

GIT

The distributed VCS

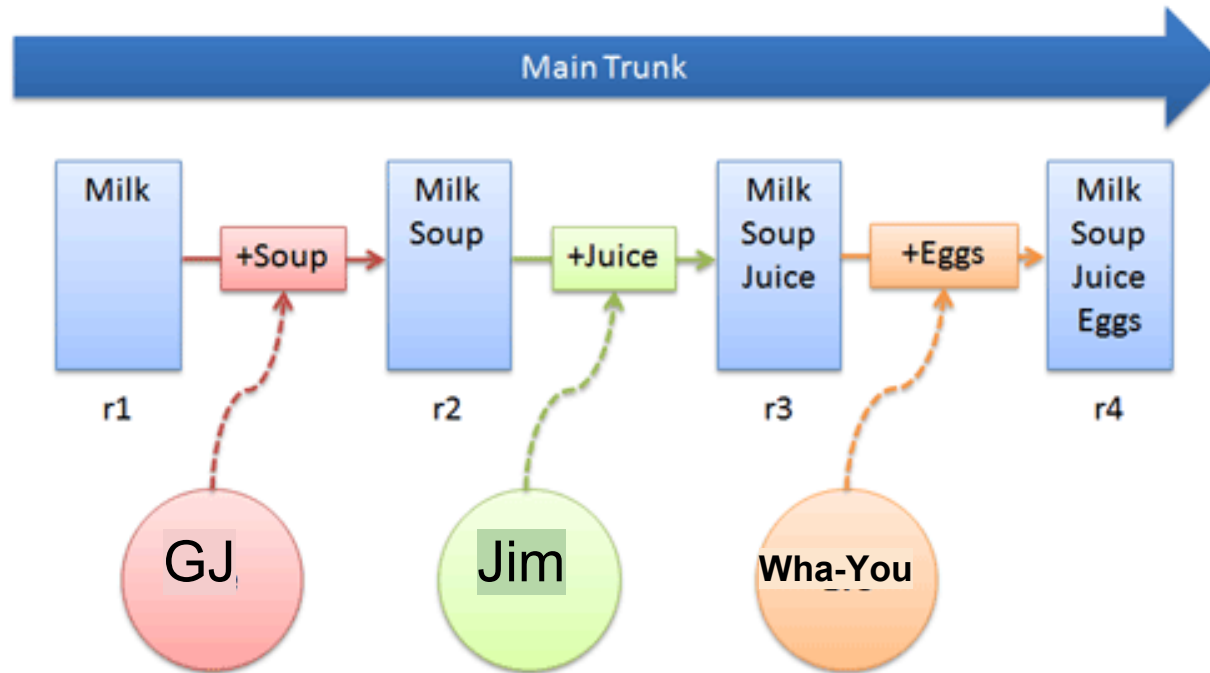


by

Shauvik Roy Choudhary

Distributed? Whats wrong with Centralized?

