Lab: Continuous Deployment: Web-App deployment to Tomcat server using Deployment to Container plugin

Pre-Requisites: -

1. Jenkins server should be set up. If already set up, take the IP dedicated for your group. If not set, please coordinate with trainer for the same. Use the credentials provided by Trainer for login into Jenkins
2. Tomcat server should be set up. This will be our Deployment Server. After installing Tomcat on ubuntu, some additional settings are required to be done for working with Jenkins. To know the configuration done on Tomcat Server level, follow instructions mentioned in section **Configuring Tomcat to support deployment through Jenkins**.

3. Git Repository with the code to be deployed should be available. The git repo to be used in training: https://github.com/LovesCloud/java-tomcat-maven-example Fork the repo in your account.

4. Jenkins should already be configured to support following features and tools: -
   a. GitHub (Steps used mentioned in **Configure Jenkins for GitHub**
   b. Maven (Steps used mentioned in **Configure Jenkins for building Maven Projects)**
5. GitHub should be configured to support Jenkins. Steps used mentioned in **Configure GitHub Webhook for Jenkins**
6. Jenkins already have following plugins installed: -
   a. **GitHub Integration:** This provides configurations fields required for Jenkins and GitHub Integration.
   b. **Maven Integration Plugin:** This plugin provides integration between Jenkins and Maven and help in automatic building projects.
   c. **Deploy to container plugin:** This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment
   d. **Parameterized Trigger:** This will give option in Post build actions for triggering parameterized build.

Note: To know how to install plugin, follow instructions mentioned in document **Installing Plugins in Jenkins**

Step A: *Create a job for building the project*

1. Click on **New Item**
2. Enter **Name** such as <yourname>_buildjob
3. Select **Maven project**
4. Click **OK**
5. Under Source Code Management section, Select **Git** radio button
   a. Enter Repository URL - <Git repo>
      (Note: Git Repository URL is the one forked as part of prerequisites)

6. Under Build Triggers section, Select **GitHub hook trigger for GITScm polling** checkbox
7. Under Build section
    a. Enter **Root POM**-pom.xml
    b. Enter **Goals and options**-clean package
8. Under Post Steps, Run only if build succeeds
9. Under Post-build Actions, Select **Add post-build action**- Editable Email Notification;
    a. In **Project Recipients List** field, add comma and <email id where you want to send notification>
10. Click **Save**

Step B: *Create a job for deploying the application on Tomcat Server*

1. In Jenkins, Click on **New Item**
2. Enter **Name** such as <yourname_deployjob>
3. Select **Freestyle project**
4. Click **OK**
5. Under General, Select **This project is parameterized** checkbox
    a. Select **Add Parameter**-String Parameter
    b. Enter **Name-**DEPLOY_VERSION
    c. Enter **Default value**-0
    d. Enter **Description**-To deploy latest war file built in build job
    e. Select **Trim the string** checkbox
6. Under Build section, Select **Add Build Step**- Execute Shell and paste the following script:
   cp /var/lib/jenkins/jobs/<yourname>_buildjob/builds/$DEPLOY_VERSION/com.example\$java-tomcat-maven-example/archive/com.example/java-tomcat-maven-example/1.0-SNAPSHOT/java-tomcat-maven-example-1.0-SNAPSHOT.war /var/lib/jenkins/workspace/<yourname>_deployjob/

Note: In the above script, you have to replace job names with your build and deploy job names respectively.

7. Under Post-build Actions, Select **Add post-build action**- Deploy war/ear to a container.
    a. Enter WAR/EAR files: **/*.war
    b. Context path: java-tomcat-maven-example_<yourname>
    c. **Containers** field:
        i. Click **Add Containers** dropdown : Select Tomcat 8.x
        ii. **Credentials**: select tomcat user;
            Note: For Lab, this is already configured in jenkins and can be selected for use. New user can be added using Add button. New user if created in jenkins, has to be added in tomcat server also.
        iii. Enter **Tomcat URL**: http://<Deployment Server IP>:9090
8. Under Post-build Actions, Select **Add post-build action**- Editable Email Notification;

        a. In **Project Recipients List** field, add comma and <email id where you want to send notification>

    9. Click **Save**

Step C: *Modify the build job*

1. Open the build job created in Step A.
2. Under Post-build Actions, Select **Add post-build action**- Trigger parameterized build on other projects
    a. Enter **Build Triggers->Projects to build**-<Name of the Deploy job>
    b. Select **Trigger when build is-**Stable
    c. Select **Add Parameters**-Predefined parameters
        i. Enter **Parameters-**DEPLOY_VERSION=${BUILD_NUMBER}
3. Click **Save**

Step D: *Running the build job*

1. Open the Job created in Step A
2. Click **Build Now**
3. Follow the same steps to see the log as we did in previous labs.
4. Click on the job name (triggered after the successful completion of build job) present at the end of the page;
5. View the Console Output of the latest build executed in this deploy job

It should display the Finished status as Success

Step E: *Verifying the deployed application*

1. Open any browser
2. http://<Deployment Server IP>:9090/<Context path>

Note: the <Context path> was set in the deploy job, please take from there.

3. Hit Enter;

Configure Jenkins for GitHub

1. Navigate to Manage Jenkins ->Open Configure System
2. In GitHub section, Select Add GitHub Server->GitHub Server
    a. Enter **Name**-Git Server
    b. Click on the Advanced… button present below the Add GitHub Server drop down
    c. Select Additional actions->Manage additional github actions->Select Convert login and password to token
    d. Select **from login and password** radio button
    e. Enter your GitHub login credentials
    f. Click **Create token credentials** button
    g. Scroll up and select the generated token in the **Credentials** field
3. Click **Save** on Configure System

Configure Jenkins for building Maven projects

1. Navigate to Manage Jenkins
2. Click Global Tool Configuration
3. In Section Maven,
   a. Click **Add Maven** button
   b. Enter **Name** - Maven 3.5
   c. Install from Apache, Select **Version** 3.5.4
4. Click **Save**

Configure GitHub Webhook for Jenkins

1. Open GitHub
2. Navigate to Git Repo of Frontend code;
3. Navigate to Settings of repository
4. Click Webhook
5. Click Add Webhook
6. Enter Payload URL-http://<Public IP of Jenkins Server>:8080/github-webhook/
7. Click Save Webhook

Configuring Tomcat to support deployment through Jenkins

1. Ssh the tomcat server;
2. Type command: sudo su
3. Type command: cd /opt/tomcat/conf
4. Type command: vim tomcat-users.xml
5. Press i
6. Use down arrow till second last line in the file
7. Add the code
   ```
   <role rolename="manager-script"/>
   <user username="tomcat" password="tomcat" roles="manager-script"/>
   ```
8. Press escape
9. Press :wq!
10. Type command: cd /opt/tomcat/webapps/manager/META-INF
11. Type command: vim context.xml
12. Press i
13. Use down arrow and take the cursor to the Value section;
14. comment Valve section by prefixing the code with “<!--” and suffixing the code with “-->”

For Eg:-

```
<Context antiResourceLocking="false" privileged="true" >
<!--
  <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|Number|String)|org\.apache\.catalina\.filters\.CsrfPreve
ntionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
```

15. Press escape
16. Press :wq!
17. To change the port of Tomcat Server.
18. Type command: cd /opt/tomcat/conf
19. Type command: vim server.xml
20. Press i
21. Replace the port 8080 with 9090
22. Press escape
23. Press :wq!