CD Summit & Jenkins Days

CD Summit
& Jenkins Days

Seven Habits of Highly Effective Jenkins Users

What is this talk about?

- Lessons learned:
  - Maintaining multiple large Jenkins instances.
  - Working on Jenkins itself, and many of its plugins.
  - Seeing customer usage of Jenkins in the "real world."

Your mileage may vary

- These habits can be valuable on every Jenkins instance.
- Some will be more relevant to larger instances.

These are **my** recommendations, you may need to experiment and learn what is best for *your* Jenkins installation(s).

Habit #1:
Make your Jenkins master stable *and* restorable

Use LTS Releases

- LTS release lines are created every 12 weeks.
- Each LTS line is updated (roughly) three times.
- Avoids bleeding-edge instability.
- LTS releases go through automated acceptance testing *and* a manual testing matrix.

Jenkins 2 LTS

- Jenkins 2 has been LTS for a few months (2.7.x)
- Currently on 2.19.x
- Now is a great time to upgrade!

Have a Jenkins staging environment

- Test out upgrades and new plugins in the testbed/staging environment.
- Setup example jobs to cover your plugin usage.
- If possible, test with heavier loads.
- Give significant changes a few days of builds before deploying to production.

Conservatively upgrade plugins

- Always read the changelog!
- Plugins can change a lot between releases.
- Backwards compatibility can and will sometimes break
  - Extended EMail plugin recipient/trigger settings in 2014
  - Green Balls plugin
- New features can be unstable/problematic

Backup your configuration

- Easy to do, many possible solutions.
- Within Jenkins:
  - plugins such as thinBackup
- On the system:
  - Full copies of `JENKINS_HOME`
  - `config.xml` files can be copied without all the build data
- CloudBees has commercial solutions too.

Avoid the "Maven" job type

- Maven build steps are perfectly fine in freestyle and Pipeline jobs.
- Maven job type's implementation can lead to problems
  - Compatibility with other plugins.
  - Lazy-loading of projects and other performance side-effects.
  - Intercepts Maven execution internals, etc.
- Many strange edge-cases at scale. Be Careful™

Habit #2:
Break up the bloat

Multiple masters

- With many projects and teams, multiple masters allow more agility and control.
- Split across team, function, access, etc.
- Isolates impact of upgrades/plugin installations.
- More masters with fewer jobs tend to be more stable

Break up or transform your jobs

- Modularization and reuse are good in Jenkins too.
- Multi-job builds allow generic jobs across multiple projects, releases, etc.
- Jenkins Pipeline doesn't break up your jobs but makes larger jobs more understandable and durable.

Tools for breaking up jobs

- A few (of many) examples:
  - Parameterized Trigger + Conditional Build Step
  - Copy Artifact
  - Promoted Builds
  - ..
- Very powerful tools, not very easy to configure.

Transform Jobs to Pipelines

- Jenkins Pipeline
  - Define your job(s) in code
  - Check the `Jenkinsfile` into the repository
  - One Pipeline can do far more than one job
    - run steps on different nodes
    - checkout multiple repositories
    - *etc*

Habit #3:
Jenkins tasks as code

Script Console and Scriptler

- Why do things by hand?
- Control Jenkins internals and get full visibility into what is happening
- Access the entire Jenkins data model
- Use Scriptler to store and share Groovy scripts for reuse.
  - A public collection can be found:

    github.com/jenkinsci/jenkins-scripts

Scriptler examples

- Disable/enable jobs matching a specified pattern.
- Clear the build queue.
- Set log rotation/discard old builds across all jobs.
- Disable SCM polling at night across all jobs.
- Run the log rotator, discarding old builds.
- More at github.com/jenkinsci/jenkins-scripts

System Groovy build steps

- Run system Groovy scripts as part of the build.
- **Caution:** this gives the build full access to Jenkins.
- Good way to test plugin concepts or tasks not "big" enough to warrant a plugin.
- Run Scriptler scripts as build steps, reusing system scripts in multiple builds

Generate jobs from code

- REST API and CLI let you POST new jobs and changes to existing jobs
  - localhost:8080/api
  - localhost:8080/cli
- Use plugins which allow defining job configuration as code

Some DSL-ish tools

- Job DSL plugin
  - Groovy DSL for job definitions
  - Check definitions in and let a "seed" job create your jobs
- Jenkins Job Builder
  - Translates simple YAML into freestyle jobs

Jenkins Pipeline

- Define multiple stages of a delivery pipeline in one relatively simple DSL.
- Check the pipeline into the repository as a `Jenkinsfile`.
- Auto-discovery and creation of items in Jenkins with Multibranch, GitHub Organization Folder, and BitBucket Team/Project Folder plugins.

Jenkins Pipeline

- Durability: steps executing on agents survive master restarts
- Visualization: see how changes progress through the pipeline with details on various stages.
- Reuse steps via the Shared Libraries feature
- Blue Ocean, a new user experience on top of Pipeline

Habit #4:
Tend your plugin garden

Do you really need that plugin?