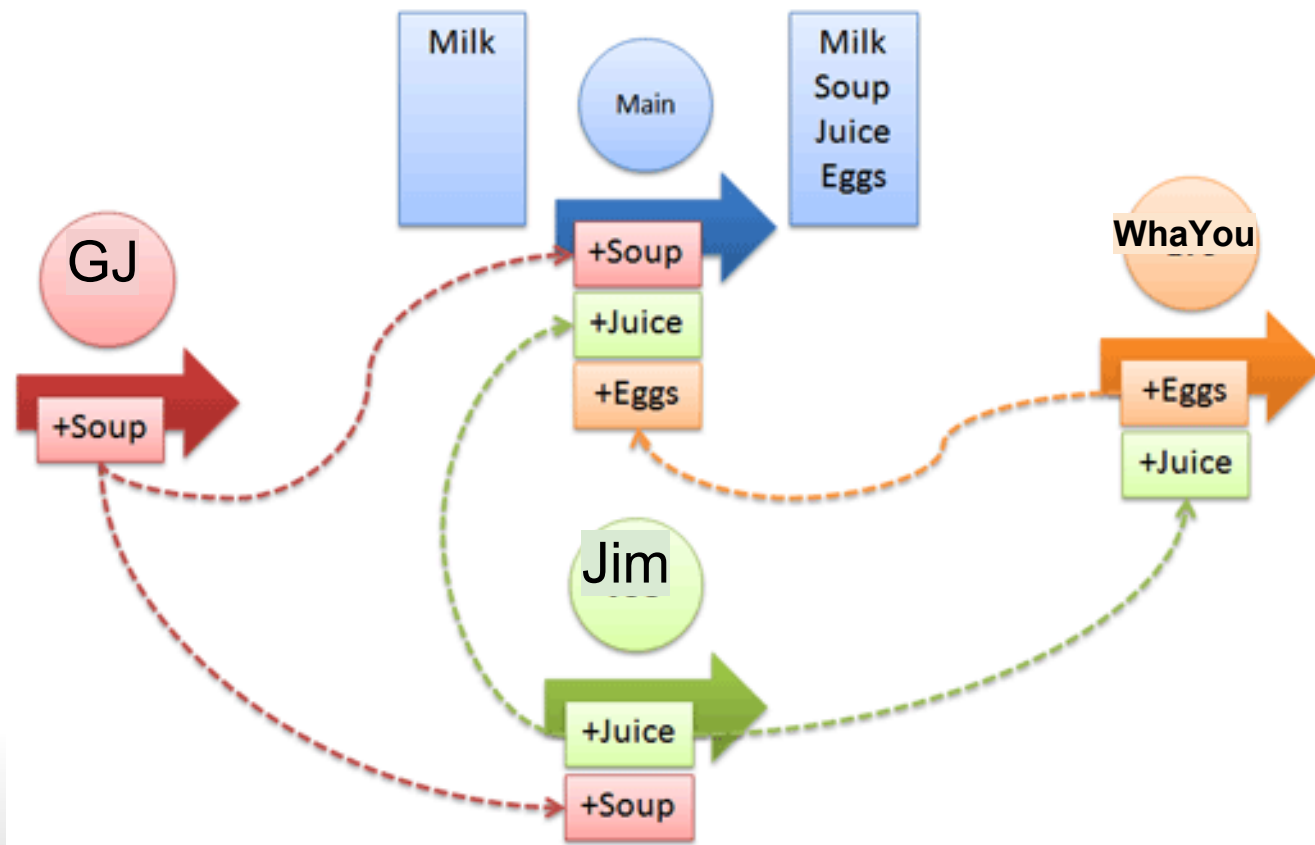
Centralized VCS



Works for **backup, undo and synchronization**
But isn't great for **merging and branching** !

Functioning of a Distributed VCS

Distributed VCS



But will it be a circus with no ringleader? **NOPES !!!**
Everyone can push changes into a common repo.

Key Concepts

Core Concepts

- Centralized VCS focuses on **synchronizing, tracking, and backing up files.**
- Distributed VCS focuses on **sharing changes**; every change has a guid or unique id.
- **Recording/Downloading** and **applying** a change are separate steps (in a centralized system, they happen together).
- **Distributed systems have no forced structure.** You can create “centrally administered” locations or keep everyone as peers.

New Terminology

- **push**: send a change to another repository (may require permission)
- **pull**: grab a change from a repository

Key Advantages

- **Everyone has a local sandbox.** You can make changes and roll back, all on your local machine. No more giant checkins; your incremental history is in your repo.
- **It works offline.** You only need to be online to share changes.
- **It's fast.** Diffs, commits and reverts are all done locally. There's no shaky network or server to ask for old revisions from a year ago.
- **It handles changes well.** Distributed version control systems were *built* around sharing changes. Every change has a guid which makes it easy to track.
- **Branching and merging is easy.** Because every developer “has their own branch”, every shared change is like reverse integration. But the guids make it easy to automatically combine changes and avoid duplicates.
- **Less management.** Distributed VCSes are easy to get running; there's no “always-running” server software to install. Also, DVCSES may not require you to “add” new users; you just pick what URLs to pull from. This can avoid political headaches in large projects.

Key Disadvantages

- **You still need a backup.** Some claim your “backup” is the other machines that have your changes. With a DVCS, you still want a machine to push changes to “just in case”.
- **There’s not really a “latest version”.** If there’s no central location, you don’t immediately know whether to see GJ, Jim or Wha-You for the latest version. Again, a central location helps clarify what the latest “stable” release is.
- **There aren’t really revision numbers.** Every repo has its own revision numbers depending on the changes. Instead, people refer to change numbers: *Pardon me, do you have change fa33e7b?* (Remember, the id is an ugly guid). Thankfully, you can tag releases with meaningful names.

