Git and/or the New Sage Development Workflow

Making distributed version control work for you

Volker Braun

Oxford University

September 23, 2013

Outline

Linguistic Approach

git /gɪt/
v Appalachian & southern US
    variant of *get*
n Brit slang pejorative
    foolish or worthless person

```
GIT(1)                    Git Manual                    GIT(1)

NAME
       git - the stupid content tracker

SYNOPSIS
       git [--version] [--help] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path]
           ...
```

Git, the DVCS

- Developed in 2005 to manage the Linux source code

  I'm an egotistical bastard, and I name all my projects after myself. First "Linux", now "git" – Linus Torvalds

- Slated to overtake *Subversion* as the most popular VCS this year.
- **D**istributed – there is no central server
- **V**ersion **C**ontrol **S**ystem – manage changes to documents
- Git is free and open: `http://git-scm.com`
- Official `git` implementation: command-line program
- Various graphical user interfaces; I like `gitg` and `git-cola`
- Various websites offer git hosting (Github, Bitbucket, Mathematical Institute `https://git.maths.ox.ac.uk`)

Demo

Introduce the following commands:

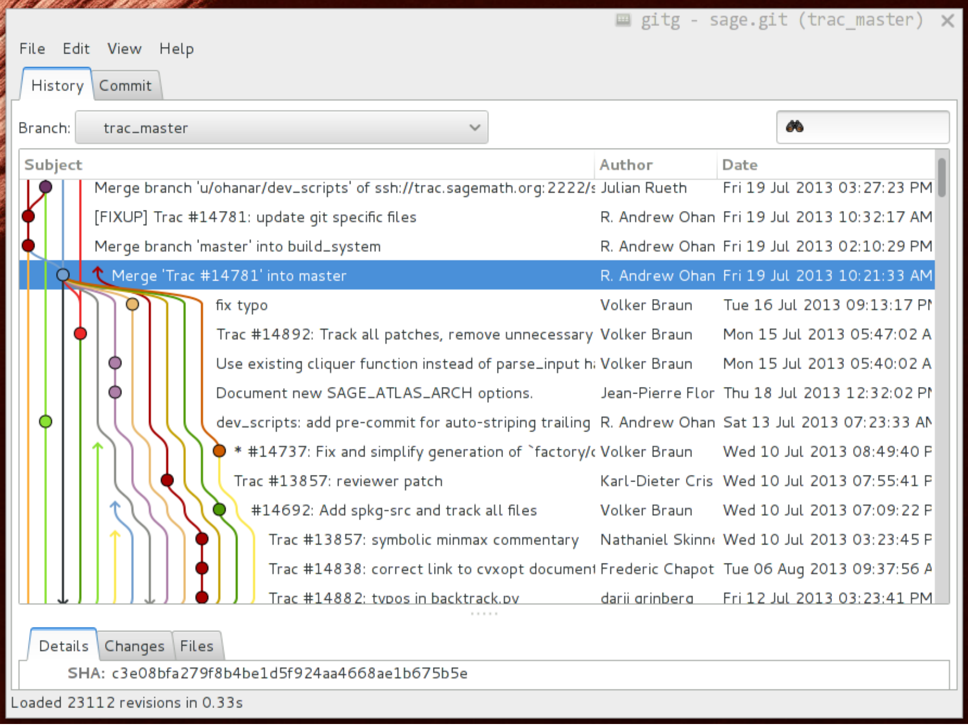
- Copy repository from github:
  `git clone https://github.com/vbraun/talk-git-sage-workflow.git`
- View history:
  `git log`
- Show current branch:
  `git branch`
- Switching between branches:
  ```
  git checkout master
  git checkout my_branch
  ```

The Git Directed Acyclic Graph

Whenever you run `git commit`, a snapshot of the current state[1] is added to the repository.

- Only forward: you can add commits, but never remove them.
- But: you can abandon them.
- Most of the time, commits have one (direct) parent commit and one child commit.
- Multiple parents: *Merge* commit
- Multiple children: number can always increase in the future...

