

Version control with Git

Pascal Escher

Centre for innovative Care, University Hospital Tübingen

April 5th, 2022

Agenda

9:00 - 9:10	Introduction to Git
9:10 - 9:30	Principle function & basic command
9:30 - 9:45	1 st task set
9:30 - 9:40	10 Min ☕ break
9:40 - 10:20	2 nd & 3 rd task set

Most important help



Goal of course: bring people with least IT skills to a point where they can start their masters courses in bioinformatics.

For fast learners: Here are some additional interesting articles

- 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 a software for VERSION CONTROL
- What happens if you delete the file by accident?
- Sending files back and forth get out of sync

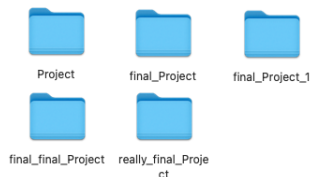
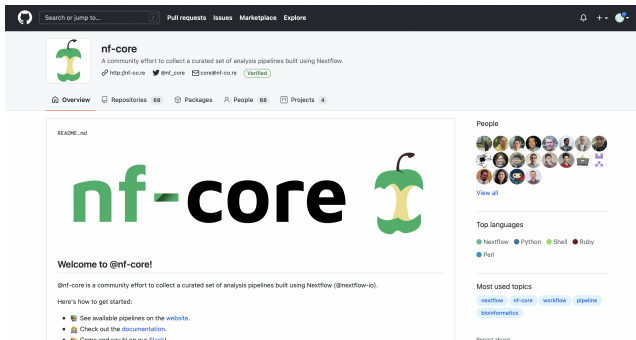


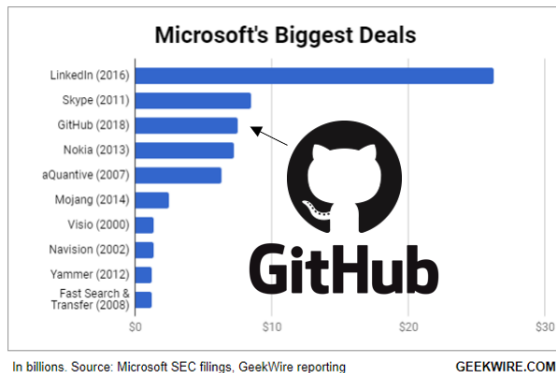
Figure 1: This is not version control!

Git & Software Projects



- Version Control is indispensable for bigger software projects with lots of developers, e.g. <https://github.com/nf-core/>
- Useful for smaller projects/(thesis) application/scripts/ as well!

Git: Github



Microsoft bought GitHub in 2018 for 7.5 billion \$ (source)

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>
- Contribution basics
<https://www.youtube.com/watch?v=gTEXDXWf4hE>
- Git Cheat sheets
<https://www.cems.uwe.ac.uk/~jd7-white/images/git-cheat-sheet-education.pdf>

Git = based on tree model

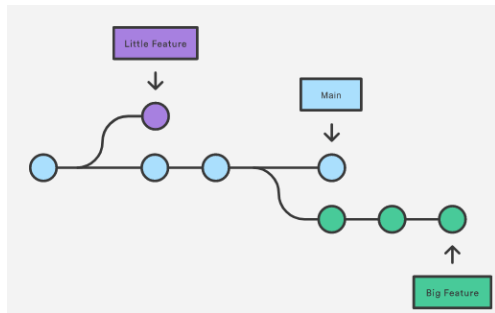
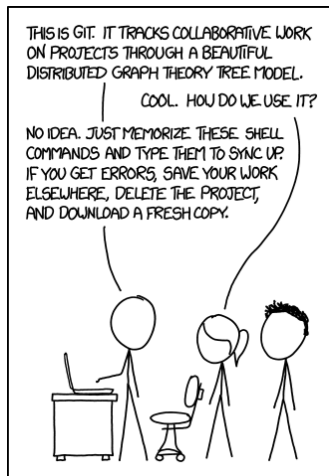


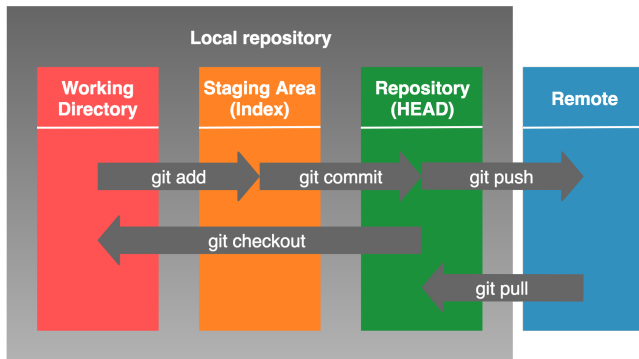
Figure 2: Git works like a tree with a root (master) and multiple branches



