Git: Vom Einstieg bis zur Teilnahme an Open-Source-Projekten

20.03.2016

Danny Messig (Lehrstuhl NTFD - TU Freiberg)
Steffen Weise (Mercateo AG - Leipzig)

Introduction - University

- Founded in 1765 (oldest mining school of the world)
- Famous scientists: Humboldt, Werner, Winkler, Lomonosov
- Discovery of Indium and Germanium
- Approx. 1350 scientific staff / 5000 students
- 56 Mio €/year third party funding 2014

Hamburg
Berlin
Dresden
Frankfurt
Freiberg
Munich

Department of Energy Process Engineering and Chemical Engineering (IEC)

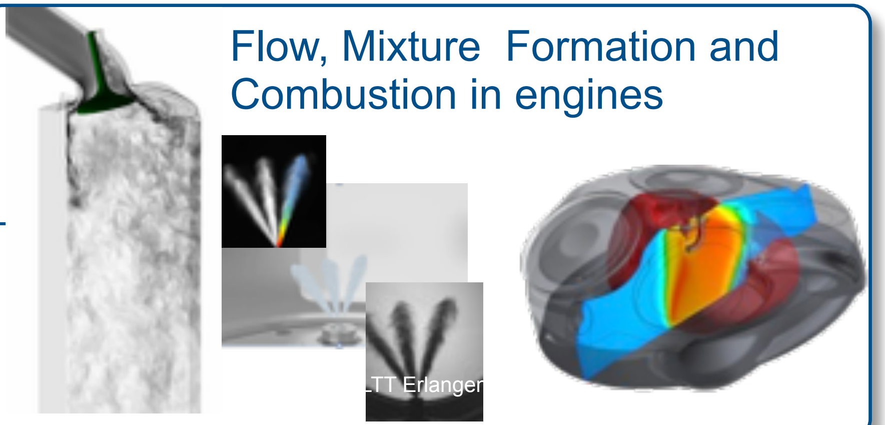
- More than 130 research and technical staff

Numerical Thermo-Fluid Dynamics

PROFESSUR FÜR NUMERISCHE THERMOFLUIDDYNAMIK

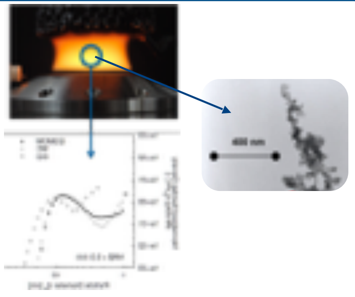
20 PhD students and Post-Docs
1 team assistant
6-8 students

Flow, Mixture Formation and Combustion in engines



Pollutants

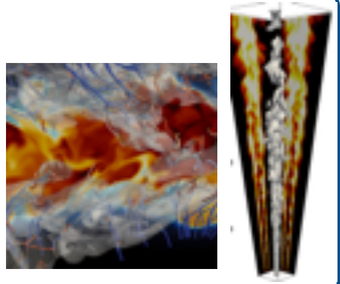
- Soot
- NOx
- HC
- CO



From combustion science to combustion technology

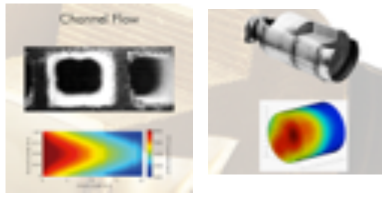
Turbulent Flames

- Combustion models
- URANS
- LES
- DNS



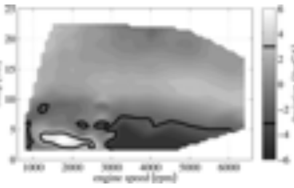
Exhaust Gas Aftertreatment

- TWC
- SCR



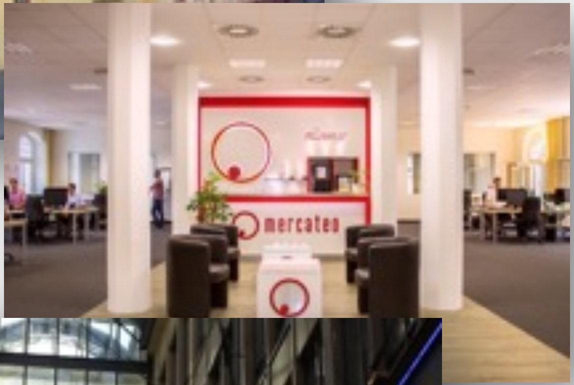
1D Models

- coupled to 3D combustion models
- emission models
- driving cycles



Zahlen, Daten, Fakten

Gründung:	1999 in München
Sitz der Mercateo AG:	München
Niederlassungen:	Köthen (Anhalt) u. Leipzig
International:	13 Länder
Mitarbeiter:	380
Kunden:	1, 3 Mio. Geschäftskunden
Artikel:	> 19 Mio. auf dem offenen Marktplatz
Umsatz:	rund 200 Mio. Euro
Geschäftsfelder:	B2B Online-Marktplatz, E-Sourcing-Lösungen, Transaktionsplattform



Mercateo in Zahlen

1.332.789
registrierte Geschäftskunden

32.000
Besucher pro Tag

4.000
Bestellungen pro Tag

3.600
Neukunden pro Monat

9.700
Hersteller

515
Lieferanten

19.706.016
verfügbare Artikel

13
europäische Länder

Januar 2016

Agenda

1. Introduction
2. Basics
3. Branching
4. Git on the server
5. Distributed Git
6. Git Tools

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

https://xkcd.com/1597/

(All other pictures in this presentation are taken from:
Pro Git by Scott Chacon ISBN-13: 978-1-4302-1834-0)

History & Introduction

What is **Git**:

- Git is a distributed **version control system**
- Developed
    - by Linus Torvalds in April 2005
    - for managing Linux kernel project (since kernel 2.6.12 release in June 2005)
- Since July 2005 maintained by Junio Hamano

