UNIVERSITÄTSKLINIKUM AUGSBURG

Universität Augsburg University

Introduction to Github

Sarra Zaghdoudi
Dr. Mostafa Sheikhalishahi

Institute for digital medicine

Dec 9th 2022

Objective

1. GitHub
2. GitHub Collaborative Coding

GitHub

- Version Control
- Git and GitHub
- Repositories
- Branch
- Merge

Version Control ?

Version Control ?

- Keep track of the changes made to our files at discrete points of time “snapshots” and who made them.**“Understanding”**
- Keep track of history and different versions of the project. **“Versioning”**
- Ability to revert to older versions, retrieve deleted information or determine when a bug was introduced. **” Rolling back ”**
- Share work with others, collaborate and sync easily. **“ Collaboration”**

Git?

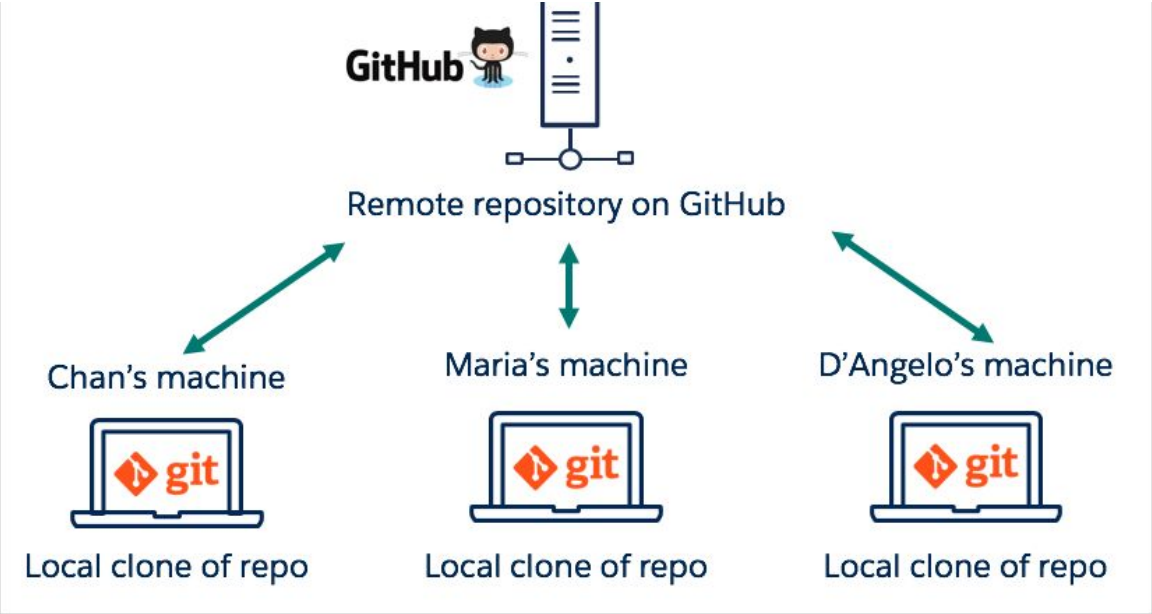
- A **Version Control System** (VCS)
- Linus Torvalds, 2005
- Has a **distributed architecture**: Every developer has a copy of the whole working directory on his local machine
- A **standalone**: Works as a server or as a machine with or without internet connection
- **Not centralized**: Only used locally!!
- How to collaborate? ⇒ We need to set Git on a server machine acting as a Hub for all users

GitHub?

- A web based git repository hosting service
- VCS + Extra Features
- Free access to a git server for public and private repositories
- It lets you and others work together on projects from anywhere
- Share and access repositories on the web
- Download or clone repositories to your local machine and work on them
- Cloud based storage space
- A good and safe way to collaborate
- Similar : Bitbucket, GitLab ..

