DevOps Certification Training

# Module 3: Git And Jenkins

Assignment – 2

## Problem Statement

You are working as a DevOps Engineer in a company named Sanders & Fresco Pvt ltd. You have been asked by your manager to create a Maven Project using Jenkins and build a war file of that project. As a proof of concept, you have been given a web application to build. And once done with building the war file, deploy it over the tomcat server.

Steps to solve:

* Open Jenkins and create a Maven project using it.
* You will have to create the following jobs, which are as follow:
* Compile
* Code Review
* Unit test
* Package
* Deploy

## Solution:

1. Configure Maven in Jenkins

Go To: Manage Jenkins > Global Tool Configuration

In the JDK Section

> Click Add JDK

> Uncheck Auto install Java

> Name: myjava

Java Home : /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.amzn2.0.2.x86\_64

(Get the path from mvn --version copy only till 64)

Scroll down to Maven Section

> Add Maven

> Name: mymaven

> Install automatically should be checked and version should be latest

Click on Apply and Save

All configurations and settings are complete

### 1.Compile

1. Create a freestyle project

Graphical user interface, text, application

Description automatically generated

1. Add Description

Graphical user interface, text, application, email

Description automatically generated

1. Click on Source Code management and provide the Git Source

Graphical user interface, text, application, email

Description automatically generated

1. Build Trigger – Set to trigger the build whenever there is a new commit using Poll SCM

Graphical user interface, text, application, email

Description automatically generated

1. a. From Build – Select Invoke top-level Maven targets

Graphical user interface, application, Word

Description automatically generated

b. Select the maven version, enter the command “compile” and save

Graphical user interface, text, application, email

Description automatically generated

1. Now Build “1. Compile” and check the logs

Table

Description automatically generated with medium confidence

### 2.CodeReview

1. New Item > 2.CodeReview > Freestyle Project> OK

Graphical user interface, text, application, email

Description automatically generated

1. Add the Git Repository

Graphical user interface, text, application, email

Description automatically generated

1. Click on Build and Select mymaven. In Goals enter “pmd:pmd”

Graphical user interface, text, application, email

Description automatically generated

1. Click on Apply and then Save. Then Build

Graphical user interface, application, Teams

Description automatically generated

1. After the build is successful, Go to Workspace> target

Graphical user interface, application

Description automatically generated

1. The pmd.xml file is generated

Graphical user interface, application

Description automatically generated

1. Open and view it

Text

Description automatically generated

1. It is not in a readable format. So we will take help from Jenkins
2. Go to Dashboard> Manage Jenkins> Manage Plugins
3. Install the Plugin “Warnings Next Generation”

Graphical user interface, text, application

Description automatically generated

1. It should get installed

Graphical user interface, application

Description automatically generated

1. Now go to the 2.CodeReview job and click on Configure
2. Click on Post Build Actions and Add Post Build Actions and select Record Compiler Warnings

Graphical user interface, text, application, Word

Description automatically generated

1. Tool > PMD
2. Enter the path target/pmd.xml

Graphical user interface, text, application, email

Description automatically generated

1. Now build the code again

Graphical user interface, application

Description automatically generated

1. Once the Build completes, we will see a new tab called PMD Warnings

Graphical user interface, application

Description automatically generated

### 3.UnitTest

1. This will also be a Freestyle Project
2. In Source code management enter the same Git repo
3. Go to Build and select Invoke Top level maven Targets

Graphical user interface, text, application, email

Description automatically generated

1. Build the project and it will execute the test cases
2. Surefire test reports will be available in the path – target/surefire-reports but this will also not be readable
3. Junit Plugin will be needed to read, and it is installed by default.
4. So go to Post-build Actions and Select Publish jUnit test result report

Graphical user interface, text, application, email

Description automatically generated

1. The post build files will only be generated after the first build. So the xml files wont be available in the first build.
2. Go to the newly generated Test Result tab

Graphical user interface, text, application

Description automatically generated

### 4.CodeCoverage

1. This will also be a Freestyle Project
2. In Source code management enter the same Git repo
3. Go to Build and select Invoke Top level maven Targets
4. Select Maven Version and Goals

