# Training Unit 4: git

## 1 INTRODUCTION TO TRAINING UNIT

After learning the theory of version control with Git in the lecture, it's time to gain hands-on experience with the tool. Since Git works best with text-based files such as *.c, *.h, *.py, and *.txt, some of Git's built-in functionality won't fully apply to many MATLAB file formats like *.mlx, *.mat, *.mdl, and *.slx. These file types should be treated as binary files (see the .gitattributes file for details).
To properly work with Git in MATLAB, make sure to follow the "Set Up Git Source Control" guide and the "Use Git in MATLAB" tutorial provided by Mathworks.
If any issues arise during the training, consult the Matlab documentation online or press the "F1" key for access to a variety of examples and copy-paste code.
By the end of this unit, you will be able to use Git within a MathWorks project. This includes adding files, viewing file changes, tracking change history, creating, merging, and deleting branches, and pushing/pulling changes.

## 2 PREPARATION

- For this unit you will need Matlab / Simulink
- git installed on your machine and accessible via terminal (git bash)
    - make sure git is correctly installed and works properly, before starting with this training unit

## 3 TECHNICAL BACKGROUND – GIT [1][2]

git is a distributed version control system (VCS) that tracks changes in files and enables users to collaborate on projects. While git is widely used by software developers, it can also be applied to other types of projects, such as managing documents or presentations.
git operates by creating snapshots of a project's state at different points in time. These snapshots, called commits, each have a unique identifier. git keeps a history of all commits made to the project, making it easy to revert to a previous version if needed.
git also supports branching, which allows users to work on different versions of a project simultaneously without affecting each other. Once the work on a branch is complete, it can be merged back into the main branch of the project.

A basic Git workflow consists of the following steps:

1. **Clone the repository**: This creates a local copy of the remote repository on your machine.
2. **Make changes**: Edit or modify the code or files as needed.
3. **Stage the changes**: Use Git to select which changes you want to include in the next commit.
4. **Commit the changes**: Create a new snapshot of the project's state by committing the staged changes.
5. **Push the changes to the remote repository**: Upload your changes to the remote repository, making them available to others who have cloned the project.

Git is a powerful tool that helps you manage your projects more effectively. It's also a valuable skill for anyone working in software development or fields that require collaboration.

Some of the benefits of using git:

- **Collaboration**: git makes it easy to collaborate on projects with other users. You can share your code with others, and they can make changes and push them back to the repository.
- **Version control**: git keeps track of all the changes that have been made to a project. This allows you to easily revert to a previous version of the project if necessary.
- **Branching**: git supports branching, which allows you to work on different versions of a project without interfering with each other.
- **Efficiency**: git is a very efficient version control system. It can handle large projects with many users without any problems.
- **Distributed Development**: Each developer has a complete copy of the repository, providing autonomy and allowing work to continue even if the central server is down.
- **Offline**: It is possible to work without internet connection. Only when pushing / pulling from the remote repo, an internet connection is required.
- **Community Support**: git has a large and active community, which means extensive resources, tutorials, and help are available online.

Some of the drawbacks of using git can be:

- **Learning Curve**: For newcomers, git can be a bit challenging to grasp initially, especially the more advanced features.
- **Command Line Interface (CLI)**: While GUIs exist, some operations are more efficiently performed through the command line, which might be intimidating for those less comfortable with it.
- **Binary Files**: git is optimized for handling text files. Managing large binary files can lead to increased repository size and slower performance.
- **Merge Conflicts**: In collaborative projects, conflicting changes may arise when merging branches. Resolving these conflicts can be time-consuming, especially for git beginners.
- **Storage Size**: git repositories can become large over time, especially for projects with a long history, potentially requiring significant storage space.

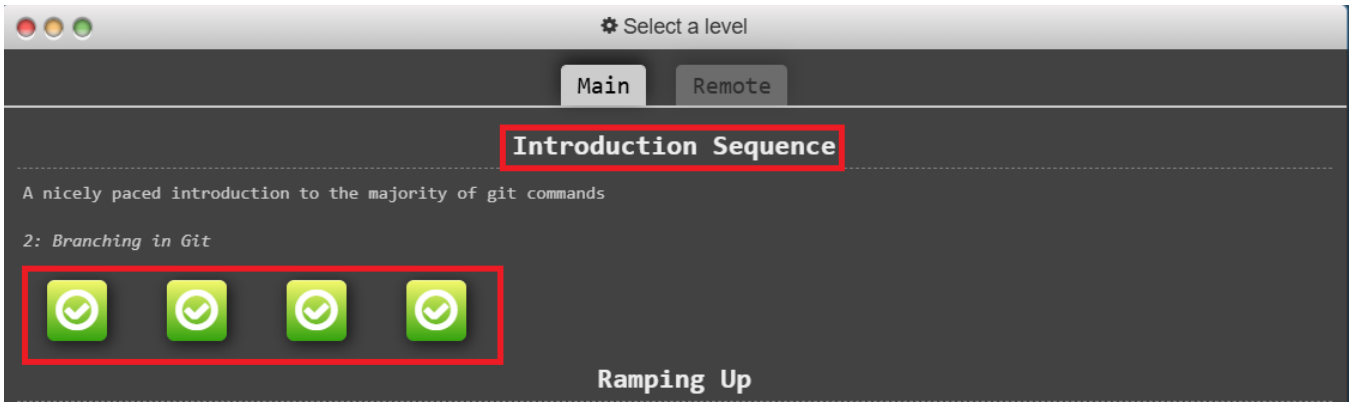
Despite its drawbacks, git remains an indispensable tool for modern software development, providing a robust and flexible foundation for version control and collaboration.

