Proposal: Minimal Migration of IceTrayCombo from SVN to Git

Software Call 2020-03-17

Sebastian Fiedlschuster
https://github.com/fiedl/icecube-git-migration
sebastian.fiedlschuster@fau.de

Erlangen Centre for Astroparticle Physics

March 17, 2020

Document 2020-nie9Ook9

Federal Ministry of Education and Research

IceCube South Pole Neutrino Observatory

ECAP Erlangen Centre for Astroparticle Physics

FAU Friedrich-Alexander Universität Erlangen-Nürnberg

Resources

Repository with proposal, tests and experiments:
https://github.com/fiedl/icecube-git-migration

? This repo is private. Can I make it public?

Build and run combo on github actions:
https://github.com/fiedl/icecube-combo-install

Current combo on github:
https://github.com/IceCube-SPNO/IceTrayCombo

Current combo on svn:
http://code.icecube.wisc.edu/svn/meta-projects/combo/trunk/

These slides as as LaTeX:
https://github.com/fiedl/icecube-git-migration-talk/releases/

Motivation and Scope

- We've talked about the git migration at 2019-10-30 and opened the `##git_migration` slack channel.
- Have talked about the major blockers with Alex on 2020-03-05.
- Want to help to bring the migration one step forward by proposing a tested set of choices for discussion.
  - ✓ Bundle a minimal concept
  - ⇒ Test individual parts of this concept
  - ☐ Prepare a proof of concept and try it out with a couple of people
- This talk: Current state

Motivation and Scope

Goal

Idea for a minimal combo migration:

- Have combo writable on git
- keep other meta projects on svn

See also: Define minimal first migration step, https://github.com/fiedl/icecube-git-migration/issues/1

Choices

- Use the existing https://github.com/IceCube-SPNO/IceTrayCombo repository
- Make https://github.com/IceCube-SPNO/IceTrayCombo git-writable
- Allow non-linear histories
- Allow pushes to master
- Encourage, but don't require pull-request workflow, including reviews
- Enable checks for pull requests:
  - Reject if the icecube password is committed
  - Reject if large (gigabyte) files are committed
  - Make sure the commit authors have valid email addresses
  - Make sure the tests are green (CI)
- Run same checks for pushes to master and send warnings (or even reset?)

Continued on next slide ...
For the reasons, check https://github.com/fiedl/icecube-git-migration/issues/9

Choices

- Make people collaborators and allow feature branches in origin
- Allow issues on github
- Allow private forks of combo
- Have `env-shell.sh` print combo's git commit id. Also log the current datetime and the commit id to log file on icetray execution.
- Welcome to IceCube! Add a `README.md` with getting-started and practical usage information. how to run the tests, links to documentation, and further resources. (like monopole-generator/README.md)
- Test the build instructions with github actions
- Run the tests with github actions (CI) — for all pull requests and branches
- (almost) never rewrite history, i.e. never change commit hashes that have already been pushed
- Import histories of the individual projects, as well as histories of icerec and icesim.
- Allow, but discourage shadow repositories of individual projects
- Make a series of screencasts to help the adoption of git by showing how-tos and highlighting the advantages — I do have the necessary equipment and volunteer
- Define the `##git_migration` slack channel as central point of asking for help.

Pros

- We can have combo in git!
- We can use all the niceties of github (issues, actions, online code browser, reviews, ...)
- Automatic checks and CI for pull requests and commits
- All histories are preserved
- No redundant repos, i.e. no need to sort and migrate commits by hand, later
- Meta projects in svn still work
- As frustration-free as possible for the users
- People can still push to master or to trunk on the svn bridge; but we do encourage pull requests and code reviews; it has a nice feel to get code reviewed and approved

See also: https://github.com/fiedl/icecube-git-migration/issues/9

Cons

- This is no silent change, but every icecuber has to do something
  - Get a github account
  - Get access to the IceTrayCombo github repo
  - Do a fresh checkout, or change the source url, or do a git clone
- There is a time delay of possibly several seconds after pushing to master, after which a push can be rejected and reset.
- People that love the beauty of linear commit histories may be disappointed at first (but we can still provide a beautified `git log` command)

See also: https://github.com/fiedl/icecube-git-migration/issues/9

Tests

Run a series of tests for individual steps of the concept:

- ✓ Test merging a project into combo while preserving both histories.
- ✓ Use svn externals to get github combo code into svn meta projects.
- ✓ Use gitman to get projects into svn meta projects.
- ✓ Use github actions to run build
- ✓ Use github actions to run unit tests
- ⇒ Test how to change the origin sources of local repos: `svn external`? `svn relocate`? `git clone`?
- ⇒ Import pre-combo history while preserving the current combo git history

See also: https://github.com/fiedl/icecube-git-migration/issues/9

Tests

After that, test the concept as a whole:

- ☐ Create a private, but writable proof-of-concept copy of IceTrayCombo and usage and migration info in `README.md`
- ☐ Test local migration of combo with a group of people
- ☐ Test local migration of a meta project with a group of people
- ☐ Test committing to combo with a group of people
- ☐ Test updating a meta project with a group of people

Local migration steps

What would an icecuber need to change locally?