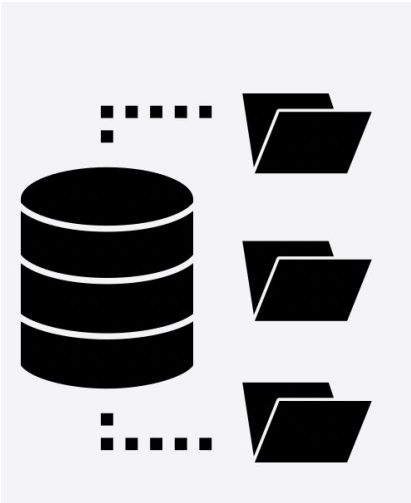
Git & GitHub

- You can use Git without GitHub but you cannot use GitHub without Git

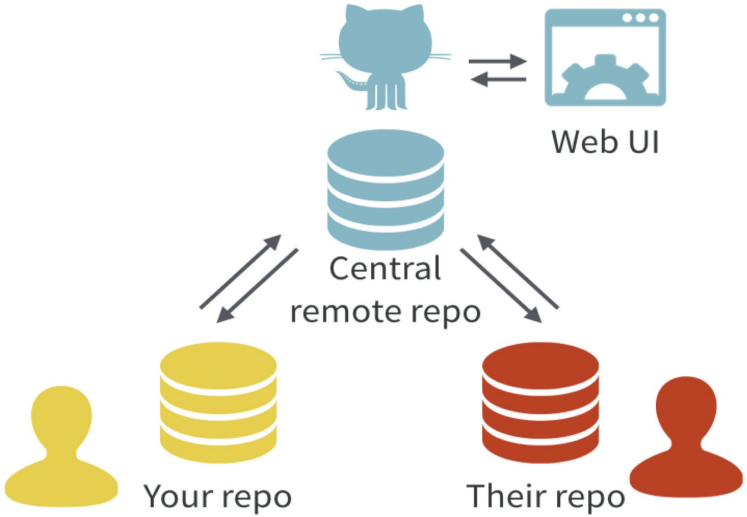


Creating repository

- $ **git -init** : To initialize a new project
- **.git/** folder is created ⇒ The git repository
- A Git repository **tracks** and **saves** the history of all changes made to the files in a git project.



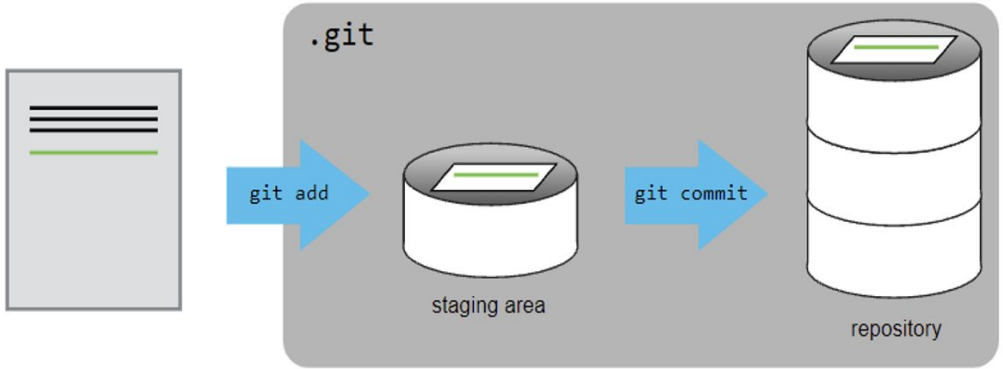
Remote and Local repositories



- All contributors have a copy of the repo, with all files and the full history.
- GitHub provides access to the repo through a web browser.

Tracking files

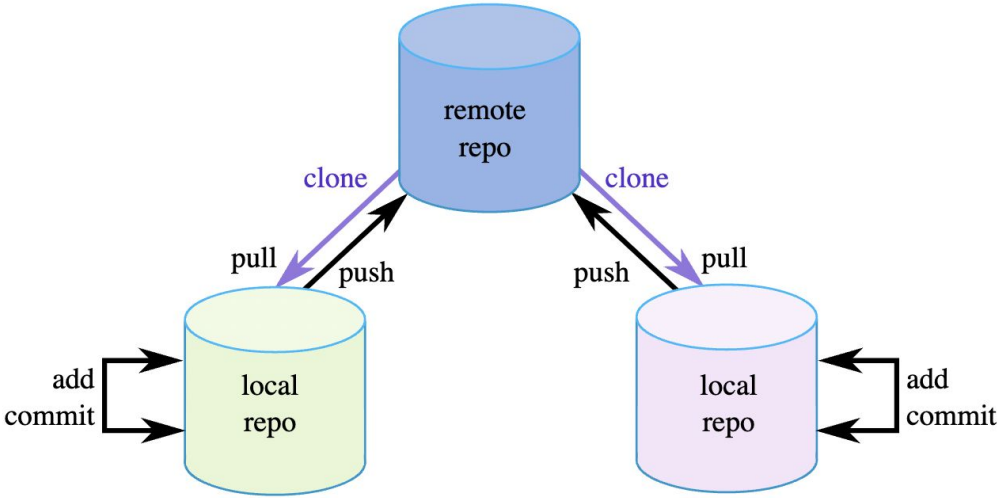
- Three area in a git project:
  - **Git directory** History of all the files and their changes
  - **Staging area** Changes marked to be included in the next commit
  - **Working tree** Current state of the project
- A file can be : Modified, staged or commited (snapshot)



