

Continuous Integration

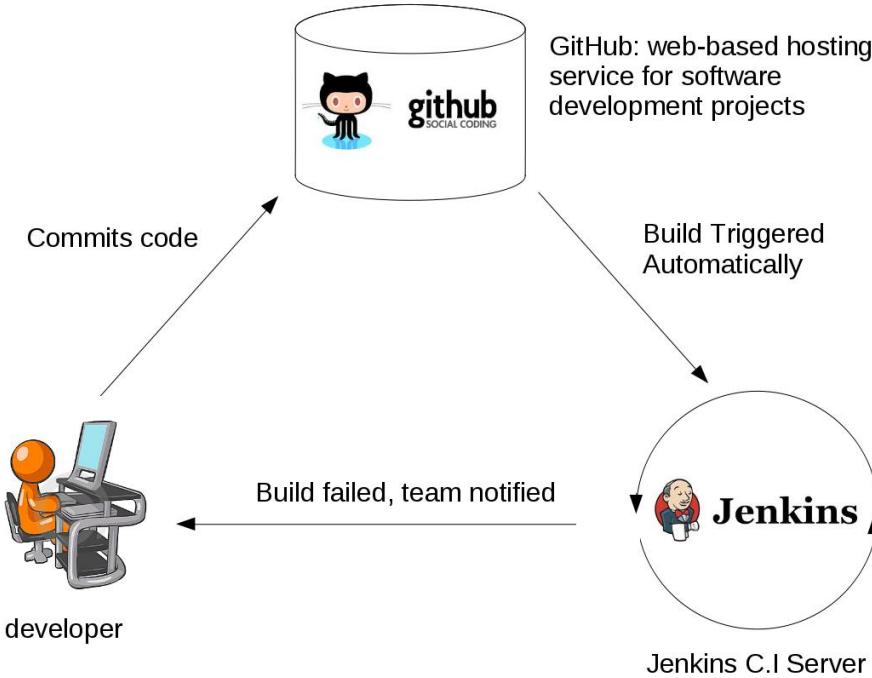


Jenkins

Continuous Integration

- What is Continuous Integration?
- Why do we need it?
- Different phases of adopting Continuous Integration





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.

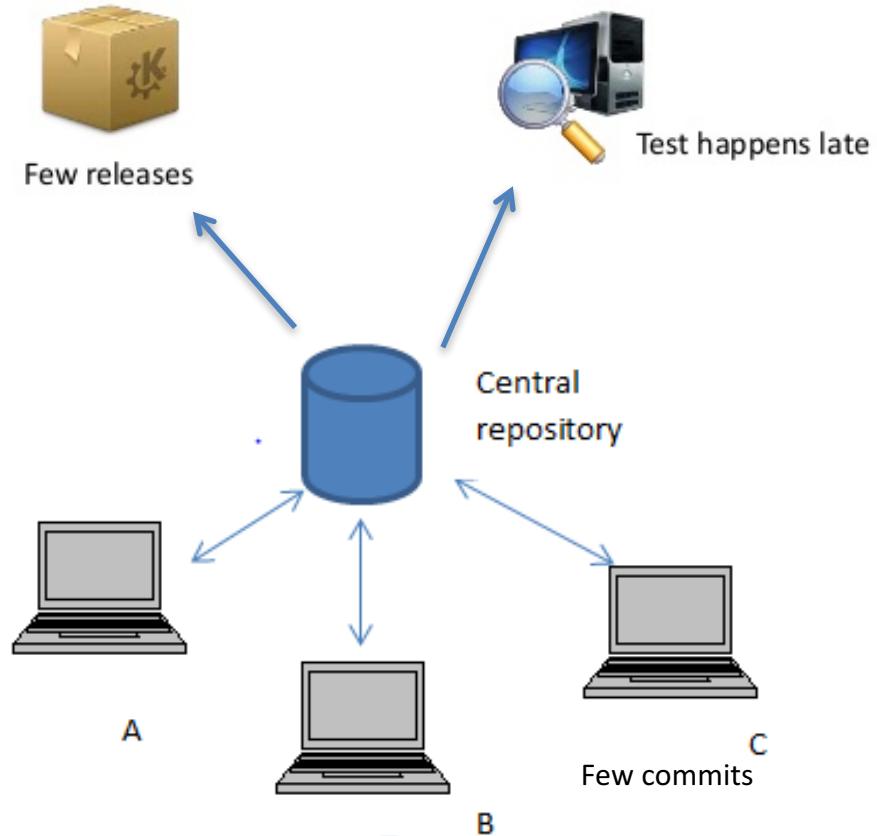
Why do we need Continuous Integration?

- 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.

