

# Version Control Systems for Astronomy and Astrophysics

James Tocknell

September 20, 2013

# Outline

- 1 Introduction to Version Control
- 2 Mercurial
  - Basics
  - Further hg
- 3 Git

# Brief History

- 1970's diff created, SCCS (early VCS) created
- 1982 RCS created (first mainstream VCS)
- 1986 CVS created
- 2000 Subversion created by some of CVS developers; Bitkeeper launched
- 2001 GNU arch created, which would evolve into GNU Bazaar
- 2005 Bitkeeper controversy, Git, Mercurial created

# Why hg or git

- Distributed:
  - Fast
  - Work anywhere
  - Easily manage alternate versions
- Open Source
- Interface with Older/Propriety VCS (CVS/SVN/Perforce)
- Easy to host/find hosting
- Not funded by Canonical ;) (bzd)

## Difference between hg and git

- hg written in Python and C, git is C + unix + others
- hg first to be ported to windows, git and windows is “fun”
- Branching
- hg config via text files, git via commands (by default)
- hg uses extensions, git is scriptable (in a unix way)
- hg is more black box than git
- git has more interesting and unique tools built off it (git-annex, vcsh, bup)
- hg developed by VCS developers, git developed by kernel/filesystem developers

# Outline

- 1 Introduction to Version Control
- 2 Mercurial  
Basics**  
Further hg
- 3 Git

# Setup

- .hgrc
  - Username and Password
  - merge tool
  - Extensions (if we get to it)

- `export EDITOR=editor`

- `hg help {command}`

is useful

repo contains:

- .hg directory (magical apart from .hg/hgrc, which is per repo config)
- working directory, the directory containing .hg directory

## hg init (clone)

```
hg init
# new hg repo in current dir
hg init project
# new hg repo in project dir (doesn't have to exist)

hg clone ssh://path/to/project
# copy project via ssh into project dir
hg clone https://path/to/project
# clone via https (can also use http)
hg clone /path/to/project
# clone repo on local machine
hg clone /path/to/project new-project
# same as before but clone into new-project dir
```



# hg add(remove)

```
hg add paper.tex
# start tracking paper.tex to repo
hg remove paper.tex
# stop tracking paper.tex
hg addremove
# add all untracked files and stop tracking
# non-existent files
```

## hg diff (vdiff)

Show changes (by default difference between last commit and working dir)

Show diff via tool using new command vdiff

```
[extensions]
```

```
  hgext.extdiff =
```

```
[extdiff]
```

```
  cmd.vdiff = meld
```

# hg commit

```
hg commit
# stores current state of repo with a message
# created using a text editor
hg commit -m "A silly message"
# stores current state of repo with message
# "A silly message"
hg commit paper.tex
# stores the current state of file paper.tex
```

## hg commit

In hg, commits have an id (hash), number (will be different for different people), committing user, commit time, branch (default by default), and one or more tags.

See [http:](http://who-t.blogspot.com.au/2009/12/on-commit-messages.html)

[//who-t.blogspot.com.au/2009/12/on-commit-messages.html](http://who-t.blogspot.com.au/2009/12/on-commit-messages.html)

and <http://tbagery.com/2008/04/19/>

[a-note-about-git-commit-messages.html](http://tbagery.com/2008/04/19/a-note-about-git-commit-messages.html) for how to write commit messages.

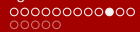
## hg update (revert)

```
hg update
# move repo to latest commit on current branch
hg update -r 42
# move repo to commit 42
hg revert paper.tex
# remove uncommitted changes to paper.tex
```

## Aside: Branching

There are 4 types of branch in hg:

- 1 Anonymous heads
- 2 Bookmarks (git-like branches)
- 3 Named branches
- 4 Forks (not really a branch but how some VCSs do branching, see <http://stevelosh.com/blog/2009/08/a-guide-to-branching-in-mercurial/>)



# hg merge

# Aside: SSH Keys





# hg push/pull



# Outline

- 1 Introduction to Version Control
- 2 Mercurial**
  - Basics
  - Further hg
- 3 Git



# hg log (glog)



# hg bisect



# hg tag



# Extensions



# Setup