Version Control with Git

Pascal Escher[1], Philipp Thiel[2]

1 Centre for Innovative Care, University of Tübingen
2 Institute for Bioinformatics and Medical Informatics, University of Tübingen

April 08th, 2021

Agenda

Content of this pre-course

- Information about Git
- Principle function
- Basis commands
- 3 Task Sets in breakout rooms

Most important help

... btw ... do you know this awesome service?

... `https://lmgtfy.com`

FYI

The intention of this course is to bring the people with least IT skills to a point where they can start their masters courses in bioinformatics. If you're a fast one and manage to solve todays issues in a short time please be patient! Here are some interesting things you can read in the meantime:

- About Git:
  `http://tom.preston-werner.com/2009/05/19/the-git-parable.html`
- Microsoft and GitHub:
  `https://medium.com/@ow/microsoft-acquiring-github-is-a-good-thing-heres-w`
- Dilbert: `http://dilbert.com`
- PhdComics: `http://www.phdComics.com`

Git: Motivation

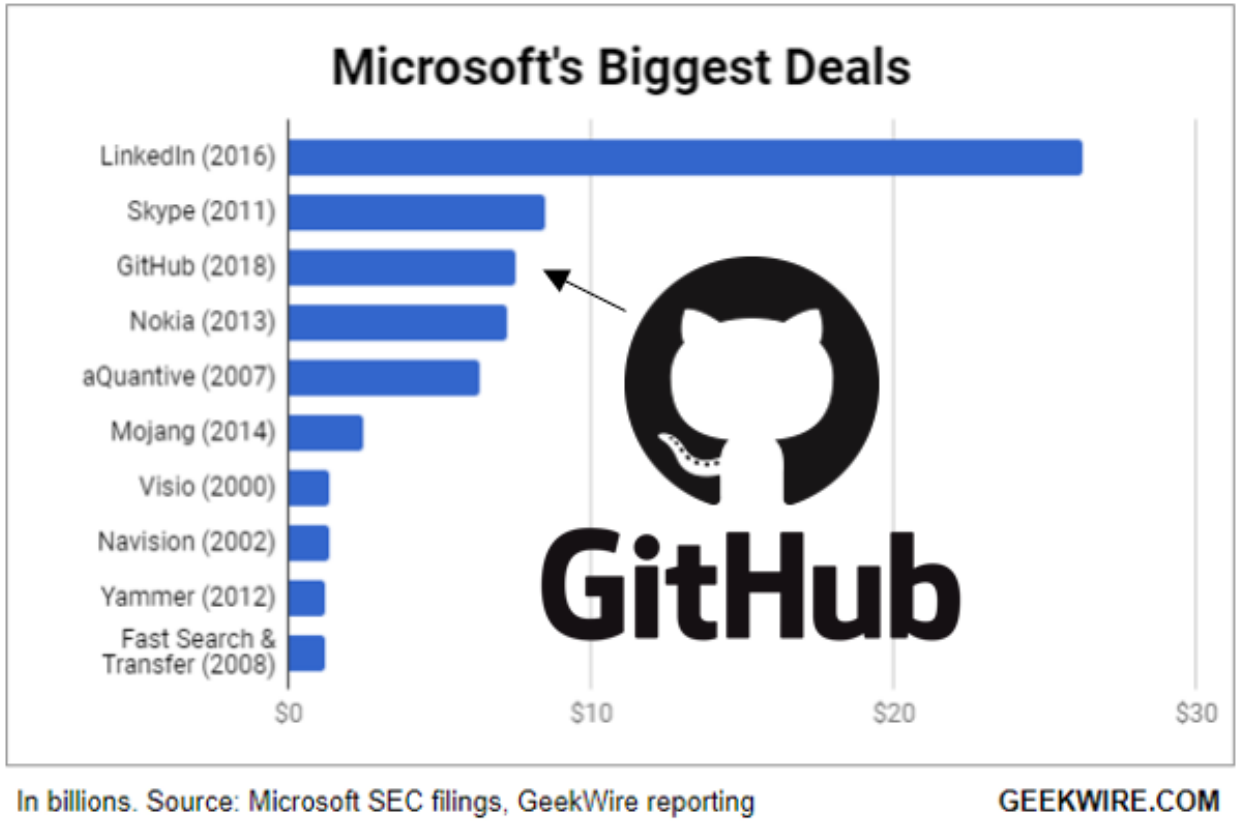
- Git is one software for VERSION CONTROL
- It's not the only one
  - Subversion (SVN)
  - Mercurial
  - Perforce
  - Deprecated: Concurrent Versions System (CVS)
  - https://en.wikipedia.org/wiki/List_of_version_control_software
- However, maybe it's nowadays the most popular one ...