Different stages of adopting Continuous Integration

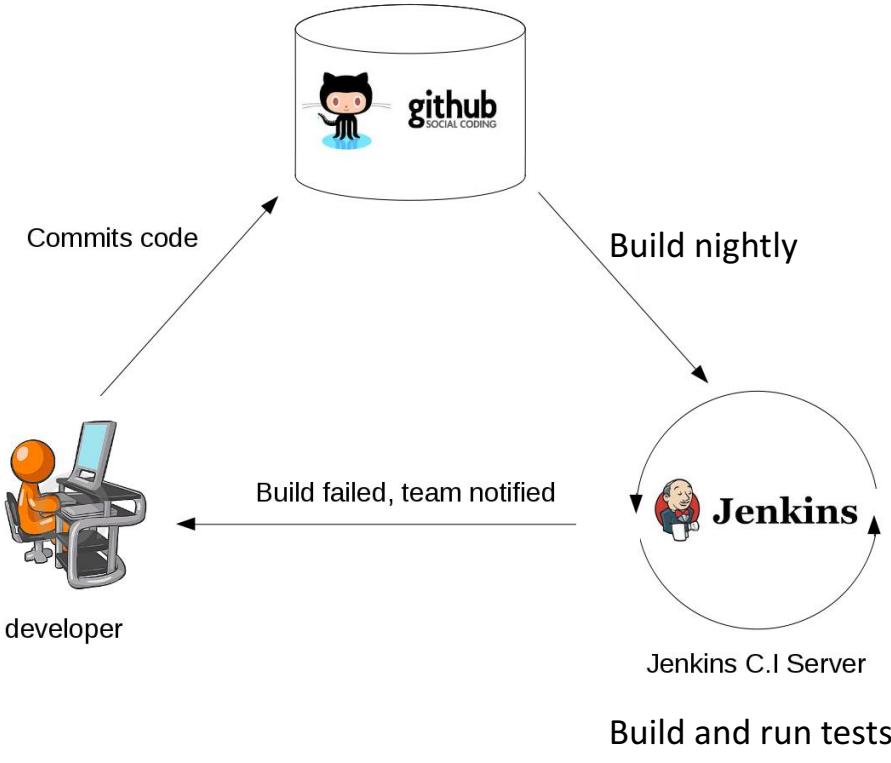


Jenkins



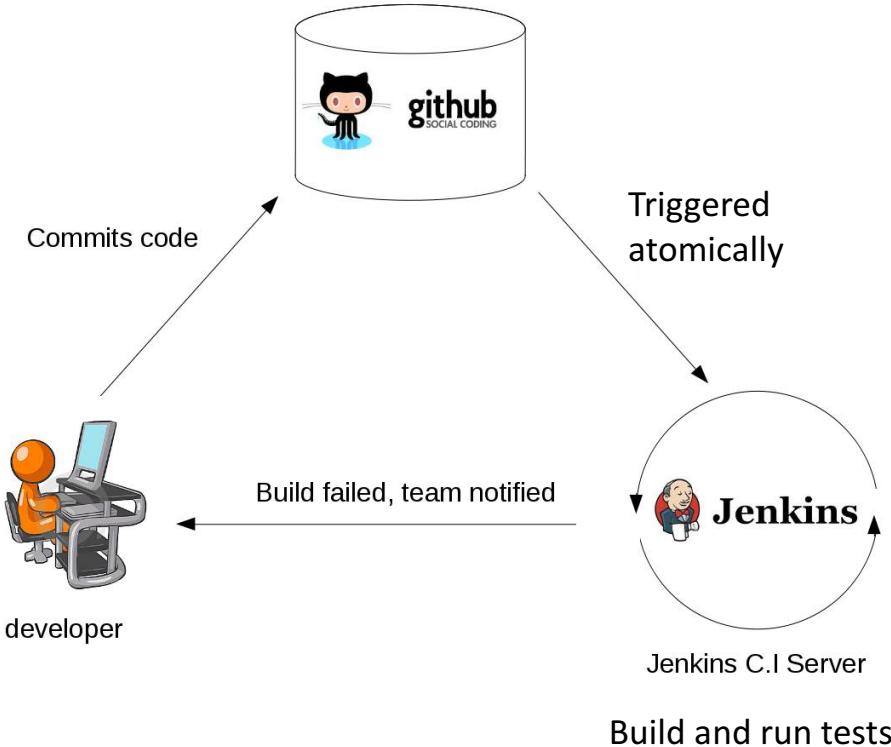
Stage 1:

- No build servers.
- Developers commit on a regular basis.
- Changes are integrated and tested manually.
- Fewer releases.



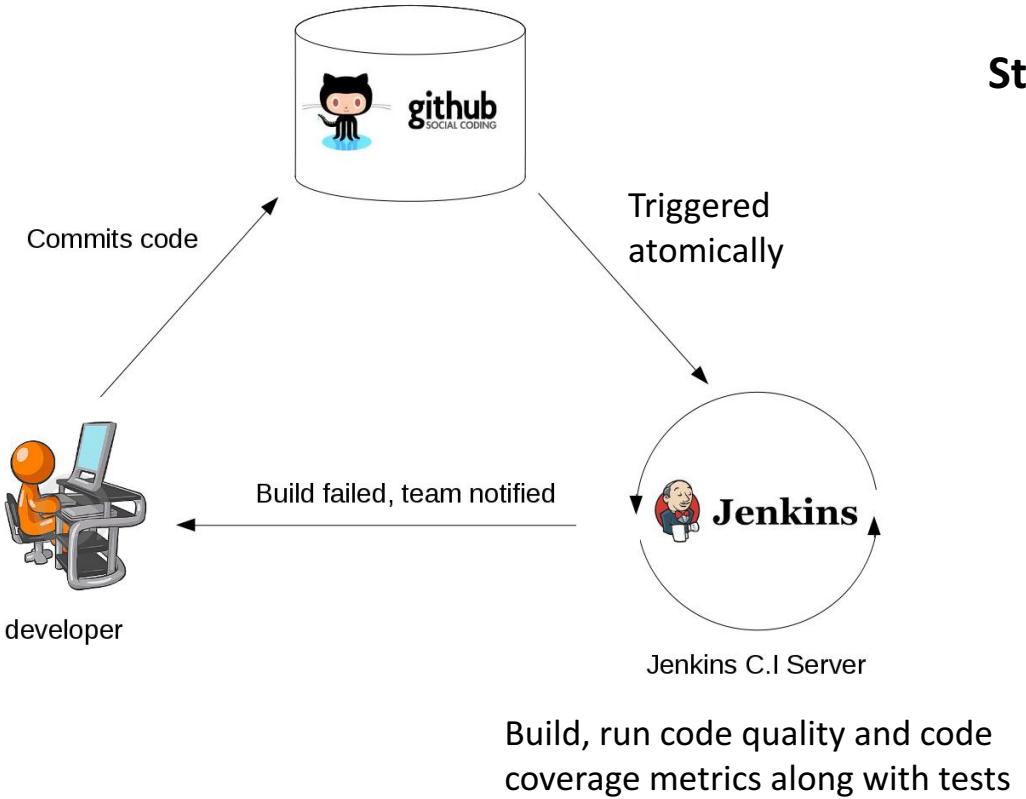
Stage 2:

- Automated builds are scheduled on a regular basis.
- Build script compiles the application and runs a set of automated tests.
- Developers now commit their changes regularly.
- Build servers would alert the team members in case of build failure.



Stage 3:

- A build is triggered whenever new code is committed to the central repository.
- Broken builds are usually treated as a high priority issue and are fixed quickly.

