Configuring Httpd with Tomcat In RHEL:

step 1:

Change normal user to Root user

yum install tar 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 -zxvf apache-tomcat-9.0.75.tar.gz

cd /apache-tomcat-9.0.75

step3:

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

Paste the below file in Proxy.conf

```
<VirtualHost *:80>

<Proxy balancer://mycluster>
   BalancerMember http://15.207.107.133:8080/  --- Tomcat Url
</Proxy>

   ProxyPreserveHost On

   ProxyPass / balancer://mycluster/
   ProxyPassReverse / balancer://mycluster/
</VirtualHost>
```

Save and Exit

service httpd restart

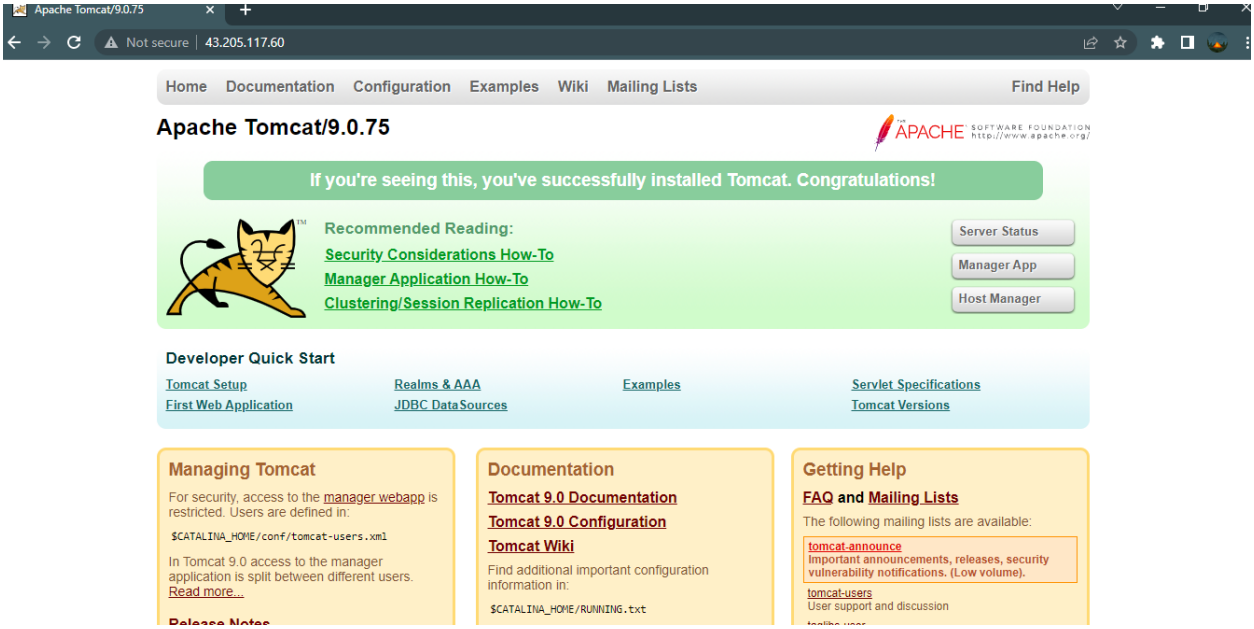
root@ip-172-31-38-229:/etc/httpd/conf.d

```
<VirtualHost *:80>

<Proxy balancer://mycluster>
    BalancerMember http://15.207.107.133:8080/
</Proxy>

    ProxyPreserveHost On

    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/
</VirtualHost>

~
~
~
~
~
~
~
~
~
~
~
~
~
```

if u get 503 service unavailable error then execute below command:

/usr/sbin/setsebool -P httpd_can_network_connect 1

After Changing these things when ever u hit public ip you will be redirect to tomcat webpage..



Installing Self Signed Certificate On Httpd In RHEL:

To install self signed certificate we have install some prerequisites:-

pre install httpd and mod_ssl

yum install httpd mod_ssl -y

Go to the cd /etc/httpd and create directory name certs:

mkdir certs

Go to the certs directory and execute below commands to generate ssl keys:

openssl genrsa -out server.key 2048

openssl req -new -key server.key -out server.csr

Above command wil ask some information about our ssl certificate

To Read above command server.csr key in human readle format execute below command:

openssl req -in server.csr -text → for human readle format.

openssl x509 -req -in server.csr -signkey server.key -days 365 -out server.crt

To Read above command server.crt key in human readle format execute below command:

openssl x509 -in server.crt -text → for human readle format.

now go back to httpd and go to conf.d and edit ssl.conf.

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"

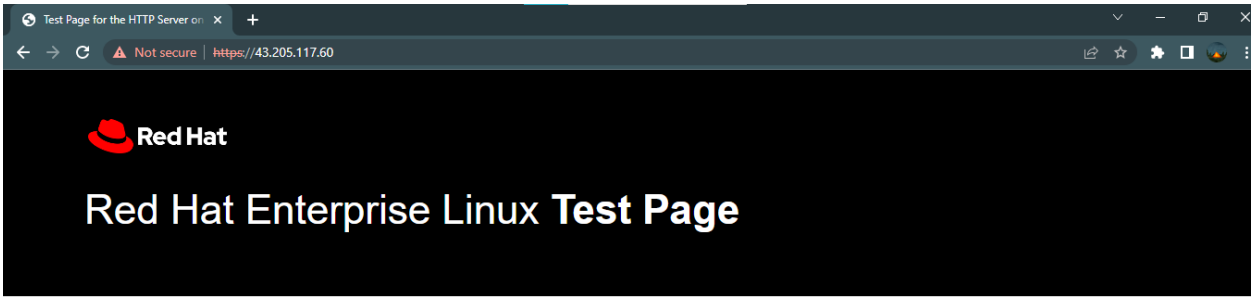
save and exit

```
## SSL Virtual Host Context
##

<VirtualHost *:443>

# General setup for the virtual host, inherited from global configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443

# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
LogLevel warn

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   List the protocol versions which clients are allowed to connect with.
#   The OpenSSL system profile is used by default.  See
#   update-crypto-policies(8) for more details.
#SSLProtocol all -SSLv3
#SSLProxyProtocol all -SSLv3

#   User agents such as web browsers are not configured for the user's
#   own preference of either security or performance, therefore this
#   must be the prerogative of the web server administrator who manages
#   cpu load versus confidentiality, so enforce the server's cipher order.
SSLHonorCipherOrder on

#   SSL Cipher Suite:
#   List the ciphers that the client is permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
#   The OpenSSL system profile is configured by default.  See
#   update-crypto-policies(8) for more details.
SSLCipherSuite PROFILE=SYSTEM
SSLProxyCipherSuite PROFILE=SYSTEM

#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that restarting httpd will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
#   Some ECC cipher suites (http://www.ietf.org/rfc/rfc4492.txt)
#   require an ECC certificate which can also be configured in
#   parallel.
SSLCertificateFile /etc/httpd/certs/server.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
#   ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile /etc/httpd/certs/server.key
```

and restart httpd

service httpd restart

Now we have can access with https httpd page: https://public-ip



Red Hat Enterprise Linux **Test Page**

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`.