Git: Inventor

One of the founder of Linux and Git: Linus Torvalds

Source: `https://cdn.britannica.com/`

Git: Github



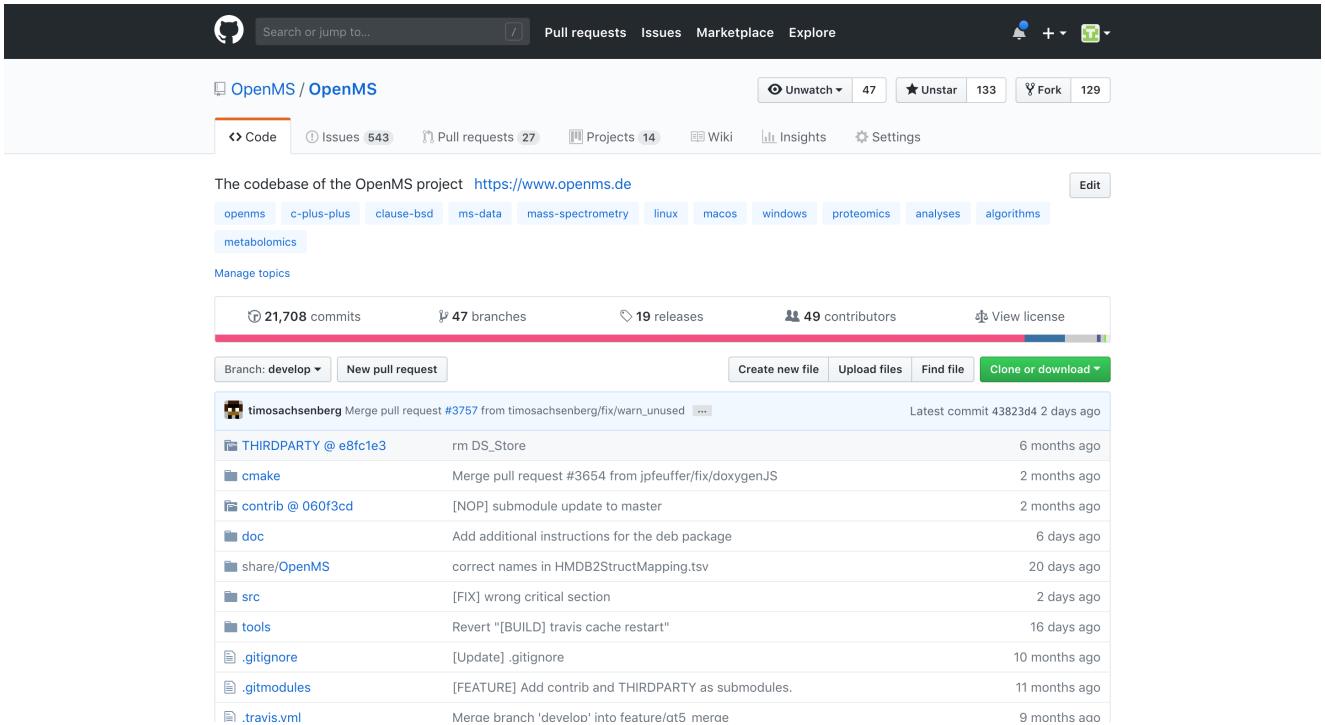
Microsoft bought GitHub in 2018 for 7,5 billion $

Source: `https://www.geekwire.com/2018/heres-microsofts-github-acquisition-ranks-among-tech-`

Git: Important Resources

- The docu and installer
  `https://git-scm.com`
- A nice tutorial
  `https://try.github.io/levels/1/challenges/1`
- Maybe to be read first
  `http://tom.preston-werner.com/2009/05/19/the-git-parable.html`

