Seven Habits of Highly Effective Jenkins Users

Andrew Bayer
Build and Tools Architect, Cloudera
Twitter: @abayer

June 25, 2014
Who Am I?

- Build and tools architect at Cloudera.
- Contributor to Jenkins core and author of plugins since spring 2009.
- Committer and PMC member of multiple Apache projects, inc. jclouds, Whirr, Bigtop…
- ASF Member and volunteer for maintaining builds.apache.org.
What’s this talk about?

These are lessons learned from maintaining multiple large Jenkins instances over the years.

Cloudera’s three masters with 1000+ jobs each with dozens running at a time.

builds.apache.org’s 1000+ jobs from ~100 different project teams.

Oh, and my time on IRC, working on core and plugins, etc.
Your mileage may vary

- These habits can be valuable on every Jenkins instance.
- Some will be more relevant for larger instances, those with more complex jobs, and/or production-critical instances.
- But these are *my* recommendations - you need to learn what’s best for your Jenkins setup.
HABIT 1: MAKE YOUR MASTER STABLE AND RESTORABLE
Use LTS Releases

- LTS release trains created every 12 weeks.
- Active train updated three times before the next one starts.
- Avoid bleeding edge instability.
- LTS releases go through automated acceptance testing and a manual testing matrix before going out.
Be conservative about upgrading plugins

- Plugins can change a lot without it being obvious.
- Backwards compatibility can sometimes break.
  - Example - Extended Email plugin recipient/trigger settings recently.
- New features can be unstable/problematic in the wild.
Have an upgrade testbed

- Test out upgrades and new plugins in a testbed environment before going live in production.
- Set up jobs to cover your plugin usage.
- If possible, test your usage at scale.
- Give significant changes a few days, at least, of builds running before going live.
Back up your Jenkins configuration

- Kind of obvious, isn’t it? =)
- Lots of possible solutions
  - Within Jenkins, I recommend the thinBackup plugin.
  - Full copies of $JENKINS_HOME work great, but can be slow and use lots of disk.
  - Config files can be copied without copying all the builds, etc as well - see an example here
Avoid using the Maven job type

• Maven build steps are perfectly fine in freestyle jobs, but the Maven plugin’s Maven job type is...questionable.

• Implementation leads to some potential problems in plugin support, lazy loading of projects, interception of Maven execution internals, etc...

• I’ve seen a lot of strange edge case problems show up with it at scale. Be careful with it.
HABIT 2: BREAK UP THE BLOAT
Multiple Masters

• If you have a lot of projects and teams, multiple masters give you a lot more agility and control.
• Split up masters by team, function, access, etc.
• Makes it easier to restart for plugin installs/upgrades, etc without disrupting everyone.
• More masters with less jobs each are more stable, less prone to edge case bugs.
Break up your jobs

• Modularization and reuse are good in programming - and good in Jenkins too.
• Multi-job builds allow reusability of generic jobs across multiple projects, releases, etc.
• Few things more frustrating than a 10 hour build crashing 9.5 hours in.
  • Multi-job builds can be relaunched at any step in the process, if designed properly.
Tools for breaking up your jobs

• Just some examples - there are a ton of ways you are able to do this in Jenkins.

• Parameterized Trigger + Conditional Build Step, Copy Artifact, Promoted Builds:
  • Very powerful, not as easy to configure.

• Build Pipeline plugin:
  • Visualize your workflow, integrate manual steps.

• Workflow plugin:
  • Define the relationship between your steps in a DSL.
HABIT 3: AUTOMATE JENKINS TASKS!
The script console and Scriptler

• Why do things by hand?
• Get deep into Jenkins itself - control the internals and get full visibility of what’s happening.
• Access the entire Jenkins model - make changes to jobs, find problem configurations and more.
• Use Scriptler to store and share Groovy scripts for reuse.
Some examples from the Scriptler catalogs

• Disable/enable jobs matching a pattern
• Clear the build queue
• Set log rotation/discard old builds configuration across all jobs
• Disable SCM polling at night across all jobs
• Run the log rotator (actually discard old builds) for all jobs
• etc…
System Groovy build steps

• Run system Groovy scripts as part of your actual build.

• Note - gives full access to Jenkins to the build, so be careful about who can run it and what it does.

• Good way to pilot plugin concepts or do things not big enough to be worth a plugin on their own.

