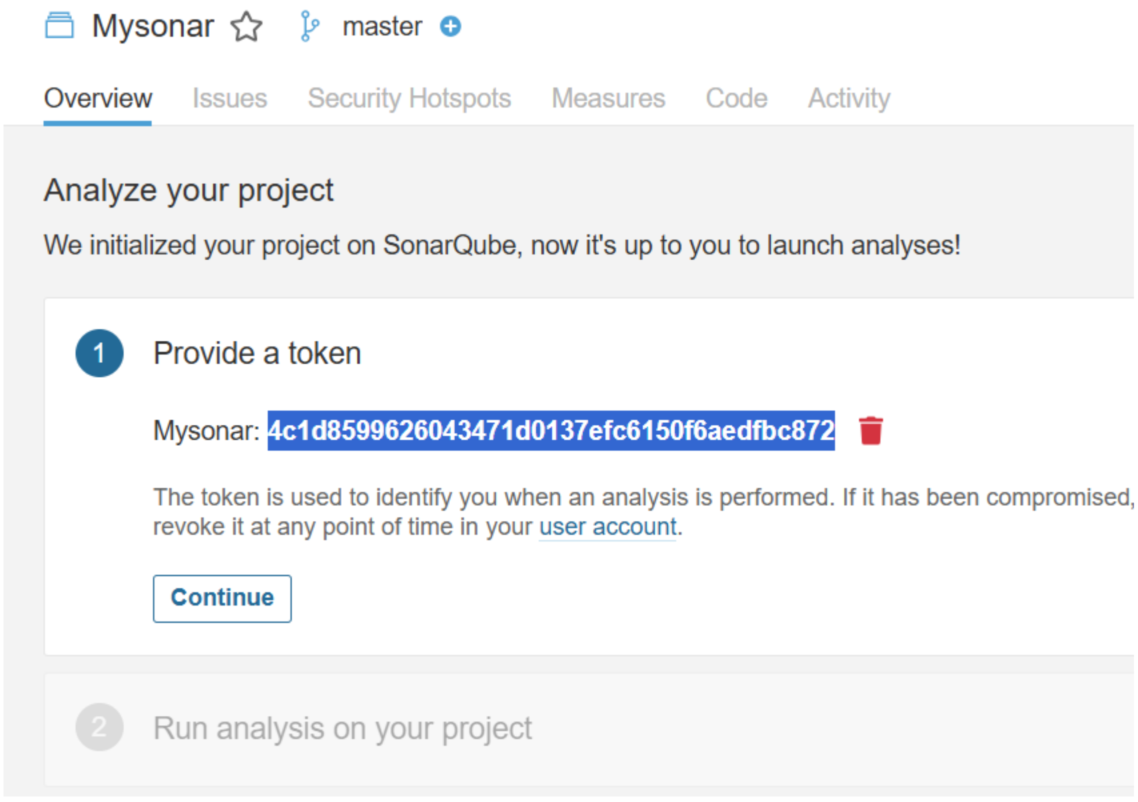
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA

Community Edition - Version 8.9.6 (build 50800) - LGPL v3 - Community - Documentation - Plugins - Web API - About

Step 5:

- Generate the Token
- Name and click generate
- Continue



Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Mysonar: 4c1d8599626043471d0137efc6150f6aedfbc872

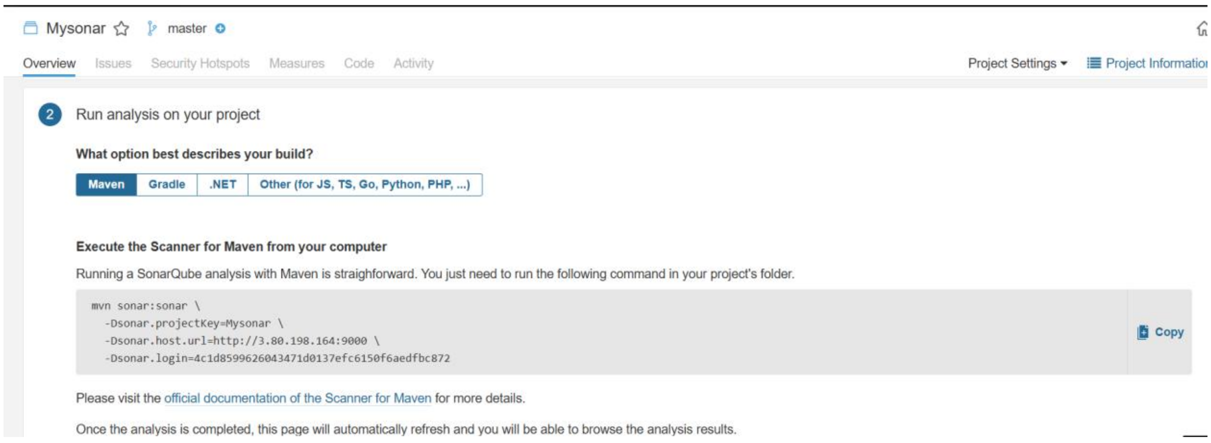
The token is used to identify you when an analysis is performed. If it has been compromised, revoke it at any point of time in your user account.

2 Run analysis on your project

Step 7:

- Select maven

- Copy the commands
- Provide these commands in pipeline script stage as “COA”



2 Run analysis on your project

What option best describes your build?

Execute the Scanner for Maven from your computer

Running a SonarQube analysis with Maven is straighforward. You just need to run the following command in your project's folder.

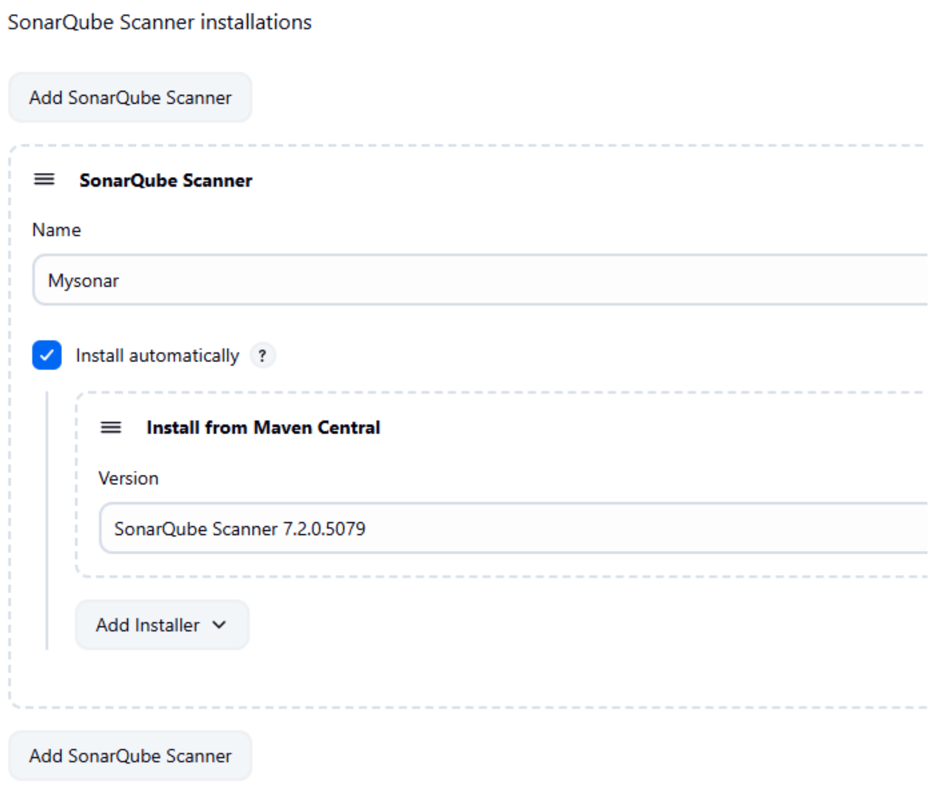
```
mvn sonar:sonar \
  -Dsonar.projectKey=Mysonar \
  -Dsonar.host.url=http://3.80.198.164:9000 \
  -Dsonar.login=4c1d8599626043471d0137efc6150f6aedfbc872
```

Please visit the official documentation of the Scanner for Maven for more details.

Once the analysis is completed, this page will automatically refresh and you will be able to browse the analysis results.

Step 8:

- Manage Jenkins → Tools
- Add SonarQube Scanner Installations



SonarQube Scanner installations

SonarQube Scanner

Name

Mysonar

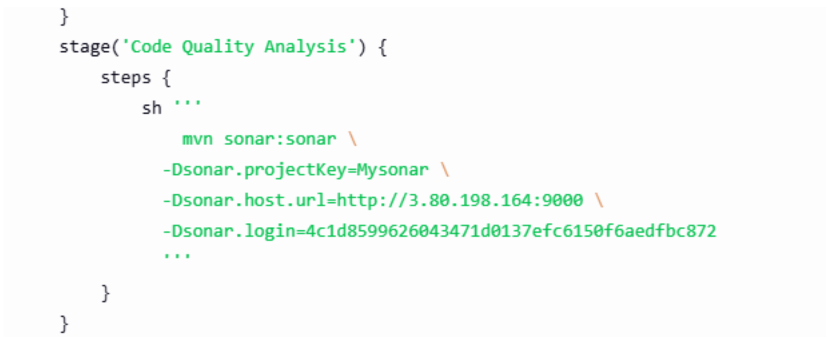
[✓] Install automatically

Install from Maven Central

Version

SonarQube Scanner 7.2.0.5079

Pipeline COA code:



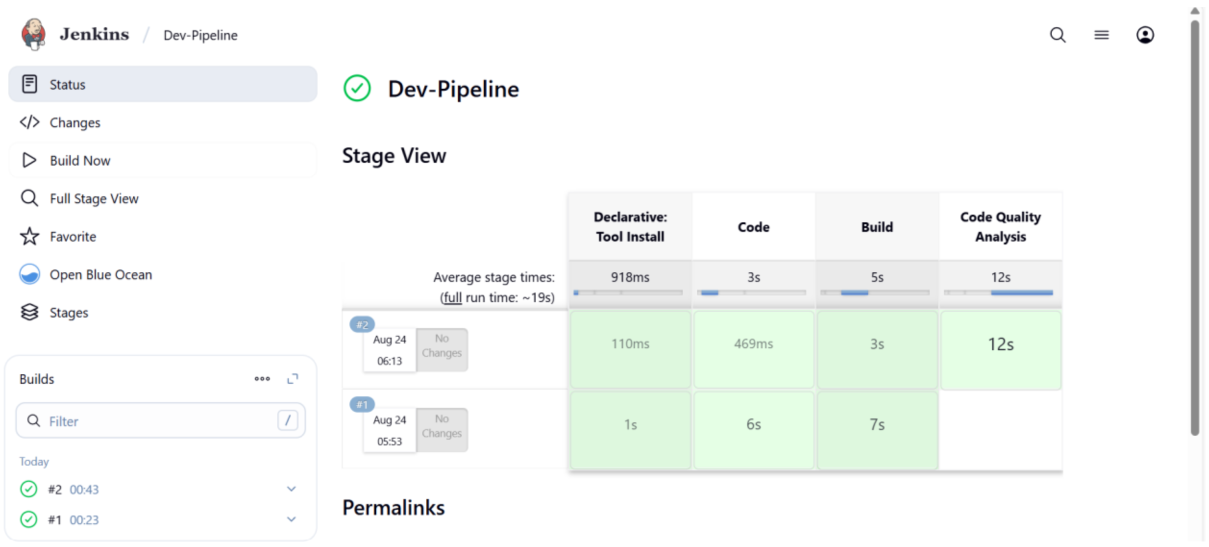
```
    }
    stage('Code Quality Analysis') {
        steps {
            sh '''
                mvn sonar:sonar \
              -Dsonar.projectKey=Mysonar \
              -Dsonar.host.url=http://3.80.198.164:9000 \
              -Dsonar.login=4c1d8599626043471d0137efc6150f6aedfbc872
              '''
        }
    }
```

