Version Control with Git

1 Getting Started with Git

1.1 Introduction

This exercise can be done on any computer running Linux. Using the *Raspberry Pi* is perfectly fine. It is recommended to also learn how to use Git on Windows with the Git Bash. However this is not covered in this exercise since its practically the same as on Linux. More information on how to use the Git Bash is found at https://git-scm.com/.

Git is a command line tool. There are multiple frontends and IDE plugins for a graphical usage. It is recommended to learn the command line interface first since it is easy, very efficient and the same on all major plattforms.

1.2 Installation and Basic Configuration

On a default *Raspberry Pi OS* installation, Git is already installed. Its version can be checked with the version option shown on the third line in listing 1.
If Git is not installed, the first line in listing 1 shows how to install it on a Debian based Linux like Ubuntu or Raspberry Pi OS.
A minimal configuration must be done either way so Git knows who the user is. This is done by the lines five to seven. The real name or also a pseudonym can be used but the email address should of course exist.

```
1 sudo apt-get update && sudo apt-get install git
2
3 git --version
4
5 git config --global user.name John Doe
6 git config --global user.email john.doe@ost.ch
7 git config --global core.editor nano
```

Listing 1: Installation and configuration of Git using the command line interface

2 Exercises

2.1 Tutorial

The first exercise is a tutorial. It is found at: https://gitlab.ost.ch/tech/inf/public/learn-git/-/blob/main/tutorial/README.md.

It is recommended to actively work through the tutorial and also to try out all the steps.

2.2 Group Work - Develop with a Fork

This is a group work for which two to three participants are recommended. Before this exercise can be started, each participant should own a GitHub account. There should be a repository, which was cloned to the local machine. A few commits should be made and pushed to the GitHub Repo.

To Do:

1. Learn with online resources, how the *Develop with a Fork* workflow works. A possible source is the following video: https://www.youtube.com/watch?v=nT8KGYVurIU
2. Each participant should create a **Fork** of a repository of a co-student.
3. The fork is then cloned from the own repo (not from the co-student one) to the local machine.
4. A few changes should be made and commited.
5. The commits are then pushed to the remote repository (GitHub platform).
6. On the GitHub platform web-UI, a *Pull-Request* to the Repo which was forked is created. Check out what detailed information GitHub provides in the Pull-Request.
7. The co-student should accept it by merging the changes in his repository.
8. To update the local repository on the development machine, the *git fetch* and *git pull* commands will do the work. Browse the documentation if needed at https://git-scm.com/doc.

2.3 Good practices - Guidelines

Good commits matter - or in other words:
If the developers do messy commits, Git will not really help to version control the software.

Read the guideline and also the two linked articles at: https://wiki.eeros.org/for_developers/commit_guidelines. Afterwards, answer the questions below and discuss them with your co-student.

1. Why shoudn't I put a bugfix and a logical change into the same commit?
2. Is it ok to fix a few typos when doing a layout change commit? Why?
3. Why should a commit message be well formatted?
4. What is a commit message for? What does it communicate?
5. For whom do I write a good commit message?