GIT commands

Calling convention: "git foo" OR "git-foo"

Documentation: "man git-foo" OR "git help foo"

Introduce yourself to GIT

```
$ git config --global user.name "Shauvik Roy Choudhary"  
$ git config --global user.email shauvik@cc.gatech.edu
```

Importing a new project

```
$ tar xzf project.tar.gz  
$ cd project  
$ git init
```

Add files to index and commit

```
$ git add .  
$ git commit
```

or "git commit -a"

Demo

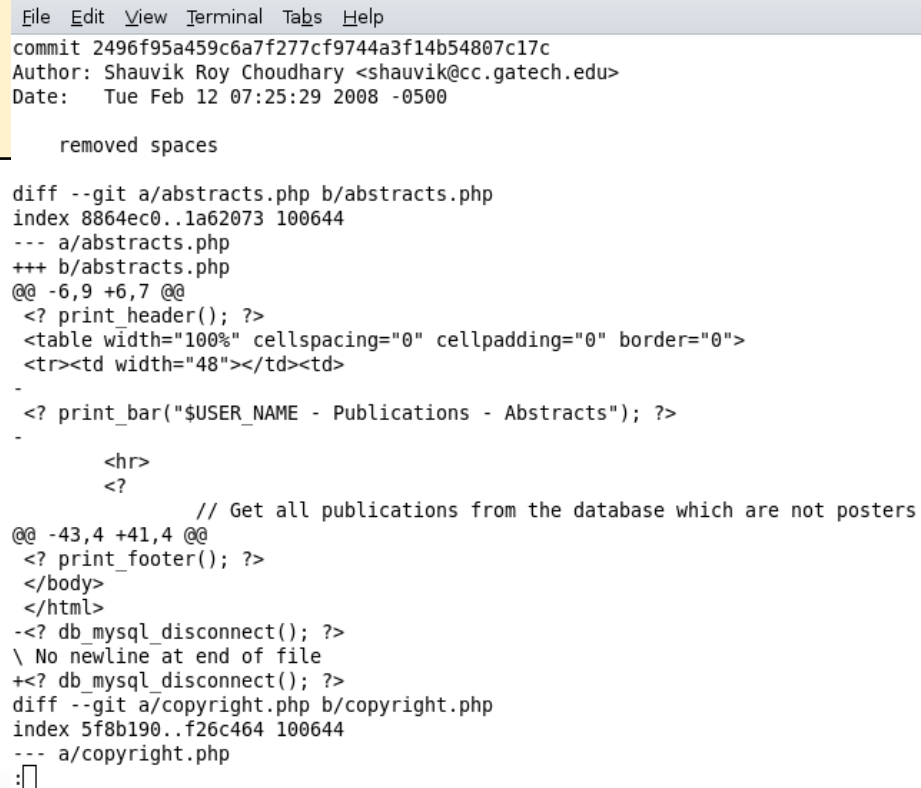
```
File Edit View Terminal Tabs Help
shauvik@cheetah:~$ cd GIT/piproj/
shauvik@cheetah:~/GIT/piproj$ ls
abstracts.php  css          include      publickey.php  teaching.php
admin          CVS         index.php    research.php
AlexOrso.vcf  DB          papers       service.php
bib.php       download.php pictures      software.php
copyright.php funding.php  publications.php students.php
shauvik@cheetah:~/GIT/piproj$ git init
Initialized empty Git repository in .git/
shauvik@cheetah:~/GIT/piproj$ git add .
shauvik@cheetah:~/GIT/piproj$ git commit
Created initial commit aa30731: Initial Version of Personal Information Manager
project obtained from Alex
170 files changed, 8908 insertions(+), 3 deletions(-)
create mode 100644 .cvsignore
create mode 100644 AlexOrso.vcf
create mode 100644 CVS/Entries
create mode 100644 CVS/Entries.Extra
create mode 100644 CVS/Entries.Extra.01
create mode 100644 CVS/Entries.Old
create mode 100644 CVS/Repository
create mode 100644 CVS/Root
create mode 100644 CVS/Template
create mode 100644 DB/CSV/Entries
create mode 100644 DB/CSV/Entries.Extra
create mode 100644 DB/CSV/Entries.Extra.01
create mode 100644 DB/CSV/Entries.Old
create mode 100644 DB/CSV/Repository
shauvik@cheetah:~/GIT/piproj/.git$ ls -l
total 52
drwxr-xr-x  2 shauvik shauvik 4096 2008-02-12 06:28 branches
-rw-r--r--  1 shauvik shauvik  92 2008-02-12 06:28 config
-rw-r--r--  1 shauvik shauvik  58 2008-02-12 06:28 description
-rw-r--r--  1 shauvik shauvik  23 2008-02-12 06:28 HEAD
drwxr-xr-x  2 shauvik shauvik 4096 2008-02-12 06:28 hooks
-rw-r--r--  1 shauvik shauvik 17256 2008-02-12 06:29 index
drwxr-xr-x  2 shauvik shauvik 4096 2008-02-12 06:28 info
drwxr-xr-x 125 shauvik shauvik 4096 2008-02-12 06:29 objects
drwxr-xr-x  4 shauvik shauvik 4096 2008-02-12 06:28 refs
shauvik@cheetah:~/GIT/piproj/.git$
```