Installing Self Signed Certificate On Tomcat In RHEl:

Pre Requistes Java open-JDK-11 and Tomcat

keytool -genkey -keyalg RSA -alias Private IP DNS name -keystore tomcat.jks -validity 90 -keysize 2048

eg:

keytool -genkey -keyalg RSA -alias ip-172-31-38-229.ap-south-1.compute.internal -keystore tomcat.jks -validity 90 -keysize 2048

Above Command will ask some information regarding our ssl certificate and it will ask some password remember it and it will be use in next steps.

Now Go to tomcat file and move to conf folder and edit server.xml

In the below file if your keystorefile is in another path give that path:

<Connector

port="8443" 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="kommi123"

/>

Paste the above file as per the below image:

root@ip-172-31-38-229:~/tomcat1/conf

```
      so you may not define subcomponents such as "Valves" at this level.
      Documentation at /docs/config/service.html
  -->
  <Service name="Catalina">

    <!--The connectors can use a shared executor, you can define one or more named thread pools-->
    <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
    -->


    <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               maxParameterCount="1000"
               />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               maxParameterCount="1000"
               />
    -->
    <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
         This connector uses the NIO implementation. The default
         SSLImplementation will depend on the presence of the APR/native
         library and the useOpenSSL attribute of the AprLifecycleListener.
         Either JSSE or OpenSSL style configuration may be used regardless of
         the SSLImplementation selected. JSSE style configuration is used below.
    -->
    <Connector
        port="8443" 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-172-31-38-229.ap-south-1.compute.internal"
        keystoreFile="/root/tomcat.jks" keystorePass="kommi123"/>
    <!--
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
               maxThreads="150" SSLEnabled="true"
               maxParameterCount="1000"
               >
        <SSLHostConfig>
            <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                         type="RSA" />
```

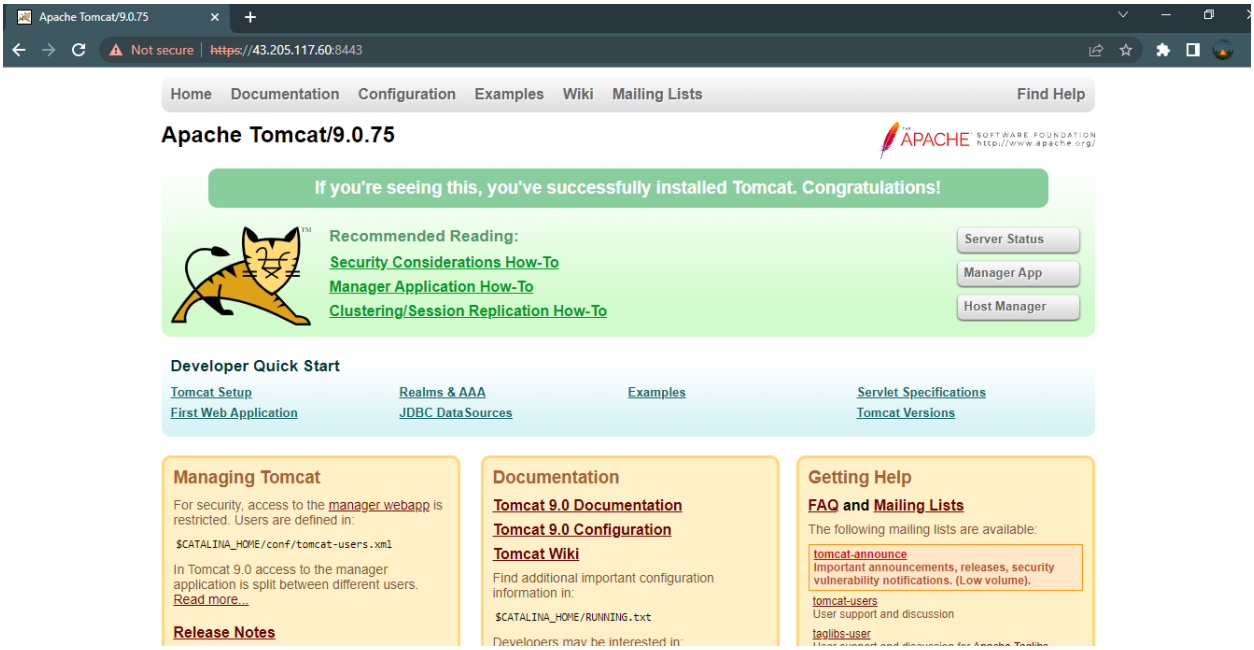
Save and Exit

Then Go back one step and go back bin folder and start tomcat and shutdown tomcat and start tomcat by using below command:

```
./startup.sh –start tomcat command
./shutdown.sh -shutdown tomcat command
./startup.sh –start tomcat command
```

Now you can access tomcat with https

By using this command https://public-ip:8443



Configuring Https Httpd with Https Tomcat In RHEL:

step 1:

Change normal user to Root user

Before configuring this we have to install self signed certificate as above we have generated.

yum install tar wget httpd mod_ssl 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 -zxvf apache-tomcat-9.0.75.tar.gz

cd /apache-tomcat-9.0.75

cd/conf

vi server.xml

and edit this as we have edited while generating ssl in tomcat.

```
root@ip-172-31-38-229:~/tomcat1/conf
      so you may not define subcomponents such as "Valves" at this level.
      Documentation at /docs/config/service.html
  -->
  <Service name="Catalina">

    <!--The connectors can use a shared executor, you can define one or more named thread pools-->
    <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
    -->


    <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               maxParameterCount="1000"
               />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               maxParameterCount="1000"
               />
    -->
    <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
         This connector uses the NIO implementation. The default
         SSLImplementation will depend on the presence of the APR/native
         library and the useOpenSSL attribute of the AprLifecycleListener.
         Either JSSE or OpenSSL style configuration may be used regardless of
         the SSLImplementation selected. JSSE style configuration is used below.
    -->
    <Connector
        port="8443" 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-172-31-38-229.ap-south-1.compute.internal"
        keystoreFile="/root/tomcat.jks" keystorePass="kommi123"/>
    <!--
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
               maxThreads="150" SSLEnabled="true"
               maxParameterCount="1000"
               >
        <SSLHostConfig>
            <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                         type="RSA" />
```

And go to the bin folder and shutdown tomcat and start tomcat.