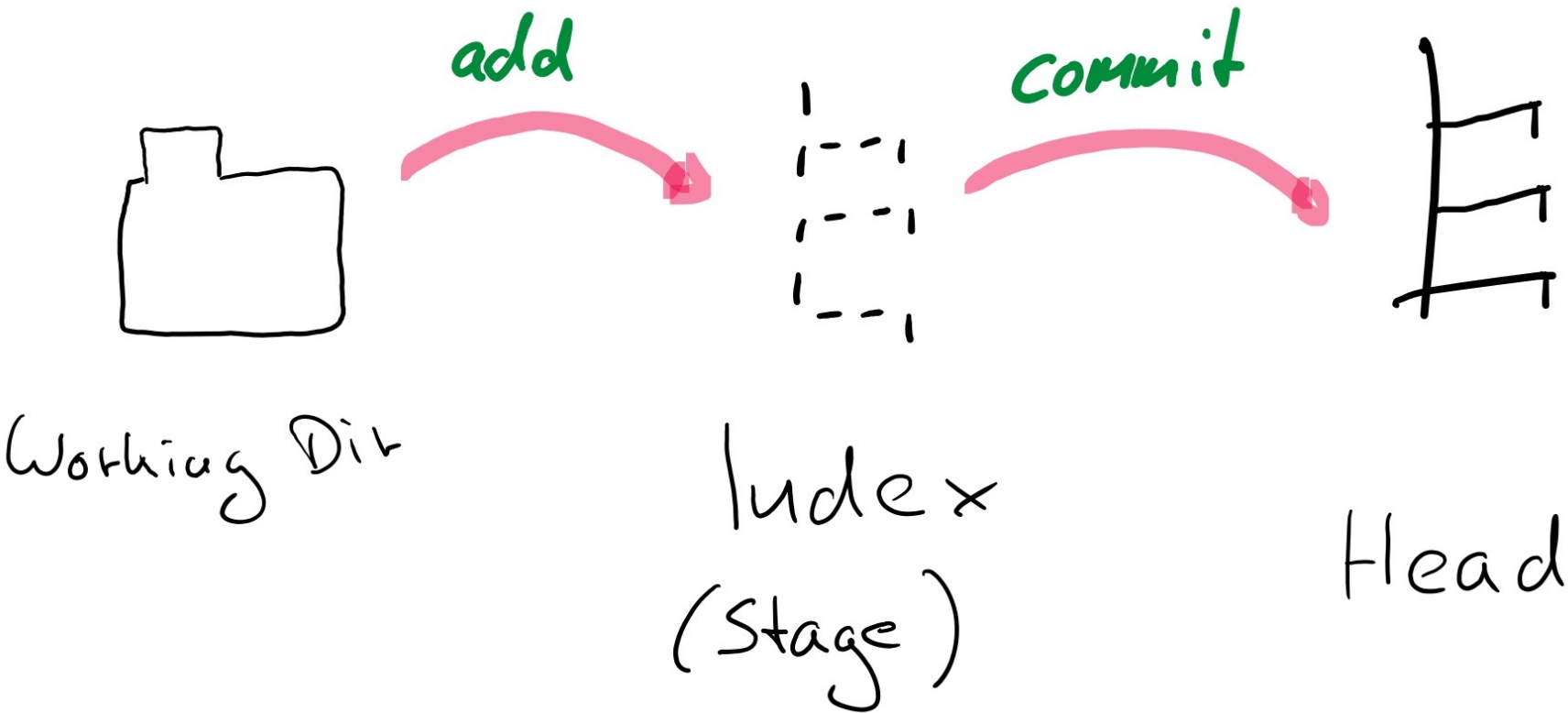
Git & Software Projects



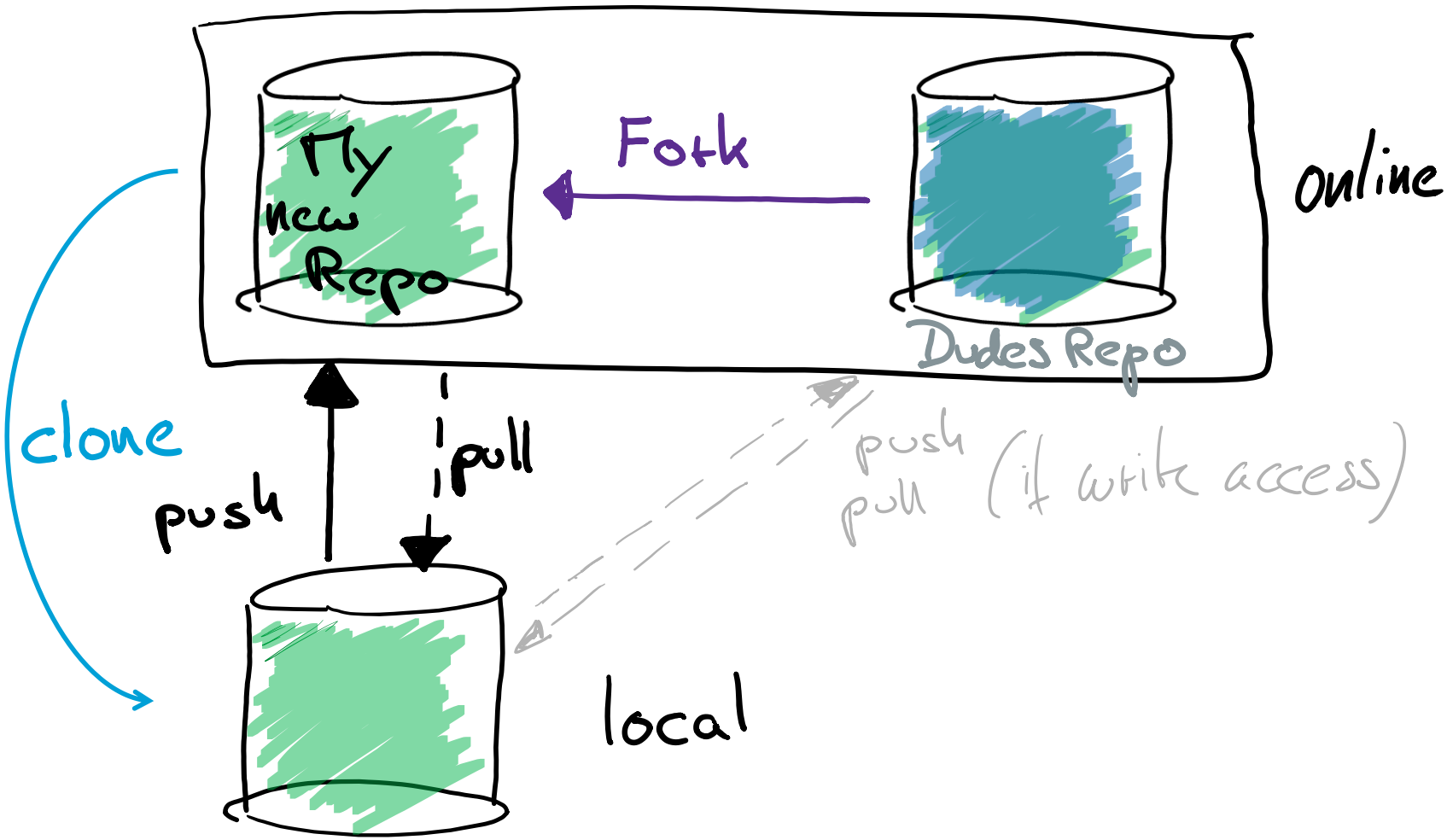
- Version Control is indispensable for bigger software projects with lots of developers, e.g. https://github.com/OpenMS/OpenMS
- Useful for smaller projects/application/scripts as well!

Git & Software Projects

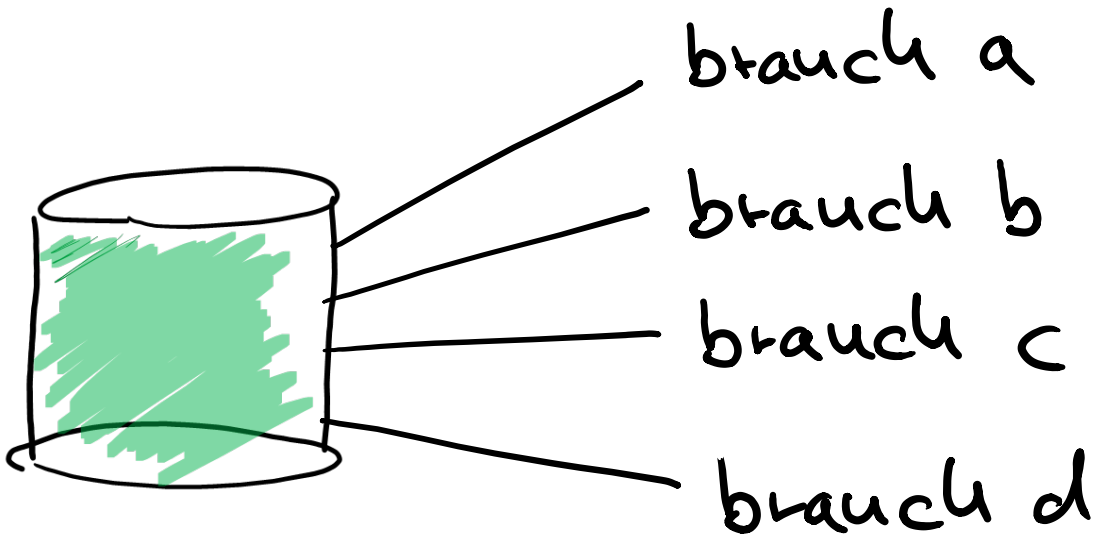
You can depict a local repository by three trees. The first one is the working directory, which holds the files. The second is an index which is used as staging area. The last one is head, which points to your last commit. Git saves the changes you make and stores it in the index/head based on adding and commit the changes.



