Jenkins

Continuous Delivery

> Continuous Delivery (CD) is a DevOps practice that is used to deploy an application quickly while maintaining a high quality with an automated approach. It is about the way application package is deployed in the Web Server or in the Application Server in environment such as dev, test or staging. Deployment of an application can be done using shell script, batch file, or plugins available in Jenkins. Approach of automated deployment in case of Continuous Delivery and Continuous Deployment will be always same most of the time. In the case of Continuous Delivery, the application package is always production ready

Jenkins with Maven Build

Setup maven

- Go to `Jenkins Dashboard` -> `Manage Jenkins` -> `Global Tool Configuration` > `Maven` > Give a Name `Maven_Local` > Check `Install Automatically` > Install from Apache (specify a version) > `Save`
- You can give a logical name to identify the correct version while configuring a build job

Jenkins Job

- Click on **New Item** then enter an item name, select **Freestyle project**.
- Under Source Code Management tab, select Git and then set the Repository URL to point to your GitHub Repository. `https://github.com/YourUserName/repo-name.git`
- Under Build Environment Build Step > Select `Invoke top-level Maven targets` from dropdown > select the Maven Version that we just created, specify `clean install`.
- Under Advanced tab, specifiy the pom.xml file relative path location from git repository.
- Click on `Save`

> it will run command `mvn clean install -f pom.xml`

- Click OK and Build a Job and you will see that a `war file` is created.

> `clean` -> Deletes `/var/lib/jenkins/workspace/jenkins-maven-build/java-tomcat-sample/target`

Maven build phases

- Maven itself requires Java installed on your machine.
- You can verify if Maven is installed on your machine by running **mvn -v** in your command line/terminal.
- Maven is based on the `Project Object Model (POM)` configuration, which is stored in the XML file called the same – `pom.xml`.
- It is a structured format that describes the project, it's dependencies, plugins, and goals. `pom.xml` file in your project directory
- **Validate** : Validate Project is correct & all necessary information is available.
- **Compile** : Compile the Source Code
- **Test** : Test the Compiled Source Code using suitable unit Testing Framework (like JUnit)
- **package** : Take the compiled code and package it.
- **Install** : Install package in Local Repo, for use as a dependency in other project locally.
- **Deploy** : Copy the final package to the remote repository for sharing with other developers.
- The above are always are sequential, if you specify `install`, all the phases before `install` are checked.

Artifacts Archive

- Go to `Jenkins dashboard` -> `Jenkins project or build job` -> `Post-build Actions` -> `Add post-build action` -> `Archive the artifacts`:
- Enter details for options in `Archive the artifacts` section:
    - For `Files to archive` enter the Path of the `.war` file like : `java-tomcat-sample/target/*.war`
- `Save` the changes and `Build Now`.
- Check the directories as below to validate above information:

```
ls /var/lib/jenkins/jobs
ls /var/lib/jenkins/jobs/<JOB_NAME>
ls /var/lib/jenkins/jobs/<JOB_NAME>/builds/<BUILD_NUMBER>
ls /var/lib/jenkins/workspace/<JOB_NAME>
```

- If you check the directory structure, there will be `archive` directory present under the subsequent build number for which the job is executed with Post build action as `Archive the artifacts`

Jenkins Build and Deploy

- Below steps assume that, you have a Jenkins Server Up and Running on one of the EC2 instance.

Setup Apache Tomcat on Amazon Linux:

- Launch a new EC2 Instance for Webserver Configuration

```
sudo hostnamectl set-hostname tomcat.example.com
sudo yum install java-1.8.0 -y
cd /opt/
sudo wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.35/bin/apache-
tomcat-9.0.35.tar.gz
sudo tar -zvxf apache-tomcat-9.0.35.tar.gz
----------------------------
z – The file is a “gzipped” file
v – Verbose, print the file names as they are extracted one by one
x –  Extract files
f – Use the following tar archive for the operation
----------------------------
sudo ls -ltr /opt/apache-tomcat-9.0.35/bin
sudo cd /opt/apache-tomcat-9.0.35/bin
```

- To Start Apache Tomcat : Run the `./startup.sh` file in `/opt/apache-tomcat-9.0.35/bin`
- We can make the scripts executable and then create a symbolic link for this scripts.

```
sudo chmod +x /opt/apache-tomcat-9.0.35/bin/startup.sh
sudo chmod +x /opt/apache-tomcat-9.0.35/bin/shutdown.sh
```

