Jenkins Class – 4

In the devops cycle we have Plan, Code, Build, Test, deploy, monitoring.

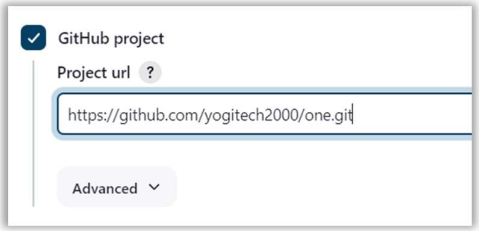
We already know till test the code. So now we have to deploy the code in server.

Tomcat is one of the deployment servers. We need to create new instance for deploy the code. Once build the code in maven, we will get the war file. We should deploy it to the application server.

Before create tomcat application server we have to create the code and host it in public/private repository then build the code by maven.

So, we need to install maven in current instance with java1.8.0. Now go to the Jenkins then configure it as follow below steps.

1. Add the github repo link in GitHub project.

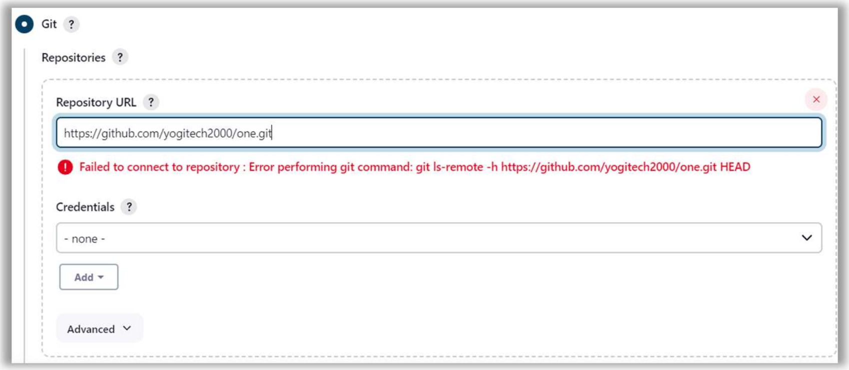


☑ GitHub project

Project url ?

https://github.com/yogitech2000/one.git

2. Add the SCM (Source Code Management) url. If the instace doesn't install git we will get the below error. After install the git in aws instance where the Jenkins has been installed the git error will disappear.



● Git ?

Repositories ?

Repository URL ?

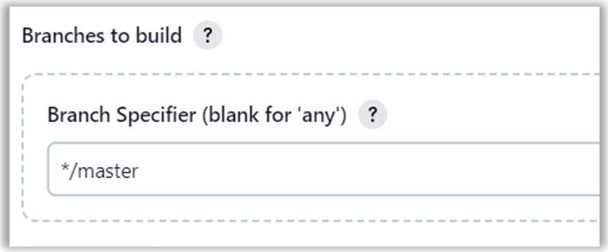
https://github.com/yogitech2000/one.git

Failed to connect to repository : Error performing git command: git ls-remote -h https://github.com/yogitech2000/one.git HEAD

Credentials ?

- none -

3. Select the branch.

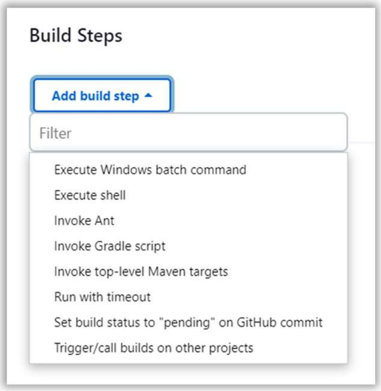


Branches to build ?

