Jenkins: Continuous Integration

Sang Shin
JPassion.com
"Learn with Passion!"

Topics

- What is and why Continuous Integration (CI)?
- What is and Why Jenkins?
- Jenkins architecture & flow of operations
- Plug-in's
- How to get started
- Configuring Jenkins
- Fingerprint
- Distributed building
- Jenkins best practices
- Alternative solutions

What is & Why Continuous Integration (CI)?

What is Continuous Integration (CI)?

- “Continuous Integration (Ci) 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

Principles of CI

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

Benefits of CI (1)

- Constant availability of a "current" build for testing, demo, or release purposes
- Immediate feedback to developers on the quality, functionality, or system-wide impact of code they are writing
- Metrics generated from automated testing and CI (such as metrics for code coverage, code complexity, and features complete) help developers write quality code

Benefits of CI (2)

- Developers detect and fix integration problems continuously - avoiding last-minute chaos at release dates, (when everyone tries to check in their highly likely incompatible versions)
  - Early warning of broken/incompatible code
  - Early warning of conflicting changes
  - Immediate unit testing of all changes

CI Impact to All Software Lifecycle

- Build Management
- Release Management
- Deployment Automation
- Test Orchestration

What is & Why Jenkins?

What is Jenkins?

- Jenkins is an “Continuous Integration” (CI) server
- It is a derivative of Hudson
  - Jenkins and Hudson share same configuration file format

Key Features of Jenkins (1)

- Easy installation
  - Just *java -jar jenkins.war –httpPort 9001* for testing
  - No additional install, no database setup
  - Use a native package or deploy it in a servlet container for production use
- Easy configuration with Web-based interface
  - Jenkins can be configured entirely from its friendly web GUI with extensive on-the-fly error checks and inline help
  - There's no need to tweak XML manually anymore, although if you'd like to do so, you can do that, too.

Key Features of Jenkins (2)

- Change set support
  - Jenkins can generate a list of changes made into the build from SCM systems like CVS, Subversion, Git and many others..
  - This is done in a fairly efficient fashion, to reduce the load on the repository
- Permanent links
  - Jenkins gives you clean readable URLs for most of its pages, including some permalinks like "latest build" or "latest successful build", so that they can be easily linked from elsewhere
- RSS/E-mail/IM Integration
  - Monitor build results by RSS or e-mail to get real-time notifications on failures

Key Features of Jenkins (3)

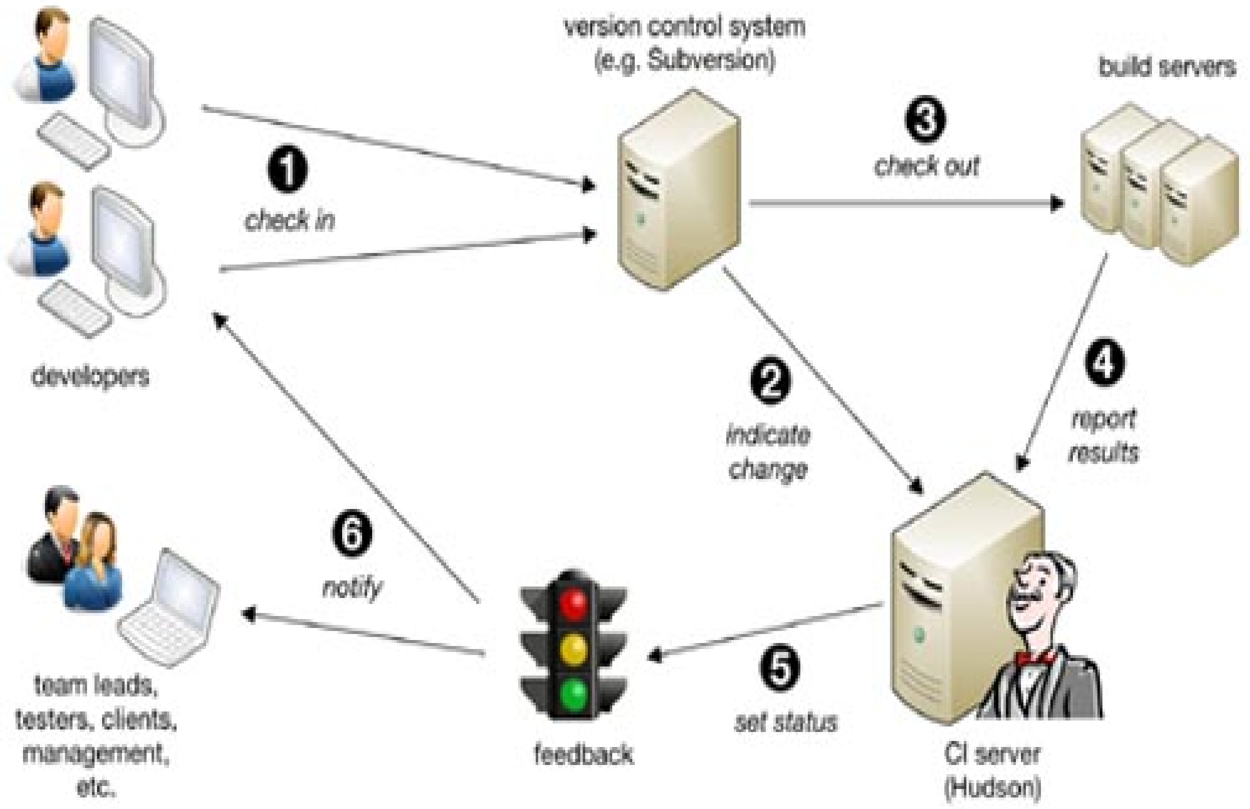
- After-the-fact tagging
  - Builds can be tagged long after builds are completed
- JUnit/TestNG test reporting
  - JUnit test reports can be tabulated, summarized, and displayed with history information, such as when it started breaking, etc. History trend is plotted into a graph.
- Distributed builds
  - Jenkins can distribute build/test loads to multiple computers. This lets you get the most out of those idle workstations sitting beneath developers' desks.

