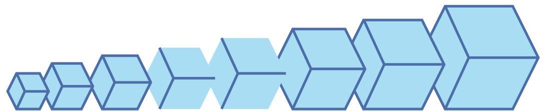
Course Objectives

At the end of this course, you will be able to:

- Understand Continuous Integration
- Introduction to Jenkins
- Installation of Jenkins
- Creation of build job using Jenkins
- Automated Testing using Jenkins
- Creation of Pipeline in Jenkins

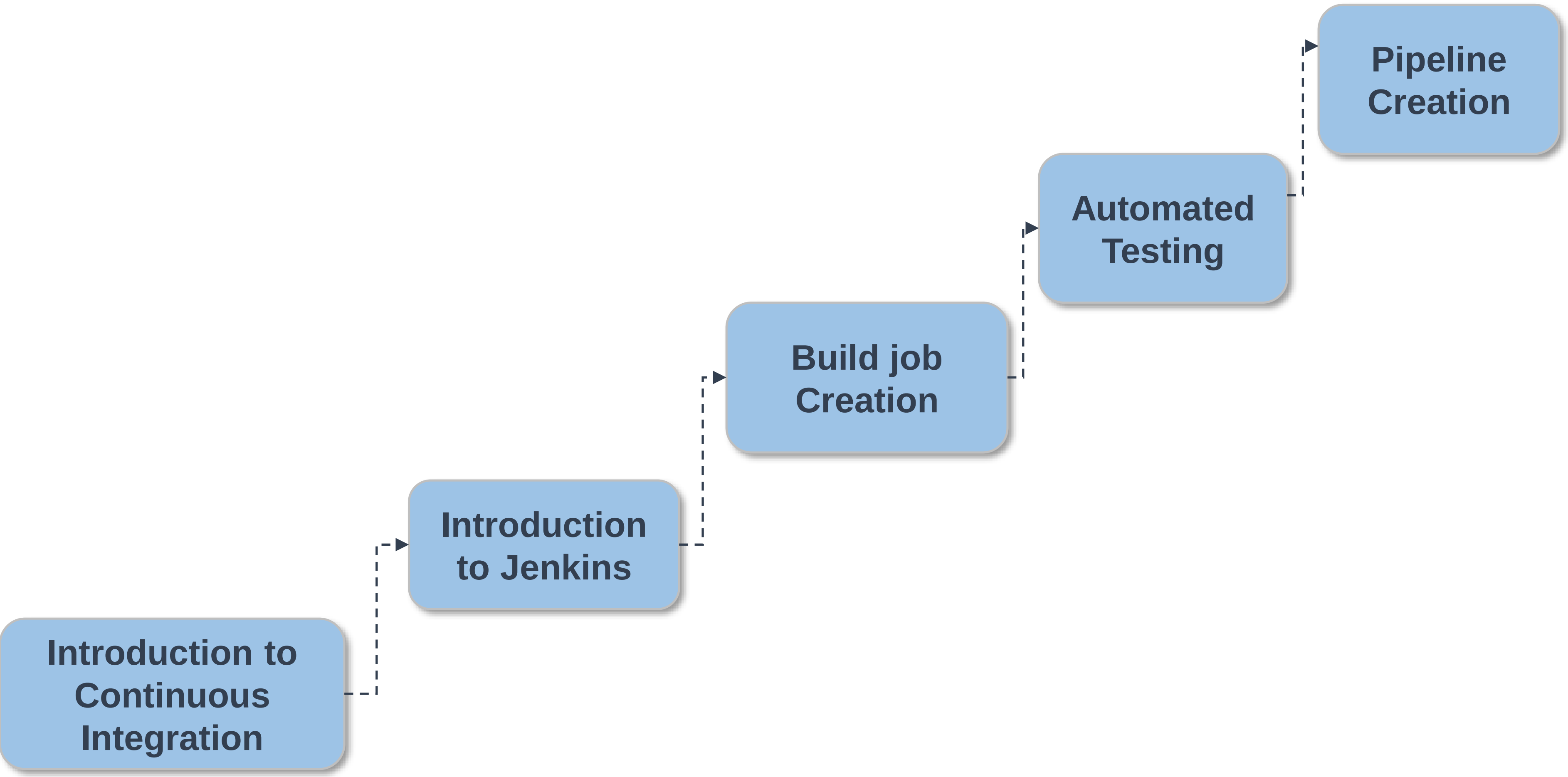


Ground Rules

For a successful class, please:

- Arrive on time
- Turn all cell phones off
- Wear business formal attire
- Assist your colleagues; show respect to all individuals regardless of their skill and knowledge level
- Do not use class time to surf the net, check e-mail, or use instant messaging
- Adhere to attendance policy as directed by your local training coordinator

Module Map


Introduction to Continuous Integration
Introduction to Jenkins
Build job Creation
Automated Testing
Pipeline Creation

Course Overview

Course Description

This course introduces to Continuous Integration and Jenkins. The details about Jenkins installation, creation of build job and pipeline are covered.

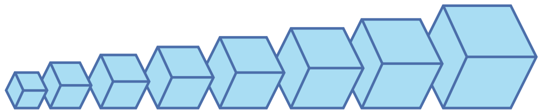
Audience

The intended audience for this course are:

- Engineers who want to automate the process.

Prerequisites

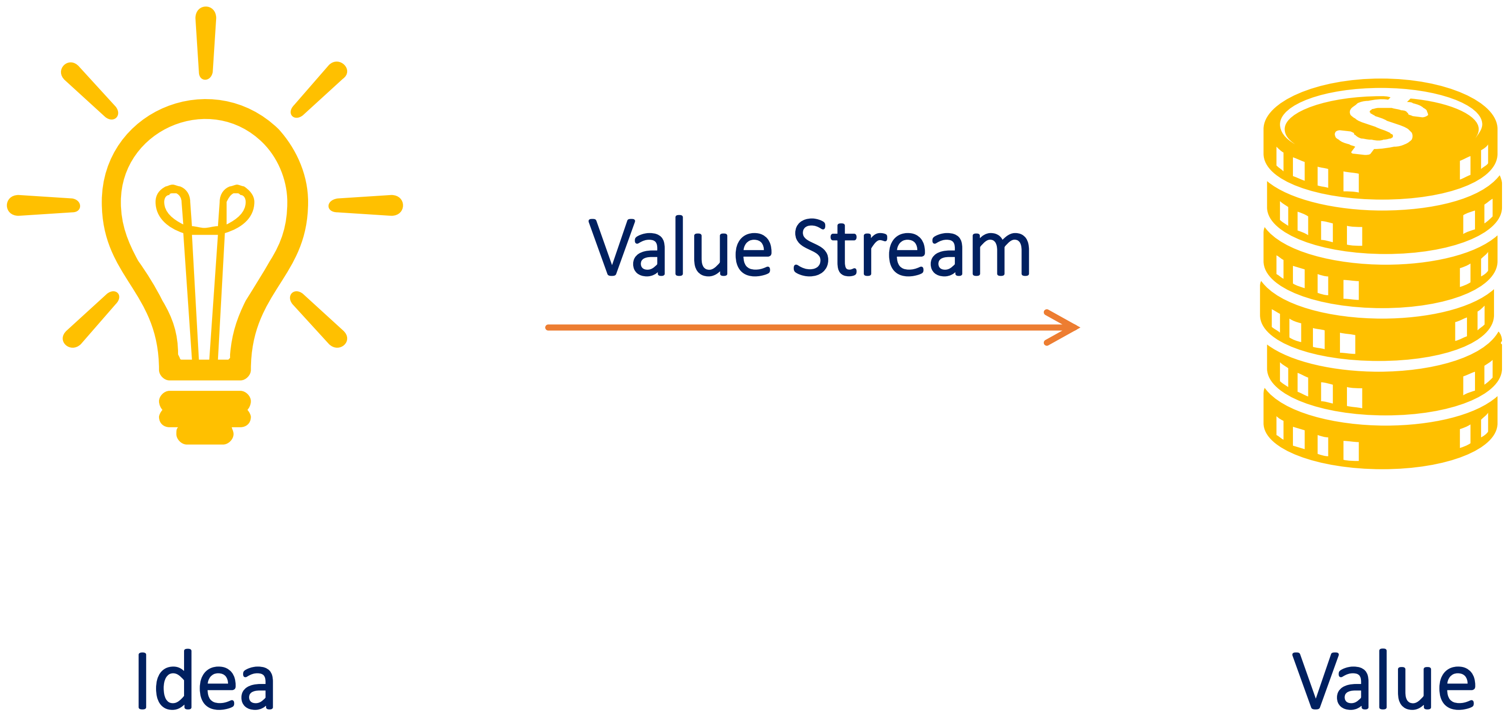
- Any one Version Control System
- Ant/Maven



