**Simple DevOps project -1 for CI/CD**

**INSTALLING TOMCAT SERVER**

1. Creating an aws instance with red hat linux and establishing a connection with putty using **.ppk** file.
2. Using **sudo su –** command to become root user and installing java using **Yum install java-1.8\*** command.
3. Now installing **apache tomcat server** using **wget** command in the new directory named **opt.**
4. Extracting **tomcat apache server** zip file using **tar -zvxf** **filename** command.
5. Now we are moving into the **apache-tomcat-8.5.79** directory and from this directory we are moving into the **bin** directory.
6. In the bin directory , we will be able to see the **startup.sh** and **shutdown.sh** and we are making them executable using **chmod +x filename** command.
7. Now we are creating a softlinks to start and stop the tomcat server, using the command **ln -s /opt/apache-tomcat-8.5.79/bin/startup.sh /usr/local/bin/tomcatup and ln -s /opt/apache-tomcat-8.5.79/bin/shutdown.sh /usr/local/bin/tomcatdown(**After executing this commands we can just give tomcatup and tomcatdown to start and stop the server**)**
8. Then add the port number **8080** to the **inbound rules** which is available under **security groups**.
9. After adding the port number, open a webpage and insert the public ip address of the instance followed by the port number to check whether the tomcat page is opening. Eg: (**13.232.54.60:8080)**
10. To change the port from 8080 to 8090 go to the conf file and inside that edit the server.xml file using the command **vi server.xml.**
11. While editing change the port inside the server.xml file from 8080 to 8090 and save it using **:wq** . Asusual add the port 8090 in the inbound rules and give (**13.232.54.60:8090)** and check whether tomcat page is getting opened.
12. To access the sub modules inside the tomcat page, we need to edit the context.xml file as (<!- -and  -- >) for values that are available in webapps like **/opt/apache-tomcat-8.5.79/webapps/host-manager/META-INF/context.xml** and **/opt/apache-tomcat-8.5.79/webapps/manager/META-INF/context.xml.**
13. Then you need to add the username and password to access the **manager app page.**
14. To add username inside the conf folder edit the **tomcat-users.xml** file with the code like

**<role rolename="manager-gui"/>**

**<role rolename="manager-script"/>**

**<role rolename="manager-jmx"/>**

**<role rolename="manager-status"/>**

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

**<user username="deployer" password="deployer" roles="manager-script"/>**

**<user username="tomcat" password="s3cret" roles="manager-gui"/>**

Then save the **tomcat-users.xml**using **:wq** command.

1. After this check whether the **manager app page** is opening with the login credentials

**USERNAME: tomcat**

**PASSWORD: s3cret .**

**INSTALLING JENKINS SERVER**

1. Creating a new instance on AWS as **Jenkins Server.**
2. Using **sudo su –** command to become root user and installing java using **Yum install java-1.8\*** command.
3. Then check whether java is installed in the system by using **java -version.**
4. Now install Jenkins by using the following commands one by one

**yum -y install wget**

Then, **wget -O /etc/yum.repos.d/jenkins.repo** [**https://pkg.jenkins.io/redhat-stable/jenkins.repo**](https://pkg.jenkins.io/redhat-stable/jenkins.repo)

Then, **rpm --import** [**https://pkg.jenkins.io/redhat-stable/jenkins.io.key**](https://pkg.jenkins.io/redhat-stable/jenkins.io.key)

Then, **yum -y install Jenkins**

1. To start Jenkins service use the command

**systemctl start Jenkins.**

1. To Setup Jenkins to start at boot

**systemctl enable Jenkins**

1. **Accessing Jenkins**

By default jenkins runs at port 8080, You can access jenkins at

[**http://YOUR-SERVER-PUBLIC-IP:8080**](http://YOUR-SERVER-PUBLIC-IP:8080)

1. **Configuring Jenkins**

By creating new user admin id with

-username

-password

-name

-email address

1. Must download the necessary plugin which are suggested at the beginning of the installation.

1. We must now test the Jenkins Job by the following steps
2. **Create “new item”**
3. **Enter an item name – My-First-Project**
   * **Chose Freestyle project**
4. **Under Build section Execute shell : echo "Welcome to Jenkins Demo"**
5. **Save your job**
6. **Build job**
7. **Check "console output"**

**CI/CD through JENKINS**

1. First login to the AWS account and restart the jenkins server and Tomcat server (Web Server and Jenkins\_Server).
2. Using the IPV4 address of the instances connect them using putty.
3. Using the localhost addresses 13.233.114.89 (Jenkins-Server) and 15.206.167.215(Web-Server) and start the Jenkins and tomcat server inside the server like tomcatup and systemctl start Jenkins. Then, check if the server is running or not by using ps -ef | grep tomcat and systemctl status Jenkins(Localhost address may vary).
4. Login to the Jenkins webpage by giving the username and password to get into the Jenkins dashboard.
5. Before creating the job, install the needed plugins like maven integration, deploy to container and some plugins needed for git and maven.
6. Now create a Jenkins job in the name of **hello maven** and select the option maven project then click ok.
7. Now must configure the following:

1. SOURCE CODE MANAGEMENT: Select Git

2. GOALS AND OPTIONS: Give clean install package

3. POLL SCM: Give it as \*/1 \* \* \* \* (Continuous integration)

NOTE: [ \*/1 \* \* \* \*] it will build automatically after a min when the code is changed.

1. POST BUILD ACTION: Click on add Post-Build action and select the option Deploy war/ear to a container
2. Now apply and save the **hello maven** job.
3. As we have set the Poll SCM, edit the code file and commit the changes then it will build automatically after a min
4. Then check whether the build is FINISHED: SUCCESS and now give the tomcat IPV4 address like **http://15.206.167.215:8090/webapp**/
5. Finally check the webpage whether the code you have changed is reflecting as the desired output in the address **http://15.206.167.215:8090/webapp/**