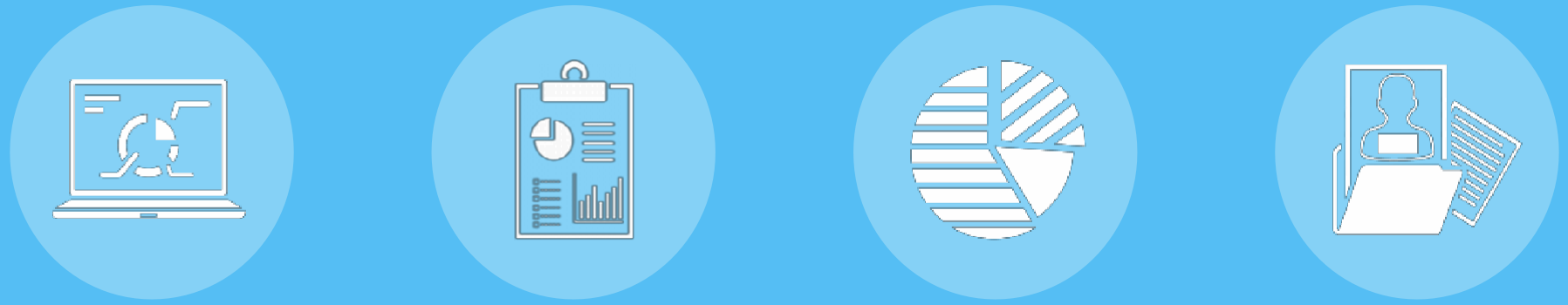
DevOps

Continuous Integration



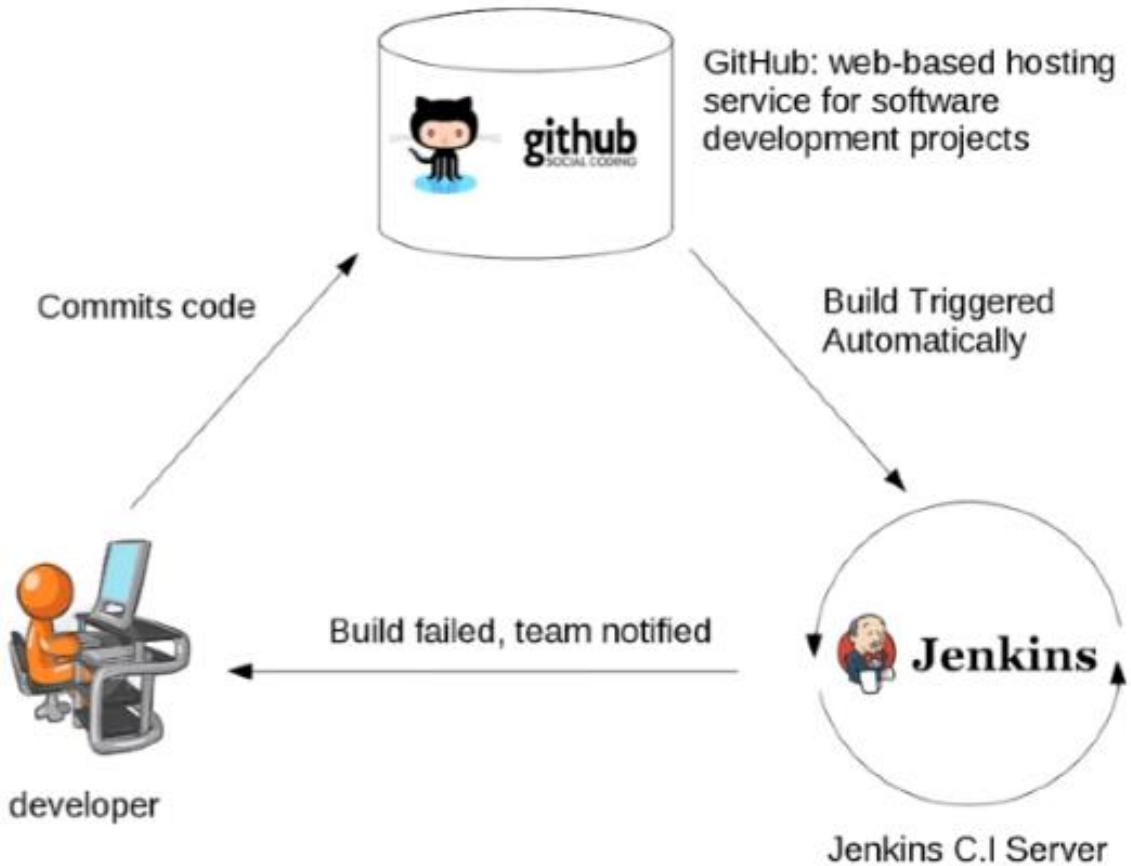
What's in It for Me

What is Continuous Integration

Understanding Jenkins as CI Tool

Why we need Continuous Integration

Learn how to configure, manage, and run CI Pipeline in Cloud



What is Continuous Integration?

- Developers commit code to a shared repository on a regular basis.
- Version control system is being monitored. When a commit is detected, a build will be triggered automatically.
- If the build is not green, developers will be notified immediately.

Continuous Integration

Process of automating the
build and testing of code

Shares code and unit tests by merging changes into a shared version control repository

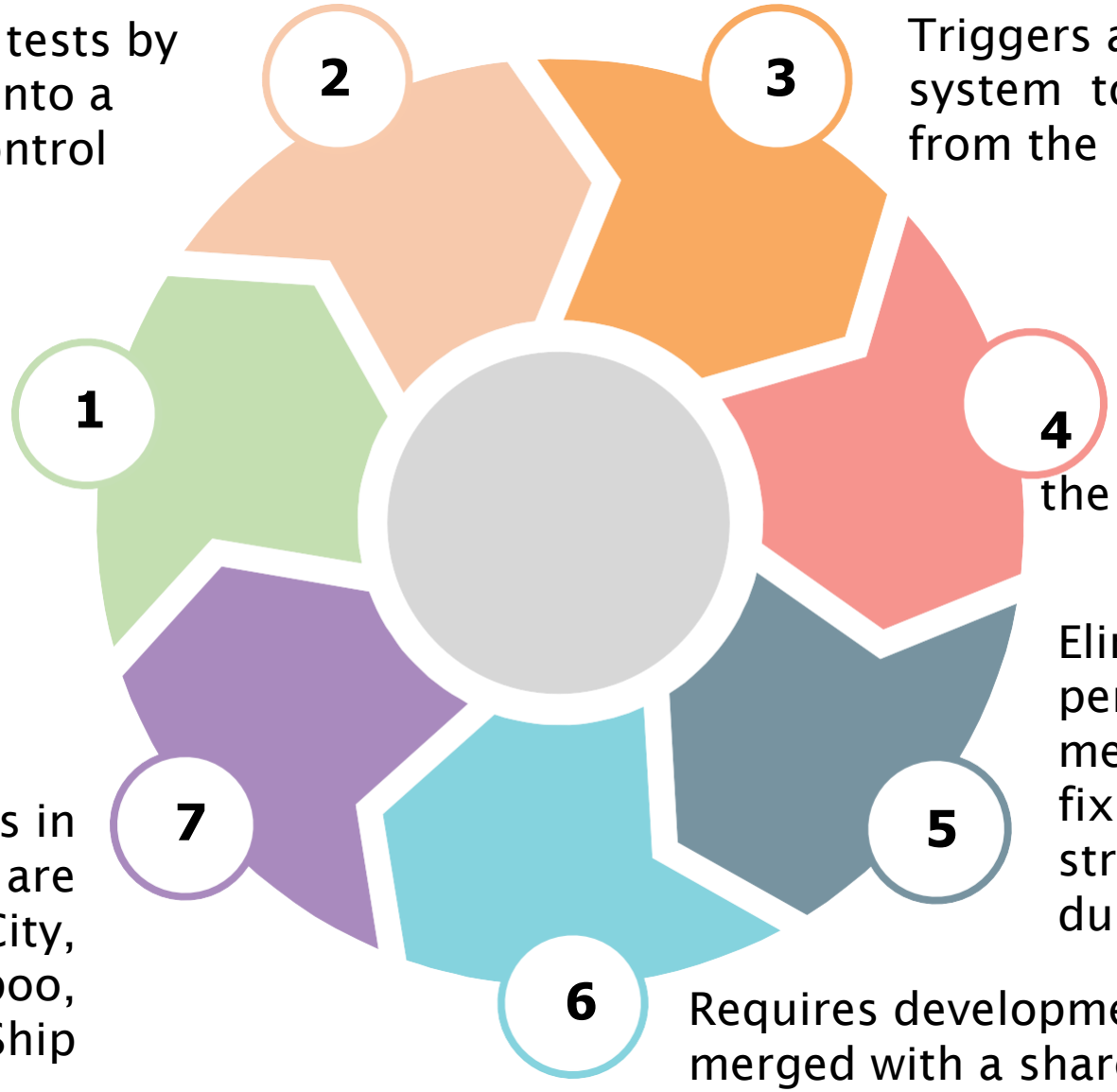
Triggers an automated build system to get the latest code from the shared repository