Full Code: [CODE + BUILD + SONARQUBE]

```
pipeline {
  agent {
    node{
      label 'dev'
    }
  }
  environment {
    SCANNER = 'Mysonar'
  }
  tools {
    maven 'Mymaven'
  }

  stages {
    stage('Code') {
      steps {
        git branch: 'develop-1.0', url: 'https://github.com/devops0014/demorepo.git'

      }
    }
    stage('Build') {
      steps {
        sh 'mvn clean package'
      }
    }
    stage('COA') {
      steps {
        sh '''
         COMMANDS WHICH ARE SHOWN IN THE SONARQUBE REPO
         '''
      }
    }
  }
}
```

Step 9:

- Run the dev-pipeline
- Now the pipeline is completed stages [Code + Build + Code Quality Analysis]

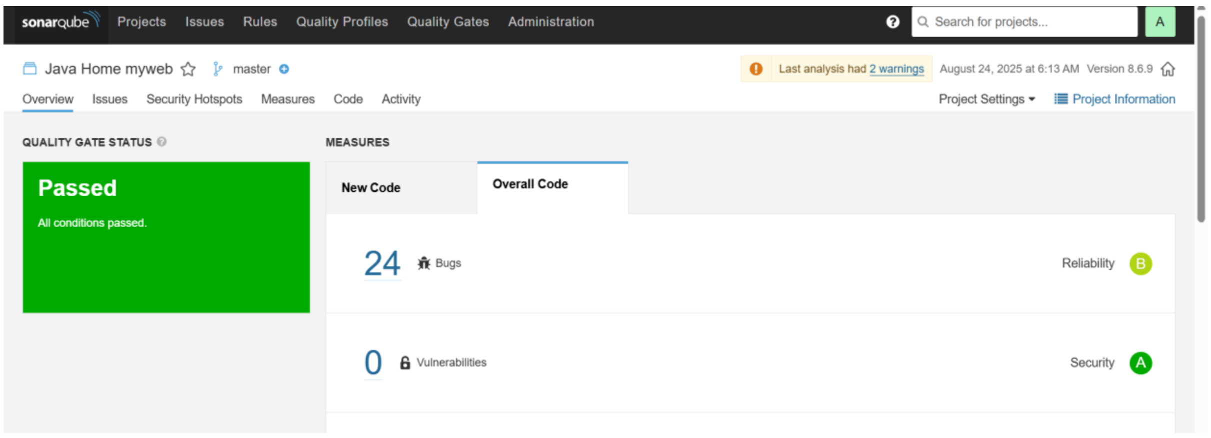


Dev-Pipeline

Stage View

	Declarative: Tool Install	Code	Build	Code Quality Analysis
Average stage times: (full run time: ~19s)	918ms	3s	5s	12s
#2 Aug 24 06:13 No Changes	110ms	469ms	3s	12s
#1 Aug 24 05:53 No Changes	1s	6s	7s	

SONARQUBE Results:



QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

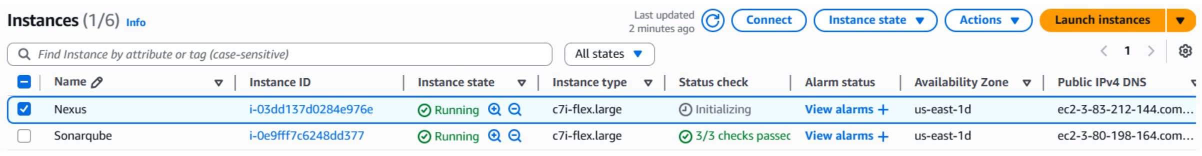
24 Bugs — Reliability B

0 Vulnerabilities — Security A

Launch Nexus server:

Step 1:

- Take the same config as SonarQube Server
- Nexus will store the artifacts


Instances (1/6) Info
Last updated 2 minutes ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
✓	Nexus	i-03dd137d0284e976e	Running	c7i-flex.large	Initializing	View alarms +	us-east-1d	ec2-3-83-212-144.com...
	Sonarqube	i-0e9fff7c6248dd377	Running	c7i-flex.large	3/3 checks passec	View alarms +	us-east-1d	ec2-3-80-198-164.com...

Step 2:

- Install nexus in the server using below commands

- sudo yum update -y
- sudo yum install wget -y
- sudo yum install java-17-amazon-corretto-jmods -y
- sudo mkdir /app && cd /app
- sudo wget https://download.sonatype.com/nexus/3/nexus-3.79.1-04-linux-x86_64.tar.gz
- sudo tar -xvf nexus-3.79.1-04-linux-x86_64.tar.gz
- sudo mv nexus-3.79.1-04 nexus
- sudo adduser nexus
- sudo chown -R nexus:nexus /app/nexus
- sudo chown -R nexus:nexus /app/sonatype*
- sudo sed -i '27 run_as_user="nexus"' /app/nexus/bin/nexus

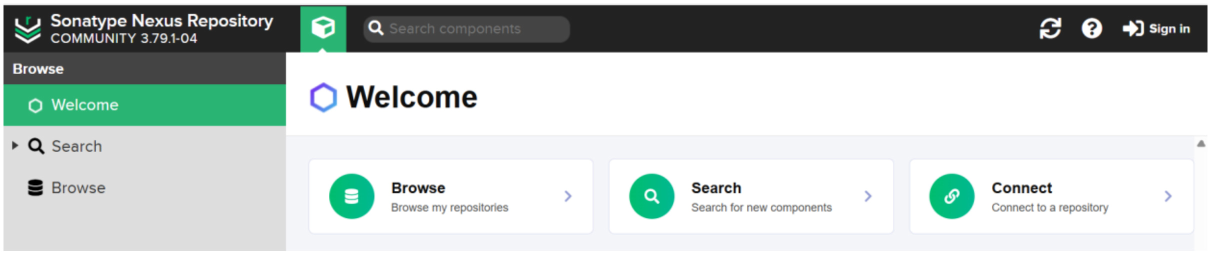
- sudo tee /etc/systemd/system/nexus.service > /dev/null << EOL
- [Unit]
- Description=nexus service
- After=network.target

- [Service]
- Type=forking
- LimitNOFILE=65536
- User=nexus
- Group=nexus
- ExecStart=/app/nexus/bin/nexus start
- ExecStop=/app/nexus/bin/nexus stop
- User=nexus
- Restart=on-abort

- [Install]
- WantedBy=multi-user.target
- EOL
- sudo chkconfig nexus on
- sudo systemctl start nexus
- sudo systemctl enable nexus
- sudo systemctl status nexus

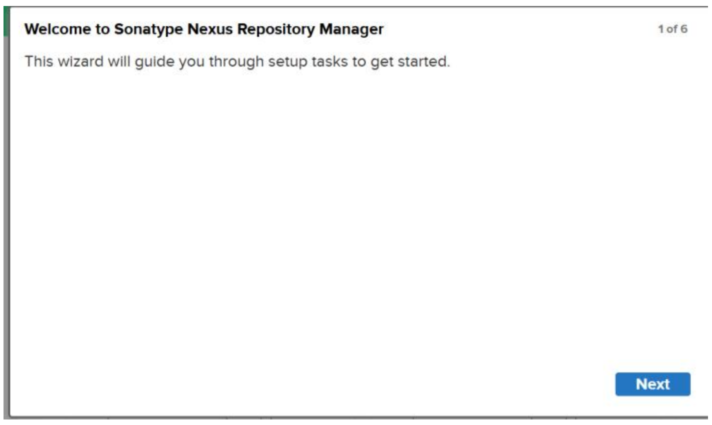
Step 3:

- Connect to the Nexus server using nexus publicip:8081



Step 4: Nexus credentials:

- Username: admin
- Password: cat **/app/sonatype-work/nexus3/admin.password**

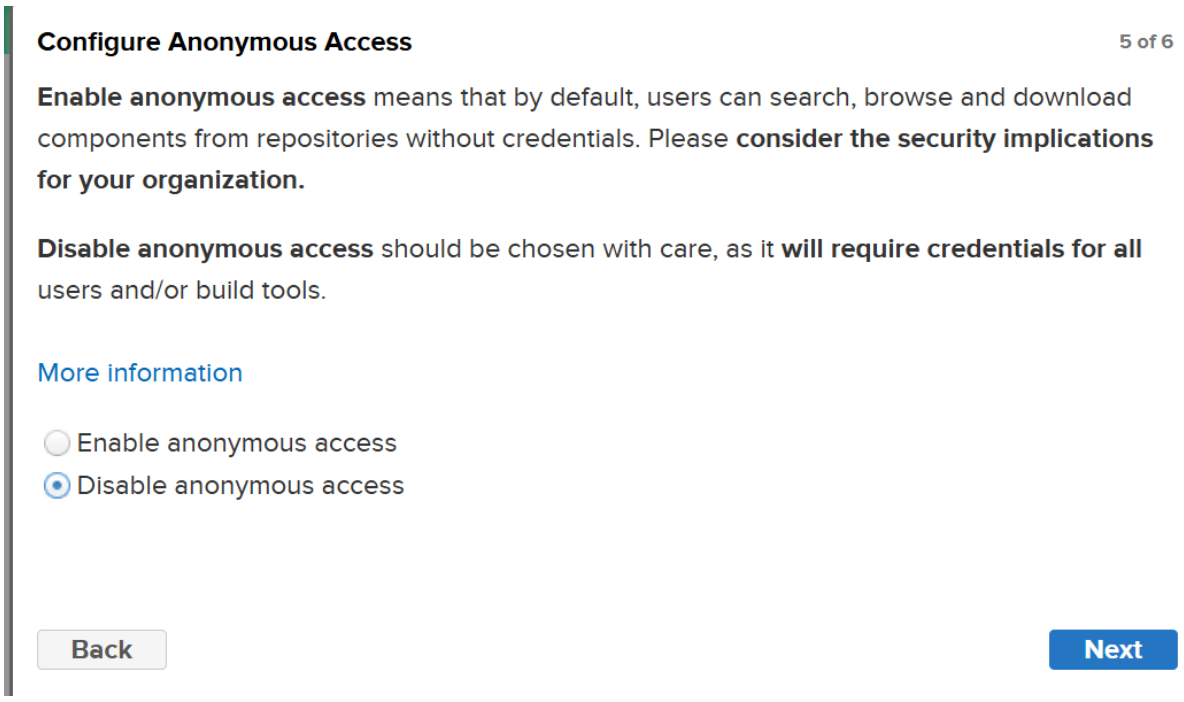

Welcome to Sonatype Nexus Repository Manager

1 of 6

This wizard will guide you through setup tasks to get started.

