Git

Pau Garcia i Quiles <pgquiles@elpauer.org>

Slides: http://www.elpauer.org/stuff/git.pdf

What is a (D)VCS

- (Distributed) Version Control System
- Management of multiple revisions of the same unit of information (for instance, files)
- Very useful in software development:
  - Know who and when changed what
  - Be able to go back, in case you screwed something
- VCS vs DVCS
  - Central repository vs distributed (no central) repository

Architecture

Created by Linus Torvals in April 2005 for the Linux kernel

Maintained by Junio Hamano since July 2005

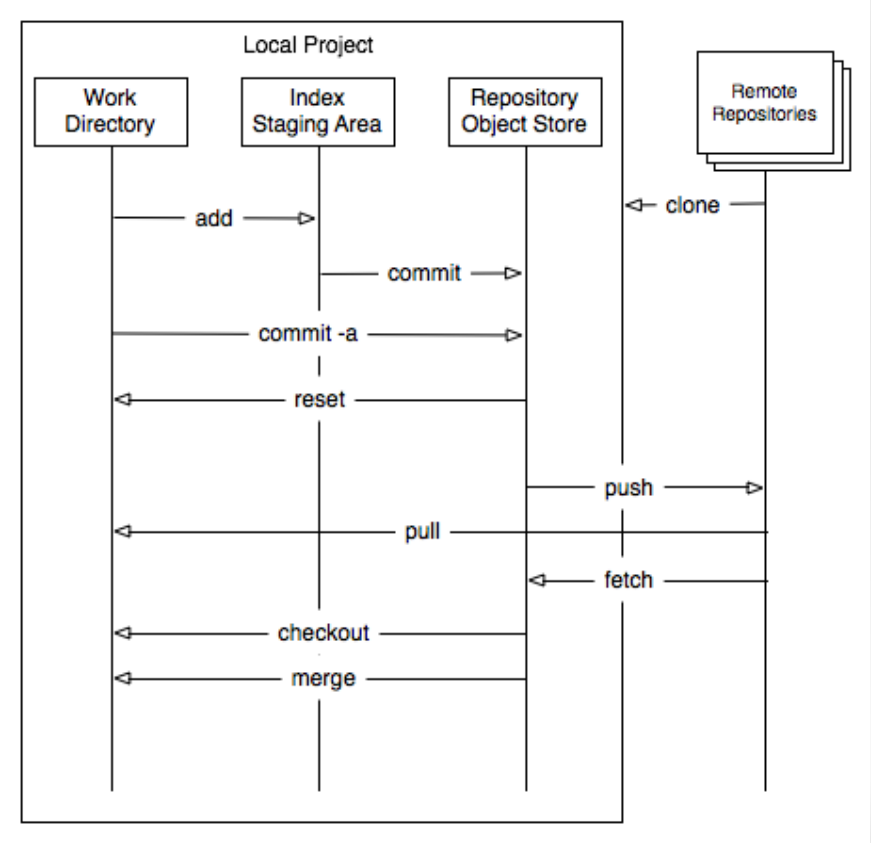
Each command is a different program (porcelain vs plumbing)

Originally: a few C programs, lots of shell-scripts around them

Now: shell-scripts mostly replaced by C programs

Future: libgit2 library + exactly 1 program

Overview



Repository

- Repository
  - Your stuff, managed by git
  - There is a **single** '.git' directory per repository, at the root of the repository
- Working tree
  - The repository at a particular moment in time (a “snapshot” of the repository)
- Index (AKA “staging area”)
  - Kind of a “pre-filter”: you must make git aware of your changes or your changes will not be “committed” by “git commit”

Config

```
$ git config user.name "FirstName LastName"
$ git config user.email "user@example.com"
$ git config --global color.branch "auto"
$ git config --global color.status "auto"
$ git config --global color.diff "auto"
$ git config --global pack.threads "0"
```

Workflow

```
$ git init
$ vi file.cpp
$ git add main.cpp
$ git commit -m “Initial import”
$ vi main.cpp
$ git commit -m “I'm screwing it”
$ vi main.cpp
$ git commit --amend
```

…

SHA-1

No sequentian revision numbers

Commits in git are identified by a SHA-1 hash

You can go back to a “revision” by checking out that SHA-1:

git checkout

Reflog

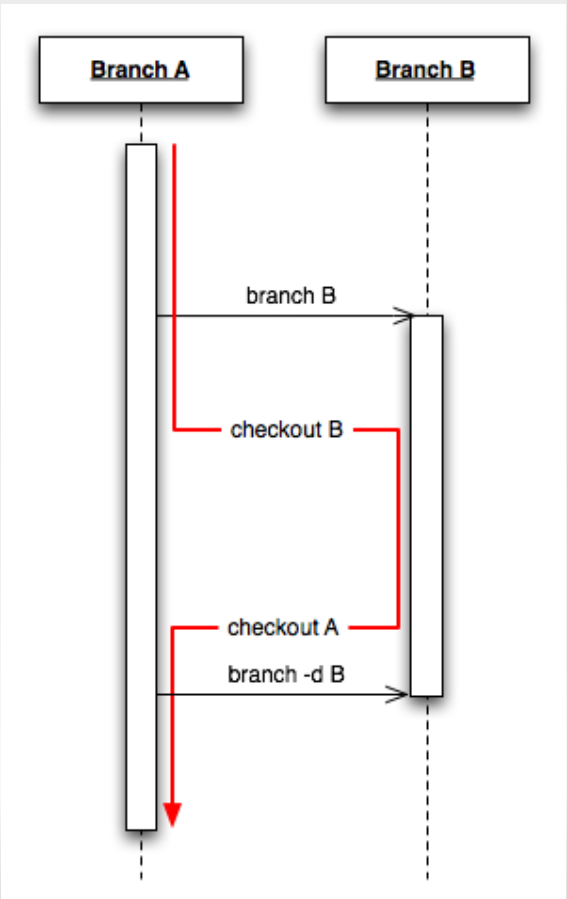
Shows actions done to your repository:

```
$ git reflog
78c80b7... HEAD@{0}: pull : Fast forward
c516234... HEAD@{1}: checkout: moving to master
c516234... HEAD@{2}: pull : Fast forward
01fdb2a... HEAD@{3}: rebase: mplayer: adapt configure
  options to latest svn
da8ed38... HEAD@{4}: rebase
```