Tracking files

- To commit a file, a.k.a, save the changes as a snapshot we use:
  - **git add** filename ⇒ Put the file in the staging area
  - **git commit - m** ‘commit message’ ⇒ To save the staging area content in the repository
  -
- It is important to use clear commit message that reflects the new change as well as its importance so that other developers can understand
- **git commit -a -m** could be used as a shortcut to stage any changes to tracked files, already committed before, and commit them in one step

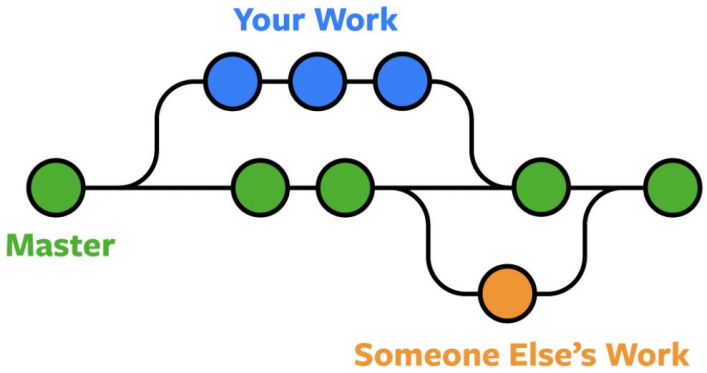
Work with remote



- **git clone** 'repo_url' to download a copy of the remote repo
- **git pull** to synchronize the local repository when change are done remotely
- **git push** to update the remote repository with the changes made locally **(after git add and git commit)**

Branch

- A branch is a pointer to a commit
- The project's main branch is called Master, created when we use git -init
- There can be be many branches, every time we commit a new node is added
- Usually a branch is created when we want to introduce new feature



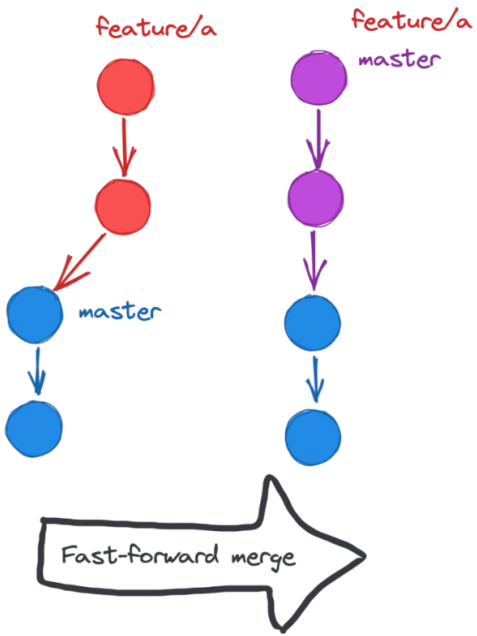
Branch

- **git status** to see in which branch you are (marked by *)
- **git branch** to see local branches
- **git branch -r** to see remote branches
- **git checkout -b new_branch** to create and switch to a new branch
- **git checkout branch_name** to switch to a new branch
- **git pull** to get remote branches
- **git push -u** origin my-branch-name to push your new branch to remote

Merge

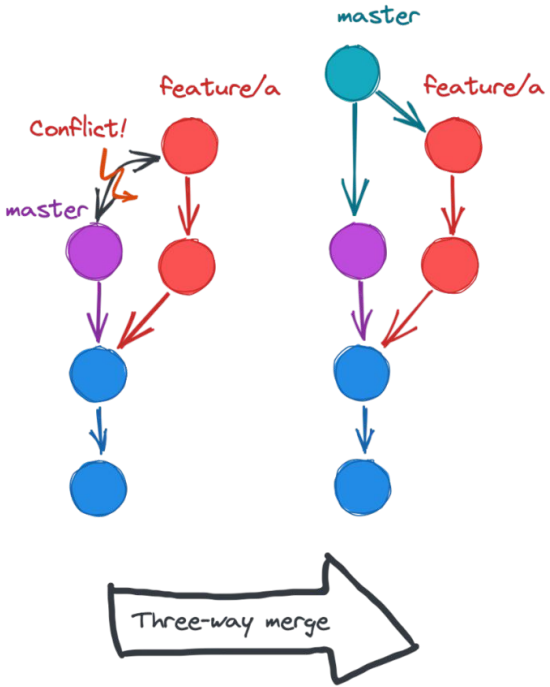
- Once the work on the branch it is complete, it is time to merge it back into the main branch , the master branch
- Merging takes the separate branch changes and implements them back into the main branch
- Depending on the commit history, git performs two merging algorithms : **Fast-forward** and **Three-way-merge** (case of conflict)
- **Conflict:** When at least two people edit the same part of the same file at the same time.
- **git merge**

