

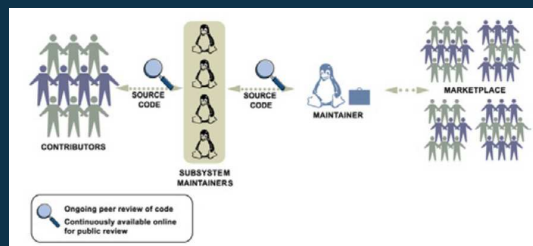
## Git and the Linux Kernel

Dave Hansen  
d@vehansen.com

## Background

- Dave
  - BS in CS at Purdue, hired in to IBM's Linux Technology Center in 2001 right out of college
- Nish
  - BS CS/BA Philosophy at University of Illinois, started at IBM in 2004
- Linux Community (to me) - broad collection of folks with varied interests in Linux: hobbies, cell phones, supercomputers
- We work *for* IBM, but *in* and *with* the Linux Community
  - Contribute to and help steer the direction of the Community

## Kernel Development



## Problem: Development Bottlenecks

- 2.6.35: 8,000 lines of code added per *DAY!*
- All code was moved with email, patches and tarballs
- Linus's email became the bottleneck to development
  - "Linus doesn't scale!"
- Solution: BitKeeper (free, but proprietary, so a few rules)
  - Code logs must be public
  - Do not reverse engineer
  - Code could be moved with BitKeeper!
- Problem solved!
  - ... until someone reverse engineered it!

1. It's debatable whether it was truly reverse engineered

## New Problem: Need an Open Bitkeeper

- Linus sat down, thought about some of the ideas he learned about in BitKeeper, and implemented git
1. Take CVS as an example of what *not* to do; if in doubt, make the exact opposite decision.
  2. Support a distributed, BitKeeper-like workflow.
  3. Very strong safeguards against corruption, either accidental or malicious.
  4. Very high performance.

1. Wikipedia Git Page

## Centralized Source Control

- One central repository
- Server runs special software which coordinates versions, handles over-the-wire transport, users, etc...
- Recording your edits requires that you write to the server
  - Can be done in to branches, but still requires permissions and access on the server
  - Multiple projects mean multiple branches
- Lots of code checkouts stress the central server
- Server goes down? Wifi not working? Too bad!

## Distributed Version Control

- Distributed != Clustered
- Distributed means decentralized
- Either:
  - There is no server
  - Everyone is the server
- You only talk to a "server" when code changes hands
- You can *always* edit and record your changes
  - Since there is no server, there is no one to approve or disapprove of your branches
- "Server" goes down?
  - Makes it harder to exchange code, but everyone can still work
- Dave Hansen's copy is the same as Linus Torvalds' copy

## Git Implementation

- Blobs
  - Random data, compressed, and cryptographically summed (SHA1)
  - Written to a file corresponding to the SHA1
  - `.git/objects/ab/cdef123...`
- Trees
  - Matches files to Blobs
  - `foo/bar.c -> .git/objects/ab/cdefg123...`
  - `foo/baz.c -> .git/objects/01/abcd654...`
- Commits
  - Parent commit(s)
  - Tree
  - Blob for a description

## Git Performance

- Biggest Boost: built to run on Linux
  - (its designer knows about Linux a bit)
- Linux has very fast system calls, and very aggressive in-memory caches of filesystem metadata and contents
  - So git has none
- Reduce sheer numbers of files with "packfiles"
  - Reading huge file is faster than many little ones
- Packfile objects stored with delta compression
  - Very nice for incremental source code changes
- Every Repository has full, shared history
  - Easy to find differences between arbitrary repositories
  - Easy to minimize the amounts of data over the wire
  - Helps figure out who to blame

## Git Protection

- Protection from accidental (bad disks) or malicious (bad people) corruption
- SHA1's of objects, but also trees and commits
- Commit SHA1: made up of parent(s) SHA1, plus self
  - Change one commit, you change entire history
- Distribution helps!
  - Since everyone has the same tree, even if the entire history got changed, people would see the wide delta when they synchronize
- This was tested in practice, kernel.org was breached
  - It was easy to move "servers" during the repair
  - Folks could verify that no trees had been modified

## Powerful History

- April 16, 2005 -> January 18, 2012
  - 286,217 Commits
  - Every bit of history, every version of every file, every description of every change
  - Remember, ~8,000 lines of code/day
  - 536MB
- gzip'd tarball of version 3.2: 94MB

## Examples

- `git log`
- `gitk` (notice the directed acyclic graph)
- `gitweb`
- `git blame`
- `git bisect`

## Everyday Use

- No longer any need to keep multiple versions of the kernel
- Git "server" is unnecessary
  - read-only access over http is common
  - Writes to repositories done via ssh, so entire classes of security vulnerabilities are not possible, nor do you need users other than UNIX users
- My perl and shell scripts in ~/bin
- Local, persistent changes to a project
  - "I want this button to be blue instead of red"

## Git Users

- Git
- Linux Kernel
- Perl
- Eclipse
- Gnome
- KDE
- Qt
- Ruby on Rails
- Android
- PostgreSQL
- Debian
- [X.org](#)
- Thousands of other projects...

Source: [http://git-scm.com/front\\_page](http://git-scm.com/front_page)

## Capstone Project

- Automated tools to find bugs in source code (sparse)
- Rich, full history
- Most developers fail to run the tools
- Use the git history to locate problematic points in the code and try to determine who is responsible

## Links

<http://excess.org/article/2008/07/ogre-git-tutorial/>

Questions?