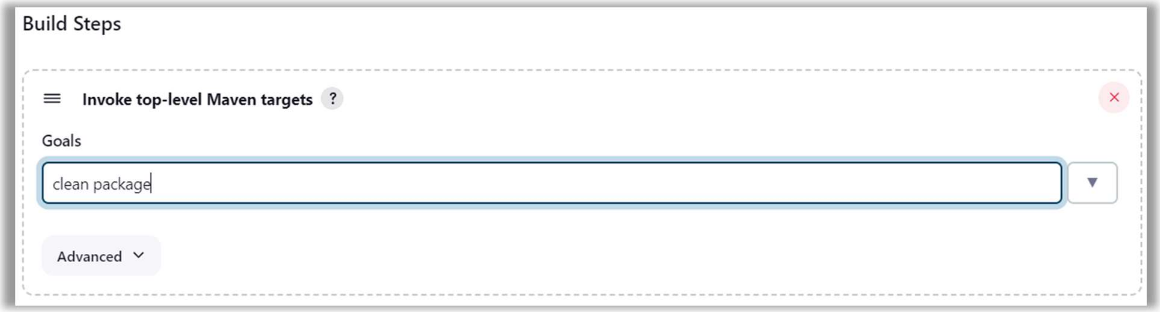
Branch Specifier (blank for 'any') ?

*/master

4. Select the build tool in build steps. We are using maven to build the code. So we have to choose the **Invoke top-level Maven targets.**

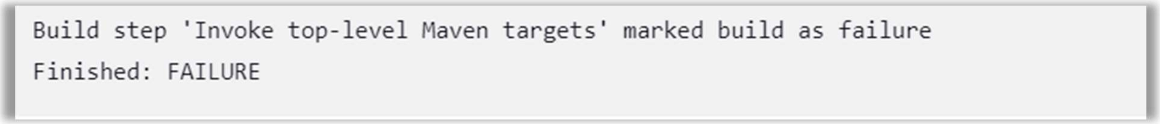

Build Steps
Add build step
Filter
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
Trigger/call builds on other projects

Then enter the maven command in goals field. We have a command to build the code in single command which is **mvn clean package.** but we can ignore **mvn** command because we already selected the maven tool in build steps.



Goals: clean package

Before build the job make sure that **git and maven with java 1.8.0 has been installed**. If you did not install maven we will get the below error in console log and job also will not build.



```
Build step 'Invoke top-level Maven targets' marked build as failure
Finished: FAILURE
```

After build the job we can see the **target** folder in **workspace** of the Jenkins job. Now we have war file. So we should deploy it on the tomcat application server.

Now we need to launch new instance then follow the below steps.

Tomcat will work all either java 1.8.0 or java 11. But java is must to be run tomcat.

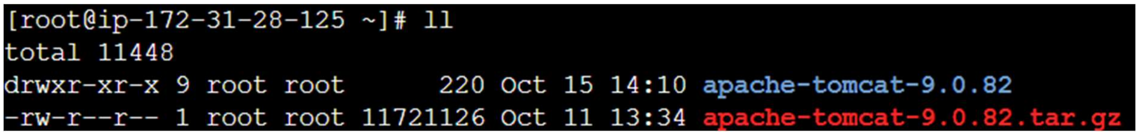
Tomcat is the one of the product of apache. So we can get the tomcat file form the apache website follow by the link **https://dlcdn.apache.org/tomcat/ .** select the tomcat version from there and click on the bin folder finally we will have the **.tar.gz** file, then copy the link address of the tomcat file. Then execute the below command in instance.

wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.82/bin/apache-tomcat-9.0.82.tar.gz

Now the file has been downloaded and need to extract the tar file by follow the below command

tar -zxvf <file_name>.tar.gz (to unzip the tar file)

now you will have a folder and tar file.



