Gitify your life

web, blog, configs, data, and backups

Richard Hartmann,
RichiH@{freenode,OFTC,IRCnet},
richih.mailinglist@gmail.com

2013-10-21

Outline

Outline

Who am I?

- Richard ”RichiH” Hartmann
- Backbone and project manager at Globalways AG
- freenode & OFTC staff
- Debian Developer
- Author of vcsh & metamonger

What is Git?

- Version control system
- Distributed
  - No need for central repository
  - Allows you to commit while offline
- Stores commits (parent commit reference, commit message, root tree object) and tree objects (blobs and other tree objects)
- Light-weight branches
- pre-/post-action hooks
- Full history in every checkout

Outline

etckeeper

etckeeper is a collection of tools to let /etc be stored in a Git, Mercurial, Darcs, or Bazaar repository

In a word

- Implemented in POSIX shell
- Auto-commits /etc prior to and after all actions by package manager
- Hooks into apt, yum, pacman-g2, and cron
- Allows manual commits
- Various back-ends
  - Bazaar
  - Darcs
  - Git
  - Mercurial
- Easy way to recover from failures, misconfiguration or to clone machines

Outline

bup

Highly efficient file backup system based on the Git packfile format

In a word...

- Written in Python
- Fast
- Very space-efficient (reduced 120 GiB (rsnapshot) to 45 GiB)
- Built-in de-duplication
- Can be mounted via FUSE
- Can not drop old data (there is a branch that supports this)

Outline

ikiwiki

ikiwiki is a wiki compiler. It converts wiki pages into HTML pages suitable for publishing on a website

What is ikiwiki?

- Written in Perl
- Supported back-ends: Git, Bazaar, Darcs, GNU Arch, Mercurial, monotone, and Subversion
- Parses various markup languages
- Offers different ways of editing content
- Extensive templating and CSS support
- Acts as Wiki, CMS, and blog
- RSS and Atom feed for whole site, per page, per tag, etc
- Supports OpenID

Supported markup languages

- MarkDown
- WikiText
- reStructuredText
- Textile
- plain HTML
- ikiwiki-specific extensions:
  - WikiLink (`[[LinkToArticle]]`)
  - directives, e.g.
    - `[[!tag talk/gitify]]`
    - `[[!author RichiH]]`
    - etc

How does it work?

- User edits web page or commits and pushes source files
- Partial/full rebuild triggered by cgi or commit hook
- Parses input files
- Compiles into HTML, create new pages, updates RSS, etc
- Commits MarkDown source for autocreated/-changed pages into repository
- User can optionally pull changes to local repository

Common uses

- Public Wiki
- Private notes
- Blog
- CMS

Adding/editing content

- Web-based text editing (useful, but boring)
- Web-based WYSIWYG (via plugins/wmd)
- CLI-based (awesome!)

Advanced usage

- Interface with source files, only
- Maintain wiki and docs in the same repository as your source code
- Separate staging or even preview branches with output into different directories

Outline

git-annex

manage files with Git, without checking their contents in

What is git-annex?

- Based on Git
- Maintains metadata in Git, actual files in the annex
- Still allows you to check files into Git if you want to
- Written with low bandwidth and flaky connections in mind
- Various work-flows (more on that soon)

Internal workings 1/2

- Written in Haskell, so strong typing etc, internally
- Uses rsync to transfer data
- Indirect mode
  - Moves files into `.git/annex/objects`
  - Makes them read-only
  - Replaces them with symlinks
  - Forces you to `git annex edit` and `git annex add`, leading to concious decisions about changes
- Direct mode without symlinks; especially useful for Mac OS X and Windows
- Can either discard or keep old data, depending on setup

Internal workings 2/2

- Uses UUIDs to identify each repository
- Stores tracking information in `git-annex` branch
- Gives every single repository full information about all files
- Tracking information designed to work with union merge:

```
1361402708.089154s 1 0d39904f-de8d-1638-92af-ecd2cea783cb
1361402822.110498s 1 d1ffde43-f3d9-107b-aa2d-7e4e1ff88b46
```

- Neat tool: `git annex sync`
  - `git commit`
  - `git merge synced/master`
  - `git annex merge`
  - `git push $remote master:synced/master`