Key Features of Jenkins (4)

- File fingerprinting
  - Jenkins can keep track of which build produced which jars, and which build is using which version of jars, and so on.
  - This works even for jars that are produced outside Jenkins, and is ideal for projects to track dependency.
- Plugin Support:
  - Jenkins can be extended via 3rd party plugins.
  - There are extensive set of plugin's available
  - You can write plugins to make Jenkins support tools/processes that your team uses

Architecture & Flow of Operations

Jenkins Architecture & Flow



Flow of Operations

- (1) Developers check in changes to version control system - Jenkins polls the version control system regularly for changes, e.g. every minute
- (2) If changes are indicated, the new code is checked out to build severs
- (3) A new build of your project is executed. The results are reported back to Jenkins
- (4) New status of the project - *stable*, *instable*, or *failed* - is created
- (5) Notifications are sent to developers and other stakeholders
- (6) Stakeholders either fix any problems found or proceed implementing new features

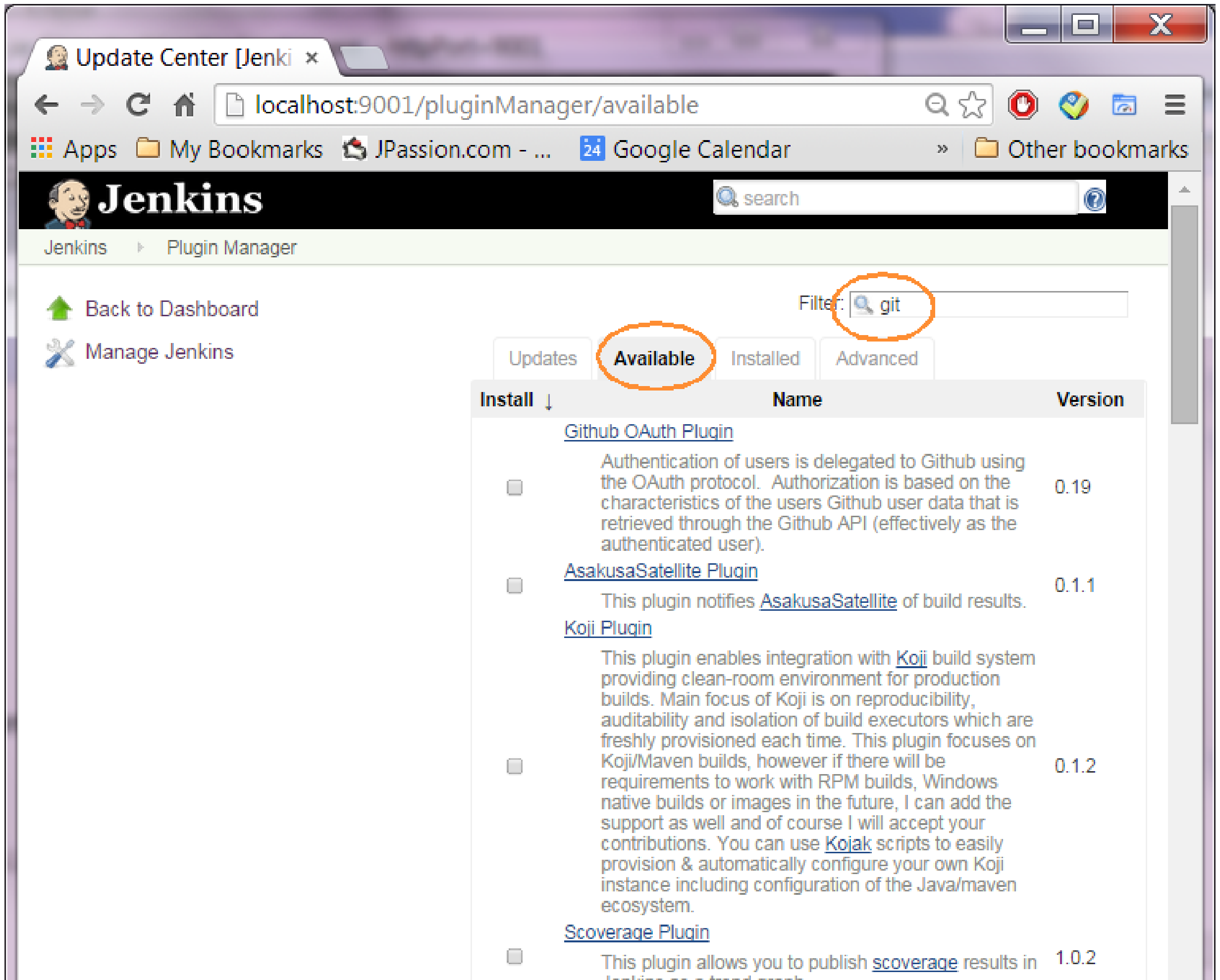
Plug-in's

Plug-in's Installed By Default

- Jenkins is designed to be extensible through plug-ins
- By default, some plug-ins are pre-installed (they are called pinned plugins)
  - CVS
  - SVN
  - Maven
  - SSH

