Advanced Practical Programming for Scientists

gdb, Git and Doxygen

Matthias Miltenberger

Zuse Institute Berlin

May 19th, 2017

gdb, Git and Doxygen

Debugging

- popular free debugger: `gdb` (GNU debugger)
- can debug code written in any language covered by the GNU Compiler Collection (C/C++, Ada, Java, FORTRAN, . . .)
- more convenient than putting `printf()` at critical parts of your code
- see complete callstack/backtrace of the functions in your code
- inspect values of variables while your code is running
- pause execution whenever a variable is accessed
- . . .

Extensions

- use `gdb` within `Eclipse` (or your favorite IDE) to immediately see where you currently are
- undodb-gdb: go backwards in your code
    - `http://undo-software.com/`
    - easy to find out what caused a bug
    - unfortunately not for free

Demo

Time for a demo!

gdb, Git and Doxygen

Who should use git?

- every programmer
  - version/history control
  - branches
- everyone who uses more than one computer
  - file server (use `git-lfs` for large files)
- everyone who works together with other people
  - file server
  - diff and log
- everyone who forgets what he/she did last week
  - diff, log, show

Introduction

Features of git:

- **distributed version control system**
  - every participant has the complete history of the repository
- branching and merging is very quick and easy
  - local branches do not affect other repositories
- fast and memory efficient
  - only one directory per repository
  - tracks content instead of single files
  - most actions work offline
- powerful yet user-friendly
  - detailed help for every single command and use case

Origin

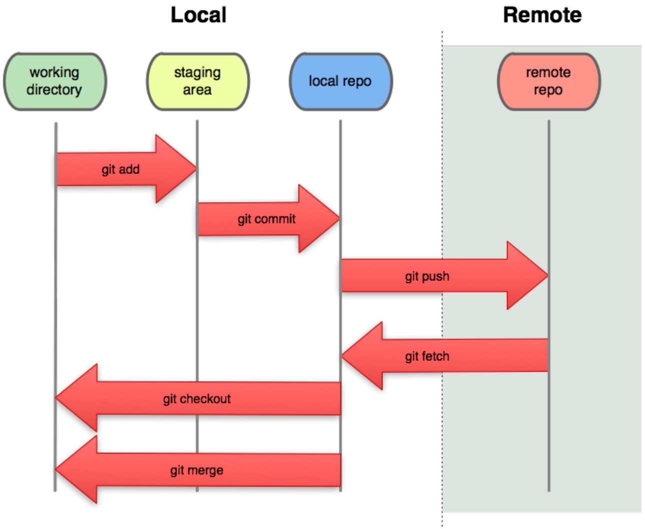
- `git` was created by Linus Torvalds

  I'm an egoistical bastard, and I name all my projects after myself. First 'Linux', now 'git'.

git (Oxford dictionary):

- An unpleasant or contemptible person:
  - *that mean old git*
  - *a warped, twisted little git*

How git works



Basic git commands

- `git`: list all common commands available
- `git help command`: extensive documentation for **every** command
- `git config`: change details about your personal git (name, e-mail, editor, . . .)
- `git status`: check whether changes are to be commited or if files are not (yet) tracked
- `git log`: see all previous commits on your current branch
  `--stat`: also show the changes that were made
- `git branch`: list all available branches of your repository
- `git diff`: list all changes between your current state and the last commit or between two certain commits

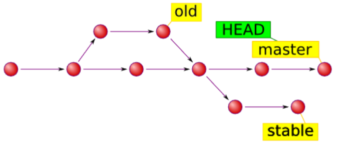
Local git commands

- `git add <file>`: add `file` to the index to be tracked by git
- `git commit`: save current changes in the index to a new commit
  - `-a`: put all changed files into the index first and commit them afterwards
  - `--amend`: correct the last commit
  - `-m`: write commit-message directly after the command (no editor opens)
- `git branch <branchname>`: create new branch `branchname` based on current state
- `git checkout <hash/tag/branch>`: switch to another commit or branch
- `git merge <branchname>`: merge `branchname` into current branch
- `git reset --hard`: revert all changes made since the last commit

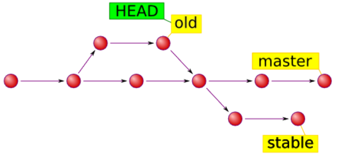
Important note

- Always read the output of git!
- It often contains very helpful tips on what to do next or how to revert what you have just done!

