Jenkins project

Steps for a project

Part 1 :

* Install a tomcat version using below url:

Wget <https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.76/bin/apache-tomcat-9.0.76.tar.gz>

* Untar apache-tomcat using below command :

tar -xvf apache-tomcat-9.0.76.tar.gz

3. Move to apache folder

Cd apache-tomcat-9.0.76

4. For access a tomcat manager add below codes:

Use this command to add code

vi ./conf/tomcat-users.xml

<role rolename="admin-gui,manager-gui,manager-script,manager-jmx,manager-status,admin-gui"/>

<user username="admin" password="admin" roles="manager-gui,admin-gui,manager-script"/>

<user username="robot" password="admin" roles="manager-script"/>

5. To remote access, remove or comment below two lines:

Use this command to remove or comment code:

vi ./webapps/manager/META-INF/context.xml   
  
<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"  
 allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->

6. Stop and start tomcat :

./bin/shutdown.sh

./bin/startup.sh

7. check tomcat using ip address and port like this

Ip/port - <http://3.141.152.173:8080/>

**Jenkins CI/CD Project for Java Spring Boot Tomcat**

This README file contains the necessary steps for setting up a Jenkins CI/CD pipeline for a Java Spring Boot project using Tomcat as the application server. We will be using the sample project at (<https://github.com/piyushprajapatice/java-code1.git>)

**Part 2: Create a Jenkins Job for Deployment with the Deploy to Container Plugin**

* Go to the Jenkins dashboard and click on 'New Item'.
* Enter a name for your build job and select 'Freestyle project' as the build job type.
* In the configuration page, scroll down to the 'Source Code Management' section and select 'Git'.

git clone <https://github.com/piyushprajapatice/java-code1.git>

* Install and configure the 'Deploy to Container' plugin:
* Go to 'Manage Jenkins' > 'Manage Plugins'.
* Click on the 'Available' tab and search for 'Deploy to container'.
* Select the 'Deploy to container Plugin' checkbox and click on 'Install without restart'.
* Configure the build triggers, build steps, and post-build actions for the deployment job:
* Build Triggers: Choose the desired build trigger, such as 'Build after other projects are built', 'Poll SCM', or 'Build periodically'.
* Build Steps:
* Click on 'Add build step' and select 'Invoke top-level Maven targets'.
* Set the 'Maven Version' and enter 'clean package' in the 'Goals' field.
* Post-build Actions:
* Click on 'Add post-build action' and select 'Deploy war/ear to a container'.
* In the 'WAR/EAR files' field, enter '\*\*/\*.war'.
* Click 'Add Container' and choose 'Tomcat 9.x'.
* Enter the 'Tomcat URL' (e.g., <http://3.141.152.173:8080/>).
* Add the credentials for the Tomcat manager (username and password).
* Set the 'Context path' for the deployed application (e.g., /app).
* Save the configuration and click on 'Build Now' to start the deployment job.
* Check your Application at context name given

<http://3.141.152.173:8080/app>

* Now use the below code to deploy from JenknsFile

pipeline {

agent any

stages {

stage("Checkout") {

steps {

git branch: 'master', url: '<https://github.com/piyushprajapatice/java-code1.git>'

}

}

stage('Maven Clean') {

steps {

sh "mvn clean"

}

}

stage('Maven Build') {

steps {

sh "mvn compile"

}

}

stage("Unit Test") {

steps {

sh "mvn test"

}

}

stage("Maven Package") {

steps {

sh "mvn package"

}

}

stage('deploy to tomcat') {

steps {

deploy adapters: [tomcat9(credentialsId: 'admin', path: '', url: '<http://13.58.15.193:8080/>')], contextPath: 'app', onFailure: false, war: 'target/\*\*.war' }

}

}

}