Integrates the changes made by software developers with the team's code base

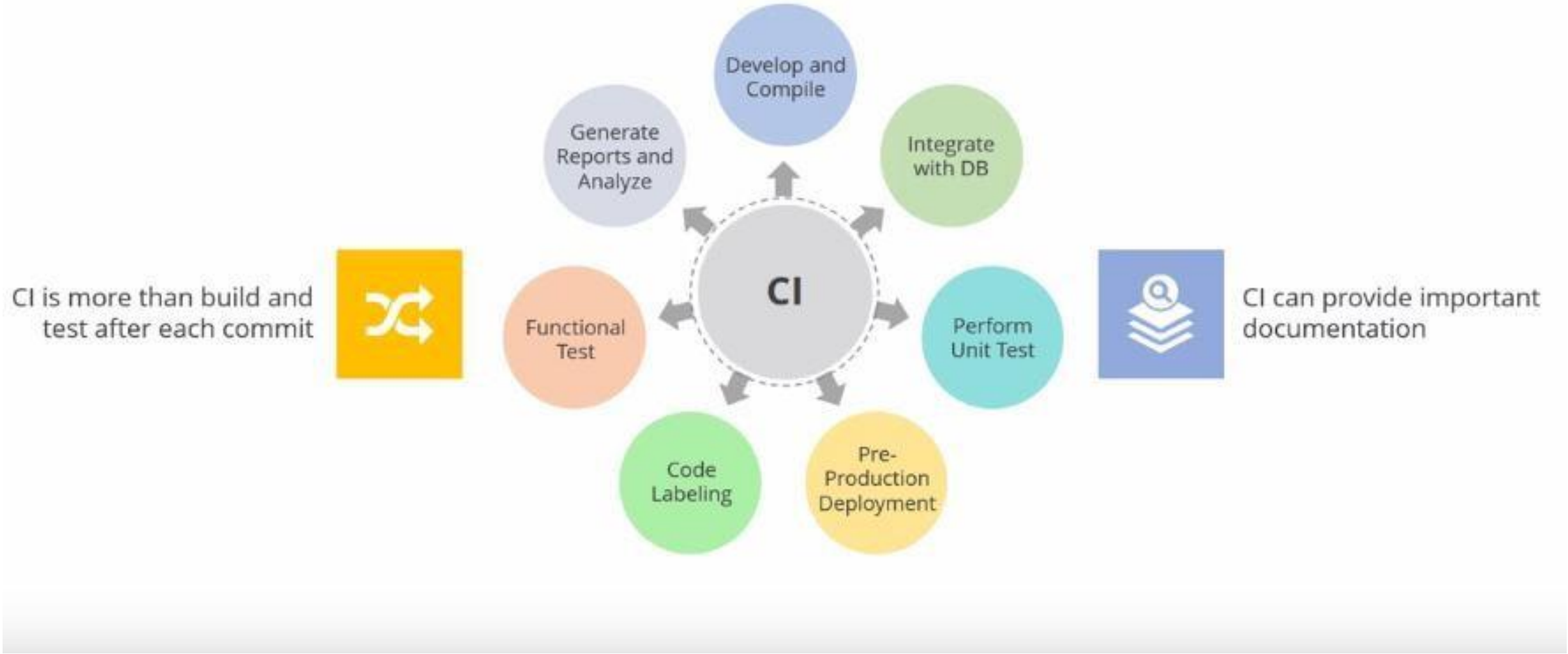
Eliminates long waiting periods to minimize merge conflicts, hard to fix bugs, diverging code strategies, and duplicated efforts

Requires development team's code to be merged with a shared version control branch continuously to avoid problems

Popular CI tools in the market are Jenkins, TeamCity, Travis CI, Bamboo, and CodeShip

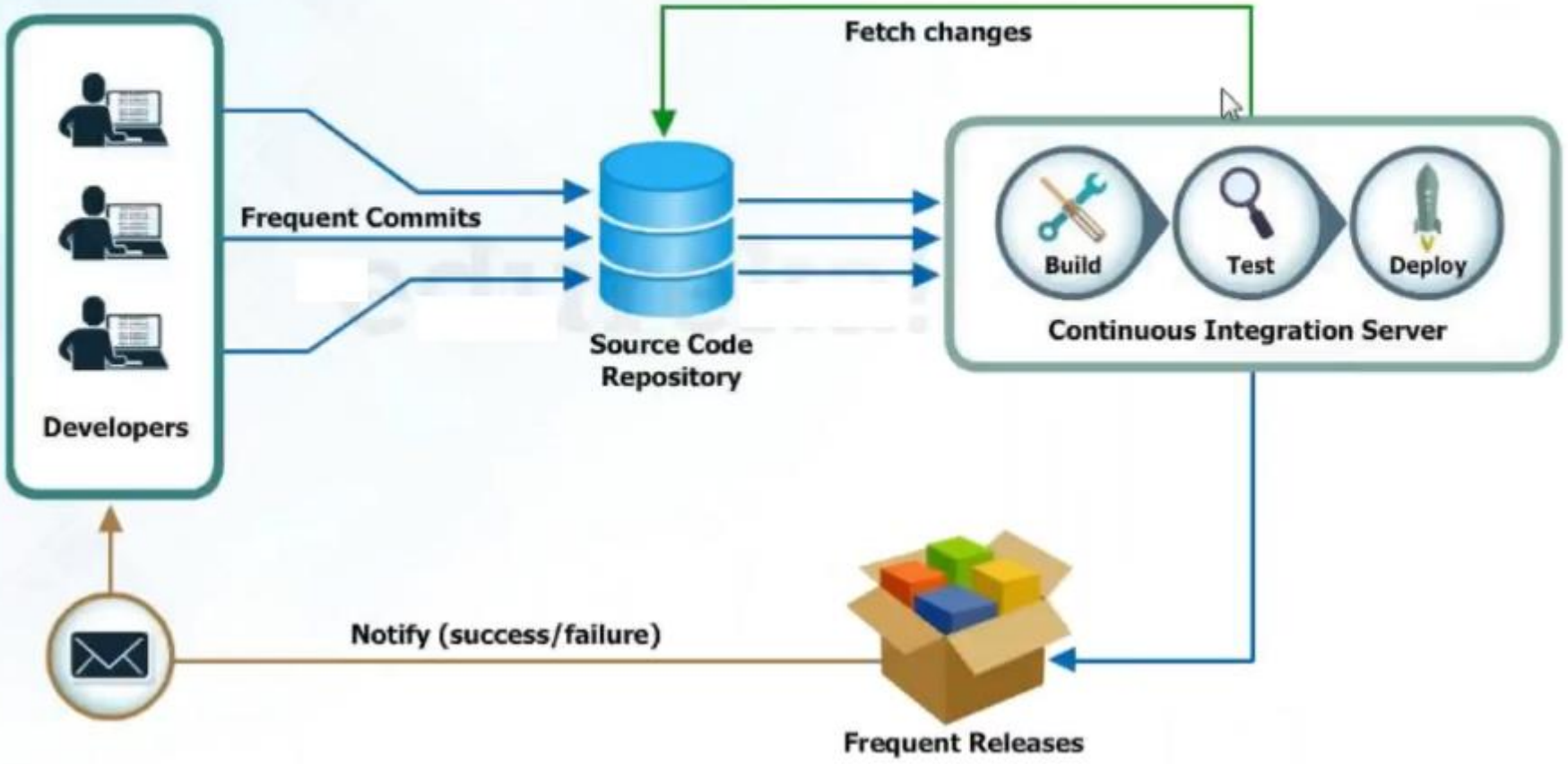


Continuous Integration (Contd.)



Continuous Integration (Contd.)

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to deploy code quickly to production.



Why do we need Continuous Integration?

- Detect problems or bugs, as early as possible, in the development life cycle.

- Since the entire code base is integrated, built and tested constantly , the potential bugs and errors are caught earlier in the life cycle which results in better quality software.

Jenkins

Using Jenkins as Continuous Integration Tool

Continuous Integration is the most important part of DevOps that is used to integrate various DevOps stages. Jenkins is the most famous Continuous Integration tool,

What is Jenkins?

- Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose.
- Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.
- It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.
- With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis and much more.
- Jenkins achieves Continuous Integration with the help of plugins. Plugins allows the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example: Git, Maven 2 project, Amazon EC2, HTML publisher etc.

The History of Jenkins

Hudson 

 Jenkins

- Hudson was started in 2004 at Sun by Kohsuke Kawaguchi as a hobby project.
- First release in 2005.
- Kohsuke worked on Hudson full time in early 2008.
- Became the leading Continuous Integration solution with a market share of over 70% in 2010.
- Renamed to Jenkins in 2011.