Now go back to cd /etc/httpd/conf.d

And edit ssl.conf file and and the below script in ssl.conf as per the below image.

```
<Proxy balancer://mycluster>
    BalancerMember http://43.205.117.60:8080/ --- tomcat http URL
</Proxy>

    ProxyPreserveHost On

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

```
root@ip-172-31-38-229:/etc/httpd/conf.d
#     This exports two additional environment variables: SSL_CLIENT_CERT and
#     SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#     server (always existing) and the client (only existing when client
#     authentication is used). This can be used to import the certificates
#     into CGI scripts.
#   o StdEnvVars:
#     This exports the standard SSL/TLS related `SSL_*' environment variables.
#     Per default this exportation is switched off for performance reasons,
#     because the extraction step is an expensive operation and is usually
#     useless for serving static content. So one usually enables the
#     exportation for CGI and SSI requests only.
#   o StrictRequire:
#     This denies access when "SSLRequireSSL" or "SSLRequire" applied even
#     under a "Satisfy any" situation, i.e. when it applies access is denied
#     and no other module can change it.
#   o OptRenegotiate:
#     This enables optimized SSL connection renegotiation handling when SSL
#     directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory "/var/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>
<Proxy balancer://mycluster>
    BalancerMember http://43.205.117.60:8080/
</Proxy>

