Day 05 - 20 Jan 2023 (Jenkins)

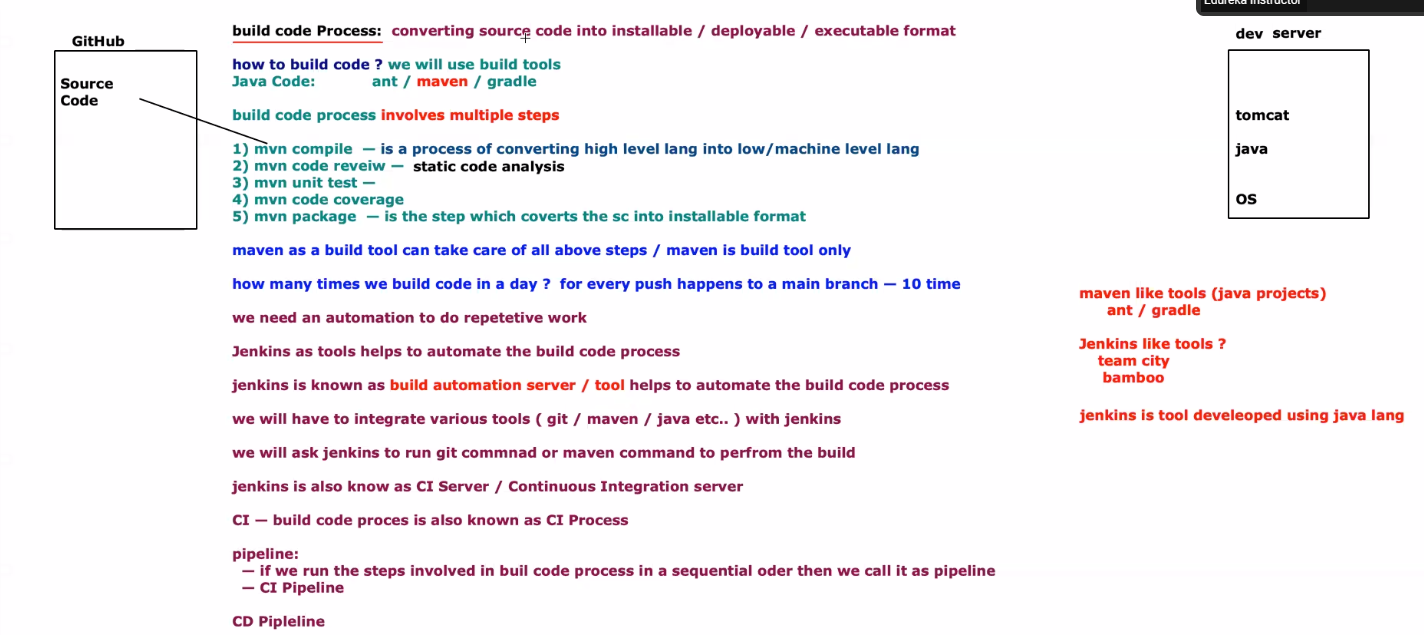
20 January 2023

20:38

<https://learning.edureka.co/classroom/recording/1414/11761/1574515?tab=ClassRecording>

**Agenda**

Continuous Integration With Jenkins



1. Dev team work on source code which is saved in repo like github
2. Code has to be converted into installable / deployable / executable format
3. We will use
4. Here we will use **MAVEN** to build code. Other example, ANT, GRADLE

Installation steps

<https://github.com/lerndevops/labs/tree/master/jenkins/Install>

1. Install **Java**

sudo add-apt-repository ppa:openjdk-r/ppa

sudo apt-get update

sudo apt-get install -y openjdk-11-jdk

*[check: java -version]*

1. Install **Maven**

cd /tmp ; sudo wget <https://dlcdn.apache.org/maven/maven-3/3.8.7/binaries/apache-maven-3.8.7-bin.tar.gz>