Introduction to CI

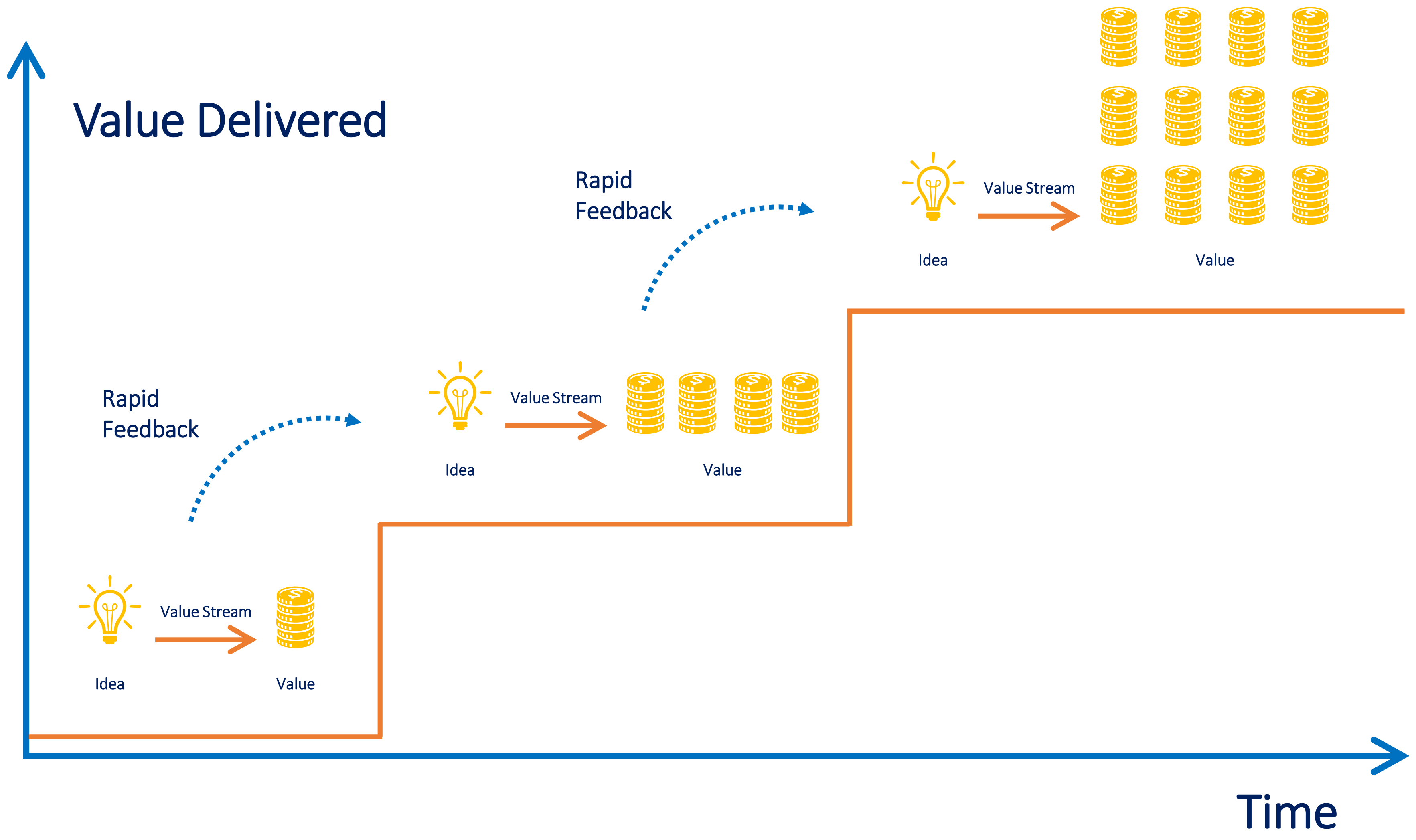
- *“Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.”* ***– Martin Fowler 01 May 2006***

http://www.martinfowler.com/articles/continuousIntegration.html


Value Stream
Idea
Value

Ideas

Values

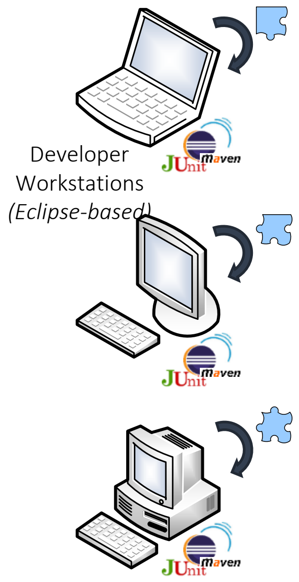

Value Delivered
Rapid Feedback
Rapid Feedback
Value Stream
Value Stream
Value Stream
Idea
Value
Idea
Value
Idea
Value
Time

What does this means?

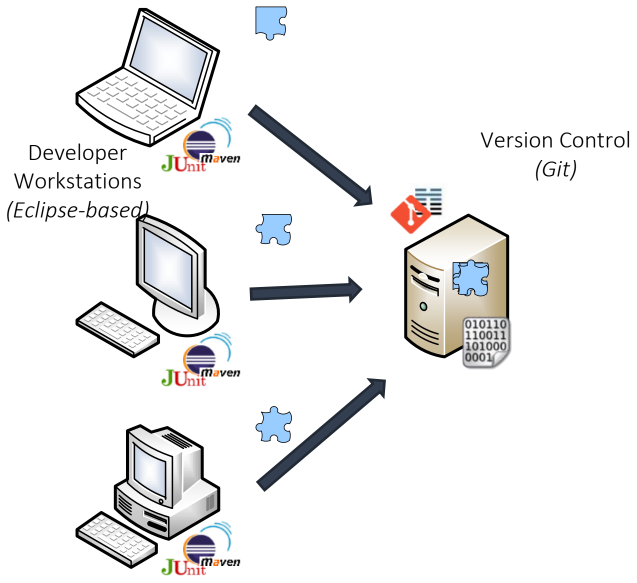
At a regular frequency, the following operations will be performed:

- Integration
- Build
- Testing
- Deployment

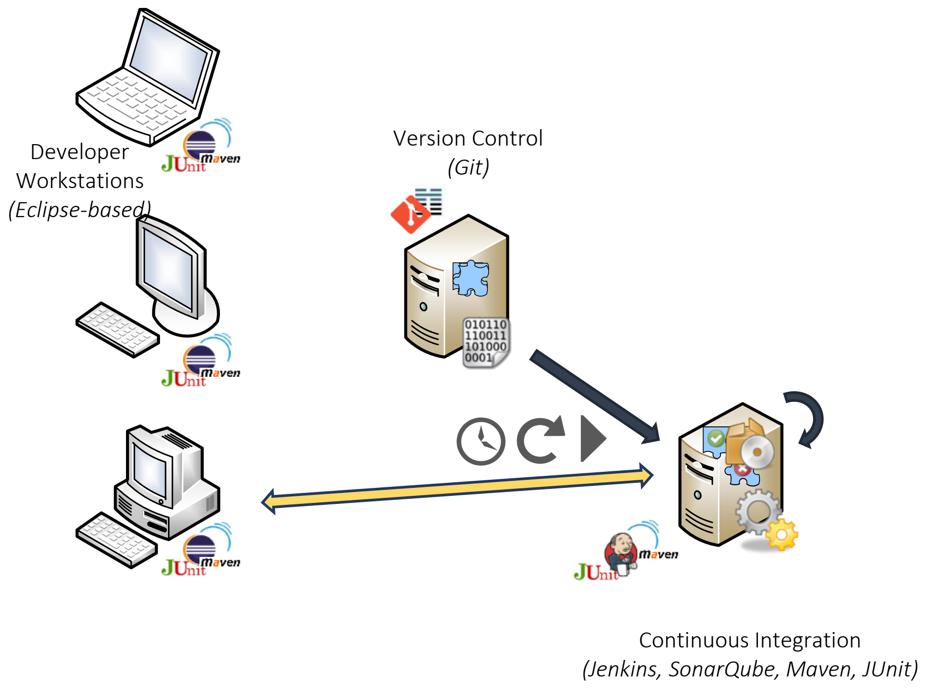
Sequence of a CI workflow



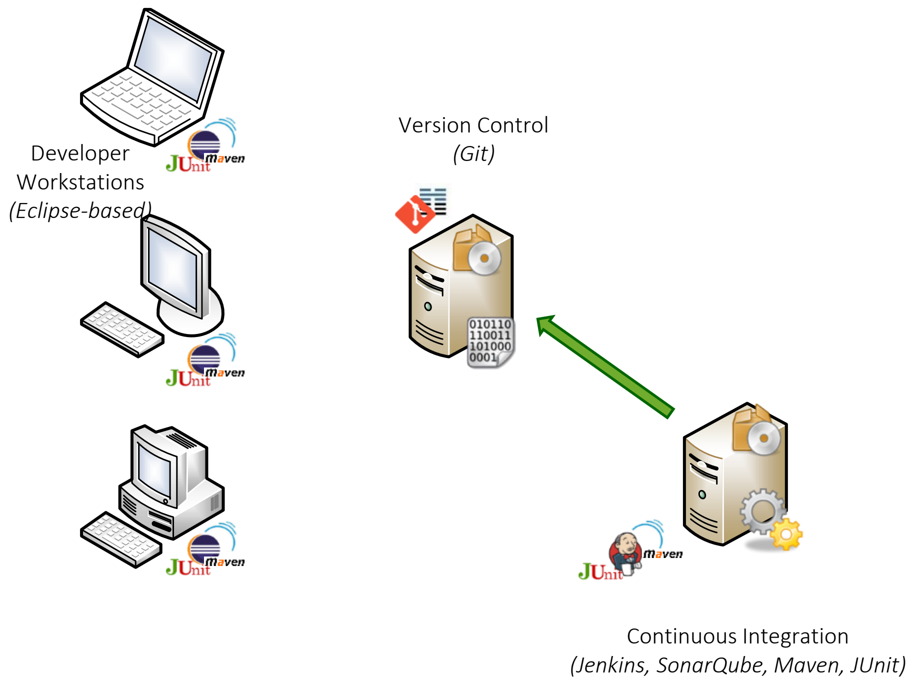
Sequence of a CI workflow



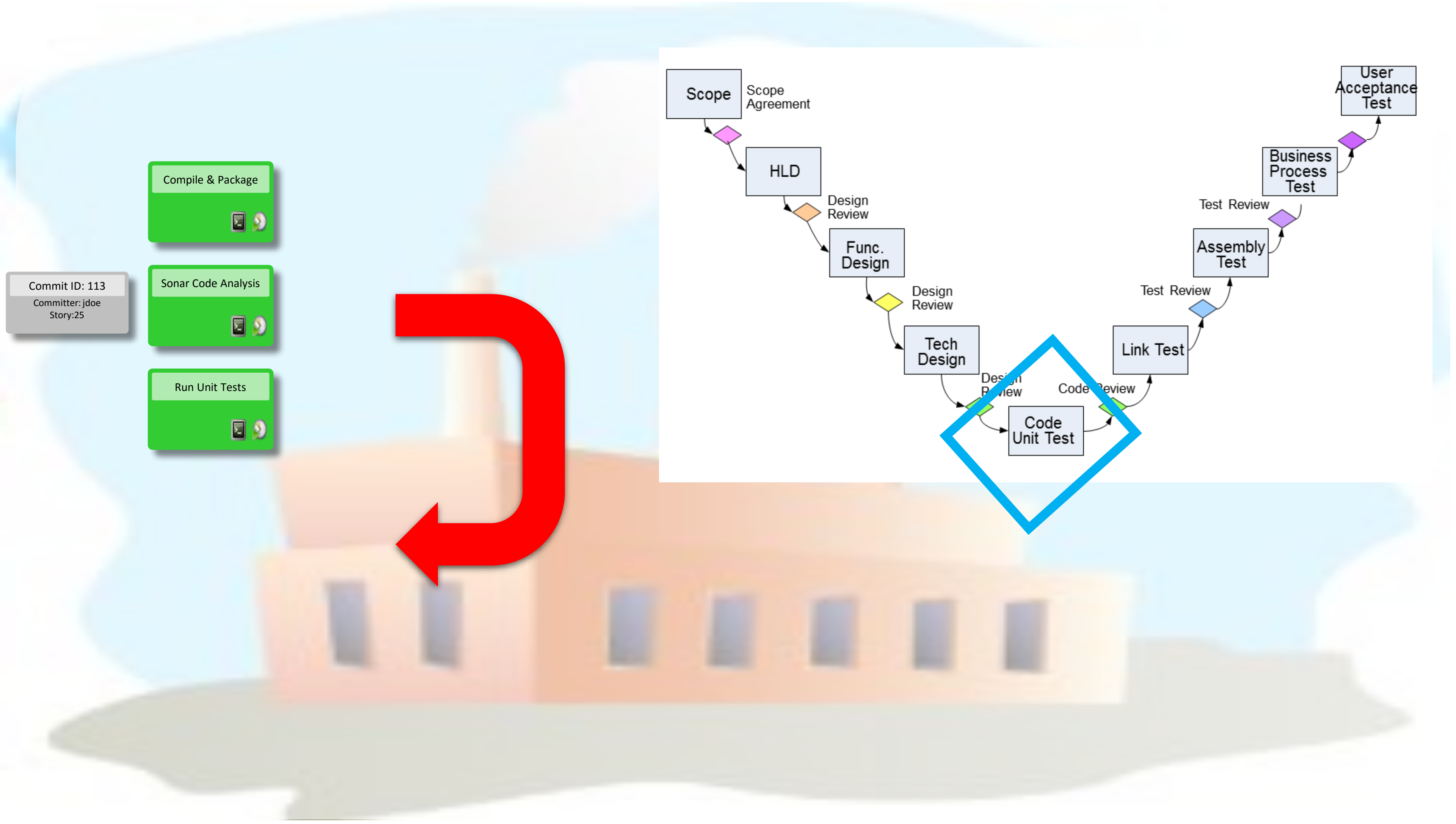
Sequence of a CI workflow



Sequence of a CI workflow



Continuous Integration



Benefits of CI

Benefits of CI

- Improve Productivity by enabling the automation of build and testing of an application.
- Early detection of build failure/defects
- Test case failure can be reported at the earliest and clearly
- Makes process more visible for everyone.

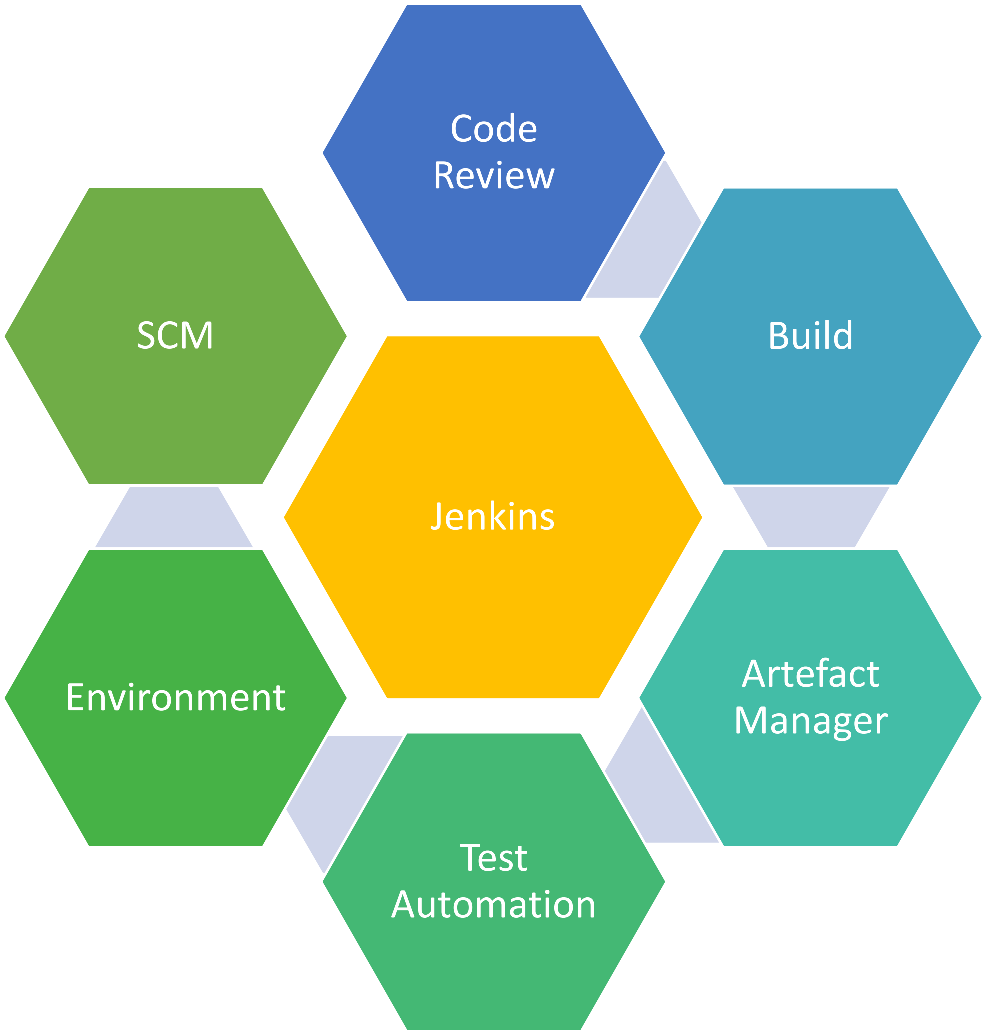
Content Outline

- Introduction to CI
- Introduction to Jenkins
- Build job Creation
- Automated Testing
- Pipeline Creation

Introduction to Jenkins

- An open source Continuous Integration (CI) tool written in Java.
- Jenkins is used to perform the below steps:
  - To automate build process by creating build job
  - Configure build by integrating with Apache Maven or Gradle
  - Integrate with version control systems
  - Publish Test result
  - Configure trigger to incremental process improvements.
- Jenkins can be installed either at command line or run as Java web application on container such as Tomcat
- Installable on most OS and compatible with many popular version control systems(SVN, Git)

CI Tool


Code Review
SCM
Build
Jenkins
Environment
Artefact Manager
Test Automation

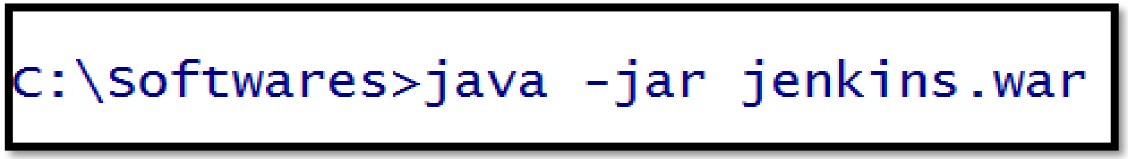
Jenkins Plugins

- Jenkins supports 400+ plugins in different categories to automate the process.
- List of few plugins:

SCM plugins	Build Tools	Test Plugins	Static Analysis
• Git	• Ant	• Junit	• Checkstyle
• Bazaar	• Maven	• Nunit	• PMD
• Bit Keeper	• MSBuild	• Selenium	• FindBugs
• Sub Version	• Cmake	• Fitnesse	• SonarQube
• TFS	• Gradle	• TestNG	• Fxcop
• Clear Case	• Grails	• Cucumber	• Code Scanner
• Visual Source safe	• Scons		
	• Groovy		

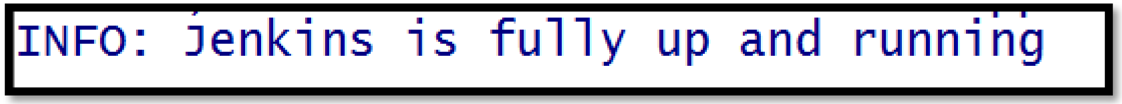
Download and Install Jenkins

- Navigate to https://jenkins-ci.org/
- Download jenkins.war and store it in your local drive. For Example, c:\softwares



```
c:\softwares>java -jar jenkins.war
```

- Open Command prompt and type the below command to install Jenkins
- Jenkins will be ready to use, after getting the below message in command prompt

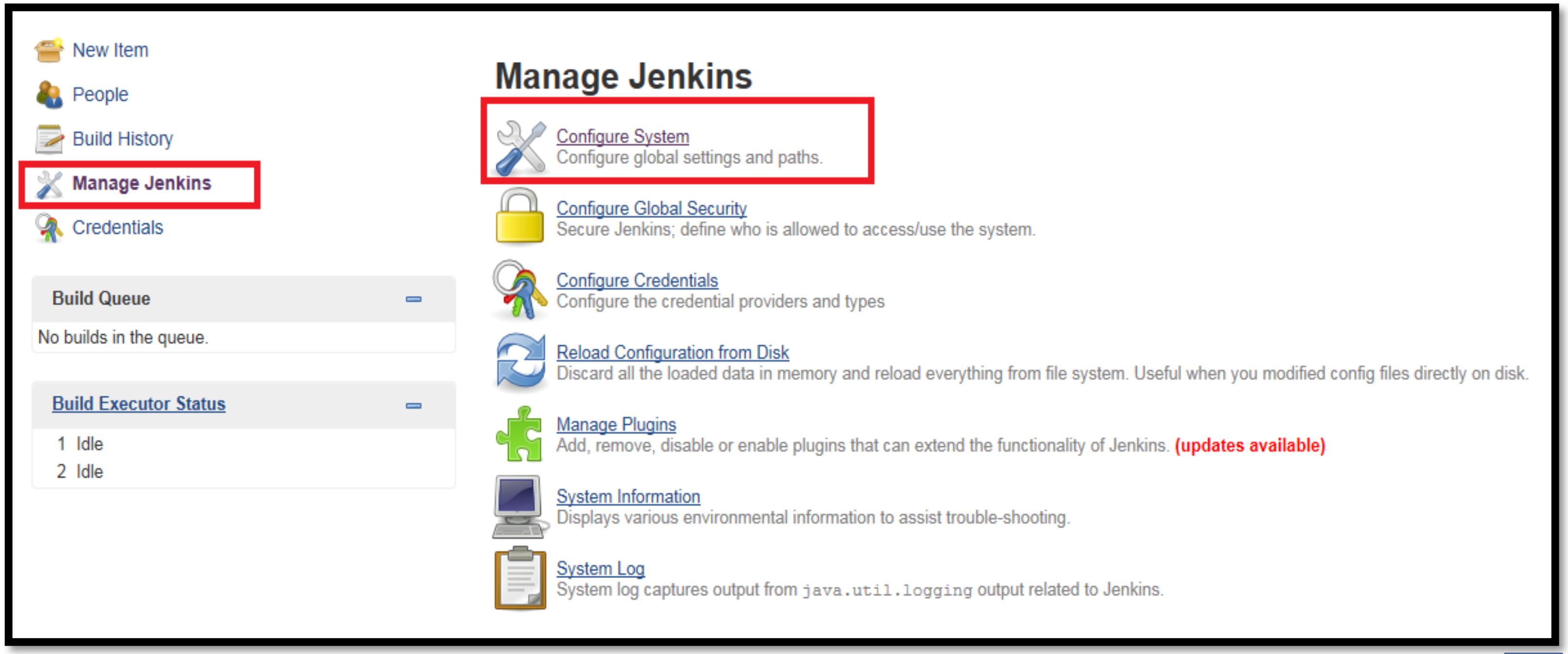


```
INFO: Jenkins is fully up and running
```

- By default, Jenkins will get started in the port no 8080.
- Jenkins Dashboard will be accessible using the below path
  - http://localhost:8080

Configure Jenkins

- After Jenkins is installed, JDK, Maven and Git has to be configured.
- Follow the below steps to start with configuration.
    - In Jenkins dashboard, Click on Manage Jenkins and then click on Configure System.



No builds in the queue.

1 Idle
2 Idle

Manage Jenkins

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

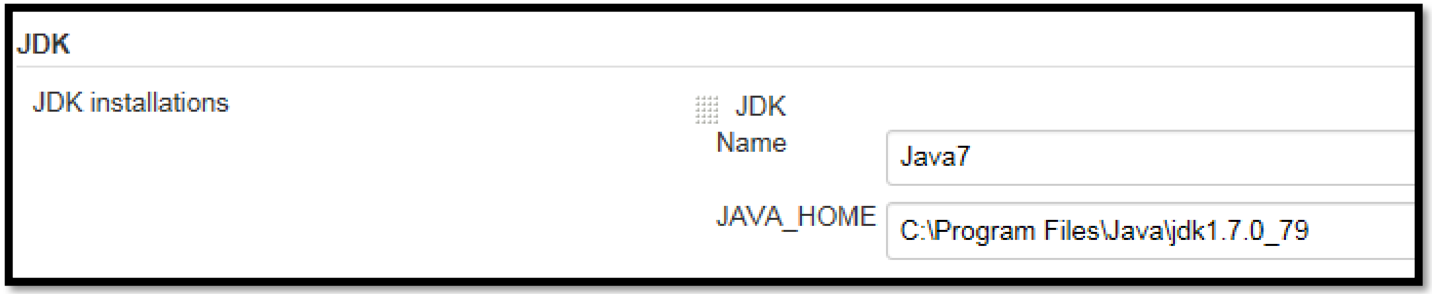
Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins. **(updates available)**

System Information
Displays various environmental information to assist trouble-shooting.

System Log
System log captures output from `java.util.logging` output related to Jenkins.

Configure Jenkins

- Navigate to JDK section and Type Name for the JDK. For example: Java7
- Specify the path of the JDK installed in the system in JAVA_HOME field.



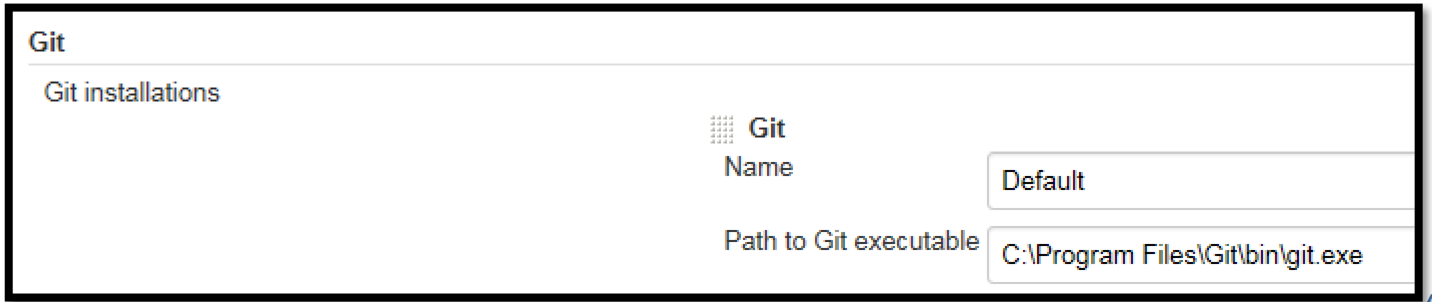
JDK

JDK installations

JDK

Name	Java7
JAVA_HOME	C:\Program Files\Java\jdk1.7.0_79

- Navigate to Git section and Type Name for the Git(Optional).
- Specify the path of the Git installed in the system in Path to Git executable field.



Git

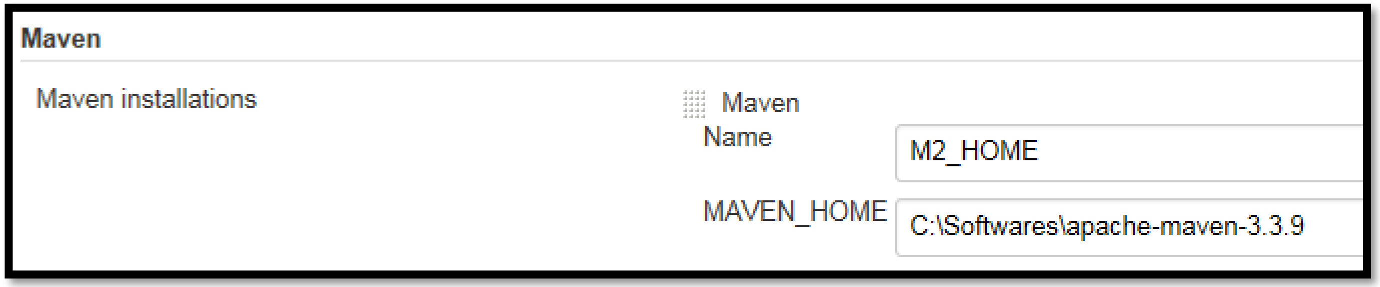
Git installations

Git

Name	Default
Path to Git executable	C:\Program Files\Git\bin\git.exe

Configure Jenkins

- Navigate to Maven section and Type Name for the Maven. For example: M2_HOME
- Specify the path of the Maven installed in the system in MAVEN_HOME field.