Jenkins

Java based

Jenkins is written in Java. It was forked from Hudson when Oracle bought Sun Microsystems

Open source automation server

It provides hundreds of plugins to support build creation, deployment, and automation of any software project

Software development

It helps automate non-human part of software development process with CI and continuous deployment (CD)

Cross-platform tool

It is a cross-platform tool, and it offers configuration both through GUI interface and console commands

Jenkins

CI server

It can be used as a CI server or as a continuous delivery hub for a project

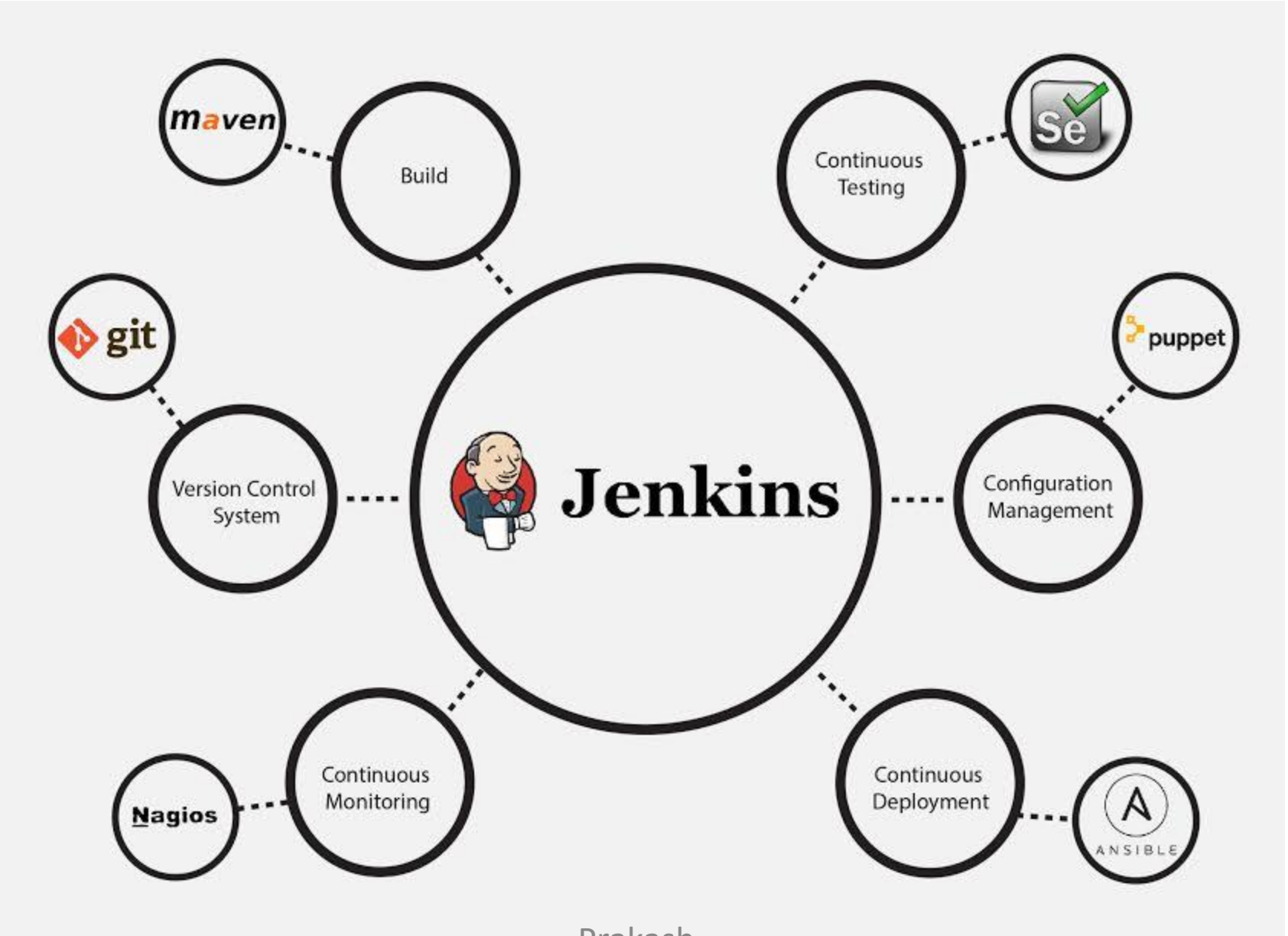
Distribution

It can easily distribute work across different machines and help trigger builds, tests, and deployments to multiple machines and platforms faster

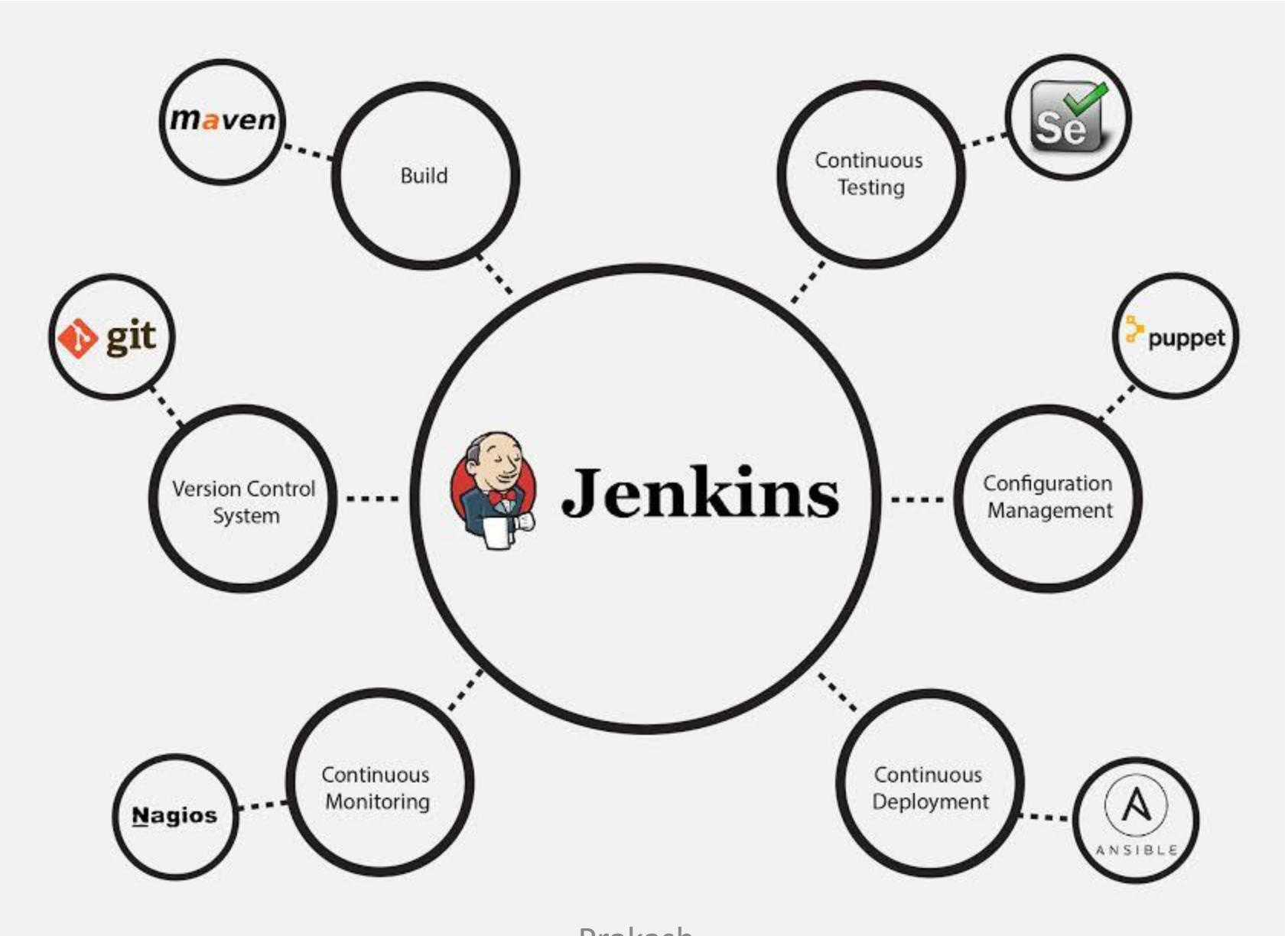
Cross-platform

It works on iOS, .Net, Android Development, Ruby, and Java

The image below depicts that Jenkins is integrating various DevOps stages:



The image below depicts that Jenkins is integrating various DevOps stages:



Advantages of Jenkins include:

- It is an open source tool with great community support.
- It is easy to install.
- It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share with the community.
- It is free of cost.
- It is built with Java and hence, it is portable to all the major platforms.

Jenkins Key Metrics:

Following are some facts about Jenkins that makes it better than other Continuous Integration tools:

- **Adoption:** Jenkins is widespread, with more than 147,000 active installations and over 1 million users around the world.

