VisIt SVN to Git + GitHub Transition Notes

11/30/17

Lawrence Livermore National Laboratory

Outline

- Repo Structure
- Release Workflow

Current SVN Repo Layout

- data
- docs
- src
- releases
- test
- third_party
- windowsbuild
- vendor_branches

Current SVN Repo Layout

- data (4.3 gb history, 1.1 gb head)
- docs (1 gb history, 400 mb head)
- src
- releases
- test
- third_party (5 gb history, 1.5 gb head)
- windowsbuild (2.3 gb history, 3.1 gb head)
- vendor_branches

Things we want to migrate to git

- data
- docs
- src
- ~~releases~~
- test
- ~~third_party~~
- windowsbuild
- ~~vendor_branches~~

Rationale:
releases and *third_party* are really just used to host files, they don't need to be rev controlled. We can host them with a simple webserver instead of in git.

vendor_branches was used to keep our own forks of tpl source (like vtk). There are now other practical options for this (like a proper git fork), and we don't currently have any active dev like this

Proposed Top Level Source Repo Design (“visit”)

- scripts (extract relevant stuff from svn_bin, including build_visit)

- src
  - new sub-dir: data (extract source code for creating example data from old “data”)
  - new sub-dir: test (extract source code related to testing from old ”test”)
  - new sub-dir: docs (extract website (maybe) and sphinx docs from old “docs”)

Proposed Other Repos

- visit-project-resources
  - (other stuff from docs?, website (maybe), presentations, etc)

- visit-data
  - test-baselines (from tests)
  - new sub-dir: test-data (tarballs of example data from old “data”)

- visit-dependencies
  - windows-build

Proposed Top Level Repo Design (visit w/ submodules)

- scripts (extract relevant stuff from svn_bin)

- src
  - new sub-dir: data (extract source code for creating example data from old “data”)
  - new sub-dir: test (extract source code related to testing from old ”test”)
  - new sub-dir: docs (extract sphinx docs from old “docs”)

- visit-project-resources (included as submodule)
  - (other stuff from docs?, website, presentations, etc)

- visit-data (included as submodule)
  - test-baselines
  - test-data

- visit-dependencies (included as submodule)
  - windows-build

What is Git LFS?

- Git LFS = Large File Storage
  - Optimizations for revision control and use of large binary files w/ git
    - Checksums are stored in the git repo instead of actual files
    - Only pull specific version (not full history) of the actual data
  - Bandwidth and storage are metered on github, we will buy data packs

Things that will require git-lfs

- visit-project-resources
- visit-data
- visit-dependences

- In these projects, we will setup rules to store all binary files (*.tar.gz, *.pdf, *.gif, *jpg, etc) with git-lfs, regardless of size

Notes

- git-lfs:
  - Main visit repo doesn't require git-lfs (reduces complexity)
  - other repos use git-lfs

- submodules:
  - When using main repo, pulling submodule contents is optional, only needed for specific cases:
    - (windows builds)
    - (using testing data)
    - (accessing project docs other than manual)
  - Why even include as submodules?
    - It allows us to rev-control which commit of each repo we match to the main repo.
    - (for example, which commit of visit-test-data we are using)

- web hosting:
  - Github can host tarballs w/ release binaries (no limit on # of files, each file must be <1 gb)
  - We can use web server to host third_party tarballs for build_visit

Git Workflow

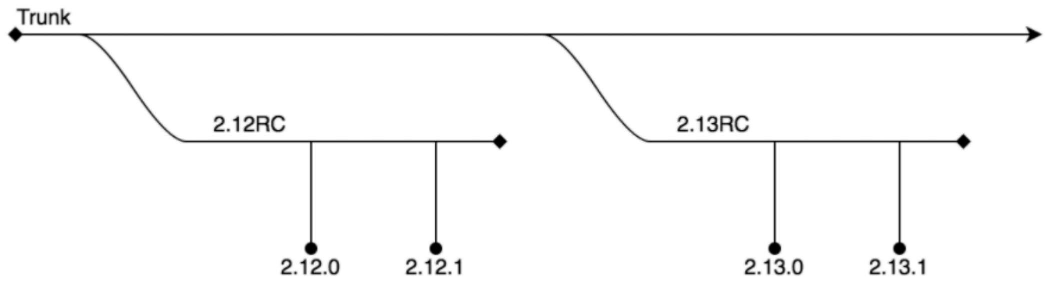
- A git workflow is a recipe for how to use branches, for development, releases, etc.