Step 5:

- Disable Anonymous access
- Save and finish


Configure Anonymous Access

5 of 6

Enable anonymous access means that by default, users can search, browse and download components from repositories without credentials. Please **consider the security implications for your organization.**

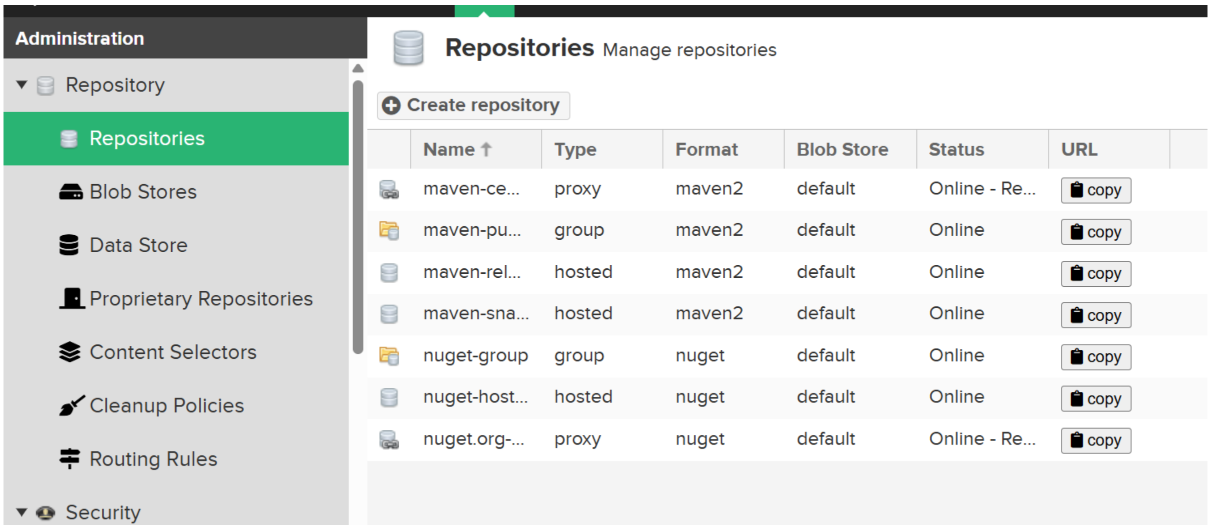
Disable anonymous access should be chosen with care, as it **will require credentials for all** users and/or build tools.

More information

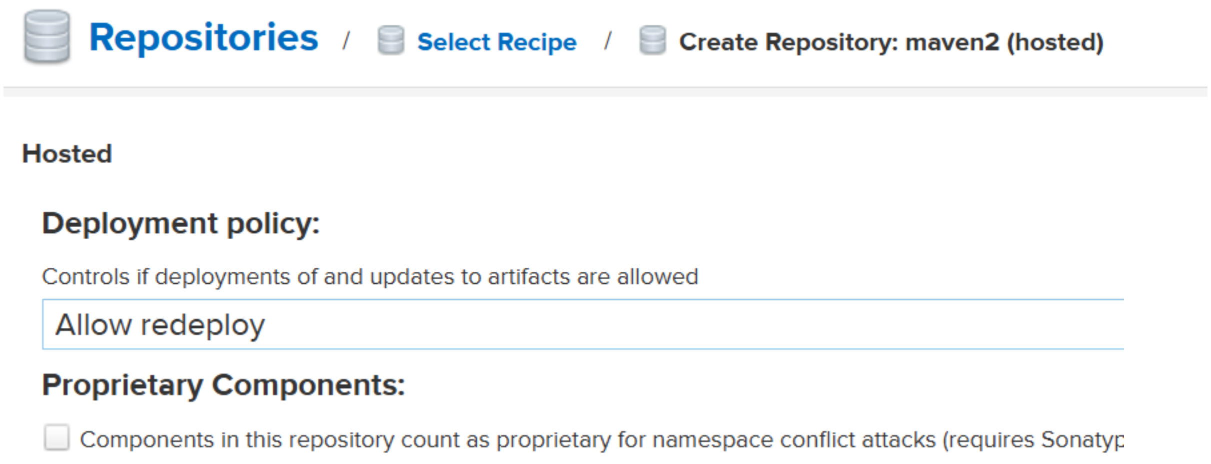
Enable anonymous access [✓ Disable anonymous access]

Step 6:

- Click on settings icon and go to Repositories
- Create a new repo as 'Myrepo'
- Select maven2 (hosted) option
- Allow redeploy



Name	Type	Format	Blob Store	Status	URL
maven-ce...	proxy	maven2	default	Online - Re...	copy
maven-pu...	group	maven2	default	Online	copy
maven-rel...	hosted	maven2	default	Online	copy
maven-sna...	hosted	maven2	default	Online	copy
nuget-group	group	nuget	default	Online	copy
nuget-host...	hosted	nuget	default	Online	copy
nuget.org-...	proxy	nuget	default	Online - Re...	copy


Repositories / Select Recipe / Create Repository: maven2 (hosted)

Hosted

Deployment policy:

Controls if deployments of and updates to artifacts are allowed

Allow redeploy

Proprietary Components:

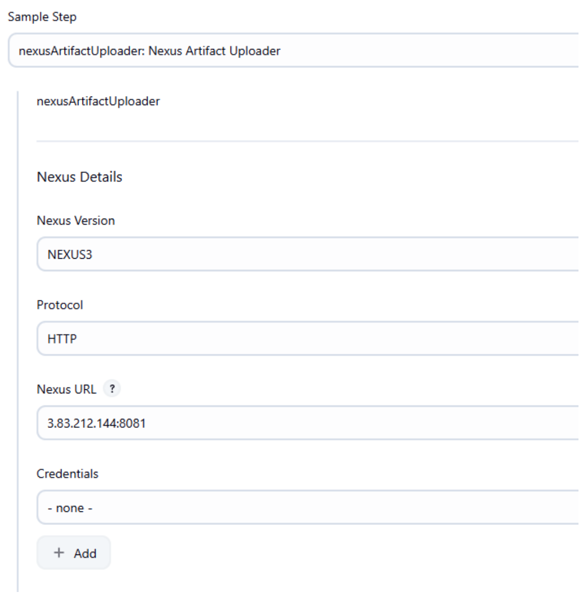
☐ Components in this repository count as proprietary for namespace conflict attacks (requires Sonatyp

Writing Nexus script using pipeline syntax:

Step 1:

- Select nexus artifact uploader
- Fill the required details as shown in below screenshots

Screenshot: 1



Sample Step

nexusArtifactUploader: Nexus Artifact Uploader

nexusArtifactUploader

Nexus Details

Nexus Version

NEXUS3

Protocol

HTTP

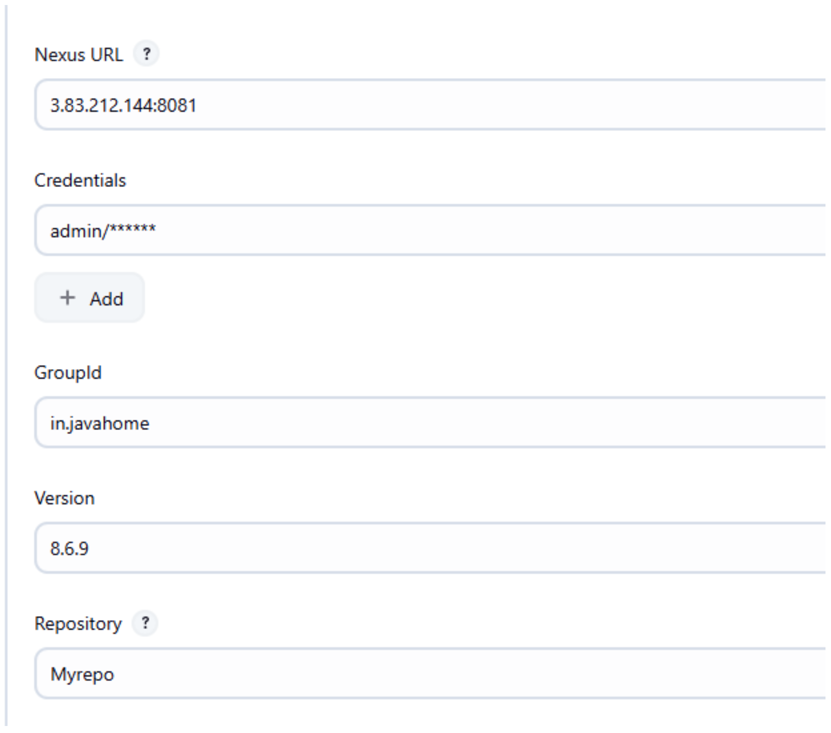
Nexus URL ?

3.83.212.144:8081

Credentials

- none -

Screenshot 2:



Nexus URL ?

3.83.212.144:8081

Credentials

admin/******

GroupId

in.javahome

Version

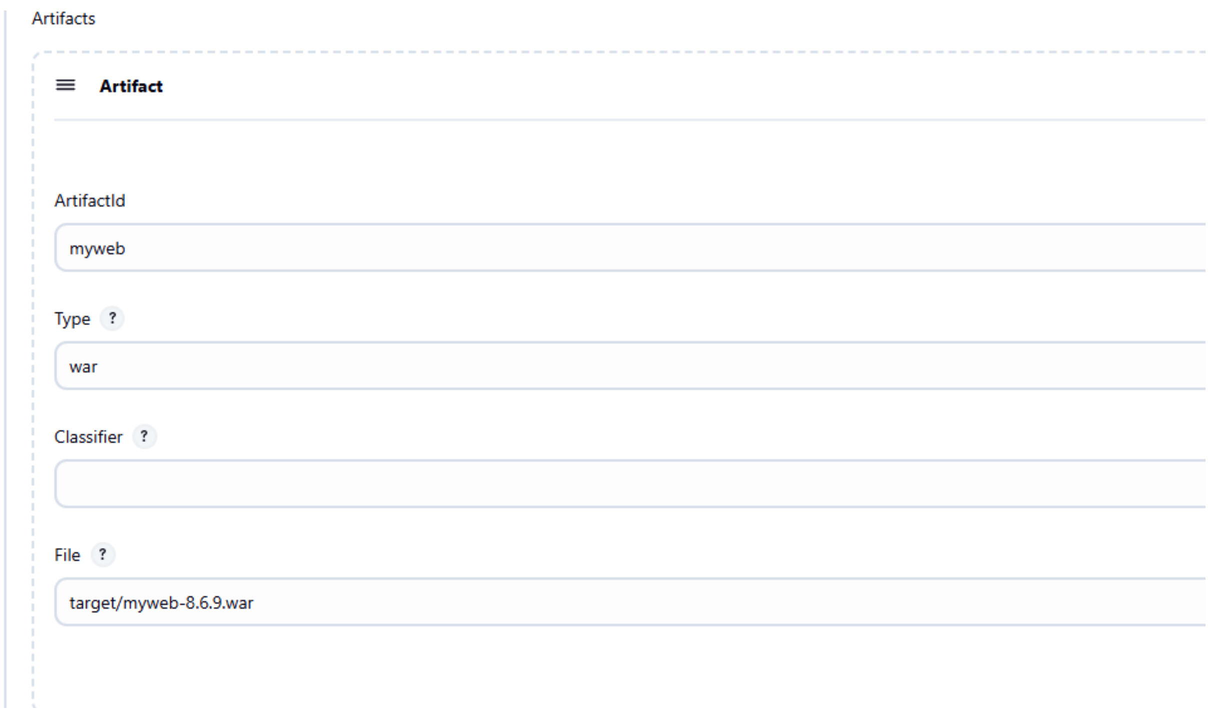
8.6.9

Repository ?

