DevOps

Caltech | Center for Technology & Management Education

Post Graduate Program in DevOps

simplilearn

Automated Testing in Jenkins

Learning Objectives

By the end of this lesson, you will be able to:

◉ Explain automation testing

◉ Select appropriate tool for your application

◉ Generate and display test reports

◉ Work with Cobertura and Clover

◉ Implement acceptance and performance testing

Jenkins as an Automated Testing Tool

What Is Automated Testing?

Test automation is the use of software to control the execution of tests and the comparison of actual and predicted outcomes.

- One of the most efficient ways to perform automated testing is to write test scripts first using approach techniques, like Test-Driven Development (TDD) or Behavior-Driven Development (BDD).

- BDD framework,Cucumber, presents acceptance test in such a way that testers, product owners, and end users easily understand.

- During the test phase, Jenkins plays a major role in the Agile project.. It provides various plugins for different testing tools to integrate and automate tests during builds before deployment.

Caltech | Center for Technology & Management Education

simpli-learn
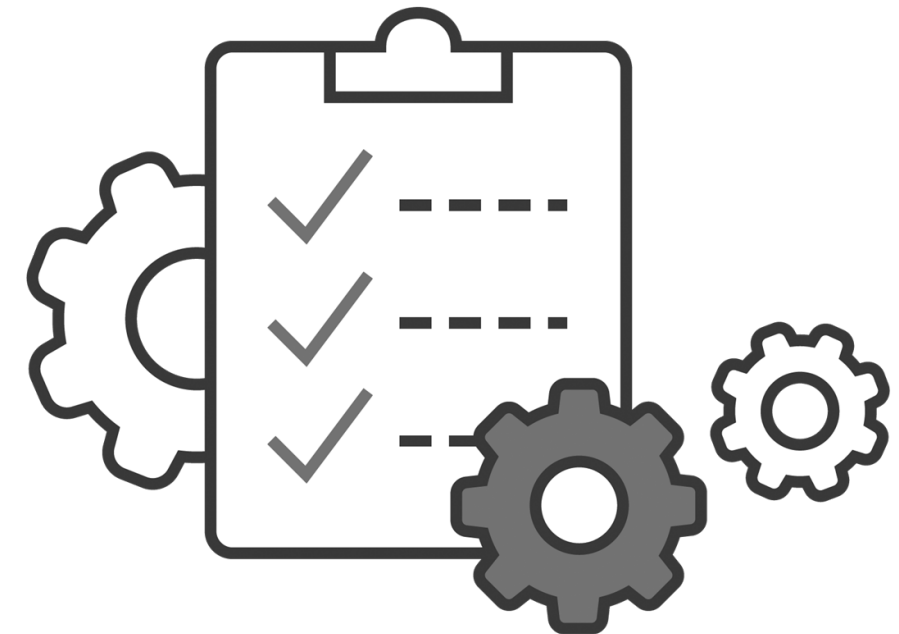
Testing Tools

There are many unit testing tools with the xUnit family for various languages, like Java, C#, C, or C++.

- JUnit is considered as a standard testing tool for unit tests of Java, although TestNG is another popular Java unit testing framework.

- NUnit is a testing framework that proposes similar operations like Junit.

- For C/C++, there is CppUnit, whereas PHP developers use PHPUnit.