Available Plug-in's

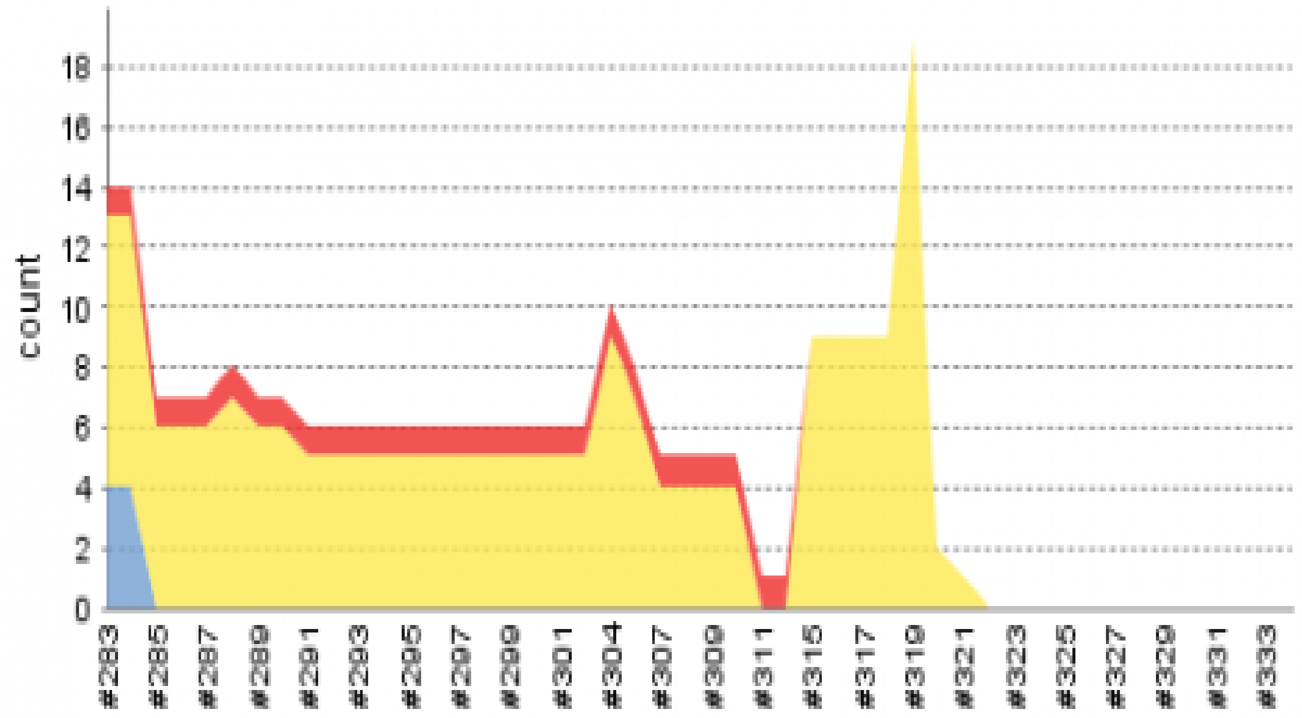
- There are many plug-ins available for various features



Install ↓	Name	Version
☐	Github OAuth Plugin Authentication of users is delegated to Github using the OAuth protocol. Authorization is based on the characteristics of the users Github user data that is retrieved through the Github API (effectively as the authenticated user).	0.19
☐	AsakusaSatellite Plugin This plugin notifies AsakusaSatellite of build results.	0.1.1
☐	Koji Plugin This plugin enables integration with Koji build system providing clean-room environment for production builds. Main focus of Koji is on reproducibility, auditability and isolation of build executors which are freshly provisioned each time. This plugin focuses on Koji/Maven builds, however if there will be requirements to work with RPM builds, Windows native builds or images in the future, I can add the support as well and of course I will accept your contributions. You can use Kojak scripts to easily provision & automatically configure your own Koji instance including configuration of the Java/maven ecosystem.	0.1.2
☐	Scoverage Plugin This plugin allows you to publish scoverage results in	1.0.2

Static Code Analysis Plugin Suite

- The static code analysis plugin provides a common visualization backend that produces Trend Graphs for code analysis tools



Static code analysis tools

- FindBugs
- DRY
- Checkstyle
- Warnings - Scans several compiler outputs for warnings
- Task Scanner – Scans for TODOs in the code