- **Plugins:** Jenkins is interconnected with well over 1,000 plugins that allow it to integrate with most of the development, testing and deployment tools.

Continuous Integration With Jenkins

Let us imagine a scenario where the complete source code of the application was built and then deployed on test server for testing. It sounds like a perfect way to develop a software, but, this process has many flaws. I will try to explain them one by one:

- Developers have to wait till the complete software is developed for the test results.

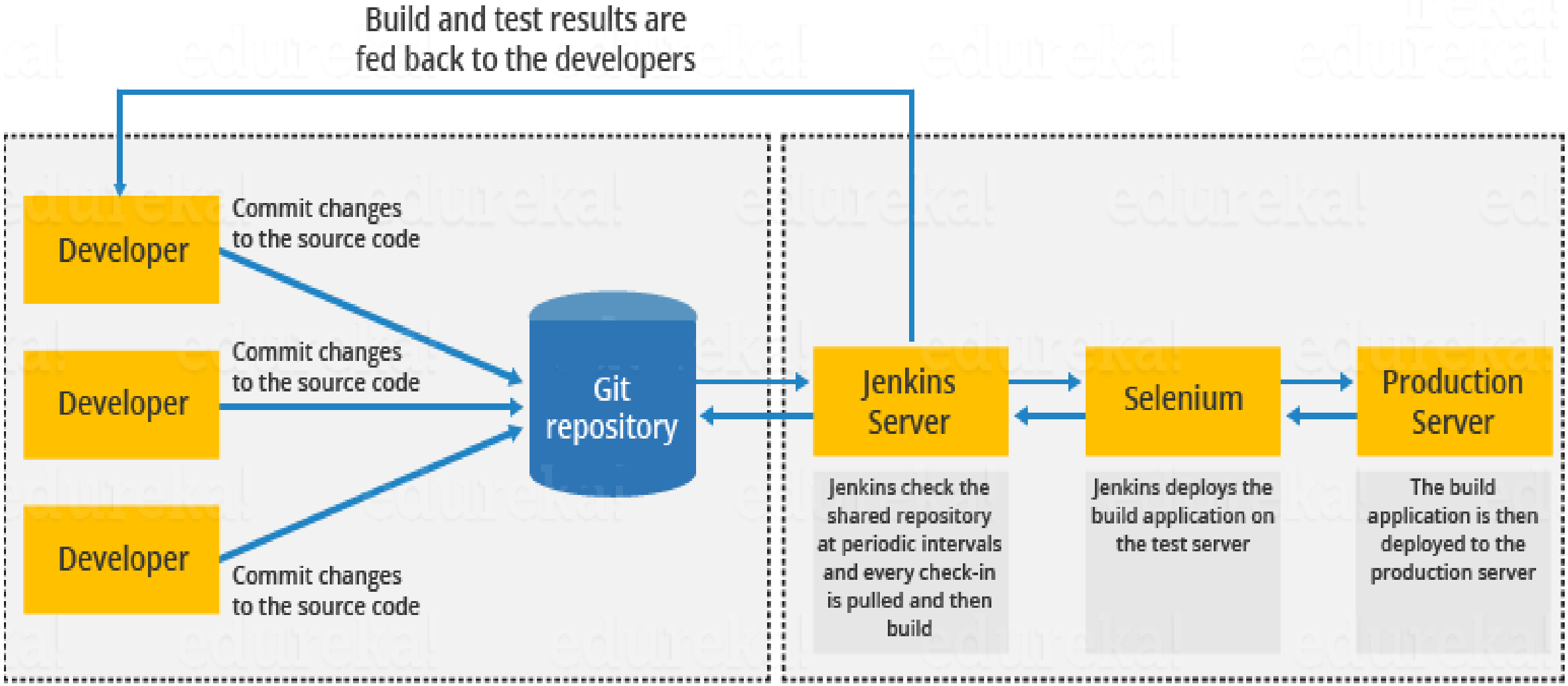
- There is a high possibility that the test results might show multiple bugs. It was tough for developers to locate those bugs because they have to check the entire source code of the application.

- It slows the software delivery process.

- Continuous feedback pertaining to things like coding or architectural issues, build failures, test status and file release uploads was missing due to which the quality of software can go down.

- The whole process was manual which increases the risk of frequent failure.

It is evident from the above stated problems that not only the software delivery process became slow but the quality of software also went down. This leads to customer dissatisfaction. So to overcome such a chaos there was a dire need for a system to exist where developers can continuously trigger a build and test for every change made in the source code. This is what CI is all about. Jenkins is the most mature CI tool available so let us see how Continuous Integration with Jenkins overcame the above shortcomings.


Build and test results are fed back to the developers
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Git repository
Jenkins Server
Jenkins check the shared repository at periodic intervals and every check-in is pulled and then build
Selenium
Jenkins deploys the build application on the test server
Production Server
The build application is then deployed to the production server

The above diagram is depicting the following functions:

- First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.
- Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.
- If the build fails, then the concerned team will be notified.
- If built is successful, then Jenkins deploys the built in the test server.
- After testing, Jenkins generates a feedback and then notifies the developers about the build and test results.
- It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.

Before Jenkins	After Jenkins
The entire source code was built and then tested. Locating and fixing bugs in the event of build and test failure was difficult and time consuming, which in turn slows the software delivery process.	Every commit made in the source code is built and tested. So, instead of checking the entire source code developers only need to focus on a particular commit. This leads to frequent new software releases.
Developers have to wait for test results	Developers know the test result of every commit made in the source code on the run.
The whole process is manual	You only need to commit changes to the source code and Jenkins will automate the rest of the process for you.

Continuous Integration Using Jenkins

- Builds can be triggered from a commit in version control system or scheduling a cron job or by other

  builds in the queue

- Support version control systems like CVS, Subversion, Git, Perforce, Clearcase
- Can be integrated with bug tracking databases Jira, Bugzilla, Sonar Quality Gate
- Integrates with testing tools like Nunit, Junit, TestLink, Celenium Capability Axis, qTest, QMetry

  for Jira, Sonar

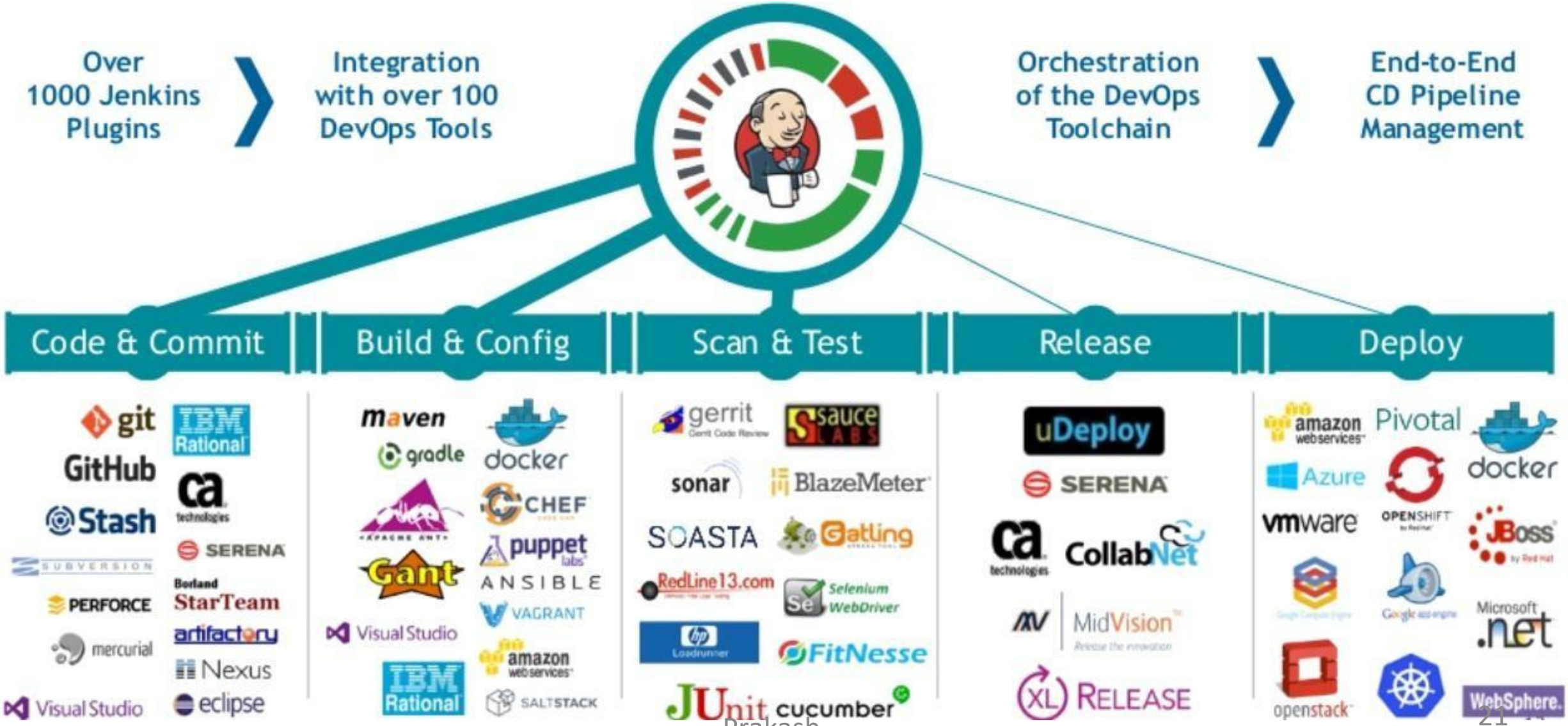
- Integrates with build tools like NAnt, EasyAnt, Ansible, Ant, Maven, Gradle, Visual Studio Code Metrics,

  SaltStack, Python, Ruby, Shell and Windows commands

- Integrates with config tools like Chef, Puppet, Ansible, Vagrant, IBM Rational, SaltStack
- Has plugins for Puppet Enterprise Pipeline, Ansible, OctopusDeploy, Docker Pipeline, Google

  Deployment Manager, Amazon Web Services, VMWare, Azure, Microsoft .Net, OpenStack

Continuous Integration Using Jenkins (Contd.)