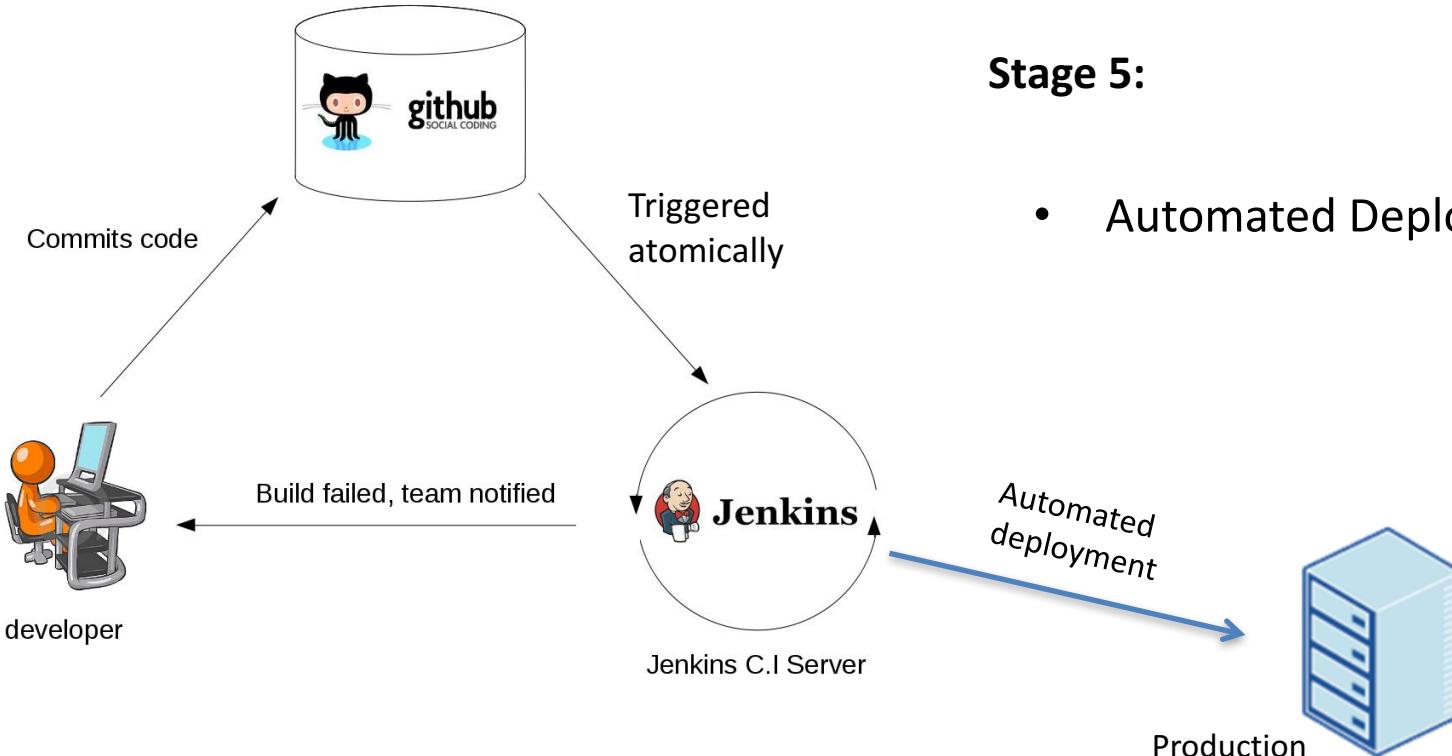


Stage 4:

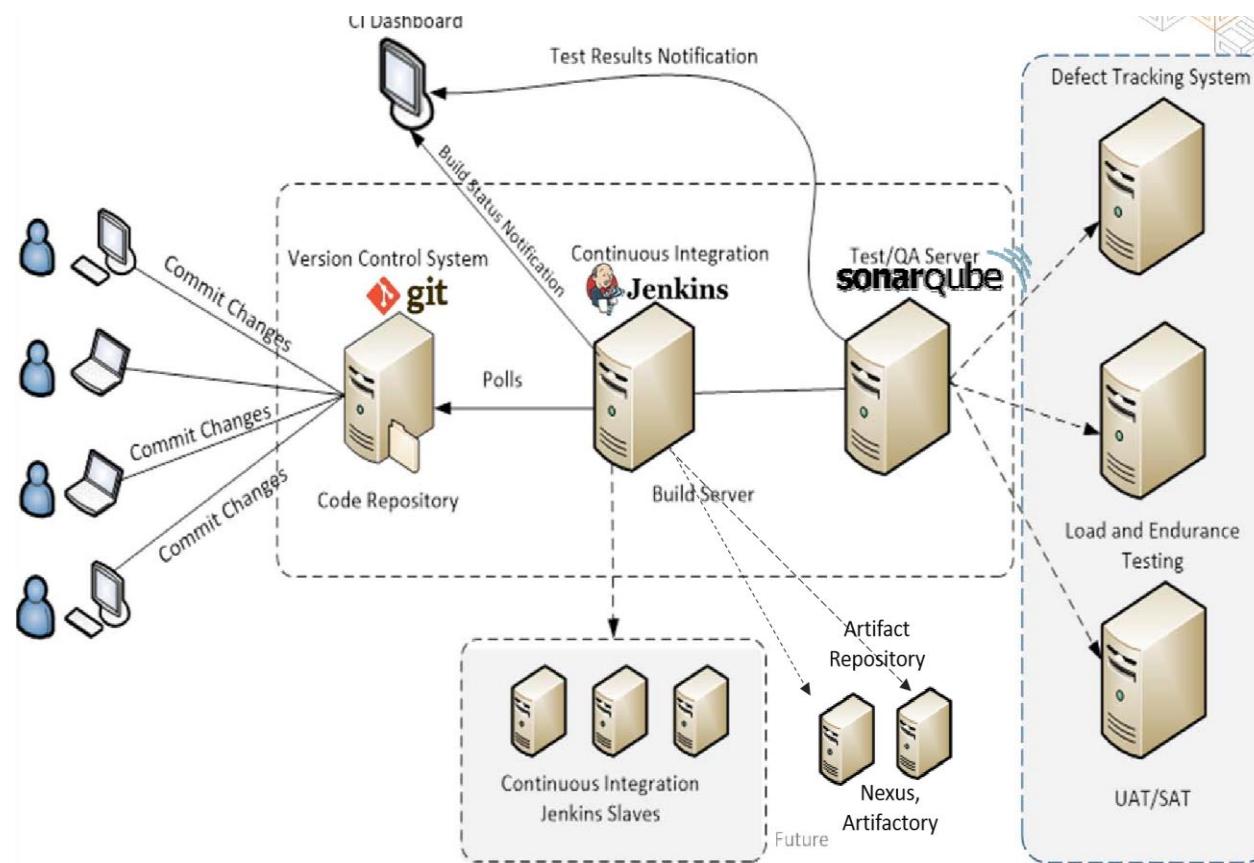
- Automated code quality and code coverage metrics are now run along with unit tests to continuously evaluate the code quality.