[1] Of the *staging* directory tree, see next slide.



Subject	Author	Date
Merge branch 'u/ohanar/dev_scripts' of ssh://trac.sagemath.org:2222/s	Julian Rueth	Fri 19 Jul 2013 03:27:23 PM
[FIXUP] Trac #14781: update git specific files	R. Andrew Ohan	Fri 19 Jul 2013 10:32:17 AM
Merge branch 'master' into build_system	R. Andrew Ohan	Fri 19 Jul 2013 02:10:29 PM
Merge 'Trac #14781' into master	R. Andrew Ohan	Fri 19 Jul 2013 10:21:33 AM
fix typo	Volker Braun	Tue 16 Jul 2013 09:13:17 PM
Trac #14892: Track all patches, remove unnecessary	Volker Braun	Mon 15 Jul 2013 05:47:02 A
Use existing cliquer function instead of parse_input h	Volker Braun	Mon 15 Jul 2013 05:40:02 A
Document new SAGE_ATLAS_ARCH options.	Jean-Pierre Flor	Thu 18 Jul 2013 12:32:02 PM
dev_scripts: add pre-commit for auto-striping trailing	R. Andrew Ohan	Sat 13 Jul 2013 07:23:33 AM
* #14737: Fix and simplify generation of `factory/c	Volker Braun	Wed 10 Jul 2013 08:49:40 F
Trac #13857: reviewer patch	Karl-Dieter Cris	Wed 10 Jul 2013 07:55:41 F
#14692: Add spkg-src and track all files	Volker Braun	Wed 10 Jul 2013 07:09:22 F
Trac #13857: symbolic minmax commentary	Nathaniel Skinne	Wed 10 Jul 2013 03:23:45 F
Trac #14838: correct link to cvxopt document	Frederic Chapot	Tue 06 Aug 2013 09:37:56 A
Trac #14882: typos in backtrack.py	darij grinberg	Fri 12 Jul 2013 03:23:41 PM

SHA: c3e08bfa279f8b4be1d5f924aa4668ae1b675b5e

Loaded 23112 revisions in 0.33s

The Staging Area

Three places to store files:

- The git database (the `.git` directory)
- Staging area
- The working directory: all files outside of `.git`

Staging area

The staging area are the files that will be committed by `git commit`

- Show staging: `git status`
- Add to staging: `git add <filename>`
- Remove from staging: `git reset HEAD <filename>`

Committing Changes

Creating a commit

- `git commit`
- Specify commit message on the command line:
  `git commit -m "my commit message"`

Each commit is uniquely specified by the SHA1 hash[2] of

- All changes to files
- All parent commits
- The commit message

None of these can ever be changed, including all direct and indirect parents.

[2] a 40 digit hex number

Branches

Branches organize parallel development

- A branch is just a shortcut for a particular commit
- If you create a new commit, the branch automatically advances to it
- The default branch is `master`, but you can use any name
- `HEAD` is the commit at the tip of the branch:
  `git show HEAD`
- `HEAD`∼ is the parent of `HEAD`
- `HEAD`∼2 is the parent of the parent of `HEAD`
- etc.

Remote Repositories

- Remotes repositories are bookmarks.
- Configure with `git remote`
- **D**istributed VCS: all remotes are equal.
- The "important" one (to you) is usually called `origin`

If there are no conflicts:

- Upload your changes to the remote repository:
  `git push <remote>`
- Download changes from the remote repository and update the local working directory:
  `git pull <remote>`
- There is a default remote for each branch, see
  `git remote show <remote>`

Merge Conflicts

Don't Panic!

- Merge conflicts happen if there are overlapping edits.
- Resolving them is common and easy.