Myrepo

Screenshot 3:

- Below details are able to find at pom.xml file in github repo


Artifacts

Artifact

ArtifactId

myweb

Type ?

war

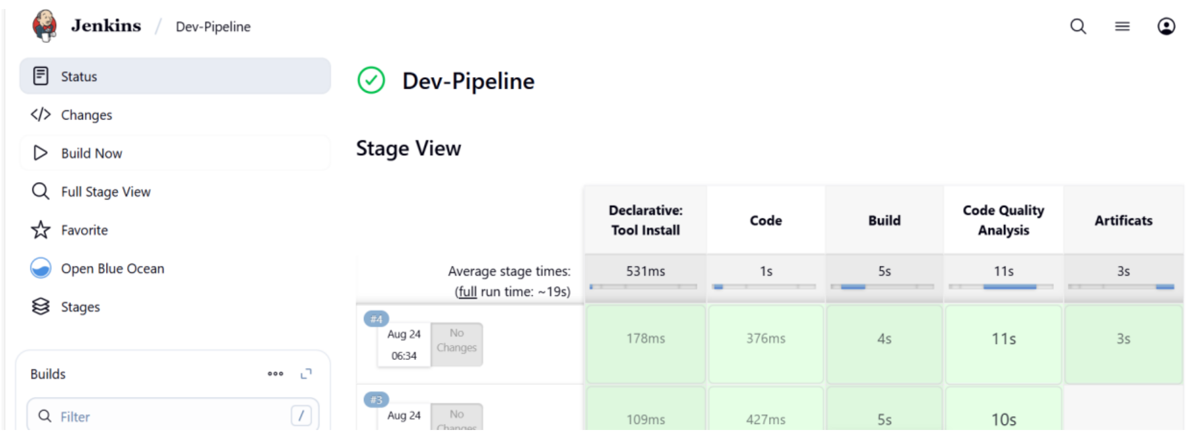
Classifier ?

File ?

target/myweb-8.6.9.war

Step 2:

- Run the pipeline job
- Now you have completed stages [Code + Build + COA + Artificats]


Jenkins / Dev-Pipeline

Status
Changes
Build Now
Full Stage View
Favorite
Open Blue Ocean
Stages

Builds
Filter

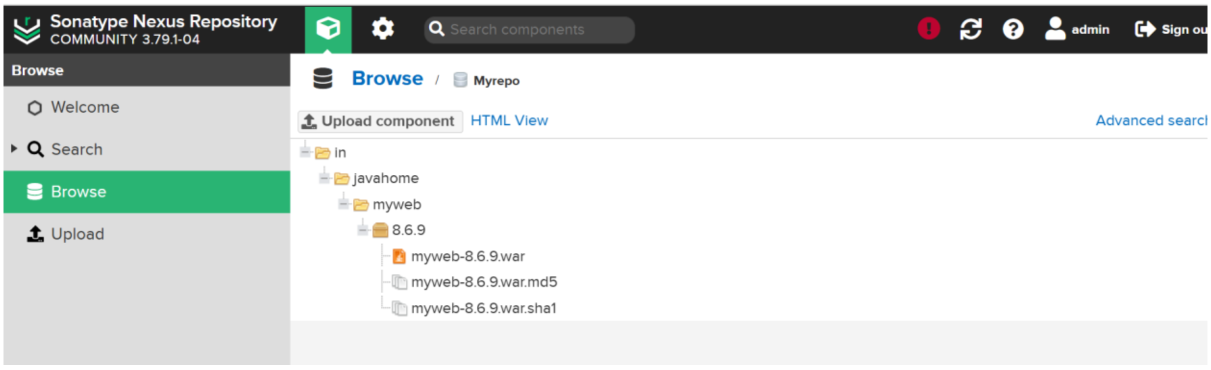
Dev-Pipeline

Stage View

	Declarative: Tool Install	Code	Build	Code Quality Analysis	Artificats
Average stage times: (full run time: ~19s)	531ms	1s	5s	11s	3s
#4 Aug 24 06:34 No Changes	178ms	376ms	4s	11s	3s
#3 Aug 24 No Changes	109ms	427ms	5s	10s	

Step 3:

- Open the Nexus repo and go to browser
- Open the Myrepo
- Check the artifacts



```
in
  javahome
    myweb
      8.6.9
        myweb-8.6.9.war
        myweb-8.6.9.war.md5
        myweb-8.6.9.war.sha1
```

Pipeline code upto Artifacts:

```
pipeline {
  agent {
    node{
      label 'dev'
    }
  }
  environment {
    SCANNER = 'Mysonar'
  }
  tools {
    maven 'Mymaven'
  }

  stages {
    stage('Code') {
      steps {
        git branch: 'develop-1.0', url: 'https://github.com/devops0014/demorepo.git'
      }
    }
    stage('Build') {
      steps {
        sh 'mvn clean package'
      }
    }
    stage('COA') {
      steps {
        sh '''
         COMMANDS WHICH ARE SHOWN IN THE SONARQUBE REPO
         '''
      }
    }
   stage('Artifacts') {
      steps {
        nexusArtifactUploader artifacts: [[artifactId: 'myweb', classifier: '', file: 'target/myweb-8.6.9.war', type: 'war']], credentialsId: 'Nexus', groupId: 'in.javahome', nexusUrl: '3.83.212.144:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'Myrepo', version: '8.6.9'
      }
    }

  }
}
```

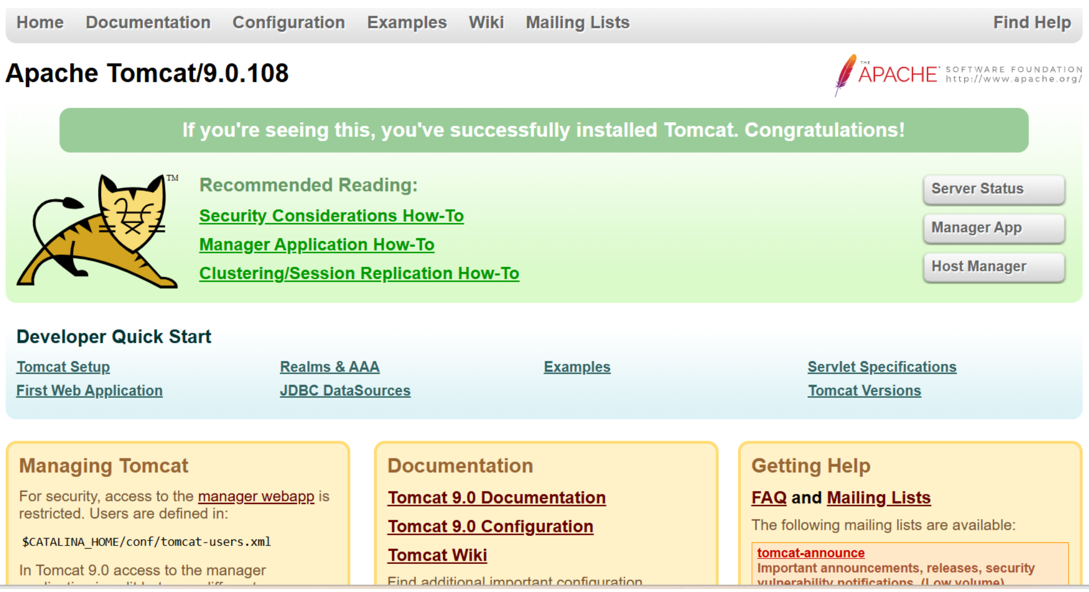
Launching Tomcat:

Step 1:

- Installing Tomcat app server in Developer server ec2- instance
- Using the below commands:
- yum install java-17-amazon-corretto -y
- wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.tar.gz
- tar -zxvf apache-tomcat-9.0.108.tar.gz
- sed -i '56 a\<role rolename="manager-gui"/>' apache-tomcat-9.0.108/conf/tomcat-users.xml
- sed -i '57 a\<role rolename="manager-script"/>' apache-tomcat-9.0.108/conf/tomcat-users.xml
- sed -i '58 a\<user username="tomcat" password="admin@123" roles="manager-gui, manager-script"/>' apache-tomcat-9.0.108/conf/tomcat-users.xml
- sed -i '59 a\</tomcat-users>' apache-tomcat-9.0.108/conf/tomcat-users.xml
- sed -i '56d' apache-tomcat-9.0.108/conf/tomcat-users.xml
- sed -i '21d' apache-tomcat-9.0.108/webapps/manager/META-INF/context.xml
- sed -i '22d' apache-tomcat-9.0.108/webapps/manager/META-INF/context.xml
- sh apache-tomcat-9.0.108/bin/startup.sh

Step 2:

- connect to Tomcat UI using publicip:8080
- go to manger app



Apache Tomcat/9.0.108

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To

Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 9.0 access to the manager

Documentation
Tomcat 9.0 Documentation
Tomcat 9.0 Configuration
Tomcat Wiki
Find additional important configuration

Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume)

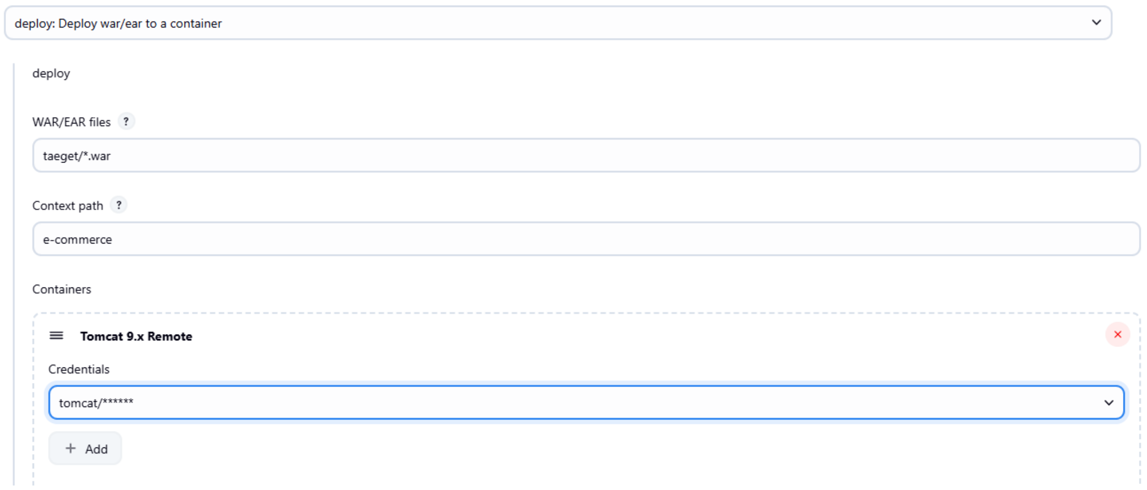
Step 3:

- Username: tomcat
- Password: admin@123
- The above credentials are mentioned in the tomcat installation script