    ProxyPreserveHost On

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

#   SSL Protocol Adjustments:
#   The safe and default but still SSL/TLS standard compliant shutdown
#   approach is that mod_ssl sends the close notify alert but doesn't wait for
#   the close notify alert from client. When you need a different shutdown
#   approach you can use one of the following variables:
#   o ssl-unclean-shutdown:
#     This forces an unclean shutdown when the connection is closed, i.e. no
#     SSL close notify alert is sent or allowed to be received.  This violates
#     the SSL/TLS standard but is needed for some brain-dead browsers. Use
#     this when you receive I/O errors because of the standard approach where
#     mod_ssl sends the close notify alert.
#   o ssl-accurate-shutdown:
#     This forces an accurate shutdown when the connection is closed, i.e. a
```

Save and exit

And restart the httpd

Systemctl restart httpd.

Now we can access tomcat with https public ip :

https://public-ip

Apache Tomcat/9.0.75

Not secure | https://43.205.117.60

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/9.0.75

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

Server Status

Manager App

Host Manager

Developer Quick Start

Tomcat Setup | Realms & AAA | Examples | Servlet Specifications

First Web Application | JDBC DataSources | 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 application is split between different users. Read more...

Release Notes

Changelog

Documentation

Tomcat 9.0 Documentation

Tomcat 9.0 Configuration

Tomcat Wiki

Find additional important configuration information in:

$CATALINA_HOME/RUNNING.txt

Developers may be interested in:

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).

tomcat-users
User support and discussion

taglibs-user
User support and discussion for Apache Taglibs

Continuous Integration and continuous Deployment (CI/CD):

Continuous Integration Jenkins with Git, SonarQube and continuous Deployment with Tomcat:

Take 3 RHEL servers.

One for Jenkins – T2small

One for SonarQube – T2Small

One for Tomcat – T2Micro

Installation Of Jenkins:

1. Go to the Jenkins.io website click on download click on RedHat and execute First Four Commands.
2. Start the Jenkins by using below command
3. systemctl start Jenkins and check the status systemctl status Jenkins.
4. After coming into the running state Jenkins access the Jenkins in web by using your public and with port number 8080
5. Ex : - public-ip:8080
6. Install git maven in this server.
7. Yum install git maven -y

Installation Of SonarQube:

1. Pre installation for SonarQube is java open-jdk-11
2. Install SonarQube by going to this website sonarqube.org click on download go to the bottom of the page and copy the link address of that link and go to the cd /opt and paste the link by using wget command .
3. And unzip that file by using unzip command .
4. Create a user with name of sonar and for that sonar file give user permissions and group permissions as sonar by using below command.