cd /tmp ; sudo tar -xzf apache-maven-3.8.7-bin.tar.gz -C /opt/

mv /opt/apache-maven-3.8.7 /opt/maven

sudo echo "MAVEN\_HOME=\"/opt/maven\"" >> /etc/profile

sudo echo "PATH=\$MAVEN\_HOME/bin:\$PATH" >> /etc/profile

source /etc/profile

*[check: mvn --version]*

1. Install **Jenkins**

sudo wget -q -O - <https://pkg.jenkins.io/debian-stable/jenkins.io.key> | sudo apt-key add -

sudo echo deb [https://pkg.jenkins.io/debian-stable binary/](https://pkg.jenkins.io/debian-stable%20binary/) | sudo tee /etc/apt/sources.list.d/jenkins.list

sudo apt-get update

sudo apt-get install jenkins

1. Git clone <https://github.com/lerndevops/samplejavaapp>
2. Maven requires
   1. pom.xml is requried
   2. Src folder - source code must be saved here

1. Maven commands
   1. /opt/maven/bin/mvn compile
   2. /opt/maven/bin/mvn package  
        
      Building manually every time is tedious work.  
      Automate manual task in automation usign Jenkins
2. Jenkins is a UI tool, a web application.
3. Jenkins defaults:
   1. Folder: cd /var/lib/lib/jenkins/
   2. Port: 8080
   3. <http://IP:8080>
      1. Unlock - copy password file
      2. Cat path copied
      3. Copy the password
      4. Paste it in web site
      5. Select **Install suggested plugins**
      6. Set username, password, full name and email address
      7. Remove public IP and replace IP with local IP

<https://github.com/lerndevops/labs/tree/master/jenkins/Sonar>

# Integrate tools with Jenkins

Login to Jenkins

1. Manage Jenkins
2. Global tool configuration
   1. JDK
      1. Add jdk  
         1. Name = java1.11 (any name will work)
         2. JAVA\_HOME = /usr/lib/jvm/java-11-openjdk-amd64  
            Default installation pathof java  
            *untick - install automatically checkbox*
         3. Path to GIT executable  
            /usr/bin/git
   2. Maven
      1. Add maven
         1. Name = maven3.8.7
         2. MAVEN\_HOME = /opt/maven  
            default path  
            install automatically untick

# Create a Job / New Item

1. Name = compile
2. Freestyle
3. Source Code Mangement
   1. Select Git
   2. Repository URL = <https://github.com/pradeepviswa/samplejavaapp.git>
   3. Credentials - add
      1. Username = pradeepviswa  
         tick treat username as secret
      2. Password = ghp\_QHng0nLUcjVMBqWVgBhNWg3Rqklo1G19f8tK
      3. Id = GITHUB-CREDS
   4. Branch specifier = \*/master
4. Build Steps
   1. Execute Shell
      1. /opt/maven/bin/mvn compile
5. Click on Build Now

Where is the close got created

Cd /var/lib/jenkins/workspace

# Create code review step

1. New item
2. Name = codereview
3. Source Code Mangement
   1. Select Git
   2. Repository URL = <https://github.com/pradeepviswa/samplejavaapp.git>
4. Build Steps
   1. Execute Shell
      1. /opt/maven/bin/mvn -P metrics pmd:pmd
5. Click on Build Now
6. Build Now
7. Check console output - should be sucecess at the end
8. Result in workspace -> target -> pmd.xml  
   this is not in readable format  
   install plugin to read and present output in graphical format

# Install Plugin

About Jenkins Plugins <https://Plugins.jenkins.lo>

1. Dashboard -> manage Jenkins
2. Manage plugins
3. Available plugins
4. Search for - **warnings next generation**
5. Install

# Configure Code Review step in more effective way

1. Click on Codereview
2. Configure -> build steps
3. Post-builds action
   1. Record compiler warnign in static analysis results
   2. Tool = PMD
   3. Report file format - \*\*/pmd.xml
4. Rebuild job
5. Check log
6. PMD option will appear

Practice Done

25 January 2023

19:13

Day 06 - 23 Jan 2023 (Jenkins)