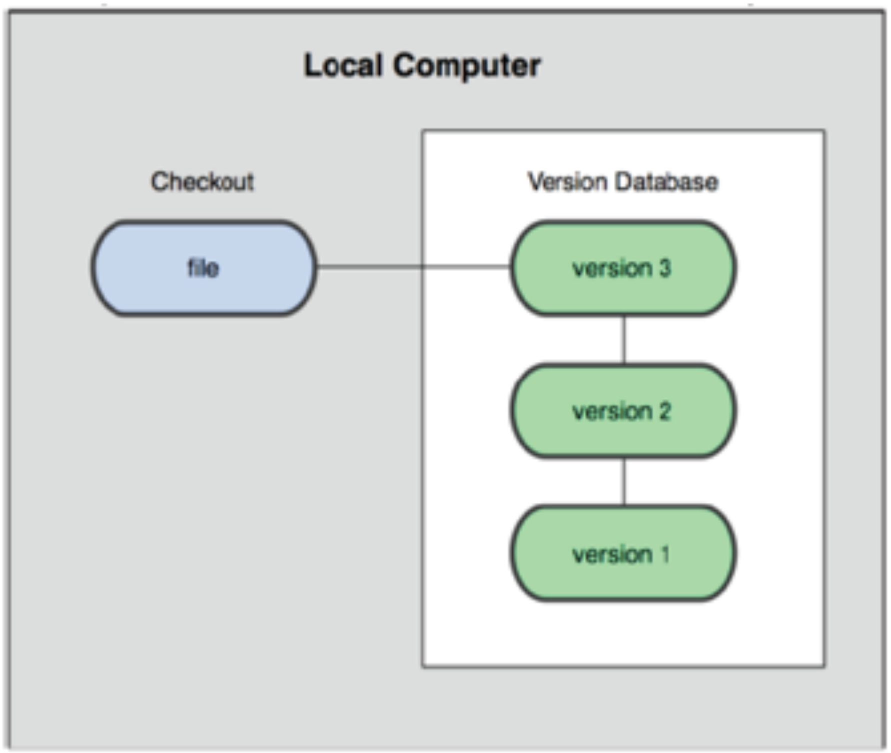
What is **Version Control**?

- Version control is a system that **records changes** to a file or set of files **over time** so that **you can recall specific versions later**.

History & Introduction

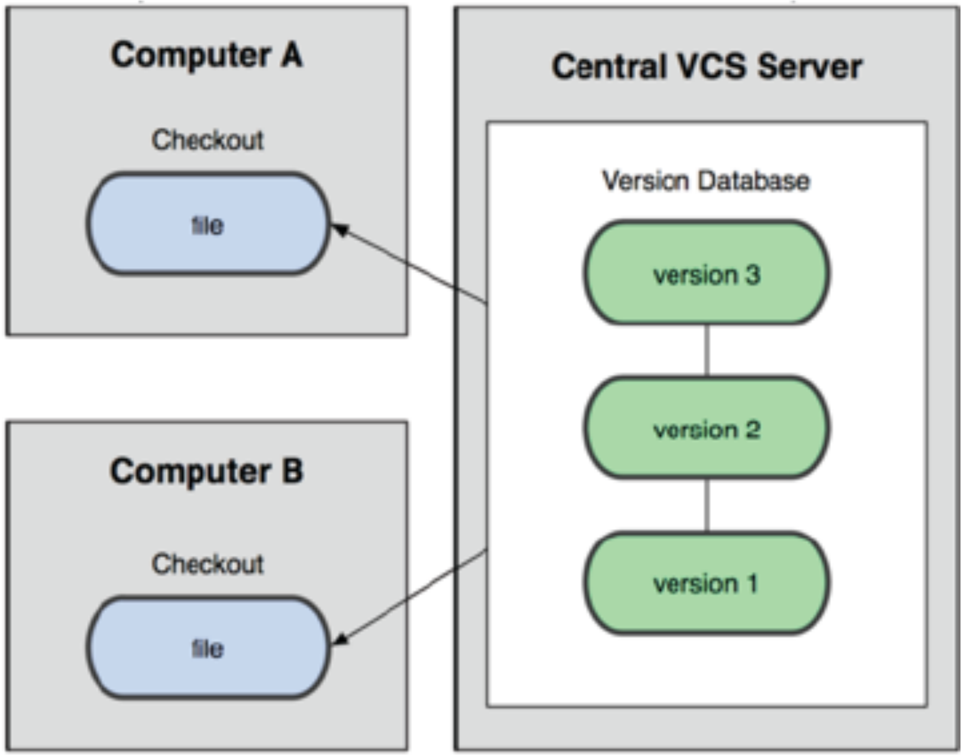
Types of Version Control System:

- Local Version Control Systems (e.g.: rcs)
    - A simple database that kept all the changes to files under revision control



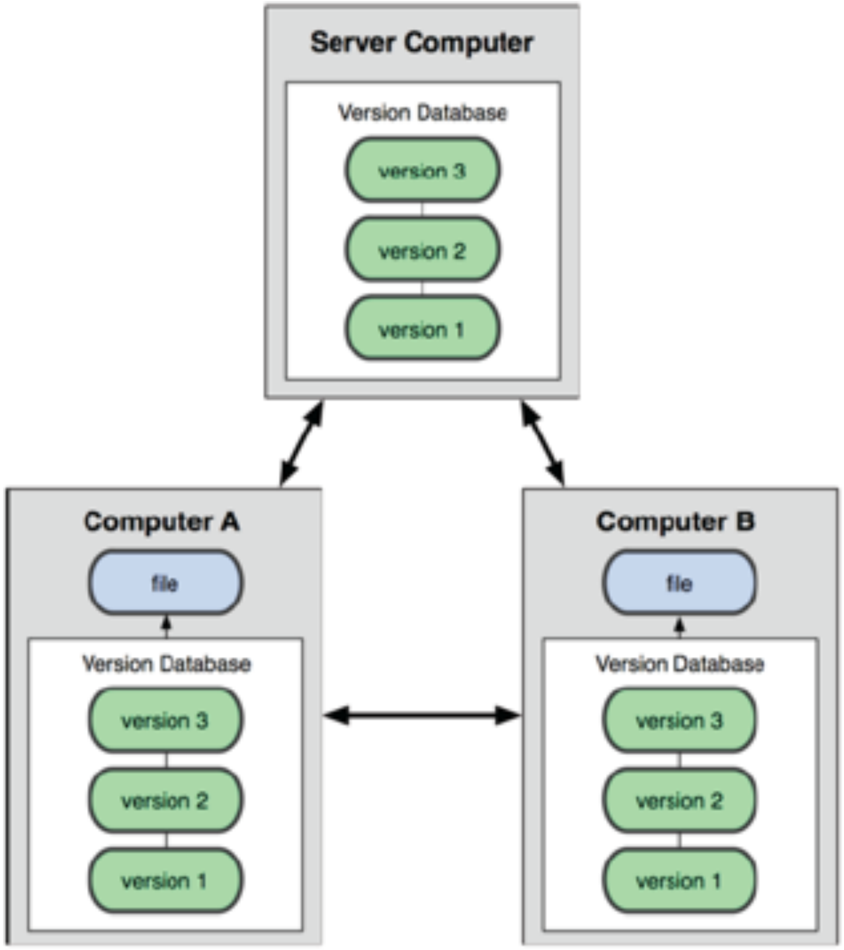
History & Introduction

- Centralized Version Control Systems (CVS, SVN)
  - **single** server that contains all the versioned files
  - clients check out files from that server and save the changes theres (➔drawback)