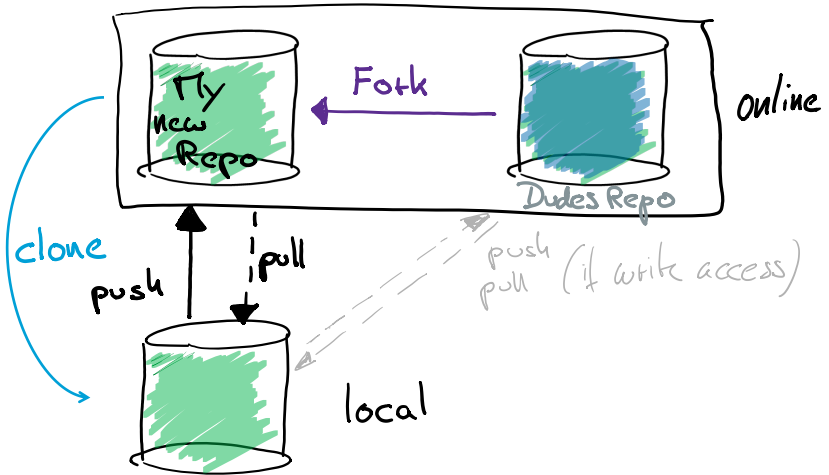
Git & Software Projects

Local repository = 3 trees

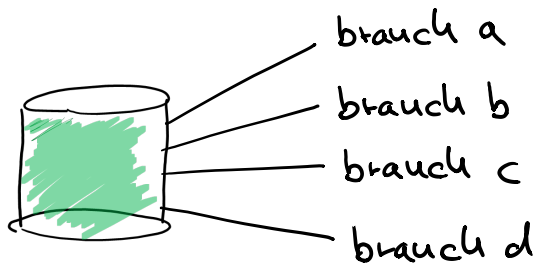
- 1st tree = working directory
- 2nd tree = index (staging area)
- 3rd tree = head, which points to last commit



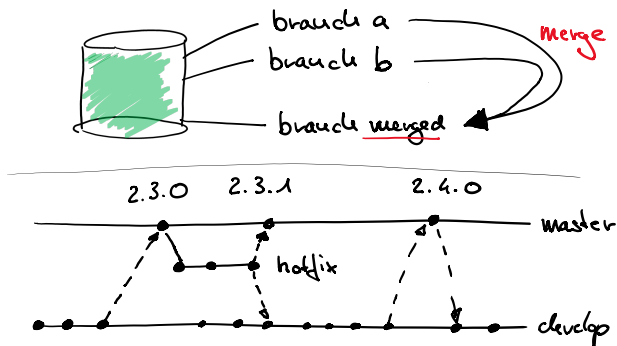
Git & Software Projects



Git & Software Projects




Git & Software Projects



Git & Software Projects

Prerequisites:

- Create an account for Github.com 
 - <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
 - Create a new empty directory and change into it.
- 💡 You can use the VirtualBox Image (Linux course) - if you like.
- 💡 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" and
try to solve the given tasks.

- 💡 For people with a visual mindset, activate invisible files in your OS:
 - Windows: Show all hidden files via View>Show>Hidden items
 - MacOS: Cmd + Shift + . (Point)
 - Terminal: ls -a, Win/(Power)Shell: dir -a

Git: Important Commands

```
$ git init
```

- Create a new empty repository
- Play around in the directory

```
$ git status
```

- Check 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 committed files and message

Git: Important Commands

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

- Unstage a file (Undo adding)

Git: Important Commands

```
$ git log
```

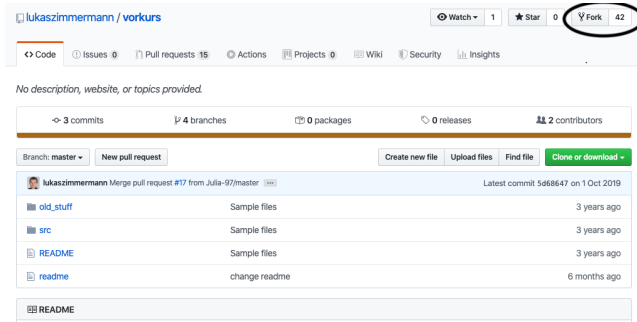
- Purpose: Check commit history

Git: Task Set 1

- 1 Download the following archive: https://www.dropbox.com/s/y2bwsfvv2ctn6a9/awesome_project.zip?dl=0
💡 A dropbox account is not needed
- 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 to 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 ...

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 - Authentication

- **At home:** `https://docs.github.com/en/authentication/connecting-to-github-with-ssh/checking-for-existing-ssh-keys` **(Then clone with ssh)**
- **Via a "Personal Access Token":**
`https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token` **(Clone with HTTPS)**

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

Become a professional

- Use an IDE, e.g. IntelliJ
- All Git command built-in available
- Color highlighting for files associated with Git
- A terminal is included!

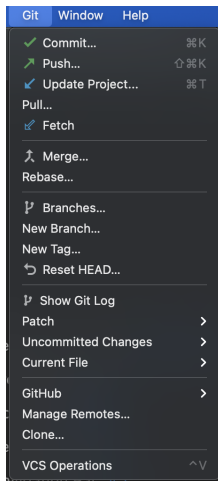


Figure 3: ? Just the build-in terminal first!

Git: Task Set 2

- 1 Fork the repository of Lukas:
`https://github.com/lukaszimmermann/vorkurs`
- 2 Open IntelliJ and have a look on the Git menu
- 3 Clone your forked repository
- 4 Checkout new branch for modification
- 5 Make commit to remove old_stuff/ (to the new branch!)
- 6 Push the branch to your remote
- 7 Add Lukas repository as a remote (Hint: `git remote add`)
- 8 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

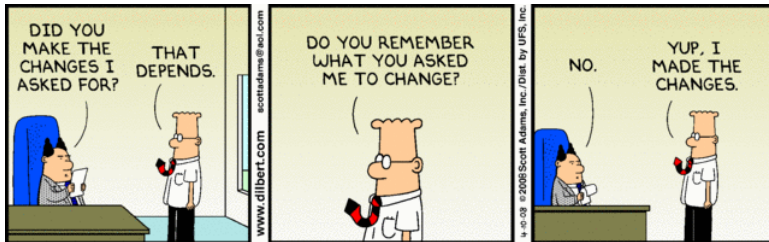
- `$ git merge modification`

- `$ 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
```

Thank you



Acknowledgements

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

Thanks to all contributors:

- Lukas Zimmermann
- Oliver Alka
- Simone Lederer
- Pascal Escher
- Antonia Schuster, Friederike Hanssen