

## Configuring Apache with Tomcat:

### Installation and Apache Configuration with Tomcat:

#### Step 1:

```
yum install wget httpd java-11-openjdk-devel -y
service httpd start
service httpd status
```

#### Step2:

```
wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.75/bin/apache-tomcat-9.0.75.tar.gz
tar -xvf apache-tomcat-9.0.75.tar.gz
cp -pr apache-tomcat-9.0.75 tomcat1
cp -pr apache-tomcat-9.0.75 tomcat2
ref : https://crunchify.com/how-to-run-multiple-tomcat-instances-on-one-server/
```

#### Step3:

```
sudo vi /etc/httpd/conf.d/proxy.conf

<VirtualHost *:80>

<Proxy balancer://mycluster>

    BalancerMember http://13.233.80.182:9090/
    BalancerMember http://13.233.80.182:8080/

</Proxy>

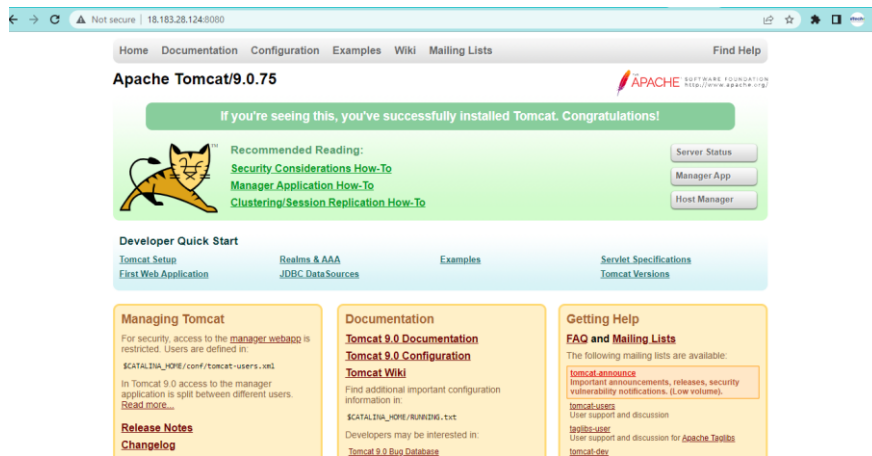
ProxyPreserveHost On

ProxyPass / balancer://mycluster/
ProxyPassReverse / balancer://mycluster/

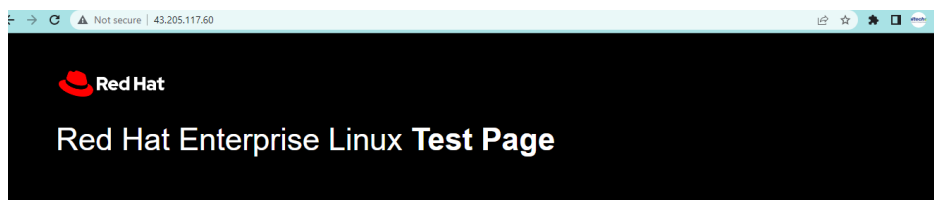
</VirtualHost>

service httpd restart
```

## Accessing with 8080 checking it working or not:



## Configure Apache with 80 port:



This page is used to test the proper operation of the HTTP server after it has been installed. If you can read this page, it means that the HTTP server installed at this site is working properly.

### If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

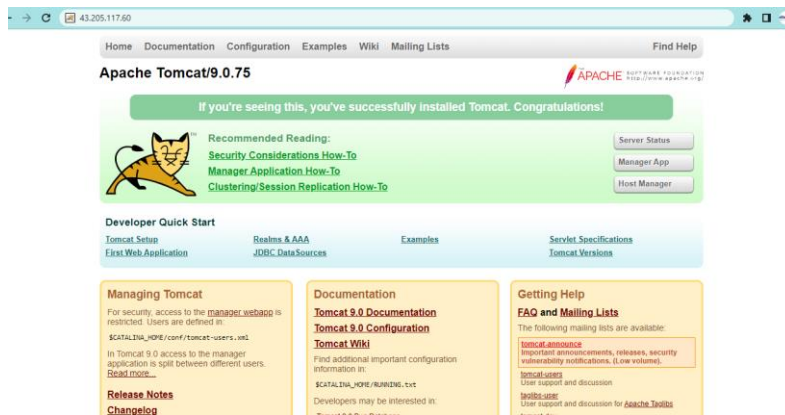
If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

### If you are the website administrator:

You may now add content to the webroot directory. Note that until you do so, people visiting your website will see this page, and not your content.