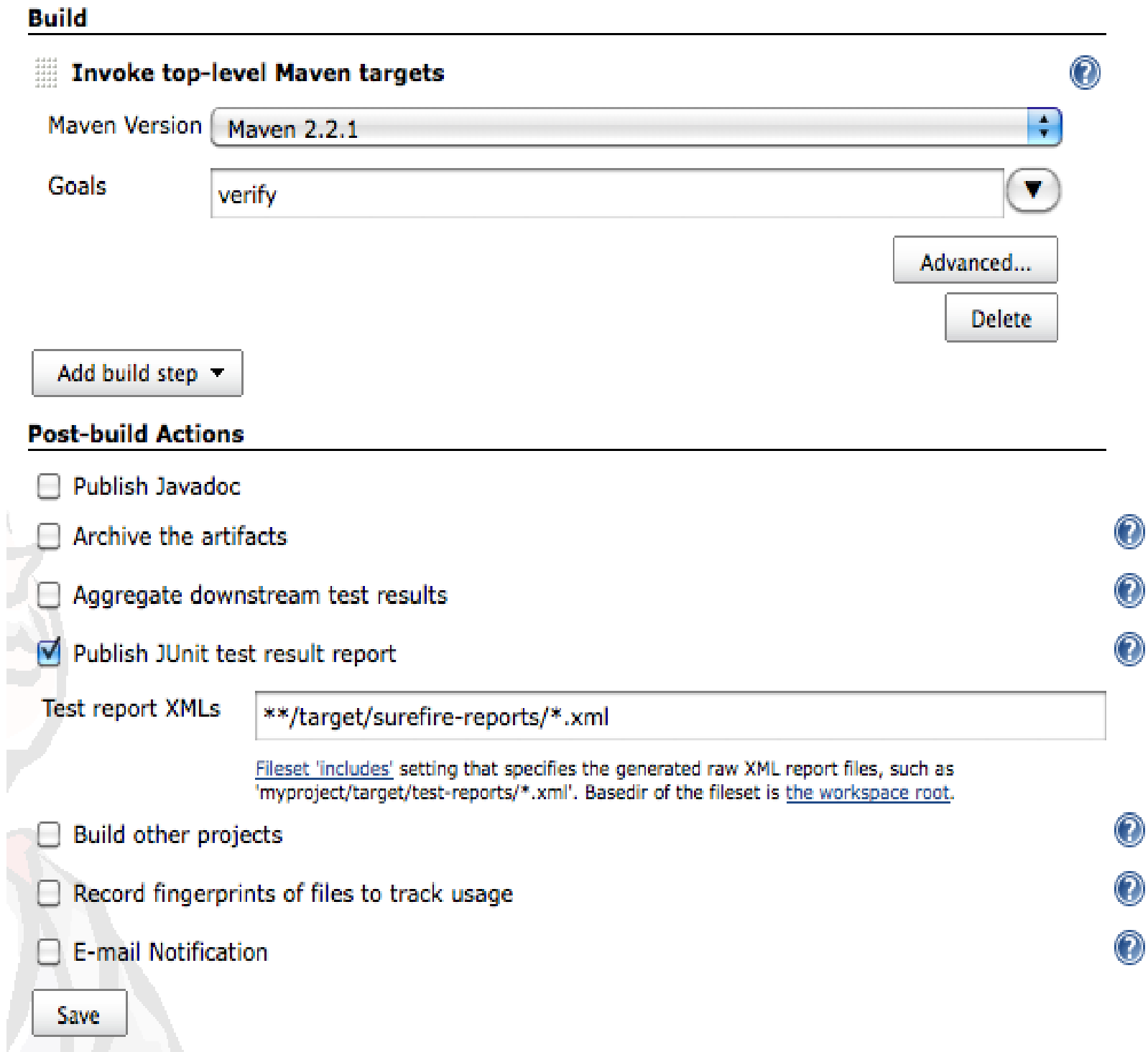# Configuring Test Reports in Jenkins

Jenkins doesn't require any plugin or external dependency for generating test reports. Below are the different changes to be made in the job properties for different frameworks:

- For Maven projects, invoke the goals field and type **verify** as shown in the figure below:
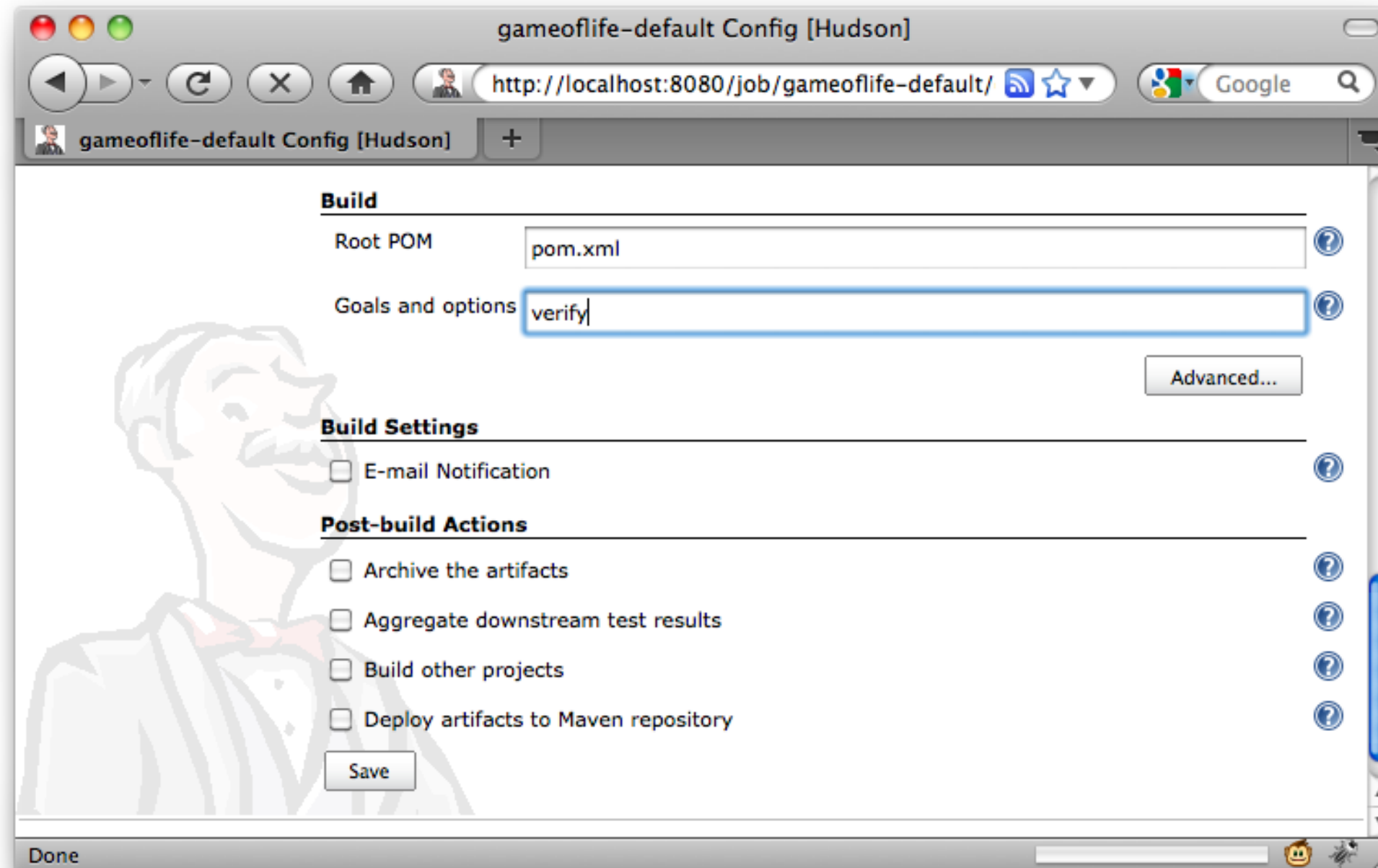
Configuring Test Reports in Jenkins

- For free style project, configure tests in the **Post Build Actions** section and tell Jenkins to publish the test report as shown in the figure:

- Directory of the test report XML will depend on the project. For a Maven project, a path like ***/target/surefirereports/*. pom.xml** is used.

- For an Ant-based project, report file location will depend on how you configured the Ant JUnit task.

Build

⣿ **Invoke top-level Maven targets** ⊙

Maven Version   Maven 2.2.1 ▲▼

Goals   verify ▼

Advanced...

Delete

Add build step ▼

Post-build Actions

☐ Publish Javadoc

☐ Archive the artifacts   ⊙

☐ Aggregate downstream test results   ⊙

☑ Publish JUnit test result report   ⊙

Test report XMLs   **/target/surefire-reports/*.xml

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.

☐ Build other projects   ⊙

☐ Record fingerprints of files to track usage   ⊙

☐ E-mail Notification   ⊙

Save

Configuring Test Reports in Jenkins

- For Java projects, regardless of the testing framework, Jenkins has in-built configurations for configuring test reports.

- If you are using Jenkins for non-Java projects, you need to install the **xUnit Plugin**.

- This plugin helps Jenkins process test reports from non-Java tools. It supports **MSUnit** and **NUnit** (for C# and other .NET languages), **UnitTest++** and **Boost Test** (for C++), **PHPUnit** (for PHP).

☐	This plugin provides an eXtreme Feedback Panel that can be used to expose the status of a selected number of Jobs.		1.0.8
☑	xUnit Plugin This plugin allows you to publish testing tools test result report.		0.6.1

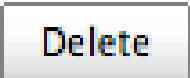Build Tools

Configuring Test Reports in Jenkins

- After installing **xUnit Plugin**, configure the reports in the **Post-build Actions** section.

- Check the **Publish testing tools result report** checkbox and enter the path to the XML reports generated.

