



# Seven Habits of Highly Effective Jenkins Users

Andrew Bayer  
Cloudera  
OSCON Java 2011

# Introduction – Who am I?

I'm a build guy, with a focus on Java builds.

I've been using and working on Jenkins since early 2009.

I am currently working at Cloudera.

I am a contributor to Jenkins core, and maintainer of a number of Jenkins plugins, including the git plugin and the Throttle Concurrent Builds plugin.

I'm a member of the Jenkins interim governance board.

@abayer on Twitter

# What is Jenkins?



Jenkins is the leading open source continuous integration server.

It is primarily written in Java, but also used for Ruby, Python, Objective-C, and much, much more.

It's essential to development workflow at such companies as GitHub, Netflix, LinkedIn, and many, many more.

# HABIT #1:

Use plugins productively.



# Habit #1: Use plugins productively

Search for plugins to fit your needs.

Plugins available are listed in the update center and at

<https://wiki.jenkins-ci.org/display/JENKINS/Plugins>

Plugins are organized by category on the wiki and in the update center.

Plugins may have overlapping functionality.

If you're not using a plugin any more, disable it.

Save memory.

Reduce clutter on job configuration pages.

# Habit #1: Use plugins productively, continued

Plugins you should be using:

JobConfigHistory

See the difference between job configurations now and then.

With authentication enabled, see who changed the job and how.

Not quite as nice as true SCM-backed Jenkins configurations, but far, far simpler to put in place.

Seriously, this is my favorite plugin. Go install it!

# Habit #1: Use plugins productively, continued

**Jenkins**

[?](#) **abayer** | [log out](#)

[Jenkins](#) » [jenkins\\_main\\_trunk](#) » [Job Config History](#)

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Job Config History](#)

[Git Polling Log](#)

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

#964

[Jul 20, 2011 11:01:45 AM](#)

63MB

#963

[Jul 19, 2011 6:31:53 PM](#)

63MB

#962

[Jul 19, 2011 11:36:51 AM](#)

63MB

#961

[Jul 17, 2011 5:31:34 PM](#)

63MB

## Job Configuration History

Show Diffs

Date ↓	Operation	User	Show File	Diff	
				File A	File B
2011-07-04_23-18-30	Changed	<a href="#">jieryn</a>	<a href="#">View as XML (RAW)</a>	<input type="radio"/>	<input checked="" type="radio"/>
2011-05-04_20-12-25	Changed	<a href="#">SYSTEM</a>	<a href="#">View as XML (RAW)</a>	<input checked="" type="radio"/>	<input type="radio"/>
2011-05-04_17-59-11	Changed	<a href="#">abayer</a>	<a href="#">View as XML (RAW)</a>	<input type="radio"/>	<input type="radio"/>
2011-05-04_17-59-10	Changed	<a href="#">abayer</a>	<a href="#">View as XML (RAW)</a>	<input type="radio"/>	<input type="radio"/>
2011-03-31_21-03-15	Changed	<a href="#">kohsuke</a>	<a href="#">View as XML (RAW)</a>	<input type="radio"/>	<input type="radio"/>
2011-03-31_21-03-14	Changed	<a href="#">kohsuke</a>	<a href="#">View as XML (RAW)</a>	<input type="radio"/>	<input type="radio"/>
2011-03-19_09-39-49	Changed	<a href="#">olamy</a>	<a href="#">View as XML (RAW)</a>	<input type="radio"/>	<input type="radio"/>
2011-03-19_09-08-38	Changed	<a href="#">olamy</a>	<a href="#">View as XML (RAW)</a>	<input type="radio"/>	<input type="radio"/>
2011-03-19_09-07-10	Changed	<a href="#">olamv</a>	<a href="#">View as XML (RAW)</a>	<input type="radio"/>	<input type="radio"/>

# Habit #1: Use plugins productively, continued

Plugins you should be using, continued:

## Disk Usage

How much space is that job taking up on the slave? How large are the archived artifacts for that build? Find the answers easily.

## Build Timeout

Hanging tests a problem? Set a timeout – the build will be aborted if it takes longer.