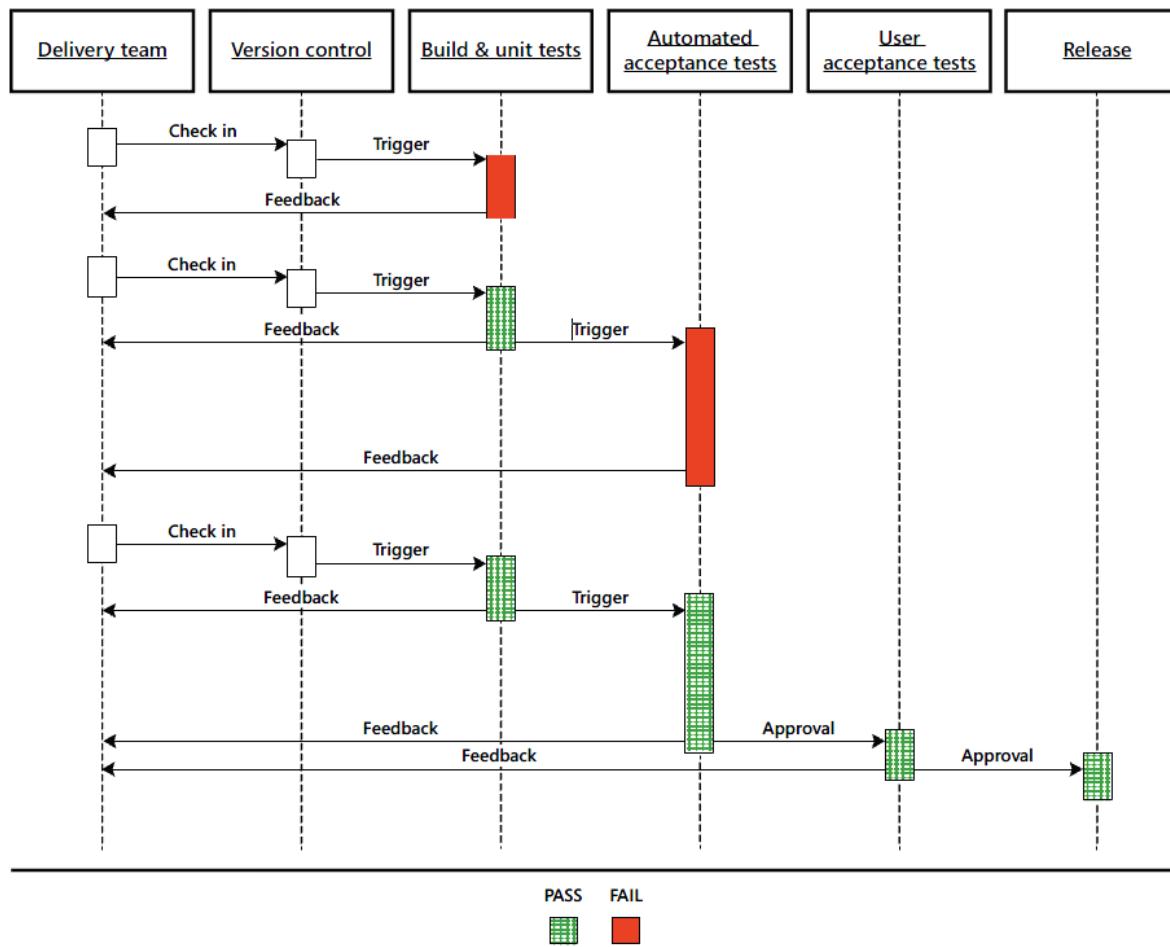
Is the code coverage increasing?

Do we have fewer and fewer build failures?



CI/CD Environment





Continuous Integration

Continuous Delivery

Continuous Deployment

- **Continuous Integration**

The practice of merging development work with the main branch constantly.

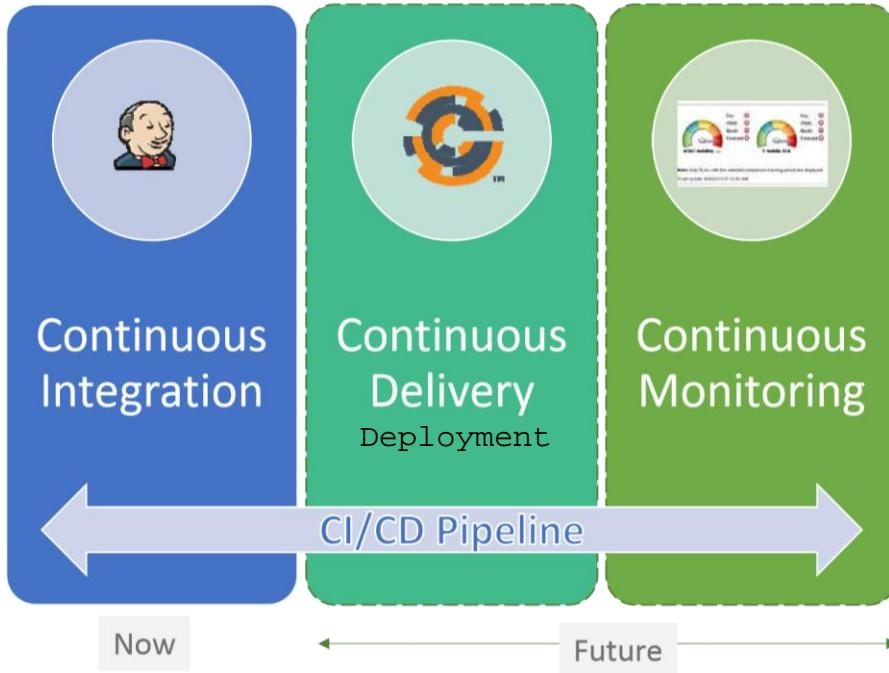
- **Continuous Delivery**

Continual delivery of code to an environment once the code is ready to ship. This could be staging or production. The idea is the product is delivered to a user base, which can be QAs or customers for review and inspection.

- **Continuous Deployment**

The deployment or release of code to production as soon as it is ready.

DevOps



How to implement Continuous Integration?



Non-hosted solutions



Hosted solutions

Continuous Integration is also a mindset

- Fixing broken builds should be treated as a high priority issue for all team members.
- The deployment process should be automated, with no manual steps involved.
- All team members should focus on contributing to high-quality tests because the confidentiality of the CI process highly depends on the quality of the tests.

What is Jenkins

- Jenkins is a continuous integration and build server.
- It is used to manually, periodically, or automatically build software development projects.
- It is an open source Continuous Integration tool written in Java.
- Jenkins is used by teams of all different sizes, for projects with various languages.