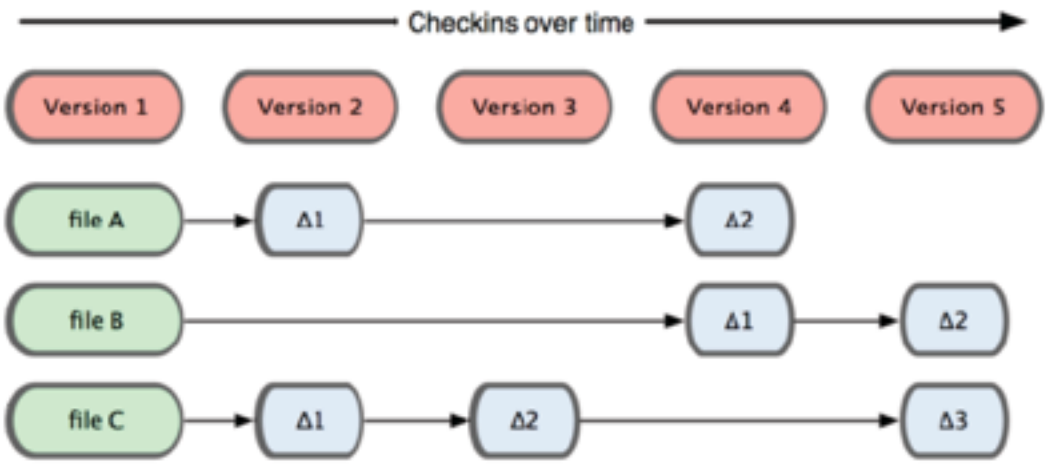
History & Introduction

- Distributed Version Control Systems (Git, mercurial)
  - clients fully mirror the repository

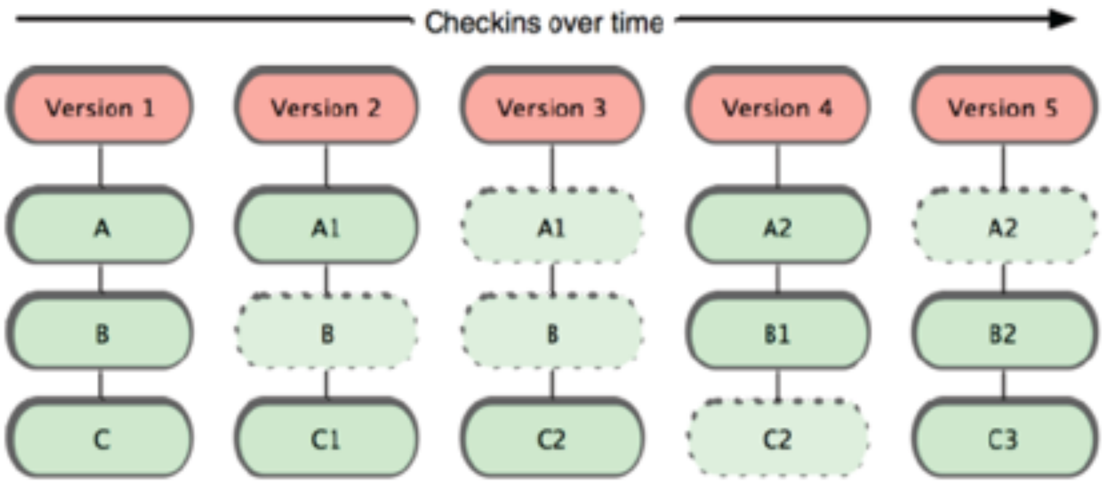


Data representation

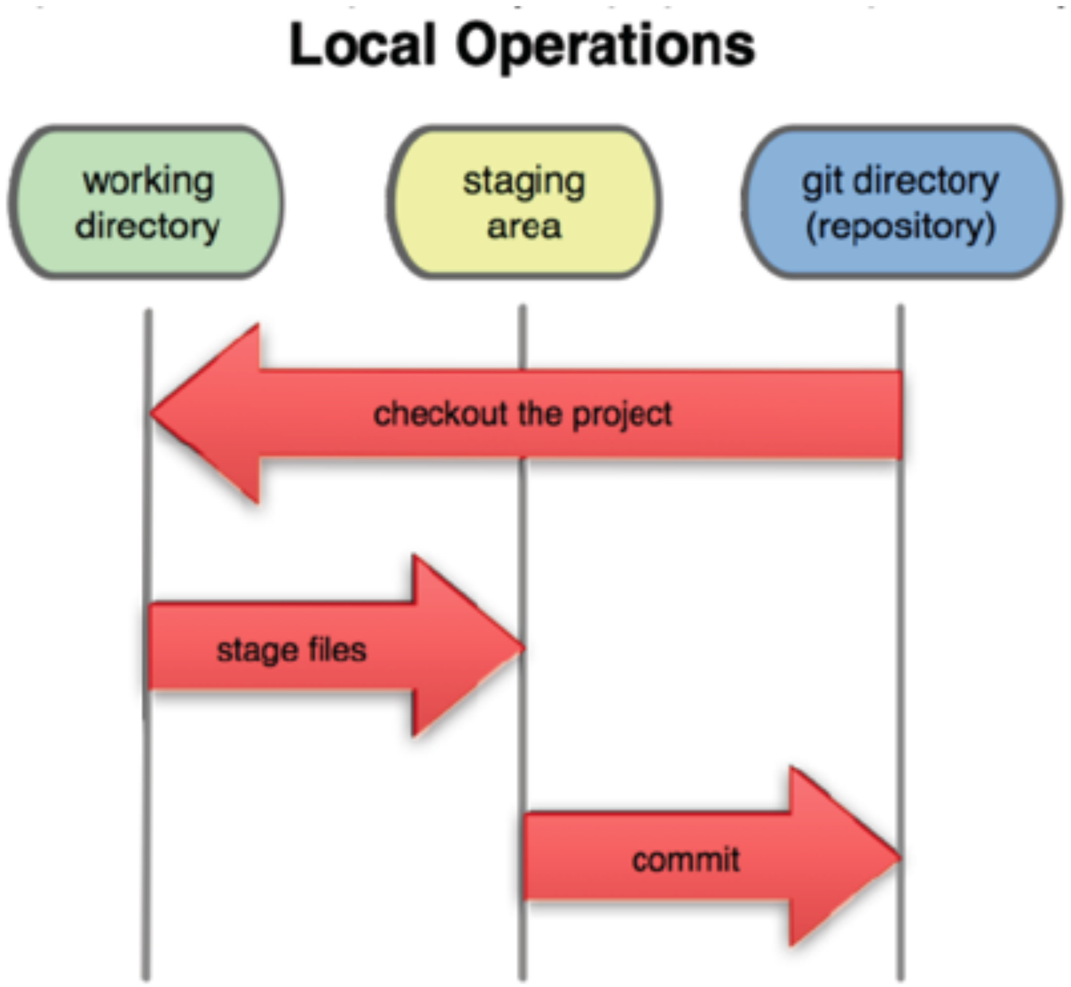
Others (Differences)



