Version Control

PU Tools, Ressourcen, Infrastruktur

Nils Reiter,
nils.reiter@uni-koeln.de

November 12, 2020
(Winter term 2020/21)

Introduction

- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

Introduction

- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

Why is this useful?

- Programming projects quickly become massive
  - Windows 2000: 28mio LoC (ca. 930k standard pages)
  - CorefAnnotator: 27k LoC (ca. 770 standard pages)

Introduction

- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

Why is this useful?

- Programming projects quickly become massive
  - Windows 2000: 28mio LoC (ca. 930k standard pages)
  - CorefAnnotator: 27k LoC (ca. 770 standard pages)
- Large teams
  - working on the same project
  - over a long time (don't rely on human memory)

Introduction

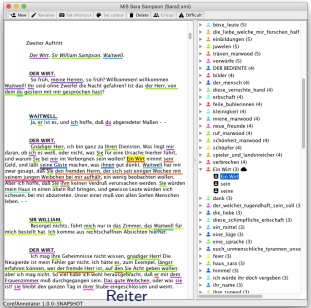
- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

Why is this useful?

- Programming projects quickly become massive
  - Windows 2000: 28mio LoC (ca. 930k standard pages)
  - CorefAnnotator: 27k LoC (ca. 770 standard pages)
- Large teams
  - working on the same project
  - over a long time (don't rely on human memory)
- A single conceptual change often distributed over many files

CorefAnnotator

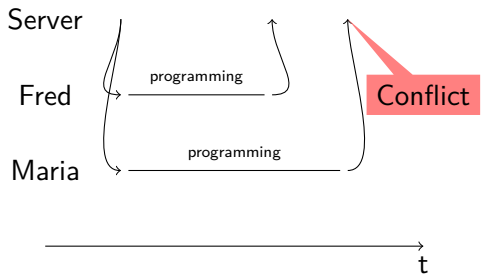
- Typical research software
- `https://github.com/nilsreiter/CorefAnnotator/`
- Annotation tool for coreference chains
- Open source, Apache License
- Version 1.0: February 12, 2018



Reiter

Version	LoC
1.0	13 087
1.1	14 028
1.2	15 781
1.3	18 988
1.4	20 742
1.5	22 034
1.6	22 792
1.7	22 967
1.8	26 225
1.9.2	27 052

Situations



Conflict resolution options

- Ignore, let Maria overwrite Freds code (this is bad!)
- Create a second copy (this is what Dropbox does)
- Force Maria to *explicitly* merge the code

What do we put under version control?

plain text files

- source code (python, java, perl, c, ...)
- texts (plain, latex, markdown)
- primary data (xml, csv)
  - but beware of large files
- vector graphics (svg)

What do we put under version control?

plain text files

- source code (python, java, perl, c, ...)
- texts (plain, latex, markdown)
- primary data (xml, csv)
  - but beware of large files
- vector graphics (svg)

Don't put these in VC:

Binary files

- word documents, pdf files
- images (jpg, png)
- compiled code (executables)

Software

- Very old
  - CVS (concurrent versioning system)
  - Rarely used today
- Old
  - SVN (subversion)
  - Sometimes used
- State of the art
  - git
- More solutions are available commercially

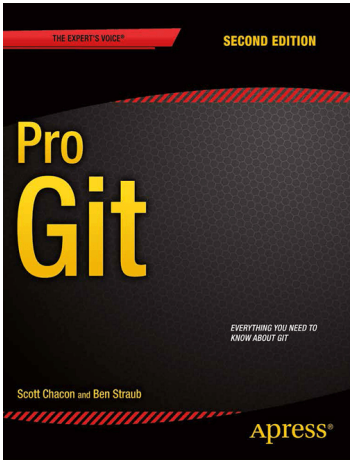
git

- Developed by the Linux kernel developers
- Open source – `https://git-scm.com`
- Distributed
  - No central server required
    - ...but still useful to have one
- Fast
- Data assurance
  - Checksums to make sure you get out what you put in

git vs. GitHub vs. GitLab

- git is an open source software
  - `https://git-scm.com`
- GitHub is a (commercial) web platform
  - Recently bought by Microsoft
  - GitHub provides a central server for git repositories *and* additional services (wiki, ticket system, ...)
  - `https://github.com`
- GitLab is an open source software
  - Provides a central server that you can install on your own server (e.g., at the IMS)
  - `https://about.gitlab.com`