5. chown -R sonar:sonar /opt/sonarfile.
6. Switch to sonar user and go to the cd /opt/sonarqube/bin/linux-64 and start the sonar by using below command
7. ./sonar.sh start and check the status by using below command
8. ./sonar.sh status
9. Access the SonarQube portal in the web with the public ip
10. Below command to access the sonarqube
11. Public-ip:9000
12. By default SonarQube credentials are admin and admin.

Installation Of Tomcat:

1. Pre-Requisite for tomcat is java
2. Install java by using below command:
3. Yum install java-11-openjdk-devel -y
4. Go to the web search this tomcat.apache.org and click side on download Tomcat9 copy the link of tar file by using wget command paste that link and tar file will be downloaded and untar that by using tar -zxvf command.
5. Now go to the vi apache-tomcat-9.0.75/webapps/docs/META-INF/context.xml file delete these 2 lines <Valve className="org.apache.catalina.valves.RemoteAddrValve"
6. allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
7. vi apache-tomcat-9.0.75/webapps/examples/META-INF/context.xml delete these 2 lines <Valve className="org.apache.catalina.valves.RemoteAddrValve"
8. allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
9. vi apache-tomcat-9.0.75/webapps/host-manager/META-INF/context.xml delete these 2 lines <Valve className="org.apache.catalina.valves.RemoteAddrValve"
10. allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
11. vi apache-tomcat-9.0.75/webapps/manager/META-INF/context.xml delete these 2 lines <Valve className="org.apache.catalina.valves.RemoteAddrValve"
12. allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
13. Now go back to conf folder and edit tomcat-users.xml in this file delete all the lines and paste the below script.

    <tomcat-users>
    <role rolename="manager-gui"/>
    <user username="admin" password="Admin" roles="manager-gui,manager-script,manager-admin,manager-status"/>
    </tomcat-users>

    Now go back to the bin folder and start the tomcat by using below command
    ./startup.sh

Now We will Integrate Jenkins with SonarQube and Git:

1. Go to the SonarQube server and create a project name as java and generate a code and save that code in notepad.

sonarqube Projects Issues Rules Quality Profiles Q

Create a project

All fields marked with * are required

Project display name *

java

Up to 255 characters. Some scanners might override the value you provide.

Project key *

java

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Set Up

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

java master

Overview Issues Security Hotspots Measures Code Activity

Analyze your project

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

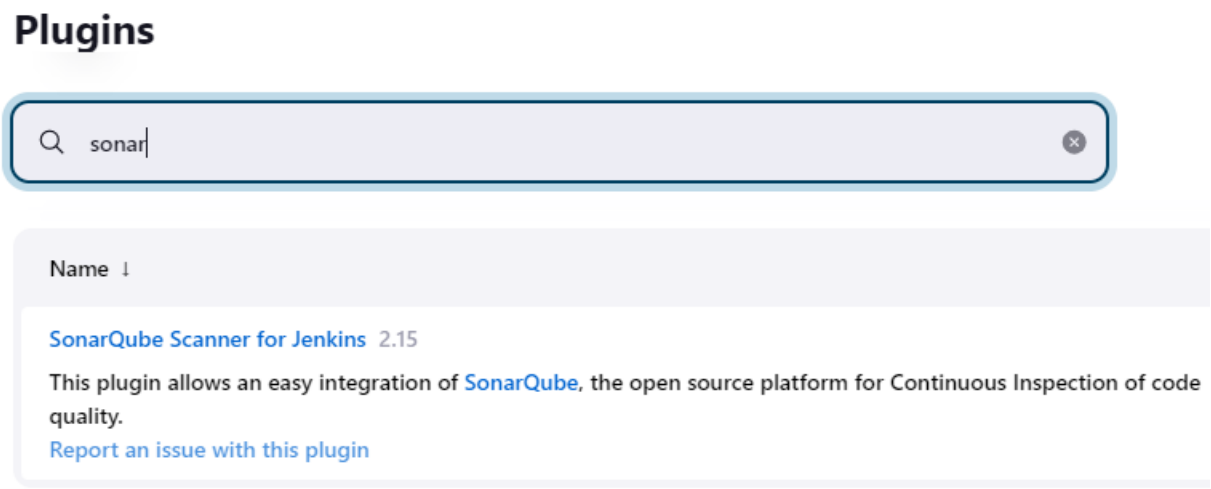
1 Provide a token

java: **4523add268267672dc17ccae19b75c7685ed3dfc**

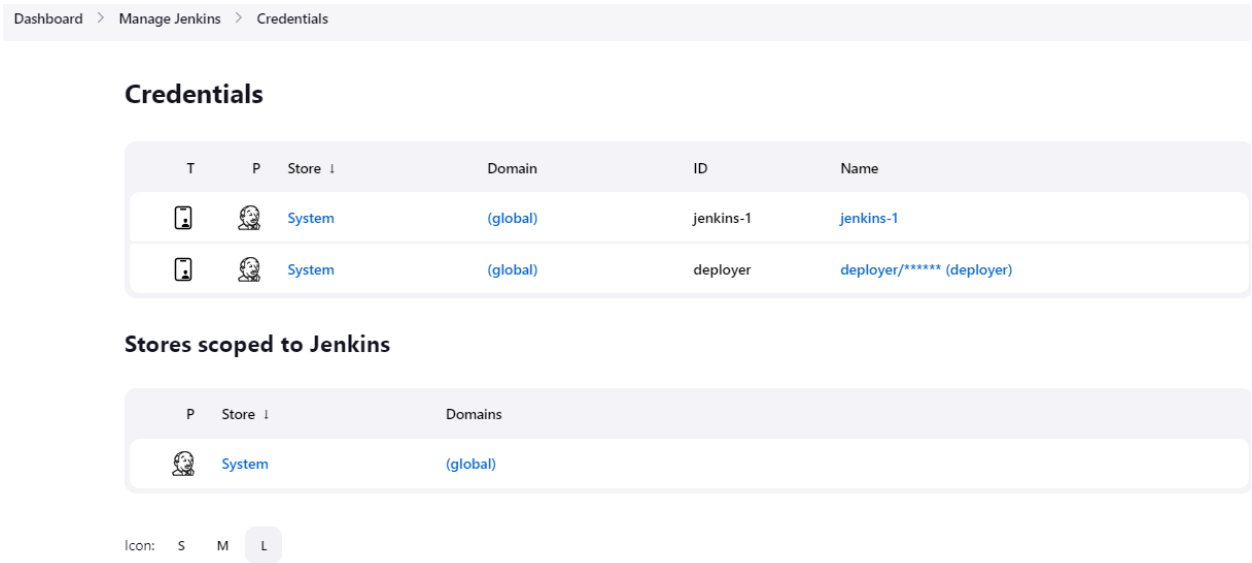
The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your user account.

Continue

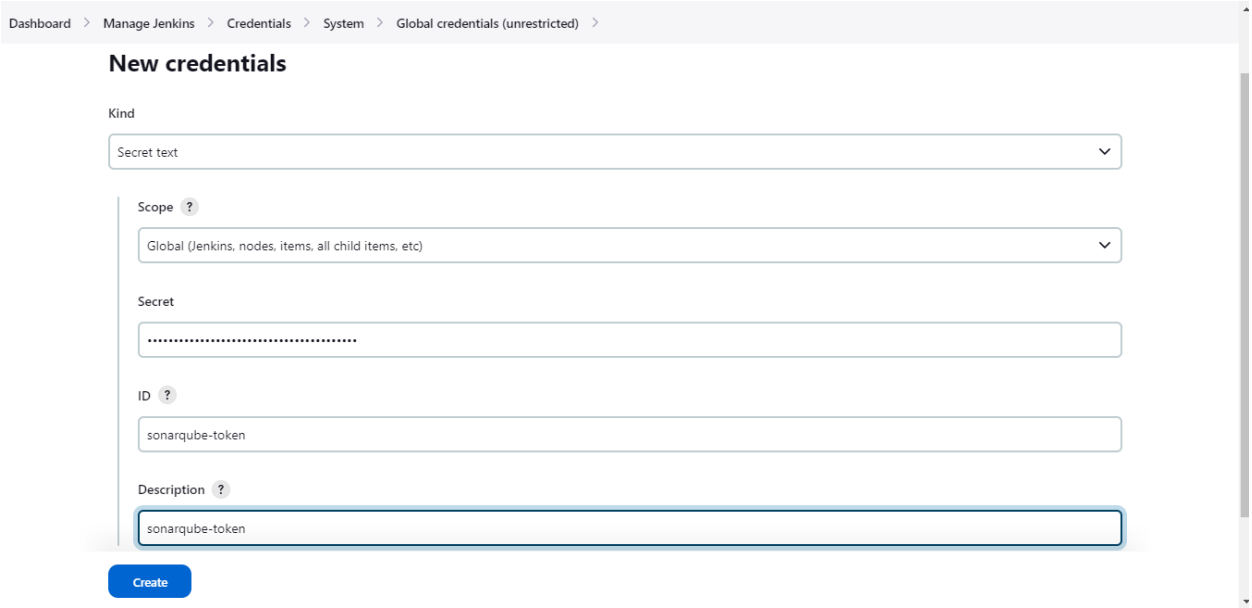