How They Work with Jenkins

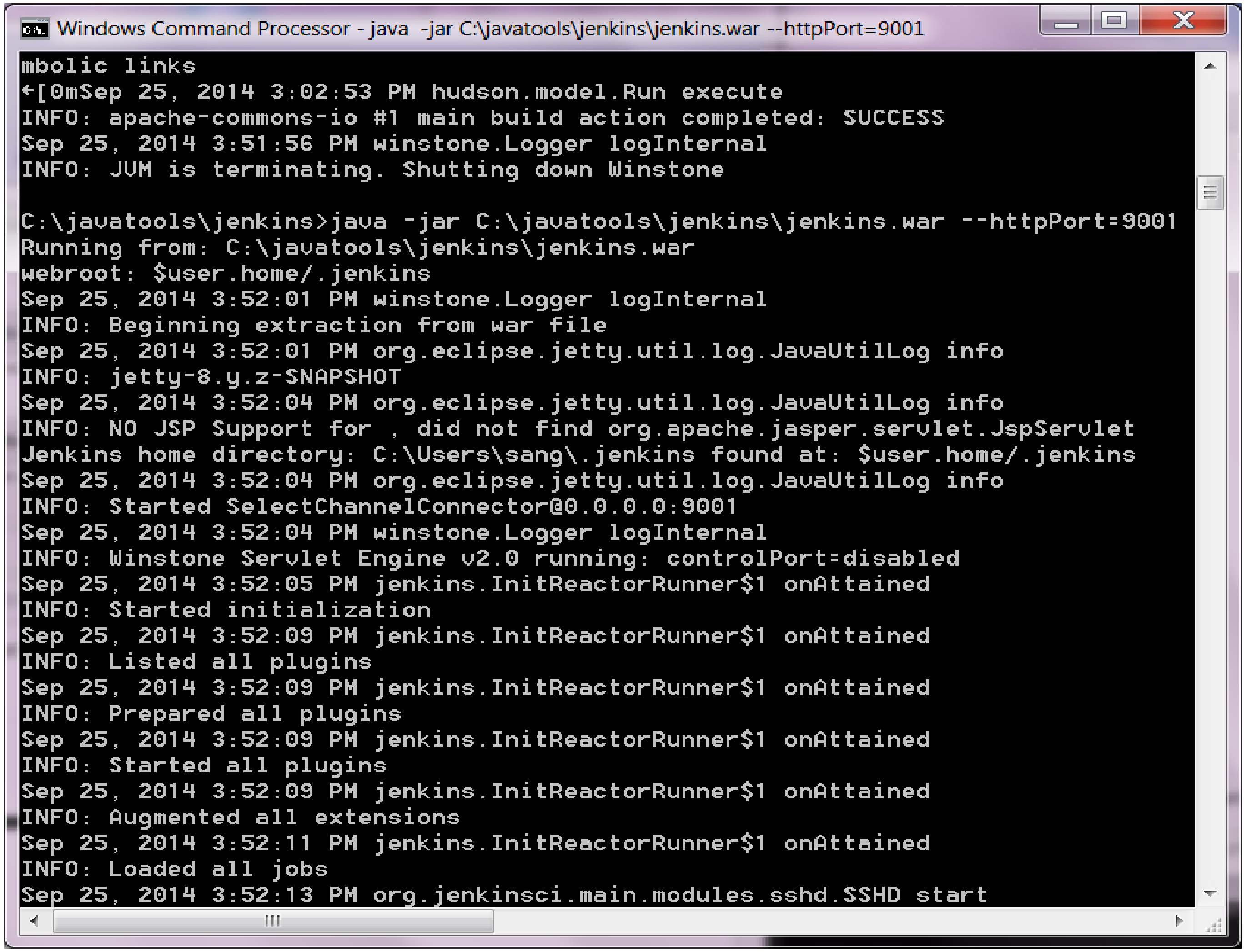
- The code analysis plugin does not actually run the code analysis tools; it merely analyzes the output of these tools.
- To exemplify this, the process for FindBugs is as follows:
  1. Jenkins calls the build tool (ANT or Maven) to run FindBugs.
  2. FindBugs writes its results to an XML file.
  3. The static code analysis plugin analyses the XML file and generates a report page, including the trend graph.

How to Get Started with Jenkins

How to Get Started with Jenkins

- Download the "jenkins-<version>.war" from *http://jenkins-ci.org/* and unzip it
- Open a console and start the server. Then from your browser, go to *http://myjenkinsserver:9001/*
  - *java -jar jenkins-<version>.war –httpPort 9001*
- Or deploy jenkins-<*version*>*.war* over a Java EE container of your choice
  - Tomcat, GlassFish, Jetty, WebLogic, WebShephe, Winstone