For systems using the Apache HTTP Server: You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

Whenever u enter Ip then it should go automatically hit Tomcat:



## # Configuring the SSL Certificate on Tomcat:

- `keytool -genkey -keyalg RSA -alias <local hostname> -keystore tomcat.jks -validity 90 -keysize 2048`

Example:

- `keytool -genkey -keyalg RSA -alias ip-172-31-10-159.ap-northeast-1.compute.internal -keystore tomcat.jks -validity 90 -keysize 2048`

## Set a password and remember it will be used in the next steps:

- after that setup first name and last name one should be <local hostname which was given in alias above>
- then go to `/home/ec2-user/conf/`
- `vi server.xml`

- remove the 8080 port then paste the below content in the same format :

<Connector

```
port="8080" maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
maxSpareThreads="75" enableLookups="false" disableUploadTimeout="true"
acceptCount="100"
```

```
scheme="https" secure="true" SSLEnabled="true" clientAuth="false"
sslProtocol="TLS" keyAlias="local-hostname"
```

```
keystoreFile="/root/tomcat.jks" keystorePass="Password which u had set above"
```

```
/>
```

Example:

```
<Connector
```

```
port="8080" maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
```

```
maxSpareThreads="75" enableLookups="false" disableUploadTimeout="true"
```

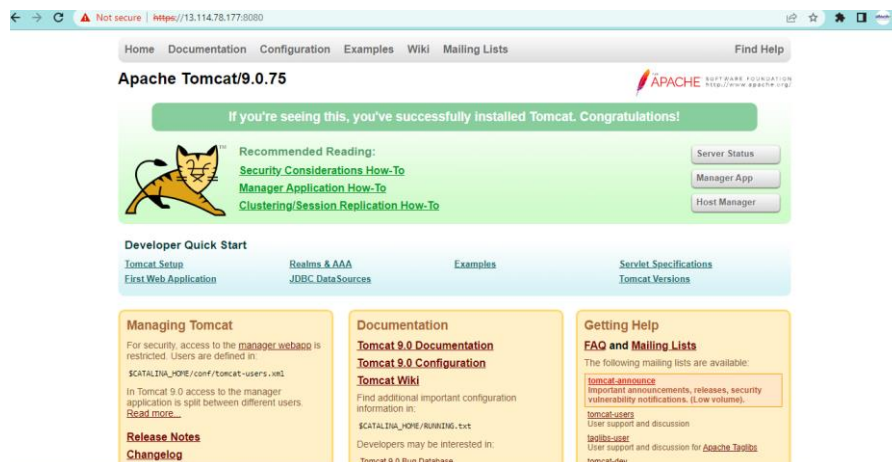
```
acceptCount="100"
```

```
scheme="https" secure="true" SSLEnabled="true" clientAuth="false"
```

```
sslProtocol="TLS" keyAlias="ip-172-31-10-159.ap-northeast-1.compute.internal"
```

```
keystoreFile="/root/tomcat.jks" keystorePass="anand@123"
```

```
/>
```



- Pre-install httpd and mod\_ssl:
- Installing SSL on Apache:
- yum install httpd mod\_ssl -y
- go to the cd /etc/httpd and create directory name certs and execute below commands

Go to the certs directory

- openssl genrsa -out server.key 2048
- openssl req -new -key server.key -out server.csr |||||---> openssl req -in server.csr -text --- this command to see the file in human readable format

**The above command will ask for information about our SSL certificate**

- openssl x509 -req -in server.csr -signkey server.key -days 365 -out server.crt |||||---> openssl x509 -in server.crt -text --- this command to see the file in human readable format
- Go to the <VirtualHost \_default\_:443> this line and change like this  
<VirtualHost \*:443>
- And check below lines has to be same on this file
- SSLEngine on
- SSLCertificateFile "/etc/httpd/certs/server.crt"
- SSLCertificateKeyFile "/etc/httpd/certs/server.key"

save and exit

and restart httpd

**And after doing this to make work load balance work in Apache go to :**

- /etc/httpd/conf.d/ssl\_conf
- And paste the below code:

<Proxy balancer://mycluster>

BalancerMember http://13.233.80.182:9090/

```
BalancerMember http://13.233.80.182:8080/

</Proxy>

ProxyPreserveHost On

ProxyPass / balancer://mycluster/

ProxyPassReverse / balancer://mycluster/
```

## # Installing and setup of SONARQUBE:

### Pre-Requirements of Installation:

It must have 4 CPUs or 4 GB of RAM workspace required to work on SonarQube.

step1:

Install java

sudo java-11-openjdk-devel wget unzip -y

step2:

Go to SonarQube Downloads:

wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.1.0.47736.zip

step3:

Install unzip command.

Yum install unzip

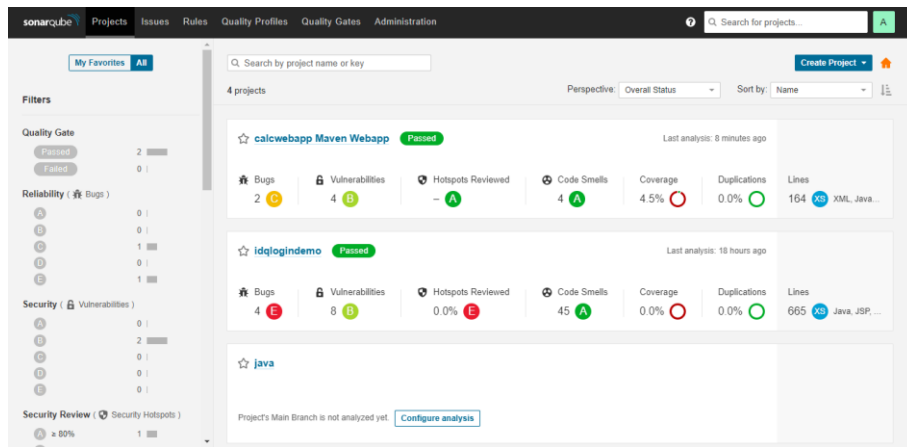
step4:

sudo unzip https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.1.0.47736.zip

**Then:**

Use the T2 Medium instant type :

1. Add user to perform in Sonar cube (Don't run as Root for the best practices).  
useradd sonar
2. passwd Shashi  
set password
3. Set the permissions using Chown:  
chown -R sonar:sonar sonar/ (it creates all same permission through out the sonarqube directory)
4. Then go to bin folder:  
cd bin  
cd linux.  
(before starting the sonar shift to the user and start otherwise we will face the errors)
5. Then start the sonar.  
sh sonar.sh start
6. To check the status of the sonar:  
sh sonar.sh status and check the port (ip:9090).  
User id & passwd:  
admin & admin

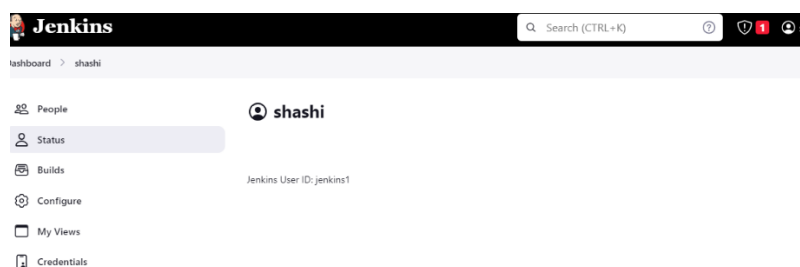


After logging in:

Go to administration and create a token which later used in Jenkins Integration:

- Jenkins Installation on Red hat:
- Install from this link: <https://pkg.jenkins.io/redhat-stable/>

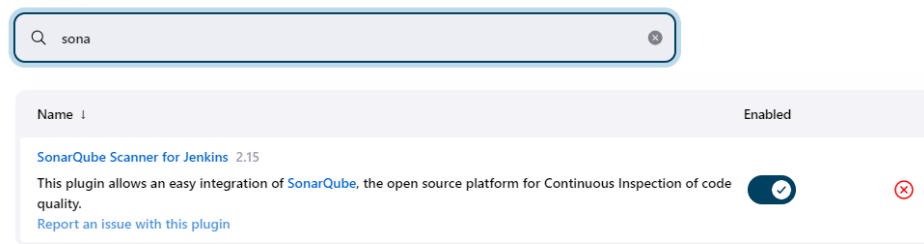
## Integrating & Deploying the Java Project on Tomcat Using Jenkins:



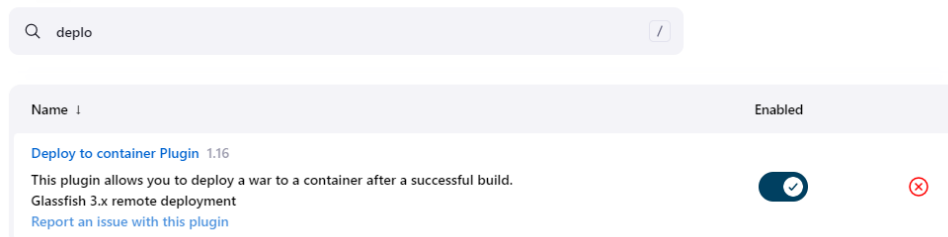
- Pre-Setup for Integrating and Deploying:

**Step1:**

- Install the Plugins which you can see below:

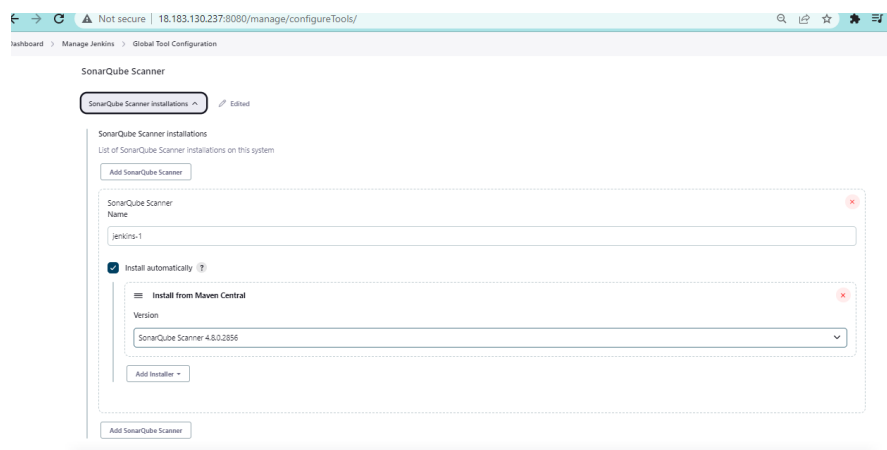


## Plugins



## Step2:

- Go to Global Configuration Tools and try to write same name of the token name in sonar :



## Step 3:

- Add your Tomcat user details which u edited in tomcat-users.xml file.

The example I have added:

- Username: admin & password: s3cet
- Then add: <http://18.183.28.124:8080/>
- Sonar one u already added in the above step so need not be required.



## Step 4:

- Add a new project and create a name:

The screenshot shows the Jenkins Configuration page for a project named 'shashi'. The 'Source Code Management' section is active, showing 'Git' as the selected provider. A repository URL is entered: 'https://github.com/rchidana/cacirebapp.git'. The 'Credentials' dropdown is set to '- none -'. There are buttons for 'Add' and 'Advanced'.

Dashboard > shashi > Configuration

**Configure**

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

**Source Code Management**

☐ None

☒ Git

Repositories

Repository URL:

Credentials:

Then:

The screenshot shows the Jenkins Configuration page for a project named 'shashi'. The 'Build Triggers' section is active, showing various build triggers. The 'Build Environment' section is also visible, showing options for workspace management and SonarQube integration. The 'Prepare SonarQube Scanner environment' checkbox is checked.

**Configure**

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

**Build Triggers**

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☐ Build periodically

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

**Build Environment**

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s)

☐ Add timestamps to the Console Output

☐ Inspect build log for published build scans

☒ Prepare SonarQube Scanner environment

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled. Will default to the one defined in the SonarQube installation.

And next step:

The screenshot shows the Jenkins Configuration page for a project named 'shashi'. The 'Build Steps' section is active, showing a single step named 'Execute shell'. The command entered is 'mvn package sonar:sonar'. There are buttons for 'Add' and 'Advanced'.