Git (Snapshots)



File locations



Properties

Remarks:

- Nearly every operation is **local**
- Git has **integrity** because of SHA-1 hash

  e.g.:
  24b9da6552252987aa493b52f8696cd6d3b00373

- Git generally only **adds data**

Setup

First-Time Git Setup via **git config**

```
$> git config --global user.name “Max Mustermann“
$> git config --global user.email mustermann@server.de
$> git config --global core.editor vim
$> git config --global merge.tool vimdiff
$> git config --list
user.name=Max Mustermann
user.email=mustermann@server.de
core.editor=vim
merge.tool=vimdiff
```

- **Global and user specific configurations** for Git in:
  - /etc/gitconfig and
  - ~/.gitconfig

Getting help

Getting **help**

- syntax:
    - git help *command*
    - git *command* --help
    - man git-*command*

$> **git help config**

... (manpage of git config)

http://git-scm.org/doc

https://progit.org

Agenda

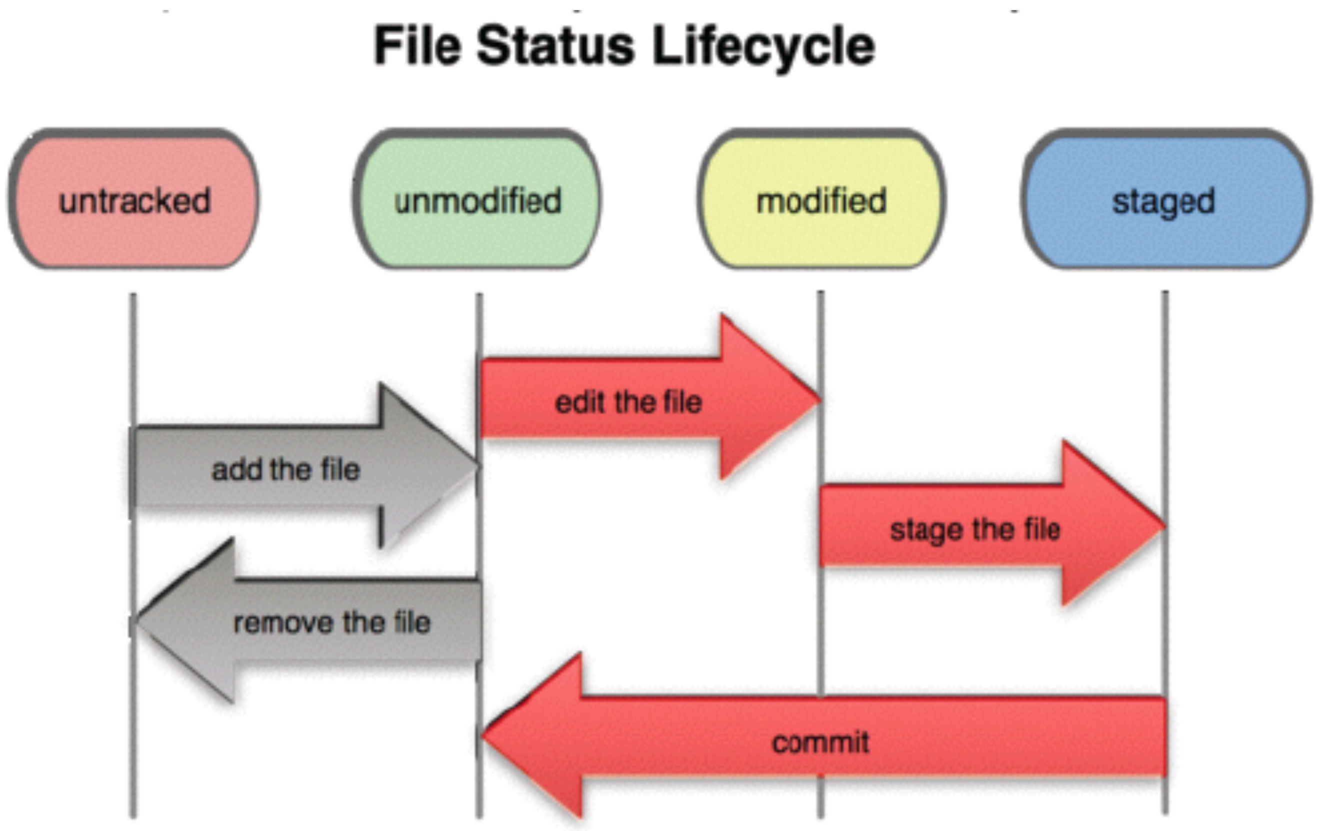
training session - setup

```
$> mkdir mygit
$> cd mygit/
$> git init .
Initialized empty Git repository in
/home/weise/mygit/.git/

$> git config user.name "Steffen"
$> git config user.email
"steffen@iec.de"
$> git config core.editor "vim"
$> git config alias.st "status"
$> ls .git/
branches  config  description  HEAD
hooks  info  objects  refs
```

```
$> cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
        editor = vim
[user]
        name = Steffen
        email = steffen@iec.de
[alias]
        st = status
```