Making changes

Add files to index and diff with a --cached flag

```
$ git add file1 file2 file3  
$ git diff --cached
```

without --cached, it gives the diffs of all files changed since last commit



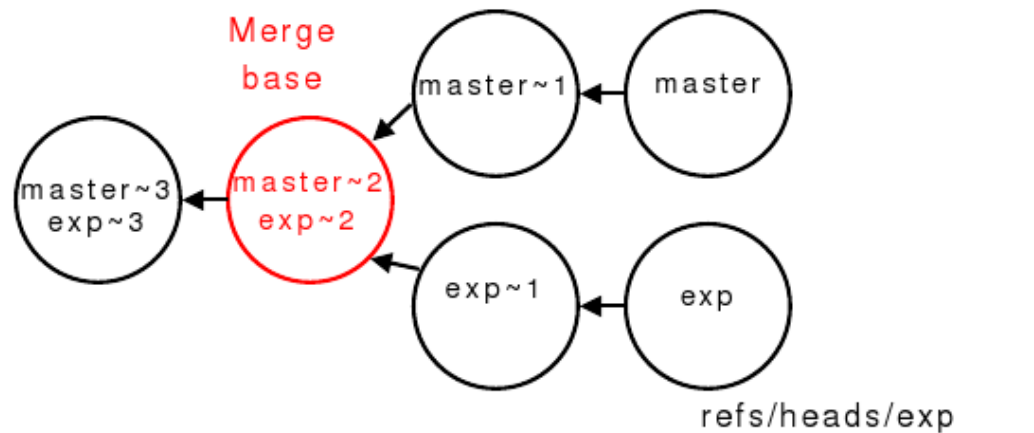
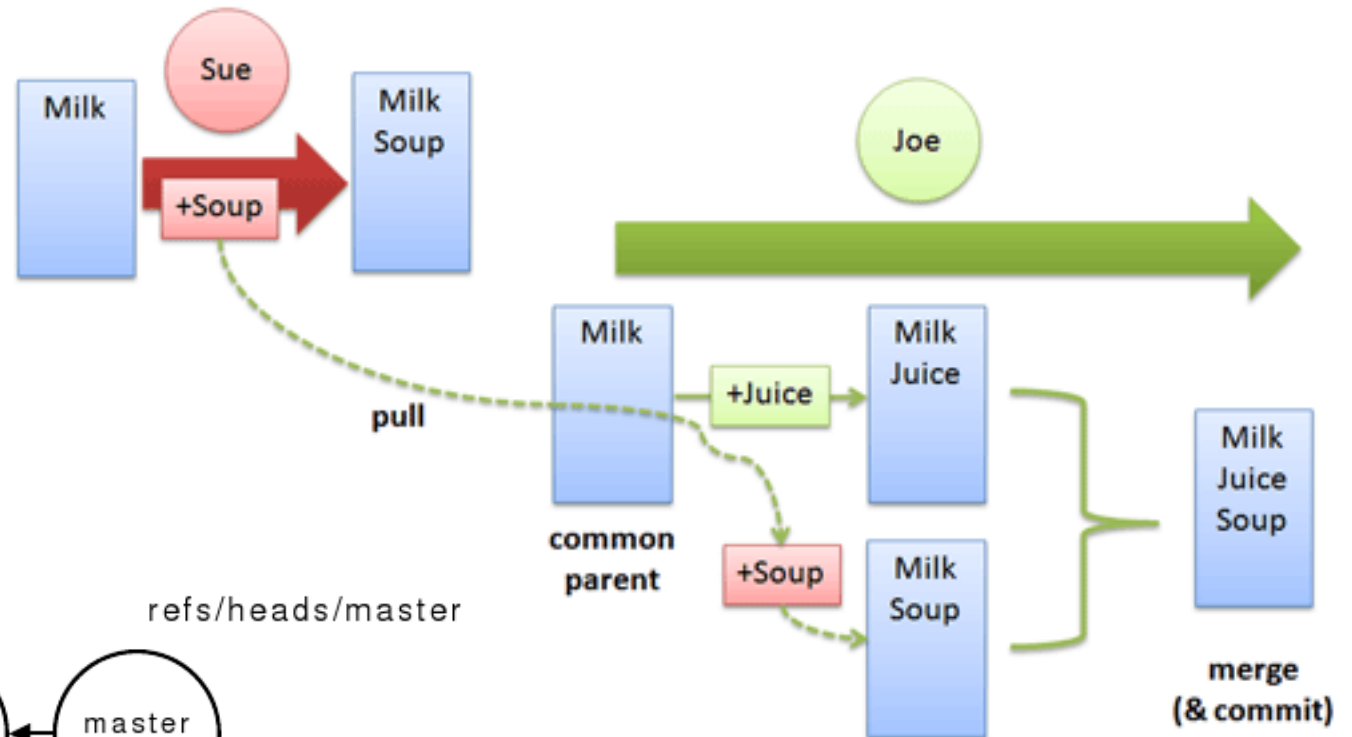
```
File Edit View Terminal Tabs Help  
commit 2496f95a459c6a7f277cf9744a3f14b54807c17c  
Author: Shauvik Roy Choudhary <shauvik@cc.gatech.edu>  
Date: Tue Feb 12 07:25:29 2008 -0500  
  
removed spaces  
  
diff --git a/abstracts.php b/abstracts.php  
index 8864ec0..1a62073 100644  
--- a/abstracts.php  
+++ b/abstracts.php  
@@ -6,9 +6,7 @@  
 <? print_header(); ?>  
 <table width="100%" cellpadding="0" cellspacing="0" border="0">  
 <tr><td width="48"></td><td>  
 -  
 <? print_bar("$USER_NAME - Publications - Abstracts"); ?>  
 -  
 <hr>  
 <?  
 // Get all publications from the database which are not posters  
@@ -43,4 +41,4 @@  
 <? print_footer(); ?>  
 </body>  
 </html>  
-<? db_mysql_disconnect(); ?>  
 \ No newline at end of file  
+<? db_mysql_disconnect(); ?>  
diff --git a/copyright.php b/copyright.php  
index 5f8b190..f26c464 100644  
--- a/copyright.php  
:~
```