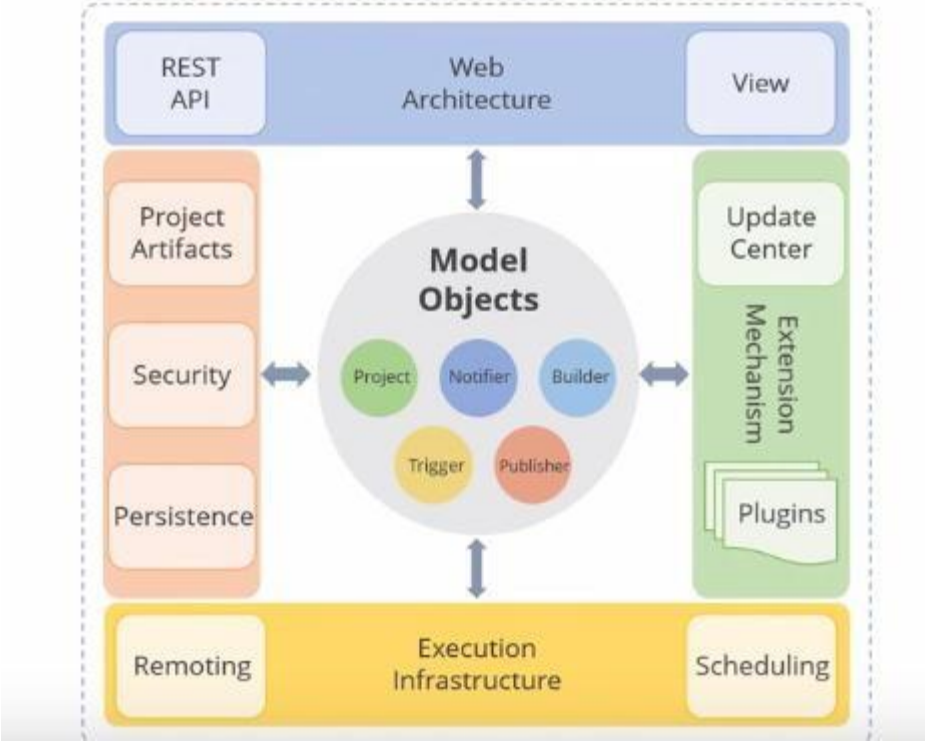


Jenkins Architecture

- Jenkins is a set of Java classes that model the concepts of a build system in a straight-forward fashion
- There are classes like Project, Build, that represents what the name says
- There are interfaces and classes that model code that performs a part of a build, such as SCM for accessing source code control system, Ant for performing an Ant- based build, Mailer for sending out e-mail notifications
- Jenkins classes are bound to URLs by using Stapler. Stapler is a HTTP request handling server.

Jenkins Architecture in Detail

- Views
  - Jenkins' model objects have multiple "views" that are used to render HTML pages about each object.
  - Jenkins uses Jelly as the view technology.
- Persistence
  - Jenkins uses the file system to store its data. Directories are created inside $JENKINS_HOME in a way that models the object model structure.
- Plugins
  - Jenkins' object model is extensible and it supports the notion of "plugins," which can plug into those extensibility points and extend the capabilities of Jenkins.



Step 1. Launch an instance (Amazon Linux) , Login to the instance, install and setup java environment

sudo yum install -y git java-1.8.0-openjdk-devel aws-cli

sudo alternatives --config java

In this command change version of java to the latest version.

Step 2. Install Apache Maven

sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

yum install maven

mvn –version

Step 4. Install Jenkins

wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo

rpm --import http://pkg.jenkins-ci.org/redhat-stable/jenkins-ci.org.key

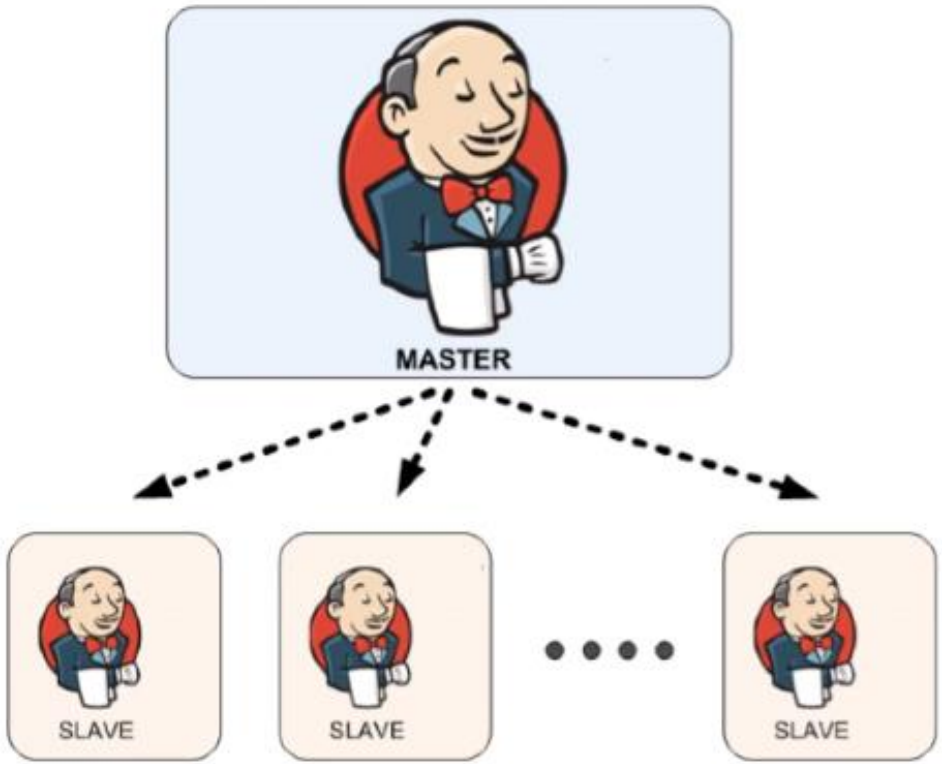
sudo yum install jenkins

sudo service jenkins start

sudo chkconfig --add jenkins

That’s it! Now you can go to URL http://<instance ip>:8080

Jenkins' Master and Slave Architecture



Master:

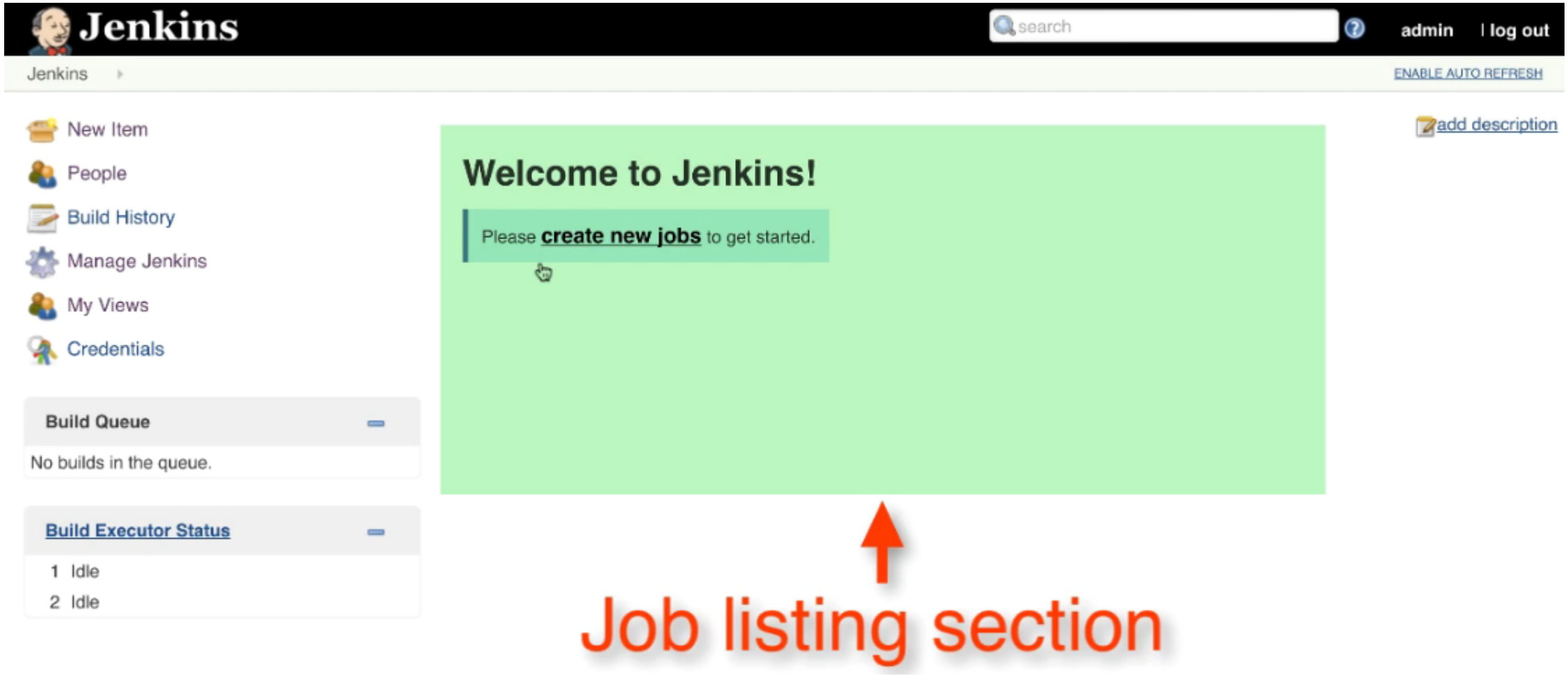
- Schedule build jobs.
- Dispatch builds to the slaves for the actual job execution.
- Monitor the slaves and record the build results.
- Can also execute build jobs directly.

Slave:

- Execute build jobs dispatched by the master.

Plugin

- A Plugin, like plugins on any other system, is a piece of software that extends the core functionality of the core Jenkins server.


Jenkins
search
admin
log out
Jenkins
ENABLE AUTO REFRESH
New Item
People
Build History
Manage Jenkins
My Views
Credentials
add description
Welcome to Jenkins!
Please create new jobs to get started.
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Job listing section

KNOWLEDGE CHECK	What are the advantages of Jenkins?

a. At integration stage, build failures are cached

b. It achieves continuous integration agile development and test driven development

c. For each code commit changes, an automatic build report notification is generated

d. All of the above

KNOWLEDGE CHECK	What are the advantages of Jenkins?

a. At integration stage, build failures are cached

b. It achieves continuous integration Agile development and test driven development

c. For each code commit changes, an automatic build report notification is generated

d. All of the above

The correct answer is **d. All of the above**

Advantages of Jenkins are that at integration stage, build failures are cached. It achieves continuous integration Agile development and test driven development. For each code commit changes, an automatic build report notification is generated.

Exercise 1

Exercise to set up CI pipeline using Jenkins

To configure and explore Docker networking, perform the following steps:

1. Create a HelloWorld Java program
2. Add it to Git repository
3. Install Jenkins
4. Create a project in Jenkins to trigger build for code changes to HelloWorld in GitHub
5. Run test cases of HelloWorld using Jmeter in Jenkins

Key Takeaways

- Continuous Integration automates the build and testing of code for every commit to version control.
- Tools used for Continuous Integration are Jenkins, TeamCity, Travis CI, Bamboo, and CodeShip.
- Jenkins is a cross-platform tool and offers configuration both through GUI interface and console commands.
- TeamCity is a Java-based build management and continuous integration server.

QUIZ 1	What is Jenkins?

a. A Build Server

b. A Test Server

c. An Open source automation server for builds, test, and deployment

d. A Mail Server

QUIZ 1	What is Jenkins?

a. A Build Server

b. A Test Server

c. An Open source automation server for builds, test, and deployment

d. A Mail Server

The correct answer is **c. Open source automation server for builds, test, and deployment**

Jenkins has plugins for a build server, test server, and an open source automation server.

QUIZ 2	What of the following is NOT a plugin of Jenkins?

a. Maven

b. Git

c. Amazon EC2

d. Source Code

QUIZ 2	Which of the following is NOT a plugin of Jenkins?

a. Maven

b. Git

c. Amazon EC2

d. Source Code

The correct answer is **d. Source Code**

Various source code repositories have a plugin for Jenkins; however, source code itself cannot be a plugin.

QUIZ 3	Which of the following is is NOT true for Continuous Integration with Jenkins?

a. Producing clean build

b. Automating unit test execution

c. Providing continuous feedback

d. Integrating with source code

QUIZ 3

Which of the following is NOT true for Continuous Integration with Jenkins?

a. Producing clean build

b. Automating unit test execution

c. Providing continuous feedback

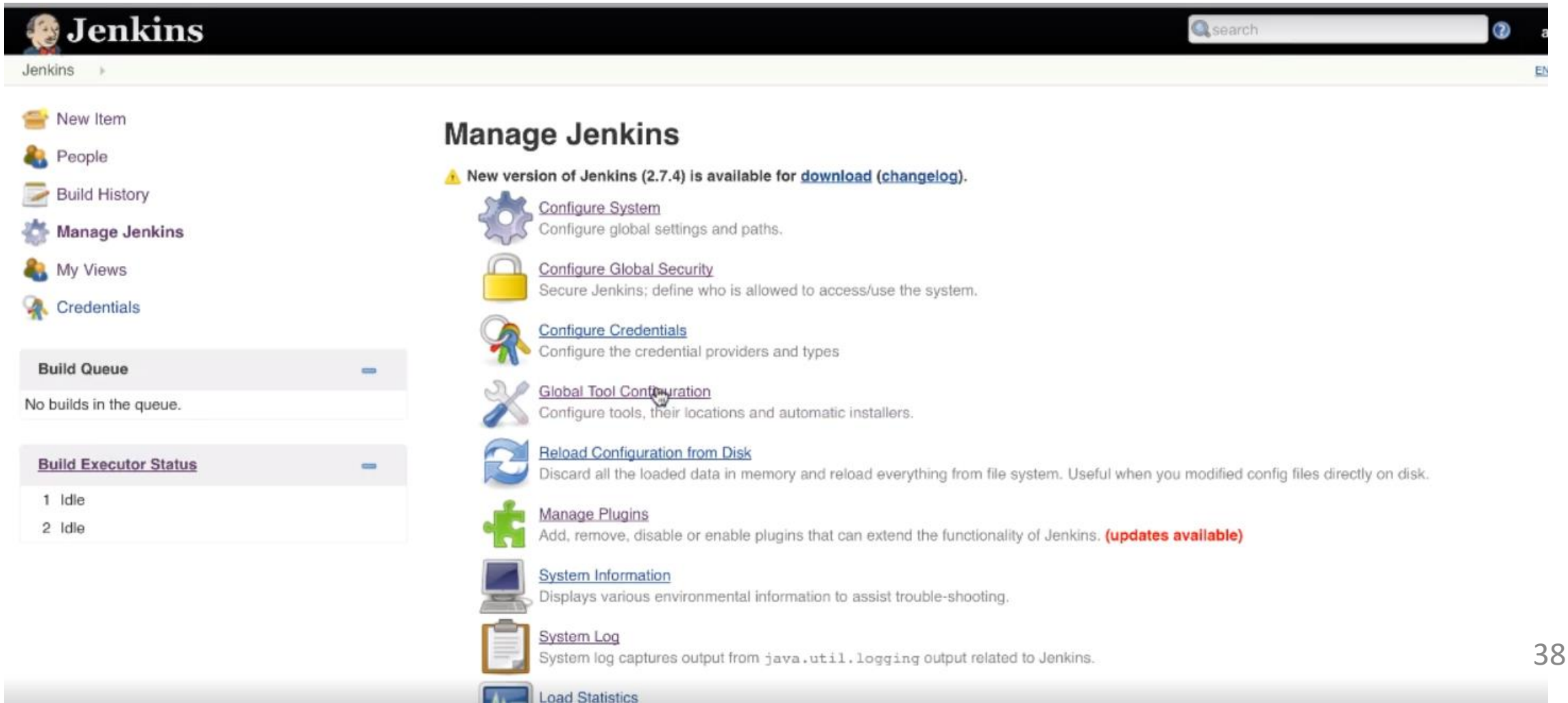
d. Integrating with source code

The correct answer is **c. Providing continuous feedback**

Jenkins does not have any tool/plugin for providing continuous feedback.

