

# Jenkins

Configuring Jobs

# What is Jenkins?

Originally called Hudson, it is an open source Continuous Integration tool written in Java.

# **Well what is Continuous Integration?**

CI born to address the pitfalls of the Integration Phase of waterfall practices.

# Continuous Integration attributes

- Immediate change detection
- Code base health (code quality/code coverage)
- Automated acceptance tests
- Communication
- Automated deployment process

# Continuous Integration summary

Essentially CI is about reducing risk by providing faster feedback by:

- identify and fix integration/regression issues faster
- better visibility for tech/non-tech on state of project
- automation gets software to testers and end-users faster

# Continuous Deployment

Automation of deployment process for pushing every successful build directly into production is generally known as Continuous Deployment.

# Continuous Delivery

is a slight variation of Continuous Deployment where every successful build can *potentially* be deployed into production.

# CI areas to address

- time to get small code change out to production
- problems due to conflicts between developers resolved earlier
- labor intensive manual testing by QA teams
- manual deployment steps where knowledge known by select few



# Implementing Continuous Integration

CI is no “silver bullet”. Takes time to fully implement with incremental improvements to technical infrastructure as well as improvements to practices and culture of teams.

# Jenkins

- Branched from Hudson
- Java based Continuous Build System
- Runs in servlet container Glassfish, Tomcat or as service
- Supported by hundreds plugins
  - SCM, Testing, Notifications, Reporting, Artifact Saving, Triggers, External Integration
- Under development since 2005
- <http://jenkins-ci.org>

# What Jenkins offers

- Building/testing software projects continuously
  - Jenkins provides an easy-to-use continuous integration system, easier for developers to integrate project changes, and easier to obtain a fresh build. The automated, continuous build increases the productivity.
- Monitoring executions of externally-run jobs
  - Jenkins keeps those outputs and makes it easy for you to notice when something is wrong.

# Why Jenkins?

- Flexibility
  - Jenkins is highly configurable by itself
  - Community developed plugins offer more flexibility
  - Combine with Ant, Maven, Gradle, or other build systems

# Why Jenkins?

- Free / OSS
  - Jenkins is released under the MIT license
  - There is a large support community and thorough documentation
  - It's easy to write plugins
  - Think something is wrong with it, you can fix it!

# What can Jenkins do?

- Generate test reports
- Integrate with many different Version Control Systems
- Push to various artifact repositories
- Deploys directly to production or test environments
- Notify stakeholders of build status
- and more

# Jenkins out of the box

- Associating with a version control server
- Triggering builds
  - Polling, Periodic, Building based on other projects
- Execution of shell scripts, bash scripts, Ant targets, and Maven targets
- Artifact archival
- Publish JUnit test results and Javadocs
- Email notifications

As stated earlier, plugins expand the functionality even further

# Jenkins pre-requisites

- Java 6 or higher (JDK 1.6+)
- Maven



# Install JDK

- go to <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- download Java SE 7u51 JDK
- create JAVA\_HOME variable

# Install Maven

- go to: <http://maven.apache.org/download.cgi>
- grab apache-maven-3.1.1-bin.zip/tar.gz
- unzip into desired location: C:\apache-maven-3.1.1 or /root/apache-maven-3.1.1
- add to Path

# Generate Maven Project

- create project directory
  - o C:\Projects or /root/Projects
- run command

```
mvn archetype:generate \
    -DgroupId=com.orasi \
    -DartifactId=maven-training \
    -Dversion=1.0 \
    -DinteractiveMode=false
```

# Let's Install Jenkins

- download jenkins.war
  - <http://mirrors.jenkins-ci.org/war-stable/latest/jenkins.war>
- move to C:\ or /root
- run war
  - `java -jar <path to war file>`

# Jenkins Dashboard

## Jenkins

search

log in | sign up

Jenkins

ENABLE AUTO REFRESH

add description

New Item

People

Build History

Manage Jenkins

Credentials

Build Queue

No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

All Pipeline +

S	W	Name	Last Success	Last Failure	Last Duration
		<a href="#">blah</a>	N/A	N/A	N/A
		<a href="#">demo</a>	19 days - <a href="#">#67</a>	18 days - <a href="#">#89</a>	6.6 sec
		<a href="#">HelloWorldVS</a>	2 mo 25 days - <a href="#">#125</a>	2 mo 25 days - <a href="#">#120</a>	5.1 sec
		<a href="#">release-notes-check</a>	4 mo 7 days - <a href="#">#33</a>	4 mo 7 days - <a href="#">#40</a>	2.3 sec
		<a href="#">test</a>	21 hr - <a href="#">#5</a>	N/A	11 sec

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

# Jenkins

Jenkins



[New Job](#)



[People](#)



[Build History](#)



[Manage Jenkins](#)

Welcome to Jenkins! Please [create new jobs](#) to get started.

## Build Queue

No builds in the queue.