2. Now go to the Jenkins go to the manage Jenkins and go to the manage plugins click on available search SonarQube scanner for Jenkins install that without restart.



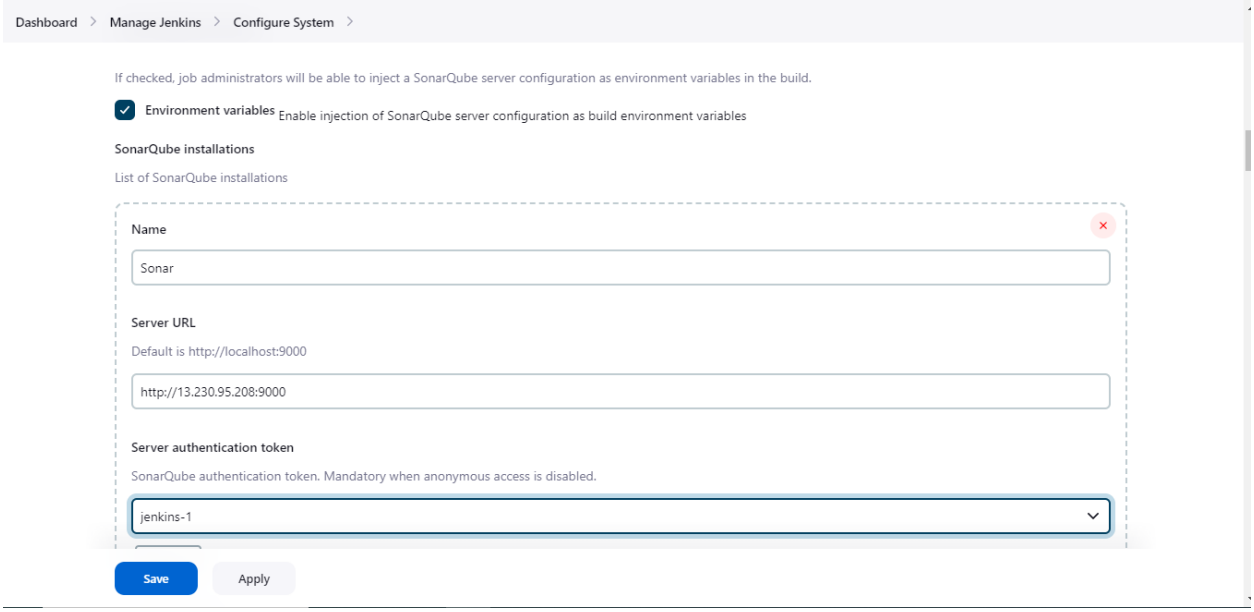
3. Now go to the manage Jenkins and go to the credentials and go to the global credentials.



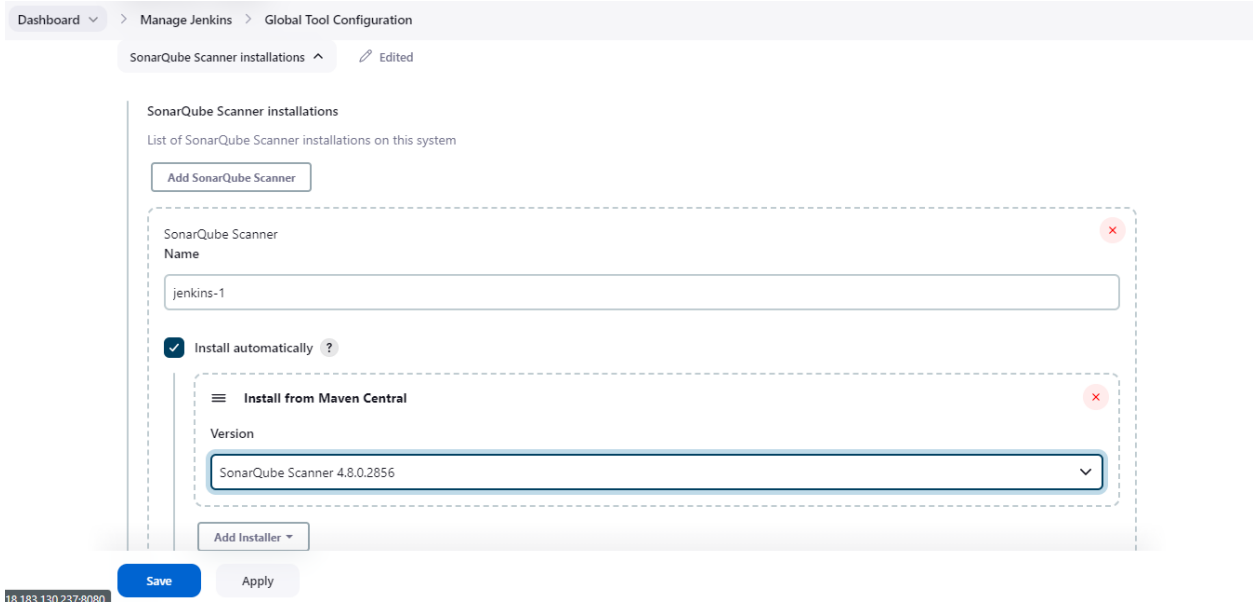
4. Now click on add credentials on side select their secret text and paste the code that we generated in sonarqube. And details as per the below image or your wish.



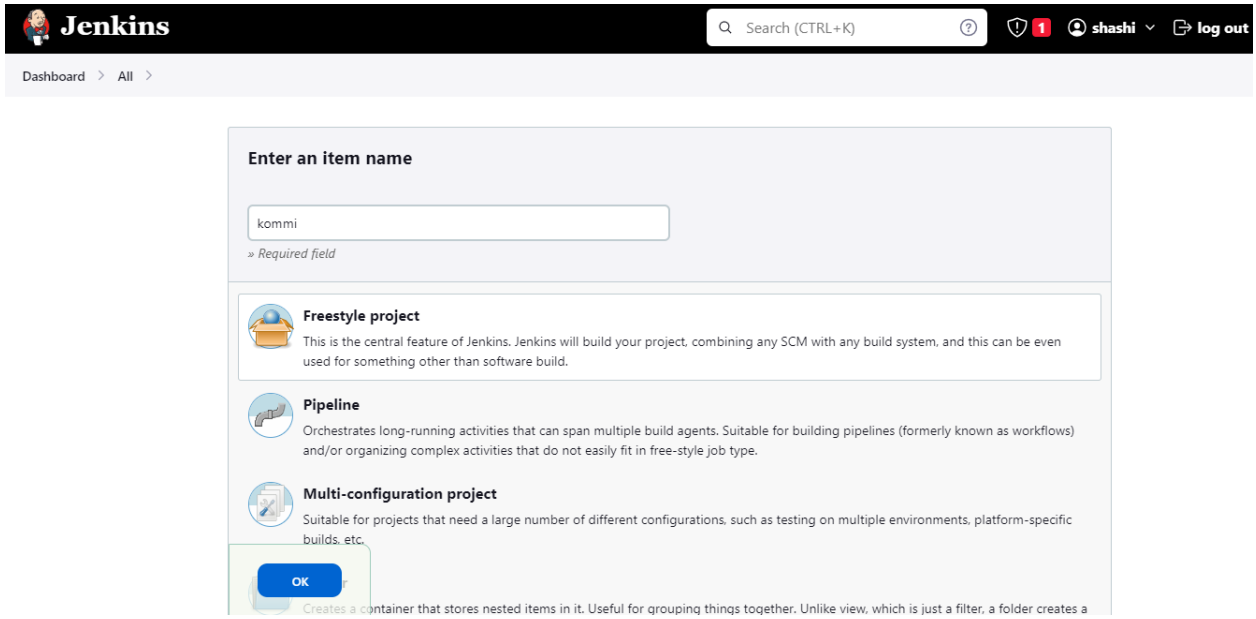
5. Now go to manage Jenkins and go to configure system scroll down to sonarqube servers and enable environment variables and next click on add the name as sonar and your soanrqube version number like sonarqube-9.1 and give the sonarqube url their like this http://13.230.95.208:9000 and give your credentials name their before you added in 4th step. And click on save and apply.



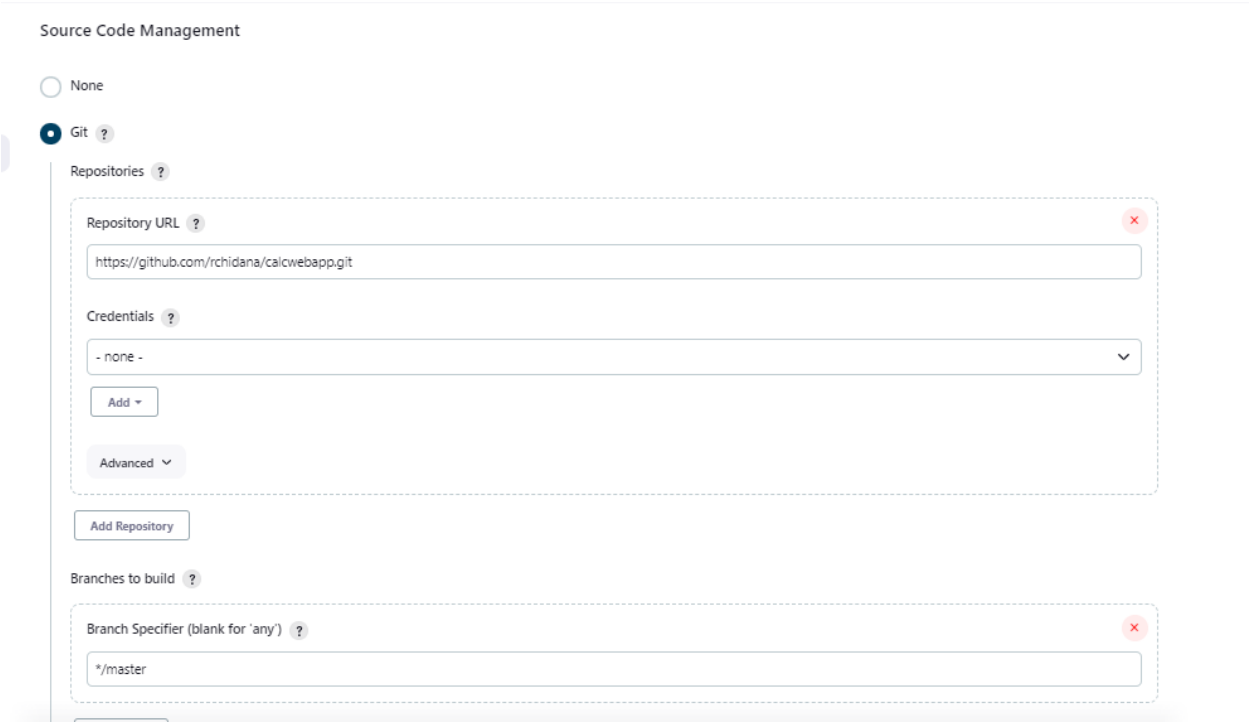
6. Now go to the manage Jenkins and go to the global tool configuration scroll down to sonarQube scanner installations click on add SonarQube Scanner give name their how ever you want and select the version their in picklist click on apply and save



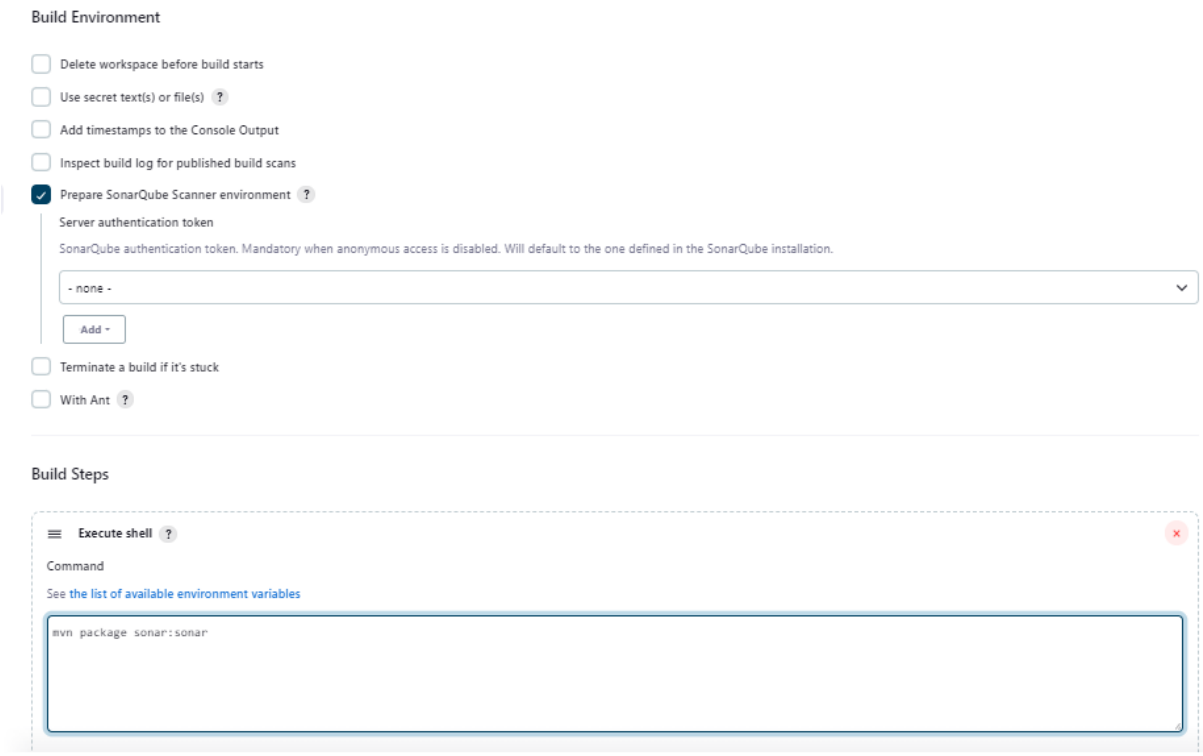
7. Now go back to dashboard click on new item enter name for your job and select freestyle project



8. Scroll down to the source code management and select git their and provide your git hub repository link and select your if the code is in master then give master their if the code is in main give the main their like below image.



9. Next go to the build environment enable on Prepare SonarQube Scanner environment give it none their it will automatically take your credentials and next go to the build steps and click on execute shell in that shell enter this mvn package sonar:sonar

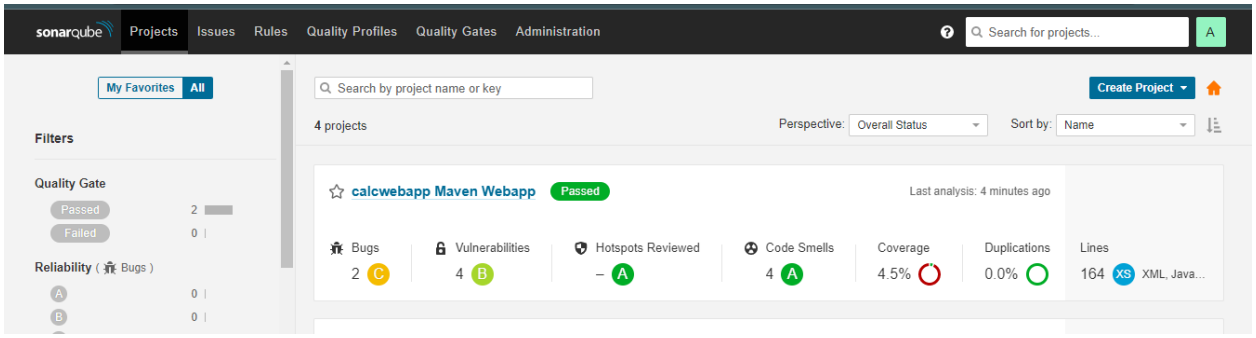


10. Click on Save and build if the build was success then you will see the output in sonarqube as static code analysis

Dashboard > kommi > #1 > Console Output

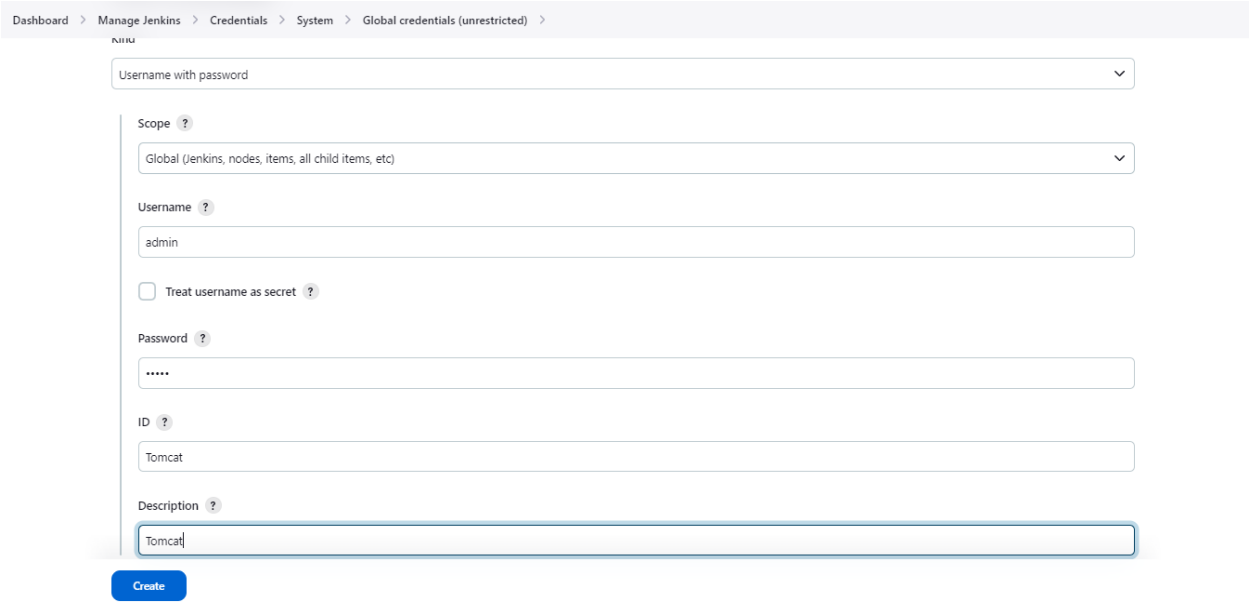
```
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=0ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=23ms
[INFO] CPD Executor 1 file had no CPD blocks
[INFO] CPD Executor Calculating CPD for 2 files
[INFO] CPD Executor CPD calculation finished (done) | time=8ms
[INFO] Analysis report generated in 102ms, dir size=116.7 kB
[INFO] Analysis report compressed in 27ms, zip size=23.5 kB
[INFO] Analysis report uploaded in 28ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://13.230.95.208:9000/dashboard?id=com.cruds.demo%3Acalcwebapp
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis
report
[INFO] More about the report processing at http://13.230.95.208:9000/api/ce/task?id=AYguXsdJ8xWsJ7XZhR9F
[INFO] Analysis total time: 7.334 s
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  14.066 s
[INFO] Finished at: 2023-05-18T10:19:03Z
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

