Uncle John's Bathroom Reader

Testing Every Commit 1

by Leeroy Jenkins

Writing code at Lookout is only part of the software development story. After code has been written, it starts a long (and mostly automated) journey to "production." For this issue of **Uncle John's Bathroom Reader**, we're not going to focus on code, but rather how code is tested with **Jenkins** and **Gerrit**.

The projects which use Gerrit, follow a "pre-tested commit" workflow. Meaning each commit must be pushed to Gerrit for review and testing prior to being merged into the destination branch. Testing of these projects is generally accomplished by two "**jobs**" in Jenkins. For example, a project named "cryptoserv" would have the jobs:

- **`gerrit_cryptoserv`:** Will only run on pending (In Review) changes which have been submitted by developers.
- **`cryptoserv`:** Will run when changes are merged into the destination branch (typically **`master`**), depending on the project this job may also start an automated deployment process through "Job Promotion" in Jenkins.

The **`gerrit_cryptoserv`** job uses a Jenkins plugin to listen for changes as they're pushed by developers into Gerrit. When changes are pushed, Jenkins receives a notification along with the SHA-1 of the change (and other meta-data) allowing the job to check out that specific commit and run the pre-defined test suite. If that test suite passes, Jenkins will tell Gerrit the change is **+1, Verified**; otherwise the change will receive **-1, Not Verified**.

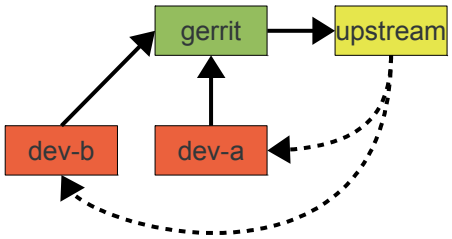
Gerrit will **not** allow changes to be merged unless they receive a **+1, Verified** in conjunction with a **+2, Code Review** flag.

At its core, Jenkins is a straight-forward queuing and script-running engine but it can be used for much more in terms of reporting and analysis.

Across Lookout it is used as an automation hub for various needs, outside of the Gerrit pre-tested workflow described above. If you spend some time digging through our Jenkins instance you will find jobs for all variations of testing, various shapes of client builds, production and staging deployment automation, package generation, and report creation. In future issues we'll dig deeper into the various forms of automation used at Lookout.

That's it for this issue of **Uncle John's Bathroom Reader**, now go wash your hands.

Where the commits go:



For projects which use Gerrit, changes can only be submitted via review and testing in Gerrit. After they've been approved and merged, they will be mirrored to GitHub, where the rest of the team will then fetch from.

Publishing a Draft:

Gerrit supports the concept of a Draft, which is a private version of a change. Drafts will not be touched by Jenkins and nobody can see them unless you explicitly add them as a reviewer. To push commits as a draft use:

```
% git push gerrit \
        HEAD:refs/drafts/master
```

When you want to turn the changes from drafts into testable changes, run the usual command to push to Gerrit.

```
% git push gerrit \
        HEAD:refs/for/master
```

Clarification:

In our last issue (#04), the sidebar mentioned a caveat with "Gemfile Groups" which was poorly worded. When using groups and the **`--without`** option, Bundler will not *install* the gems in that group, but it must be able to *find* them in the gem sources or the gem's specified location.

Docs on the various use cases of Jenkins: https://wiki.jenkins-ci.org/display/JENKINS/Use+Jenkins

Gerrit project home page: https://code.google.com/p/gerrit/

Send comments and suggestions for future issues to: **ujbr@lookout.com**