## [Build Executor Status](#)

#	Status	
1	Idle	
2	Idle	



[Help us localize this page](#)

Click Manage Jenkins then Configure System

# Need to point to JDK/Maven

## JDK

JDK installations

⌵ JDK

Name

jdk1.7.0\_51

JAVA\_HOME

C:\Program Files\Java\jdk1.7.0\_51

☐

Install automatically



Delete JDK

Add JDK

List of JDK installations on this system

## Git

Git installations

⌵ Git

Name

Default

Path to Git executable

git.exe

☐

Install automatically



Delete Git

Add Git ▼

description

## Gradle

Gradle installations

⌵ Gradle

name

Gradle-1.8

GRADLE\_HOME

C:\gradle-1.8

☐

Install automatically



Delete Gradle

[New Item](#)[People](#)[Build History](#)[Manage Jenkins](#)[Credentials](#)**Build Queue**

No builds in the queue.

**Build Executor Status**

#	Status
1	Idle
2	Idle

Item name

☒ **Build a free-style software 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.

☐ **Build a maven2/3 project**

Build a maven 2/3 project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

☐ **Build multi-configuration project**

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

☐ **Monitor an 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](#).

☐ **Copy existing Item**

Copy from

OK



 [Back to Dashboard](#) [Status](#) [Changes](#) [Workspace](#) [Build Now](#) [Delete Project](#) [Configure](#)**Build History** (trend)  [RSS for all](#)  [RSS for failures](#)Project name 

Description

[\[Raw HTML\]](#) [Preview](#)



- ☐ Discard Old Builds
- ☐ This build is parameterized
- ☐ Promote builds when...
- ☐ Disable Build (No new builds will be executed until the project is re-enabled.)
- ☐ Execute concurrent builds if necessary

**Advanced Project Options**[Advanced...](#)**Source Code Management**

- ☐ CVS
- ☐ CVS Projectset
- ☐ Git
- ☒ None
- ☐ Subversion

**Build Triggers**

- ☐ Build after other projects are built
- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build periodically
- ☐ Build when another project is promoted
- ☐ Poll SCM

**Build**[Add build step](#) **Post-build Actions**[Add post-build action](#) [Save](#)[Apply](#)

☐ Poll SCM



## Pre Steps

Add pre-build step ▼

## Build

Root POM

C:\Projects\maven-training\pom.xml



Goals and options

clean install



Advanced...

## Post Steps

☒ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☐ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

## Execute Windows batch command



Command

```
java -cp J:\Projects\maven-training\target\maven-training-1.0-SNAPSHOT.jar com.orasi.App
```

# Job page

# Jenkins

[log in](#) | [sign up](#)

Jenkins > test > [ENABLE AUTO REFRESH](#)

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

## Project test

[Workspace](#)

[Recent Changes](#)

[add description](#)

[Disable Project](#)

**Build History** [\(trend\)](#)

#5 [Feb 25, 2014 4:28:56 PM](#)

#4 [Feb 25, 2014 4:28:07 PM](#)

#3 [Feb 25, 2014 3:36:52 PM](#)

#2 [Feb 25, 2014 3:35:27 PM](#)

#1 [Feb 25, 2014 3:34:05 PM](#)

[RSS for all](#) [RSS for failures](#)

## Permalinks

- [Last build \(#5\), 21 hr ago](#)
- [Last stable build \(#5\), 21 hr ago](#)
- [Last successful build \(#5\), 21 hr ago](#)

[Help us localize this page](#)

Page generated: Feb 26, 2014 2:15:37 PM

[REST API](#)

[Jenkins ver. 1.551](#)

# Changing Jenkins port in jenkins.xml

```
31 <service>
32   <id>jenkins</id>
33   <name>Jenkins</name>
34   <description>This service runs Jenkins continuous integration system.</description>
35   <env name="JENKINS_HOME" value="%BASE%" />
36   <!--
37     if you'd like to run Jenkins with a specific version of Java, specify a full path to java.exe.
38     The following value assumes that you have java in your PATH.
39   -->
40   <executable>%BASE%\jre\bin\java</executable>
41   <arguments>-Xrs -Xmx256m -Dhudson.lifecycle=udson.lifecycle.WindowsServiceLifecycle -jar "%BASE%\jenkins.war" --httpPort=9090</arguments>
42   <!--
43     interactive flag causes the empty black Java window to be displayed.
44     I'm still debugging this.
45   <interactive />
46   -->
47   <logmode>rotate</logmode>
48
49   <onfailure action="restart" />
50 </service>
```

# Extending Jenkins with Plugins

Manage Jenkins -> Manage plugins

# Conclusion

- Continuous integration is a necessity on complex projects due to the benefits it provides regarding early detection of problems
- A good continuous build system should be flexible enough to fit into pre-existing development environments and provide all the features a team expects from such a system
- Jenkins, a continuous build system, can be an integral part of any continuous integration system due to its core feature set and extensibility through a plugin system