- Jenkins will convert these reports to **JUnit** reports as soon as the build job runs to display in Jenkins.
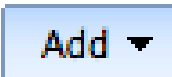

☑ Publish testing tools result report   ❓

Boost Test Library Pattern

boost/*.xml

Fail the build if test results were not updated this run ☑
Delete temporary JUnit files ☑

Delete

Add ▼
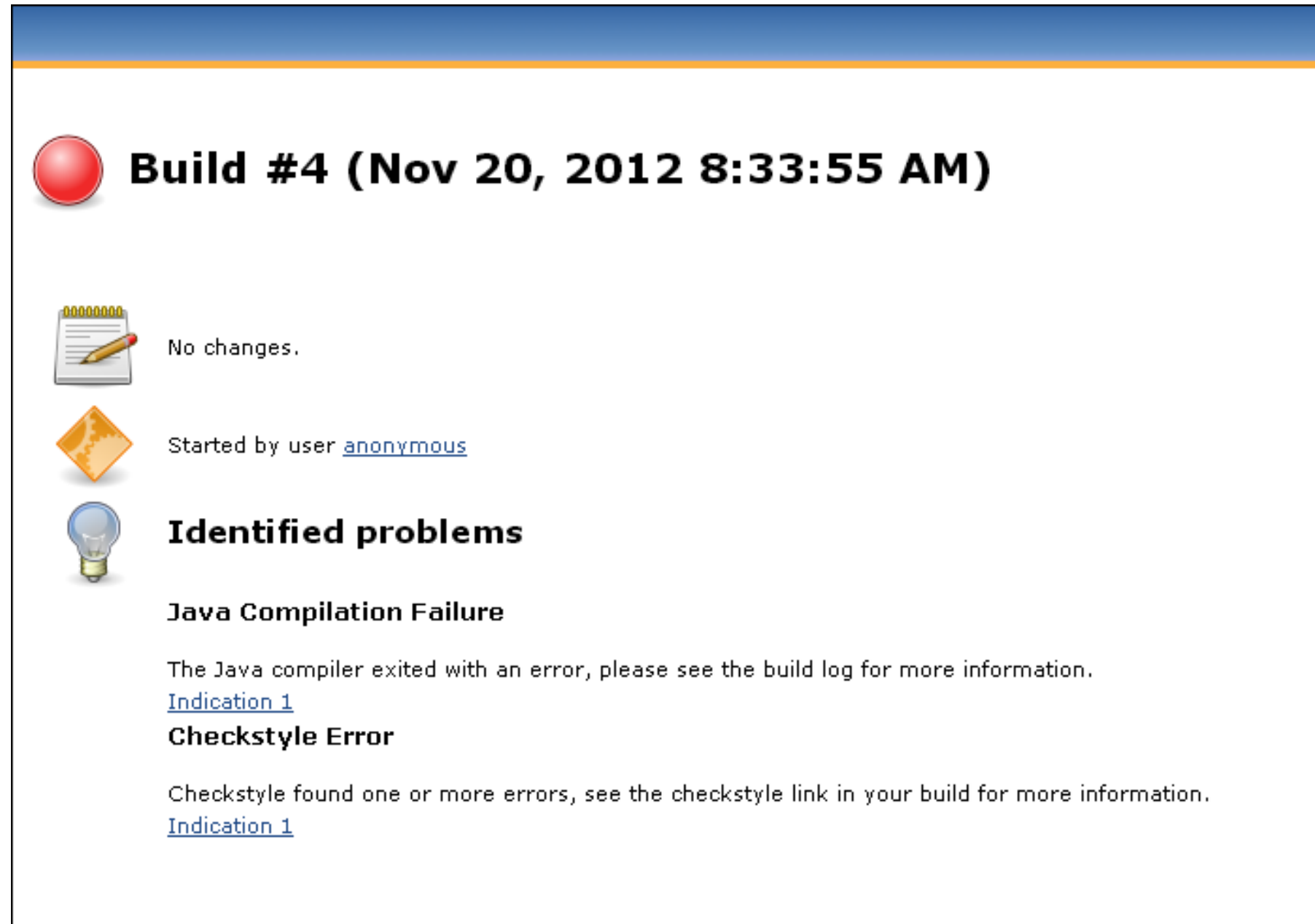
Custom Tool
NUnit
MSTest   container
Boost Test Library   ed build on other projects
UnitTest
Free Pascal Unit
PHPUnit

Caltech | Center for Technology & Management Education

simpli learn

Displaying Test Results

In Jenkins, a failed build (indicated by a red ball) indicates test failures, such as a compilation error. The below screenshot shows the build failure in Jenkins:

Displaying Test Results

Jenkins understands Maven project structures and initially shows a summary view of test results per module. The same is shown in the below screenshot:

Test Result

1 errors (±0) 2 failures (±0)

7 tests (±0)
Took 18 ms.
📝 add description

All Failed Tests

Test Name	Status	Duration	Age
>>> tbrugz.jenkinstest.TestIt.testFail1	Failed	12 ms	9
>>> tbrugz.jenkinstest.TestIt.testFail2	Failed	3 ms	9
>>> tbrugz.jenkinstest.TestIt.testError1	Error	1 ms	9

All Tests

Package	Duration	Error	(diff)	Fail	(diff)	Skip	(diff)	Total	(diff)
tbrugz.jenkinstest	18 ms	1		2		0		7	

Caltech | Center for Technology & Management Education

simplilearn

Assisted Practice
Generating Test Results

Problem Statement: You are given a project to generate and display test results in Jenkins.