- Create a github account
- Become a member of the github icecube group
- Copy ssh key to github
- To get combo, one now needs to run something different:

```
# old:
svn checkout http://code.icecube.wisc.edu/svn/meta-projects/combo/trunk src

# new:
git clone git@github.com:IceCube-SPNO/IceTrayCombo.git src
svn checkout https://github.com/IceCube-SPNO/IceTrayCombo.git/trunk src
```

? Do we know how to "symlink" the meta-projects/combo folder to github?

See also: https://github.com/fiedl/icecube-git-migration/issues/9

Steps until the goal is reached I

Proposal phase:

- ✓ Import combo history to git: IceTrayCombo
- ✓ Propose a set of choices (#9)
- ⇒ Check with experiments and tests that this proposal would actually work
- ☐ Challenge proposal by posting scenarios in the `##git_migration` slack channel or in the issues
- ☐ Approve this proposal
- ☐ Make an early announcement that everyone should get a github account and collaborator access

Test phase:

- ☐ Import simulation and reconstruction history to git
- ☐ Provide a `README.md`: How to migrate, clone, build, run tests, commit, find further resources.
- ☐ Test these instructions with a group of people and a copy of IceTrayCombo

Steps until the goal is reached II

Going live:

- ☐ Enable github issues
- ☐ Enable private forking
- ☐ Publish announcement of the upcoming change, including how-to instructions
- ☐ Make svn read-only and refer to instructions on how to proceed
- ☐ Update the meta-project svn externals to point to read-only svn
- ☐ Add git as svn external for the combo meta project in the svn repo
- ☐ Make the repo git-writable

See also: https://github.com/fiedl/icecube-git-migration/issues/9

Next steps after the goal is reached

- ☐ Add CI build using github actions
- ☐ Add CI tests using github actions
- ☐ Add commit id to env-shell.sh output
- ☐ Log commit id and datetime when running env-shell.sh and icetray
- ☐ Publish a series of screencasts to help with the migration
- ☐ Add checks (author email, file size, ...) to pull requests
- ☐ Add checks (author email, file size, ...) to pushes to master
- ☐ Import histories of individual projects into git
- ☐ Import tags for project versions to git that are referenced in meta projects
- ☐ Import histories of other individual projects, e.g. residing in other git repos
- ☐ Import pre-highlander history to git
- ☐ Possibly migrate other meta projects to git, possibly using gitman

See also: https://github.com/fiedl/icecube-git-migration/issues/9

And now?

Are there any
Questions? Ideas? Concerns? Opinions?

Next steps:

- Please challenge this proposal by posting scenarios in the `##git_migration` slack channel or in the fiedl/icecube-git-migration repo's issues section.
- I will report back in the software call when the tests are finished.

Keep up to date: https://github.com/fiedl/icecube-git-migration

? Can this repo and these slide be made public?

Thanks for your attention!

Any input you might have is welcome:

https://github.com/fiedl/icecube-git-migration/issues

sebastian.fiedlschuster@fau.de
Slack: @fiedl

Wishlist Nathan Whitehorn I

See also: Slides | #9 | #5

- Commit authors need to be identifiable (have meaningful emails)
  - We can ensure this in pull requests on github with checks
  - We can run the same tests after pushes to master, and then send warnings, or even reset master, and move the changes into a feature branch
- Answer questions like "what was the repository state on May 25th?"
  - We can achieve this by logging date and commit id into a log file when executing `env-shell.sh` and icetray.
- Is access control for sub-folders an issue?
  - We cannot restrict write access to sub folder when allowing pushes to master (this would require gitlab or github enterprise)
  - What would be the benefit of having project-wise access control? Would it be worth the effort?
  - We can run access checks after pushes to master, and then send warnings, or even reset master