Writing script for tomcat deployment:

Step 1:

- Select deploy war to a container
- Provide config details as shown below



deploy: Deploy war/ear to a container

deploy

WAR/EAR files ?

taeget/*.war

Context path ?

e-commerce

Containers

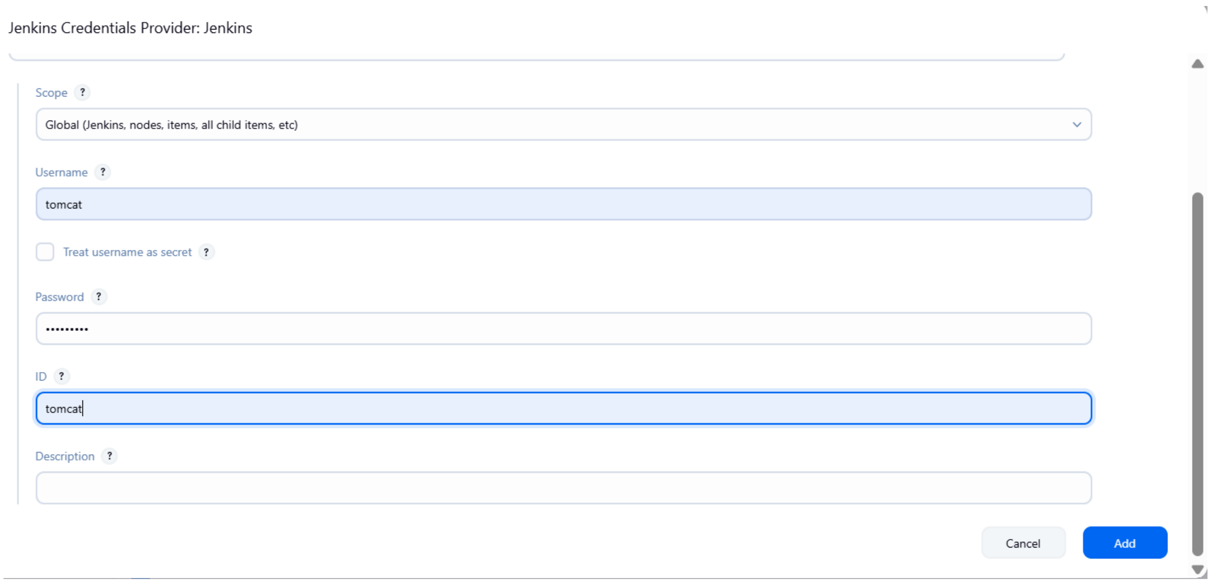
Tomcat 9.x Remote

Credentials

tomcat/******

Step 2:

- Provide tomcat credentials



Jenkins Credentials Provider: Jenkins

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

tomcat

☐ Treat username as secret ?

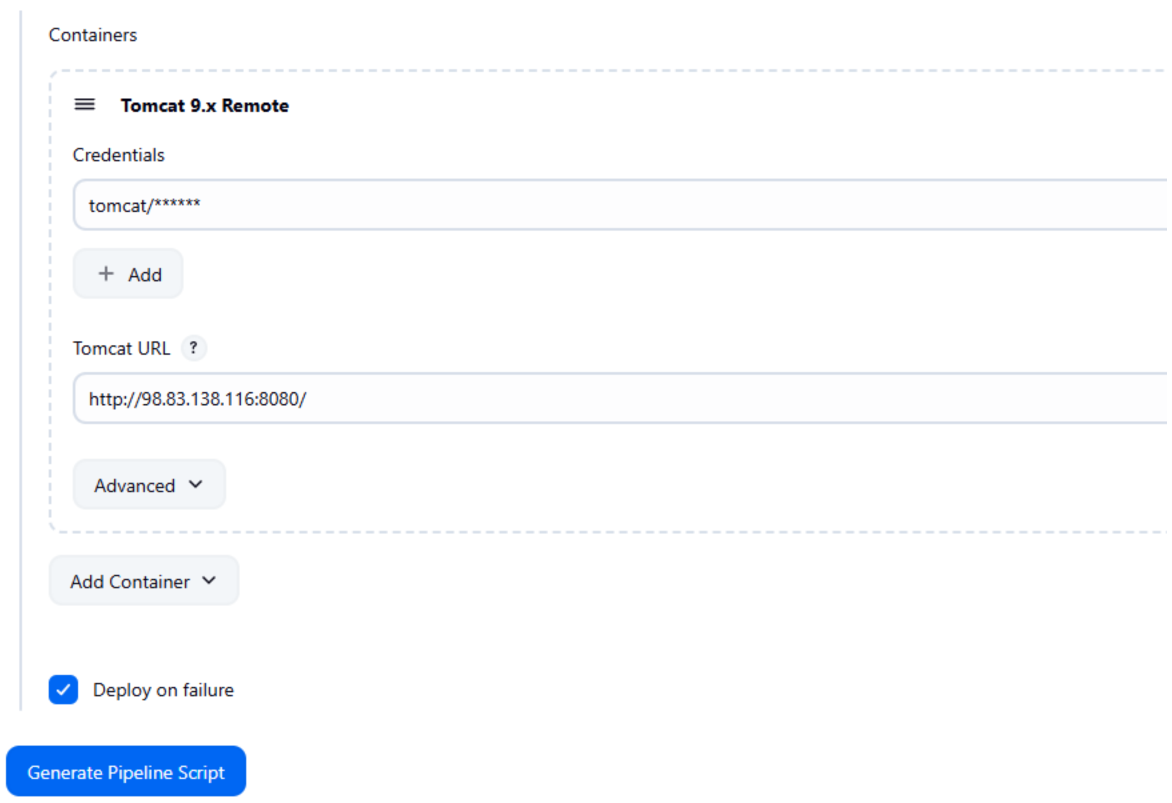
Password ?

•••••••••

ID ?

tomcat

Description ?


Containers

Tomcat 9.x Remote

Credentials

tomcat/******

Tomcat URL

http://98.83.138.116:8080/

[✓] Deploy on failure

Optional Step:

- The code is for Manual Approval and Deploy:
- Yes, to deploy
- No to abort the pipeline

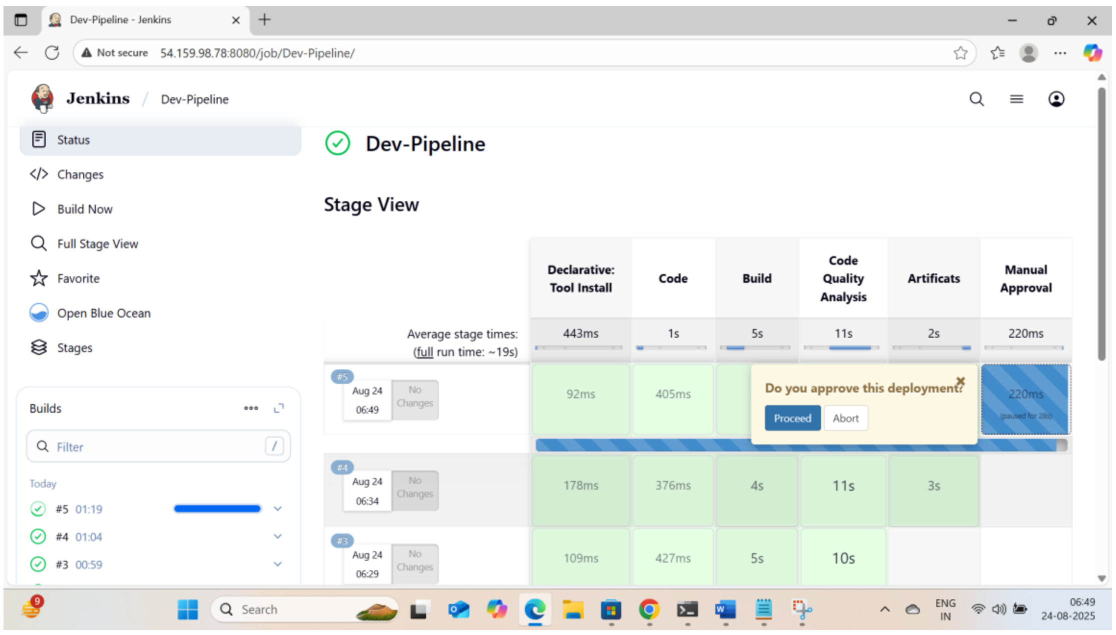
Full Pipeline Script:

```
pipeline {
  agent {
    node{
      label 'dev'
    }
  }
  environment {
    SCANNER = 'Mysonar'
  }
  tools {
    maven 'Mymaven'
  }

  stages {
    stage('Code') {
      steps {
        git branch: 'develop-1.0', url: 'https://github.com/devops0014/demorepo.git'


      }
    }
    stage('Build') {
      steps {
        sh 'mvn clean package'
      }
    }
    stage('COA') {
      steps {
        sh '''
         COMMANDS WHICH ARE SHOWN IN THE SONARQUBE REPO
         '''
      }
    }
   stage('Artificats') {
      steps {
        nexusArtifactUploader artifacts: [[artifactId: 'myweb', classifier: '', file: 'target/myweb-8.6.9.war', type: 'war']], credentialsId: 'Nexus', groupId: 'in.javahome', nexusUrl: '3.83.212.144:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'Myrepo', version: '8.6.9'
      }
    }
```

```
        stage('Manual Approval') {
          steps {
            timeout(time:10, unit: 'MINUTES'){
              input(message: 'Do you approve this deployment?', ok: 'Proceed')
            }
          }
        }
        stage('Deploy') {
          steps {
            deploy adapters: [tomcat9(alternativeDeploymentContext: '', credentialsId: 'tomcat', path: '', url: 'http://98.83.138.116:8080/')], contextPath: 'e-commerce', war: 'target/*.war'
          }
        }

      }
    }
```

OUTPUT:

- Here you can see the approval step in the pipeline stage



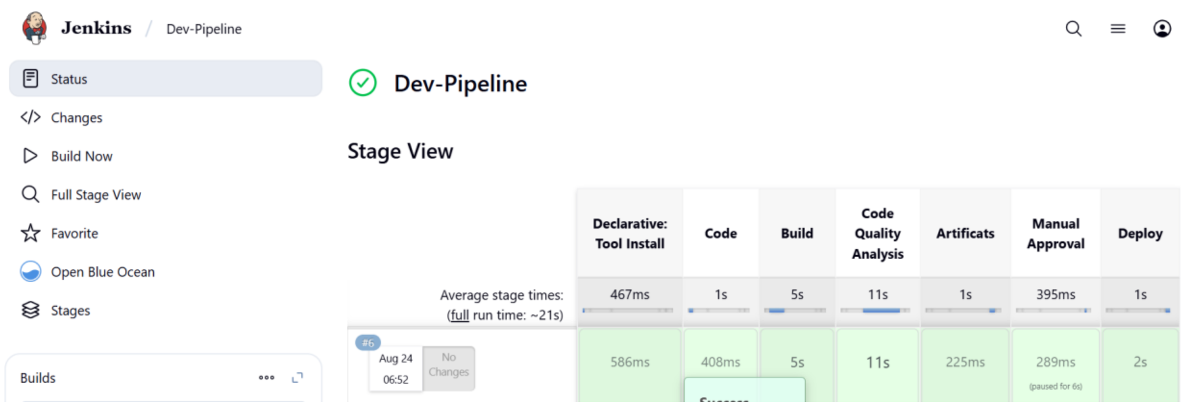
Dev-Pipeline

Stage View

	Declarative: Tool Install	Code	Build	Code Quality Analysis	Artificats	Manual Approval
Average stage times: (full run time: ~19s)	443ms	1s	5s	11s	2s	220ms
#5 Aug 24 06:49 No Changes	92ms	405ms				220ms (paused for 28s)
#4 Aug 24 06:34 No Changes	178ms	376ms	4s	11s	3s	
#3 Aug 24 06:29 No Changes	109ms	427ms	5s	10s		

Do you approve this deployment?