File state


File Status Lifecycle
untracked
unmodified
modified
staged
edit the file
add the file
stage the file
remove the file
commit

training session - file states 1

```
$> echo "First line in my first
file" >> firstfile.txt
$> git st
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include
in what will be committed)
#
#       firstfile.txt
nothing added to commit but
untracked files present (use "git add"
to track)

$> git add firstfile.txt
```

```
$> git st
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to
unstage)
#
#       new file:   firstfile.txt
#
$> git commit -m "first commit"
[master (root-commit) eb157f9] first
commit
 1 files changed, 1 insertions(+), 0
deletions(-)
 create mode 100644 firstfile.txt
$> echo "Second line" >>
firstfile.txt
```

training session - file states 2

```
$> git st
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will becommitted)
#   (use "git checkout -- <file>..." to discard changes inworking directory)
#
#       modified:   firstfile.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
$> echo "First line of my second file" >> secondfile.txt
```

```
$> git st
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in workingdirectory)
#
#       modified:   firstfile.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       secondfile.txt
no changes added to commit (use "git add" and/or "git commit -a")
```

training session - file states 3

```
$> git add .
$> git st
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   firstfile.txt
#       new file:   secondfile.txt
#

$> git commit -m "second commit, added a few files"
[master b3a2f80] second commit, added a few files
 2 files changed, 2 insertions(+), 0 deletions(-)
create mode 100644 secondfile.txt
$> echo "Second line of my second file" >> secondfile.txt
$> git add secondfile.txt
$> echo "Third line of my second file" >> secondfile.txt
$> git st
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   secondfile.txt
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   secondfile.txt
#
```

training session - file states 4

```
$> git checkout -- secondfile.txt
$> git st
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   secondfile.txt
#
$> git commit -m "working my way through"
[master 36664b2] working my way through
 1 files changed, 1 insertions(+), 0 deletions(-)

$> git log
commit 36664b2c9bb5ab7191cf831ccdc936cf06bef31b
Author: Steffen <steffen@iec.de>
Date:   Wed Oct 12 10:29:44 2011 +0200

    working my way through

commit b3a2f80a98b9e5a2270b76f05090d6430dd09dc0
Author: Steffen <steffen@iec.de>
Date:   Wed Oct 12 10:22:25 2011 +0200

    second commit, added a few files

commit eb157f90120dc232aefdd20ca6a7ef5b8fa2f38e
Author: Steffen <steffen@iec.de>
Date:   Wed Oct 12 10:17:55 2011 +0200

    first commit
```

training session - ignore & diff

```
$> ls
firstfile.txt  secondfile.txt
$> echo "*.[oa]" >> .gitignore
$> touch my.a
$> touch this.o
$> git st
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will
be committed)
#
#       .gitignore
nothing added to commit but untracked files
present (use "git add" to track)
$> ls && git add .gitignore
firstfile.txt  my.a  secondfile.txt  this.o
$> git commit -m "added some usefull
ignore statements“
...
```

```
$> vi secondfile.txt (What happens???)
$> git st
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will
be committed)
#   (use "git checkout -- <file>..." to discard
changes in working directory)
#
#       modified:   secondfile.txt
#
no changes added to commit (use "git add"
and/or "git commit -a")
$> git diff
diff --git a/secondfile.txt b/secondfile.txt
index c7dc2ec..a7ee1d5 100644
--- a/secondfile.txt
+++ b/secondfile.txt
```

training session - diff & patch

```
@@ -1,2 +1,3 @@
-First line of my second file
-Second line of my second file
+first line of my second file
+second line of my second file
+third line of my second file
$> git diff > my.patch
$> git add secondfile.txt
$> patch -p1 -R < my.patch
patching file secondfile.txt
$> git st
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   secondfile.txt
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   secondfile.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       my.patch
$> git diff --cached secondfile.txt
diff --git a/secondfile.txt b/secondfile.txt
index c7dc2ec..a7ee1d5 100644
--- a/secondfile.txt
+++ b/secondfile.txt
```

training session - diff & patch 2

```
@@ -1,2 +1,3 @@
-First line of my second file
-Second line of my second file
+first line of my second file
+second line of my second file
+third line of my second file
$> git diff HEAD secondfile.txt
$> git checkout -- secondfile.txt
git checkout -- secondfile.txt
$> git st
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   secondfile.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       my.patch
$> git commit -m "corrected case in secondfile"
...
$> rm my.patch
$> echo "first line of fourth file" >> thirdfile.txt
$> git add thirdfile.txt
$> git commit –m "added my fourthfile“
...
$> git mv thirdfile.txt fourthfile.txt
$> git st
# On branch master
# Changes to be committed:
```

training session - file rescue

```
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    thirdfile.txt -> fourthfile.txt
#
$> git rm secondfile.txt
rm 'secondfile.txt'
$> git st
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    thirdfile.txt -> fourthfile.txt
#       deleted:    secondfile.txt
#
```

```
$> git commit -m "moving some files around"
...
$> git rev-list  HEAD -- secondfile.txt
030b7c08d21e2f9380794aee66775c0e779de96c
f183967a9d2d6e9bc624d6195befff529b2e966f
36664b2c9bb5ab7191cf831ccdc936cf06bef31b
b3a2f80a98b9e5a2270b76f05090d6430dd09dc0
$> ls
firstfile.txt  fourthfile.txt  my.a  this.o
$> git checkout 030b7c08d21e2f9380794aee66775c0e779de96c^ secondfile.txt
$> git st
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   secondfile.txt
#
```