Configure Jenkins to work Maven, Java and GIT

1. Go to Manage Jenkins then select Global Tool Configuration
2. As you see we need to setup couple of things like JDK, GIT, Maven etc. So Jenkins to know where to locate those tools while build process.



JDK

JDK installations

JDK

Name: LocalJDK

JAVA_HOME: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.36.amzn1.x86_64

Install automatically

Delete JDK

Add JDK

List of JDK installations on this system

Git

Git installations

Git

Name: LocalGit

Path to Git executable: git

Maven

Maven installations

Maven

Name: LocalMaven

MAVEN_HOME: /usr/share/apache-maven

Install automatically

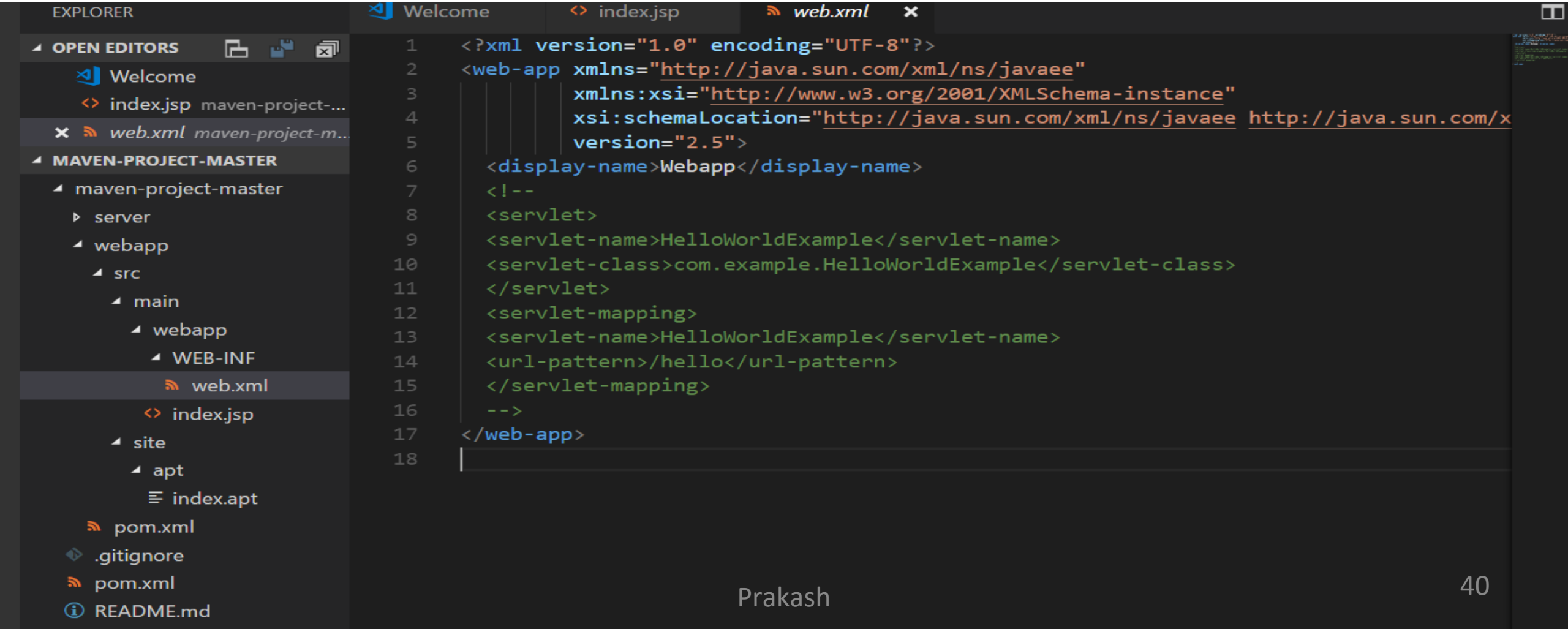
Delete Maven

First Maven Based Project:

Github repo UTR https://github.com/prakashk0301/maven-project.git

Web.xml: it will tell you how to deploy your application

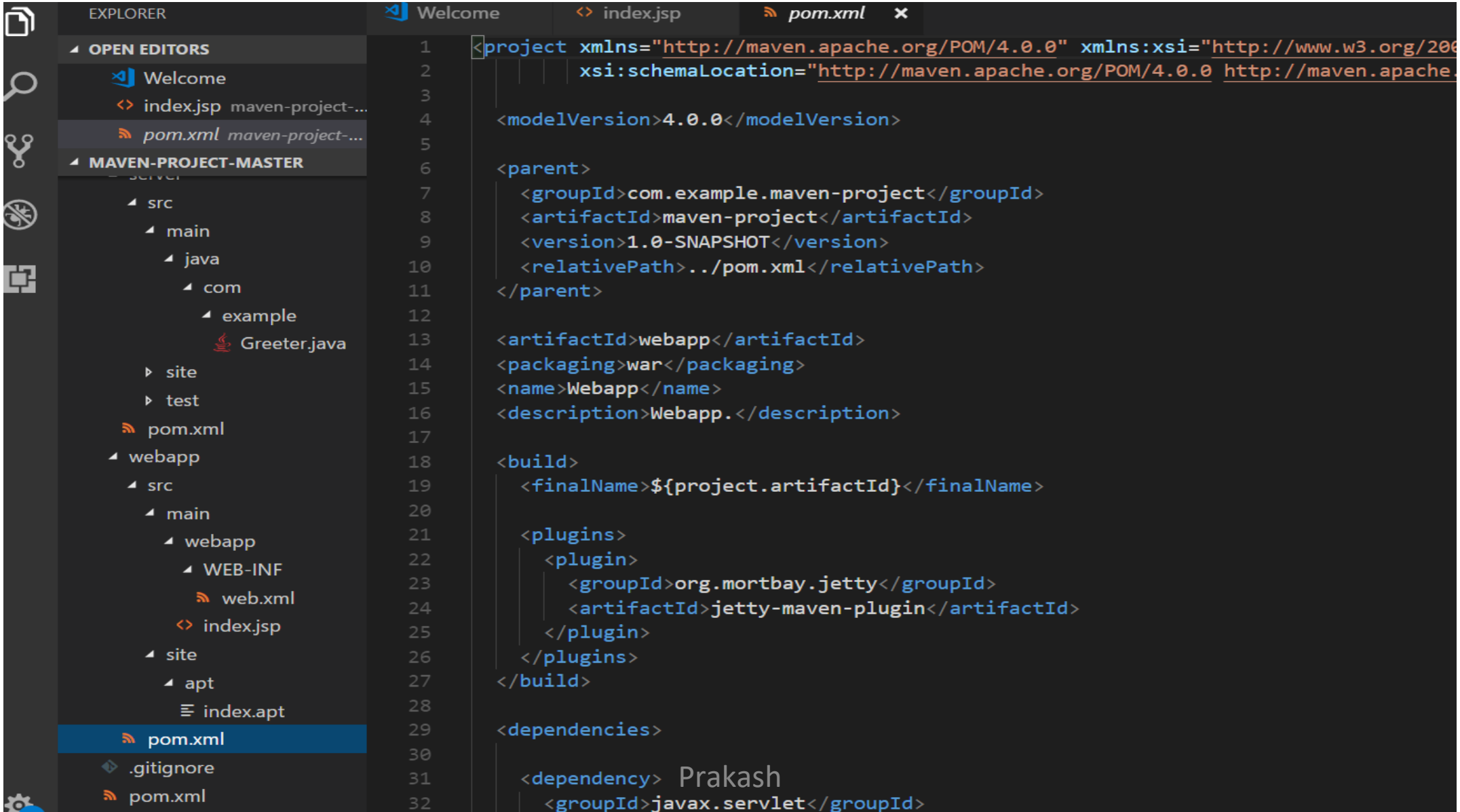
Index.jsp: Html page information ,which we can see after successful deployment.