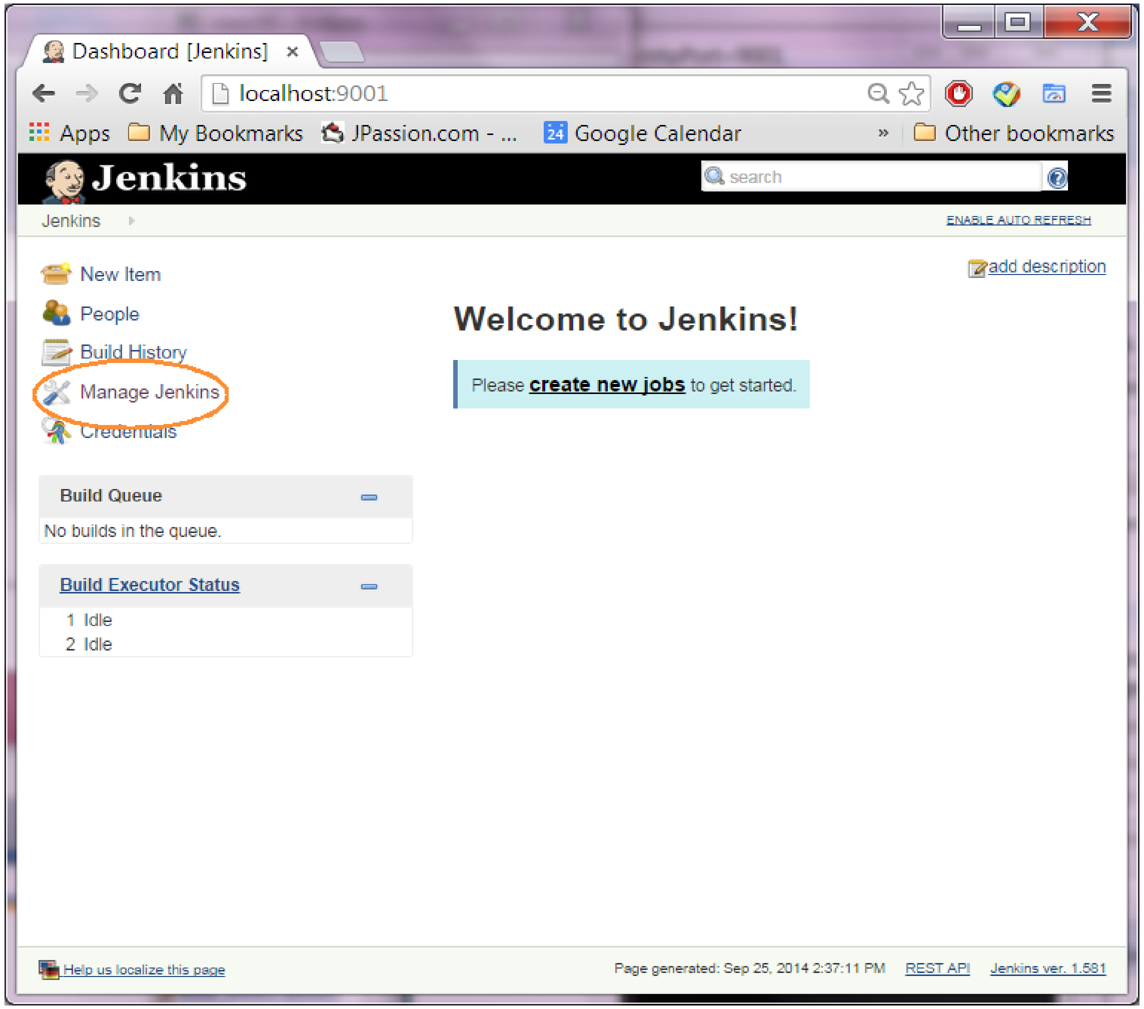
Starting Jenkins from Commandline


Windows Command Processor - java -jar C:\javatools\jenkins\jenkins.war --httpPort=9001

```
mbolic links
←[0mSep 25, 2014 3:02:53 PM hudson.model.Run execute
INFO: apache-commons-io #1 main build action completed: SUCCESS
Sep 25, 2014 3:51:56 PM winstone.Logger logInternal
INFO: JVM is terminating. Shutting down Winstone

C:\javatools\jenkins>java -jar C:\javatools\jenkins\jenkins.war --httpPort=9001
Running from: C:\javatools\jenkins\jenkins.war
webroot: $user.home/.jenkins
Sep 25, 2014 3:52:01 PM winstone.Logger logInternal
INFO: Beginning extraction from war file
Sep 25, 2014 3:52:01 PM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: jetty-8.y.z-SNAPSHOT
Sep 25, 2014 3:52:04 PM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: NO JSP Support for , did not find org.apache.jasper.servlet.JspServlet
Jenkins home directory: C:\Users\sang\.jenkins found at: $user.home/.jenkins
Sep 25, 2014 3:52:04 PM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: Started SelectChannelConnector@0.0.0.0:9001
Sep 25, 2014 3:52:04 PM winstone.Logger logInternal
INFO: Winstone Servlet Engine v2.0 running: controlPort=disabled
Sep 25, 2014 3:52:05 PM jenkins.InitReactorRunner$1 onAttained
INFO: Started initialization
Sep 25, 2014 3:52:09 PM jenkins.InitReactorRunner$1 onAttained
INFO: Listed all plugins
Sep 25, 2014 3:52:09 PM jenkins.InitReactorRunner$1 onAttained
INFO: Prepared all plugins
Sep 25, 2014 3:52:09 PM jenkins.InitReactorRunner$1 onAttained
INFO: Started all plugins
Sep 25, 2014 3:52:09 PM jenkins.InitReactorRunner$1 onAttained
INFO: Augmented all extensions
Sep 25, 2014 3:52:11 PM jenkins.InitReactorRunner$1 onAttained
INFO: Loaded all jobs
Sep 25, 2014 3:52:13 PM org.jenkinsci.main.modules.sshd.SSHD start
```