• Run Scriptler scripts as build steps - reuse system scripts in multiple builds easily.
Generate jobs programmatically

- Jenkins REST API and CLI let you post new jobs and changes to jobs.
- Or you can define your whole job and/or workflow of multiple jobs in a DSL.
Some DSL-like plugins

• Job DSL plugin
  • Full Groovy DSL for job definitions - check in your DSL and create your jobs as a build step.

• DotCI plugin
  • Define your jobs in YAML and check them in - jobs created automatically.

• The upcoming Literate plugin
  • Markdown-like syntax for defining your job in your project source.
HABIT 4: TEND YOUR PLUGIN GARDEN
Do you really need that plugin?

- Don’t install plugins on the master if you aren’t going to actually use them.
- Lots of duplication of functionality across plugins - pick the right one for the job.
- Plugins can cause instability in areas you don’t expect, and can add to load and run time for jobs - why take a hit from plugins you don’t use?
Clean up old plugins and their data

- Uninstall unused/unneeded plugins.
- In Manage Jenkins, watch for the note about old data - clear it out when you uninstall plugins, to slim down your job and build config files.
- Speeds up loading of the master and individual jobs.
My essential plugins

• Job Config History
• Disk Usage
  • Not any more - newer versions don’t scale well at all!
• Static analysis plugins
• xUnit
  • Translates lots of test output into junit format for Jenkins to consume.
• Parameterized Trigger and Conditional Build Step
  • My Swiss Army Knife for build workflows!
My essential plugins

- **Tool Environment**
  - Use Jenkins’ tool auto installation from shell steps.
- **EnvInject**
  - Seems to be the best option for setting env vars for your build in various ways.
- **Rebuild**
  - Re-run parameterized builds easily.
- **Build Timeout**
  - Builds hang. This plugin deals with hung builds.
Don’t take my word for it

• These are *my* essential plugins, from my experience and for my use cases.
• You may not need these plugins, you may need other plugins completely.
• But these are plugins I think have a lot of versatility and value, and little risk.
Remember the global configuration

• Some plugins have global configuration settings you should remember to look at.
• The defaults may not always work for you - and sometimes the defaults aren’t great choices.
• Job Config History, for example
  • By default, saves “changes” for every Maven module separately! Ouch!
HABIT 5: INTEGRATE WITH OTHER TOOLS AND SERVICES
Jenkins plays nicely with others

- Thanks to Jenkins’ plugins and REST API, services and tools can easily interact with Jenkins and vice versa.
- Trigger builds based on GitHub pull requests, update JIRA upon successful builds and much, much more.
- I’ll only touch on a few such tools and services - you can find many more on the Jenkins wiki.
Source Control!

• ...Well, yeah.

• Moving on...
Gerrit and GitHub pull requests

- Gerrit Trigger, GitHub Pull Request Builder, Jenkins Enterprise’s version of GitHub pull request builder - all very useful.
- Build every proposed change, report back to the review tool.
- With this, you can enable automatic merging of changes, promotion from branch to branch, and much more.
JIRA

- Update JIRA issues when commits with messages containing the issues are built.
- Follow build fingerprints to update issues in related projects as well.
- Generate JIRA release notes as part of the build process.
Artifactory

• Define credentials for deployment and artifact resolution globally across Jenkins jobs.
• Override Maven distributionManagement on a per-job basis.
• Restrict where Maven jobs and build steps will look to resolve artifacts.
• Capture build info and relationship to artifacts in Artifactory.
HABIT 6: MAKE YOUR SLAVES FUNGIBLE
Fungible? What does that mean?

- “Fungibility is the property of a good or a commodity whose individual units are capable of mutual substitution.”
- A fungible slave is a slave you can replace easily with another slave.
- If one dies or is busy, no problem - just add another one.
- The easier it is to add slaves, the easier your life is.
How do you make your slaves fungible?

• Make creating the environments easily repeatable.
  • Config management - Puppet, Chef, Ansible, etc.
  • Pre-baked images - cloud, PXE, etc, using something like Packer and config management to build them.
• I have no opinion on config management tools - to be honest, it doesn’t really matter. Anything that can set up your environment consistently is good enough!
Reusability and flexibility

• Try to make your slaves general purpose
• Don’t make them customized solely for use by one job or set of jobs if you can avoid it.
  • Interchangeable slaves allow for more efficient usage.
• If you need specific custom slaves, make them on-demand.
  • Don’t tie up static resources with slaves that won’t be used all the time.
To the cloud!

