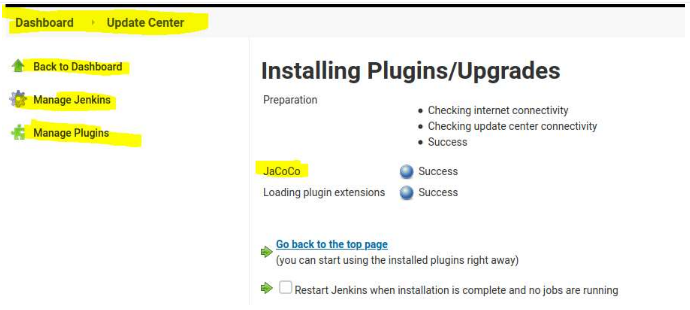
Assignment 6.3 Testing Code Coverage

Sunday, January 3, 2021   8:56 AM

6.3 Testing Code Coverage

GitHub URL: https://github.com/0ktay3min/Assignments.git

Step 1: Installing JaCoCo plugin
- In your Jenkins Dashboard click *Manage Jenkins*
- Click *Manage Plugins*
- Click *Available* Tab and search for JaCoCo Plugin
- Install JaCoCo plugin and restart Jenkins



Installing Plugins/Upgrades

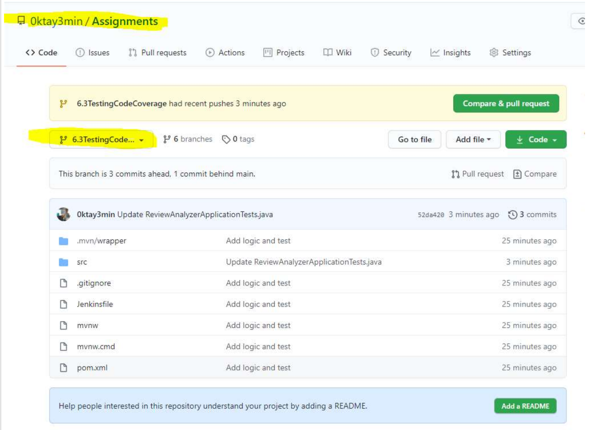
Preparation
- Checking internet connectivity
- Checking update center connectivity
- Success

JaCoCo Success
Loading plugin extensions Success

Go back to the top page
(you can start using the installed plugins right away)

Restart Jenkins when installation is complete and no jobs are running

Step 2: Create a public Github Repository



6.3TestingCodeCoverage had recent pushes 3 minutes ago

6 branches 0 tags

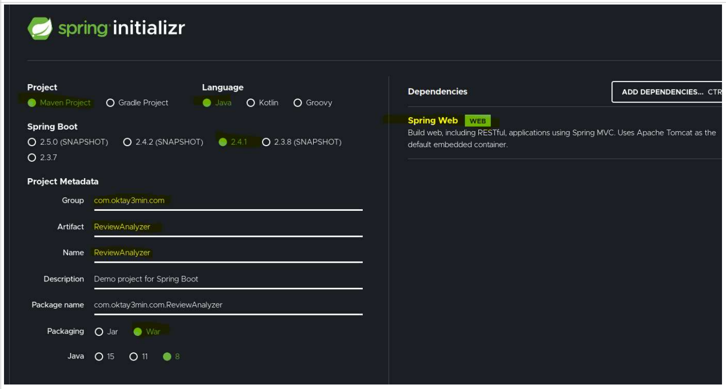
This branch is 3 commits ahead, 1 commit behind main.

0ktay3min Update ReviewAnalyzerApplicationTests.java 52da420 3 minutes ago 3 commits

.mvn/wrapper	Add logic and test	25 minutes ago
src	Update ReviewAnalyzerApplicationTests.java	3 minutes ago
.gitignore	Add logic and test	25 minutes ago
Jenkinsfile	Add logic and test	25 minutes ago
mvnw	Add logic and test	25 minutes ago
mvnw.cmd	Add logic and test	25 minutes ago
pom.xml	Add logic and test	25 minutes ago

Help people interested in this repository understand your project by adding a README.