Maven

Maven installations

Maven

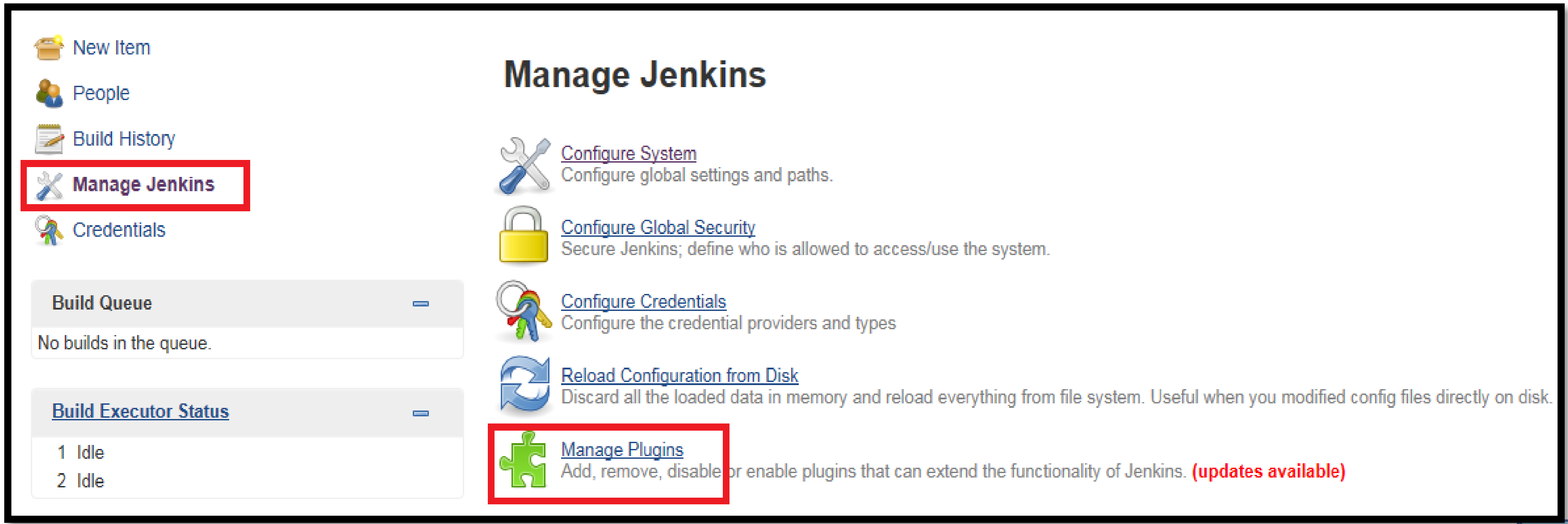
Name: M2_HOME

MAVEN_HOME: C:\Softwares\apache-maven-3.3.9

- After configuring JDK, Maven and Git, Click on save button.

Installing Plugins

- Jenkins can be extended with its support by installing plugins to support with Version Control System, integrate with Maven, to publish test results.
- Follow the below steps to install plugins:
    - In Jenkins dashboard, Click on Manage Jenkins and then click on Manage Plugins



Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Manage Jenkins

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

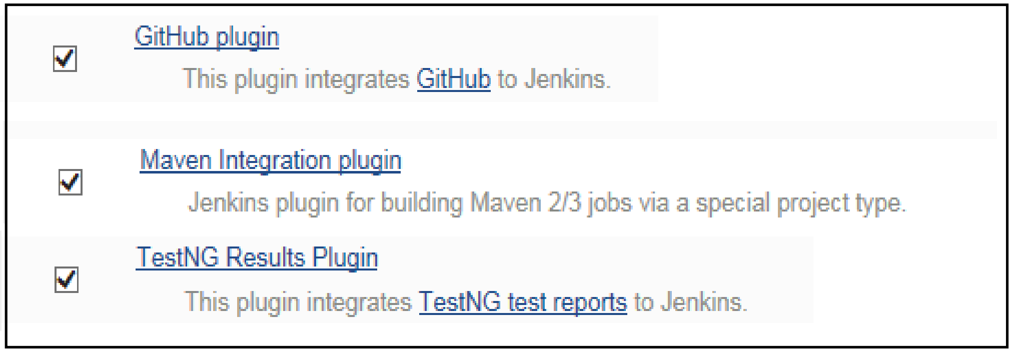
Configure Credentials
Configure the credential providers and types

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins. **(updates available)**

Installing Plugins

- Click on Available tab in Manage Plugins and then select the below plugins



☑ GitHub plugin
This plugin integrates GitHub to Jenkins.

☑ Maven Integration plugin
Jenkins plugin for building Maven 2/3 jobs via a special project type.

☑ TestNG Results Plugin
This plugin integrates TestNG test reports to Jenkins.

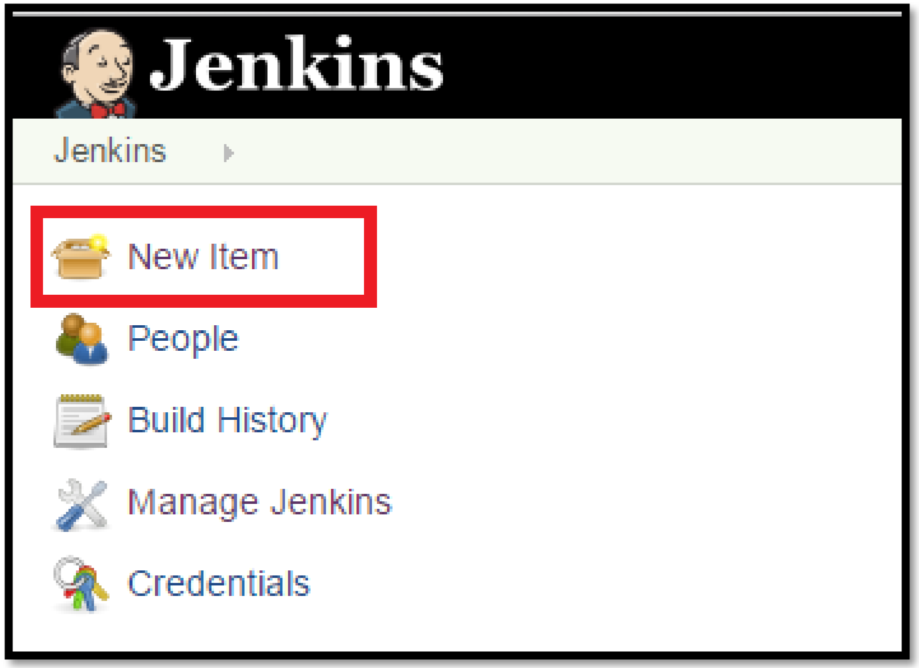
- After selecting the list of plugins, Click on “Download now and install after restart”.
- All the selected plugins will be installed and ready to use in Jenkins.

Content Outline

- Introduction to CI
- Introduction to Jenkins
- Build job Creation
- Automated Testing
- Pipeline Creation