11. Now go to the SonarQube dashboard there you will see the code analysis .

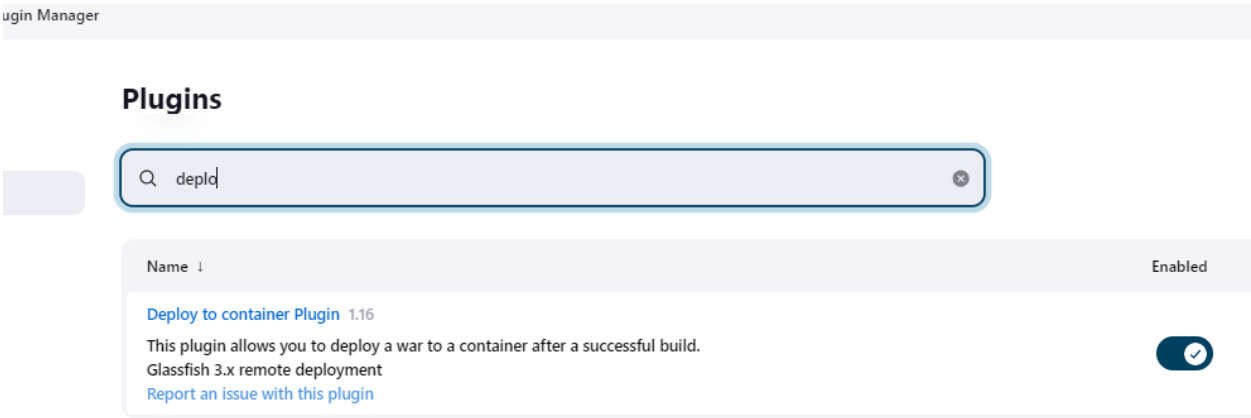


Now we will deploy Jenkins with Tomcat:

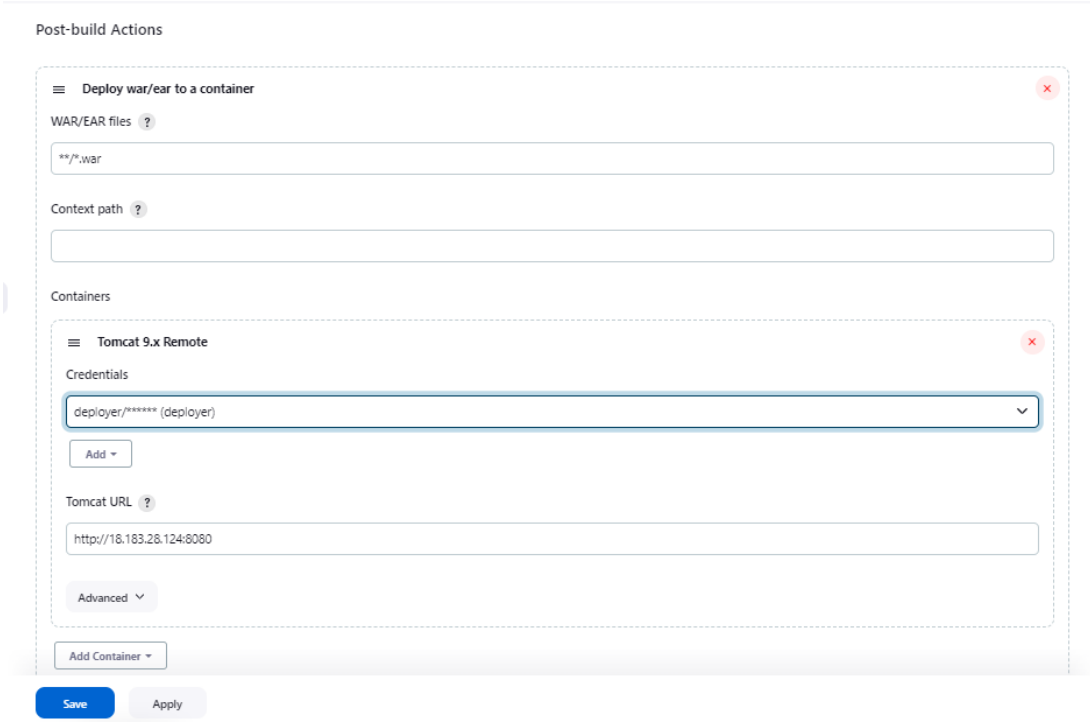
1. Go to the manage Jenkins and go to the credentials manager click on add credentials and select their as username with password give the credentials as before you have given in tomcat user.xml file give id as tomcat and description as tomcat.