Step 3: Creating a Spring boot project
- Go to www.start.string.io
- Select Maven as the project type.
- Fill Group and Artifact with appropriate values.
- Add **Web** to Dependencies.
- Select Packing option as *War* file
- Select Java Version.
- Click on **Generate Project.**
- The generated skeleton project should be downloaded as a zip file.



Project: [✓ Maven Project] Gradle Project

Language: [✓ Java] Kotlin Groovy

Spring Boot: 2.5.0 (SNAPSHOT) 2.4.2 (SNAPSHOT) [✓ 2.4.1] 2.3.8 (SNAPSHOT) 2.3.7

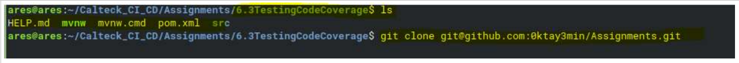
Project Metadata

Group	com.oktay3min.com
Artifact	ReviewAnalyzer
Name	ReviewAnalyzer
Description	Demo project for Spring Boot
Package name	com.oktay3min.com.ReviewAnalyzer
Packaging	Jar [✓ War]
Java	15 11 [✓ 8]

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

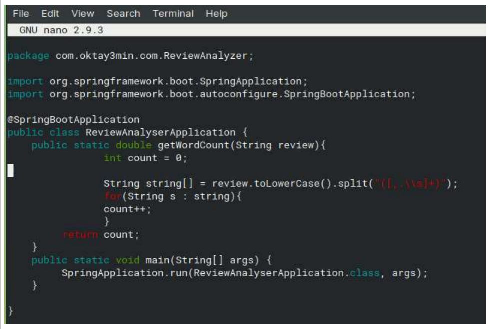
Step 4: Adding the code to remote repository
- Create a directory and save the downloaded files inside this repository
- Clone Github Repository URL git@github.com:0ktay3min/Assignments.git



```
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ ls
HELP.md  mvnw  mvnw.cmd  pom.xml  src
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ git clone git@github.com:0ktay3min/Assignments.git
```

- Navigate to the ReviewAnalyzer folder inside 6.3TestingCodeCoverage/src/main/java/com/oktay3min/com/ReviewAnalyzer
- Open the **ReviewAnalyserApplication.java** in any text editor.
- Add the following method to the file and save it.

```
package com.oktay3min.com.ReviewAnalyzer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ReviewAnalyserApplication {
   public static double getWordCount(String review){
            int count = 0;

            String string[] = review.toLowerCase().split("([,.\\s]+)");
            for(String s : string){
            count++;
            }
      return count;
   }
   public static void main(String[] args) {
      SpringApplication.run(ReviewAnalyserApplication.class, args);
   }
```

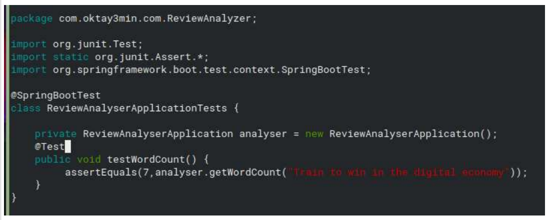


```
 GNU nano 2.9.3

package com.oktay3min.com.ReviewAnalyzer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ReviewAnalyserApplication {
    public static double getWordCount(String review){
                int count = 0;

                String string[] = review.toLowerCase().split("([,.\\s]+)");
                for(String s : string){
                count++;
                }
         return count;
    }
    public static void main(String[] args) {
         SpringApplication.run(ReviewAnalyserApplication.class, args);
    }

}
```

```
}
```

- Navigate to the *ReviewAnalyser* folder within the 6.3TestingCodeCoverage/src/test/java/com/oktay3min/com/ReviewAnalyzer folder.
- Open the **ReviewAnalyserApplicationTests.java** in any text editor.
- Add the following *test* method to the file and save it.