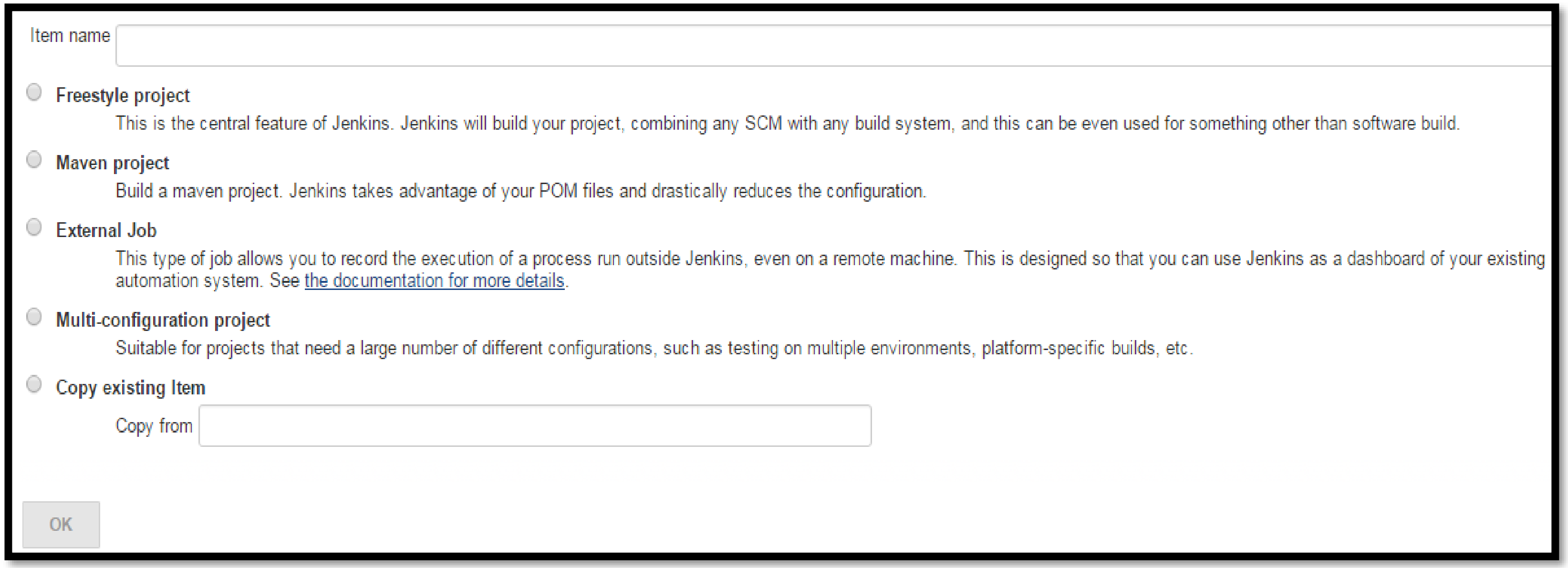
Build job Creation

- In the form of creating jobs, each process of the project to be performed will be configured.
- Steps to be followed to create job:
  - Access Jenkins Dashboard using the URL http://ipaddress:8080
  - Select New Item from the menu as highlighted in the below image



Build job Creation

- Type job name and select type of project as Freestyle/Maven project.
- Click Ok to successfully create build job and Jenkins will display the project configuration screen immediately.



Item name

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.

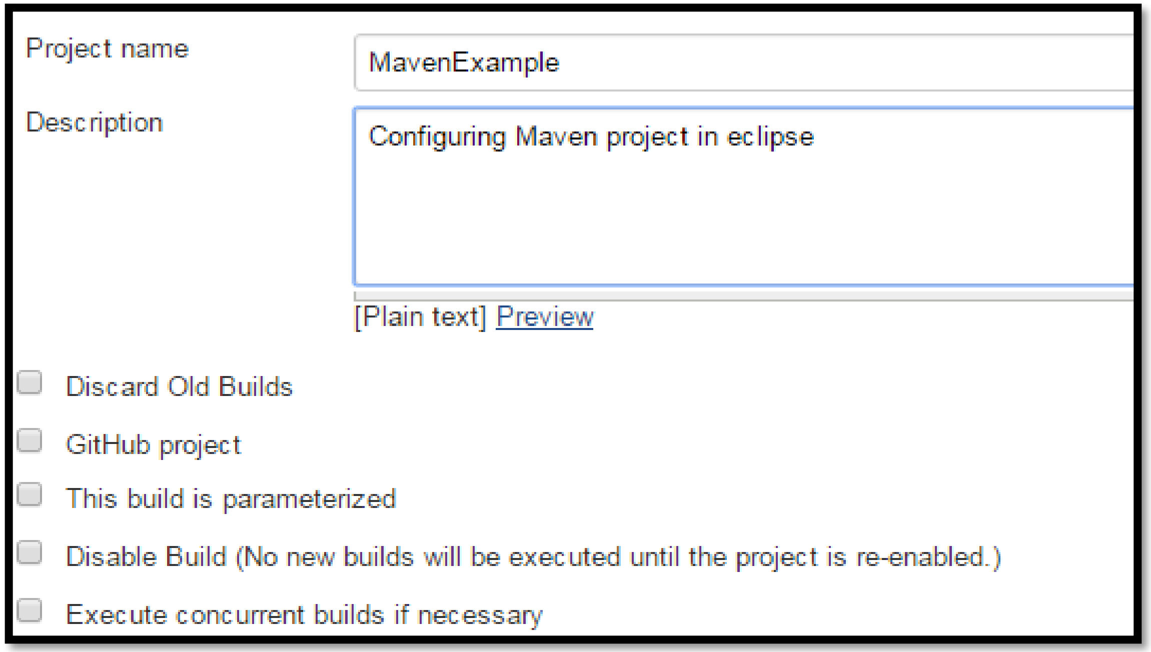
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Copy existing Item
Copy from

- Jenkins supports different types of build jobs out of which one should be selected during build job creation

Configure build job

- Type the description of the project in “Project Configuration” page as shown below.
- All other configurations are optional.



Project name: MavenExample

Description: Configuring Maven project in eclipse

[Plain text] Preview

☐ Discard Old Builds

☐ GitHub project

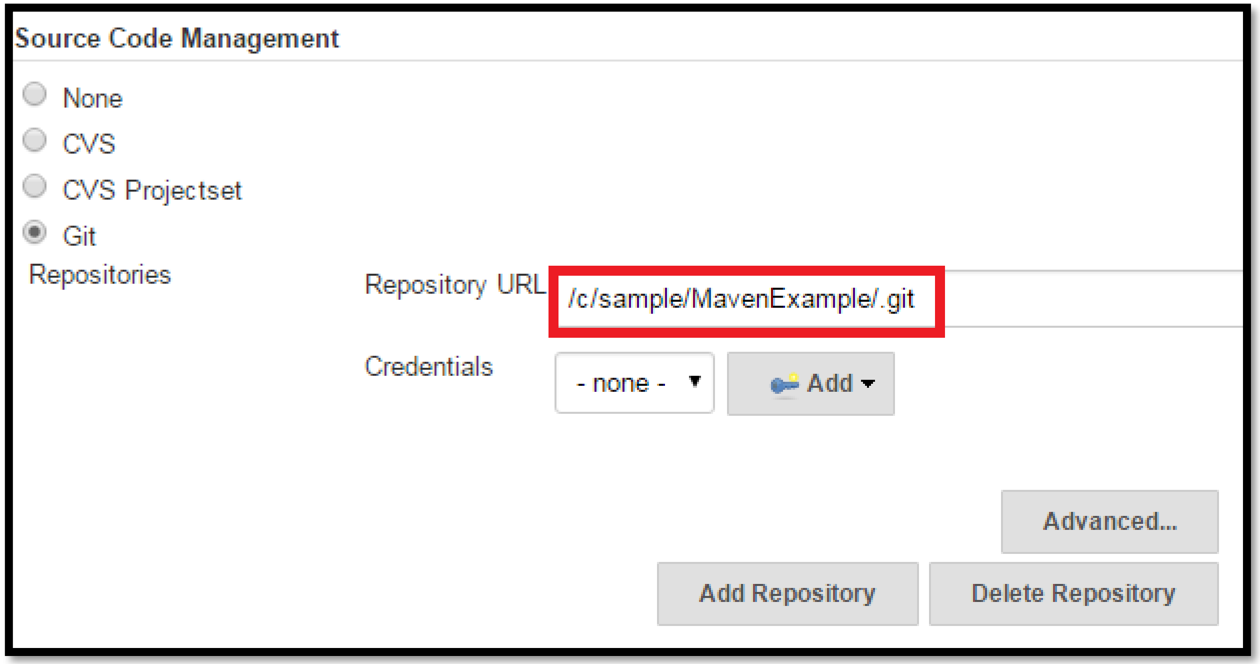
☐ This build is parameterized

☐ Disable Build (No new builds will be executed until the project is re-enabled.)

☐ Execute concurrent builds if necessary

Configuring SCM

- Navigate to Source Code Management section
- Select "Git" in SCM section
- Specify Repository URL



Source Code Management

- None
- CVS
- CVS Projectset
- [✓ Git]

Repositories

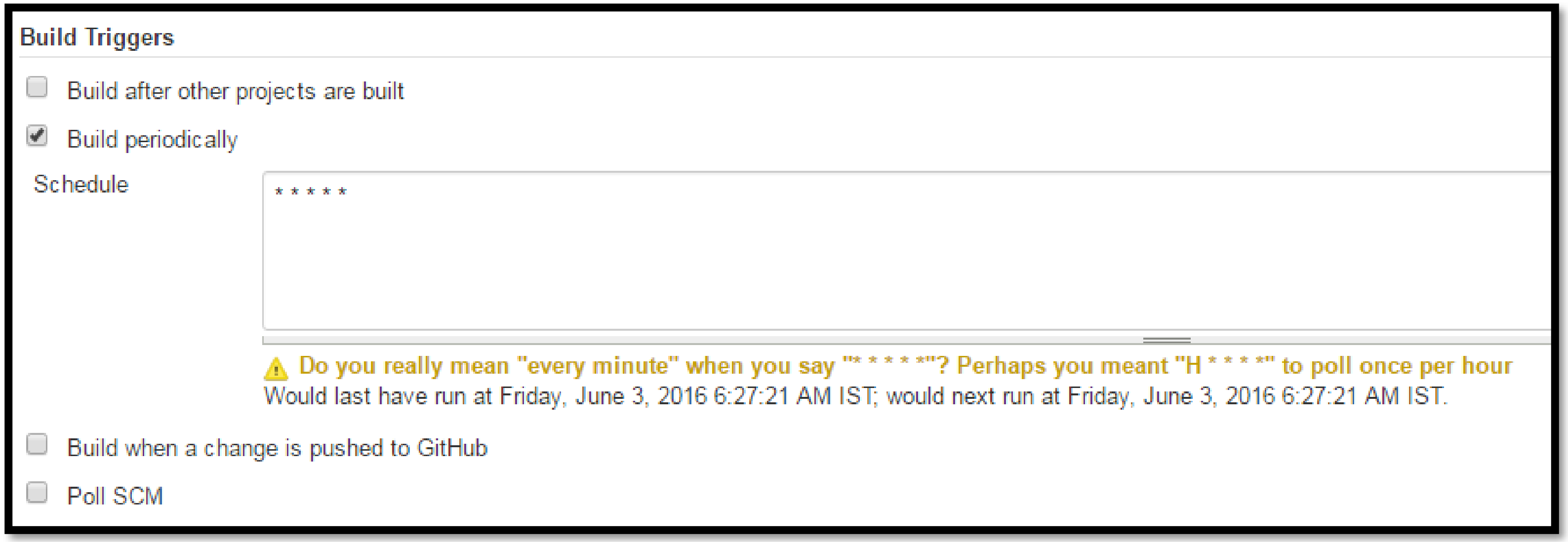
Repository URL: /c/sample/MavenExample/.git

Credentials: - none -

Scheduling build job

Schedule build job triggering by following the below steps:

- Navigate to Build Triggers section
- To initiate build for every minute, Select "Build Periodically" and type * * * * *(Min Hours Day Month Dayweek) in the schedule as shown below.