Data integrity

- SHA1, SHA2-{224,256,384,512} for integrity
- Set minimal number of required copies per suffix, directory, etc
- All remotes and special remotes can be verified
  - remotes verify locally and transmit the result
  - special remotes have to transfer all data to verify
- Verification takes required amount of copies into account
- You can *always* get your data out of a broken indirect annex
  - All data is stored as normal files on disk
  - Symlinks work without git-annex
  - git-annex objects carry their own checksum in their filename
- Direct mode needs no recovery as it's based on plain files

Special remotes - 1/3

- Stores data in non-git-annex remotes
- Still tracks all data stored in special remotes
- Supports encryption for storage on untrusted machines/media
- Hook system lets you write to and read from arbitrary remotes

Built-ins - Special remotes 2/3

- Amazon Glacier
- Amazon S3
- bup
- directory
- rsync
- webdav
- web (media.ccc.de, Project Gutenberg, archive.org, etc)
- hook

Hook-based - Special remotes 3/3

- archive.org via Amazon S3
- IMAP
- box.com
- Google Drive
- Google Cloud Storage
- mega.co.nz
- Microsoft SkyDrive
- OwnCloud
- Usenet
- Tahoe-LAFS
- Flickr

git-annex assistant

- One full year of dedicated programming by Joey Hess, financed via `kickstarter.com`
- One more year financed directly to avoid overhead
- Daemon that adds data to the repository and syncs it between other repositories
- Web GUI on localhost
- Content notification via XMPP/Jabber
- Advanced ruleset for content distribution
- Android & Windows ports

The Archivist

- Put data into git-annex
- Distribute data among any number of drives, tapes, remotes, etc
- Store offline media in a safe place
- Maintain full information about number and location of all copies

Media consumption

- Import podcasts, videos, slides, and other media
- Built-in podcatching client (very neat)
- Sync or export to consumption devices
- Consume media
- Drop consumed media content from annex
- (Feature to propagate deletions is upcoming)

The Nomad

- Keep copies of data on the Internet
- Optionally sync between several local devices for backup
- Add data locally and/or remotely while on the road
- Sync data between local and remote once at an Internet café or similar
- Perfect for photos while travelling

Create different views or sets of the same data

- Sometimes, you disagree with other people about the best way to organize data
- Different repositories can show a different view of the same data
  - Completely delete some files or file types, for example RAW files
  - Rename files and directories
- Maintain a rebasing branch on top of the remote:

```
git config branch.master.rebase true
```

Outline

metamonger

Like metastore, but done right

Why metamonger?

- People don't seem to care about persistent file metadata enough to actually do something about it
- See discussion on if to store `mtime` etc in Git...
- `metastore` exists and was written to be used with Git, but
  - Binary storage is not merge-safe
  - Author has no interest to fix this
  - ...
- `metamonger` uses plain text (line-wise JSON) storage
- Will always merge nicely and integrate well with Git

Outline

vcsh

Version Control System for $HOME
(multiple Git repositories in $HOME)

What is vcsh?

- Implemented in POSIX shell
- Based on Git, but...
  - Git is unable to maintain several working copies in one directory
  - This is a safety feature...
  - ...which sucks if you want to keep your configs in Git
- vcsh uses fake bare Git repositories to work around this limitation
- Think of it as an extension to, or a wrapper around, Git
- Powerful and extensible hook system

fake bare.. what?

- Normal Git repository:
  - working copy in `$GIT_WORK_TREE`
  - Git data in `$GIT_WORK_TREE/.git` aka `$GIT_DIR`
- Bare Git repository:
  - Git data in `$GIT_DIR`
  - no `$GIT_WORK_TREE`
- Fake bare Git repository:
  - working copy in `$GIT_WORK_TREE`
  - Git data in `$GIT_DIR`
  - `core.bare = false`
- vcsh default:
  - `$GIT_WORK_TREE == $HOME`
  - `$GIT_DIR == $XDG_CONFIG_HOME/vcsh/repo.d/$repo.vcsh`

Problems with fake bare Git repos

- Fake bare repositories are messy to set up and use, and very easy to get wrong
- Reason why Git disallows shared `$GIT_WORK_TREE`: complexity due to sudden context-dependency
- Mistakes lead to confusion and/or data loss; imagine `$GIT_WORK_TREE` set and
  - `git add`
  - `git reset --hard HEAD`
  - `git checkout -- *`
  - `git clean -f`