Steps to perform:

1. Login to Jenkins
2. Adding Junit dependencies and classes in Maven project
3. Creating Jenkins job for Maven

Code Coverage with Cobertura

Ignoring Tests

Jenkins can identify the different tests that are ignored or failed. Below is an example of ignored test which uses **@ignore** annotation to indicate the compiler that a particular test is ignored. Jenkins also displays the information of the skipped tests as shown below:

```
@Ignore("Pending more details from the BA")
@Test
public void
cashWithdrawalShouldDeductSumFromBalance() throws
Exception {
        Account account = new Account();
        account.makeDeposit(100);
        account.makeCashWithdraw(60);
        assertThat(account.getBalance(), is(40));
}
```

Test Result

4 failures (+4) , 16 skipped (+16)

40 tests (+20)
Took 3 min 43 sec.
add description

Caltech | Center for Technology & Management Education

simplilearn

Code Coverage

Code coverage provides the information on parts of code in the application that were executed during the tests. It is an efficient way to identify the code that has not been tested during the execution of the application.

- Code coverage analysis is a CPU and memory-intensive process.

- Run code coverage operations in a separate Jenkins build job after unit and integration tests are built successfully.

- There are many code coverage tools available in Jenkins through dedicated plugins, such as:

1. For Java, it is Cobertura, Emma, JaCoCo and Clover
2. For .NET, it is NCover

JaCoCo

JaCoCo is a code java-based code coverage tool. It provides access to the **JaCoCo** runtime agent to analyze coverage data and generate code coverage report.

- JaCoCo is an open source toolkit that is distributed under Eclipse Public License.

- JaCoCo uses counters to calculate coverage metrics that are created based on the byte code of java files.

- The data derived from the byte code is mapped with the source code to the lowest granularity.

Measuring Code Coverage with JaCoCo

Counter divide the code into smaller unit based on the byte code data. Below are the four major classifications of data derived from Java byte code:

Instructions (C0 Coverage)
It is the smallest piece of byte code data mapped to the source code. It is an independent metric from the source formatting.

Cyclomatic Complexity
It indicates the complexity of the non-abstract methods, classes, and packages. This helps in determining number of unit test cases required.

Branches (C1 Coverage)
It covers the data from conditional statements and integrated branches in the source code. However, exception handling is not considered as a branch.

Caltech | Center for Technology & Management Education

simplilearn

JaCoCo with Maven

- As JaCoCo is a Java-based code coverage tool, it can be easily integrated with Jenkins and Maven.

- For Jenkins, navigate to **Manage Plugins->** search for **JaCoCo Plugin,** and install the same.

- Once Jenkins is set up, add the given code in **pom.xml** file to integrate JaCoCo with Maven project**.**

- A maven job in Jenkins can handle the build and generate corresponding report for code coverage.

```xml
<build>
<plugins>
<plugin>
<groupId>org.jacoco</groupId>
<artifactId>jacoco-maven-plugin</artifactId>
<version>${jacoco.version}</version>
<executions>
<execution>
<goals> <goal>prepare-agent</goal> </goals>
</execution>
<execution>
<id>report</id>
<phase>prepare-package</phase>
<goals> <goal>report</goal> </goals>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

Assisted Practice
Code Coverage with JaCoCo

Problem Statement: You are given a project to perform code coverage using JaCoCo.

Steps to perform:

1. Login to Jenkins.
2. Add JaCoCo plugin in Jenkins.
3. Add JaCoCo in Maven.
4. Create Jenkins job for Maven.

Code Coverage with Clover

Introduction to Clover

Clover is a code coverage tool from Atlassian.
It works well for projects with Maven, Ant, and Grails.

- **Clover** is open source and does not require a license tier.

- Its source code can be downloaded from https://bitbucket.org/atlassian/clover.

- To run Clover as open source, clone and compile locally.

- It uses a shift-left approach to testing.

- Clover works on Windows, Linux, and Mac OS X operating systems.

- It is used in Java and Groovy for code coverage.

- Redundant tests are noted through a **per-test** coverage feature.

Adding Clover in Maven

To add Clover in a Maven project, add the following Maven dependency in **pom.xml:**