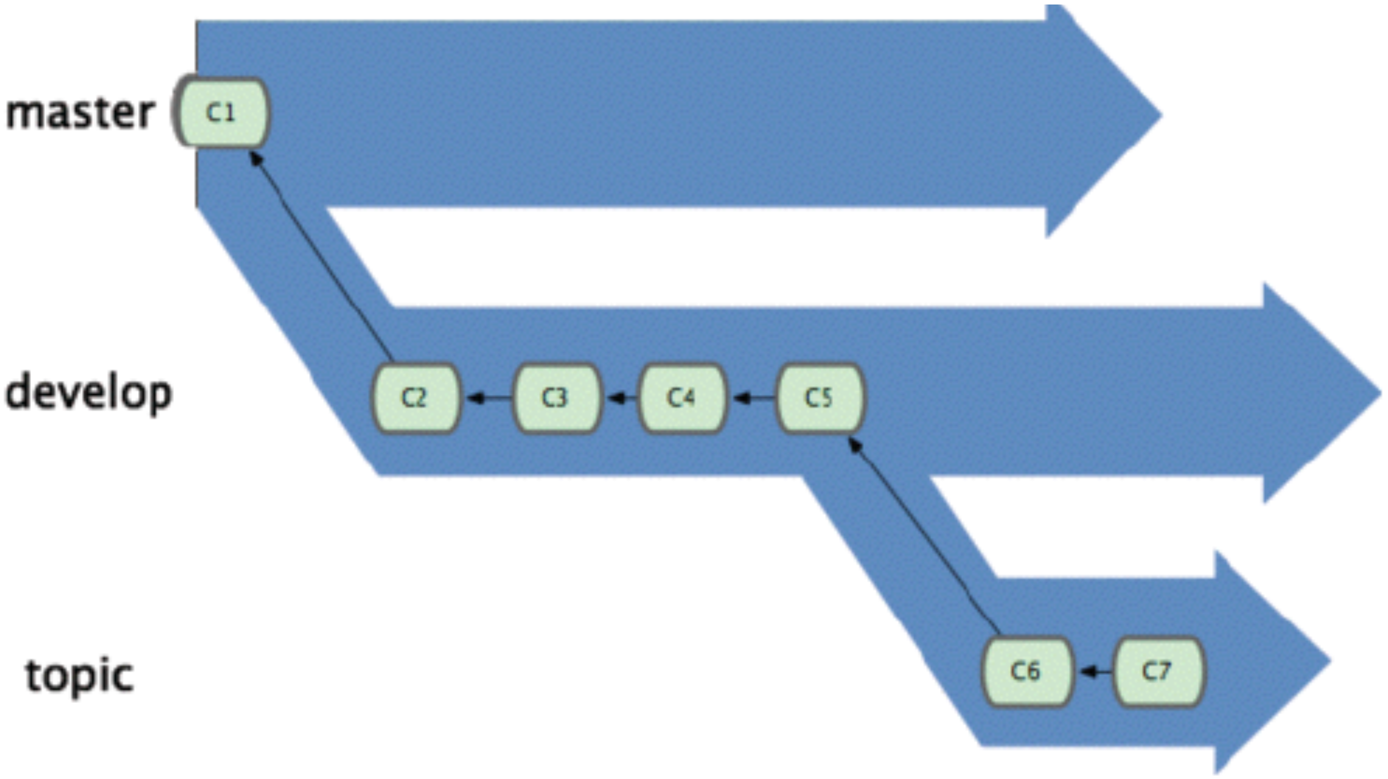
training session - file rescue 2

```
$> git add .
$> git commit -m "thank you git“
...
$> git config alias.lg "log --pretty=oneline --graph --abbrev-commit"
$> git lg
* 165dc66 thank you git
* 030b7c0 moving some files around
* c5e00df added my fourthfile
* f183967 corrected case in secondfile
* 6f119aa added some usefull ignore statements
* 36664b2 working my way through
* b3a2f80 second commit, added a few files
* eb157f9 first commit
```

Agenda

1. Introduction
2. Basics
3. Branching
4. Git on the server
5. Distributed Git
6. Git Tools

Branching



Branching visualisation

http://pcottle.github.io/learnGitBranching/

for exploration use:

http://pcottle.github.io/learnGitBranching/?NODEMO

Branching visualisation - showcase

01 create product branch & commit on both

02 merge product

03 undo (only in pcottle **NEVER** in git)

04 show rebase

05 merge again (explain difference)

training session - branching

```
$> git branch
* Master
$> git branch longterm
$> git checkout longterm
Switched to branch 'longterm'
$> git branch
* longterm
  master
$> vi longterm.txt
$> git add longterm.txt
$> git commit -m "added longterm branch“
...
$> git checkout master
Switched to branch 'master'
$> echo "Third line" >> firstfile.txt
$> git commit -am "more modifications on first file“
...
```

```
$> git checkout longterm
$> cat firstfile.txt
First line in my first file
Second line
$> echo "wow something in first file had to be modified" >> firstfile.txt
$> git commit -am "mod in firstfile"
...
$> git checkout master
$> git merge longterm
Auto-merging firstfile.txt
CONFLICT (content): Merge conflict in firstfile.txt
Automatic merge failed; fix conflicts and then commit the result
$> cat firstfile.txt
First line in my first file
Second line
<<<<<<< HEAD
```

training session - branching 2

```
Third line
=======
wow something in first file had to be modified
>>>>>>> longterm
$> vi firstfile.txt
$> git commit
[master e00dd7c] Merge branch 'longterm'
$ git lg
*   e00dd7c Merge branch 'longterm'
|\
| * a92c6d1 mod in firstfile
| * 15dc341 added longterm branch
* | f181430 more modifications on first file
|/
* 165dc66 thank you git
...
$> git branch -d longterm
Deleted branch longterm (was a92c6d1).
```

Agenda

Setup up your server

Setting up a git server

$> **ssh *user@server.com***

...

$> **mkdir myNewRepo.git**

$> **cd myNewRepo.git**

$> **git init --bare**

$> **logout**

... (back in mygit)

Adding server to Git-Repository

$> **git remote add *name user@server.com:*myNewRepo.git**

Communication with server

$> **git pull *name* master**

$> **git push *name* master**

Work with server - showcase (pcottle)

