Every day usage of Git
with/without Perforce

Dénes Mátételki

www.emerson.com

October 18, 2013

Table of contents

The scope

About this presentation

- Summarize main advantages of git over other version control systems (SVN, perforce).
- Present the most usual, every day process of working with git.
- Short summary of how git can work on a perforce system.
- No technical depth of git details (non-programmer friendly).

Short history of Git

History

- Initially designed and developed by Linus Torvalds for Linux kernel development in 2005.
- The name is a British English slang roughly equivalent to "unpleasant person".
- Free and open source. (GNU GPLv2)

Project goals

- Patches to take 3 seconds.
- Support a distributed workflow.
- Very strong safeguards against corruption.
- Performance.

Comparing Git to other VCSs

Advantages

- Distributed (offline work, everyone is a full backup).
- Very fast (work local files, advanced algorithms).
- Small (30x than Subversion).
- Lightweight branches (easy branching, merging).
- Flexible, great tools (There is more than one way to do it).

Disadvantages

- Difficult to learn due to complexity.
- Revisions don't have version numbers.
- ~~Lacks good GUI tools.~~

Every day git - collaboration

Workflow

- Update from remote, solve conflicts.
- Change local files.
- View changes.
- Adding them to a changeset.
- Create local commit.
- Submit local commits to remote.

Every day git 2 - lone wolf

Workflow

- Turn a directory into a Git repository.
- Change local files.
- View changes.
- Adding them to a changeset.
- Create local commit.

Git with perforce

Git has bridges to most other VCS

- Git repository created from perforce. (separate directory)
- Update git from perforce.
- Work on git repository (no explicit file checkout).
- Submit git commits to perforce.
- Commit description can contain defect number and commit can be linked.

View history

Why? When? By whom?

- View graph of branches.
- Search inside of commits.
- View changes bentween (ranges of) revisions.
- File history.

The buckets

Git tracks not files but content

1. Untracked files (easily clean them up).
2. Working tree. (view history).
3. Changes on local files (list of modified files, diffs).
4. Staging area for commints. (add diffs, add/remove files).
5. Local commits (can be edited, reverted).
6. Remote commits (1984 - Ministry of Truht: don't mess up others' timeline).

Handy tools

Stash

- Need to switch branches without abandoning or commiting half-done changes.
- The changes of local files can be saved as a patches on stash.
- The working directory will be clean (before merge, etc)

Sending the patch to each other without a central server.

- Local state (diff) can be exported as patch (can be attached to e-mail, or copied from a pendrive).

Taking ownership of patches.

- Patches can applied as commits (by a “maintainer“).
- Cherry pick: apply a commits (from other branches) on current branch.

Branches

Need to switch branches without abandoning or commiting half-done changes.

- Switching between branches and commits are very fast (history stored locally, branches are very lightweight)
- Quick, effortless local branch creation for each task or feature.
- Easy merges and rebases smart algorithms)

Advanced

Bisect

- Binary search for the commit which introduced a bug.
- Marking the last known good state.
- Marking a commit known to be broken.
- Git selects a commit in the middle, which can be marked again either good or bad.

Subtrees and submodules

- Part of the repositoty is a link to another repository.
- Pulling the changes of the (maybe 3rd party) subsystem.
- Having access to the history of the subsystem.

Step right up, step rigth up...

To managers - the project should use Git, because:

- It is free and open Source (maintenance)
- Most advanced VCS out there, check list of projects using it.

To programmers - use it to track your code, because:

- Fast with advanced tools.
- Collaboration and branching and merging are very easy.
- Huge community, lots of online help and examples.

To everyone - use it to track your content, because:

- Very easy to use to backup your text files (documents and configuration files).
- Lots of convenient services (github, gitourious - free online, firmware of your network attached storage)

Thank you for your attention!

This presentation can be found at:
`http://github.com/cs0rbagomba/git_p4/git_p4.pdf`

or

Any questions?