Jenkins UI



Dashboard [Jenkins]

localhost:9001

Apps · My Bookmarks · JPassion.com - ... · Google Calendar · Other bookmarks

Jenkins

search

Jenkins

ENABLE AUTO REFRESH

New Item

People

Build History

Manage Jenkins

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

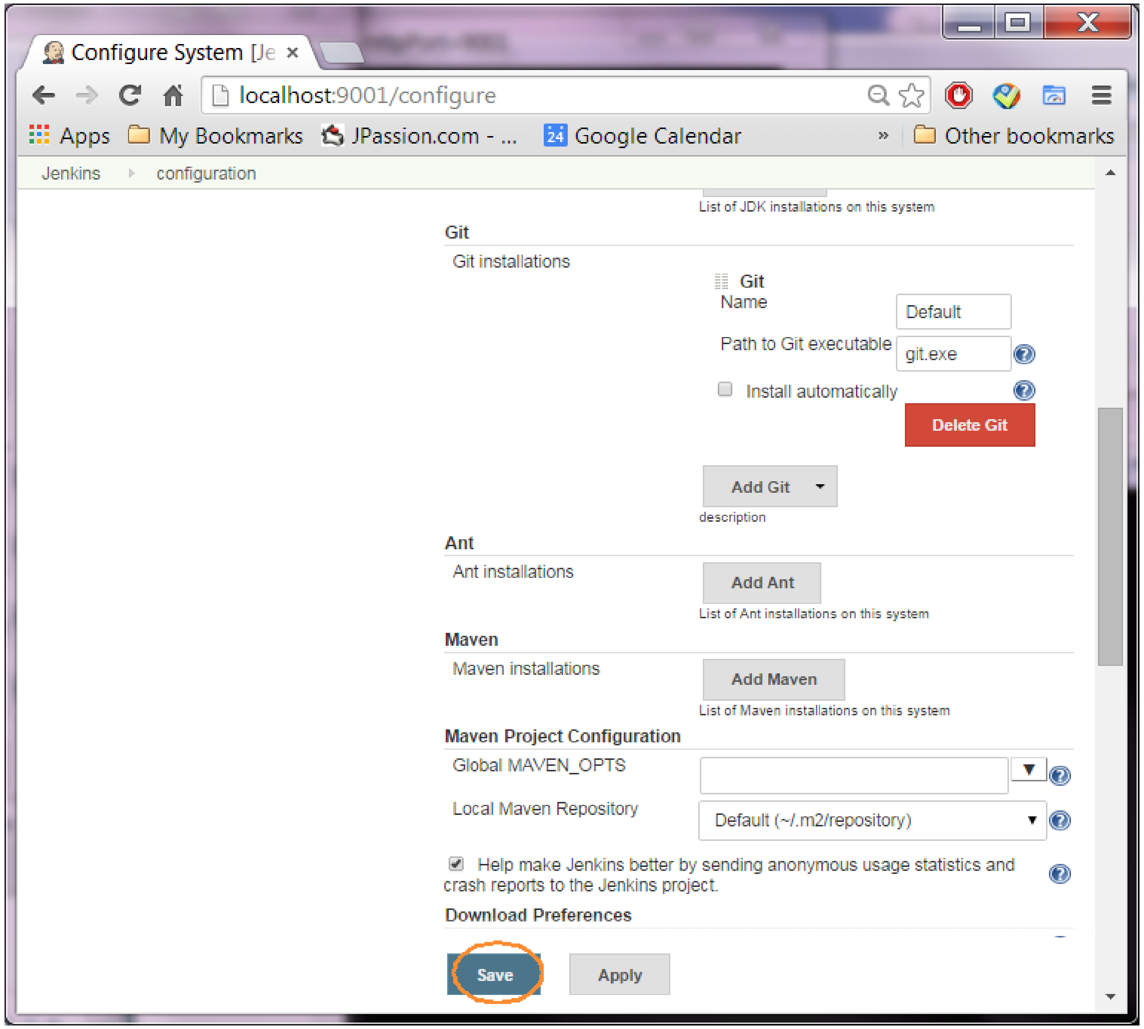
2 Idle

Help us localize this page

Page generated: Sep 25, 2014 2:37:11 PM REST API Jenkins ver. 1.581

Configuring Jenkins

Configuring Source Control Management


Configure System [Je ×

localhost:9001/configure

Apps My Bookmarks JPassion.com - ... Google Calendar » Other bookmarks

Jenkins ▸ configuration

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

Ant