Why Jenkins is popular

- Easy to use
- Great extensibility
 - Support different version control systems
 - Code quality metrics
 - Build notifiers
 - UI customization



U.S. Department of Commerce
Economics and Statistics Administration
U.S. CENSUS BUREAU
census.gov

Plugins by topic

Source code management

Jenkins has native support for Subversion and CVS as well as the following plugins:

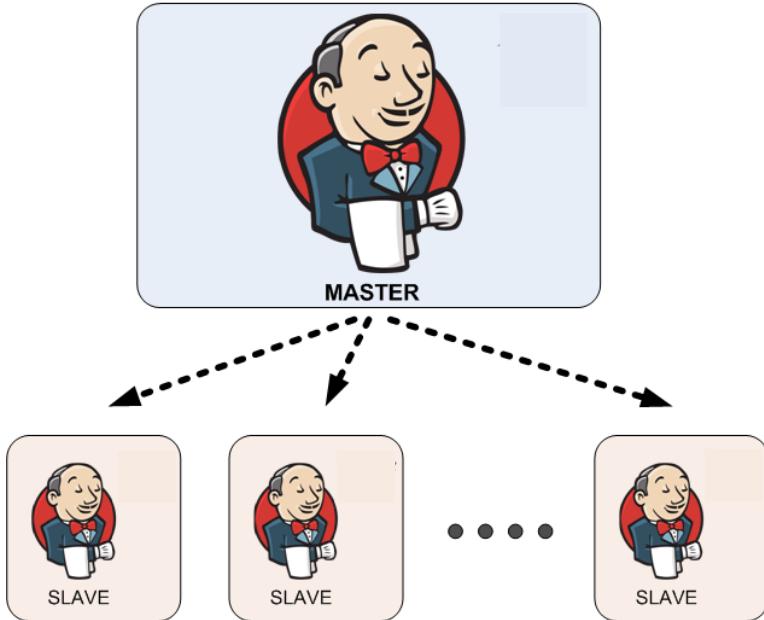
-  [AccuRev Plugin](#) — This plugin allows you to use AccuRev as a SCM.
-  [Anchore Container Image Scanner Plugin](#) — Allows users to add a build step to run the Anchore container image scanner.
-  [archive-files-scm-plugin](#) — ArchiveFilesSCM - This plugin for Jenkins checkouts archive files and extracts to Jenkins job workspace
-  [AWS CodePipeline Plugin](#) — AWS CodePipeline is a continuous delivery service for fast and reliable application updates.
-  [Bazaar Plugin](#) — This plugin integrates Bazaar version control system to Jenkins. The plugin requires the Bazaar binary (bzr) to be installed on the target machine.
-  [Bitbucket Branch Source Plugin](#) — Multibranch projects and Team/Project folders from Bitbucket Cloud and Server. Please note that this plugin requires a server running BitBucket 4.0 or later; Stash 3.x and earlier are not supported.
-  [BitKeeper Plugin](#) — Add BitKeeper support to Jenkins
-  [BlameSubversion](#) — This plug-in provides utilities for getting svn info from upstream job to downstream job
-  [ClearCase Plugin](#) — Integrates Jenkins with ClearCase.
-  [ClearCase UCM Baseline Plugin](#) — Allows using ClearCase UCM baselines as the input of builds: When using this SCM, users will be asked at build-time to select the baseline on which the job has to work.
-  [ClearCase UCM Plugin](#) — A Pragmatic integration to ClearCase UCM, simplifying continuous integration with Jenkins.
-  [Clone Workspace SCM Plugin](#) — This plugin makes it possible to archive the workspace from builds of one project and reuse them as the SCM source for another project.
-  [CMVC Plugin](#) — This plugin integrates CMVC to Hudson.
-  [Compuware Source Code Download for Endevor, PDS, and ISPW Plugin](#) — The Compuware Source Code Download for Endevor, PDS, and ISPW plugin allows Jenkins users to download Endevor, PDS, or ISPW members from the mainframe to the PC.
-  [Config Rotator Plugin](#)
-  [CVS Plugin](#) — This bundled plugin integrates Jenkins with CVS version control system.
-  [Darcs Plugin](#) — This plugin integrates Darcs version control system to Jenkins. The plugin requires the Darcs binary (darcs) to be installed on the target machine.
-  [Dimensions Plugin](#) — This plugin integrates the Serena Dimensions CM SCM with Jenkins.
-  [File System SCM](#) — Use File System as SCM.

- Jenkins' Master and Slave Architecture
- Some Important Jenkins' Terminologies



Jenkins

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.

Jenkins UI Overview



Jenkins

Install GIT and GitHub plugin



Jenkins

Install and Configure Maven



Jenkins

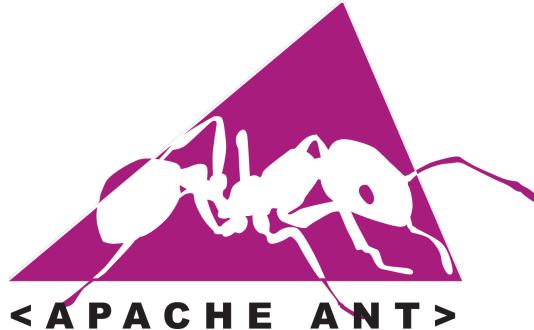
What does Maven do?

- Maven describes how the software is built.
- Maven describes the project's dependencies.

Java Build Tools

Maven™

 gradle



Configure Jenkins for a Maven -based project



Jenkins

Create a Maven -based Jenkins project



Jenkins

Run Maven-based Jenkins project



Jenkins

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.

Different Phases in Maven Build Lifecycle

- validate** Validate the project is correct and all necessary information is available.
- compile** Compile the source code of the project.
- test** Test the compiled source code using a suitable unit testing framework.
- package** Take the compiled code and package it in its distributable format.
- verify** Run any checks on results of integration tests to ensure quality criteria are met.
- install** Install the package into the local repository, for use as a dependency in other projects locally.
- deploy** Copy the final package to the remote repository for sharing with other developers and projects.

Maven Build Phases

- These lifecycle phases are executed sequentially to complete the default lifecycle.
- We want to specify the maven **package** command, this command would execute each default life cycle phase in order including **validate**, **compile**, **test** before executing **package**.
- We only need to call the last build phase to be executed.