- Now all stages are completed [Code + Build + COA + Artifacts + Manual Approval + Deploy]



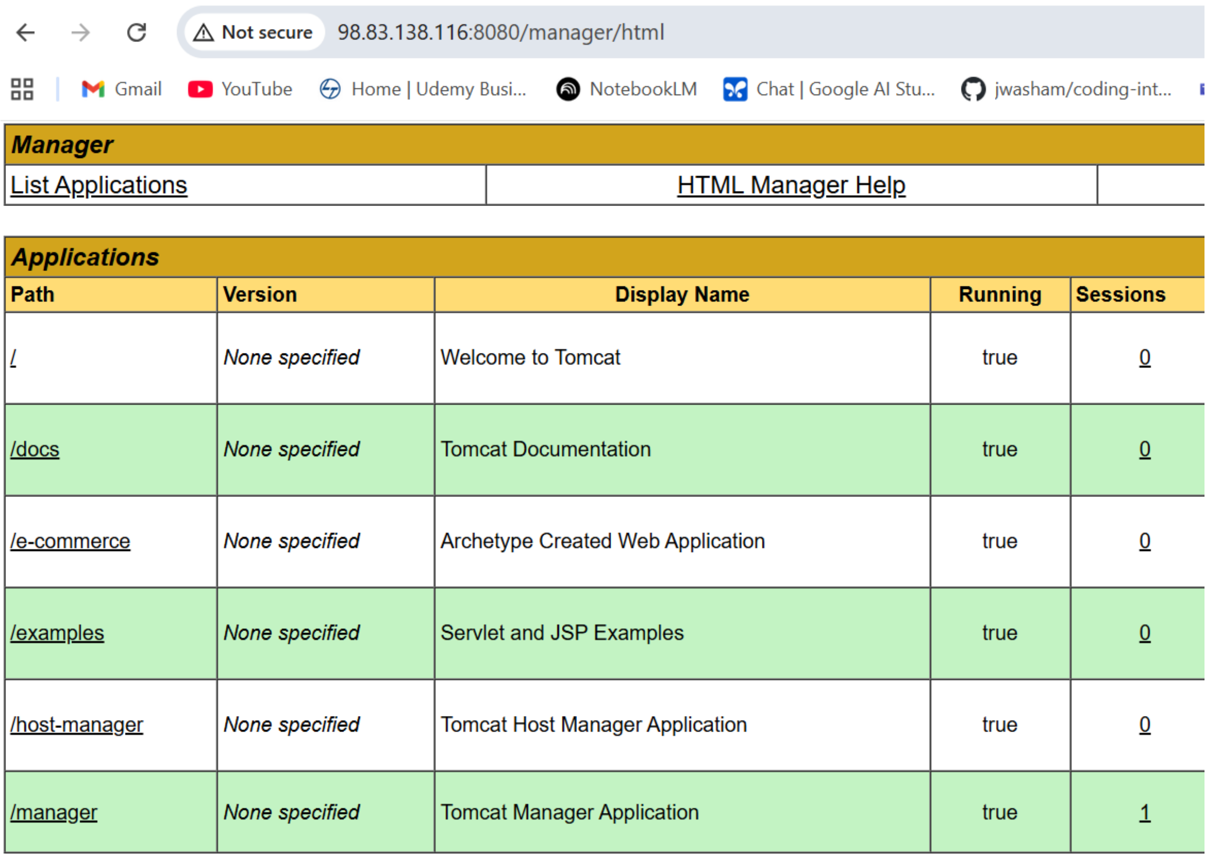
Jenkins / Dev-Pipeline

Dev-Pipeline

Stage View

	Declarative: Tool Install	Code	Build	Code Quality Analysis	Artificats	Manual Approval	Deploy
Average stage times: (full run time: ~21s)	467ms	1s	5s	11s	1s	395ms	1s
#6 Aug 24 06:52 No Changes	586ms	408ms	5s	11s	225ms	289ms (paused for 6s)	2s

- Checking the application in the tomcat server
- Click on e-commerce as you mentioned in the tomcat context path



Not secure 98.83.138.116:8080/manager/html

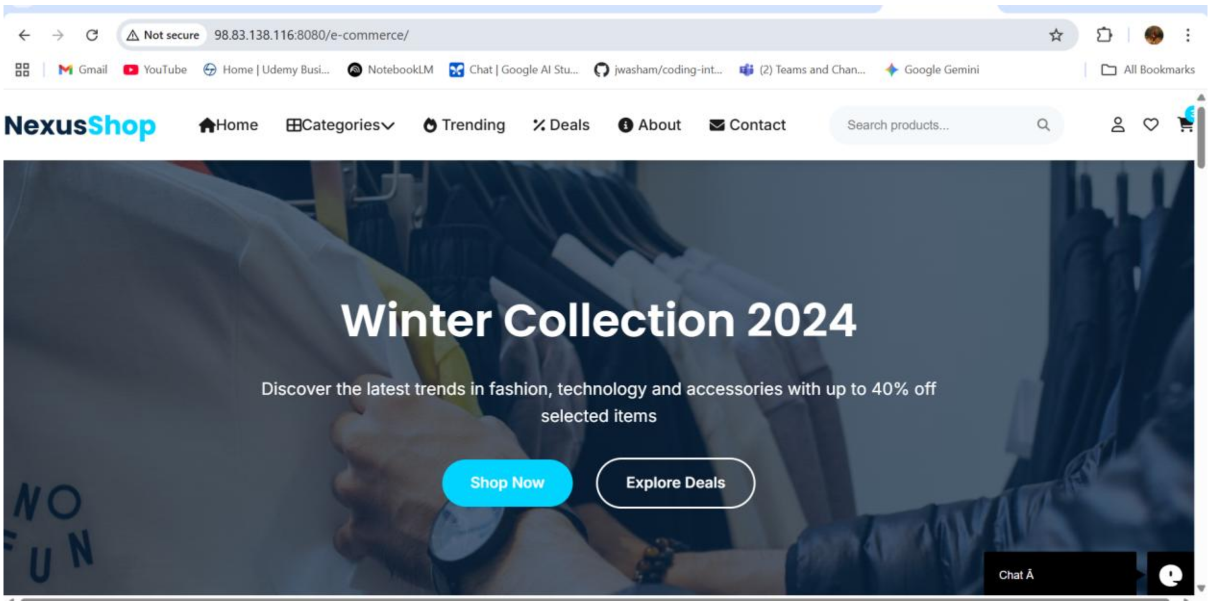
Manager

List Applications | HTML Manager Help

Applications

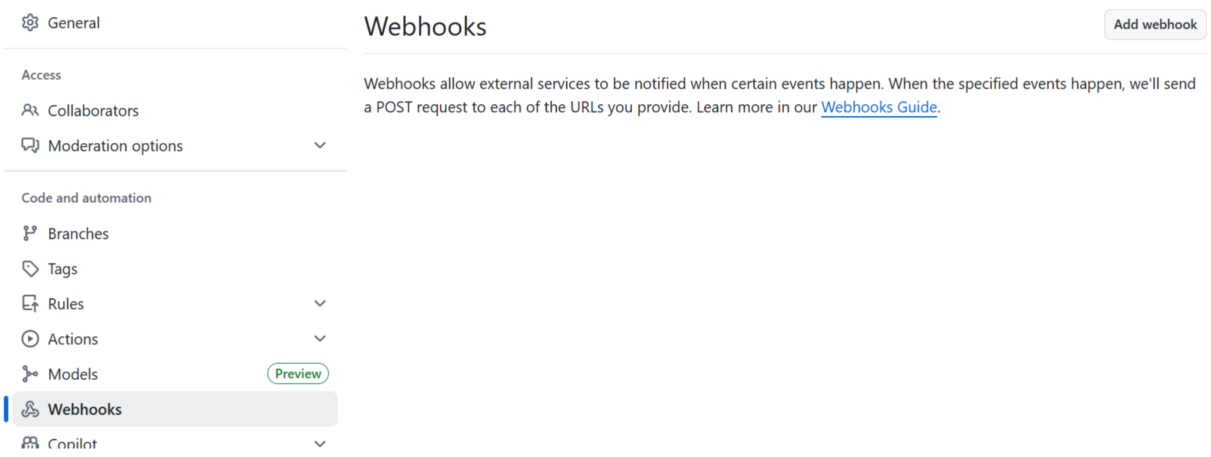
Path	Version	Display Name	Running	Sessions
/	*None specified*	Welcome to Tomcat	true	0
/docs	*None specified*	Tomcat Documentation	true	0
/e-commerce	*None specified*	Archetype Created Web Application	true	0
/examples	*None specified*	Servlet and JSP Examples	true	0
/host-manager	*None specified*	Tomcat Host Manager Application	true	0
/manager	*None specified*	Tomcat Manager Application	true	1

- Application deployed successfully


Not secure 98.83.138.116:8080/e-commerce/
NexusShop
Home
Categories
Trending
Deals
About
Contact
Search products...
Winter Collection 2024
Discover the latest trends in fashion, technology and accessories with up to 40% off selected items
Shop Now
Explore Deals

Additional Steps:

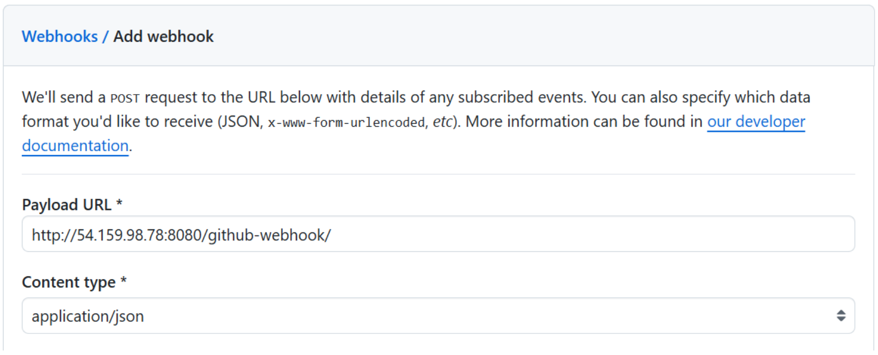
- Now we do Automation running pipeline using webhooks
- Github repo settings → webhooks
- Add webhooks



Webhooks

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.