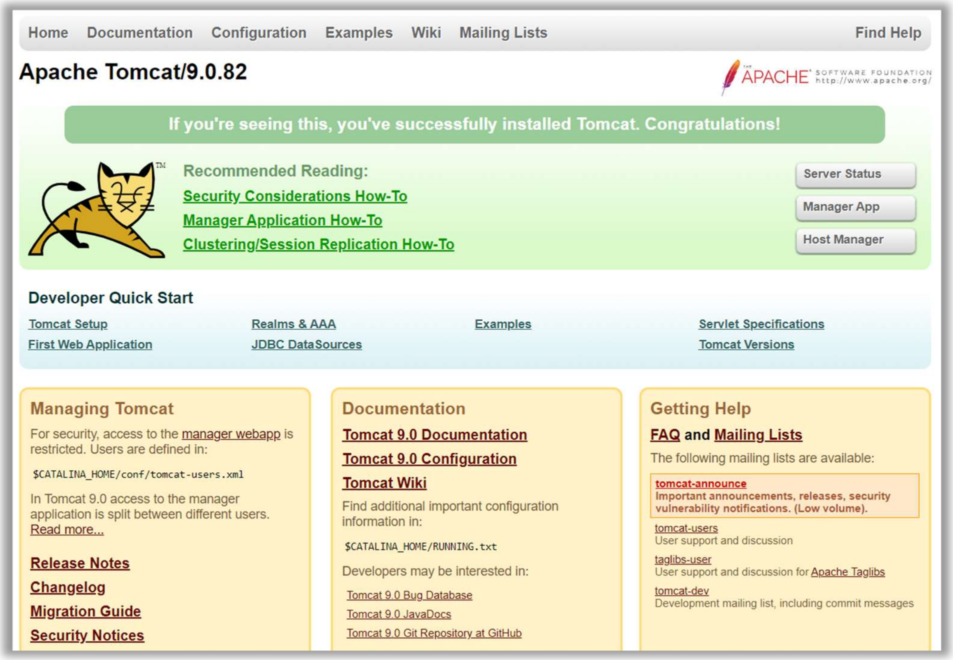
```
[root@ip-172-31-28-125 ~]# ll
total 11448
drwxr-xr-x 9 root root      220 Oct 15 14:10 apache-tomcat-9.0.82
-rw-r--r-- 1 root root 11721126 Oct 11 13:34 apache-tomcat-9.0.82.tar.gz
```

After unzipped the tar file need to start the tomcat by run the startup.sh file. Go to the path

cd apache-tomcat-9.0.82/bin/ then run the startup.sh file by the command **./startup.sh**

the tomcat is started. **<Public IP of the instance>:8080.** Ex , **54.167.91.182:8080**.

The Tomcat will look like the below pic.



Apache Tomcat/9.0.82

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 application is split between different users. Read more...

Release Notes
Changelog
Migration Guide
Security Notices

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:

Tomcat 9.0 Bug Database
Tomcat 9.0 JavaDocs
Tomcat 9.0 Git Repository at GitHub

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

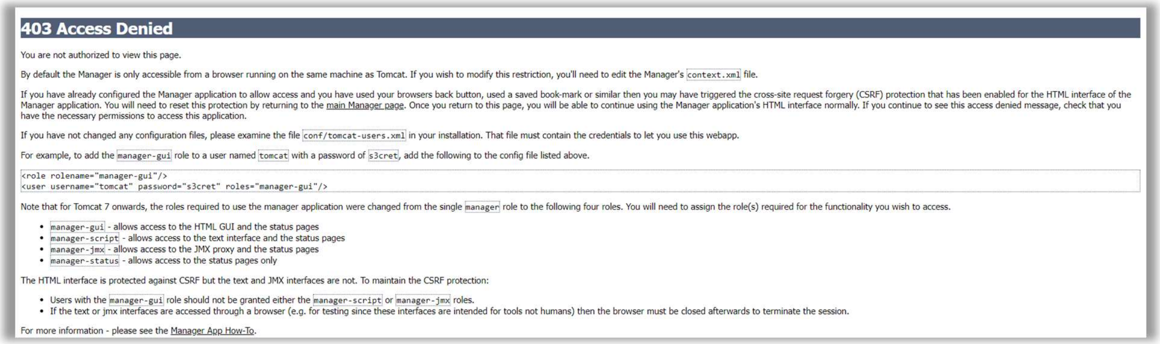
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).

tomcat-users
User support and discussion

taglibs-user
User support and discussion for Apache Taglibs

tomcat-dev
Development mailing list, including commit messages

If we launch the manage app. We will get the below error.



403 Access Denied

You are not authorized to view this page.

By default the Manager is only accessible from a browser running on the same machine as Tomcat. If you wish to modify this restriction, you'll need to edit the Manager's `context.xml` file.

If you have already configured the Manager application to allow access and you have used your browsers back button, used a saved book-mark or similar then you may have triggered the cross-site request forgery (CSRF) protection that has been enabled for the HTML interface of the Manager application. You will need to reset this protection by returning to the main Manager page. Once you return to this page, you will be able to continue using the Manager application's HTML interface normally. If you continue to see this access denied message, check that you have the necessary permissions to access this application.

If you have not changed any configuration files, please examine the file `conf/tomcat-users.xml` in your installation. That file must contain the credentials to let you use this webapp.