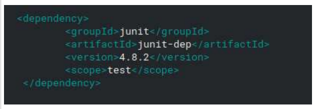
```
package com.oktay3min.com.ReviewAnalyzer;

import org.junit.Test;
import static org.junit.Assert.*;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class ReviewAnalyserApplicationTests {

    private ReviewAnalyserApplication analyser = new ReviewAnalyserApplication();
    @Test
    public void testWordCount() {
        assertEquals(7,analyser.getWordCount("Train to win in the digital economy"));
    }
}
```



```
package com.oktay3min.com.ReviewAnalyzer;

import org.junit.Test;
import static org.junit.Assert.*;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class ReviewAnalyserApplicationTests {

    private ReviewAnalyserApplication analyser = new ReviewAnalyserApplication();
    @Test
    public void testWordCount() {
        assertEquals(7,analyser.getWordCount("Train to win in the digital economy"));
    }
}
```

- Save the file and exit the text editor.
- Open the pom.xml and add the following dependency.

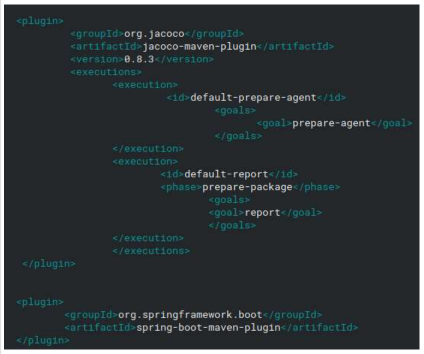
```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit-dep</artifactId>
  <version>4.8.2</version>
  <scope>test</scope>
</dependency>
```



```
 <dependency>
        <groupId>junit</groupId>
        <artifactId>junit-dep</artifactId>
        <version>4.8.2</version>
        <scope>test</scope>
  </dependency>
```

- Add the jacoco plugin to pom.xml with the following xml code:

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.3</version>
    <executions>
      <execution>
      <id>default-prepare-agent</id>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
      </execution>
      <execution>
      <id>default-report</id>
      <phase>prepare-package</phase>
      <goals>
        <goal>report</goal>
      </goals>
      </execution>
    </executions>
  </plugin>
```



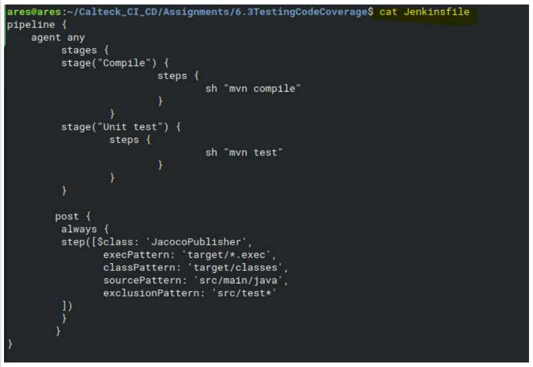
```
 <plugin>
          <groupId>org.jacoco</groupId>
          <artifactId>jacoco-maven-plugin</artifactId>
          <version>0.8.3</version>
          <executions>
                <execution>
                        <id>default-prepare-agent</id>
                                <goals>
                                        <goal>prepare-agent</goal>
                                </goals>
                </execution>
                <execution>
                        <id>default-report</id>
                        <phase>prepare-package</phase>
                                <goals>
                                <goal>report</goal>
                                </goals>
                </execution>
                </executions>
  </plugin>


 <plugin>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-maven-plugin</artifactId>
 </plugin>
```

- Save the file and exit the text editor

Step 5: Creating and commuting a Jenkinsfile

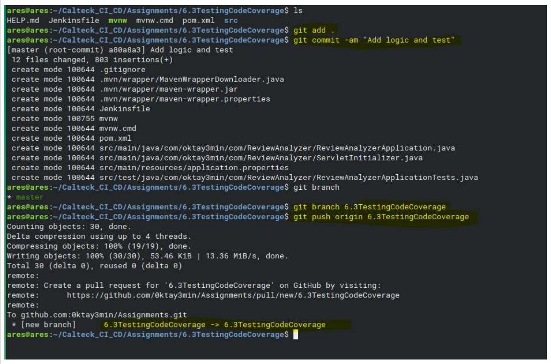
- Navigate to the *ReviewAnalyser* root directory where the pom.xml is.
- Open nano text editor and create a new text file called *Jenkinsfile* and add the following script to it.
- $ sudo nano Jenkinsfile => to create a file called Jenkinsfile using nano text editor