Viewing project history

```
$ git log  
$ git log -p  
$ git log --stat --summary
```

Branching and Merging

Merging Changes



Managing branches

```
$ git branch experimental
```

creates a new branch

```
$ git branch  
experimental
```

```
* master
```

```
$ git checkout experimental
```

```
-- (edit file) --
```

```
$ git commit -a
```

```
$ git checkout master
```

```
-- (edit file) --
```



changes made in experimental not seen

```
$ git commit -a
```

```
$ git merge experimental
```

```
$ git diff
```

```
$ git commit -a
```

```
$ gitk
```

```
$ git branch -d experimental
```

requires merge

```
$ git branch -D crazy-idea
```

doesn't require merge

Using GIT for collaboration

```
bob$ git clone /home/alice/project myrepo
-- (edit files) --
bob$ git commit -a
-- (repeat as necessary) --
```

```
alice$ cd /home/alice/project
alice$ git pull /home/bob/myrepo master
```

master is optional

OR

```
alice$ git remote add bob /home/bob/myrepo
alice$ git fetch bob
```

```
alice$ git log -p master..bob/master
alice$ git merge bob/master OR "git pull . remotes/bob/master"
```

```
bob$ git pull
```

More...

Bob from another host

```
$ git clone alice.org:/home/alice/project myrepo
```

Alternatively, git has a native protocol, or can use rsync or http

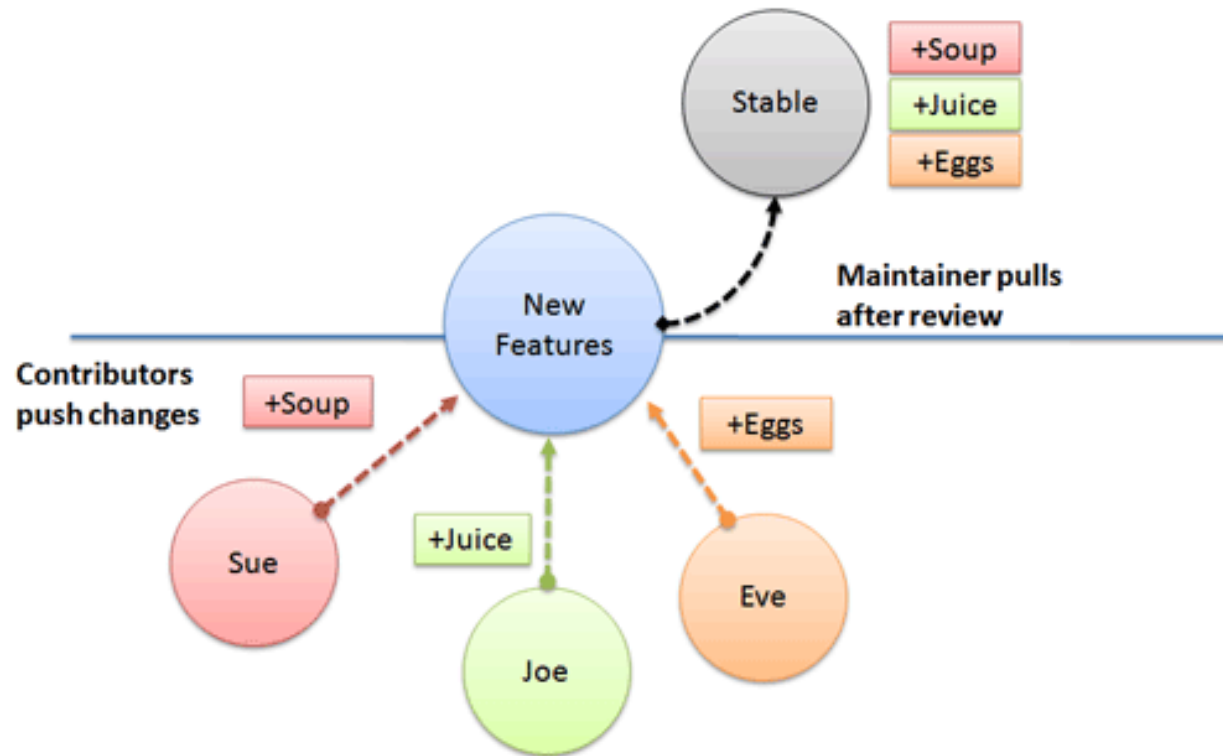
CVS-like mode, with a central repository that various users push changes to:

```
$ git clone foo.com:/pub/repo.git/ my-project  
$ cd my-project  
$ git pull origin  
  -- (make changes) --  
$ git push origin master
```

Organizing a Distributed Project

An Example