For example, to add the `manager-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

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

Note that for Tomcat 7 onwards, the roles required to use the manager application were changed from the single `manager` role to the following four roles. You will need to assign the role(s) required for the functionality you wish to access.

- `manager-gui` - allows access to the HTML GUI and the status pages
- `manager-script` - allows access to the text interface and the status pages
- `manager-jmx` - allows access to the JMX proxy and the status pages
- `manager-status` - allows access to the status pages only

The HTML interface is protected against CSRF but the text and JMX interfaces are not. To maintain the CSRF protection:

- Users with the `manager-gui` role should not be granted either the `manager-script` or `manager-jmx` roles.
- If the text or jmx interfaces are accessed through a browser (e.g. for testing since these interfaces are intended for tools not humans) then the browser must be closed afterwards to terminate the session.

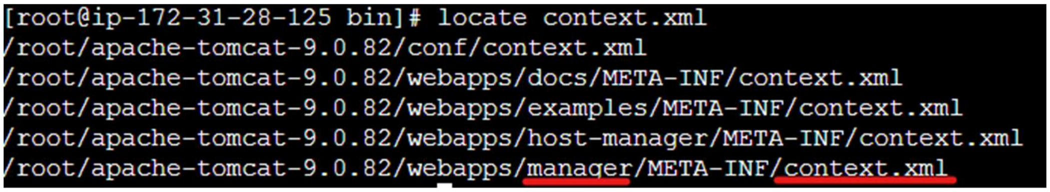
For more information - please see the Manager App How-To.

To resolve it follow the instructions over there.

By default the Manager is only accessible from a browser running on the same machine as Tomcat. If you wish to modify this restriction, you'll need to edit the Manager's `context.xml` file.

To find the context.xml file follow the below commands.

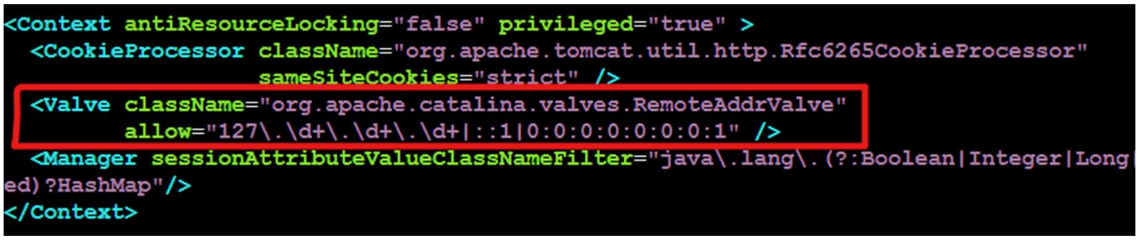
1. We to update the DB before find the file. – **updated**
2. To find the context.xml file – **locate context.xml**



```
[root@ip-172-31-28-125 bin]# locate context.xml
/root/apache-tomcat-9.0.82/conf/context.xml
/root/apache-tomcat-9.0.82/webapps/docs/META-INF/context.xml
/root/apache-tomcat-9.0.82/webapps/examples/META-INF/context.xml
/root/apache-tomcat-9.0.82/webapps/host-manager/META-INF/context.xml
/root/apache-tomcat-9.0.82/webapps/manager/META-INF/context.xml
```

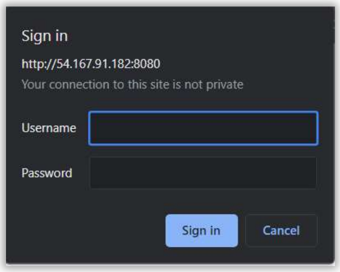
As per the instruction we need to edit the manager’s context.xml file. Open it in vim.

Now remove the following two lines by using vim editor commands.



```
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
                   sameSiteCookies="strict" />
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long
ed)?HashMap"/>
</Context>
```

After delete the lines we’ll get the login popup notification.