Example:

```
\begin{equation}
  \label{eq:quad}
  x = \frac{-b+-\sqrt{b^2-4ac}}{2a}
\end{equation}
are the two roots of the quadratic equation.
```

On the flight to a conference I change this to

```
    \begin{equation}
      \label{eq:quad}
      x_{1,2} = \frac{-b+-\sqrt{b^2-4ac}}{2a}
    \end{equation}
    are the two roots of the quadratic equation.
```

While I'm still in the air, Jennifer corrects

```
    \begin{equation}
      \label{eq:quad}
      x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}
    \end{equation}
    are the two roots of the quadratic equation.
```

and pushes it to our common remote repository.

Reconnecting...

When I try to push my commit, git rightfully refuses:

```
[vbraun@laptop]$ git push
To git@github.com:vbraun/talk-git-sage-workflow.git
 ! [rejected]        quadratic_equation -> quadratic_equation (non-fast-forward)
error: failed to push some refs to 'git@github.com:vbraun/talk-git-sage-workflow.git
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

The `git status` command tells me the same thing:

```
[vbraun@laptop]$ git status
# On branch quadratic_equation
# Your branch and 'origin/quadratic_equation' have diverged,
# and have 1 and 1 different commit each, respectively.
#   (use "git pull" to merge the remote branch into yours)
#
nothing to commit, working directory clean
```

I have to first pull[3] Jennifer's overlapping edit:

```
[vbraun@laptop]$ git pull
Auto-merging example/quadratic_equation.tex
CONFLICT (content): Merge conflict in
                    example/quadratic_equation.tex
Automatic merge failed; fix conflicts and then commit the
result.
```

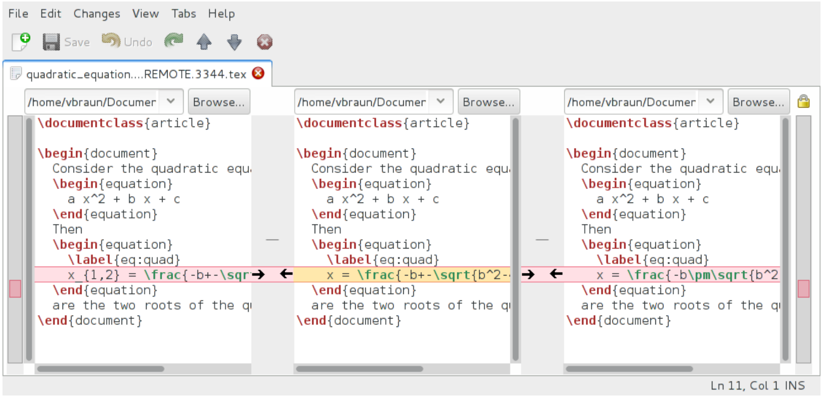
The file now looks like this:

```
      \begin{equation}
    \label{eq:quad}
<<<<<<< HEAD
    x_{1,2} = \frac{-b+-\sqrt{b^2-4ac}}{2a}
=======
    x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}
>>>>>>> d0615cf02b5615a07c34633dabaf3c0eb57cac7a
  \end{equation}
  are the two roots of the quadratic equation.
```

[3] That is, download and merge

Resolving the Conflict

- Open the file in your favorite editor and fix, or
- Use a specialized program (I like `meld`): `git mergetool`


File Edit Changes View Tabs Help
Save
Undo
quadratic_equation....REMOTE.3344.tex
/home/vbraun/Documer
Browse...
Ln 11, Col 1 INS

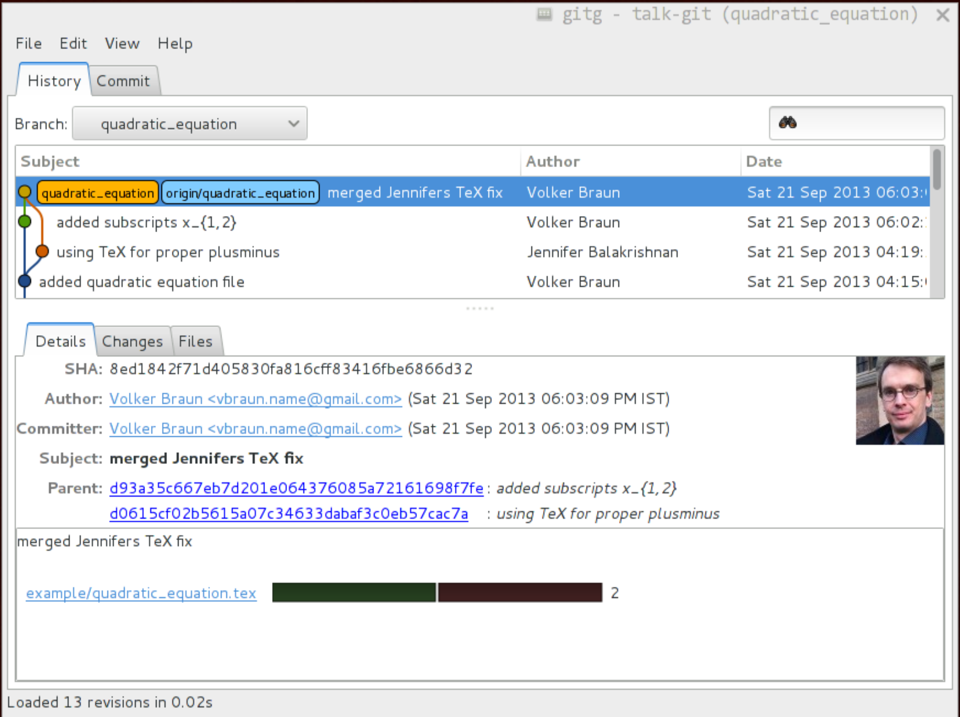
```
\documentclass{article}

\begin{document}
  Consider the quadratic equ
  \begin{equation}
    a x^2 + b x + c
  \end{equation}
  Then
  \begin{equation}
    \label{eq:quad}
    x_{1,2} = \frac{-b+-\sqr
  \end{equation}
  are the two roots of the q
\end{document}
```

```
\documentclass{article}

\begin{document}
  Consider the quadratic equ
  \begin{equation}
    a x^2 + b x + c
  \end{equation}
  Then
  \begin{equation}
    \label{eq:quad}
    x = \frac{-b+-\sqrt{b^2-
  \end{equation}
  are the two roots of the q
\end{document}
```

```
\documentclass{article}

\begin{document}
  Consider the quadratic equ
  \begin{equation}
    a x^2 + b x + c
  \end{equation}
  Then
  \begin{equation}
    \label{eq:quad}
    x = \frac{-b\pm\sqrt{b^2
  \end{equation}
  are the two roots of the q
\end{document}
```

Finishing Up

- When you are finished resolving the conflict, just commit:
  ```
  git add quadratic_equation.tex
  git commit -m "merged Jennifers TeX fix"
  ```
- Now, git lets me push to the remote repository.
- When Jennifer pulls from the remote later, she gets my change and my resolution of the conflict.
- To abort the merge:
  ```
  git merge --abort
  ```



Subject	Author	Date
merged Jennifers TeX fix	Volker Braun	Sat 21 Sep 2013 06:03:
added subscripts x_{1,2}	Volker Braun	Sat 21 Sep 2013 06:02:
using TeX for proper plusminus	Jennifer Balakrishnan	Sat 21 Sep 2013 04:19:
added quadratic equation file	Volker Braun	Sat 21 Sep 2013 04:15:

SHA: 8ed1842f71d405830fa816cff83416fbe6866d32

Author: Volker Braun <vbraun.name@gmail.com> (Sat 21 Sep 2013 06:03:09 PM IST)

Committer: Volker Braun <vbraun.name@gmail.com> (Sat 21 Sep 2013 06:03:09 PM IST)

Subject: **merged Jennifers TeX fix**

Parent: d93a35c667eb7d201e064376085a72161698f7fe : *added subscripts x_{1,2}*
d0615cf02b5615a07c34633dabaf3c0eb57cac7a : *using TeX for proper plusminus*

merged Jennifers TeX fix

example/quadratic_equation.tex 2

Loaded 13 revisions in 0.02s

Outline

Who Are You?

Your name and email address become part of the commit message

- Global configuration stored in ∼/.gitconfig. Either open in your favorite editor to add