Branch

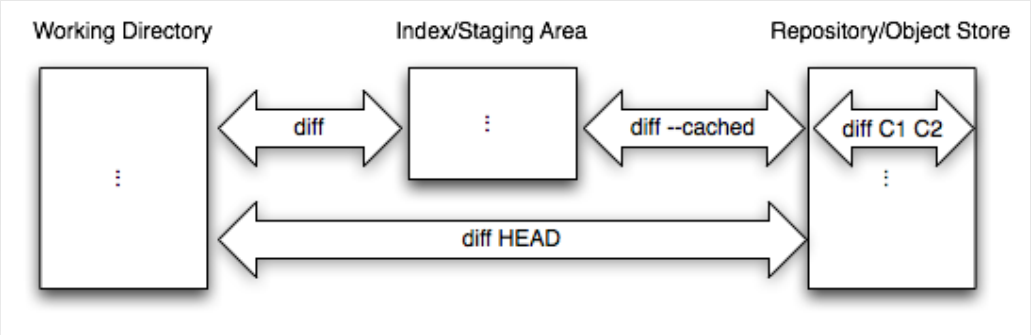
git branch ...

git checkout ...



Diff

More or less like GNU diff



git diff: changes in the working tree relative to the index

git diff –cached: changes in the index relative to the repository

git diff HEAD: changes in the working tree relative to the repository

Fetch & Merge

Get changes from another repository:

$ git fetch ...

But those changes are NOT merged locally!

Merge them:

$ git merge ...

There is a convenience command for this:

git pull ≈ git fetch + git merge

Revert/reset

Make last 2 commits disappear but do not modify HEAD:

$ git [–mixed] reset HEAD~2

Scrap uncommited changes:

$ git reset –hard

Set HEAD to three commits ago:

$ git reset –soft HEAD~3

Tag

It does the WRONG thing by default:

$ git tag 1.0 # Creates a “lightweight” tag (≈ pointer)

What you really want to do is:

$ git tag -a 1.0 # Annotated tag

or

$ git tag -s 1.0 # Signed tag

Clone

Copy a remote repository into your machine

FULL copy, including history (unlike CVS, SVN...)

$ git clone git://gitorious.org/teamgit/mainline.git

There is no difference between a cloned repository and the “original” repository

NO partial clones (i.e. NO svn checkout svn://repo/dir/subdir/)

Push

Send your changes to a remote repository

$ git push

BEWARE! 'commit' does not imply 'push'!!! It's not like 'svn commit'!!!

svn commit ≈ git commit + git push

Rebase

Forward port local commits to the updated upstream head

Done incrementally (i. e. commit by commit)

Branch development made possible! (svn branch in Subversion < 1.5 was nearly useless, as merging back was very difficult and time-consuming due to conflics)

Stash

Saves your work temporarily (you can go back at any time)

Very useful if you have local changes and you want to go forward (rebase) or go back (checkout an old commit)

```
$ git stash
$ git rebase
$ git stash apply
$ git stash clear
```

```
$ git stash
$ git checkout HEAD~5
$ git stash apply
$ git stash clear
```

Submodules

More or less like svn externals

FULL OF CRAP

There is no good replacement to svn externals in git

Bisect

Find the change that introduced a bug by binary search

Mark “good” commits and “bad” commits iteratively:

It's mostly a helper around “git checkout”

Bisect

$ git bisect start

$ git bisect bad # Currently HEAD is bad

$ git bisect good 0cadf32 # Tell git rev 0cadf32 was good and checkout the “middle point” between HEAD and 0cadf32

(Build & check)

$ git bisect bad # Tell git that middle point was bad and checkout the middle point between it and 0cadf32

(Build & check)

$ git bisect good

...

Cherry pick

Take a commit (usually from a different branch), create a patch (temporary) and apply it to the current branch

Very useful if you want to merge selectively from other branches

Interacting with other (D)VCS

Subversion

CVS

Darcs

Perforce

fast-import / fast-export

Other

grep

send email

blame

show

describe

rm, mv, cp

lots more

Server-side

git daemon

git-instaweb

Hosting

repo.or.cz (FLOSS, small projects)

gitorious.org (FLOSS)

github.com (FLOSS/commercial)

unfuddle.com (FLOSS/commercial)

Many more

libQtGit

Generic library which allows you Qt application to use git

Uses:

- Create a GUI for git (TortoiseGit anyone?)
- Add versioning to your application
  - Versioned projects
  - Collaboration: send only changes
  - etc

License: LGPL 2.1 and 3

Take a look at the example

Questions

Ask me