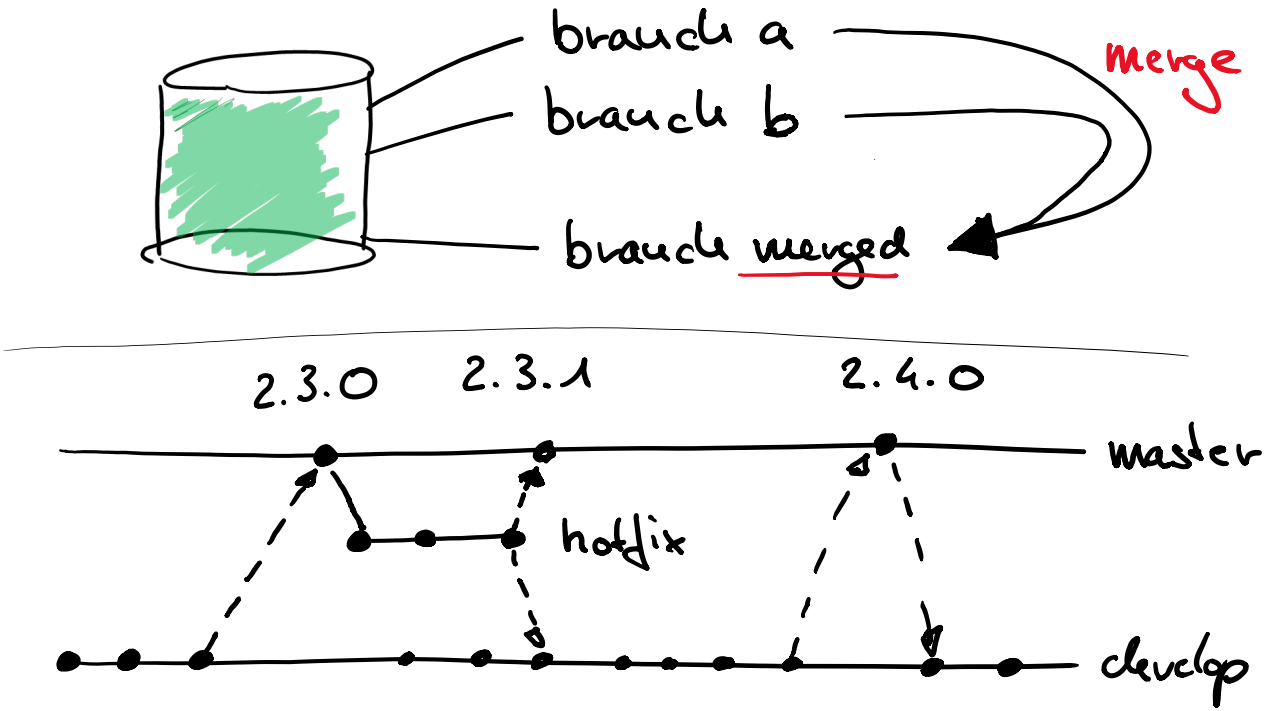
Git & Software Projects



Git & Software Projects



Git & Software Projects



Git & Software Projects

Prerequisites:

- Create an account for Github (https://github.com)
- Install git `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`
- Note: You can use the VirtualBox Image (Linux course) - if you like.
- Create a new empty directory and change into it.
- Note: If you run into a problem when using git, please read the info/error message in the terminal carefully - usually it is self explaining. If you get stuck - give google a chance. If nothing helps, don't hesitate to ask!

How to proceed

In the first set, we have a look at basic git commands.

We create a local repository and play around with it.

The following slides are separated into commands and Task sets.

Please have a peak at the Task Set first!

Then check the following ”important commands”, play around with them and try to solve the given tasks.