Build Triggers

☐ Build after other projects are built

☑ Build periodically

Schedule: * * * * *

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
Would last have run at Friday, June 3, 2016 6:27:21 AM IST; would next run at Friday, June 3, 2016 6:27:21 AM IST.

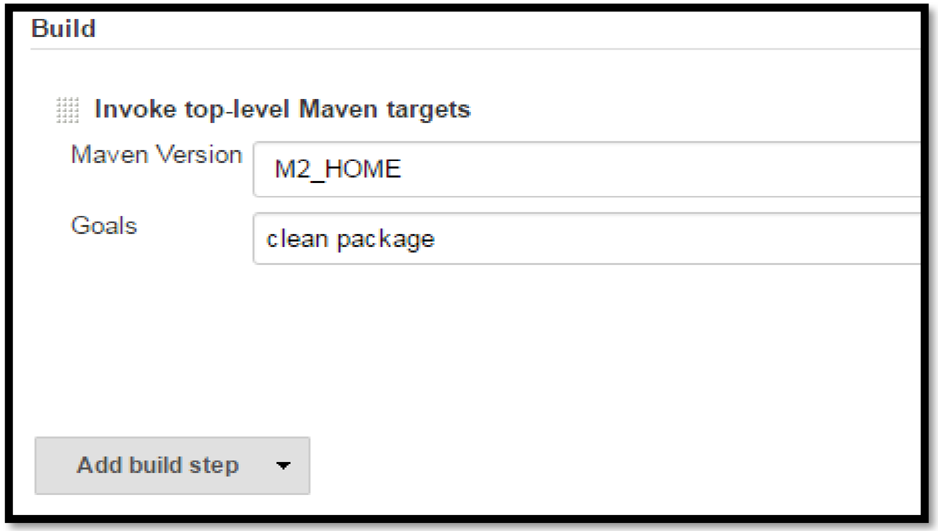
☐ Build when a change is pushed to GitHub

☐ Poll SCM

- For example, if you want to build your job for every 6 hours of a day, then expression should be used like this 0 6 * * *.

Configuring Maven in Jenkins

- To build job using maven commands, do the following steps.
  - Navigate to Build section
  - Choose "Invoke top-level Maven target" from "Add build step" drop down list.
  - Specify the Maven version and type target name as shown below to execute clean and package goals in Maven


Build

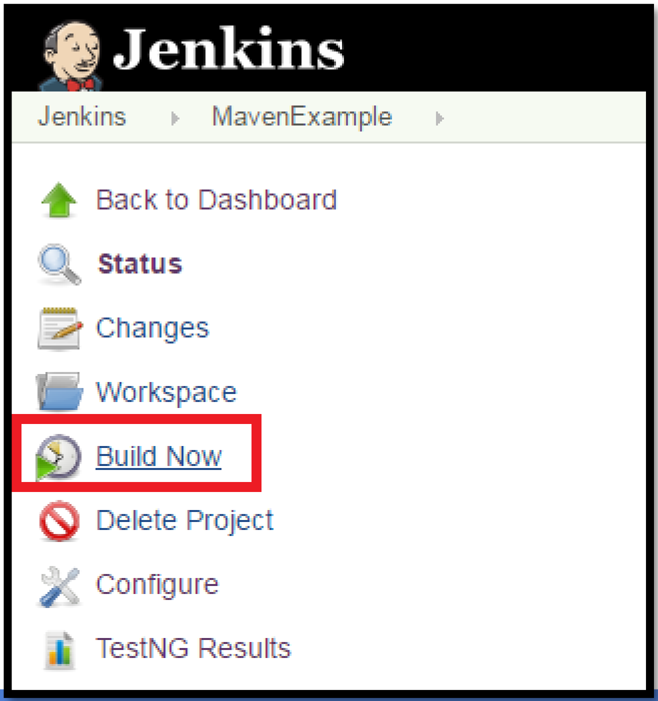
Invoke top-level Maven targets

Maven Version: M2_HOME

Goals: clean package

Execute Build job

- Once configurations are completed, execute build job by following the below steps:
  - Click Save.
  - Schedule the build to be executed immediately by clicking on “Build Now” link.



Content Outline

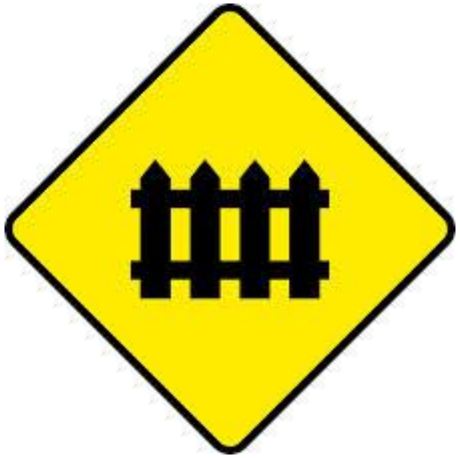
- Introduction to CI
- Introduction to Jenkins
- Build job Creation
- Automated Testing
- Pipeline Creation

Automated Testing

- To verify a build in Continuous Integration, automated testing is more suitable.

  Types of Quality gates:

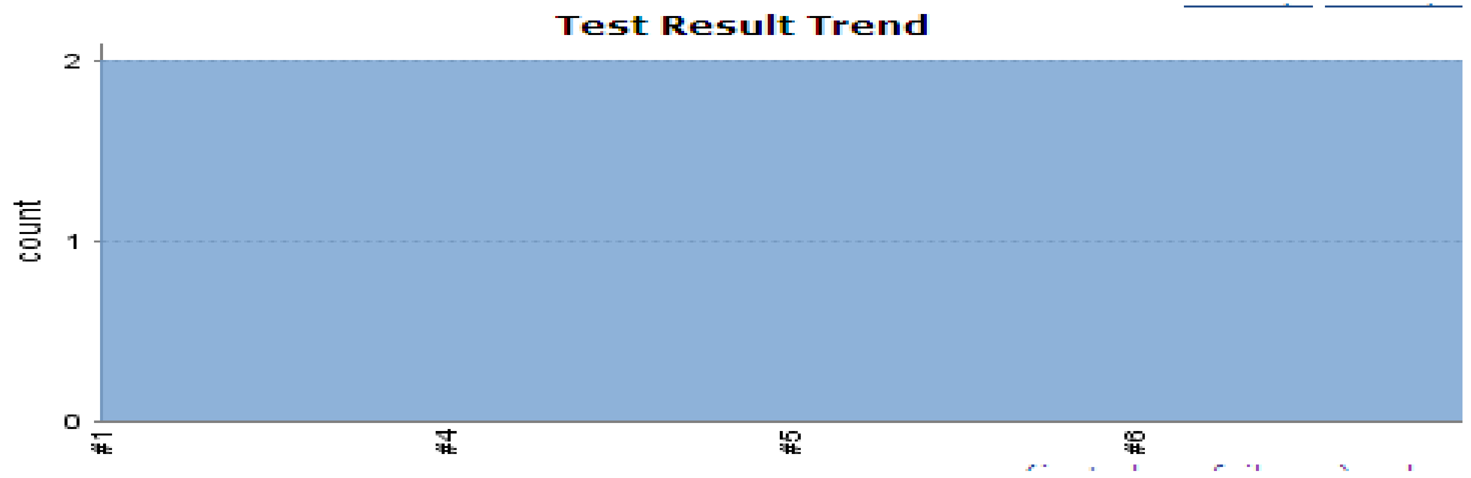
  - Static
  - Unit test
  - Functional
  - Non-functional
  - Canary testing



- For performing Unit testing of Java code, JUnit testing tool is the de facto standard .
- Jenkins does an excellent job of reporting on your test results.
- Jenkins supports with all types of testing like unit testing, integration testing, web testing, functional testing, performance testing, load testing and so on.