01 clone & commit & push to origin

02 branch & commit to branch & push to origin

03 commit to master & push

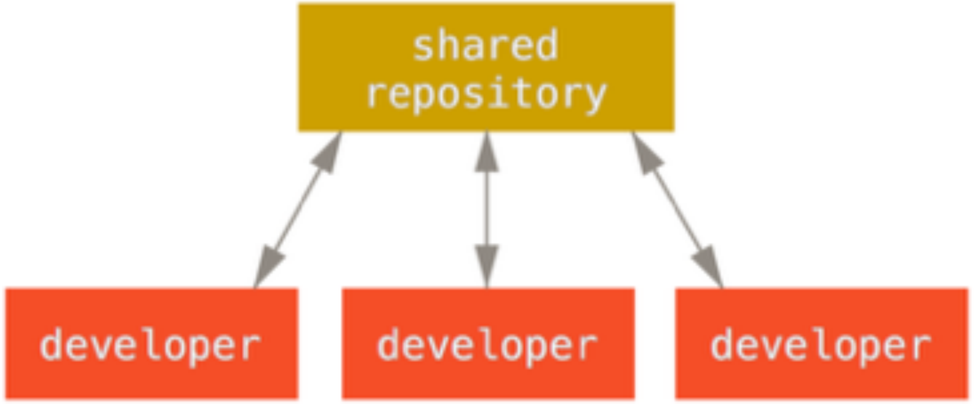
04 merge with master & delete branch

05 delete remote branch

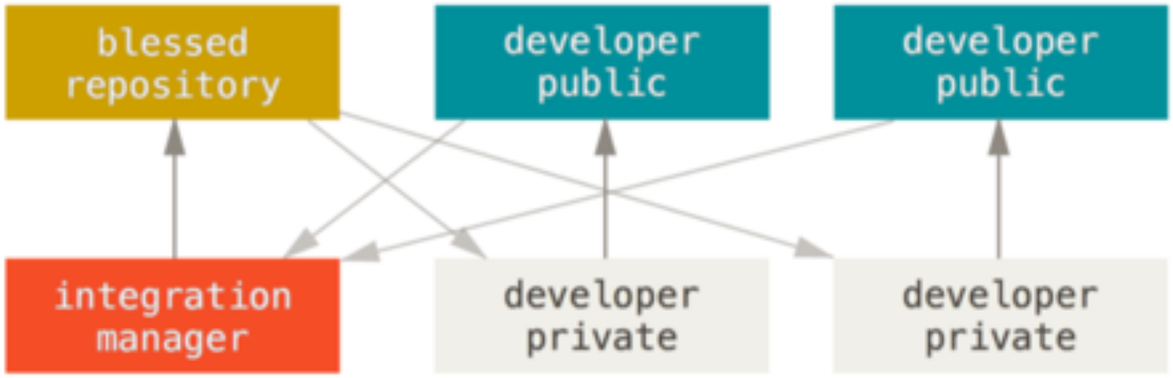
Agenda

Workflows

Centralized Workflow



Integration-Manager Workflow



Centralized workflow visualisation - showcase (pcottle)

01 clone & commit locally

02 fakeTeamwork 2 (only in pcottle **NEVER** in git)

03 try-push: error -> git pull

04 undo (only in pcottle **NEVER** in git)

05 git pull --rebase & git push (merge conflicts in real life)

06 local changes & reset

07 local changes -> push & revert

Working on GitHub

Integration-Manager Workflow

- most commonly used in Open Source Projects

https://github.com/

- showcase or common session for everybody

github - showcase

Steffen	Danny
01 create repo	
02 commit	
	03 fork
04 commit	
	05 compare (might require "switch base")
	06 pull request 'create' (directed at himself)
	07 merge
	08 merge confirm

github - showcase 2

Steffen | Danny

(same, except for merge commit)

09 commit (Danny)

10 create pull request + message (Danny)

11"pull requests" (Steffen)

12 "merge pull requests" (Steffen)

- merge of pull request can be done in web-interface or command line
- command line version uses branching model

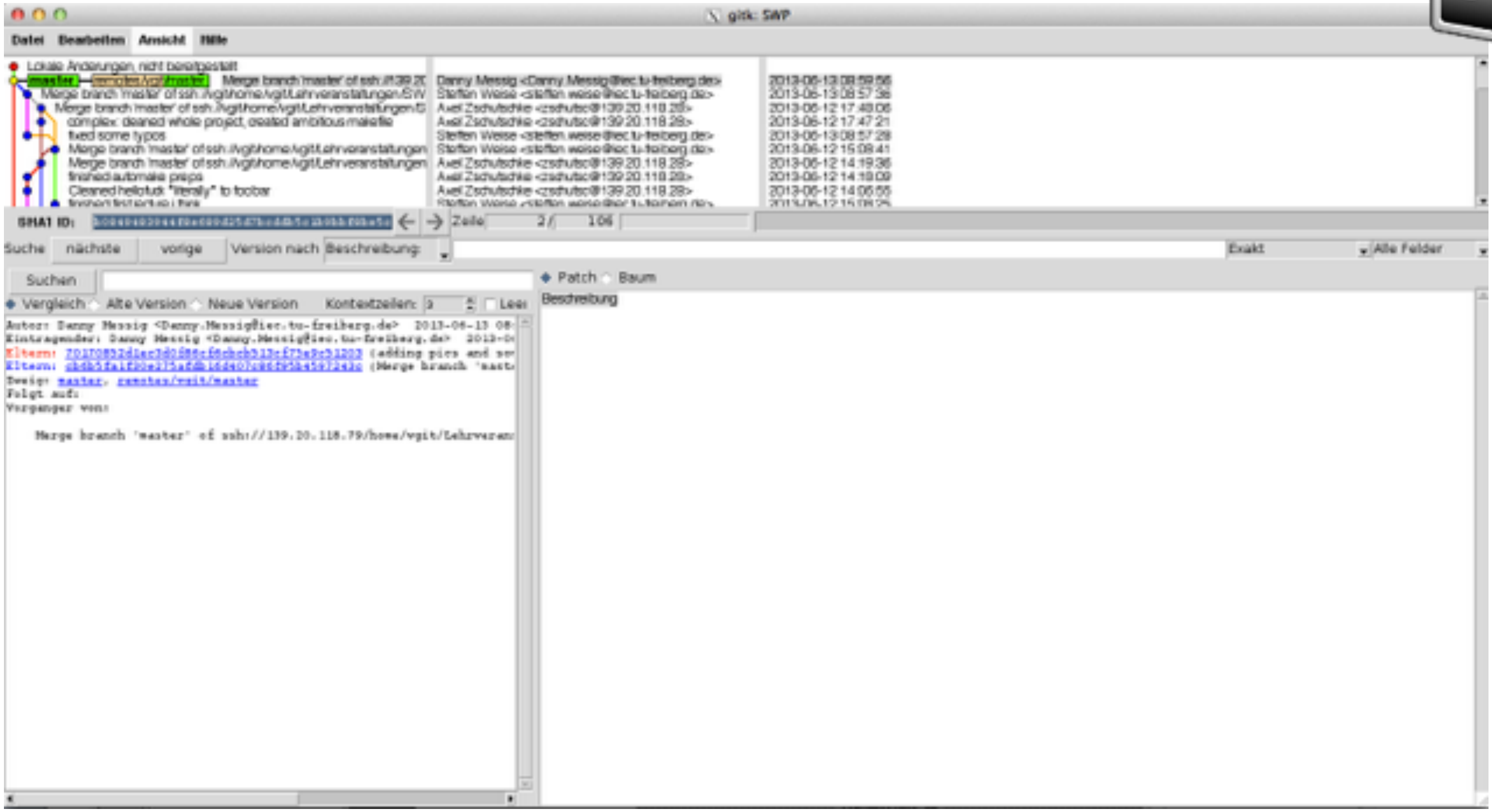
github - showcase 3

Steffen	Danny
(same, except for merge commit)	
13 commit test.txt for conflict	
	14 commit test.txt for conflict
	15 generate pull request -> conflict
16 pull requests (conflict resolution on command line)	
	17 pull request from S to D (directed at himself)