23 January 2023

21:39

[*https://learning.edureka.co/classroom/recording/1414/11762/1575041?tab=ClassRecording*](https://learning.edureka.co/classroom/recording/1414/11762/1575041?tab=ClassRecording)

From <[*https://github.com/lerndevops/labs/blob/master/jenkins/Jenkins-Setup-CI.txt*](https://github.com/lerndevops/labs/blob/master/jenkins/Jenkins-Setup-CI.txt)>

Jenkins ( CI )

CI ( continuous Integration )

Jenkins Job : maven goal : post build action

job1-compile         : compile : NA

job2-codereview         : -P metrics pmd:pmd : \*\*/pmd.xml (record compiler warnings & static analysis results)

job3-unittest : test : target/surefire-reports/\*.xml (Publish Junit test results)

job4-codecoverage : verify : \*\*/\*\*.exec (Record JaCoCo coverage report)

job5-package : package : NA

# Jenkin Unit Testing

1. Code  
    <https://github.com/pradeepviswa/samplejavaapp>
2. It is developer's responsibility to write unit test
3. Test case should be in this folder  
   /src/test
4. Dashboard --> New item --> Name - Unit Test --> freestyle project
   1. Source code management
      1. GIT
      2. Repository URL - <https://github.com/pradeepviswa/samplejavaapp.git>
   2. Build Triggers -->
      1. Build Steps --> Execute Shell:  
         /opt/maven/bin/mvn **test**
      2. Add Post-build Action Button  
         Publish Junit test result report
      3. Test report XMLs (default path)  
         target/surefire-reports/\*.xml  
         (refer this page: <https://github.com/lerndevops/labs/blob/master/jenkins/Jenkins-Setup-CI.txt>)
   3. Apply --> Save
5. Click on **Build Now**
6. Check the output

# Code Coverage - means make sure each class, function, or any other section in code has been tested thoroughly

1. New Iten - code coverage - freestyle
   1. Source code repository  
      <https://github.com/lerndevops/samplejavaapp>
   2. Add build step - execute shell  
      /opt/maven/bin/mvn **verify**  
      (internally done by java code coverage tool)
      1. Add Plugin if jacoco is not available then install plugin
      2. Manage jenkins -> manage plugins - avialable plugings -> jacoco -> install

1. Add post build step  
   Record JaCoCo coverage report  
   leave default vlues

**Save**

1. Build now
   1. Check coverage report

M means missed

C means covered

# Package - It converts code into deployable format

1. New Item -> Name: Package -> don't choose anything -> OK
2. Repo - <https://github.com/pradeepviswa/samplejavaapp>
3. Build trigger
   1. Build steps  
      /opt/maven/bin/mvn **package**
4. Jenkins home dir  
   /var/lib/jenkins/workspace/package/target  
   sampleapp.war is the executable format

# These should be automated using CI Pipeline

* Compile
* Code review
* test
* Code coverage
* package

## To set the sequence,

1. go to **code review ->** Configure
   1. Build Triggers
      1. Build after other projects are built
      2. Projects to watch --> **compile,**
         1. Trigger only if build is stable
2. Unit Test --> codereview,
3. Code Coverage --> unittest,
4. Package --> codecoverage,