- Create symbolic link to these file so that tomcat server start and stop can be executed from any directory.

```
echo $PATH
sudo ln -s /opt/apache-tomcat-9.0.35/bin/startup.sh /usr/bin/tomcatup
sudo ln -s /opt/apache-tomcat-9.0.35/bin/shutdown.sh /usr/bin/tomcatdown
tomcatup
netstat -nltp | grep 8080
```

> If you want to run Apache Tomcat on same Machine where Jenkins is Installed, then change the port of Apache Tomcat in : `/opt/apache-tomcat-9.0.35/conf/server.xml` file to `8090` as below,

```

```

```
<Connector port="8090" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

- If above changes are made, execute the command `tomcatdown` and `tomcatup`.
- Create an empty repo and clone it, add project files into the local git folder and commit -> push the local repo to remote github repo using Git Bash.
- Verify the files are available in your github repository

Tomcat War file deployment Configs

- To have access to the dashboard the admin user needs the manager-gui role. Later, we will need to deploy a WAR file using Maven, for this, we need the `manager-script` role too.
- In order for Tomcat to accept remote deployments, we have to add a user with the role `manager-script. To do so, edit the file` ../conf/tomcat-users.xml` and add the following lines:
- In this case : add below configuration in file `/opt/apache-tomcat-9.0.35/conf/tomcat-users.xml`

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

- Edit the RemoteAddrValve under this file `/opt/apache-tomcat-9.0.35/webapps/manager/META-INF/context.xml` to allow all.
- `Before`

```
        <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
```

- `After`

```
        <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow=".*" />
```

- Restart the tomcat server using `tomcatdown` and `tomcatup`

Jenkins Plugin installation

- To install the Plugin `Deploy to container` navigate to `Manage Jenkins` > `Manage Plugins`, search `Deploy to container` under `Available` tab.

Jenkins Job to deploy war file

- Click on **New Item** then enter an item name, select **Freestyle project**.
- Select the GitHub project checkbox and set the Project URL to point to your GitHub Repository. `https://github.com/YourUserName/`
- Under Source Code Management Section : Provide the Github Repository URL of the Maven Project, keep the branch as `master`.
- Go to Jenkins Project -> Configure -> Under Build Environment Build Step > Select `Invoke top-level Maven targets` from dropdown > select the `Maven Version` that is configured > Enter `clean install`
- Under `Post-build Actions`, from the `Add post-build action` dropdown button select the option `Deploy war/ear to a container`
- Enter details of the War file that will be created as:
    - For `WAR/EAR files` you can use wild cards, e.g. `**/*.war`.
    - The `context path` is the context path part of the URL under which your application will be published in Tomcat.
    - Select the appropriate Tomcat version from the Container dropdown box (note that you can also deploy to Glassfish or JBoss using this Jenkins plugin).
    - Under the `Credentials`, `Add` username and password value that is entered in the `tomcat-users.xml` file. Specify the ID of the credentials as `tomcat_creds`. This will be used later in Pipeline Script.
    - The Tomcat URL is the base URL through which your Tomcat instance can be reached (e.g http://172.31.67.85:8080)

    > Make Sure network is open on specific port.

    - `Save` the changes and `Build Now`.
    - Once Jenkins Job is build, if there is a Success for deploy, verify the deployment files on Tomcat Server under `webapps` path.
    - Make some changes in the code on the github configured branch in the Jenkins Job and build the Job again to verify the Artifact Deployment on Tomcat Path.

Jenkins Job with docker agent

```
pipeline {
    agent { docker { image 'python:3.8' } }
    stages {
        stage('build') {
            steps {
                sh 'python --version'
                sh 'echo Hello Jenkins'
            }
        }
    }
}
```

- Since above `Jenkinsfile` contains a stage with docker agent, we need to install docker on the Jenkins Node.

```
#install docker
sudo yum install docker -y
sudo systemctl start docker

#add jenkins user to docker and wheel group
sudo usermod -aG wheel jenkins
sudo usermod -aG docker jenkins

#Restart jenkins
sudo systemctl restart jenkins
```

- Also under Jenkins Plugins install : `Docker plugin` and `Docker Pipeline`.
- Click on **Build Now** to build the jenkinsfile project