Ant installations

Add Ant

List of Ant installations on this system

Maven

Maven installations

Add Maven

List of Maven installations on this system

Maven Project Configuration

Global MAVEN_OPTS

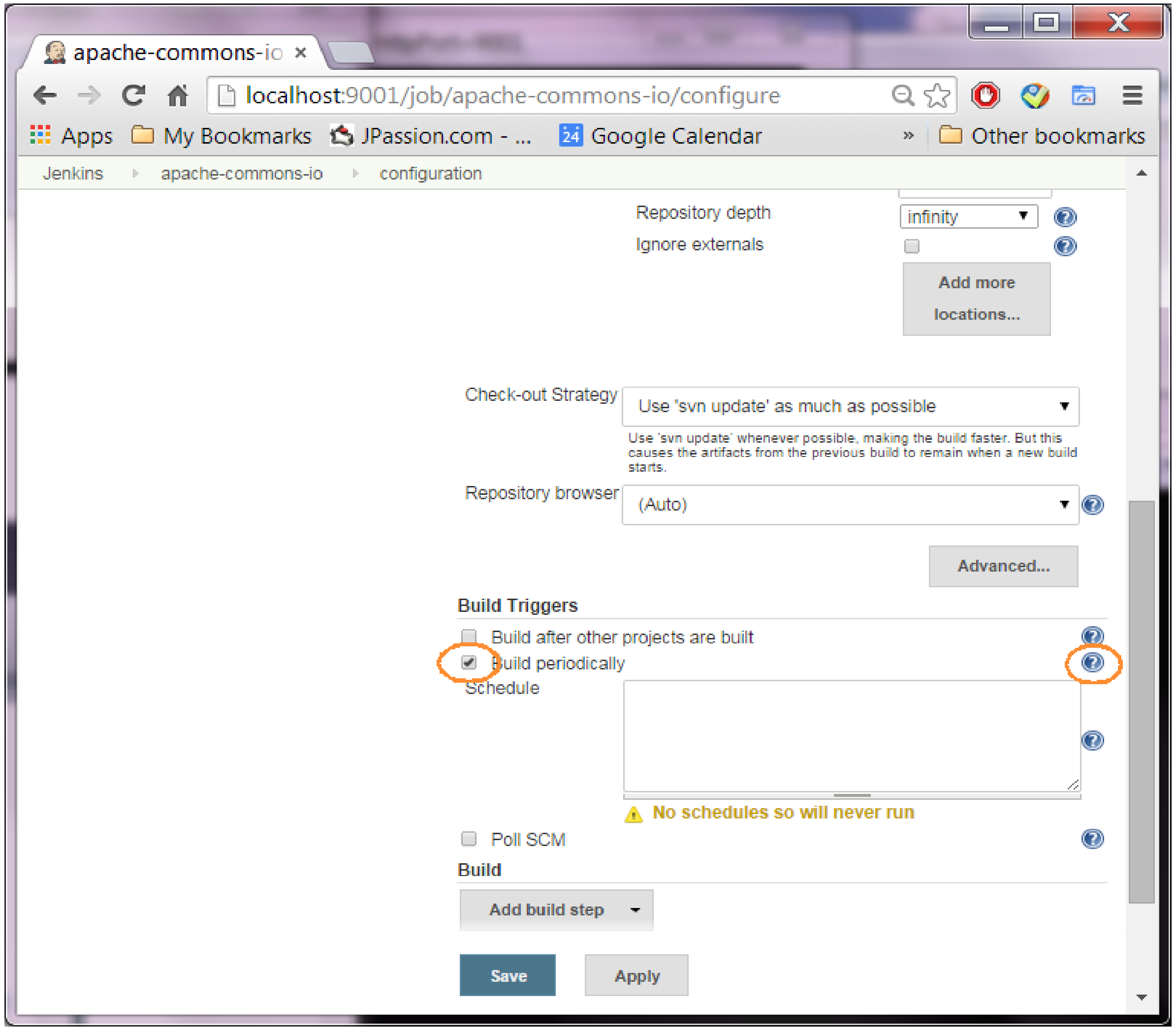
Local Maven Repository: Default (~/.m2/repository)

☑ Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project.

Download Preferences

Save Apply

Configuring Build Trigger



apache-commons-io

localhost:9001/job/apache-commons-io/configure

Jenkins › apache-commons-io › configuration

Repository depth: infinity

Ignore externals: ☐

Check-out Strategy: Use 'svn update' as much as possible

Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser: (Auto)