```
<build>
...
<plugins>
...
<plugin>
<groupId>com.atlassian.maven.plugins</groupId>
<artifactId>maven-clover2-plugin</artifactId>
<version>3.0.4</version>
<configuration>
<includesTestSourceRoots>false</includesTestSourceRoots>
<generateXml>true</generateXml>
</configuration>
</plugin>
</plugins>
</build>
```

- This dependency enables and generates both HTML and XML reports, which include the combined data of a Maven project. It works even if the project contains different modules.

- To integrate Clover in Jenkins, download and install Clover plugin from the **Manage Plugin** option.

Running Clover

There are four major steps for executing Clover in the project:

Instrument your application
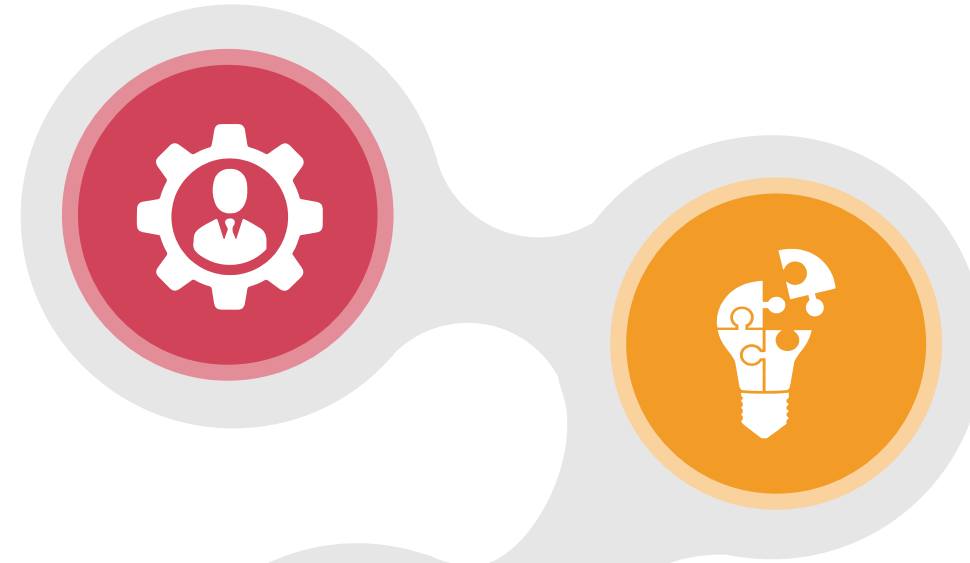
Run tests

Aggregate the test data

Generate reports

- This is a slow process and should be executed in a different Jenkins using the following script:
$ clover2:setup test clover2:aggregate clover2:clover

Caltech | Center for Technology & Management Education

simplilearn

Running Clover

Once the build job is complete, set up the Clover reporting in Jenkins as shown below:

- Select the **Publish Clover Coverage Report** and provide the path to the Clover HTML and XML report directory.

- One can also define the metrics for **Methods, Conditionals, and Statements**.

Clover Reports

Clover report is generated in the format shown below.
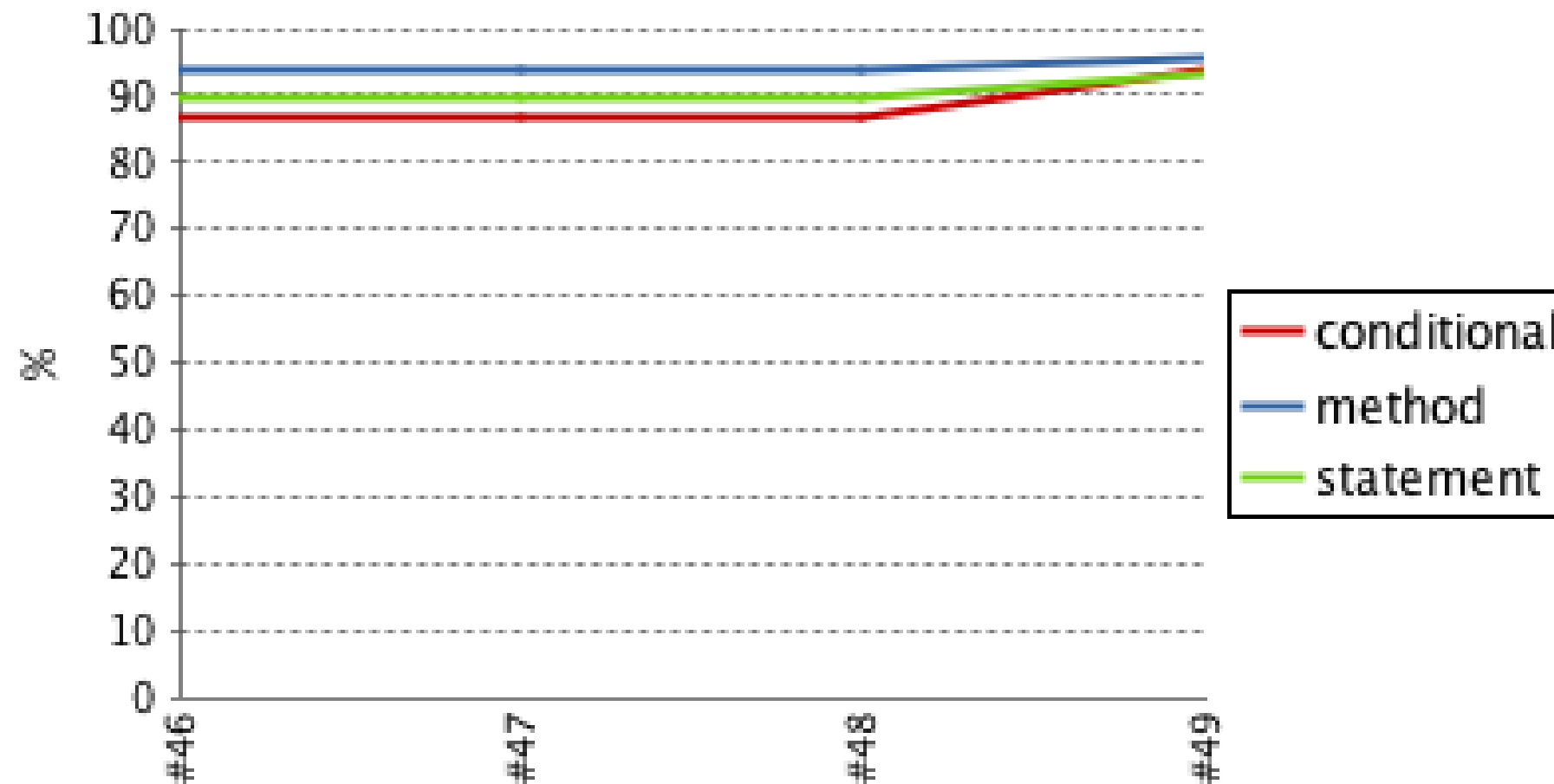
Assisted Practice
Code Coverage with Clover

Problem Statement: You are given a project to implement code coverage using Clover.

Steps to perform:

1. Login to Jenkins
2. Adding Clover in Jenkins and Maven project
3. Creating Jenkins job for Maven
4. Configuring SCM
5. Setting build goals

Acceptance and Performance Testing

Introduction to Acceptance Testing

Acceptance testing is used to evaluate the system's compliance with business requirements and assess its acceptability for delivery.

- Acceptance testing provides valuable information to stakeholders about the status of a project.

- Acceptance tests should be prototypes of the final project that demonstrates how the system works rather than test reports and other technical aspects.

- Acceptance tests can be automated using conventional tools such as JUnit.

- Behavior-Driven Development (BDD) frameworks can also be used as it is appropriate for the public-facing nature of acceptance tests.

User Acceptance Test

Caltech | Center for Technology & Management Education   simplilearn

Test Result : WhenTheUserEntersAnInitialGrid

0 failures

6 tests

Took 5.8 sec.

add description

All Tests

Test name	Duration	Status
theGridDisplayPageShouldContainANextGenerationButton	1.2 sec	Passed
theGridPageShouldHaveALinkBackToTheHomePage	1.1 sec	Passed
userShouldBeAbleChooseToCreateANewGameOnTheHomePage	1.5 sec	Passed
userShouldBeAbleToEnterLiveCellsInTheGrid	0.56 sec	Passed
userShouldBeAbleToEnterOneLiveCellInTheGrid	0.95 sec	Passed
