Jenkins

Jenkins is popularly know from **Continuous Integration** (**CI**) tool. However we can automate any kind of task using jenkins.
Jenkins is open source, i.e. free of cost.
Jenkins is web based tool implemented in java.
Previously jenkins was called as **Hudson,** initially this tool was maintained by Oracle, few developers had differences with Oracle and they came out and continued this tool with the name Jenkins.

What is **Continuous Integration** (**CI**) ?

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day.
Each check-in is then verified by an automated build, allowing teams to detect problems early.
By integrating regularly, fixing defects are easy and quick.

Install Jenkins

Prerequisite: Jenkins requires jdk8

1. Install jdk8
   sudo yum install java-1.8.0-openjdk-devel -y

   If your machine has multiple versions , we need to update default version to java8.

   ```
   sudo update-alternatives --config java
   ```

   ```
   Choose the number beside java8
   ```

   ```
   sudo update-alternatives --config javac
   ```

   ```
   Choose the number beside java8
   ```

2. Install Jenkins

```
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo
```

```
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
```

```
sudo yum install jenkins -y
```

```
sudo service jenkins start
```

Enable jenkins on reboot (jenkins auto starts when we reboot VM)

```
Sudo chkconfig jenkins on
```

Configure Jenkins

Before using jenkins we need to configure it by installing plugins & creating one admin user

1. Access jenkins via web browser
   http://ip-address:8080
2. Unlock jenkins

   a. Take initial admin password
      sudo cat `/var/lib/jenkins/secrets/initialAdminPassword`
   b. Enter initial admin password and click continue
3. Choose install suggested plugins
4. Create First Admin User, fill the details and click **Save & Finish**
5. Click start using jenkins

Jenkins Examples

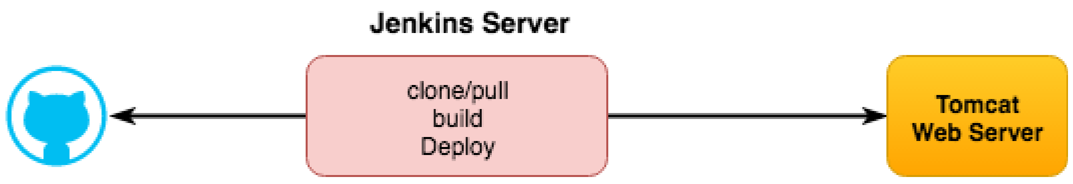
1. Create Jenkins job to clone the project and build a war file.

In jenkins, Job represents a task

- Click new item
- Enter item name as “**FirstJob**”
- Select **Freestyle project** and click **Ok**
- Under source code management select git
- Enter git URL **https://github.com/javahometech/myweb**
- Under build section add “top level maven targets”
- Under Goals enter ***clean package***
- Save the job.
- Run the job by clicking **Build Now**

2. Create Jenkins job to build a war file and deploy to tomcat server.

For deployments we are going to use jenkins plugin, this plugin deploys the war using tomcat's manager application.
Note: Tomcat manager application must be enabled



- Install Deploy to container plugin
    - Manage Jenkins → Manage Plugins → Available →
    - Search for Deploy to container Plugin and select → install without restart

  - Go back to jenkins home
- Click new item
- Enter item name as “**BuildAndDeploy**”
- Select **Freestyle project** and click **Ok**
- Under source code management select git
- Enter git URL https://github.com/javahometech/myweb
- Under build section add “top level maven targets”
- Under Goals enter ***clean package***
- Under post build actions
  - Select deploy war/ear to the container
  - War/Ear file → target/myweb.war
  - Context Path → javahome
  - Add container → tomcat8
- Credentials
  - Add → jenkins → select username with password
  - Enter tomcat managers username and password
  - For Id field enter relevant value for example tomcat-manager and ADD
- Select the credentials
- Add tomcat url for example (***http://172.31.2.228:8080/***)
- Save the job
- Run the job by clicking **Build Now**

3. Trigger the above job automatically if there are new changes in git

- Select the job “**BuildAndDeploy**” → click configure
- Select build triggers
  - Select GitHub hook trigger for GITScm polling, this tells jenkins if changes found in git then trigger this job
  - Select Poll SCM, this is the schedule give to jenkins to check for updates in SCM
  - Under schedule put five stars → * * * * *
  - Save the job
  - To test this job do some changes in git, it automatically triggers this job

4. Send automated email to the dev team if build fails

a. In Order to send and receive emails we need to configure email servers in jenkins.
   Companies has their own email servers, we have to use those servers to trigger Emails, in our example we do not have our own server, so we are going to use Gmails SMTP server.
b. Configure gmail SMTP server in jenkins
   i. jenkins → Manage Jenkins → Configure System
   ii. Under **E-mail Notification**

1. SMTP server → **smtp.gmail.com**
2. Select Use SMTP Authentication
3. Put your gmail id
4. Put your gmail password
5. Use SSL select
6. SMTP port 465
7. Save the configuration

c. Configure jenkins job to trigger email if build fails
   i. Open your job → Configure → Post Build Actions → Add post-build action → email notification
   ii. Under Recipients put teams mail id and save the configuration

Discarding Old builds

Discarding old builds are important to free up some space on the hard disk

- Select Job → Configure → select discard old builds
- Enter Days to keep builds as 5
- Max # of builds to keep 5
- Save the configuration

