Summarized Learnings on GitHub

Source: https://www.youtube.com/watch?v=eulnSXkhE7I&t=3023s

I learned that GitHub is a control and management tool for software that allows keeping track of changes in a code base and helps synchronizing it between different people. It allows to independently make changes to the code and even merge changes from different people together. It also provides a version control system that makes it easy to test changes to the code and revert back to older code versions if needed. Additionally different branches can be created to develop features independently and merged to the main branch when they are ready. The code for a project is organized inside a repository (that acts like a folder) and is uploaded to a server that can be either accessed from everyone on the internet or only by people authorized for this repository.

After setting up a repository (with an GitHub account) and installing git on a local computer, git can be easily used through a cmd using these commands:

- git clone → copies/downloads a repository to my local machine
- git add → adds code files to be tracked by git which will be uploaded when committing code
- git rm → removes a file from tracking (opposite of git add)
- git commit → commits code changes with a short description of them
- git status → prints info about the branch that the user is on and if there are open commits (not pushed)
- git push → uploads all code changes to the server that were committed
- git pull → downloads the newest code version that is on GitHub to my local machine
- git log → shows history of commits (with commit-hashes = IDs)
- git reset → resets the repository to an older code version
- git branch → shows a list of all branches and marks the one that the user currently works on "*"
- git checkout → switches to another branch (that can also be created by this command) to work on this branch
- git merge → merges different branches together
- pull request (in browser) → request other contributors to merge code from another branch

In some cases conflicts can appear when GitHub tries to merge different parts of code. In this case, the affected code part is marked in the code and the problem can be resolved by deciding which lines of conflicting code are correct and can be kept and which should be deleted. Also the markers in the code have to be deleted and the corrected code must be pushed to the server afterwards.