# Habit #1: Use plugins productively, continued

## Jenkins

search

abayer | log out

Jenkins » jenkins\_main\_trunk

ENABLE AUTO REFRESH

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

Job Config History

Git Polling Log

### Project jenkins\_main\_trunk

add description

Disable Project

Disk Usage: Workspace 2GB, Builds 4GB

#### Disk Usage Trend

disk usage (GB)

— build

— workspace

#### Compiler Warnings Trend

Build History

(trend)

#964	<a href="#">Jul 20, 2011 11:01:45 AM</a>	63MB
#963	<a href="#">Jul 19, 2011 6:31:53 PM</a>	63MB
#962	<a href="#">Jul 19, 2011 11:36:51 AM</a>	63MB
#961	<a href="#">Jul 17, 2011 5:31:34 PM</a>	63MB

# Habit #1: Use plugins productively, continued

Plugins you should be using, continued:

## Email-ext

Better control of when emails get sent and who they get sent to.

Format the emails the way you want, using information from your builds.

## Parameterized Trigger

Kick off downstream builds with information from upstream – i.e., tell a system test where to find the packages it needs to install, etc.

# Habit #1: Use plugins productively, continued



☐ Email Notification [?](#)

☒ Editable Email Notification [?](#)

Global Recipient List  [?](#)  
Comma-separated list of email address that should receive notifications.

Content Type  [?](#)

Default Subject  [?](#)

Default Content  [?](#)

Content Token Reference [?](#)

Trigger	Send To Recipient List	Send To Committers	Send To Requester	Include Culprits	More Configuration	Remove
Failure <a href="#">?</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">+ (expand)</a>	<a href="#">Delete</a>
Fixed <a href="#">?</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">+ (expand)</a>	<a href="#">Delete</a>
Still Failing <a href="#">?</a>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">+ (expand)</a>	<a href="#">Delete</a>

Add a Trigger:  [?](#)

☒ IRC Notification [?](#)

# HABIT #2:

Standardize your slaves.

# Habit #2: Standardize your slaves

If you've got more than a couple builds, you've probably got multiple slaves.

Ad hoc slaves may be convenient, but if your slaves have different environments, you're in for trouble eventually.

If you have multiple slaves playing the same role, they should be as identical as you can make them.



# Habit #2: Standardize your slaves, continued

Puppet, Chef, etc

Sysadmins are used to these tools already.

Configuring your build slaves is the same as configuring any other puppet/chef/etc-managed machine.

Have Jenkins or your job install your tools

Jenkins tool installation for Maven, Ant, JDK, etc.

Scripting downloading/installation of tools in the job itself.

May not be possible for all tools – e.g., RPM/Debian packages.

# Habit #2: Standardize your slaves, continued

Cloud or VMs

Spin up your slaves from a pre-configured image.

When you need to change your slave configurations, update the image and spin up new slaves, rather than updating the existing slaves.

Whatever method you choose, just make sure your slaves are consistent!

# Habit #2: Standardize your slaves, continued

Side note!

Don't build on master. Always build on slaves.

No conflict on memory/CPU/IO between master process and builds.

Easier to add slaves than to beef up master.

Mixing builds on master with builds on slaves means inconsistencies between builds – as before, consistency is always good.

# HABIT #3:

Use incremental builds.

# Habit #3: Use incremental builds

...if possible.

Why use incremental builds?

If your build takes 4-8 hours to run end-to-end, you can't do real continuous integration on every change.

If you're integrating with code review or otherwise using pre-tested commit processes, you want to verify changes as fast as possible.



# Habit #3: Use incremental builds, continued

Incremental builds are complementary to full builds, not replacements.

Don't do QA/deployments of incremental builds – use full, clean builds for that.

e.g., incremental build whenever a change is detected, full build once a day.

Either use multiple jobs (one for incremental, one for full) or configure one job to behave differently, depending on parameters.

# Habit #3: Use incremental builds, continued

Maven project type has built-in support for incremental builds.

Look in Advanced Options in the Maven configuration section.

When enabled, checks the changelist for the build and compares to Maven module layout.

Any modules containing changed files are built.

As are modules depending on changed modules, and any modules with failures in the previous build.

# Habit #3: Use incremental builds, continued

Incremental builds may not be possible in all cases.