Git: Important Commands

```
$ git init
```

- Create a new empty repository
- Play around in the directory
- Check the git status

```
$ git status
```

Git: Important Commands

- Create a new text file and write something into it ...
- ... and check the status again

```
$ git status
```

Git: Important Commands

```
$ git add
```

- Purpose: Prepare changes for integration into your repo

```
$ git add -A
```

- Add everything that is not ignored

Git: Important Commands

```
$ git commit
```

- Purpose: Integrate added changes into your repo

```
$ git commit -m "important-commit-message"
```

- Commit - adding a commit message directly

```
$ git commit --amend
```

- Correct last commit wrt files and message

Git: Important Commands

```
$ git reset HEAD <file> $
```

- Unstage a file (Undo adding)

Git: Important Commands

```
$ git log
```

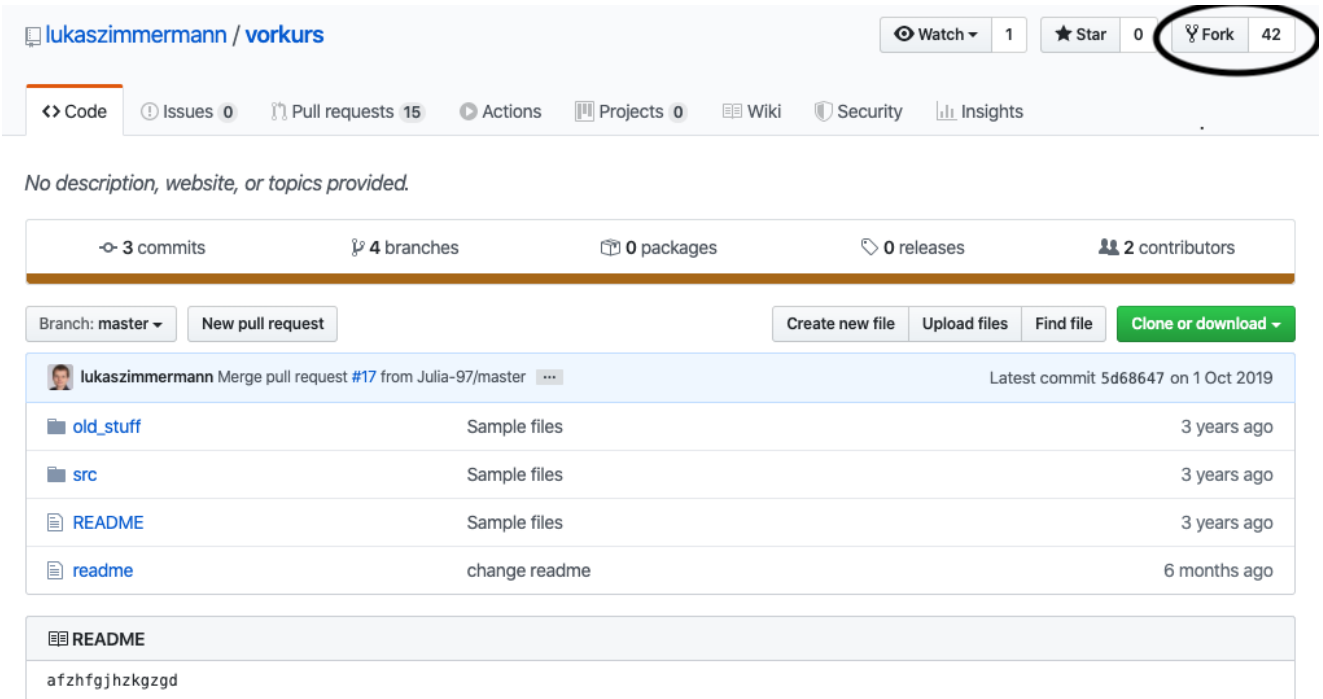
- Purpose: Check commit history

Git: Task Set 1

1. Download the following archive: `https://www.dropbox.com/s/y2bwsfvv2ctn6a9/awesome_project.zip?dl=0`
2. Extract `awesome_project.zip` somewhere in your file system
3. Initialize `awesome_project/` to a Git repository
4. Add source files, but not the config and .class files
5. Make a commit
6. Try to find out how to ignore config and .class files (So they won't be added with git -A)
7. If you are fast: Reset your repo to previous commit (Try to find out how to use git reset)

Git: Some Preliminary Work

- Purpose: Get a copy of our work project in your GitHub account (peek at Task2)
- To do this we fork the repository (repo) of interest
- Go the the online repository and use the fork button. Now you have a fork and can edit it as you like!



Git: Important Commands

```
$ git clone
```

- Purpose: Get a local copy of your remote repo (project)
- How does this work?
- Let's get help

```
$ git clone -h
```

Git: Important Commands

```
$ git status
```

- Purpose: Check for changes in your repo ...
- Well, you'll get it in a moment!