- Don't install plugins if you're not certain you're going to use them.
- Lots of duplication of functionality between plugins, pick the right one for your use-case.
- Plugins can affect areas of Jenkins you don't expect, such as load and run time for jobs.

Clean out old plugins and their data

- Periodically uninstall unused/unneeded plugins
- Under the "Manage Jenkins" view, watch for notices about "old data" and remove as necessary.
- Pruning unnecessary data will improve over all master and job load times.

Essential plugins

- **Job Config History**
- Static analysis plugins
- **xUnit**: translates various test output formats for Jenkins
- **Pipeline Stage View**: visualize the whole pipeline
- **Timestamper**: annotates console output with timestamps
- **Rebuild**: easily re-run parameterized builds
- **Build Timeout**: prevent hung builds from consuming resources

Inessential plugins

- ~~**Disk Usage**~~
  - Causes unnecessary disk I/O as you scale!
- ~~**Parameterized Trigger** & **Conditional Build Step**~~
  - Formerly a good swiss army knife for build workflows, superseded by Jenkins Pipeline

Inessential plugins

- Pipeline makes many plugins redundant
  - EnvInject
  - Rebuild
  - Build Timeout
  - Conditional Build Step
- Not all plugins intended for Freestyle jobs make sense for Pipeline

Remember global configuration

- Some plugins have global settings in "Configure Jenkins" you should remember to check.
- The defaults might not work for you.
  - Job Config History
    - by default saves "changes" for every Maven module *separately*! Ouch!

Your mileage may vary

- These are **my** essential plugins from experience with **my** use-cases.
- These plugins have lots of versatility and value, with little risk.

Habit #5:
Integrate with other tools and services

Jenkins plays nicely with others

- Via plugins and the REST API, services and tools can readily interact with Jenkins (and vice versa).
  - Trigger builds based on GitHub pull requests
  - Update JIRA tickets upon successful builds
  - etc
- Here's a few, but find more on the Jenkins wiki

Source Control

- Practically everything from Git. to ClearCase, to CVS.
- Multibranch Pipeline
  - Point Jenkins at the repository and jobs are automatically created for every branch and pull request which contains a `Jenkinsfile`.
- Organization Folders
  - Next level up, point Jenkins at a GitHub Organization or BitBucket Team and create Multibranch Pipeline projects for each repository!

Code Review

- Gerrit Trigger for integration with Gerrit.
- GitHub Pull Request Builder, Pipeline Multibranch for integration with GitHub.
- Build every proposed change, report back to the review tool.
- Enables automatic merging of changes, promotion from branch to branch, and more.

JIRA

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

Artifactory

- Define credentials for deployment/artifact resolution globally across Jenkins jobs
- Override Maven distributionManagement on a per-job basis.
- Restrict where Maven jobs and build steps will look when resolving artifacts.
- Capture build information and relationship to artifacts in Artifactory.

Docker

- Smoothly integrated into Jenkins Pipeline
- Additional plugins for:
  - building Docker images
  - running builds in containers
  - tracing containers between jobs
  - integrating with container provides (e.g. Kubernetes)

Habit #6:
Make your agents replaceable

Replaceable Agents

- Agents should be easily and readily replaced
  - if one fails or is busy, add more.
- The easier it is to add new agents, the easier managing heaps of agents can be.

Making agents replaceable

- Use off-the-shelf tooling for making repeatable environments
  - Configuration Management: Puppet, Chef, etc
  - Pre-baking images: Docker, Packer, etc.

Agent reuse and flexibility

- Try to make agents general purpose.
- Avoid customizing an agent(s) for one job or set of jobs
  - Interchangeable agents allow for more efficient use of Jenkins build resources.

Agent reuse and flexibility

- If you **must** have customized agents, try to make them on-demand

Avoid tying up static resources (compute, etc) with agents which won't be in-use very often.

To the cloud!

- Using plugins to provision agents as cloud resources leads to more efficient usage patterns.
- Cloud providers make it easier to pre-bake agent images leading to faster startup time and more consistency.
- OpenStack, AWS, Azure, Docker Swarm, Kubernetes, Mesos, whatever.

The goal is to avoid idle resources.

Habit #7:
Join the Jenkins community

Get involved!

- https://jenkins.io/
- Participate in the mailing lists
  - jenkinsci-users@googlegroups.com
  - jenkinsci-dev@googlegroups.com
- IRC: #jenkins
- Join Meetups:
  - Jenkins Online Meetup (https://www.meetup.com/Jenkins-online-meetup/)
  - Local meetup: Search for Jenkins on Meetup.com

Get more involved!

- File bugs/feature requests in JIRA
- Fix bugs/features
- Extend existing plugins
  - or create new ones where necessary

Questions?

CD Summit & Jenkins Days