userShouldBeAbleToSeedAnEmptyGridOnTheNewGamePage	0.39 sec	Passed

JMeter

Apache **JMeter** is an open-source software used for implementing load testing and measuring performance of websites.

- It can be used to simulate a heavy load on a server, group of servers, and network or object to test their strength and analyze overall performance.

- It is a stand-alone IDE that can be installed on the system.

- It works with all major types of applications.

- It has both CLI and graphical UI for performing operations.

- It generates a complete and ready to present HTML reports.

- It supports caching and offline analysis of test reports.

APACHE
JMeter™

Caltech | Center for Technology & Management Education

simplilearn

Performance Testing with JMeter

Performance testing can be used for determining how quickly an application responds to requests with a given number of simultaneous user or how well the application copes with an increasing number of users.

- JMeter simulates load on the application and measures the response time as the number of users and requests increase.

- JMeter runs as a Swing application and allows tester to configure and edit the test scripts.

- JMeter can run as a proxy and render the application on a browser, which generates initial version of the test script.

- Jenkins doesn't provide any plugin that is made specifically for Maven projects. However, one can install Maven-Ant plugin to call performance tests that are written in Ant.

Caltech | Center for Technology & Management Education

simplilearn

Assisted Practice

Integrate JMeter With Jenkins

Problem Statement: You are given a project to implement performance tests in Jenkins using JMeter

Steps to perform:

1. Install JMeter
2. Login to Jenkins
3. Adding performance plugin
4. Creating Jenkins job for JMeter

Key Takeaways

- One of the most efficient ways to perform automated testing is to write test scripts first using approach techniques.

- For an Ant-based project, report file location will depend on how you configured the Ant JUnit task.

- Run code coverage operations in a separate Jenkins build job after unit and integration tests are built successfully.

- Acceptance tests should be prototypes of the final project that demonstrates how the system works, rather than test reports and other technical aspects.

- JMeter can run as a proxy and render the application on a browser, which generates initial version of the test script.

Knowledge Check

Knowledge Check

1

What is automation testing?

A.   Automating the test cases

B.   Setting periodic builds in Jenkins

C.   Both of the above

D.   None of the above

Knowledge Check

1

What is automation testing?

A.   Automating the test cases

B.   Setting periodic builds in Jenkins

C.   Both of the above

D.   None of the above

The correct answer is **A**