Jenkins code quality metrics report



Jenkins

About
Checkstyle
Release Notes
Consulting
Sponsoring
Documentation
Configuration
Property Types
Filters
File Filters
Running
Ant Task
Command Line
Checks
Annotations
Block Checks
Class Design
Coding
Headers
Imports
Javadoc Comments
Metrics
Miscellaneous
Modifiers
Naming Conventions
Regexp
Size Violations
Whitespace
Style Configurations
Google's Style
Sun's Style
Developers
Extending Checkstyle
Writing Checks
Writing Javadoc Checks
Writing Filters
Writing File Filters
Writing Listeners
Contributing
Beginning Development
Eclipse IDE
NetBeans IDE
IntelliJ IDE
Javadoc
Project Documentation
Project Information
CI Management
Dependencies

Overview

Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task. This makes it ideal for projects that want to enforce a coding standard.

Checkstyle is highly configurable and can be made to support almost any coding standard. An example configuration files are supplied supporting the Sun Code Conventions [🔗](#), Google Java Style [🔗](#).

A good example of a report that can be produced using Checkstyle and Maven [🔗](#) can be seen [here](#) [🔗](#).

Checkstyle is a code static analysis tool to help programmers to write Java code that adheres to a coding standard such as

As of September 2013, Checkstyle is hosted using GitHub for hosting ([repo](#)) and CI:

- [GitHub Source code repository](#) [🔗](#) - replacing the Mercurial repository on SourceForge.
- [GitHub Issue management](#) [🔗](#) - replacing the Bugs/Feature/Patches on SourceForge. All new issues should be raised at GitHub, and pull requests are now the preferred way to submit patches.

SourceForge will still be used for website hosting and binary hosting for downloads.

Releases will happen at the end of each month if functional changes exists in [master branch of our repo](#) [🔗](#).

• **Avoiding multiple blank lines;**

Features

• **Removing unused variables;**

• **Enforcing correct indentations;**

Checkstyle can check many aspects of your source code. It can find class design problems, method design problems. It also has the ability to check code layout and formatting issues.

For a detailed list of available checks please refer to the [Checks](#) page.

Download

Downloaded from the SourceForge download page [🔗](#), or [Maven central](#) [🔗](#).

<https://github.com/checkstyle/checkstyle>



2 Added by [Ulli Hafner](#), last edited by [Ulli Hafner](#) on Jun 01, 2016 ([view change](#))

Edit

Add

Tools

Message from a Patron of Jenkins

Jenkins

[Home](#)
[Mailing lists](#)
[Source code](#)
[Bugtracker](#)
[Security Advisories](#)
[Events](#)
[Donation](#)
[Commercial Support](#)
[Wiki Site Map](#)

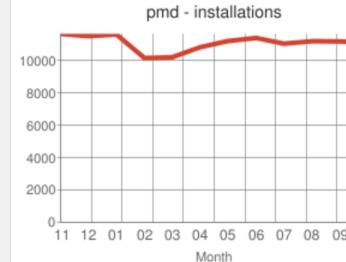
Documents

[Meet Jenkins](#)
[Use Jenkins](#)
[Extend Jenkins](#)
[Plugins](#)
[Servlet Container Notes](#)

Plugin Information

Plugin ID	pmd
Latest Release	3.45 (archives)
Latest Release Date	Jun 01, 2016
Required Core Dependencies	1.596.1 maven-plugin (version:2.9) matrix-project (version:1.2.1) token-macro (version:1.10, optional) analysis-core (version:1.77) dashboard-view (version:2.9.4, optional)

Usage



PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It supports Java, JavaScript, Salesforce.com Apex, PLSQL, Apache Velocity, XML, XSL. Additionally it includes CPD, the copy-paste-detector. CPD finds duplicated code in Java, C, C++, C#, Groovy, PHP, Ruby, Fortran, JavaScript, PLSQL, Apache Velocity, Ruby, Scala, Objective C, Matlab, Python, Go, Swift and Salesforce.com Apex.

- Latest version(s)
- Get Involved
- Plugins
- Recent Announcements
- Next development version
- Previous versions

Latest version(s)

5.5.2 (5th November 2016)

- [Release Notes](#)
- [Download \(Sourcecode, Documentation\)](#)
- [Online Documentation](#)
- Requires at least java 7, Salesforce.com Apex requires at least java 8

5.4.3 (4th November 2016)

- [Release Notes](#)
- [Download \(Sourcecode, Documentation\)](#)
- [Online Documentation](#)
- Requires at least java 7

Fork me on GitHub





FindBugs Plugin

12 Added by Ulli Hafner, last edited by Ulli Hafner on Jun 01, 2016 (view change)

Edit

Add

Tools

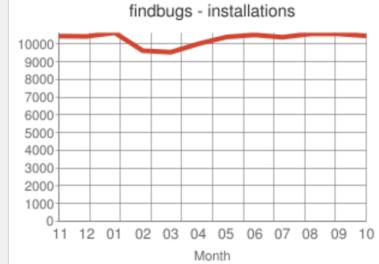
Jenkins

Home
Mailing lists
Source code
Bugtracker
Security Advisories
Events
Donation
Commercial Support
Wiki Site Map

Documents

Meet Jenkins
Use Jenkins
Extend Jenkins
Plugins
Servlet Container Notes

Plugin Information

Plugin ID	findbugs	Changes	In Latest Release Since Latest Release																										
Latest Release Latest Release Date Required Core Dependencies	4.65 (archives) Jun 01, 2016 1.596.1 <code>matrix-project</code> (version:1.2.1) <code>analysis-core</code> (version:1.77) <code>maven-plugin</code> (version:2.9) <code>dashboard-view</code> (version:2.9.4, optional) <code>token-macro</code> (version:1.10, optional)	Source Code Issue Tracking Pull Requests Maintainer(s) Ulli Hafner (id: drulli)	GitHub Open Issues Pull Requests Ulli Hafner (id: drulli)																										
Usage	 <p>findbugs - installations</p> <table border="1"><thead><tr><th>Month</th><th>Installations</th></tr></thead><tbody><tr><td>11</td><td>10439</td></tr><tr><td>12</td><td>10423</td></tr><tr><td>01</td><td>10629</td></tr><tr><td>02</td><td>9619</td></tr><tr><td>03</td><td>9531</td></tr><tr><td>04</td><td>10005</td></tr><tr><td>05</td><td>10391</td></tr><tr><td>06</td><td>10507</td></tr><tr><td>07</td><td>10376</td></tr><tr><td>08</td><td>10565</td></tr><tr><td>09</td><td>10564</td></tr><tr><td>10</td><td>10452</td></tr></tbody></table>	Month	Installations	11	10439	12	10423	01	10629	02	9619	03	9531	04	10005	05	10391	06	10507	07	10376	08	10565	09	10564	10	10452	Installations	2015-Nov 10439 2015-Dec 10423 2016-Jan 10629 2016-Feb 9619 2016-Mar 9531 2016-Apr 10005 2016-May 10391 2016-Jun 10507 2016-Jul 10376 2016-Aug 10565 2016-Sep 10564 2016-Oct 10452
Month	Installations																												
11	10439																												
12	10423																												
01	10629																												
02	9619																												
03	9531																												
04	10005																												
05	10391																												
06	10507																												
07	10376																												
08	10565																												
09	10564																												
10	10452																												

build passing

This plugin generates the trend report for [FindBugs](#), an open source program which uses static analysis to look for bugs in Java code.