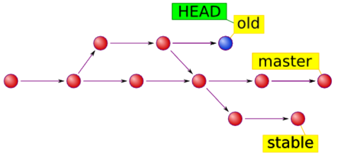
Branching



After "git checkout old":



After "git commit":



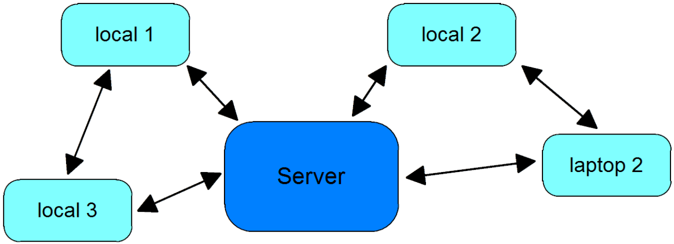
- simple example for branching
- HEAD refers to current state of working directory
- `git checkout` switches to another state (here old)
- `git commit` extends this branch or creates a new one

How to refer to a certain commit?

- (short) SHA1-hash
- `HEAD`: current position
- `HEAD~2`: go back two commits
- branch names are synonyms for the SHA1-hashes of their leading commit

Idea of Distribution

- multiple repositories everybody can pull from
  - joint work on experimental features
- server repository:
  - gather finished features
  - distribute stable versions
- example: `Github`



Remote Interaction

- `git clone`: create local copy of repository (containing complete history)
- `git fetch`: get latest updates/changes from remote repo without touching your local workspace
- `git pull`: sync your repository to remote repo
  - same as git fetch followed by git merge
- `git push`: copy local changes/branches to remote repository changes on the server must be pulled first
- `git remote add remotename`: add a new remote repository to work with (default is origin)

Useful tips

- `gitg / gitk`: visualization of the history tree
- `~/.gitconfig`: personal configuration file, valid for all your repos
- `.gitignore`: specify files (like `*.java~`) that should be ignored by git
- `hooks`: special scripts that run after certain git actions
  (example: check for trailing whitespaces)
- `git completion`: press TAB to see all your available options
- try to make lots of small commits
- don't commit binary files

- never change the history (`git rebase`) of remote commits

References

- git homepage: `http://git-scm.com/`
- official docu: `http://git-scm.com/documentation/`
- "Pro git" book: `http://git-scm.com/book/`
- any problems: `http://stackoverflow.com/`

gdb, Git and Doxygen

What is Doxygen?

- tool to automatically generate a documentation based on code and your comments
- supports all common programming languages
- can generate `HTML`, LaTeX, man pages, `XML`, and some more

Usage

- docu style is set using a config file
- `doxygen -g <config-file>` generates a default configuration
- simply run `doxygen <config-file>`
- doxygen will search for source within the current directory and subdirectories
- input paths can also be specified in configuration
- resulting documentation will be created in `html/`

Syntax

- how comments are supposed to look like:
  - `int var; ///< Brief description after the member`

  -
    ```
    /**
     * ...  text ...
     */
    void function(int x)
    ```
- some keywords doxygen will look for (@ is the same as \\):

@author:	author of the code
@param:	name and meaning of parameters a method receives
@return:	return value of a method
@todo:	notes for future work
...	there are many more

Further notes

- doxygen supports LaTeX code within comments to generate nice formulas
- generate inheritance graphs (for object-oriented languages)
- use the included GUI tool `Doxywizard` for an easy access
- read the official documentation for doxygen: `www.doxygen.org`