Reading



Scott Chacon and Ben Straub: "Pro Git". 2nd edition. Apress, 2014.
`https://git-scm.com/book/en/v2`

Table of Contents

1. Getting Started
2. Git Basics
3. Git Branching
4. Git on the Server
5. Distributed Git
6. ...

Subsection 1

How does git work?

Commit

- One version of an entire directory (including subdirectories)
- Creating commits is the central activity we do
- Each commit knows its predecessor
- Each commit is identified by a hash value:
  `0eabb4bfef80be2af18255dc19301b989da1f1a3`
- A commit can include changes in multiple files

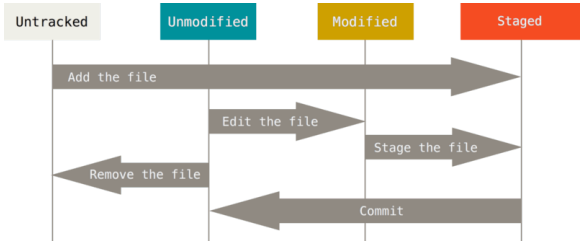


Figure: The lifecycle of the status of your files (Chacon/Straub: Pro Git)

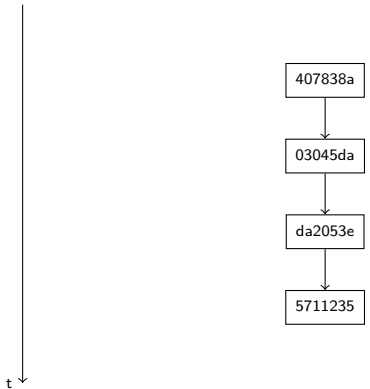
Workflow

1. (Pull changes from others)
2. Edit/add files
3. Put files in staging area
   - `git add <FILENAME>`
   - `git remove <FILENAME>`
4. Commit all files in staging area
   - Provide a useful description
   - `git commit -m "comment"`
5. (Push to others)

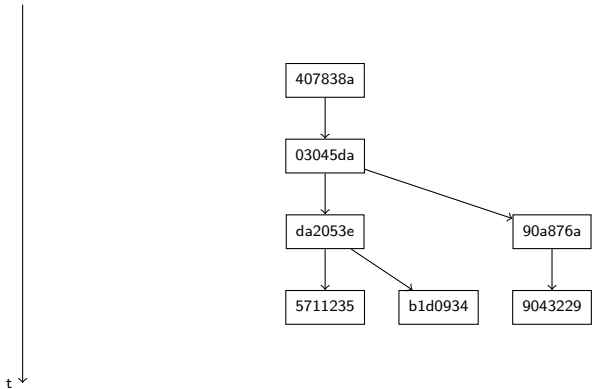
Branching

- Maintaining multiple branches is often useful
- At each time, a single branch is active
  - By default: `master`
- Switch to an existing branch
  - `git checkout <BRANCHNAME>`
  - To create a new branch, add the option `-b`:
    - `git checkout -b <BRANCHNAME>`

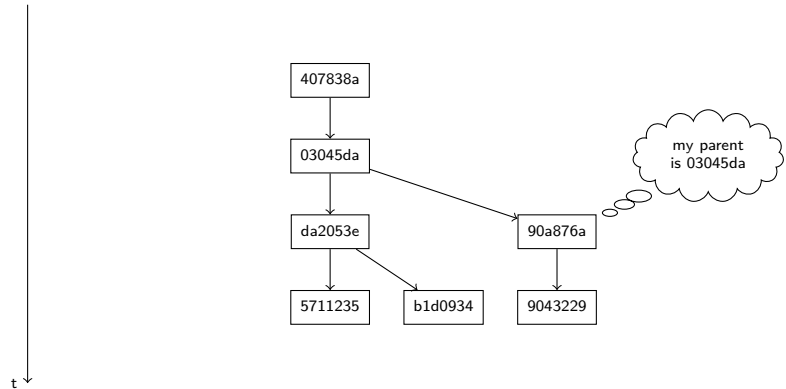
Branching and committing results in a tree