Agenda

Git

$> **gitk**

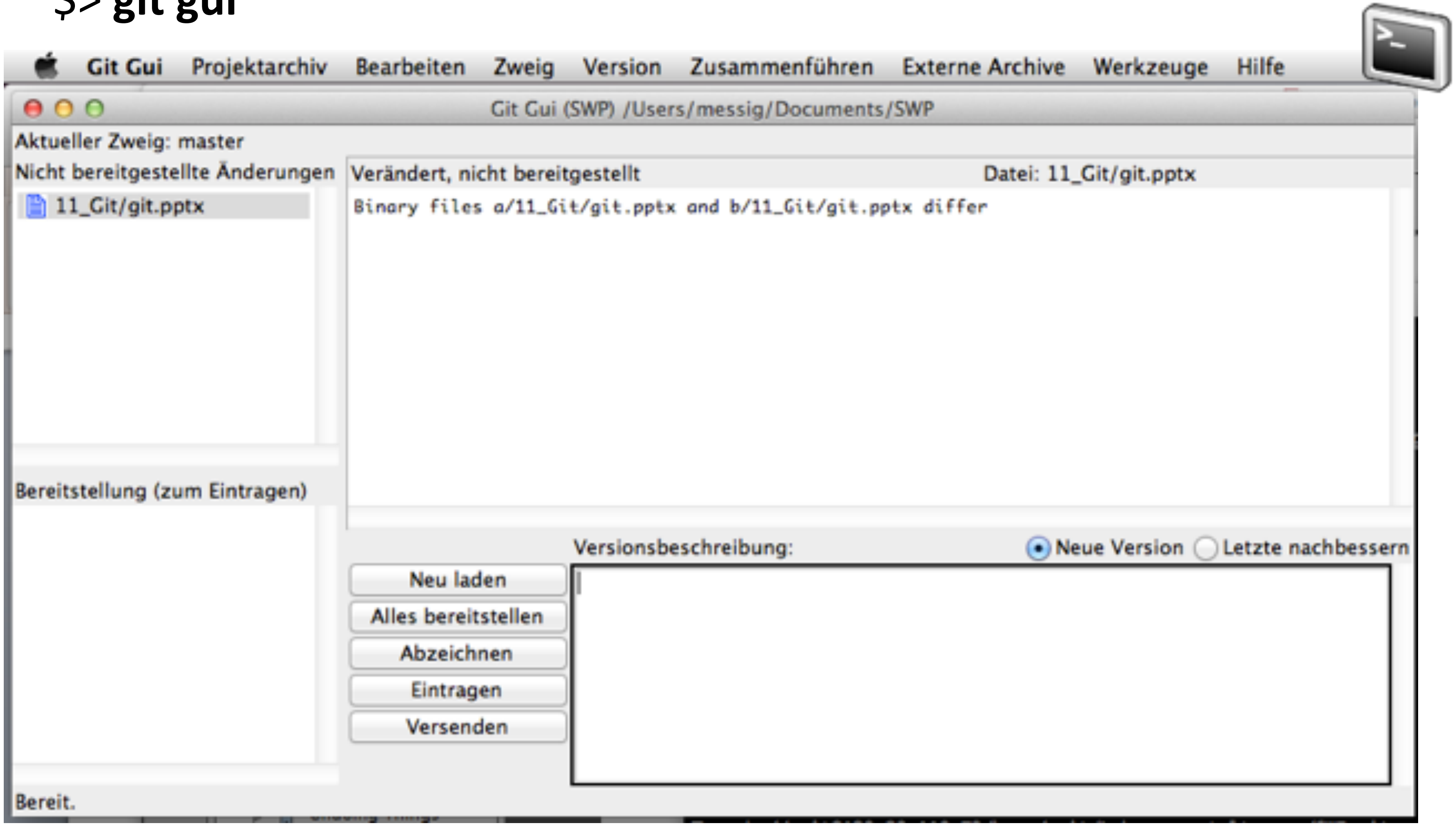


```
Lokale Änderungen, nicht bereitgestellt
master — remotes/vgit/master   Merge branch 'master' of ssh://139.20   Danny Messig <Danny.Messig@iec.tu-freiberg.de>      2013-06-13 08:59:56
Merge branch 'master' of ssh://vgit/home/vgit/Lehrveranstaltungen/SW     Steffen Weise <steffen.weise@iec.tu-freiberg.de>    2013-06-13 08:57:38
Merge branch 'master' of ssh://vgit/home/vgit/Lehrveranstaltungen/S      Axel Zschutschke <zschutsc@139.20.118.28>           2013-06-12 17:48:06
complex: cleaned whole project, created ambitious makefile               Axel Zschutschke <zschutsc@139.20.118.28>           2013-06-12 17:47:21
fixed some typos                                                         Steffen Weise <steffen.weise@iec.tu-freiberg.de>    2013-06-13 08:57:28
Merge branch 'master' of ssh://vgit/home/vgit/Lehrveranstaltungen        Steffen Weise <steffen.weise@iec.tu-freiberg.de>    2013-06-12 15:08:41
Merge branch 'master' of ssh://vgit/home/vgit/Lehrveranstaltungen        Axel Zschutschke <zschutsc@139.20.118.28>           2013-06-12 14:19:36
finished automake preps                                                  Axel Zschutschke <zschutsc@139.20.118.28>           2013-06-12 14:18:09
Cleaned hellotuck "literally" to foobar                                  Axel Zschutschke <zschutsc@139.20.118.28>           2013-06-12 14:06:55
```

```
Autor: Danny Messig <Danny.Messig@iec.tu-freiberg.de>  2013-06-13 08:
Eintragender: Danny Messig <Danny.Messig@iec.tu-freiberg.de>  2013-0
Eltern: 70170852d1ac3d0f86cf6cbcb513cf75e9c51203 (adding pics and so
Eltern: [illegible] (Merge branch 'mast
Zweig: master, remotes/vgit/master
Folgt auf:
Vorgänger von:

    Merge branch 'master' of ssh://139.20.118.79/home/vgit/Lehrveran
```

$> git gui



Git Gui (SWP) /Users/messig/Documents/SWP

Aktueller Zweig: master

Nicht bereitgestellte Änderungen

11_Git/git.pptx

Verändert, nicht bereitgestellt

Datei: 11_Git/git.pptx

```
Binary files a/11_Git/git.pptx and b/11_Git/git.pptx differ
```

Bereitstellung (zum Eintragen)

Versionsbeschreibung:

[✓ Neue Version] Letzte nachbessern

Bereit.

slides & revisions & comments

all slides, revisions and some git history to review:
https://github.com/stweise/git_workshop_clt2016

useful commands, but not discussed here:

- stash
- clean
- fetch
- gc
- cherry-pick
- tag, describe
- blame
- grep