```
pipeline {
    agent any
      stages {
      stage("Compile") {
          steps {
            sh "mvn compile"
          }
      }
      stage("Unit test") {
        steps {
            sh "mvn test"
          }
      }
    }

      post {
      always {
      step([$class: 'JacocoPublisher',
          execPattern: 'target/*.exec',
          classPattern: 'target/classes',
          sourcePattern: 'src/main/java',
          exclusionPattern: 'src/test*'
      ])
      }
      }
}
```



```
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ cat Jenkinsfile
pipeline {
    agent any
         stages {
         stage("Compile") {
                         steps {
                                sh "mvn compile"
                         }
                }
         stage("Unit test") {
                steps {
                                sh "mvn test"
                        }
                }
         }

        post {
         always {
         step([$class: 'JacocoPublisher',
                execPattern: 'target/*.exec',
                classPattern: 'target/classes',
                sourcePattern: 'src/main/java',
                exclusionPattern: 'src/test*'
         ])
         }
        }
}
```

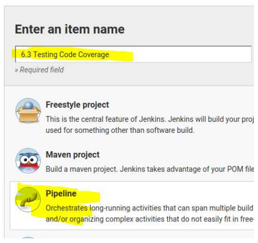
- Save the file as **Jenkinsfile** with no extension.
- Commit the changes to the remote SCM.
- $ git branch 6.3TestingCodeCoverage => this will create a new branch called 6.3TestingCodeCoverage
- $ git add . => to add all files to staging area
- $ git commit -m "Add logic and test" => to commit changes
- $ git push -u origin 6.3TestingCodeCoverage => to push the files to remote repository