Wishlist Nathan Whitehorn II

- How do we prevent big-file commits?
  - We can do checks for pull requests, but not reject pushes to master based on these checks
  - We can run the same tests after pushes to master, and then send warnings, or even reset master, and move the changes into a feature branch
- Git migration of combo drops svn commits before Oct 2018
  - We can still include other histories (which I am in favor for)
  - But from the git graph, it is not obvious then that the code has been merged in 2018
- Migrate trac issues to github tickets? How could we keep the existing links intact?
  - We can open github issues in any case
  - We can move open trac issues to github if we want, and link the new github issues in the trac thread (but we could also keep track of currently open issues in trac, i.e. live with both systems in parallel for now)

Wishlist Nathan Whitehorn III

- What to do with metaprojects?
  - They can still live in svn and still use svn externals to include the code of projects living in combo.
- What to do with sandboxes?
  - Erik suggests to have them in (possibly private) repos in the users' namespace on github.
- What to do with papers?
  - Erik suggests solutions (slide 7)
- How do we move pre-highlander history to git (important information)?
  - We can import those histories as well
  - But from the git graph it will not be clear when the merges actually have happened

See also: Slides | #9 | #5

Wishlist Erik Blaufuss and Don La Dieu

- Sandboxes: Use private github account; also svn sandboxes will not disappear.
- Also plan for analysis code and papers (slide 7)
- Suggests shadow projects to allow release-based imports into meta projects
  - We can have svn meta projects without shadow projects using svn externals and github's svn bridge, or by using a manager like gitman
  - See also: Discussion: Single combo repo vs. separate project repos
- Svn is not going away
  - We can keep our svn repo readonly
  - Maybe we can use github's svn bridge
  - But we should *not* allow a separate svn repo for write access to save us from the work of sorting and cross-syncing

See also: Slides | #9 | #7

Wishlist Alex Olivas I

See also: #9 | #8

- the main sticking point is figuring how to make the git workflow work for us at scale. Or really coming up with some dev model (gitflow variant) that works.
  - encourage pull requests and code reviews
  - but allow commits to master
  - add people as collaborators to the combo repo, such that they don't need to fork, but can push feature branches
- to answer questions like "what was the state of combo at 23 October" (because "it has worked then, why doesn't it now") is an issue, but a minor one. this is a natural problem with git that we'll just have to get used to.
  - See discussion: The code was running at 23 October
  - We should log the commit id and the current date time to a log file when executing the env-shell.sh and icetray. This way, we could ask people that need support to grep that log file (`grep 2019-10-23 /var/log/icetray.log`) and give us the commit id from the output.

Wishlist Alex Olivas II

- If we go with vanilla gitflow, where everyone forks and submits a PR, what's the process for accepting PRs that's sustainable? We have 100 projects, almost 40 developers on paper, and up to 100 collaborators committing in a year. Through the upgrade when we're expecting more activity, if we go with the Linux kernel model, then I'm probably the Linus and I approve all PRs going into the master branch. Or do we go with something closer to what we have now where people can commit directly to anything, with little oversight.
  - a good start would be a couple of automated requirements like checking for over-sized files, an email address being present, and CI tests
  - reviews, while not being mandatory, could increase the confidence in a pull request, and increase the speed it is going to be merged
  - if we deny pushes to master, the frustration might be higher, because something people were used to do does not work anymore. But the migration should feel good, instead.
  - What we can do: Run checks also after pushes to master and have the bot send feedback to people. (Your code has been moved into this feature branch, because ...; check results; this is how you can add more commits, ...) – People can still work as usual, but get constructive feedback.
  - Later on, we could still decide to deny pushes to master.

Wishlist Alex Olivas III

- I'll actively discourage collaborators from clinging to svn.
  - We can use Github's svn bridge for svn externals of meta projects
  - If people really want to still use svn, they'll probably figure out that the svn bridge is there (https://lmgtfy.com/?q=github+svn)
  - But we should tell people about the advantages of using git
- Should we allow people to push to master? Or should we require pull requests? Or some hybrid?
  - I vote for allowing pushes to master, at least in the beginning, in order to avoid mega frustration
  - it's just one setting in the github repo
- Should combo be one repo, or git submodules, or some kind of package manager? The main concern are separate meta projects (e.g. ehe or pnf) that require individual projects from combo.
  - I vote for one big repo, see discussion *Single combo repo vs. separate project repos*
  - We can use github's svn bridge to add combo projects to svn meta projects
  - Later, we can use gitman to manage dependencies
  - The other meta projects should not live in combo, because more maintenance would then be needed in preparation of a release, because we would add to the dependencies. For a combo release, the whole combo code needs to be in a good state.