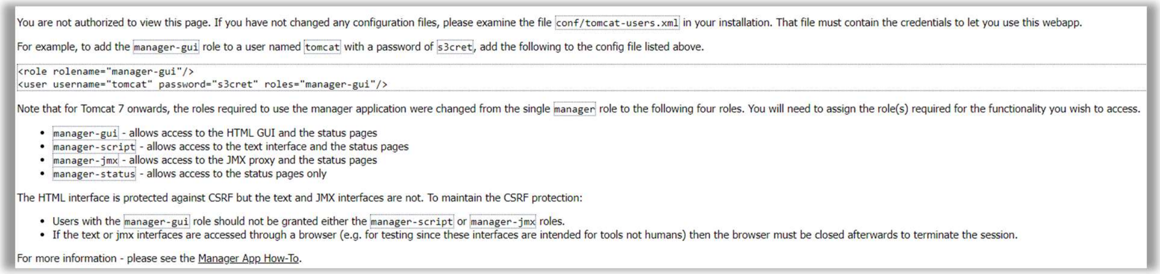


Sign in

http://54.167.91.182:8080

Your connection to this site is not private

Now click on cancel. Again, we will get instruction page to fix it. Because we didn’t set any credential yet. So we need to set the credentials by follow the instruction showing on the page.



You are not authorized to view this page. If you have not changed any configuration files, please examine the file `conf/tomcat-users.xml` in your installation. That file must contain the credentials to let you use this webapp.

For example, to add the `manager-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

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

Note that for Tomcat 7 onwards, the roles required to use the manager application were changed from the single `manager` role to the following four roles. You will need to assign the role(s) required for the functionality you wish to access.

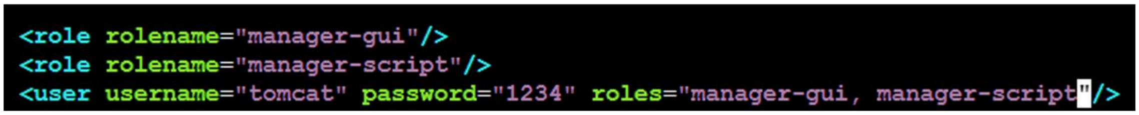
- `manager-gui` - allows access to the HTML GUI and the status pages
- `manager-script` - allows access to the text interface and the status pages
- `manager-jmx` - allows access to the JMX proxy and the status pages
- `manager-status` - allows access to the status pages only

The HTML interface is protected against CSRF but the text and JMX interfaces are not. To maintain the CSRF protection:

- Users with the `manager-gui` role should not be granted either the `manager-script` or `manager-jmx` roles.
- If the text or jmx interfaces are accessed through a browser (e.g. for testing since these interfaces are intended for tools not humans) then the browser must be closed afterwards to terminate the session.

For more information - please see the Manager App How-To.

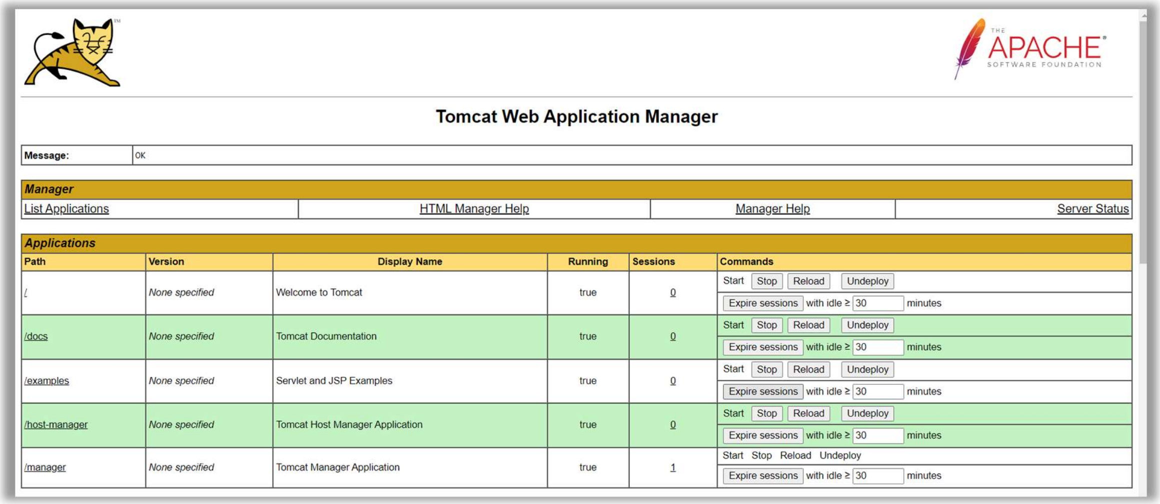
add the below 3 lines in last line of the file.



```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="tomcat" password="1234" roles="manager-gui, manager-script"/>
```

Now save it by using :wq.