- More efficient usage of your resources.
- Private cloud or public cloud - the goal is to avoid idle resources that can’t be used for anything else.
- Mesos plugin with Docker is very intriguing - run your slaves as containers in a general purpose cluster!
- Pre-bake your cloud slave images - faster startup time, more consistency.
HABIT 7: JOIN THE COMMUNITY
Get involved!

• Write plugins.
• Open JIRAs.
• Fix bugs.
• Get help on the mailing lists or IRC.
• Help others too!
QUESTIONS?
Helpful Links by Habit

• Habit 1: Make your master stable and reliable
  • [https://wiki.jenkins-ci.org/display/JENKINS/LTS+Release+Line](https://wiki.jenkins-ci.org/display/JENKINS/LTS+Release+Line)
  • [https://wiki.jenkins-ci.org/display/JENKINS/thinBackup](https://wiki.jenkins-ci.org/display/JENKINS/thinBackup)
  • Backup rsync example: [https://gist.github.com/abayer/527063a4519f205efc74](https://gist.github.com/abayer/527063a4519f205efc74)

• Habit 2: Break up the bloat
  • [https://wiki.jenkins-ci.org/display/JENKINS/Parameterized+Trigger+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Parameterized+Trigger+Plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/Conditional+BuildStep+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Conditional+BuildStep+Plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/Build+Pipeline+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Build+Pipeline+Plugin)
  • [https://github.com/jenkinsci/workflow-plugin](https://github.com/jenkinsci/workflow-plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/Copy+Artifact+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Copy+Artifact+Plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/Promoted+Builds+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Promoted+Builds+Plugin)
Helpful Links by Habit

- Habit 3: Automate Jenkins tasks
  - [https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+Script+Console](https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+Script+Console)
  - [https://wiki.jenkins-ci.org/display/JENKINS/Scriptler+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Scriptler+Plugin)
  - [https://wiki.jenkins-ci.org/display/JENKINS/Groovy+plugin](https://wiki.jenkins-ci.org/display/JENKINS/Groovy+plugin)
  - [https://wiki.jenkins-ci.org/display/JENKINS/Remote+access+API](https://wiki.jenkins-ci.org/display/JENKINS/Remote+access+API)
  - [https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+CLI](https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+CLI)
  - [https://wiki.jenkins-ci.org/display/JENKINS/Job+DSL+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Job+DSL+Plugin)
  - [https://github.com/groupon/DotCi](https://github.com/groupon/DotCi)
  - [https://wiki.jenkins-ci.org/display/JENKINS/Literate+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Literate+Plugin)
Helpful Links by Habit

• Habit 4: Tend your plugin garden
  • [https://wiki.jenkins-ci.org/display/JENKINS/JobConfigHistory+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/JobConfigHistory+Plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/Tool+Environment+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Tool+Environment+Plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/EnvInject+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/EnvInject+Plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/Rebuild+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Rebuild+Plugin)
  • [https://wiki.jenkins-ci.org/display/JENKINS/Build-timeout+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Build-timeout+Plugin)
Helpful Links by Habit

• Habit 5: Integrate with other tools and services
  • https://wiki.jenkins-ci.org/display/JENKINS/Gerrit+Trigger
  • https://wiki.jenkins-ci.org/display/JENKINS/GitHub+pull+request+builder+plugin
  • http://wiki.cloudbees.com/bin/view/DEV/Github+Pull+Request+Validation
  • https://wiki.jenkins-ci.org/display/JENKINS/JIRA+Plugin
  • https://wiki.jenkins-ci.org/display/JENKINS/Artifactory+Plugin
  • https://wiki.jenkins-ci.org/display/JENKINS/HipChat+Plugin
  • https://wiki.jenkins-ci.org/display/JENKINS/Instant+Messaging+Plugin
Helpful Links by Habit

- Habit 6: Make your slaves fungible
  - [https://wiki.jenkins-ci.org/display/JENKINS/Amazon+EC2+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Amazon+EC2+Plugin)
  - [https://wiki.jenkins-ci.org/display/JENKINS/JClouds+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/JClouds+Plugin)
  - [https://wiki.jenkins-ci.org/display/JENKINS/Mesos+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Mesos+Plugin)
Helpful Links by Habit

• Habit 7: Join the community
  • http://jenkins-ci.org/
  • https://issues.jenkins-ci.org/
  • https://wiki.jenkins-ci.org/display/JENKINS/Home
  • http://jenkins-ci.org/content/mailing-lists
  • http://jenkins-ci.org/content/chat
Thank you for attending!