Configuring JUnit

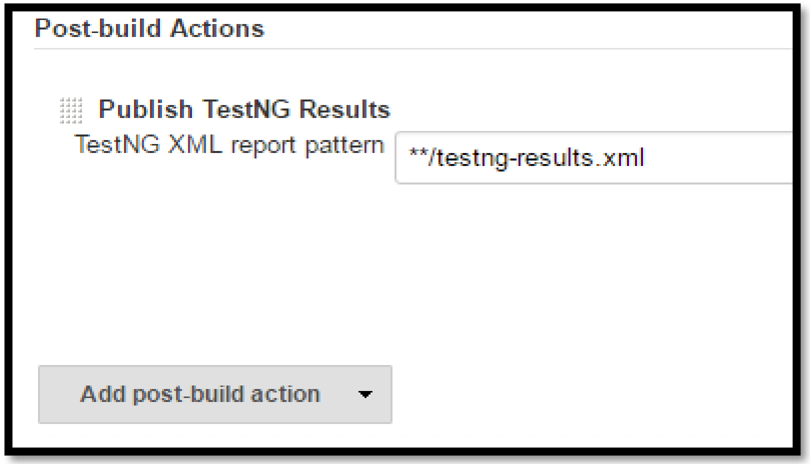
- Display result should be configured in "Post-build Actions" by following the below steps:
  - In build job configuration, select "Post-build Actions".
  - In "Publish Junit test result report" section, specify the path where the generated test report XML files should be placed.
    - For an Example, "test/data/*.xml" can be mentioned in the "Test Report XMLs" field.



Configuring TestNG

Once build generates test results, configure Jenkins build job to display them

- Navigate to Post-build Actions section
- Enable TestNG Results by choosing "Publish TestNG Results" from "Add post-build action" drop down list.
- Specify the XML report pattern as shown in order to save results during build execution

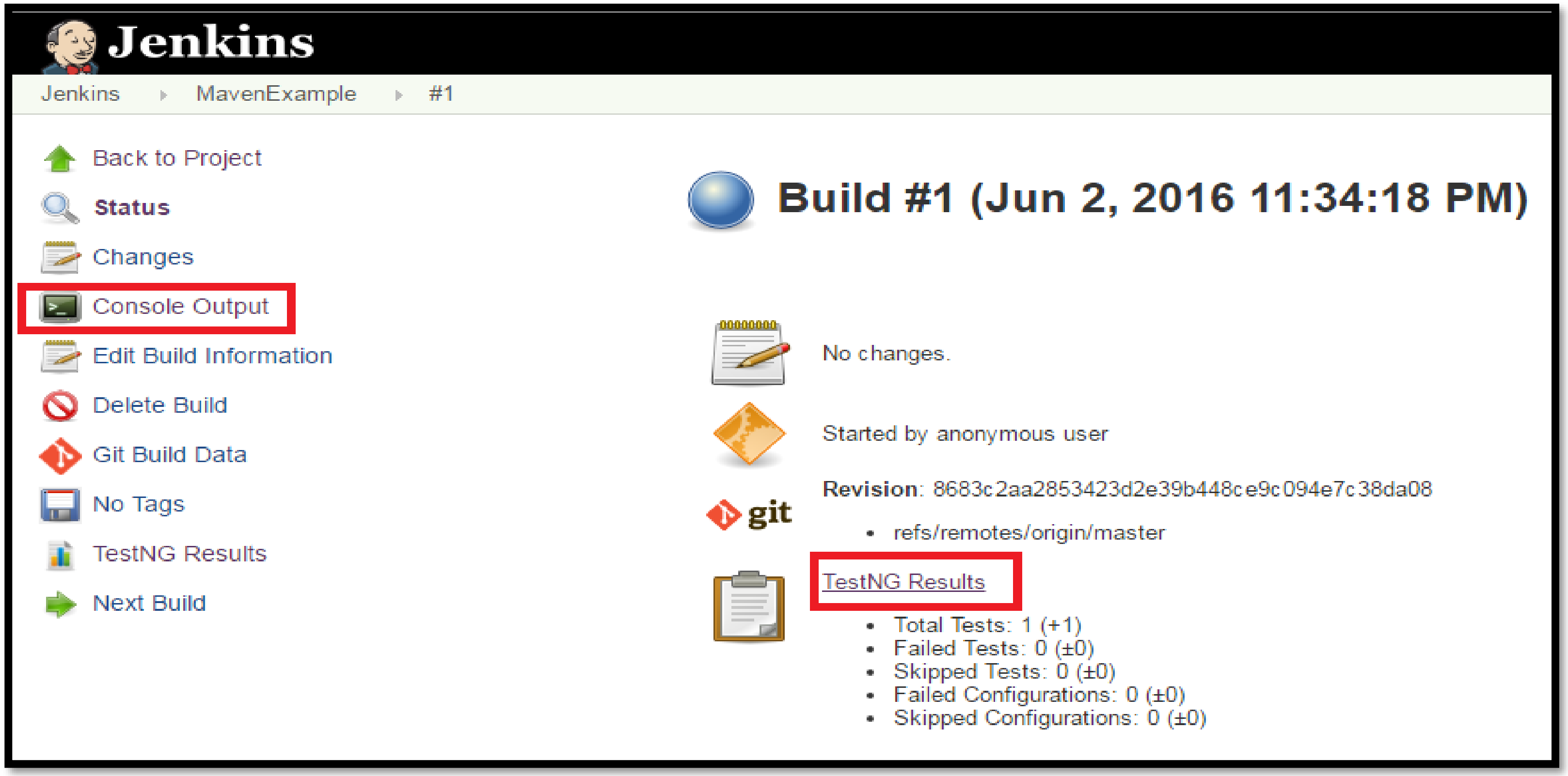

Post-build Actions

Publish TestNG Results

TestNG XML report pattern: **/testng-results.xml

Viewing Test Results

- View the generated test results by following the below steps:
    - Select the execute build number to the view the below output



Jenkins

Jenkins ▸ MavenExample ▸ #1

- Back to Project
- Status
- Changes
- Console Output
- Edit Build Information
- Delete Build
- Git Build Data
- No Tags
- TestNG Results
- Next Build

Build #1 (Jun 2, 2016 11:34:18 PM)

No changes.

Started by anonymous user

Revision: 8683c2aa2853423d2e39b448ce9c094e7c38da08

- refs/remotes/origin/master

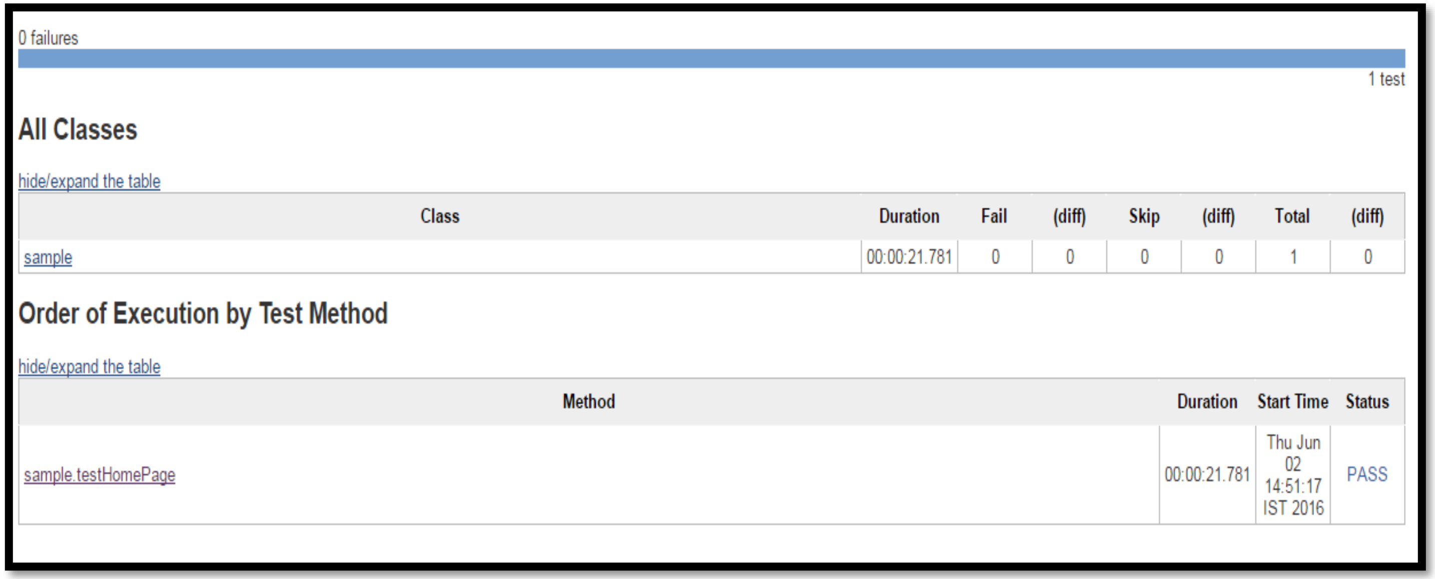
TestNG Results

- Total Tests: 1 (+1)
- Failed Tests: 0 (±0)
- Skipped Tests: 0 (±0)
- Failed Configurations: 0 (±0)
- Skipped Configurations: 0 (±0)

    - To view the console output, click on “Console Output” link.

Viewing TestNG Results

- Click on TestNG Results link to view the output as shown below



0 failures

1 test

All Classes

hide/expand the table

Class	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
sample	00:00:21.781	0	0	0	0	1	0

Order of Execution by Test Method

hide/expand the table

Method	Duration	Start Time	Status
sample.testHomePage	00:00:21.781	Thu Jun 02 14:51:17 IST 2016	PASS

Content Outline

Demo