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

Pre-Requisites: -

1. Git Repository with the code to be deployed should be available. The git repo to be used in training: https://github.com/LovesCloud/Javatomcatmavenapplicationdemo Please fork the repo in your GitHub account.

2. Configure GitHub Webhook for Jenkins on your forked repo. follow the steps mentioned in **Configure GitHub Webhook for Jenkins**
3. Deployment Server IP is with Trainer, please ask for it whenever required in Lab.

Steps To Follow:

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, Select 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, Copy and paste the script from the below URL:
https://pastebin.com/raw/6VsU44an

Note 1: In the script, you have to replace BUILDJOBNAME with your build job and DEPLOYJOBNAME with your deploy job name respectively.
Note 2: The format of the cp command is as follows: cp SOURCE space DESTINATION
If you notice any extra space present within SOURCE, please remove it.

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;
      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 for the first time manually*

1. Open the Job created in Step A
2. On Left Navigation Panel, Click on **Build now.**
3. Observe the **#1 build triggered under Build History section in Left Panel.**
4. Follow the same steps to see the log as we did in previous labs.
5. Click on the job name (triggered after the successful completion of build job) present at the end of the page;
6. 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 in **Step B->7->b**, please take from there.

3. Hit Enter;

Step F: *Triggering the Deployment automatically.*

1. On GitHub end, as you are the owner of the new forked git repo, edit any file and commit the changes.
2. Observe the new triggered build.
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
6. It should display the Finished status as Success
7. Verify the deployed application by following the steps mention in **Step D.**

Configure GitHub Webhook for Jenkins

1. Open GitHub
2. Navigate to Git Repo;
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