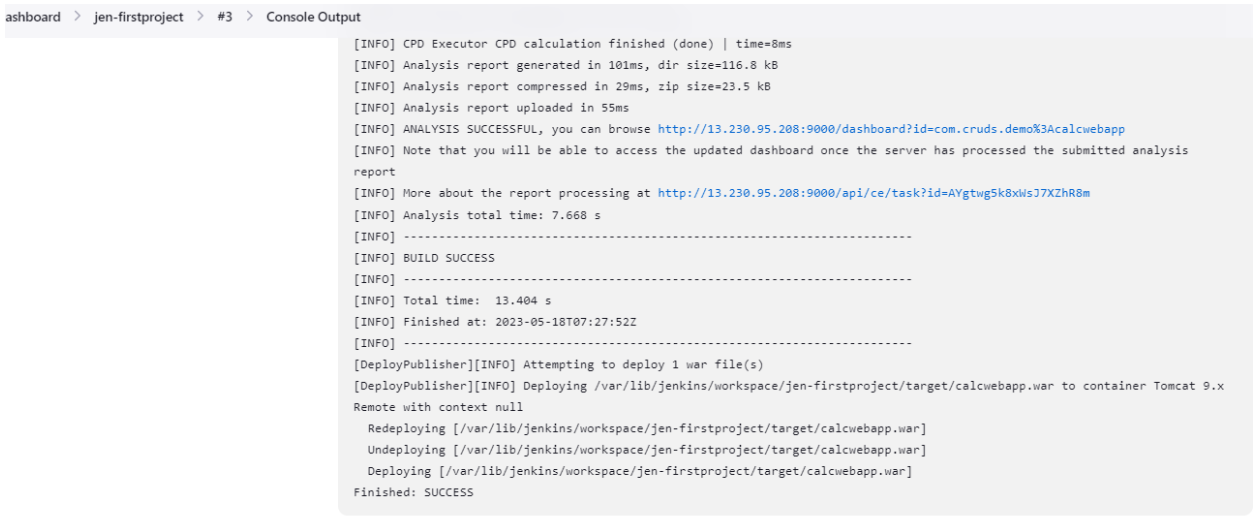
2. Now go back to manage Jenkins and go to the manage plugins and search deploy to container plugin click on install without restart.



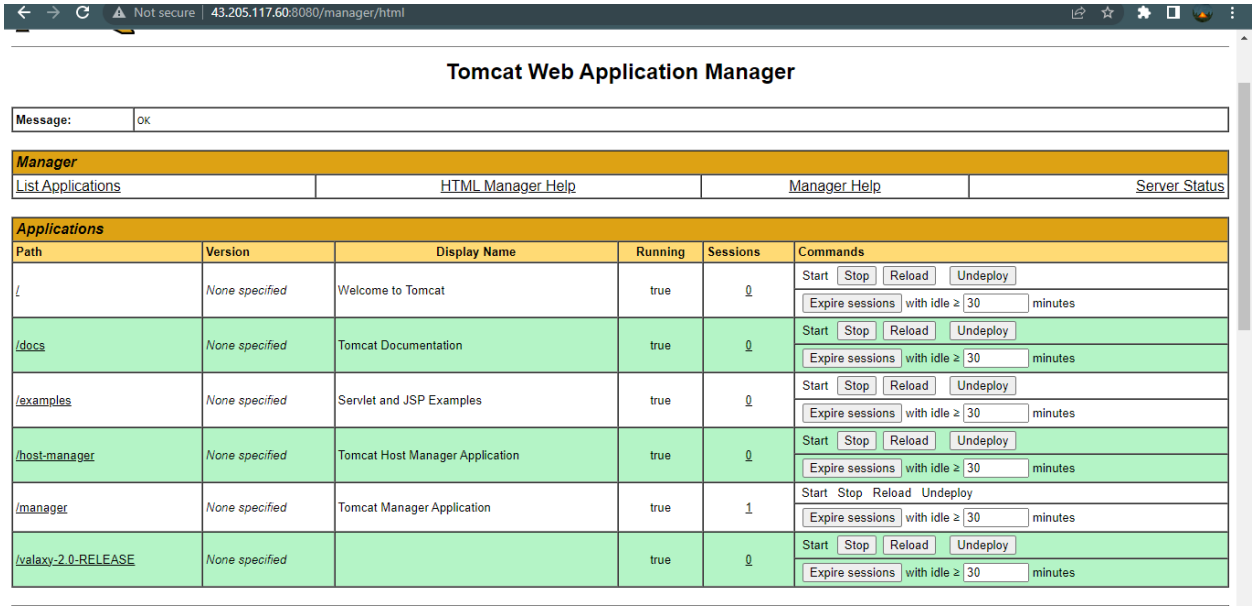
3. Now go back to your job that you have been created and click on side configure and scroll down to below and select post-build actions and select their **Deploy war/ear to a container** and give in this WAR/EAR files box as **/*.war and select your container as your tomcat version mine is 9 so that's why I have selected as tomcat 9 their and provide your credentials their before we added that one and provide your tomcat url their save and apply. Details will looks like same as below image.



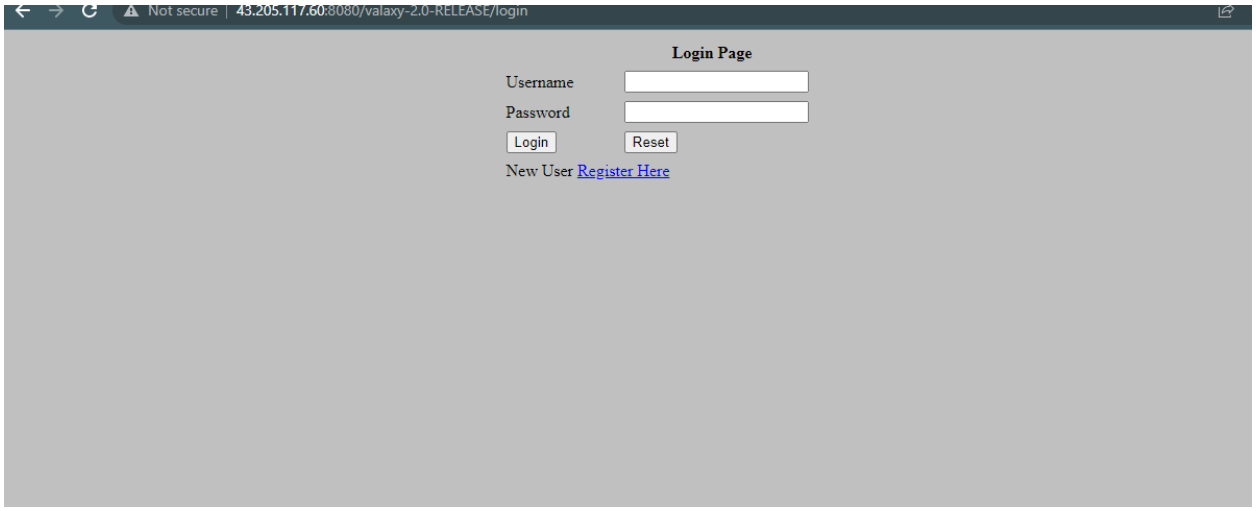
4. Now click on build if the build is success like the below output .



5. Now go to the tomcat application click on manager app give the user name and password that you have given tomcat.users.xml file.



6. Here my application name is valaxy-2.0 release when I ever I click that I can access that application.



7. Here is the complete CI/CD by using Jenkins SonarQube git and tomcat.