```
    [user]
        name = Your Name
        email = you@host.com
```

- or via the command line:

```
git config --global user.name "Your Name"
git config --global user.email you@host.com
```

Trac Account

To contribute to Sage, you need

- a trac account, see instructions at `http://trac.sagemath.org`
- upload your ssh *public* key to the trac server
- This is described in detail in `http://sagemath.github.io/git-developer-guide/trac.html#authentication`, a temporary copy of the new developer guide.

Obtaining the Sage Sources

- Download the Sage git repository from github:
  ```
  git clone git://github.com/sagemath/sage.git
  ```
- Setup the “trac” remote:
  ```
  cd sage
  git remote add trac
     ssh://git@trac.sagemath.org:2222/sage.git -t master
  ```
- Note: the `-t master` means to only fetch the master branch by default
  - Pro: Avoids downloading all branches on trac; Faster and less clutter
  - Con: You have to tell git which branches to download

Downloading a Branch from Trac

Temporary change

You should use the `public/sage-git/master` branch for now. When the git transition is finished, it well be just `master`.

So, first get this branch:

- Tell git which branch to download:
  ```
  git fetch trac public/sage-git/master
  ```
- Create a new local branch from what you just downloaded:
  ```
  git checkout -b trac_master FETCH_HEAD
  ```

Then build Sage as usual (run `make`)

Uploading Changes

- Now edit files and commit changes. Just like with any other git repository.
- If you have a (new or existing) ticket, fill in the “Branch:” field with the name that you will be using to upload.
- The remote branch name must be `u/user/description`, where
  - `user` is your trac username
  - `description` is a free-form short description (and can include further slashes)
- When you are ready to share, upload to trac:

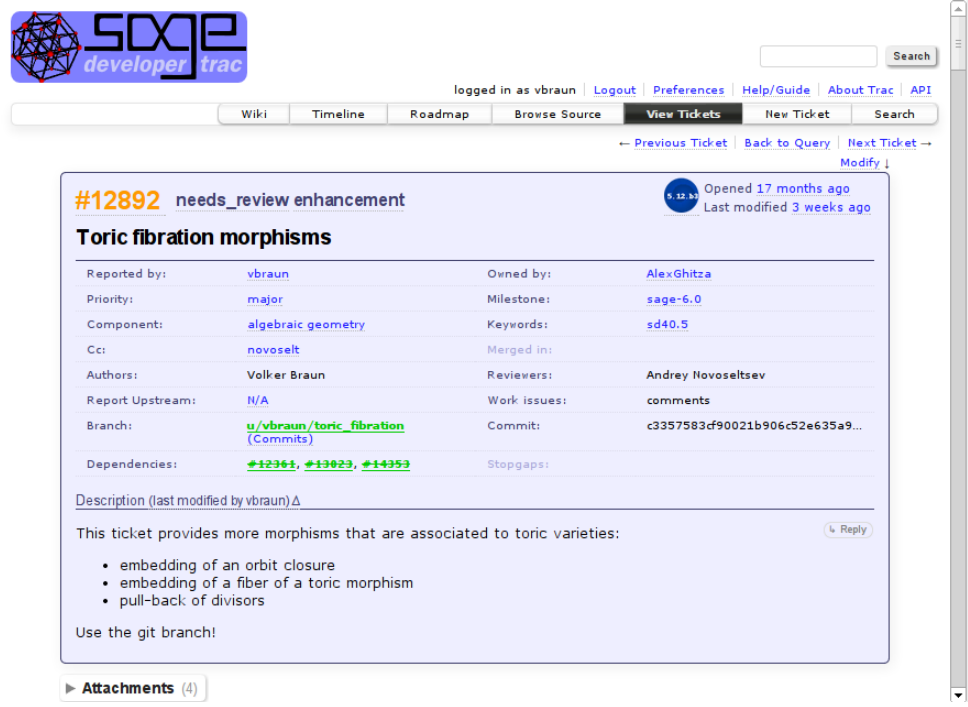
```
git push --set-upstream trac
          my_branch:u/user/description
```

- Slightly different push command for subsequent uploads:

```
git push trac HEAD:u/user/description
```

Using Trac

- When you push to a trac ticket, the “Commit:” field on the trac ticket is automatically filled out.
- The “Branch:” field is color coded:
  - Green means that it applied cleanly to the current master.
  - Red means that there is a conflict.
- If you click on the “(Commits)” link under/next to the branch, you can see the list of commits.
- Download any branch for the first time as on the “Downloading a Branch from Trac” slide.
- To get changes, use `git pull trac u/user/description`



developer trac
Search
logged in as vbraun | Logout | Preferences | Help/Guide | About Trac | API
Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search
← Previous Ticket | Back to Query | Next Ticket →
Modify ↓

#12892 needs_review enhancement

Opened 17 months ago
Last modified 3 weeks ago

Toric fibration morphisms

Reported by:	vbraun	Owned by:	AlexGhitza
Priority:	major	Milestone:	sage-6.0
Component:	algebraic geometry	Keywords:	sd40.5
Cc:	novoselt	Merged in:	
Authors:	Volker Braun	Reviewers:	Andrey Novoseltsev
Report Upstream:	N/A	Work issues:	comments
Branch:	u/vbraun/toric_fibration (Commits)	Commit:	c3357583cf90021b906c52e635a9...
Dependencies:	~~#12361~~, ~~#13023~~, ~~#14353~~	Stopgaps:	

Description (last modified by vbraun) Δ

This ticket provides more morphisms that are associated to toric varieties:

- embedding of an orbit closure
- embedding of a fiber of a toric morphism
- pull-back of divisors

Use the git branch!

Merging vs. Rebasing

While you are working on `my_branch`, Sage development continues.

```
      X---Y---Z my_branch
     /
A---B---C---D master
```

Two ways to update:

- Merge: `git merge master`

```
      X---Y---Z---W my_branch
     /           /
A---B---C-------D master
```

- Rebase: `git rebase master`

```
              X'--Y'--Z' my_branch
             /
A---B---C---D master
```

Rebasing

- Rebase: `git rebase master`

```
                    X'--Y'--Z' my_branch
                   /
A---B---C---D master
```

- Pro: Clean history.
- Con: Since the SHA1 hash includes the hash of the parent, all commits change.
- Only ever use rebase if nobody else has used one of your `X`, `Y`, `Z` commits to base their development on.
- Only rebase commits that you have not yet pushed to trac.

Merging

- Merge: `git merge master`

```
          X---Y---Z---W my_branch
         /           /
    A---B---C-------D master
```

- Pro: None of the existing commits changes
- Con: Introduces a new commit `W` that will be in the `git log` history forever.
- When you push to trac, the extra commit propagates to your collaborators.
- When in doubt, use merge instead of rebase.
- No new features in `master` that you depend on and no conflicts? Do nothing. Don't create useless merges.

Reviewing Commits

- Trac tickets are abstract goals to meet.
- Commits are individual changes of the sources.
- There is only a map ticket $\rightarrow$ subset of all commits, namely all parents of the commit listed on the “Commit:” trac field.
- In particular, a commit can be part of multiple tickets.

Commits to review

The ticket commit and all parent commits leading to the ticket are part of the review. Except for commits that are already merged into Sage:

```
git log <branch-or-sha1> ^master
```

Dependencies and Reviewing Commits

- You can list the history excluding dependencies:
  `git log <branch> ^master ^<dep1> ^<dep2>`
- But: When your ticket is merged, all parent commits are merged.
- Whether any particular parent is part of a dependency ticket can change as the dependency ticket evolves.
- In particular, you might end up with abandoned commits from a dependency.
- Hence: All parent commits are part of the review.
- To simplify review, start with the trac dependencies and have them merged into Sage.

Outline

Sage Dev Scripts

Can develop normal tickets without using git or going to the http://trac.sagemath.org web page yourself:

```
[vbraun@laptop]$ sage -dev help
usage: sage-dev [-h] subcommand ...

The developer interface for sage.

optional arguments:
  -h, --help            show this help message and exit

subcommands:
    abandon             Abandon a ticket or branch.
    checkout            Checkout another branch.
    comment             Add a comment to ``ticket`` on trac.
    commit              Create a commit from the pending
                        changes on the current branch.
    ...
```

More on Dev Scripts

- Also available in a Sage session, for example

```
        sage: dev.create_ticket?
        sage: dev.commit?
```

- Scripts will set up your name/email/ssh keys on first use.
- Not part of official Sage release yet, but usable.
- Scripts are included in the `public/sage-git/master` branch.

Working on a Ticket

- Optional: Create a ticket on trac:
  ```
  sage -dev create-ticket
  sage -dev edit-ticket
  ```
- Create a local branch to work on the ticket:
  ```
  sage -dev checkout --ticket <number>
  ```
- Work on the source code...
- Commit your changes:
  ```
  sage -dev commit
  ```
- Push your local branch to trac:
  ```
  sage -dev push
  ```

Reviewing Tickets

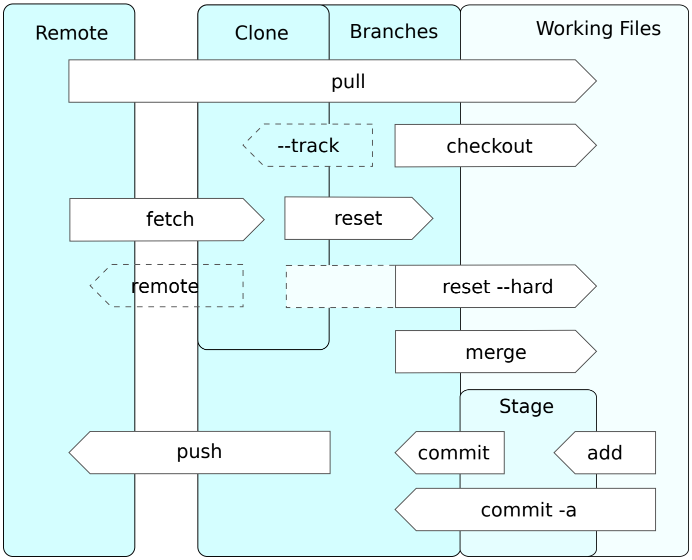
Likely to erase the dev scripts right now

If you checkout a ticket that does not contain the dev scripts, then they will be gone after the checkout.

- Checkout the ticket into a local branch:
  ```
  sage -dev checkout --ticket <number>
  ```
- If the ticket is good to go, set it to positive review:
  ```
  sage -dev positive-review
  ```
- If there is any remaining issue add a comment:
  ```
  sage -dev comment
  ```
- Or make edits yourself:
  ```
  sage -dev commit
  sage -dev push
  ```

The End. Questions?

Git Operations



More Cool Stuff

Sage+Git developer manual

The current draft for the Sage+Git development manual is here:
`http://sagemath.github.io/git-developer-guide/`

- `git help <command>` shows the help for any git command.
- `git reset` modifies the branch to point to an arbitrary commit. For example, used to abandon commits.
- `git stash` is a place to put changes temporarily
- `git reflog` history of your local git commands. Allows you to undo anything.
- Detached heads: `git checkout <sha1>` instead of `git checkout <branch>`