Git: Important Commands

```
$ git push
```

- Purpose: Upload your local changes into the remote *parent repo*
- There are different reasons to do this such as
  1. Save your work at a secure place
  2. Give others the chance to see your work
  3. Enable collaboration with others on the same work

Git: Important Commands

```
$ git config --global user.name "Your Name"
```

- Change/Add your name so everyone can distinguish the involved developers

```
$ git config --global user.email "name@domain.com"
```

- Add your email address so people can get in contact with you

```
$ git config --list
```

- Check your credentials

Git: Important Commands

```
$ git pull
```

- Purpose: Integrate changes from the remote *parent repo*

Git: Important Commands

```
$ git remote -v
```

- Lists all known remote repos

```
$ git remote set-url origin
git@github.com:githubusername/repository.git
```

- Change the remote of origin to given account and remote repository

Git: Important Commands

```
$ git branch -a
```

- See all branches that the local repo knows about

Git: Important Commands

```
$ git checkout <branch>
```

- Switch to branch

```
$ git checkout -b <new_branch>
```

- Create and switch to new_branch

```
$ git merge <branch-to-be-merged-into-current-one>
```

- Merge commits from a different branch into the current one

Git: Task Set 2

1. Fork the repository of Lukas:
   `https://github.com/lukaszimmermann/vorkurs`
2. Clone your forked repository
3. Checkout new branch for modification
4. Make commit to remove old_stuff/ (to the new branch!)
5. Push the branch to your remote
6. Add Lukas repository as a remote (Hint: git remote add)
7. Make a pull request of your changes to Lukas repository (online)

Git: Important Things Most Likely Omitted

- Branching - Why is branching so powerful and how is it used? Please check:
  `https://nvie.com/posts/a-successful-git-branching-model`
  `https://guides.github.com/introduction/flow/`
- Checking out other branches
- Merging changes from others

Mergeconflicts : TaskSet 3

- Fork
  `https://github.com/klarareichard/vorkurs_merging`
- Clone your Fork
- Create new branch modification

```
$ git checkout -b modification
```

- Insert ”Hello World” as first line into mergeconflict.txt, change second line to ”This line won’t cause a mergeconflict anymore”. Add an additional line: ”This is an additional line” to the end of the file.
- Add and commit your changes
- Switch to branch master
- Create another branch other-modification and switch to it

Mergeconflicts : TaskSet 3

- Change second line of mergeconflict.txt to ”This line will cause a merge conflict”. Add a file ”newfile.txt”.
- Commit and switch to master

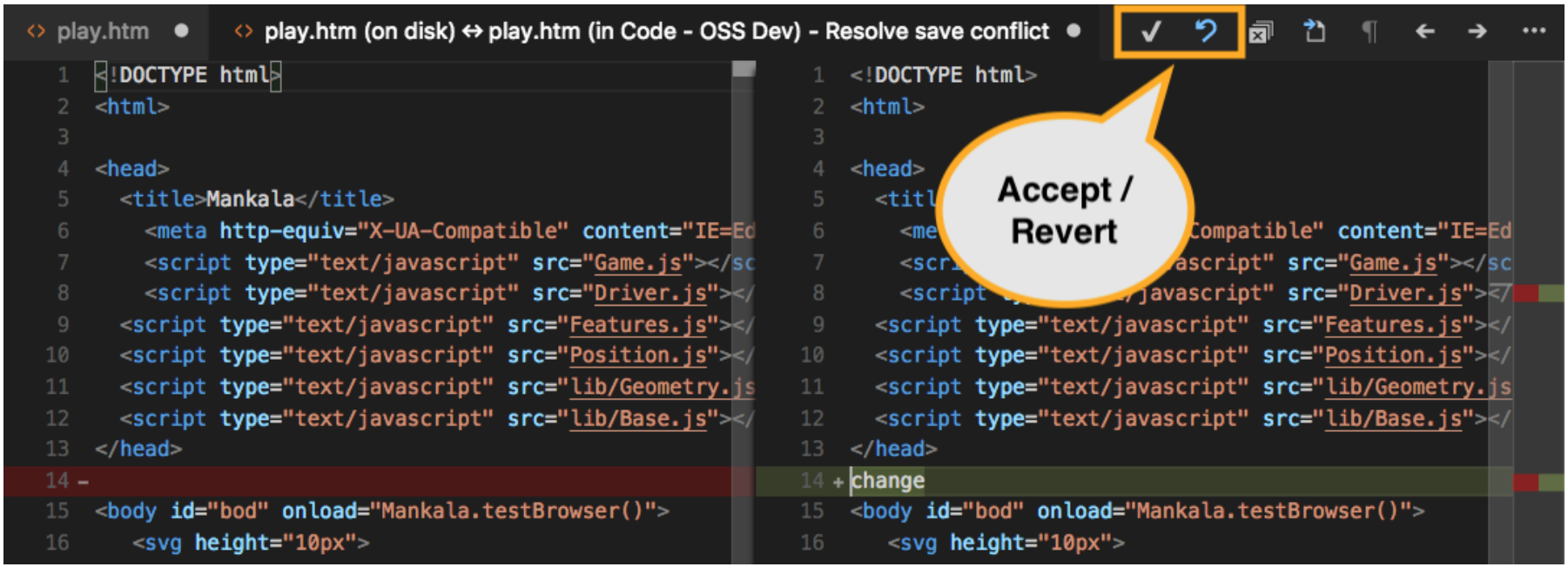
```
$ git merge modification
```

```
$ git merge other-modification
```

