



Brandon Philips  
brandon@ifup.org

# What is GIT?

---

## The Acronym

- pronounceable three-letter combination
- "global information tracker"
- "goddamn idiotic truckload of sh\*t"

## The Tool

- Directory tracker
- Source code management system
- Distributed development
- Disc instead of CPU tradeoff
- Unixy (111 small utilities)
- FAST

# Workflow of a Developer

---

- Edit/Test/Compile
- Commit to a repository on their workstation
- "Fetch" repo from another developer
- "Merge" from one branch into another
- "Push" repo to a public repository

# **shell.onid.oregonstate.edu**

---

**Are you getting ^? instead of backspace?  
Add 'stty erase ^h' to your shell rc**

# Starting a Simple Project

---

```
$ cd project  
(1) $ git init-db  
(2) $ git add .  
(3) $ git commit  
(4) $ vimacs new_file  
(5) $ git update-index --add new_file  
(6) $ git commit
```

- (1) Initialize the .git directory and object store**
- (2) Add all existing files**
- (3) Commit all existing files**
- (4) Edit a new file**
- (5) Update the "index" with new file contents**
- (6) Commit this new index**

# The Index

---

## Index? I am not writing a book.

- Git uses an index file to hold the state of the tree
- git-update-index does a few things
  - Saves specified files into the object store
  - Saves the state of the tree into .git/index

## Working the Index

- Prepare commits in the index
- Commit the index with git-commit

# The Object Database

---

## git is a content addressable filesystem

- each object named by a SHA-1 hash of contents
- blob, tree, commit, tag types

## Playing with Objects

```
$ ls .git/objects/??/*  
.git/objects/89/b4e86adcc34ffcd176df5fda4ffc90d01474c1  
...
```

```
(1) $ git-cat-file -t 89b4e86adcc34ffcd176df5fda4ffc90d01474c1  
blob
```

```
(2) $ git-cat-file blob 89b4e86adcc34ffcd176df5fda4ffc90d01474c1  
<file contents>
```

**(1) find out the type of an object**

**(2) print out the contents of the file**

# Object Types

---

- Blob
  - The contents of a file
- Tree - represents a directory
  - Holds list of file names w/ ref to object
- Commit
  - The SHA-1 of the top tree object
  - Parents, committer, author, message and date
- Tag
  - Ref to a commit, a message and optional PGP sig

# Keeping it Clean

---

(1) \$ git fsck-objects

\$ git prune

(2) \$ git count-objects (2)

(3) \$ git repack (3)

(4) \$ git prune (4)

**(1) assures the repository health reasonably well.**

**(2) how many loose objects? wasted space?**

**(3) repack loose objects incrementally**

**(4) after repack, prune removes the duplicate loose objects.**

# Whats the Diff?

---

**With the index there are two different places to diff**

- (1) \$ git diff
- (2) \$ git diff HEAD
- (3) \$ git diff HEAD^ HEAD
- (4) \$ git diff HEAD^^ HEAD^

- (1) diff between cwd and index**
- (2) diff between cwd and the HEAD**
- (3) diff between previous HEAD and HEAD**
- (4) ???**

# Branches, Branches, Branches, Branches

---

- "master" is the home of the mainline of development
  - git-branch
- other branches can be used for a variety of uses
  - Trying out a crazy idea
  - Tackling a bug report
  - Working w/ other developers (later)

# Climbing with Branches

---

- (1) \$ git branch crazyidea  
    \$ git checkout crazyidea
- (2) \$ git branch  
    \* crazyidea  
    master  
    edit/compile/update-index/commit
- (3) \$ git checkout master
- (4) \$ git pull . crazyidea

- (1) create and checkout crazyidea**
- (2) list all branches**
- (3) checkout the master branch again**
- (4) pull changes into master and commit**

# Pushin', Fetchin' and Mergin'

---

## **Problem**

- repository sitting in private directory

## **Solution**

- git-push

## **Problem**

- repository on public server not your workstation

## **Solution**

- git-fetch

# Pushin'

---

```
$ cd ~/public_html  
(1) $ GIT_DIR="project.git" git init-db  
    $ cd ~/project  
(2) $ git update-server-info  
(3) $ git push $HOME/public_html/project.git master:master
```

- (1) Initialize a git db called project.git**
- (2) "Update auxiliary info file to help dumb servers"**
- (3) Push branch "master" to your public projects "master" branch**

# Pushin' with Remotes

---

## Problem

- Typing out the entire git push command sucks

## Solution

- `.git/remotes/`

## Create a remotes file for your public\_html

```
cat > .git/remotes/public
```

```
URL: /users/u2/p/philipsb/public_html/project.git
```

```
Push: master:master
```

## Try it out!

```
git push public
```

**w00t!**

# Fetchin'

---

```
cat > .git/remotes/bob
```

```
URL: git://example.org/bob/
```

```
Push: master:bob-incoming
```

```
(1) $ git fetch bob
```

```
(2) $ git whatchanged -p master..bob-incoming
```

```
(3) $ git pull . bob-incoming
```

```
(1) fetch bob's work into a bob-incoming branch
```

```
(2) view a log and patches between bob-incoming
```

```
(3) merge changes with bob-incoming
```

# Moving from SVN and CVS

---

## **cvsimport**

- Converts all branches of a module
- Incremental, track CVS projects via git

```
$ git cvsimport -v -d <cvsroot> <module>
```

## **svnimport**

- Converts "trunk, tags, branches" style
- Incremental, track SVN projects via git

```
$ git svnimport <URL>
```

# Cool git tools

---

## **git.cgi**

- Easy to setup CGI Perl script
- ex. <http://kernel.org/git/>

## **gitk**

- GUI for viewing commits and branches
- Preview a tree before an operation

## **git-cvsserver - new in 1.3.0**

- Interop w/ CVS clients

# Running a git Server

---

## **git-daemon**

- Simple read only TCP daemon

## **git-shell**

- Restricted shell for ssh push access