**Configure**

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

**Build Steps**

☐ Add timestamps to the Console Output

☐ Inspect build log for published build scans

☒ Prepare SonarQube Scanner environment

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled. Will default to the one defined in the SonarQube installation.

☐ Terminate a build if it's stuck

☐ With Ant

**Build Steps**

☒ Execute shell

Command

See the list of available environment variables

Dashboard > shashi > #1 > Console Output

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#1'

Git Build Data

### Console Output

```
Started by user shashi
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/shashi
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/rchidana/calwebapp.git
> git init /var/lib/jenkins/workspace/shashi # timeout=10
Fetching upstream changes from https://github.com/rchidana/calwebapp.git
> git --version # timeout=10
> git fetch --tags --force --progress -- https://github.com/rchidana/calwebapp.git +refs/heads/*:refs/remotes/
> git config remote.origin.url https://github.com/rchidana/calwebapp.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
```

**Follow the above steps and make the Integration complete:**

- Then we will go into Deployment:
- Now go into the Project and click configure and go to the last step Post Build Actions:

#### Deploy war/ear to a container

WAR/EAR files ?

\*\*/\*.war

Context path ?

Containers

##### Tomcat 9.x Remote

Credentials

deployer/\* (deployer)

Add +

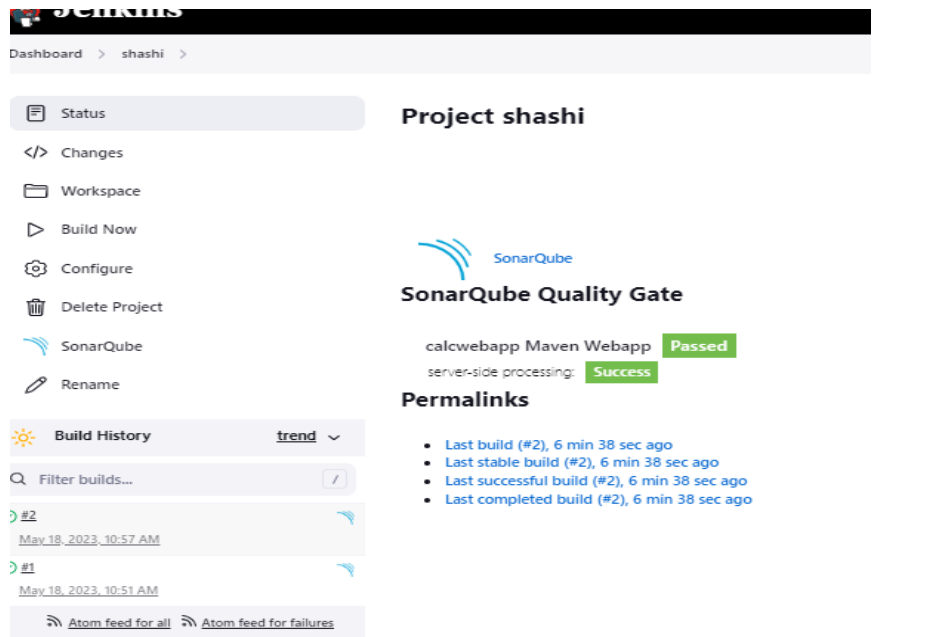
Tomcat URL ?

