Exercise Set 1
Introduction to Git

University of Oslo - IN3110/IN4110

Fall 2020

Note: These exercises are not mandatory. You don't have to put your work into your Github repository.

Exercise 1: Git Essentials

The git commands you will use the most, and therefore should be comfortable with, are the following:

- `git status`
- `git add`
- `git commit`

The simplest git workflow is to edit your files, stage them (using `git add`) and then commit them. In between these steps, you can use `git status` to see what files have been changed since the last commit, and also see which files have been staged.

Create a new git repository on your machine by running `git init` in a directory of your choice[1]. Create some file(s) , for example `exercise_1.txt`, and commit them. Change some of the files, for example by adding `I changed the file.`, and commit your changes. Repeat this, until you feel like you know the drill. Remember to use `git status` both before and after staging and committing to see what's going on. When you are done, use `git log` to see the history of your repository.

Congrats - You just finished the 1+1 of Git!

[1] What about something as simple as `IN3110_2020`?

Exercise 2: Back to the Future in Git: Looking at Old Versions of your Repository

Now that you have a repository with some history, we can begin looking at some benefits of a version control system. If our code ever breaks, we can always go back to a previous commit where it works (How great is that?!).

If you haven't already, look at the history of your repository with `git log`. (For a more condensed view, use the `--oneline` option).

Suppose we want to see what the repo looked like one commit ago. We can refer to that commit a couple of different ways: By identifying the commit in the log, we can find its hash (the first line in the log entry), and check it out with `git checkout COMMITHASH`. An easier way to do this is by referring to it as the previous commit with `HEAD~1`. `HEAD` always refers to the last commit in the currently checked out branch. When doing this, you are in "detached head state" and changes you commit will not belong to any branch.

It is also possible to checkout single files. You will do this in the mandatory assignment.

To get back to the latest commit, use `git checkout master`

Exercise 3: Stepping up the Game

Git has a lot more functionality than what is possible to cover in these exercises. Have a look at the documentation for some git commands. Here are some suggestions for commands, and some useful parts of the documentation:

- `git status`: The `-s` flag
- `git add`: The `-u` flag. What is the difference between running `git add .` and `git add -u`?
- `git log`: The `--oneline` and the `--graph` options
- `git commit`: The `-a` flag

See if you can figure out what these flags mean by checking out the documentation!

The documentation can be found in the man pages (e.g. `man git add`) or online at `https://git-scm.com/docs/git-add`

Some other commands you might find useful in this course, are `git branch` (along with `git merge`) and `git stash`. Try to find out what they do, and why they are useful.