Automation testing refers to automating the test cases when the application is compiled and built.

Knowledge Check

2

Which of the following is NOT a testing tool?

A. Selenium

B. Nunit

C. Junit

D. None of the above

Knowledge Check 2

Which of the following is NOT a testing tool?

A.   Selenium

B.   Nunit

C.   Junit

D.   None of the above

The correct answer is **D**

Selenium and Junit are used for Java, while Nunit is used for .Net testing.

What is Code Coverage?

A.   Limiting the test cases

B.   Adding execution constraints in Jenkins

C.   Checking the parts of code executed during the execution of the test cases

D.   None of the above

Knowledge Check

3

What is Code Coverage?

A. Limiting the test cases

B. Adding execution constraints in Jenkins

C. Checking the parts of code executed during the execution of the test cases

D. None of the above

The correct answer is **C**

Code coverage provides information on the parts of the code in the application that were executed during the tests.

Caltech | Center for Technology & Management Education

simplilearn

Knowledge Check

4

Which of the following tool is better in terms of reporting between clover and cobertura?

A. Cobertura

B. Clover

C. JaCoCo

D. None of the above

Knowledge Check

4

Which of the following tool is better in terms of reporting between clover and cobertura?

A.   Cobertura

B.   Clover

C.   JaCoCo

D.   None of the above

The correct answer is **B**

Clover is more efficient in generating code coverage reports than Cobertura.

Caltech | Center for Technology & Management Education

simplilearn

Knowledge Check 5

During an execution, which of the following annotations is used to ignore a test?

A.   @nonexecutable

B.   @ignore

C.   @notatest

D.   @untest

Knowledge Check 5

During an execution, which of the following annotations is used to ignore a test?

A. @nonexecutable

B. @ignore

C. @notatest

D. @untest

The correct answer is **B**

@ignore is the annotation used to ignore test while executing the application.

Problem Statement:

You're a DevOps engineer at HungryHippo, a web- and mobile-based app development company that helps users order food from various restaurants in their cities. The company is trying to analyze the review sentiments and classify the reviews accordingly. The backend developers are developing a project that will extract words from reviews and analyze it to identify the sentiment. You are required to develop a microservice that counts the words in a given review and design a full Jenkins pipeline to compile a Spring Boot application with Maven, test with JUnit, and evaluate the code coverage using JaCoCo. Your pipeline has to compile this code, run a unit test, evaluate the code coverage, and publish the results. The pipeline is expected to scale and run seamlessly as more microservices are added to the project. The pipeline should be defined as a Jenkinsfile and be committed to the SCM.