A HANDS-ON TUTORIAL

VERSION CONTROL WITH GIT

Thomas Deppisch - Freudenstadt July 10, 2017

WHAT IS GIT?

- a version control system by Linus Torvald, install with e.g.

  ```
  $ sudo apt-get install git
  ```

- keeps track of your project's history of changes
- can revert changes
- synchronises data between your PC / server / other PC
- allows collaborative developing

HOW DOES GIT WORK?

- every local repository contains the full history
- history = a series of snapshots

remote repository

- server, other PC...

git push

local directory

- contains your files and subdirectories
- ls

staging area

- index of tracked files
- git status

local repository

- contains complete history
- git show / log / ...

git add

git commit

TUTORIAL - BASIC COMMANDS

- create a new directory

```
$ mkdir myfirstgitrepo
```

- open it

```
$ cd myfirstgitrepo
```

- create a new file

```
$ echo “Hello world.“ > hello.txt
```

- initiate a git repository

```
$ git init
```

- add your file for tracking

```
$ git add hello.txt
```

- take a snapshot

```
$ git commit
```

- add a line to your file

```
$ echo “Hello sun.“ >> hello.txt
```

- list changed files

```
$ git status
```

- again, take a snapshot

```
$ git commit -a
```

TUTORIAL - PREVIOUS COMMITS

- history of your commits

```
$ git log
```

- each commit has a hash

```
commit c413f8b...
```

- go back to your first commit

```
$ git checkout c413f8b...
```

- look at your file

```
less hello.txt
```

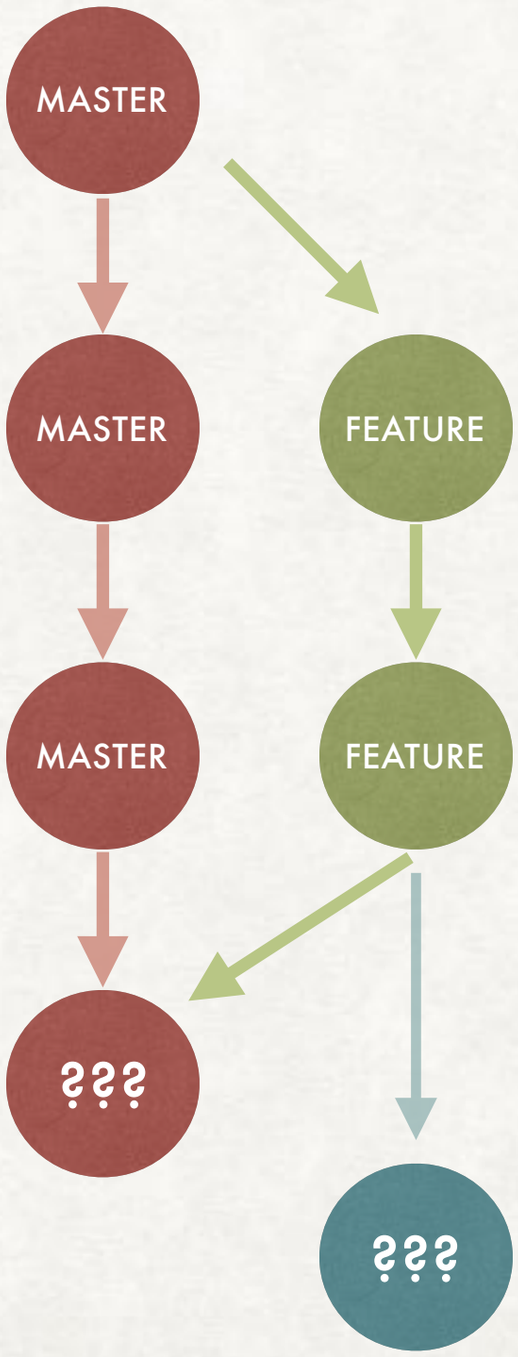
- return to the newest commit

```
$ git checkout master
```

- again, look at your file

```
less hello.txt
```

TREES AND BRANCHES



- the series of commits can be seen as a tree
- each project starts with a master branch
- additional branches allow you to work at different ends
- merging branches can lead to conflicts
- many possible *workflows* exist - a matter of taste / politics / philosophy
- Make sure to work on the right branch!

TUTORIAL – REPOSITORIES

- load the directory from the server and open it

```
$ git clone https://github.com/tdeppisch/git_tutorial.git
```

- the repository contains the following files
  - README.md — tells you what to do
  - makefile — controls compiling, needs g++
  - hello_world.cpp — the actual code
  - presentation.pdf — *THIS presentation
  - .gitignore — tells git to ignore all binary files
  - .gitattributes — tells git to treat pdf files and images “as a whole“

- compile and run the program

```
$ make

$ ./helloworld
```

TUTORIAL – CHANGES

- change the main function of hello_world.cpp to :

```
cout << "\n    Hello earth.\n“;

cout << "    Hello sun.\n“;

cout << "   Hello moon.\n“;

cout << "   Hello stars.\n\n“;
```

- compile and run the code again

```
$ make; ./helloworld
```

- if it works (!) commit your changes

```
$ git commit hello_world.cpp
```

- you can always get help

```
$ git – help
```

TUTORIAL – BRANCHES

- switch to a different branch, where a new feature has been implemented

```
$ git checkout switch_feature
```

- look at the changes the last commit on this branch made

```
$ git show
```

- the output shows you the commit message, the changes, other information
- explore the new feature by compiling and running the code

```
$ make; ./helloworld 1; ./helloworld 23; ./helloworld 67
```

TUTORIAL – MERGING

- switch back to the master branch and try to merge the branches

```
$ git checkout master; git merge switch_feature
```

- git tells you now that there is a merge conflict in hello_world.cpp
- open the file: a line has been overwritten and git shows now both versions
- the versions are delimited by ========, >>>>>>>>, <<<<<<<<
- choose one alternative and delete the other including the delimiters, then do

```
$ git commit -a
```

- all conflicts should now be cleared and no errors should occur

TUTORIAL – PUSHING

- publish your changes

```
$ git push origin master
```

- not everyone can change a public repository!
- but you can set up your own one
- ITP/TTP provide an own gitlab server:
  - git.particle.kit.edu
  - web interface to manage your projects
  - information can be fond on the wiki
- further information: google “git commands“, “git cheatsheet“,…

OTHER COMMANDS

- There are two command to update your local directory:
    - fetch adds changes to your history
    - pull also applies (merges) them
- go back to the last commit and delete all changes since then

```
$ git reset –hard
```

- go back to a certain commit and delete all changes (i.e. commits) since then

```
$ git reset –hard COMMIT
```

- reset changes your local history! It can not be reverted.
- Never change a history that has already been published (pushed)!