- We will use the commonly used ”Git Flow” workflow.
  - https://www.atlassian.com/git/tutorials/comparing-workflows#gitflow-workflow
  - We will use ”topic” branches + CI-vetted Pull Requests for all merges, these slides focus on how we use branches for releases

- **The basic release strategy is very similar to our current RC branch-based SVN workflow.**

Workflow: Release Branches

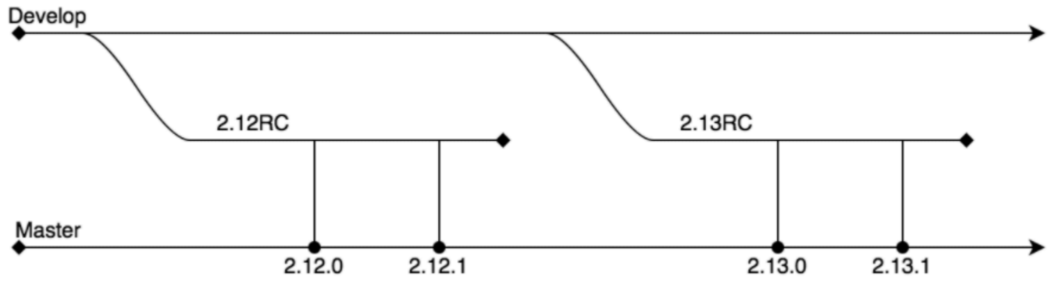
Current Release Branch Workflow



Releases are tagged off of RC branches

Workflow: Release Branches

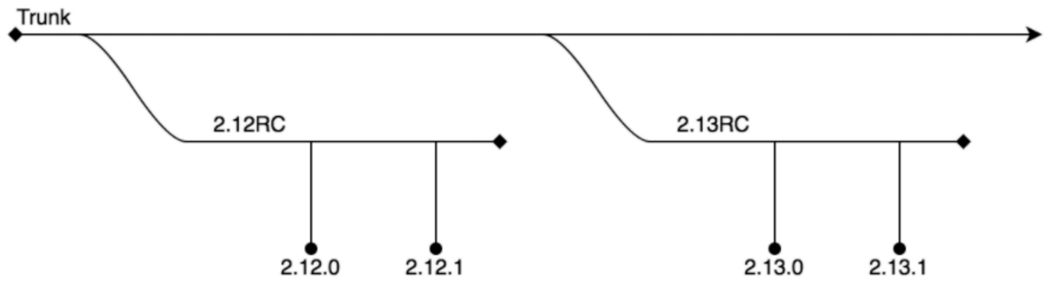
Gitflow-based Release Branch Workflow



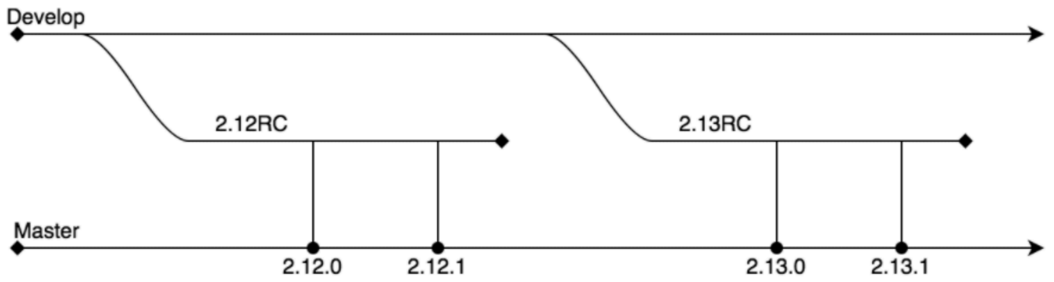
RC branches merge into Master, releases are tagged off of Master.

Workflow: Release Branches Comparison

Current Release Branch Workflow



Gitflow-based Release Branch Workflow


Develop
2.12RC
2.13RC
Master
2.12.0
2.12.1
2.13.0
2.13.1