Archiving artifacts

When build is triggered current artifact is overwritten with previous artifact, in order to retain previous build artifacts we have to enable archiving artifacts.
Open your job → Configure → Post Build Actions → Add post-build actions → archive artifacts
Files to archive → target/myweb.war

Walking through jenkins directory structure

ssh into the jenkins server
cd /var/lib/jenkins
Folders under jenkins installation

Workspace: This is the folder where jenkins keeps the project specific files
Job: This contains configuration details of the job
Plugins: This folder contains all installed plugins
Nodes: the information about jenkins slaves is kept under this folder
Users: The information about the jenkins users is kept under this folder
userContent: The files kept under this folder is served via http

Managing users in jenkins

This allows us to add/modify/delete users in jenkins
Adding new user: jenkins → Manage Jenkins → Manage Users → Create User
Fill the user details and click create user.

Configuring granular permissions for jenkins users

Jenkins has matrix based security option to configure granular permissions to the users in the jenkins
Jenkins → Manage Jenkins → Configure Global Security → under Authorization select matrix-based security.

Security Realm

This all about where jenkins maintains user credentials and their roles
1. Jenkins own database (default option)
2. LDAP (Lightweight Directory Access Protocol)
   Maintaining user credentials and their roles in LDAP give more performance than maintaining in DB.
   By integrating jenkins with LDAP we can grant access to the users already present in LDAP.
3. Unix user/group database

Master Slave Configuration

Master slave in jenkins is similar to load balancing, we use this for the following use cases.
- Improve Jenkins performance
- We can also use separate slave for different workload type, for example
  Slave-1 → Java
  Slave-2 → .Net
  Slave-3 → Terraform
  Slave-4 → Chef

Master is the machine on which we installed and configure jenkins
Slave is a machine added under the master, Slaves are controlled by master based on the job configuration.

Adding a slave to the master

- Jenkins → Manage Jenkins → Manage Nodes → New Node

- Node Name → Slave-1
- Select Permanent Agent and click OK
- # of executors → 1 (*one executor represents one thread, to run multiple jobs concurrently specify bigger value*)
- Remote root directory → /users/ec2-user/jenkins (*An agent needs to have a directory dedicated to Jenkins. Specify the path to this directory on the agent*)
- Labels → “JavaJobs”, labels are used to link a job to run on a specific slave
- Usage
    - Use as much as possible
    - Only build jobs with label expressions matching this job (*We are using this option for this demo*)
- Launch Method, Launch slave via ssh
- Host → IP address of the slave
- Create new credentials for ssh into slave, and select this credentials
- Host Key Verification Strategy → Non verifying verification strategy
- Availability → keep this agent online as much as possible and click save

Note: On slave java must exists, apart from this, for example if our job requires git, maven etc we also need to install those tools as well.

Jenkins Pipeline jobs (Jenkins 2.x)

Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
These jobs use Jenkins DSL as a syntax and groovy is the scripting language

Create Pipeline Job

- Jenkins → New item → Name = “Pipeline-Demo”
- Select Pipeline and click Ok

SonarQube

SonarQube is an open source automated static code analysis tool, It is an web application, which takes our code as an input, and produces the quality report as an output.
On this tool we also can publish Junit test coverage reports.
We can set up quality gates in this tool and we can fail the build if code quality is not met.

The ultimate use case of SonarQube is to improve the quality of the code and eliminate the defects at development stage.

Installing SonarQube on a linux machine

SonarQube is implemented using java language, to run this tool java needs to be installed on the server.

1. Make sure java8 is installed
2. cd /opt
3. Download SonarQube using the below command
   wget https://sonarsource.bintray.com/Distribution/sonarqube/sonarqube-6.5.zip
4. Unzip the file
   sudo unzip sonarqube-6.5.zip
5. Start sonar server
   cd /opt/sonarqube-6.5/bin/linux-x86-64
   sudo ./sonar.sh start

Note: If there is issue starting sonar we can check the logs,
under (/opt/sonarqube-6.5/logs)
Default port number of sonar is 9000
Access the sonar form webbrowser
http://public-ip-address:9000
Default username and password is admin/admin

Publishing java code from jenkins to sonar

1. For this to work, mvn needs java8
2. Configure sonar server detail sin maven settings.xml
   ssh into jenkins server
3. Open maven settings.xml
   a. To find where maven is installed type (mvn -version)
   b. By default maven is installed at (/usr/share/apache-maven)
   c. Open setting.xml under /usr/share/apache-maven/conf

```
<profiles>
   <profile>
    <id>sonar</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
      <sonar.host.url>http://172.31.0.73:9000</sonar.host.url>
   </properties>
  </profile>
</profiles>
```

Create a new job in jenkins
Under build select maven
Under goals → package sonar:sonar

Publishing Junit coverage report to sonar

Under maven goal specify   org.jacoco:jacoco-maven-plugin:prepare-agent

Setting quality gates in sonar and failing build if code quality is not met.

Create quality gates in SonarQube

1. Create two gates
    a. If blockers greater than zero then error
    b. If junit coverage is less than 85% then error
2. In jenkins
    a. Install plugin “Quality Gates Plugin”
    b. Configure sonar details to the quality gates plugin
    c. Manage Jenkins → Configure System
    d. Add Sonar Instance, and provide sonar details
        i. Name, URL, username, password
    e. Configure the job
        i. Post build actions add quality gate and add the project key