Fast-forward merge



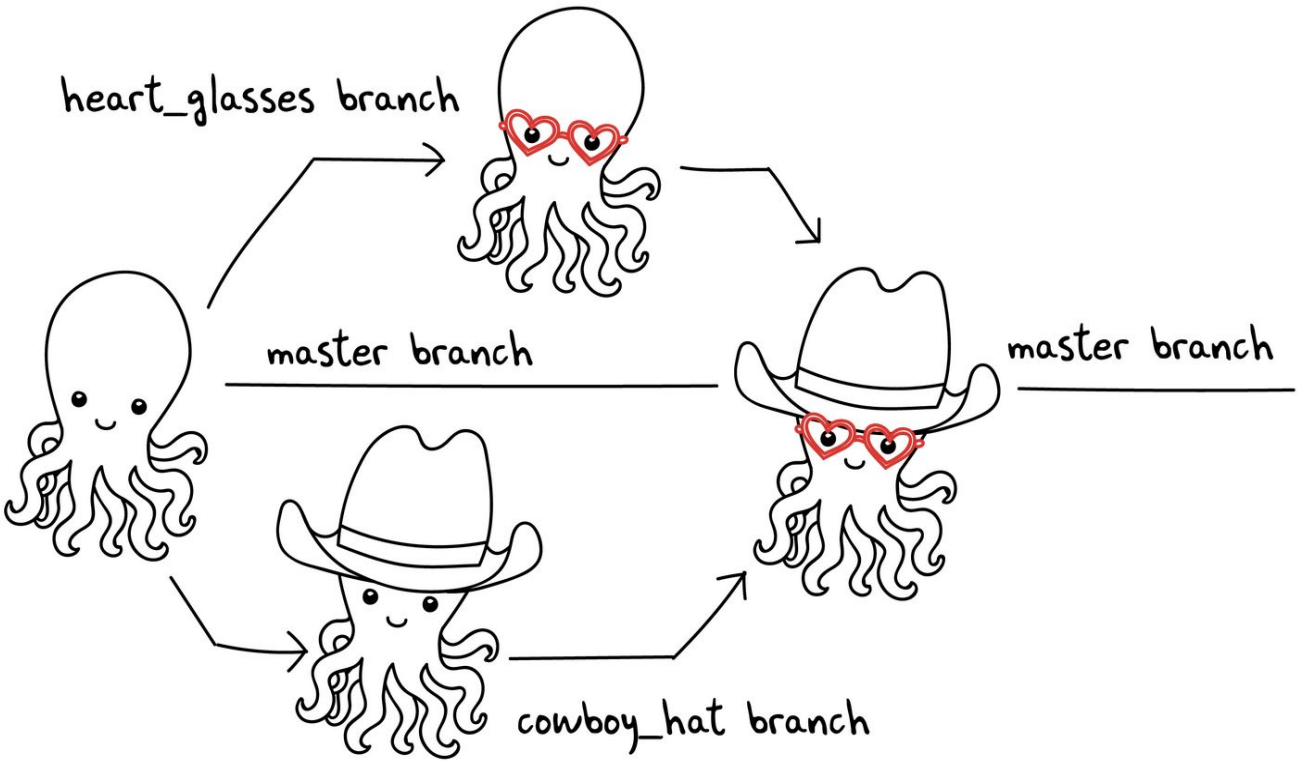
- By default, if there is no conflict
- When the history of commits is linear
- The master branch reference is being updated, such that it now points to the same commit as the feature/a branch.
- Done automatically

Three-way merge



- In case of a conflict
- Git will ask you first to solve the conflict
- Update the file causing the conflict before merging
- Better than risking Information loss

Branch and Merge



GitHub Collaborative Coding

- GitHub Codespaces
- GitHub Discussions
- GitHub Copilot

GitHub Codespaces

- A codespace is a **development environment** that's hosted in the **cloud**.
- Each codespace runs on a **virtual machine** hosted by **GitHub**.
- Let’s check them out

GitHub Discussions

- **Collaborative communication forum** for the community around an open source project.
- Community members can ask and answer questions, share updates, have open-ended conversations
- Let's check them out

GitHub Copilot

- **AI pair programmer** that offers autocomplete-style suggestions as you code.
- Powered by **OpenAI Codex**
- GitHub Copilot is trained on all languages that appear in public repositories
- Languages with less representation in public repositories may produce fewer or less robust suggestions.
- Let's check it out

Your turn!

- Create a repository with a readme file and active discussion
- Modify the readme file and commit it
- Open discussion and discuss with your colleagues
- Create a jupyter notebook in your repository and run it using codespace
- Clone a repository and follow the given instructions during the lecture

Useful resources

- https://docs.github.com/en
- https://docs.github.com/en/get-started
- https://docs.github.com/en/codespaces
- https://docs.github.com/en/copilot
- https://www.coursera.org/learn/introduction-git-github/home/

Thank you!