- Open mergeconflict.txt and resolve mergeconflict. Commit the result.
- Show branches and commits as a graph

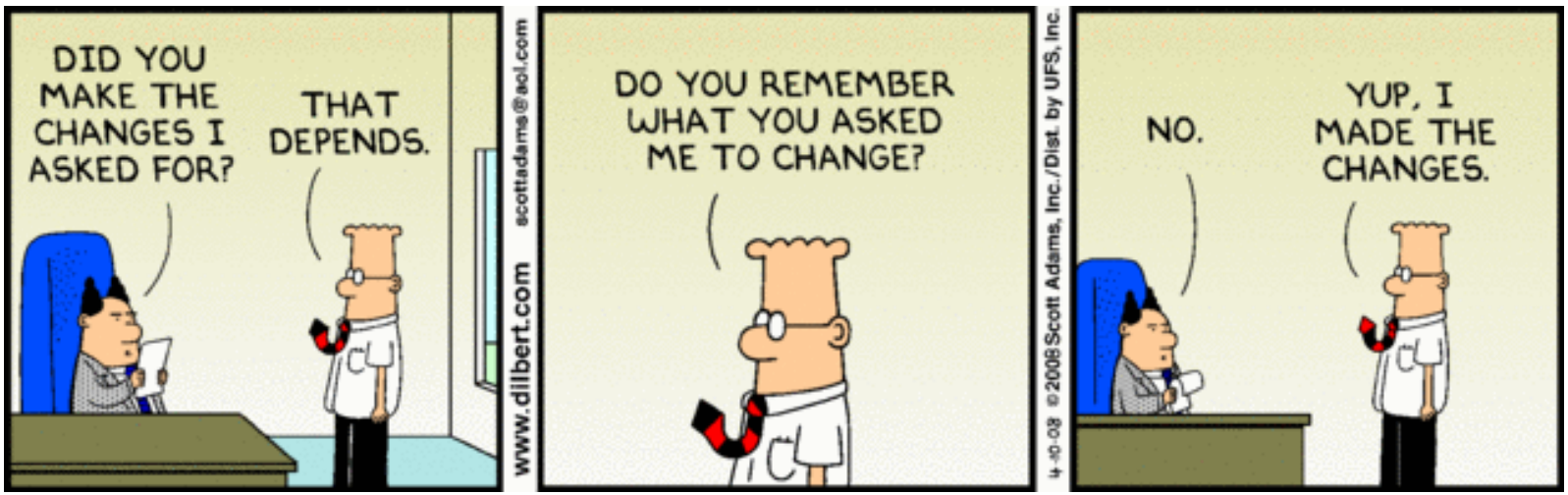
```
$ git log --graph --oneline --all
```

Git: Merging with IDEs



- Hint: Use IDEs for Merging: Visual Studio Code!
  Source:
  `https://code.visualstudio.com/docs/getstarted/tips-and-tricks#_resolve-merge-conflicts`

Thank you


DID YOU MAKE THE CHANGES I ASKED FOR?
THAT DEPENDS.
www.dilbert.com scottadams@aol.com
DO YOU REMEMBER WHAT YOU ASKED ME TO CHANGE?
4-10-08 ©2008 Scott Adams, Inc./Dist. by UFS, Inc.
NO.
YUP, I MADE THE CHANGES.

Acknowledgements

Original version of the Git Vorkurs was kindly provided by Lukas Zimmermann.

Thanks to all constributors:

- Lukas Zimmermann
- Oliver Alka
- Simone Lederer
- Pascal Escher