After edit we need to restart the tomcat by executing shutdown.sh and startup.sh files.



Tomcat Web Application Manager

Message:	OK

Manager

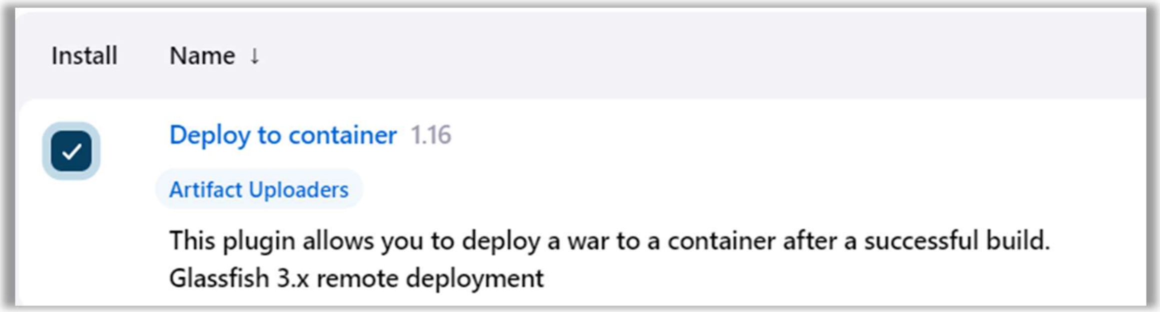
List Applications	HTML Manager Help	Manager Help	Server Status

Applications

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

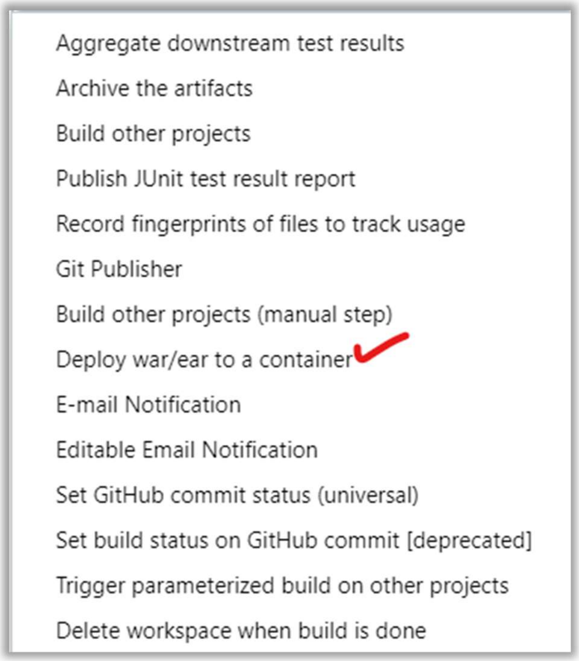
After launched the tomcat application server, we can deploy our source code to the application server.

Now we need to install the **Deploy to container** plugin.