Graphical user interface, text, application, email

Description automatically generated

1. In the Goals, we are entering “cobertura:cobertura -Dcobertura.report.format=xml” as this will generate the report in XML. By default, the report will be generated in html.
2. Save and Build to generate the report
3. The **coverage.xml** report will be in the Workspace > target/site/cobertura folder

Graphical user interface, text, application

Description automatically generated

1. Install the plugin Cobertura
2. Go to the Project 4.CodeCoverage and Click on Configure
3. Click on Postbuild Actions and Select **Publish Cobertura Coverage Report**
4. Enter the path as **target/site/cobertura/coverage.xml** and click on Save
5. Go to Console Output and check the build log
6. Now on the left side a new link ‘Coverage Report’ will be available. Click there

Chart

Description automatically generated

### 5.Package

1. This will also be a Freestyle Project
2. In Source code management enter the same Git repo
3. Go to Build and select Invoke Top level maven Targets
4. Select Maven Version and Goals
5. In the Goals, enter “package”

Graphical user interface, text, application, email

Description automatically generated

1. We won’t build the file now. It will be done once all the above projects are connected to create a pipeline

## Create Pipeline

Now all the 5 projects have been created

A picture containing table

Description automatically generated

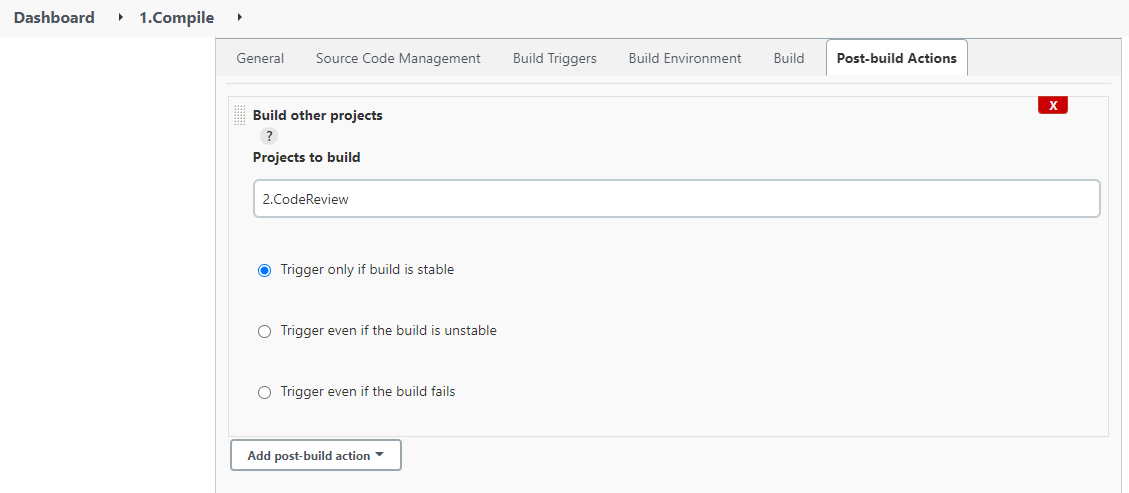
#### Compile

1. Go to 1.Compile >> Configure >> Post-build Actions
2. From Add post-build action, select Build other projects

Graphical user interface, application

Description automatically generated

1. In the Projects to build, enter 2.CodeReview



1. Click on save

#### CodeReview

1. Follow Similar step as above and Add “3.UnitTest”

Graphical user interface, text, application, email

Description automatically generated

#### UnitTest

Same Steps as above

Graphical user interface, text, application, email

Description automatically generated

#### CodeCoverage

Same steps as above.

Graphical user interface, text, application, email

Description automatically generated

#### Install Plugin

Now go to Plugin Manager and Install the pipeline “Build Pipeline”

Graphical user interface, application

Description automatically generated

1. Click on New View and name it “buildpipeline” and Select the type as Build pipeline View

Graphical user interface, text, application

Description automatically generated

1. Click on Create
2. In the Pipeline Flow, Initial job is set by default

Graphical user interface, text, application, email

Description automatically generated

1. Click on OK and this Pipeline will be generated

Graphical user interface, application

Description automatically generated

## Run Pipeline

1. Click on Run

Graphical user interface, application

Description automatically generated

1. Click on Package job and Go to Console Output

Graphical user interface, text, application

Description automatically generated

1. At the bottom it will show that the war file is generated

Graphical user interface, text, application

Description automatically generated