Distributed Push/Pull Model



Developers check changes into a common branch and trade patches with each other to do “**Buddy builds**”

Maintainer reviews & pulls changes from the experimental branch to stable branch.

Exploring History

```
$ git log
commit c82a22c39cbc32576f64f5c6b3f24b99ea8149c7
Author: Shauvik Roy Choudhary <shauvik@cc.gatech.edu>
Date: Tue May 16 17:18:22 2008 -0700
    merge-base: Clarify the comments on post processing.
```

See details about the commit

```
$ git show c82a22c39cbc32576f64f5c6b3f24b99ea8149c7
```

```
$ git show c82a22c39c # the first few characters are usually enough
$ git show HEAD      # the tip of the current branch
$ git show experimental # the tip of the "experimental" branch
$ git show HEAD^    # to see the parent of HEAD
$ git show HEAD^^   # to see the grandparent of HEAD
$ git show HEAD~4   # to see the great-great grandparent of HEAD
```

Some more commands

Tag a commit

```
$ git-tag v2.5 1b2e1d63ff
```

Search in a version of project

```
$ git grep "hello" v2.5
```

Doing patches

```
git-format-patch  
git-am
```

Perform binary search to track bugs

```
git-bisect
```

References & Tutorials

1. GIT tutorial

<http://www.kernel.org/pub/software/scm/git/docs/tutorial.html>

2. GIT for CVS users

<http://www.kernel.org/pub/software/scm/git/docs/cvs-migration.html>

3. Everyday GIT with 20 commands or so

<http://www.kernel.org/pub/software/scm/git/docs/everyday.html>

4. Intro to Distributed VCS (illustrated)

<http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/>

5. What is this GIT thing - by Jeff King