Solution: vcsh

- Wraps around Git
- Hides complexity and does sanity checks
- Several Git repositories checked out into `$HOME` at once
  - One repository for Zsh, Vim, mplayer, etc
  - Allows specific subsets of repositories per host
- Manages complete repository life-cycle

Create new repository

```
# Create new repository
vcsh init vim
# Add files
vcsh vim add  /.vimrc  /.vim
# Commit
vcsh vim commit -m 'Initial release'
# Push
vcsh vim remote add origin <remote> vcsh vim push -u
origin master
```

Life-cycle

If all this looks almost exactly like vanilla Git, that's a deliberate design feature:

```
vcsh clone git://github.com/RichiH/zshrc.git zsh
vcsh zsh ls-files # Show all files in one repository
vcsh rename zsh zshrc # rename repository
vcsh status # Show status of all repositories
vcsh pull # Pull from all repository remotes
vcsh push # Push to all repository remotes
vcsh delete zshrc # Delete repository if you want
```

The three modes of interaction

- Default mode

```
vcsh zsh commit .zshrc -m "Add foo()"
# gitk not possible!
```

- Run mode

```
vcsh run zsh git commit .zshrc -m "Add foo()"
vcsh run zsh gitk
```

- Enter mode

```
vcsh enter zsh
git commit .zshrc -m "Add foo()"
gitk
exit
```

Advanced usage

- Have your prompt display vcsh information
- git-annex within vcsh to manage non-configuration files in `$HOME`
- Floating backups in arbitrary working copies
  - `.git/`
  - Working copy
  - Complete repository, including objects, etc
- Use Git on top of or in parallel to other VCSs

Outline

mr

a tool to manage all your version control repos

Tying it all together

- Written in Perl
- Run bulk pull, push, and custom commands all, some, or one of your repositories
- Supports Git, vcsh, Bazaar, CVS, Darcs, fossil, git-svn, Mercurial, Subversion, unison, and veracity
- Trivial to extend to support more VCSs
- If you want to try all this, why not `vcsh clone` my mr repository template and run `mr up` to pull my Zsh config via vcsh?

Suggested mr layout

```
% cat ~/.mrconfig
include = cat ~/.config/mr/config.d/*
% ls .config/mr/available.d
mr.vcsh
zsh.vcsh
...
% ls -l .config/mr/config.d
mr.vcsh -> ../available.d/mr.vcsh
zsh.vcsh -> ../available.d/zsh.vcsh
...
%
```

Outline

Zsh

Best shell available. Period.

Not based on Git, but makes your life easier

- Extremely powerful tab completion for the tools in this talk (and others!)
- Versatile left *and right* prompts
- `vcs_info`
  - Displays information about the current VCS working copy in prompt
  - Lots of customization options
  - Supports Git, vcsh, Bazaar, codeville, CVS, Darcs, fossil, GNU Arch, Mercurial, monotone, Perforce, Subversion, and svk
- Can mimic Bash, Ksh, tcsh, etc.
- Too many other reasons to list (literally...)

Outline

The final pitch...

I need literally less than five minutes of Internet access to sync my entire digital life while on the road.

Project websites

Most of these are packaged for the major distributions

- etckeeper: `http://joey.kitenet.net/code/etckeeper/`
- bup: `https://github.com/bup/bup`
- ikiwiki: `http://ikiwiki.info/`
- git-annex: `http://git-annex.branchable.com/`
- vcsh: `https://github.com/RichiH/vcsh`
- mr: `http://kitenet.net/~joey/code/mr/`
- Wiki around this topic:
  `http://vcs-home.branchable.com/`

Previous talks

Previous talks, available as video download:

- vcsh:
  `http://fosdem.org/2012/schedule/event/vcsh`
- git-annex:
  `http://fosdem.org/2012/schedule/event/gitannex`

Thanks!

Thank you for listening!

Questions? Ask now or catch me after this talk, both is fine.

See slide footer for further contact information.

`http://richardhartmann.de/talks/`

#vcs-home @ irc.oftc.net
vcs-home@lists.madduck.net