http://18.183.28.124:8080

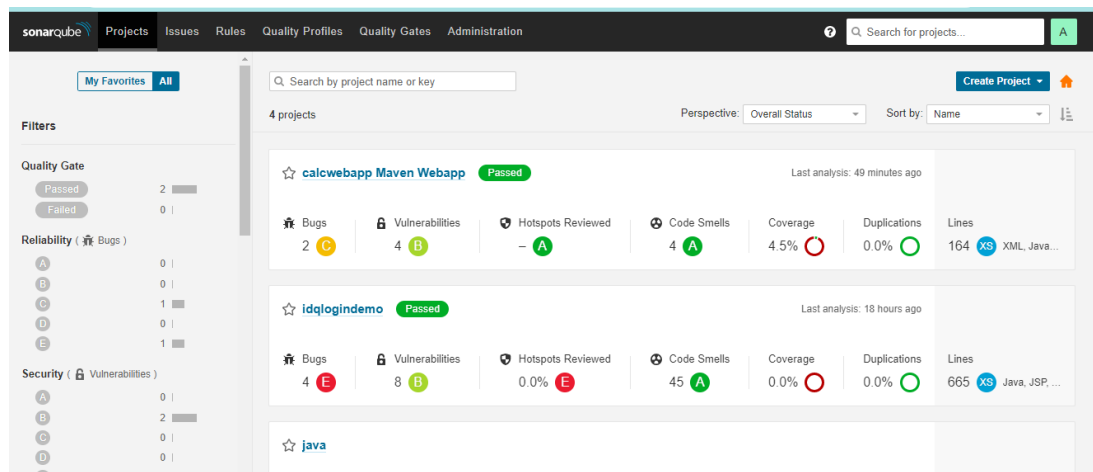
Advanced ▾

Add Container +

Save Apply

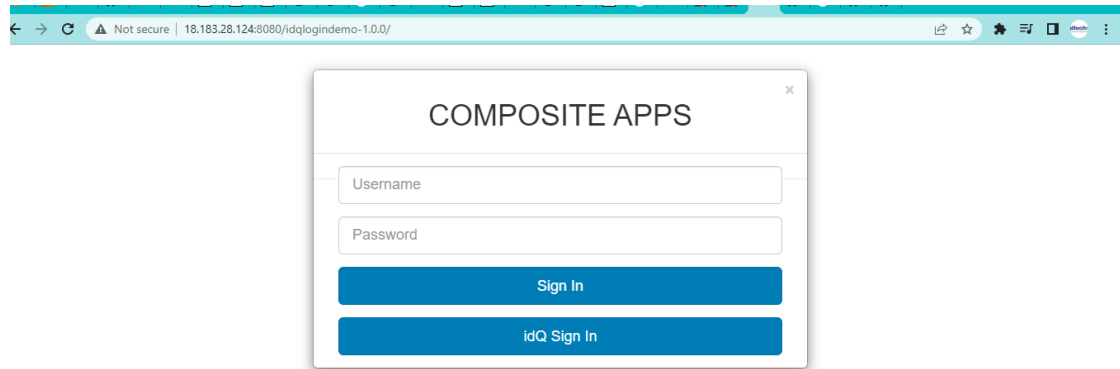


- In SonarQube the bugs, Quality of code, Code smells will be available.



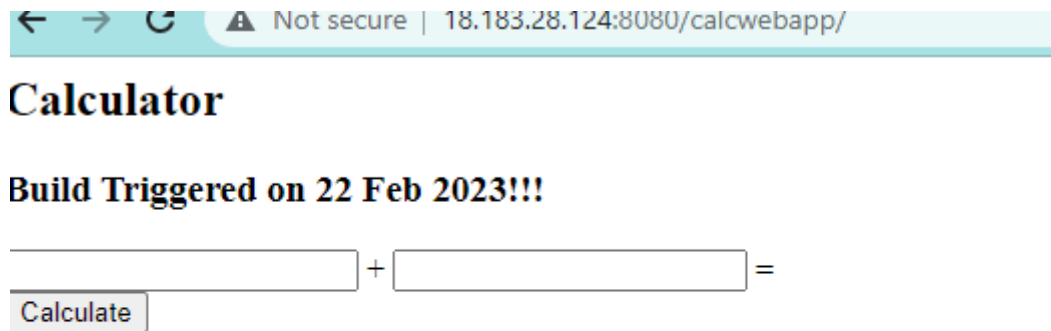
- The below two Projects I have deployed on Tomcat:

Project1:



A screenshot of a web browser window. The address bar shows a 'Not secure' warning and the URL '18.183.28.124:8080/idqlogindemo-1.0.0/'. A modal dialog box titled 'COMPOSITE APPS' is centered on the screen. It contains two input fields labeled 'Username' and 'Password'. Below these fields are two blue buttons: 'Sign In' and 'idQ Sign In'.

Project2:



A screenshot of a web browser window. The address bar shows a 'Not secure' warning and the URL '18.183.28.124:8080/calcwebapp/'. The page content includes the heading 'Calculator' and a message 'Build Triggered on 22 Feb 2023!!!'. Below the message is a simple calculator interface with two input fields separated by a plus sign, followed by an equals sign. A 'Calculate' button is positioned below the first input field.