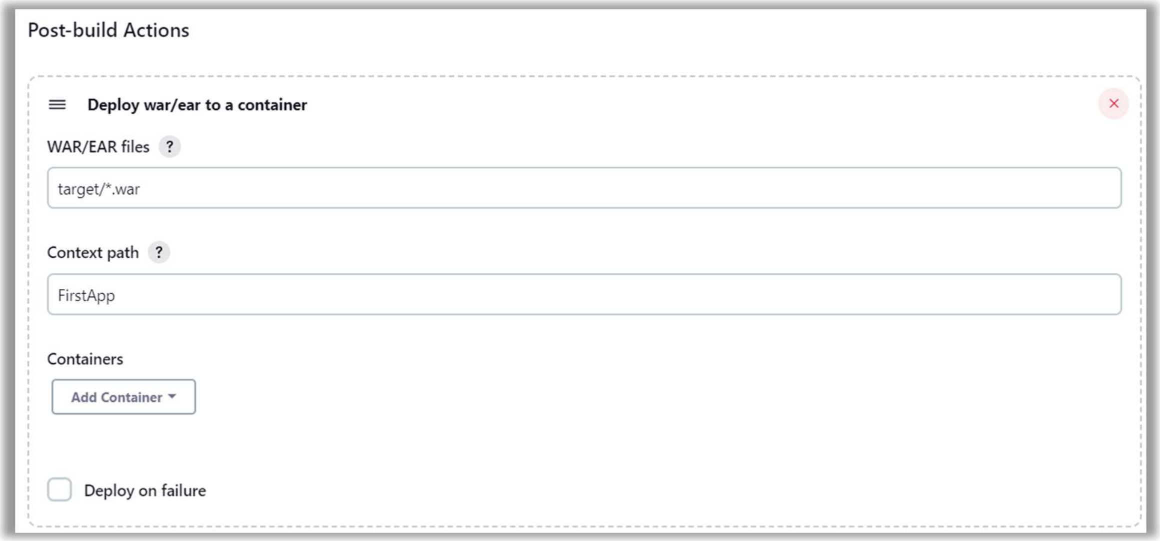


Install	Name ↓
☑	Deploy to container 1.16 Artifact Uploaders This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment

Then open Jenkins and edit the configure of the job. If you install the Deploy to container then only it'll reflect in the **Deploy war/ear to a container** option. Select it.


Aggregate downstream test results
Archive the artifacts
Build other projects
Publish JUnit test result report
Record fingerprints of files to track usage
Git Publisher
Build other projects (manual step)
Deploy war/ear to a container
E-mail Notification
Editable Email Notification
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Trigger parameterized build on other projects
Delete workspace when build is done

We have the war file in **target** folder, right? So, we need to enter the folder name and war file name. we can use ***.war** which is all the files contains .war extension. So, we can simply use the ***.war** format.


Post-build Actions

Deploy war/ear to a container

WAR/EAR files: target/*.war

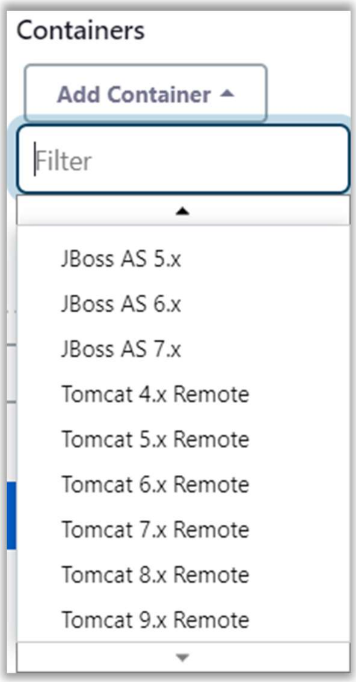
Context path: FirstApp

Containers

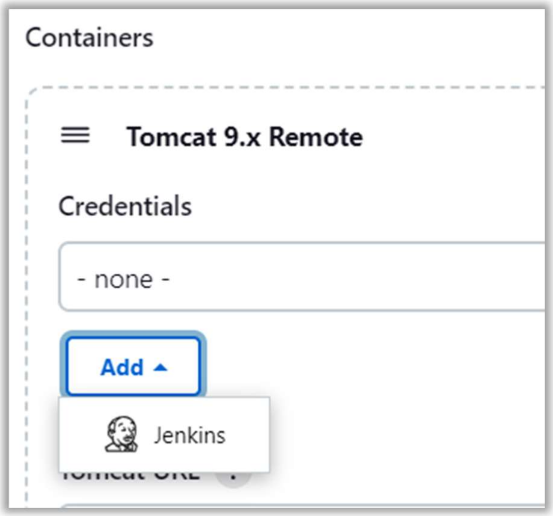
☐ Deploy on failure

We give the details of the folder and file name as well. But we didn't mention the which tool we have to use for deply in it. So we need to declare it in the containers field.