## Trigger Compile and watch

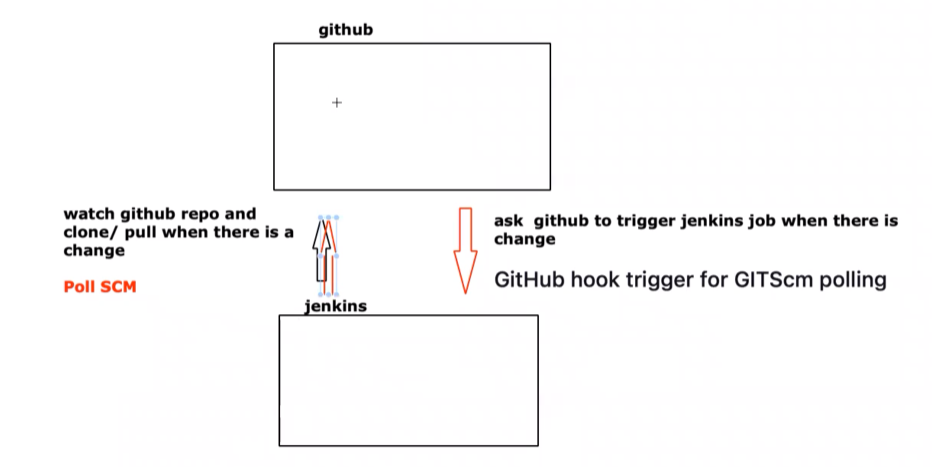
* Build queue
* Build executor Status

## Here job sequence is not clear. Change the visual.

1. Manage -> manage pluging -> plugin -> build Pipeline - > Install
2. Click on +
   1. Name: CI\_Pipeline
   2. Build Pipeline View
   3. Create
   4. Select Initial Job
      1. Compile
   5. Apply -> Ok
3. Click on **Trigger a Pipeline**

## Here pipeline trigger is human activity. Automate it

1. Check code is checked in that time pipeline should trigger automatically
2. Click on Compile job
3. Configure
4. Build triggers
   1. Build triggers -> Poll SCM or **GitHub hook triggers for GITScm Polling  
      Refer this page:** [**https://github.com/lerndevops/labs/blob/master/jenkins/Configurations/trigger-build-with-github-webhook.txt**](https://github.com/lerndevops/labs/blob/master/jenkins/Configurations/trigger-build-with-github-webhook.txt)



1. To Enable webhook in GIT
   1. Open Repo: <https://github.com/pradeepviswa/samplejavaapp.git>
   2. Settings -> webhooks -> Add webhook
      1. Payload URL (Jenkin URL) --> <http://18.188.184.96:8080/github-webhook/>
      2. Content type -> application/json
      3. Which event
         1. Just the push event
      4. Add web hook
2. On Jenkins page Apply Save

Commit something in GitHub, pipeline should trigger automatically.

# Secure Jenkins Console

1. Manage Jenkins -> Security -> Global security
   1. Security Realm:
      1. **Jenkins' own user database: own DB**
      2. LDAP: Active Directory Authentication
2. Allow users to sign up *(This will provide* ***Create Account*** *link at the time of login)*
3. Create few accounts and see them in **Dashboard -> Manage Jenkins-> Security-> manage users**  
   see all available users  
   by default all users are admin users
4. **Set authorization now : Dashboard -> Manage Jenkins -> Security -> Global Security -> Authorization**

Logged in user can do anything (means admin user)

1. Select: Matrix-based security
   1. Add user -> admin
      1. Select all boxes
   2. Authenticated Users ->
      1. read access in overall & job
   3. Specific people - specific privilege
      1. Add specific user
      2. Select required checkboxes
   4. For advanced permission setting Install required plugin like
      1. Role based authorization
      2. Active directory
      3. AD Azure, etc

# Jenkins Master - Agent Concept

1. Used to distribute the load
2. Currently these are installed on one server. Install each of them in different server

Git

Maven

Java

Jenkins server

1. On ubuntu server then process using this command  
   ps -eflgrep java

Day 07 - 24 Jan 2023 (Jenkins, Groovy)

24 January 2023

20:36

<https://learning.edureka.co/classroom/recording/1414/11763/1575380?tab=ClassRecording>

Agenda

# Install Agent on VM2

1. On vm2 installed, git, java, mvn
2. <<Install Steps.txt>>
3. Install Jenkins Agent now
4. Manage Jenkins - global security -> agent (by default is disabled). To enabe select -> Random
5. Manage jenkins - system conf - manage nodes & clouds
   1. New Node
   2. Node name - Linux-java-agent
      1. Permanent - create
      2. Labels = same as name
      3. Remote root directory (click on help)
         1. Dedicated dir for Jenkins on agent VM  
            ceate folder on VM  
            mkdir jagenthome