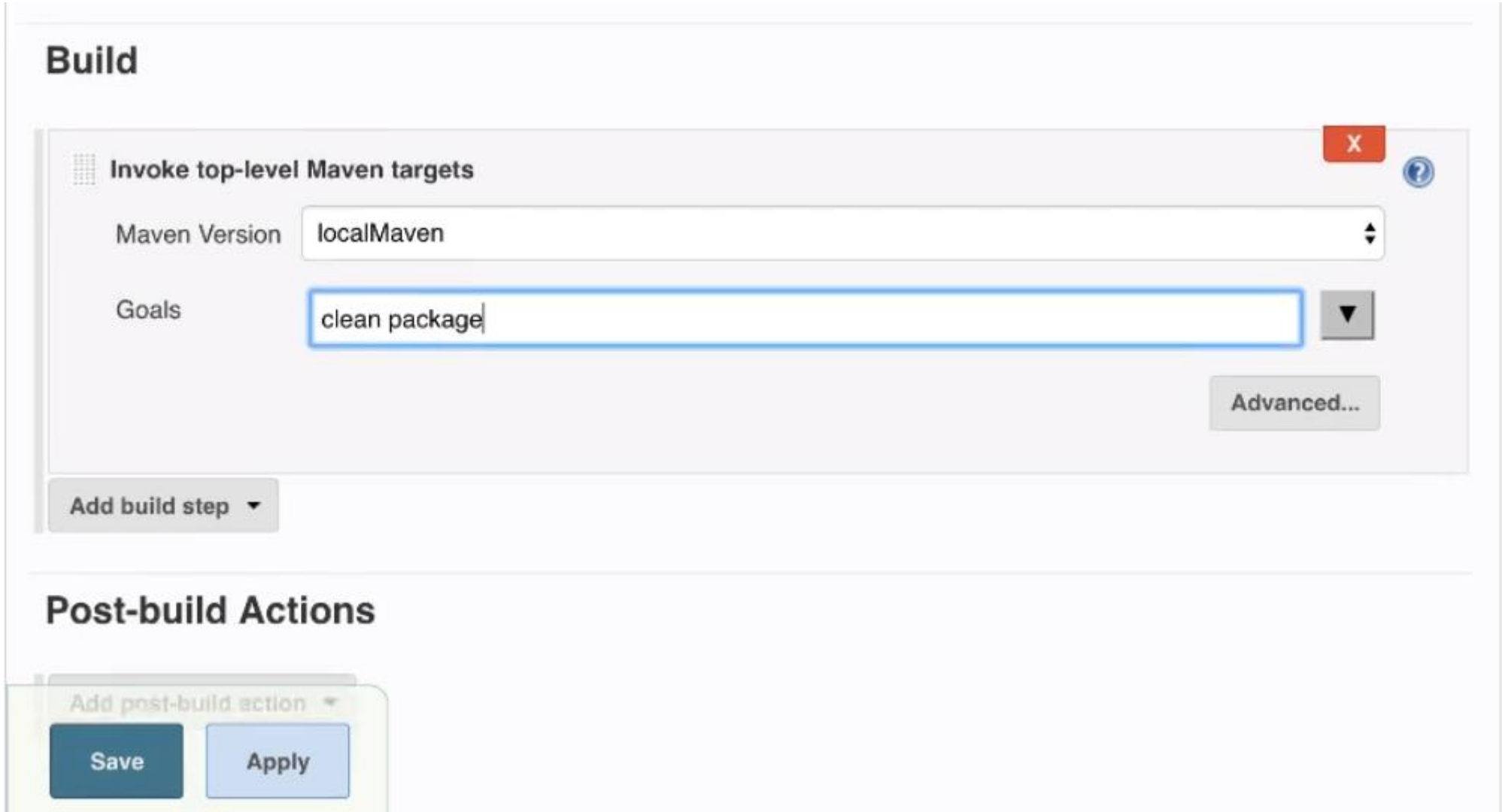


Maven pom.xml file

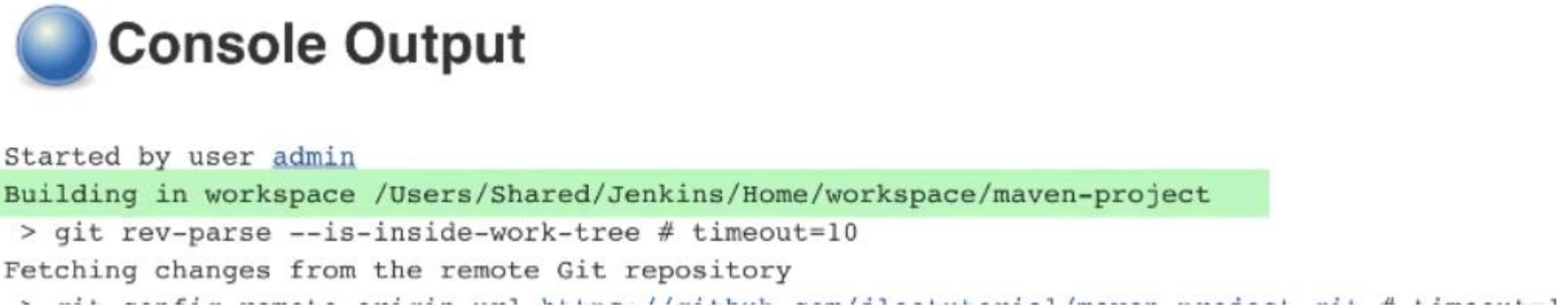
- Describe the software project being built, including
  - The dependencies on other external modules.
  - The directory structures.
  - The required plugins.
  - The predefined targets for performing certain tasks such as compilation and packaging.

In our project we have total 3 pom.xml file




Build
Invoke top-level Maven targets
Maven Version
localMaven
Goals
clean package
Advanced...
Add build step
Post-build Actions
Add post-build action
Save
Apply

Jenkins has build our project in a workspace under a shared directory on the local box



Jenkins figured out the right order to build the job, First one is the root directory of our project

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
[INFO] Maven Project
[INFO] Server
[INFO] Webapp
[INFO]
[INFO] ------------------------------------------------------------------------
```

Cleaning the environment before building the job, Basically it is removing any artifact before building

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.jar (25 kB at
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ maven-project ---
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-api/2.0.6/maven-plugin-api-2.0.6.pom
Progress (1): 1.5 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-api/2.0.6/maven-plugin-api-2.0.6.pom (1.5 kB at 182 kB/
```

Server module: Jenkins is executing the sequence of build command such as compile, test for packaging

```
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ webapp ---
[INFO]
[INFO] --- maven-resources-plugin:2.5:resources (default-resources) @ webapp ---
[debug] execute contextualize
[INFO] Using 'utf-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/maven-project/webapp/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ webapp ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-resources-plugin:2.5:testResources (default-testResources) @ webapp ---
[debug] execute contextualize
[INFO] Using 'utf-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/maven-project/webapp/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:testCompile (default-testCompile) @ webapp ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.11:test (default-test) @ webapp ---
[INFO] No tests to run.
[INFO] Surefire report directory: /var/lib/jenkins/workspace/maven-project/webapp/target/surefire-reports
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.11/surefire-jun
Progress (1): 1.7 kB
```

After server module it prints out the Unit test report

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------

Results :

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-war-plugin:2.2:war (default-war) @ webapp ---
```

Lastly it package the Server module and producing the jar file under the server directory

```
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ server ---
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/2.5/maven-archiver-2.5.pom
Progress (1): 2.1/4.5 kB
Progress (1): 4.5 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/2.5/maven-archiver-2.5.pom (4.5 kB
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-components/17/maven-shared-cc
```

```
Downloaded from central: https://repo.maven.apache.org/maven2/commons-lang/commons-lang/2.1/commons-lang-2.1.jar (208 kB at 3.6 MB/s)
[INFO] Building jar: /var/lib/jenkins/workspace/maven-project/server/target/server.jar
[INFO]
[INFO] ------------------------------------------------------------------------
```

Building the webapp module which is pretty similar to the server module

```
[INFO] ------------------------------------------------------------------------
[INFO] Building Webapp 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-war-plugin/2.2/maven-war-plugin-2.2.pom
Progress (1): 4.1/6.5 kB
Progress (1): 6.5 kB
```

```
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ webapp ---
[INFO]
[INFO] --- maven-resources-plugin:2.5:resources (default-resources) @ webapp ---
[debug] execute contextualize
[INFO] Using 'utf-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/maven-project/webapp/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ webapp ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-resources-plugin:2.5:testResources (default-testResources) @ webapp ---
[debug] execute contextualize
[INFO] Using 'utf-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/maven-project/webapp/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:testCompile (default-testCompile) @ webapp ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.11:test (default-test) @ webapp ---
[INFO] No tests to run.
[INFO] Surefire report directory: /var/lib/jenkins/workspace/maven-project/webapp/target/surefire-reports
```

In the end it packages the webapp module and produces the webapp war file which can be further deploy to servlet container like tomcat, apache

```
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/xstream/xstream/1.3.1/xstream-1.3.1.jar (431 kB at 7.3 MB/s)
[INFO] Packaging webapp
[INFO] Assembling webapp [webapp] in [/var/lib/jenkins/workspace/maven-project/webapp/target/webapp]
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/maven-project/webapp/src/main/webapp]
[INFO] Webapp assembled in [31 msecs]
[INFO] Building war: /var/lib/jenkins/workspace/maven-project/webapp/target/webapp.war
[INFO] WEB-INF/web.xml already added, skipping
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Maven Project ..................................... SUCCESS [  1.775 s]
[INFO] Server ............................................ SUCCESS [  5.571 s]
[INFO] Webapp ............................................ SUCCESS [  0.941 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.578 s
[INFO] Finished at: 2018-02-16T23:46:18Z
[INFO] Final Memory: 21M/51M
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

In workspace: It displays the directory structure of this workspace which Jenkins uses to built the project