Hg & Git for same codebase Documentation

Release 0.1

Hal Wine

May 21, 2013

CONTENTS

Speaker Note

Following are the slides I presented at the RELENG 2013 workshop on May 20th, 2013. Paragraphs like this were not part of the presented slides - rather speaker notes.

Hal Wine *hwine@mozilla.com*
Release Engineering
Mozilla Corporation

Issues and solutions encountered in maintaining a single code base under active development in both hg & git formats.

CHAPTER

ONE

BACKGROUND

- Mozilla Corporation operates an extensive build farm that is mostly used to build binary products installed by the end user. Mozilla has been using Mercurial repositories for this since converting from CVS in 2007. We currently use a 6 week “Rapid Release” cycle for most products.

  Speaker Note

  We currently have upwards of 4,000 hosts involved in the continuous integration and testing of Mozilla products. These hosts do approximately 140 hours of work on each commit.

- Firefox Operating System is a new product that ships source to be incoporated by various partners in the mobile phone industry. These partners, experienced with the Android build process, require source be delivered via git repositories. This is close to a “Continuous Release” process.

  Speaker Note

  A large part of the FxOS product is code used in the browser products. That is in Mercurial and needs to be converted to git. Most new code modules for FxOS are developed on github, and need to be converted to Mercurial for use in our CI & build systems.

CHAPTER

TWO

SUMMARY

- **What we initially set out to do:**
  - **Make it purely a developer choice which dvcs to use.**

    Speaker Note

    Ideal was to allow developers to make dvcs as personal a choice as editor.

  - **Support multiple social coding sites.**

    Speaker Note

    These social coding sites, such as github and bitbucket, make it much easier for new community members to contribute.

- **That was much tougher than anticipated.**

  In theory, git & hg are very close...
  ... In practice, “the devil is in the details”.

- **Where we are:**
  - Changed direction to support FFOS release to partners.
  - Quickly mirror Repository of Record (RoR) between git & hg.
  - CI/build system remains Mercurial centric.

CHAPTER

THREE

CHALLENGE AREAS

- **Changesets have different hashes in Mercurial and git.**
  - We added tooling to support both in static documents such as manifest files.
  - All tools continue to use hg hash as primary value for indexing and linking.
- **Propagation delays of changesets to the "other" system.**

  Speaker Note

  For most use cases, the approximately 20 minute average we're achieving is acceptable.

  - **Compounded by hash differences between two systems.**

    Speaker Note

    A common use case here is a developer wanting to start a self serve build. If the commit was to git, the self serve build won't be successful until that commit is converted to hg.

    We are continuing work on this. It is closely tied to determining which commit broke the build, when multiple repositories are involved.

- **Build details**
  - Movable tags are not popular in git based workflows, but have been a common technique at Mozilla to mark "latest".

CHAPTER

FOUR

CHALLENGE AREAS (CON'T)

- **Mixed philosophies are often linked with mixed repositories.**
  - Android never wants history to appear to change. Downstream servers allow only fast forward changesets and deny deletions.
  - **Mozilla uses "RoR is authoritative".**

    Speaker Note

    Either approach is self consistent. It is when the two need to interact that challenges arrise.

- **Conversion failures**
  - **Occasional hg-git conversion failures, due to implementation details of hg & git.**

    Speaker Note

    * Dates in export patches (e.g. hg uses seconds, git uses minutes, in time)
    * Email validation (git stricter than hg)

  - **Since commit already accepted by hg, hg-git must be modified**

    Speaker Note

    This requires inhouse resources to respond urgently to patch the conversion machinery. Without conversion, there are no builds.

CHAPTER

FIVE

ALTERNATE APPROACHES

- **To support your own "use the DVCS you want" infrastructure requires:**
  - production quality hg server
  - production quality git server
  - in house ability to address conversion issues (as already mentioned)
- I'm aware of two commercial alternatives. Both of these use a centralized RoR which supports git and/or hg interfaces for developer interaction.

Speaker Note

And at least one explicitly does not have a git back end.

- You can leave it to developers to scratch their own itch independently. Given diversity of workflows, this may be more cost effective than obtaining consensus.

CHAPTER

SIX

FUTURE RESEARCH

Areas of particular interest for further study include:

- What is the set of enforceable assertions which would ensure the tooling can maintain lossless conversion between DVCS?
- What minimum conditions must be maintained in conversions to preclude downstream conflicts?
- What workflows can be supported to minimize issues?
- Are there best practice incident management protocols for addressing problem commits.

Speaker Note

The common example is a commit contains sensitive material it should not. There are cases were limiting the scope of distribution can have significant business value.

CHAPTER

SEVEN

THANK YOU

References:

- http://oduinn.com/blog/2013/04/23/infrastructure-load-for-march-2013/
- http://oduinn.com/blog/2012/08/21/137-hours-compute-hours-every-6-minutes/
- https://www.mozilla.org/en-US/firefox/partners/

Blog:

- http://dtor.com/halfire/