Chmod 777 jagenthome/

/opt/jagenthome

1. Number of executors - 3  
   (number of parallel execution allowed)
2. Usage - use this code as much as possible
3. Launch method - launch agent by connecting it to the controller
4. SAVE
5. At this time agent will be offline. Install agent on VM2.
6. Run command from jagenthome folder  
   agent specific install command will appear in Jenkins itself, copy and execute it, it will appear like this
7. 
8. Agent is connected message should appear
9. On vm 2, under this folder package will be created  
   /opt/jagenthome/workspace/package/target

Compile step trigger

Observe the agent

Master will not take charge unless agent is busy

# Restrict a job to run on specific agent only

Job (Package) -> configurtion -> General -> Restrict where this project can be run

1. Label Expression (Agent name) - linux-java-agent (agent that we created)

# Jenkins Pipeline as a Code

It means CI pipeline we created, create a script or template to automate it

Scripted and delcarative

Each individual job can be created using script

1. New job - pipeline as a code
2. Pipeline - ok
3. Inside pipeline
   1. Script - (Groovy script)
   2. Pipeline script from SCM
   3. SCM - GIT
   4. <https://github.com/lerndevops/samplejavaapp.git>
   5. Script path - jenkins/jenkinsfile-CI

# How to create Groovy script Pipeline as Code

pipeline {

agent any # or ageny linux-java-agent

stages {

stage('stage1-compile'){

steps{

}

post{

success{ #or always

}

always{

sh 'echo send-email'

}

}

}

stage('stage2-codereview'){

steps{

}

post{

}

}

}

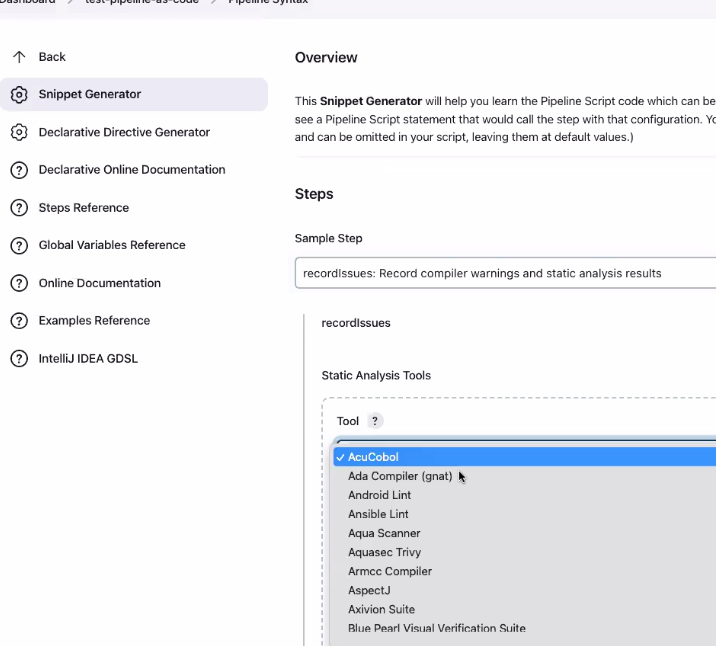
}

Generate script from here

Create a blank job, type as pipeline, just save it.

Pipeline syntax will appear on left side





# Deploy package

Use VM2 to install web server (tomcat)

Download latest from tomcat web site

Check Binary Distribution

tar,gz copy link

# wget paste URL

# tar -xzf apache-tomca-9.0.71.tar.gz -C /opt

# cd /opt

# cd apache-tomcat-9.0.71

Webapps folder is present inside apache folder

Copy .war file in webapps folder

# Cd opt/jagenthome/worspace/package/garget

# Cp sampleapp.war /opt/apache-tomcat…./webapps/

# Cd opt/apache-tomcat…

Launch tomcat in memory

# Cd bin

# Bash startup.sh

Check service running or not

# Ps -efl|grep tomcat

Check in browser

<http://ip:8080>

Default tomcat webpage should open

Test the deployment

<http://18.217.168.98:8080/sampleapp/>