# 4 LABORATORY WORK

## 4.1 Get to know git better with Learn Git Branching

### 4.1.1 Subtask description

- Access the website Learn Git Branching and follow the instructions given there
- (Attention!) This task must be done individually! So ever team member must do on its own!
- Perform the first 4 level within this online tutorial
    1. *1: Introduction to Git Commits*
    2. *2: Branching in Git*
    3. *3: Merging in Git*
    4. *4: Rebase Introduction*



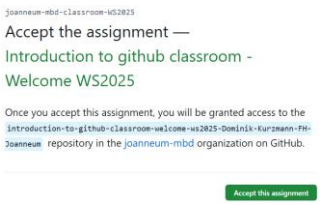

### 4.1.2 Presenting your results

- After finishing the 4 levels, make screenshot of your main page, where 4 green rectangles should be visible (refer picture above). Include also the current time when you take the screenshot (screenshot of complete screen!)
- **Upload the screenshots** of **all team members** in the Moodle **training unit task description** within a **zip file**!

## 4.2 Connect to Github's classroom first repository

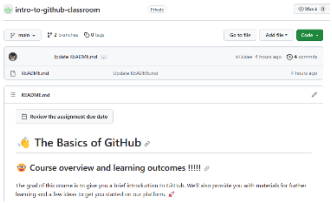
### 4.2.1 Subtask description

- For this training unit we will use *Github's classroom* functionality, which will act as our remote repository for this training unit.
- For this you need a github online account. If you do not have already one, you can create one directly from Github's homepage under following Link.
- You can use your private E-mail address or you FH Joanneum one to sign-up to Github.
- After successfully creating your Github account, you should be able to click on the following link ("Intro to github classroom") and select your name from the list. (If your name is not in the list, make sure you reach out to me)
- After you selected your name, you can continue and accept the first task ("intro_to_github") (see pictures below)




**step 1: click „accept this assignment"**


**step 2: wait some seconds until repo is cloned**




**step 4: read through task description**

### 4.2.2 Presenting your results

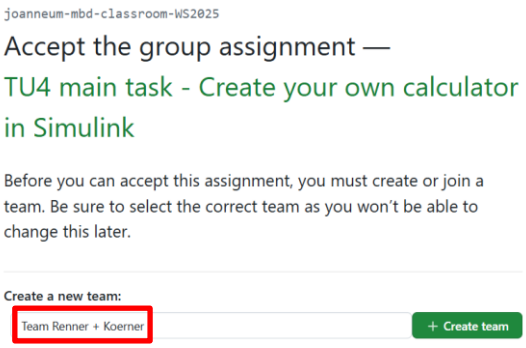
- After reading through the "README.md" file within the github-classroom repository, make a commit and push it to the online remote repo (just change something in the README.md file and upload the changes)
- This is **mandatory** for you to complete this first task
- You can either do this online by editing the README.md file and commit the change directly on your online repo, or you can clone the repo to your local PC, make some changes locally and push the changes to the online repo.

Committing changes to online repos will be necessary for the following tasks as well. Therefore, make sure that you understand the process of committing and pushing files to the remote repository, so that you do not run into problems in the upcoming tasks.

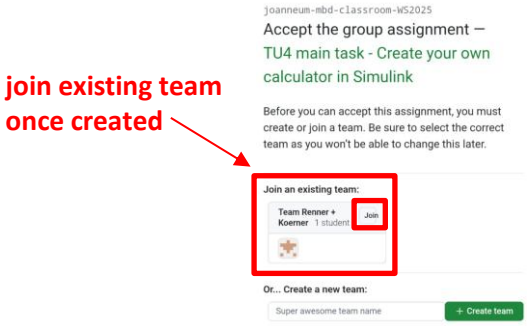
## 4.3 Connect to Github's classroom first repository

### 4.3.1 Subtask description

- After successfully reading through the documentation of task 1 "intro_to_github" and successfully creating your first commit on github, you can proceed with your second task, which will be your main task for today.
- For this task, you need to work in your chosen team.
- The next task can be accessed by clicking on the following link ("TU4 - git")
- The next step is to create a team with a team name of your choice (**Attention (!) only the first team member needs to create a team**) Please use your Surname and the Surname of your partner for this team name (**do not use** the team name from your Moodle course)
- The second or third team member should be able to see the team's name, as soon as it is created by the first team member (see pictures below)
- After successfully teaming-up, you should be able to see the same repository and be able to push and pull from the same repository.




**step 1: one person creates the team**
**team name = "Team *surname 1 + surname 2*"**




**step 2: team member joins the team**

### 4.3.2 Presenting your results

- After this subtask you should be able to see the training unit tasks (README.md file)
- Follow the task description of this file and be sure to check the tasks, as soon as you finished them
- For the upcoming tasks, your documentation must be done within the github repository
- The pictures of task 4.1.2 are the only file that you need to upload in Moodle.

## 5 REPORT

- Make sure, you follow the task description in the README.md file
- **Document** your work by using short but **precise git commit messages**, describing your work (which task was performed, was there a special problem in this task, etc)
- Make sure, to **document** your **Simulink model** as well, as required in previous training units as well
- Make sure to use explain your findings properly in your model. Use **annotation blocks**, add **pictures**, add **formulars** or *.**doc-block** to your model, to improve your documentation.
- Be sure to check the correct function of your model before uploading it to git. If your model **does not work**, you **will not receive any points** for this training unit.
- Make sure, that the script shows **your teams work** on this training unit task. Using solutions from other teams will result in zero points for you and your fellow student team.

- **Caution (!)** be sure to check the **due date** that is set for uploading the files on git!

## 6 SOURCES

[1] – pro git book, Online: https://git-scm.com/book/en/v2
[2] – git scm, Online: https://git-scm.com/