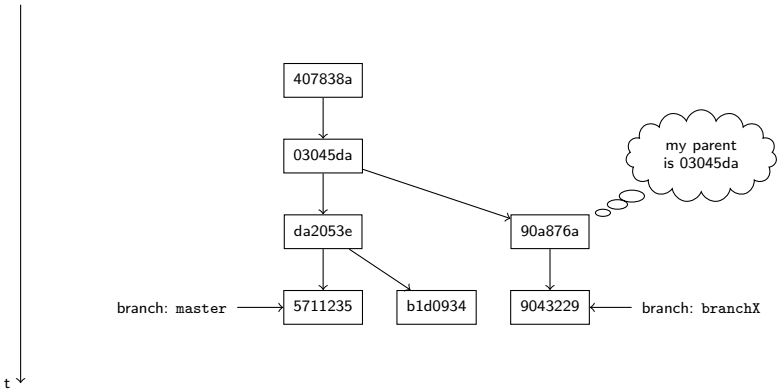
Branching and committing results in a tree



Branching and committing results in a tree



Branching and committing results in a tree



demo

Repository vs. working copy

- The git repository keeps track of *all* past versions and branches
- The working copy can be set to any of the past versions
- `git checkout REFNAME`
  - `REFNAME` can be a branch or revision hash (or tag)
- Checking out moves the HEAD pointer to another revision
  - The HEAD pointer always points to the revision that's active in your working copy

Remotes

- Git repositories can be associated with *remote* repositories
  - Remote repositories are usually on a different computer (e.g., GitHub)

Remotes

- Git repositories can be associated with *remote* repositories
  - Remote repositories are usually on a different computer (e.g., GitHub)
- A repository needs to be synchronized with its remote manually:
  - `git push`: Transfers the commits on the local branch to the same branch on the remote
  - `git pull`: Transfers the commits on the remote branch to the local branch
  - `git clone REPOURL`: Create a local copy of the repository url, setting REPOURL as 'origin' remote

Useful commands

`git status`

Shows the status of the current working copy

- Changed files
- Files in the staging area
- The current branch

Useful commands

`git status`

Shows the status of the current working copy

- Changed files
- Files in the staging area
- The current branch

`git log`

Shows information about current and past commits
Useful options:

`--oneline` Each commit is shown on a single line

`--graph` Information is rendered visually

`--all` Shows information about all branches

On GUIs

Git has a complex task and is a complex piece of software

- Graphical user interfaces do exist and make some tasks easier
- In this class: command line

On GUIs

Git has a complex task and is a complex piece of software

- Graphical user interfaces do exist and make some tasks easier
- In this class: command line
- Recommendations
  - SourceTree (Win/Mac): `https://www.sourcetreeapp.com`
    - Needs a registration with BitBucket (similar to GitHub), but free
  - GitKraken (Win/Mac/Lin): `https://www.gitkraken.com`
    - Free for open source projects

On GUIs

Git has a complex task and is a complex piece of software

- Graphical user interfaces do exist and make some tasks easier
- In this class: command line
- Recommendations
  - SourceTree (Win/Mac): `https://www.sourcetreeapp.com`
    - Needs a registration with BitBucket (similar to GitHub), but free
  - GitKraken (Win/Mac/Lin): `https://www.gitkraken.com`
    - Free for open source projects
- More can be found here:
  `https://git-scm.com/downloads/guis/`

Exercise: `https://learngitbranching.js.org`

Exercise 02

`https://github.com/idh-cologne-tools-ressourcen-infra/exercise-02`