Build Triggers

☐ Build after other projects are built

☑ Build periodically

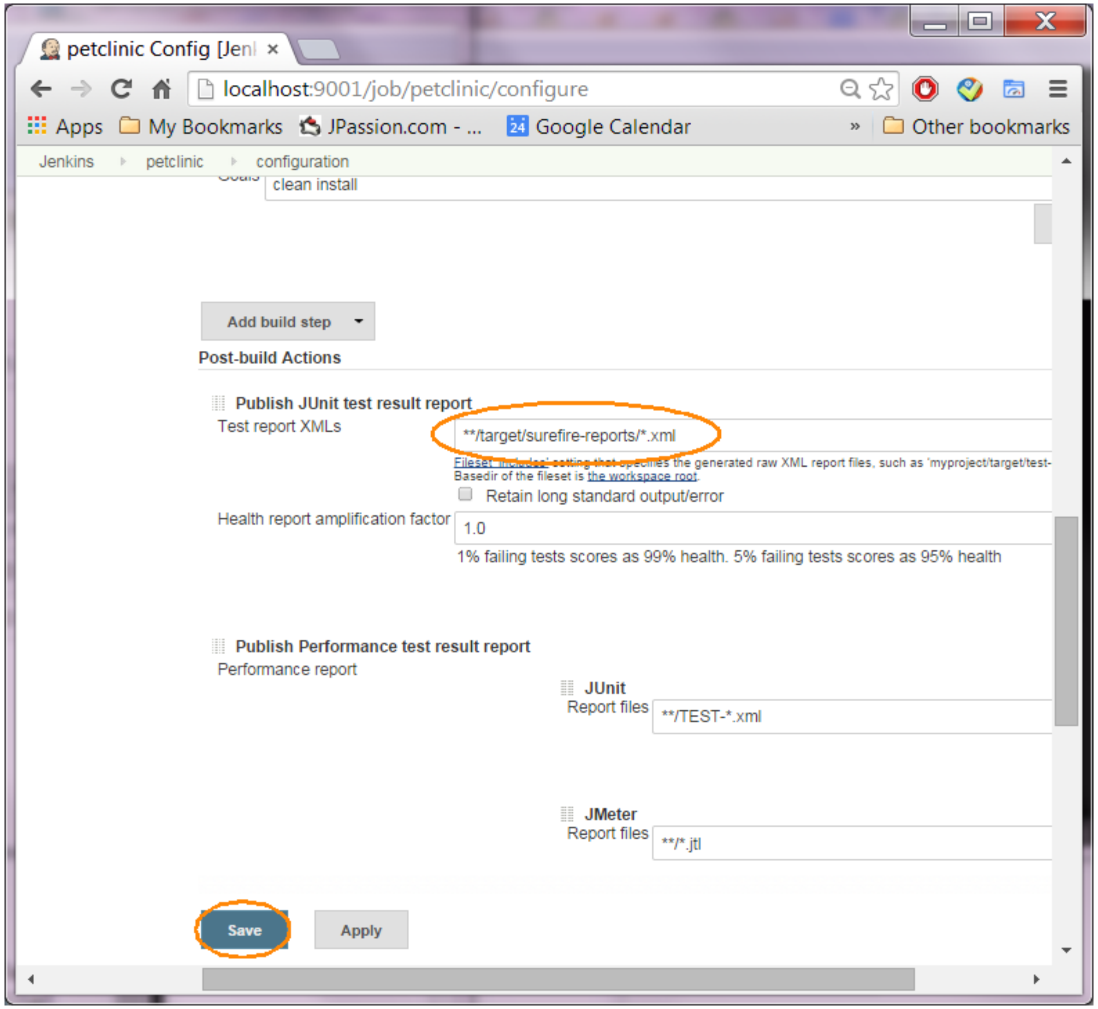
Schedule

No schedules so will never run

☐ Poll SCM

Build

Configuring Testing Report



Goals: clean install

Add build step

Post-build Actions

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-
Basedir of the fileset is the workspace root.

☐ Retain long standard output/error

Health report amplification factor: 1.0

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Publish Performance test result report

Performance report

JUnit
Report files: **/TEST-*.xml

JMeter
Report files: **/*.jtl

Save   Apply

Fingerprints

Fingerprints

- When you have interdependent projects on Jenkins, it often becomes hard to keep track of which version of this is used by which version of that. Jenkins supports "file fingerprinting" to simplify this
- Example scenario
  - Suppose you have the TOP project that depends on the MIDDLE project, which in turn depends on the BOTTOM project
  - You are working on the BOTTOM project.
  - The TOP team reported that bottom.jar that they are using causes an NullPointerException, which you thought you fixed in BOTTOM #32
  - Jenkins can tell you which MIDDLE builds and TOP builds are using (or not using) your bottom.jar #32.

How to set up Fingerprints

- To make this work, all the relevant projects need to be configured to record fingerprints of the jar files (in this case, bottom.jar.)
  - For example, if you just want to track which BOTTOM builds are used by which TOP builds, configure TOP and BOTTOM to record bottom.jar.
  - If you also want to know which MIDDLE builds are using which bottom.jar, also configure MIDDLE
- Since recording fingerprints is a cheap operation, the simplest thing to do is just blindly record all fingerprints of the followings
  - jar files that your project produce
  - jar files that your project rely on

How it works

- The fingerprint of a file is simply a MD5 checksum.
- Jenkins maintains a database of md5sum, and for each md5sum, Jenkins records which builds of which projects used.
- This database is updated every time a build runs and files are fingerprinted
- To avoid the excessive disk usage, Jenkins does not store the actual file. Instead, it just stores md5sum and their usages. These files can be seen in $HUDSON_HOME/fingerprints.

Distributed Building

Distributed Building

- Jenkins supports “Master/Slave” mode for distributed building
  - Master is an installation of Jenkins. It servers all HTTP requests, and it also builds projects on its own
  - Slaves are computers that are set up to build projects for a master. Jenkins runs a separate program called “Slave agent” on Slaves.
  - Master starts slave agents on demand
- Additional workload of building projects are delegated to multiple “Slave” nodes
- Provides different environments needed for builds/tests

Jenkins Best Practices

Jenkins Best Practices

- Always secure Jenkins
- Backup Jenkins Home regularly
- Use “file fingerprinting” to manage dependencies
- The most reliable builds will be clean builds, which are built fully from Source Code Control
- Integrate tightly with your issue tracking system, like JIRA or bugzilla, to reduce the need for maintaining a Change Log

Jenkins Best Practices

- Always configure your job to generate trend reports and automated testing when running a Java build
- Set up Jenkins on the partition that has the most free disk-space

Alternative Solutions

Alternative Solutions

- Open source
  - Hudson – Jenkins is derived from Hudson
  - Apache Continuum – CI server supporting Maven and Ant
  - CruiseControl
  - Tinderbox – from Mozilla
- Commercial
  - Bamboo – from Atlassian
  - TeamCity – from JetBrains
  - Team Foundation Server – from Microsoft
  - Rational Team Concert – from IBM

Learn with Passion!
JPassion.com