```
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ ls
HELP.md  Jenkinsfile  mvnw  mvnw.cmd  pom.xml  src
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ git add .
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ git commit -am "Add logic and test"
[master (root-commit) a80a8a3] Add logic and test
 12 files changed, 803 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .mvn/wrapper/MavenWrapperDownloader.java
 create mode 100644 .mvn/wrapper/maven-wrapper.jar
 create mode 100644 .mvn/wrapper/maven-wrapper.properties
 create mode 100644 Jenkinsfile
 create mode 100755 mvnw
 create mode 100644 mvnw.cmd
 create mode 100644 pom.xml
 create mode 100644 src/main/java/com/oktay3min/com/ReviewAnalyzer/ReviewAnalyzerApplication.java
 create mode 100644 src/main/java/com/oktay3min/com/ReviewAnalyzer/ServletInitializer.java
 create mode 100644 src/main/resources/application.properties
 create mode 100644 src/test/java/com/oktay3min/com/ReviewAnalyzer/ReviewAnalyzerApplicationTests.java
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ git branch
* master
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ git branch 6.3TestingCodeCoverage
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$ git push origin 6.3TestingCodeCoverage
Counting objects: 30, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (19/19), done.
Writing objects: 100% (30/30), 53.46 KiB | 13.36 MiB/s, done.
Total 30 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for '6.3TestingCodeCoverage' on GitHub by visiting:
remote:      https://github.com/0ktay3min/Assignments/pull/new/6.3TestingCodeCoverage
remote:
To github.com:0ktay3min/Assignments.git
 * [new branch]      6.3TestingCodeCoverage -> 6.3TestingCodeCoverage
ares@ares:~/Calteck_CI_CD/Assignments/6.3TestingCodeCoverage$
```

Step 6: Creating a multistage pipeline in Jenkins

- Go to Jenkins dashboard.
- Click on *New Item*.
- Enter a name for your build job.
- Select *Pipeline* as the build job type.



Enter an item name

6.3 Testing Code Coverage

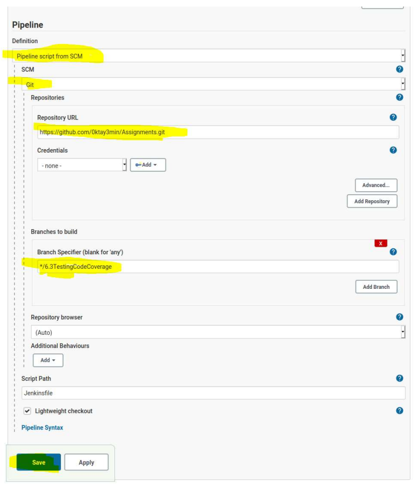
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your proj
used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM file

Pipeline
Orchestrates long-running activities that can span multiple build
and/or organizing complex activities that do not easily fit in free

- Click OK.
- On the configuration page, scroll down to the Pipeline section. Change *Definition* from *Pipeline script* to *Pipeline script from SCM*
- Select *Git in SCM.*
- Add the repository URL.
- Click Save.



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/0ktay3min/Assignments.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */6.3TestingCodeCoverage

Repository browser: (Auto)

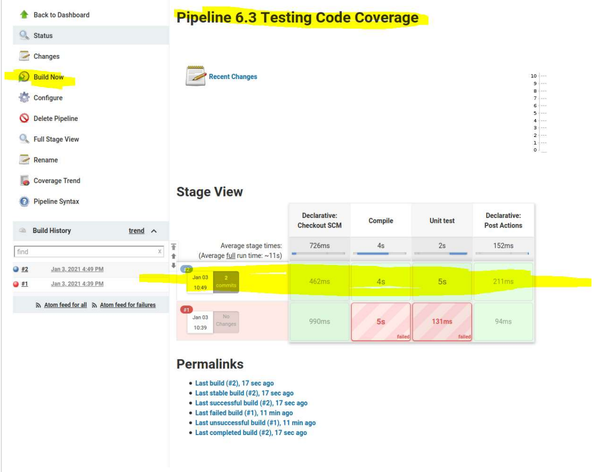
Additional Behaviours

Script Path: Jenkinsfile

✓ Lightweight checkout

Pipeline Syntax

Step 7: Running a multistage pipeline in Jenkins
- Click on *Build Now* in the project window.
- Jenkins will now build your pipeline and output the logs.



Pipeline 6.3 Testing Code Coverage

Build History

#2	Jan 3, 2021 4:49 PM
#1	Jan 3, 2021 4:39 PM

Atom feed for all Atom feed for failures

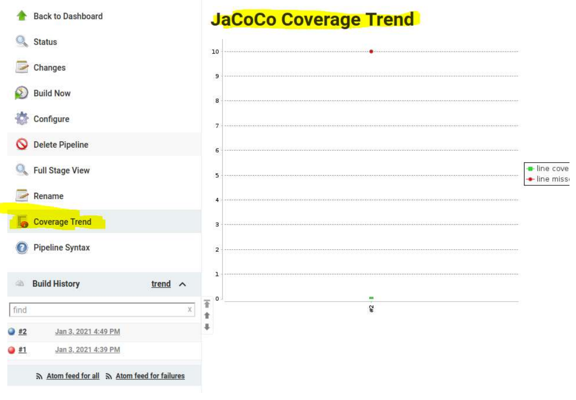
Stage View

	Declarative: Checkout SCM	Compile	Unit test	Declarative: Post Actions
Average stage times: (Average full run time: ~11s)	726ms	4s	2s	152ms
#2 Jan 03 10:49 2 commits	462ms	4s	5s	211ms
#1 Jan 03 10:39 No Changes	990ms	5s failed	131ms failed	94ms

Permalinks

- Last build (#2), 17 sec ago
- Last stable build (#2), 17 sec ago
- Last successful build (#2), 17 sec ago
- Last failed build (#1), 11 min ago
- Last unsuccessful build (#1), 11 min ago
- Last completed build (#2), 17 sec ago

- Click on *Coverage Trend* to view the coverage trend.



JaCoCo Coverage Trend

Build History

#2	Jan 3, 2021 4:49 PM
#1	Jan 3, 2021 4:39 PM

Atom feed for all Atom feed for failures