Wishlist Alex Olivas IV

- Some projects want to stay with svn
  - Meta projects and other projects can stay in svn, as long as these projects are not in combo
  - For combo projects, it would be easiest to use the svn bridge if this is really necessary
- really we'll want to provide people with the option to compose meta-projects based on combo. This can be done with svn externals. The only downside is that it's not a clone, so people can't push back and not great for development. But we can live with that.
- Before we start making decisions though we want to make sure we're choosing the optimal solution and have gamed out as many scenarios as we can.
  - Please challenge this proposal by posting scenarios, either in the `##git_migration` slack channel, or in this repo's issues.

Wishlist Alex Olivas V

- We've already run into coordination issues with PROPOSAL, where we missed a critical (to some) bug fix. A bug was discovered about a week ago in PROPOSAL. Turns out the fix was committed to PROPOSAL in GitHub last October. We've since had two releases of combo where the bug fix wasn't synced to combo in svn. I would worry about doing something similar with all of combo if we can avoid it.
  - We move combo to github
  - Meta projects use github's svn bridge – no redundancy there
  - People could use github's svn bridge to commit code – no redundancy there
  - For all pull requests, one can see if they are already merged to master
  - With a one liner, one can check whether a certain commit is included in a release, branch, etc.
- Add CI pipeline
  - We can use github actions to test the build, run the tests, and any other arbitrary code
  - But considering our build time, github actions will cost money if the repo is private
  - This can be circumvented by moving the actions into a public repo, but this will drop some niceties like having tests for pull requests

Discussion: Single combo repo vs. separate project repos

- A single repo is easier to grasp for newcomers. One single repo, one central README as entry point. Get all necessary code with one single git clone.
- The github ui makes it easy to filter by project. Projects get their own rendered READMEs, their own histories, etc.
- Meta projects can pull releases of individual projects from single-project repos as well as from a "big" combo repo alike
- We can merge single-project git repos later into combo, preserving both histories
- During the 2019-10-31 software call, it was argued that the optimal solution would be to manage dependencies using a package-management tool. Gitman is a simple python tool freeze dependencies based on release tags or commit ids. It supports both git and svn. But this cannot be used to manage inter-project dependencies if each project has its own repo (and its own gitman definition).

Discussion: The code was running at 23 October

- This discussion arose in the 2019-10-31 software call from the scenario: "I checked out the code on 23 October. Then it worked. Now it doesn't. Why?"
- In git, by design, there is no answer to the question: What was the code on 23 October. Because several people can have their own branches at this time.
- Several people were hoping that enforcing a linear history would enable us to answer the question. But if I create commits locally, push them together next week, nobody has pushed in the meantime, i.e. fast-forward, then I will have changed the answer to the question: What was the code on 2 Nov.
- I've never seen something like `git log --show-push-date`, because I think, this is not well defined in git, because git is distributed, i.e. there could be several truths to the answer. I could push the code to several remotes at different times.
- One solution would be to rewrite history on push such that I have the commit date be the push date. But I really don't like history rewrites, because they change my commit ids.
- We should log the commit id and the current date time to a log file when executing the `env-shell.sh` and icetray. This way, we could ask people that need support to grep that log file (`grep 2019-10-23 /var/log/icetray.log`) and give us the commit id from the output.

Discussion: Changing history

- During the 2019-10-31 software call, it has been suggested that history rewrites might provide solutions to some of the open issues.
- Personally, I log my commit id in my lab journals. This way, I can reproduce results later.
- Also, when I work with somebody else on a feature branch, we need to agree on commit ids. When one of us changes commit ids, then our shared history is broken.
- When changing commit ids of commits already pushed, someone else could have fetched those (outdated) commits and based his own work on these. When merging his code later, this would re-introduce the outdated commits (or force to do a rebase, which then requires to change commit ids, again)
- Knowing a commit id of some point in development, gives us the information what commits the developer already during the development. When rebasing, this information is lost.
- I would vote for: Never change commit ids of commits that have already been pushed.
- This requires us to allow (non-linear) graph histories.
- Non-linear histories are harder to read
- Non-linear histories reflect the truth of what actually happened during development

Discussion: Running checks for pushes to master

- Github allows checks for pull requests (author email, big files, ...)
- Github only allows checks for pull requests. For rejecting pushes to master based on these checks, we would need gitlab or github enterprise.
- We still can run these checks after pushes with github actions.
- We can use this to send warnings via email.
- We can even reset the master and move the commits into a feature branch using github actions. But there is a time delay of several seconds.