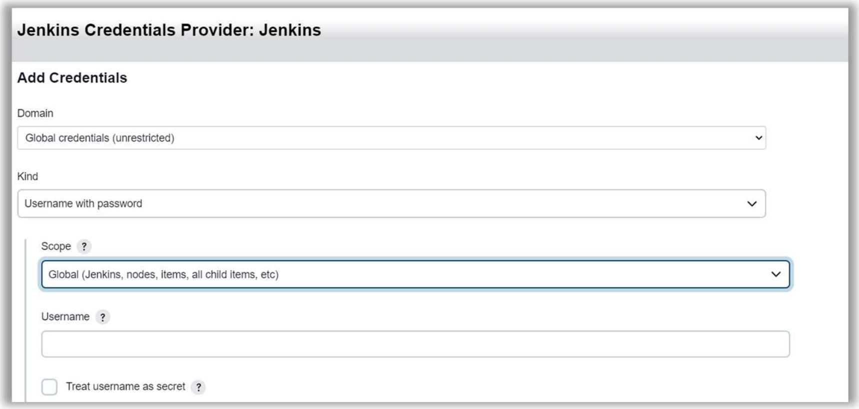
We took the tomcat 9x version so we need to select the tomcat 9x.



After select the Tomcat 9x. you will get the new fields to fill. Here we need to add the tomcat credentials to login this Jenkins.



Here we need to fill the credentials of the Tomcat which username and password has set in the **conf/tomcat-users.xml** file.



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

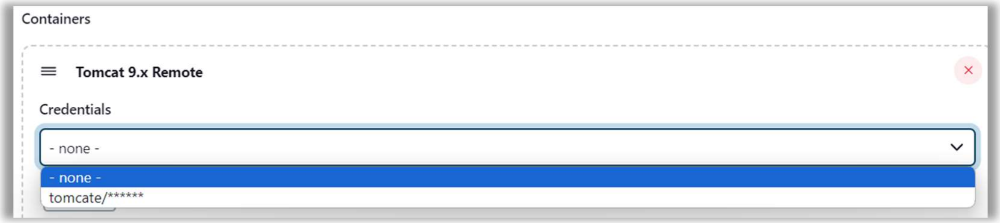
Kind: Username with password

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

Username:

□ Treat username as secret

After put the credentials in this field we can see the credentials as a choice.

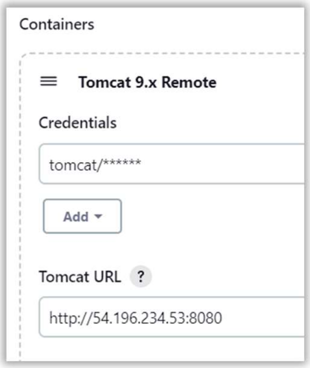


Containers

Tomcat 9.x Remote

Credentials: - none -

Then enter the tomcat URL till 8080/.



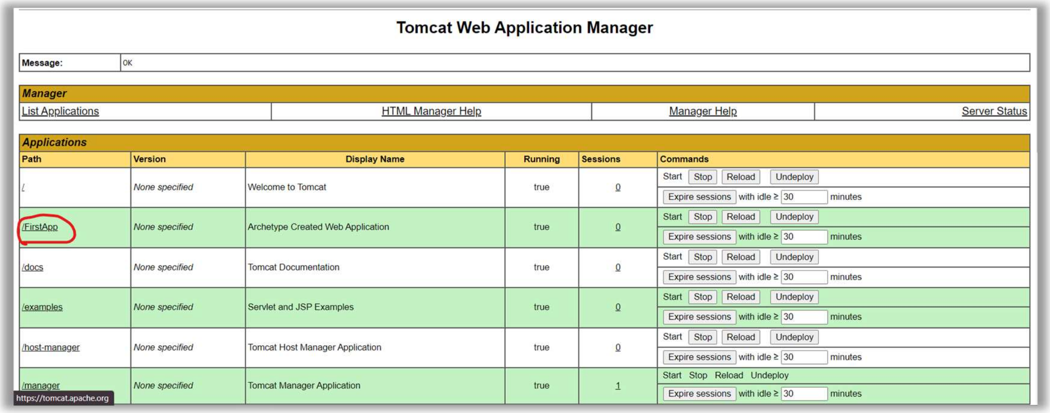
Containers

Tomcat 9.x Remote

Credentials: tomcat/******

Tomcat URL: http://54.196.234.53:8080

After build the job our code has been deployed to the tomcat application server.



Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications

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

Whenever we build the job it will store in Jenkins as well. We can see the logs in the path

/var/lib/jenkins/jobs/<job_name>/builds/<build_number>