- Provide the details of Jenkins server:8080/github-webhook/

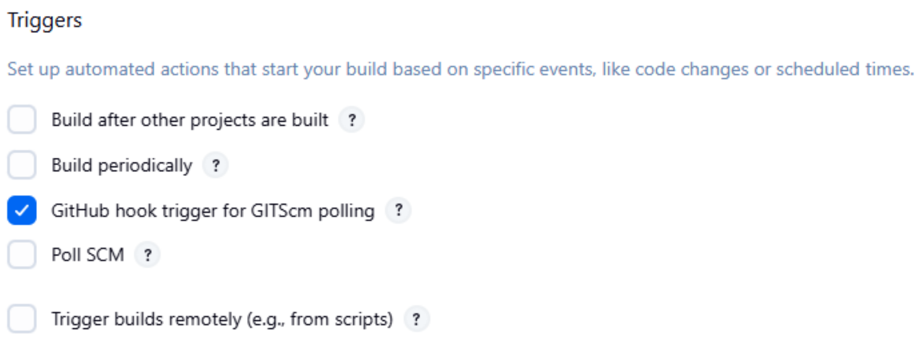

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *
http://54.159.98.78:8080/github-webhook/

Content type *
application/json

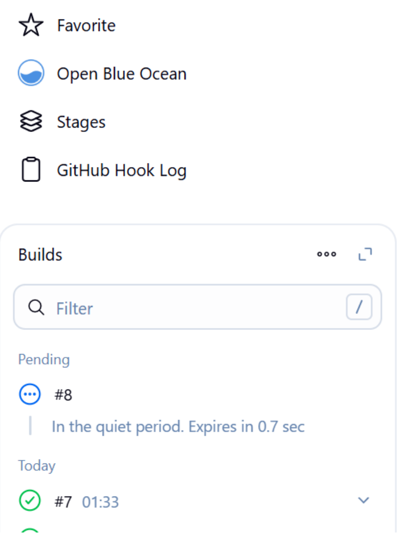
- Do some code level modifications in GitHub and commit it
- Goto git hub repo and make changes to index.jsp file and check the branch correctly
- In the pipeline job go to triggers and enable GitHub webhooks


Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [✓] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Trigger builds remotely (e.g., from scripts) ?

Automatically triggered pipeline



Favorite

Open Blue Ocean

Stages

GitHub Hook Log

Builds

Filter

Pending

#8

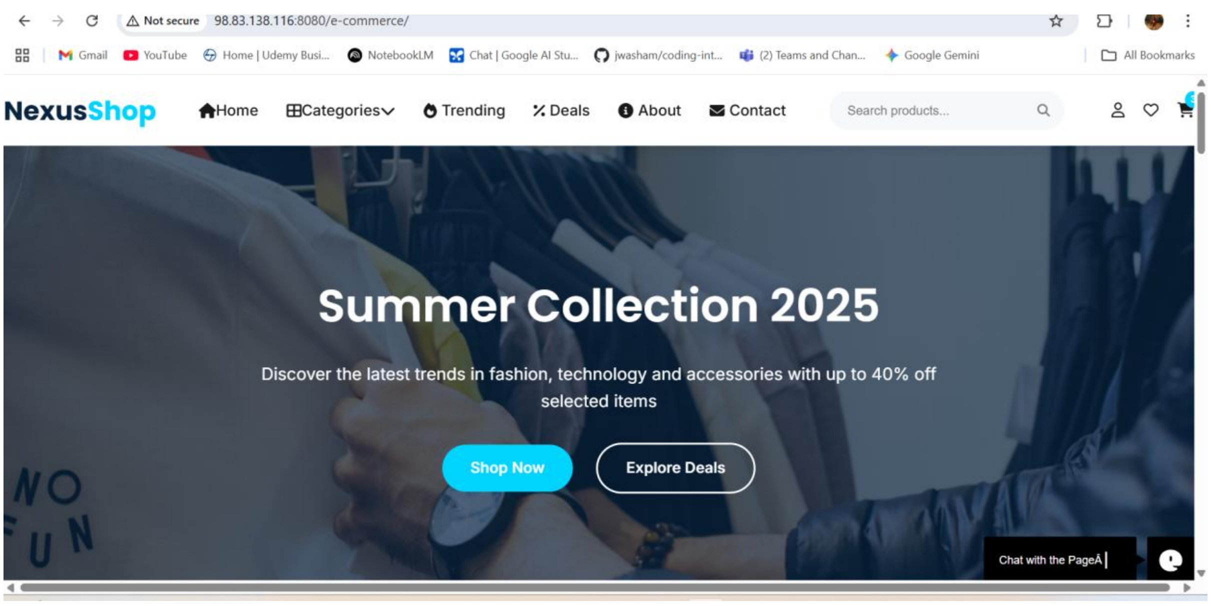
In the quiet period. Expires in 0.7 sec

Today

#7 01:33

Re-deploying the application for artifacts version:

- Now we deployed our app again with change of h1 to Summer collection 2025 from Winter collection



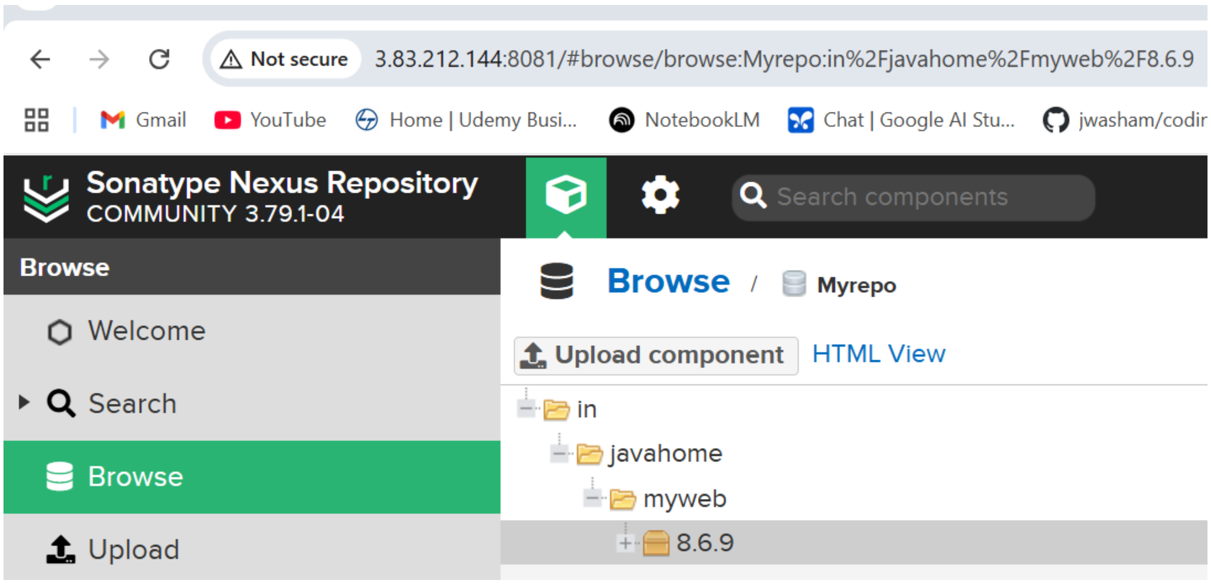
Not secure 98.83.138.116:8080/e-commerce/

Summer Collection 2025

Discover the latest trends in fashion, technology and accessories with up to 40% off selected items

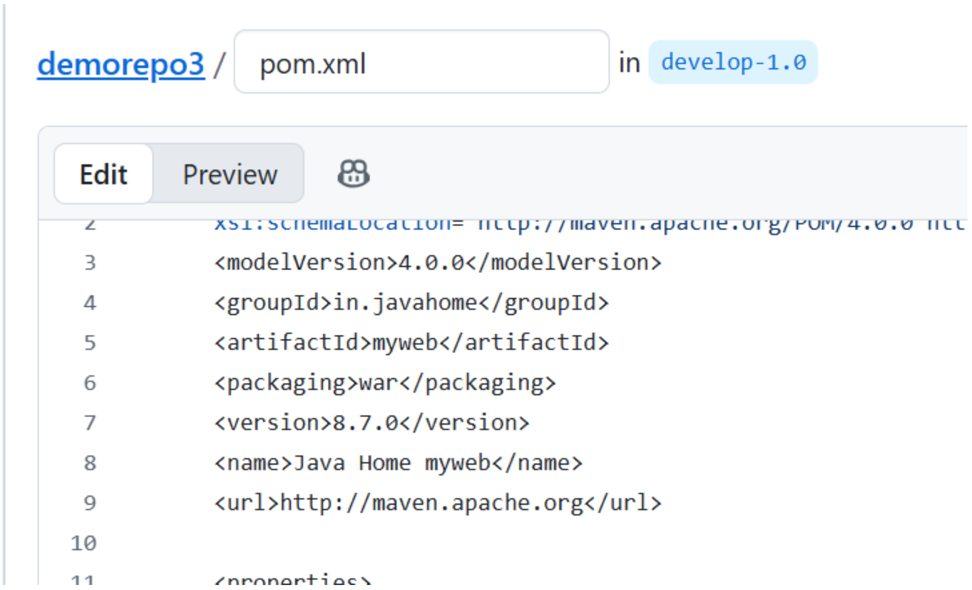
Error:

- In the Nexus repo we didn’t have new Artificats because we didn’t add so now, we will add and redeploy


3.83.212.144:8081/#browse/browse:Myrepo:in%2Fjavahome%2Fmyweb%2F8.6.9

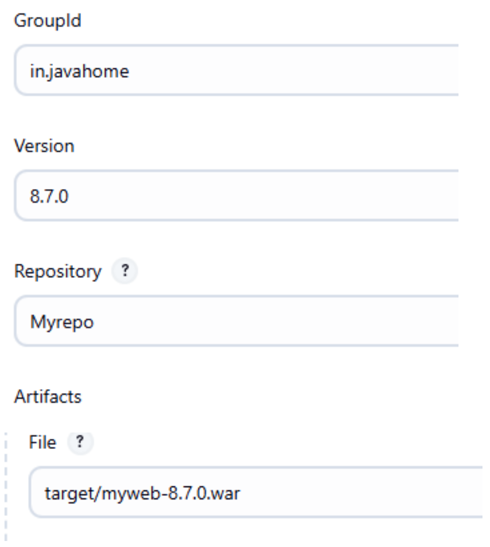
- Changing version in pom.xml file
- Change the groupid version and file version in the deploy stage code

Screenshot: 1



```
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 htt
        <modelVersion>4.0.0</modelVersion>
        <groupId>in.javahome</groupId>
        <artifactId>myweb</artifactId>
        <packaging>war</packaging>
        <version>8.7.0</version>
        <name>Java Home myweb</name>
        <url>http://maven.apache.org</url>

        <properties>
```

Screenshot 2:



GroupId

in.javahome

Version

8.7.0

Repository ?