Freestyle projects would require custom work to determine what needs to be run for incremental builds.

e.g., use code coverage data to determine what tests need to be run when a given source file is changed.

# HABIT #4:

Integrate with other tools.

# Habit #4: Integrate with other tools

Jenkins plugins are available for integrating with any number of other tools. For example...

Pre-tested commits with Gerrit (git code review)

When a change gets submitted, a build runs automatically and reports back to Gerrit.

Code can't be merged if the build doesn't pass.

Pre-tested commits have been implemented elsewhere – see Apache Hadoop build process, e.g.



# Habit #4: Integrate with other tools, continued

JIRA and other bug trackers

Update bug tracker when a fix gets built, automatically.

Sonar

Code metrics, code coverage, unit test results, etc, all in one place.

Great graphs, charts, etc – fantastic manager candy!

# Habit #4: Integrate with other tools, continued

## Artifactory

Control staging of Maven/Gradle/Ivy release candidates.

Report on dependencies.

See

<http://blogs.jfrog.org/2011/06/jenkins-new-maven-and-gradle-release.html>  
for more details.

## Mylyn

See build and test results, launch new builds and more, all from within Eclipse.

# Habit #4: Integrate with other tools, continued

## Chat/IM notifications

Most organizations have some chat/IM solution in place.

Jenkins can integrate with many of them – Jabber, IRC, Campfire, etc.

Notify committers when builds with their changes break!

# HABIT #5:

Break up bloat.

# Habit #5: Break up bloat

Too many builds make it hard to navigate the Jenkins UI and hurt performance.

Builds that try to do too much take too long and make it impossible to restart a build process partway through.

Two similar problems, with similar solutions.

Break up the bloat!



# Habit #5: Break up bloat, continued

Don't be afraid to spread your jobs across multiple Jenkins masters.

Easier to navigate each instance than the one big one.

Better performance with less jobs per instance.

Restarting Jenkins can be done much more readily when there are less jobs on the instance.

Split jobs in a logical way – separate instances per group, per product, per physical location, etc.

# Habit #5: Break up bloat, continued

Big, long builds are unwieldy.

Split builds up logically as well, into coherent phases.

Pass data/binaries from one phase to the next (using the Parameterized Trigger plugin!)

When the last of 6 phases fails due to network problems or the like, just re-run that phase without having to run the whole pipeline again. Much faster!

# HABIT #6:

Stick with stable releases.

# Habit #6: Stick with stable releases

Jenkins releases weekly.

This means rapid turnaround on fixes, new features, etc, but, inherently, not 100% stability for every release.

Plugins release whenever their developers choose to.

Update center makes it easy to upgrade core and plugins to the latest at any time, but that's not always the smart move.

# Habit #6: Stick with stable releases, continued

Use Jenkins core LTS releases.

LTS – Long Term Support, similar to Ubuntu.

Every ~3 months, a particularly stable release is chosen, a branch is created from it and the new branch is rigorously tested.

Backports of critical fixes are made to this branch and new releases (e.g., 1.409.1, 1.409.2, etc) are made.

# Habit #6: Stick with stable releases, continued

Use Jenkins core LTS releases, continued.

Plugin developers are encouraged to build against the latest LTS release, helping keep new plugin releases compatible.

If you don't have a specific reason to be on the bleeding edge, you should probably be running an LTS release.

See <https://wiki.jenkins-ci.org/display/JENKINS/LTS+Release+Line> for more information.



# Habit #6: Stick with stable releases, continued

Upgrade plugins carefully.

Check plugin wiki pages for changelog.

In the update center, click the link for the plugin name – that will take you to the wiki page.

If you don't need a bugfix or new feature, you probably don't need to upgrade.

Stick with what works until it doesn't work any longer.

# HABIT #7:

Join the community.

# Habit #7: Join the community

Write a plugin or work on an existing one.

Improve docs and the wiki.

Get on the mailing list and help out.

Join the IRC channel.

If you run into a bug, submit it at <https://issues.jenkins-ci.org!>

# Conclusion

These are just one person's thoughts and experiences.

There are many ways to use Jenkins – there is no single “Right Way” to do things.

Your organization's requirements should dictate your build process and tool usage, within reason.

# Questions?