git - the stupid content tracker

git eats trees. version control with git is fun.

Henry S. G. Gebhardt

November 7, 2016

Outline

Version Control? Version Control.

- Save history.
- Keep track of changes.
- Merge code.
- Don't be a git. Share code.
- ...

Git, Mercurial, ~~Bazaar~~, ~~SVN~~ (why bother?), ~~CVS~~, ~~Monotone~~, ~~DARCS~~, ...
"Theory of Patches"
What is a patch?

Why version control?

- Not all VCS are worth using. . .
- Keep a backup
- Keep track of changes
- Keep multiple versions.
- Collaborate
- Revert changes
- Blame people

Git Theory

`https://git-scm.com`

History is a DAG (directed acyclic graph). *Explain graph.*

Distributed, not centralized. *Every clone has the full history.*

There are *plumbing* commands and *porcelain* commands.

Git doesn't know about files. . . whaaat?

Git only knows content. (blobs)

And how that content is assembled. (trees)

And history. (commits)

blobs, trees, and commits are identified by their SHA1-sum

A hash is a (hopefully) unique number to identify some information, like a file.

SHA-1 is a 160-bit number. It happens to be cryptographically secure.

Blobs, trees, and commits are identified by their SHA1 sum.

$\Rightarrow$ Efficient de-duplication and compression

Terminology: blah, blah, blah,...

WORKDIR
GITDIR
HEAD
Index
Local repository
Upstream repository
Stash
branch
master branch

git cheat sheet

Here's the *porcelain*:

`https://services.github.com/kit/downloads/github-git-cheat-sheet.pdf`

`man gittutorial`

Initialization once per machine:

Create the file `~/.gitconfig`.

Set your `EDITOR` variable in `~/.bashrc`.

Initial checkout

Existing repository:

```
$ git clone
ssh://drake.astro.psu.edu:~hsg113/repos/git-for-astros.git
```

New repository:

```
$ mkdir newrepo; cd newrepo
$ git init
```

Commits

`git add <file>` Add your changes to the index.

`git add -p` Be selective about what to add.

`git commit` Commit your changes.

git help command

Useful commands:

`git status` Where am I?

`git diff` What did I just do?

`git diff --staged` What will I do?

`git log` What have I done?

`gitk --all` Let's climb trees!

`git describe --always --tags --dirty` Who am I?

Sending and receiving patches

`git format-patch` Create a patch

`git send-email` Send an entire set of patches as emails.

`git am, git apply` Apply other people's patches.

Trees, yum!

git eats trees. . . nom,nom

Branches are cheap!

git branch <name> Let's make a new branch.

git branch -d Never mind.

git checkout <name> Let's climb over to that branch.

git checkout -b <newname> <starthere> Checkout and make a new branch.

git merge <otherbranches>... Trees eating trees!

git rebase -i <branchname> Clean up your history!

Pushing and pulling

```
$ git push <remote> <localbranch>:<remotebranch>
$ git push --set-upstream
$ git pull
$ git remote -v
```

Workflow

```
git mergetool
```

Play with me!

```
$ git svn
```

Works by calling “`git fast-import`”.

Hosting your git repository

Others:

Github: `github.com`, `gitorious.com`, ...

PSU: `git.psu.edu`

Your own:

SSH server: `hartmann.astro.psu.edu`, ... your own workstation

SSH server: `http://gitolite.com/gitolite/` (probably overkill)

```
$ mkdir -p ~/repos/newawesomeproject.git
$ cd ~/repos/newawesomeproject.git
$ git init --bare
```

Ah, I did something stupid...

Recovery might be possible by looking into `.git/logs/`.

Other commands

Graphs: `git log --graph`

More graphs: `gitk --all`

Tags: `git tag`

Hooks: `man githooks`; `cd .git/hooks/`

Submodules: `git submodule`

Rewrite history: `git filter-branch`

Collect garbage: `git gc`