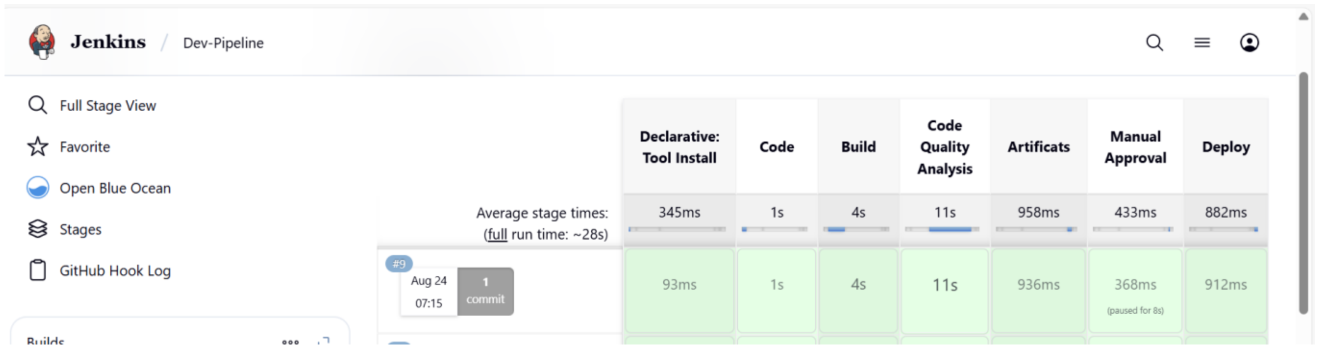
Myrepo

Artifacts

File ?

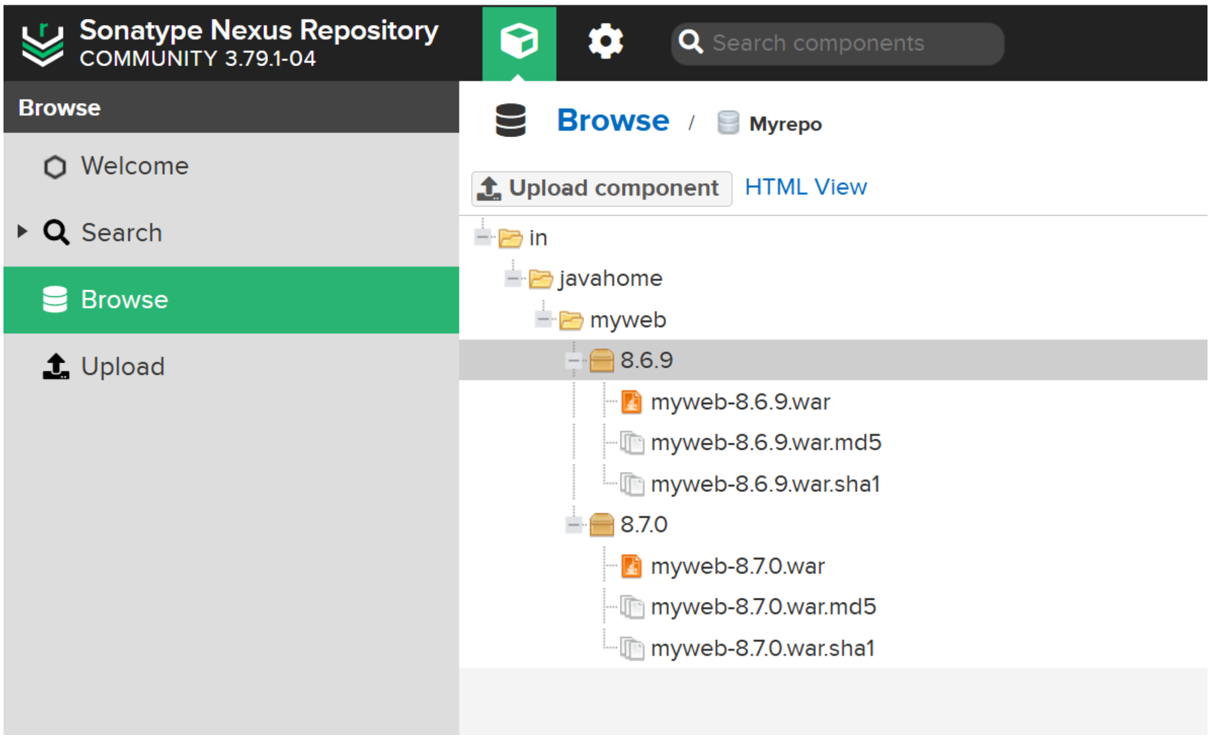
target/myweb-8.7.0.war

- Deployed again and success state



	Declarative: Tool Install	Code	Build	Code Quality Analysis	Artificats	Manual Approval	Deploy
Average stage times: (full run time: ~28s)	345ms	1s	4s	11s	958ms	433ms	882ms
#9 Aug 24 07:15 1 commit	93ms	1s	4s	11s	936ms	368ms (paused for 8s)	912ms

- Now we are able to see the new Artifact version 8.7.0

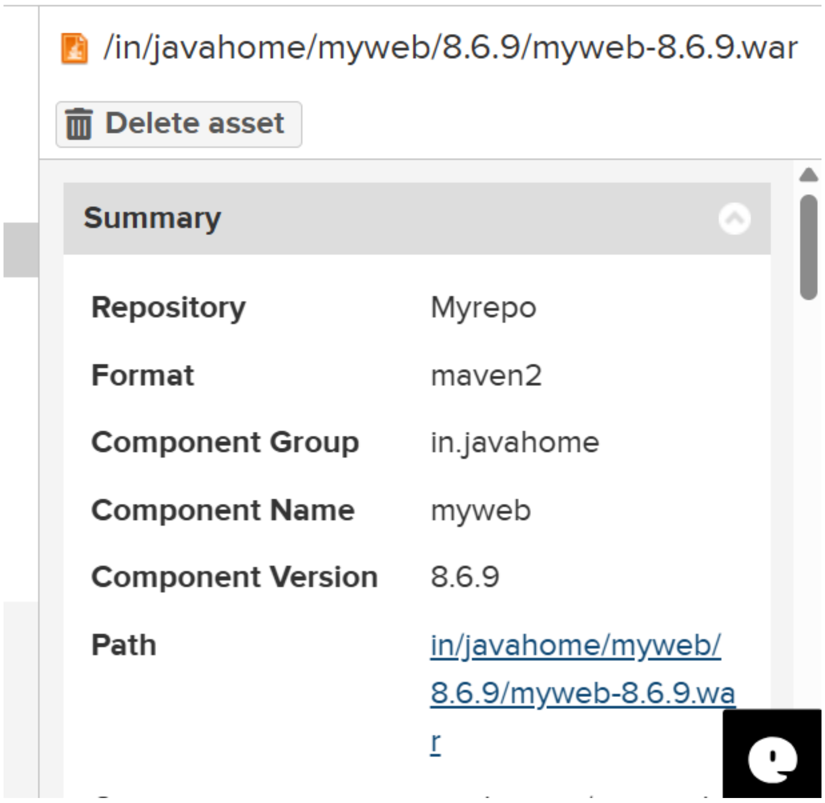


```
in
  javahome
    myweb
      8.6.9
        myweb-8.6.9.war
        myweb-8.6.9.war.md5
        myweb-8.6.9.war.sha1
      8.7.0
        myweb-8.7.0.war
        myweb-8.7.0.war.md5
        myweb-8.7.0.war.sha1
```

Rollback to previous version:

- Rollback to 8.6.9 version
- Open the nexus → Myrepo
- Select the 8.6.9 war file
- Click on path and download it
- Go to nexus server → Manger app
- Select the war file and deploy
- You had new application in the down with previous deployment

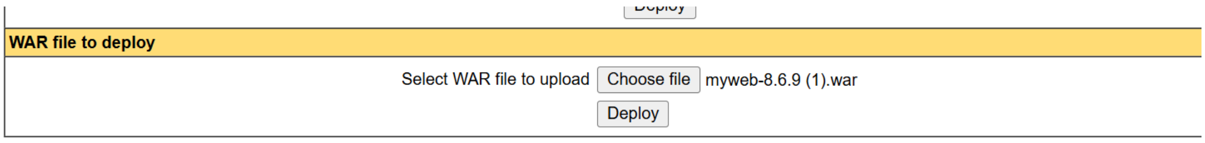
Screenshot: 1



/in/javahome/myweb/8.6.9/myweb-8.6.9.war

Repository	Myrepo
Format	maven2
Component Group	in.javahome
Component Name	myweb
Component Version	8.6.9
Path	in/javahome/myweb/8.6.9/myweb-8.6.9.war

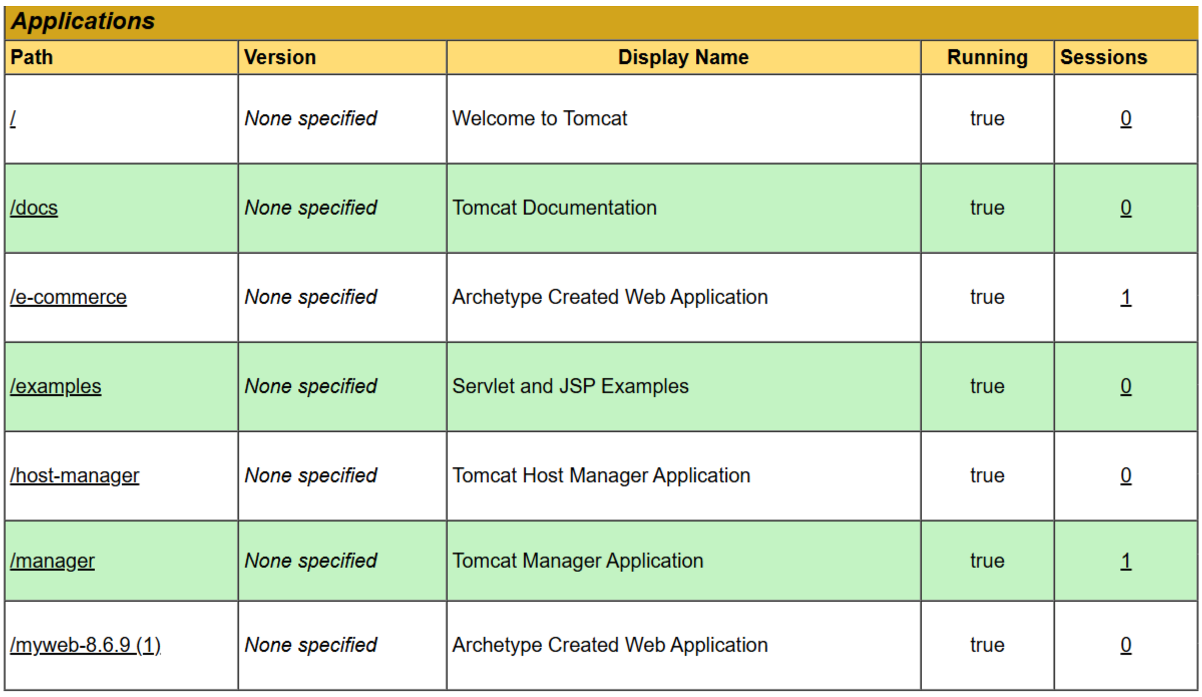
Screenshot 2:



WAR file to deploy

Select WAR file to upload myweb-8.6.9 (1).war

Screenshot 3:



Applications

Path	Version	Display Name	Running	Sessions
/	*None specified*	Welcome to Tomcat	true	0
/docs	*None specified*	Tomcat Documentation	true	0
/e-commerce	*None specified*	Archetype Created Web Application	true	1
/examples	*None specified*	Servlet and JSP Examples	true	0
/host-manager	*None specified*	Tomcat Host Manager Application	true	0
/manager	*None specified*	Tomcat Manager Application	true	1
/myweb-8.6.9 (1)	*None specified*	Archetype Created Web Application	true	0