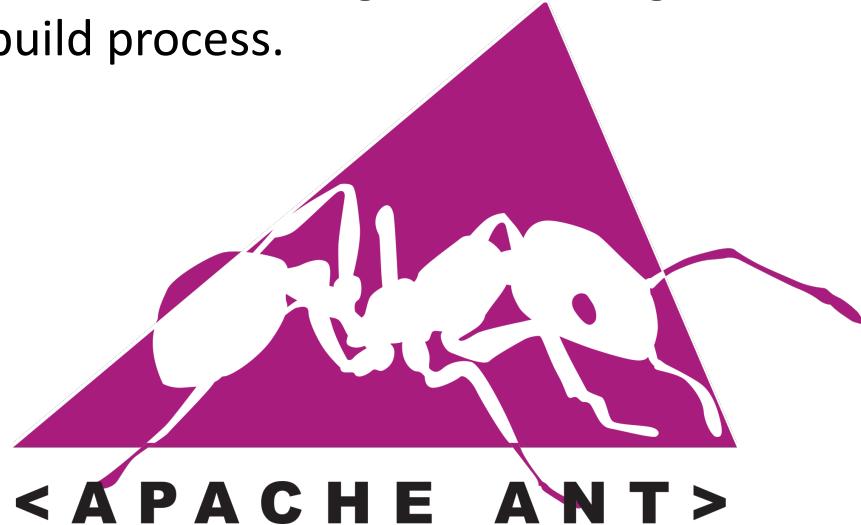
Jenkins' support for other build systems (Ant, Gradle and shell scripts)



Jenkins

Apache Ant

- Widely-used and very well-known build scripting language for Java.
- Flexible, extensible, relatively low-level scripting language.
- An Ant build script is made up of a number of targets, each target performs a particular job in the build process.



Gradle

- Gradle is a relatively new open source build tool for the Java Virtual Machine.
- Build scripts for Gradle are written in a Domain Specific Language based on Groovy.
- The concise nature of Groovy scripting lets you write very expressive build scripts with very little code.



Build Scripts

Maven Build Script

```
<project xmlns:ivy="antlib:org.apache.ivy.ant" name="java-build-tool"

<property name="src.dir" value="src"/>
<property name="build.dir" value="build"/>
<property name="classes.dir" value="${build.dir}/classes"/>
<property name="jar.dir" value="${build.dir}/jar"/>
<property name="lib.dir" value="lib" />
<path id="lib.path.id">
    <fileset dir="${lib.dir}" />
</path>

<target name="resolve">
    <ivy:retrieve />
</target>

<target name="clean">
    <delete dir="${build.dir}" />
</target>

<target name="compile" depends="resolve">
    <mkdir dir="${classes.dir}" />
    <javac srcdir="${src.dir}" destdir="${classes.dir}" classpat
</target>

<target name="jar" depends="compile">
    <mkdir dir="${jar.dir}" />
    <jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="
</target>

</project>
```

\

Gradle Build Script

Ant Build Script Sample

```
<ivy-module version="2.0">
    <info organisation="org.apache" module="java-build-tools"/>
    <dependencies>
        <dependency org="junit" name="junit" rev="4.11"/>
        <dependency org="org.hamcrest" name="hamcrest-all" rev="1.3"/>
    </dependencies>
</ivy-module>
```

Install and configure Tomcat as a staging environment



Jenkins

Tomcat

Tomcat is an open-source web server and provides a "pure Java" HTTP web server environment in which Java code can run.



- Install *copy artifact* and *deploy to container* plugins
- Deploy our application to staging environment



Jenkins

Jenkins Build Pipeline



Jenkins

Build Pipeline Plugin

Dashboard > Jenkins > ... > Plugins > Build Pipeline Plugin

Browse Search

 **Build Pipeline Plugin**

20 Added by Geoff Bullen, last edited by Dan Alvizio on Jul 28, 2016 ([view change](#))

Jenkins

- Home
- Mailing lists
- Source code
- Bugtracker
- Security Advisories
- Events
- Donation

Summary

This plugin provides a *Build Pipeline View* of upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins.

Plugin Information

Plugin ID	build-pipeline-plugin	Changes	In Latest Release Since Latest Release
-----------	-----------------------	---------	---

Pipeline version
8
No parameters

Test
Jun 26, 2012 5:30:48 PM * 8 10 sec marcinp

Release
Jun 26, 2012 5:31:03 PM * 8 12 sec

Deploy to Test
Jun 26, 2012 5:31:46 PM * 6 ... marcinp

Deploy to Pre-Prod

Deploy to Prod

Generate docs
Jun 26, 2012 5:31:20 PM * 8 9.1 sec



Parallel Jenkins Build



Jenkins

Continuous Delivery

Deploy our app to production



Jenkins

Benefits of a code-based pipeline

- Version control
- Best Practices
- Less error-prone execution of jobs
- Logic-based execution of steps

```
1
2 pipeline {
3     agent any
4     stages {
5         stage ('Initialize') {
6             steps {
7                 sh '''
8                     echo "PATH = ${PATH}"
9                     echo "M2_HOME = ${M2_HOME}"
10                '''
11            }
12        }
13        stage ('Build') {
14            steps {
15                echo 'Hello World!'
16            }
17        }
18    }
19 }
```

Additional automation

- Setup Git repository polling
- Deployment to our tomcat servers
- We will setup tasks to run in parallel

Steps

- **Step 1:** Configure security groups for Tomcat servers and create key pairs.
- **Step 2:** Provision instances to staging and production environments.
- **Step 3:** Install and run Tomcat on created instances.
- **Step 4:** Fully automate our existing Jenkins pipeline.

Introduction to Distributed Jenkins Builds



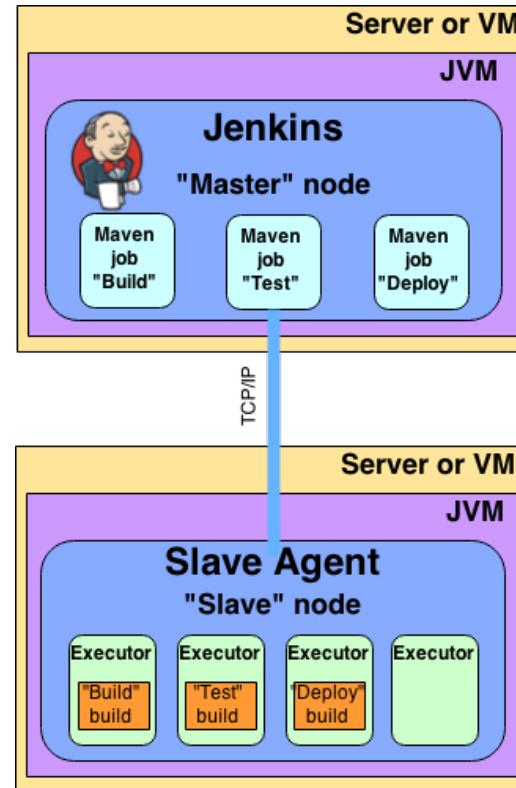
Jenkins

Install Jenkins Master in the Cloud



Jenkins

Jenkins Slave Agent



Install Jenkins slaves in the cloud and form a Jenkins cluster



Jenkins

Concurrent Builds on Jenkins Cluster

Label Jenkins Nodes

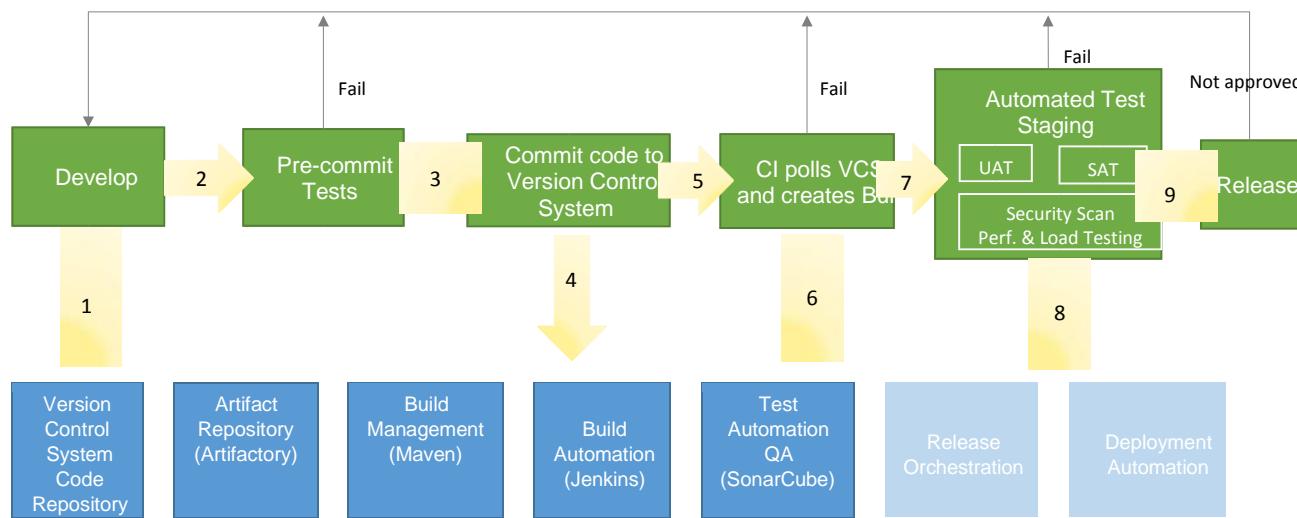


U.S. Department of Commerce
Economics and Statistics Administration
U.S. CENSUS BUREAU
census.gov

Build Orchestration: Jenkins

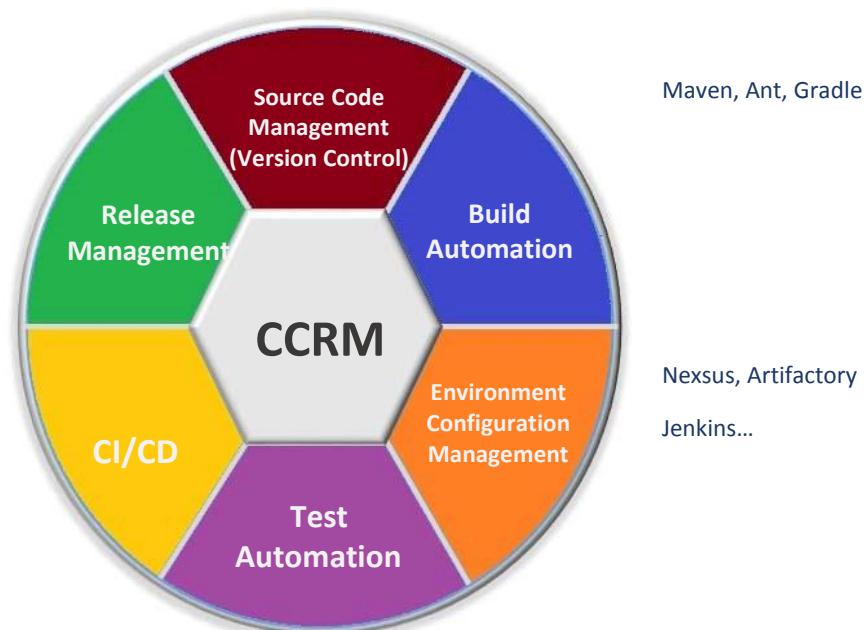
- Continuous integration system
- Enable automated build and test process
- Can monitoring executions of externally-run jobs, such as cron jobs and procmail jobs...
- Dependency tracking, allowing file finger printing and tracking for example which build is using which version of jars...
- Generates list of changes made to build from Subversion
- Distributed build/test
- Jenkins is a build orchestration, CI software
 - building/testing software projects continuously
 - monitoring executions of externally-run jobs
- FishEye allows you to extract information from your source code repository and display it in sophisticated reports.
- Crucible allows you to request, perform and manage code reviews.
- Subversion centralized version control system
- Sonar is a quality management platform for analyzing and measuring source code quality.

CI/CD Pipeline: Functional Architecture



An Automated, Integrated and End to Ent CCRM

SVN...
Git, GitLab, GitHub Enterprise,



Runners

- Distributed
- Even local

