Fontys Git Manual



Last version: 09-12-2020

Authors:  
Tim de Laat  
Kristian Snel

Contents

[Manual structure 3](#_Toc58443182)

[Git Basic 4](#_Toc58443183)

[Getting started with 4](#_Toc58443184)

[Installation 5](#_Toc58443185)

[Setting up a test project 5](#_Toc58443186)

[Cloning an existing repository 5](#_Toc58443187)

[Setting up a new repository 6](#_Toc58443188)

[Git workflow 7](#_Toc58443189)

[Push pull requests 7](#_Toc58443190)

[Branching 7](#_Toc58443191)

[Master branch 7](#_Toc58443192)

[Git Technical 8](#_Toc58443193)

[Branching 8](#_Toc58443194)

[Git Issues 8](#_Toc58443195)

[Git Management 9](#_Toc58443196)

[Git Advanced 9](#_Toc58443197)

[Extra info 10](#_Toc58443198)

[Issue Templates 10](#_Toc58443199)

[Release Templates 10](#_Toc58443200)

[Git commands 10](#_Toc58443201)

[Appendix 12](#_Toc58443202)

[A. Installation 12](#_Toc58443203)

[Bibliography 15](#_Toc58443204)

[OBSOLETE 16](#_Toc58443205)

[What is Git and why should you use it? 16](#_Toc58443206)

**Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.**

# Manual structure

Git helps improve your project structure and allows groups to work more efficiently on project files and documents. Git is a so-called version control system.

About Version Control

What is “version control”, and why should you care? Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. This helps with implementing new features and still allows you to look back at earlier concepts of your work. This can be helpful in case something useful had been discarded earlier.



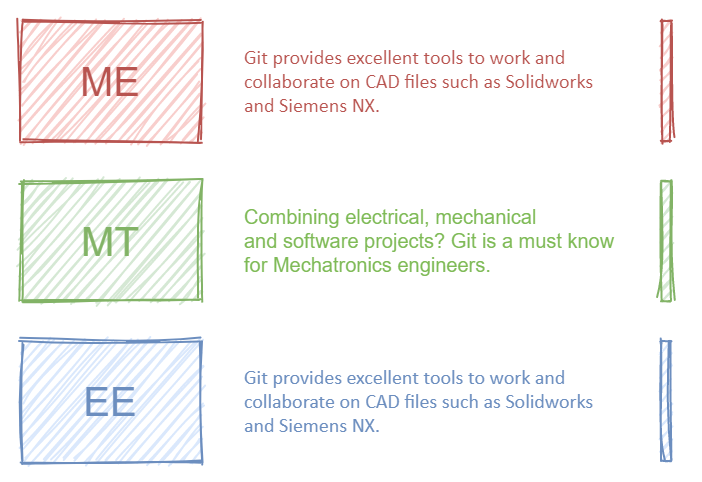
Git in EXPO projects

To improve the way of working within EXPO groups all disciplines are required to take knowledge of Git. This allows you to work together more efficiently and get started right away without discussing the need for a digital file storage solution.

To familiarize every student with Git first a Git basic introduction is given. This chapter requires you to install Github Desktop and create an account at Github.com

If you have mastered a basic workflow in Git and feel that you want to learn more about Git, this manual also features two chapters going deeper into the Management and Technical uses of Git.

Department specific tips

Tips only aimed at a specific discipline will from now on be color coded in the following representative colors:

Mechanical Engineering

Mechatronics engineering

Electrical engineering

# Git Basic

## Github.com introduction

### Create an account

### Activate Github education pack (Free!)

## Create a repository

### Manage repository structure

Add a readme.md file Markdown

### Manage access

### Transfer ownership

## Github projects

## Github issues

## Github wiki

## Github insights

# Git Technical

## Installation

Git supports a great variety of Desktop clients. One of the most popular interfaces is Github Desktop.

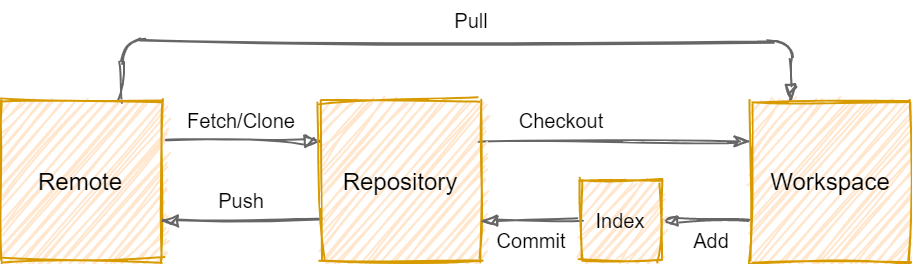
Github Desktop can be downloaded using this link: <https://desktop.github.com/>

For full details on how to install Github Desktop please refer to Appendix A. Installation.

After downloading the software create a GitHub account at <https://github.com/join?source=login>

## Version control with Git

Before using Git it is important to understand a few terms and concepts.



**Repository** A repository (or repo) is a folder or “storage location” with a .git/ folder inside to add Git version-control. A repository records versions of the files that it contains and may support multiple timelines of a project (see Branches).

**Local**

**Origin**

**Remote** A remote is a project repository that is not on your computer.

**Upstream**

**Commit** A commit used to save your changes to the local repository. To commit changes, you select one or more changed files and give those changes a descriptive title/summary. Optionally you can add a description to your changes. To upload local changes, see git Push.

**Push** Pushes your commits to your origin repository.

**Fetch** Downloads the commits, files and refs from a remote repository into your local repo.

**Pull** Downloads the commits, files and refs from a remote repository into your local repo and merges this with your local branches.

**Clone** Clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository (visible using git branch --remotes), and creates and checks out an initial branch that is forked from the cloned repository’s currently active branch.

**Pull Requests** A Pull Request (PR) is a request to ask your upstream project to pull your changes into their tree. Usually, a developer makes a pull-request when they finished a feature, or made a hotfix. The developer should add summary On GitHub, you can select one or more persons to review your changes and accept your changes.

**Stash** If two people edit the same file at the same moment a file conflict may occur. Unfortunately, this may still always occur. In this case those conflicts will be listed under the Stashed changes tab.

Now you are ready to install Github Desktop and start your first Git project.

Graphical user interface, text

Description automatically generated

### Setting up a test project



## Git workflow

### Push pull requests

Git functions by users sending updates and receiving them periodically. The power of Git is in the ability to update and receive files when you decide so. When using Git, an update from others is usually called a Pull request. While sending changed files over to the server or other people is generally called a Push request. If a user wishes to see if any files changed over at their team members, they could perform a so-called Fetch request. Therefore, A Pull request is always followed up by a Fetch request.



A Push request is initiated by committing. Committing is the process of submitting your changed files with a description about what changed.

## Branching

### Master branch

A branch is best explained as a separate timeline of a project. To allow multiple persons to work on the same project different branches may be desirable to prevent conflicts and allow testing of individual functionality. A user may create a branch, add changes, test those and then suggest the updated content to be used in the master branch. The master branch is where usually all branches come together and form the final project files. Updating a separate branch into the master branch is called merging.

Diagram

Description automatically generated

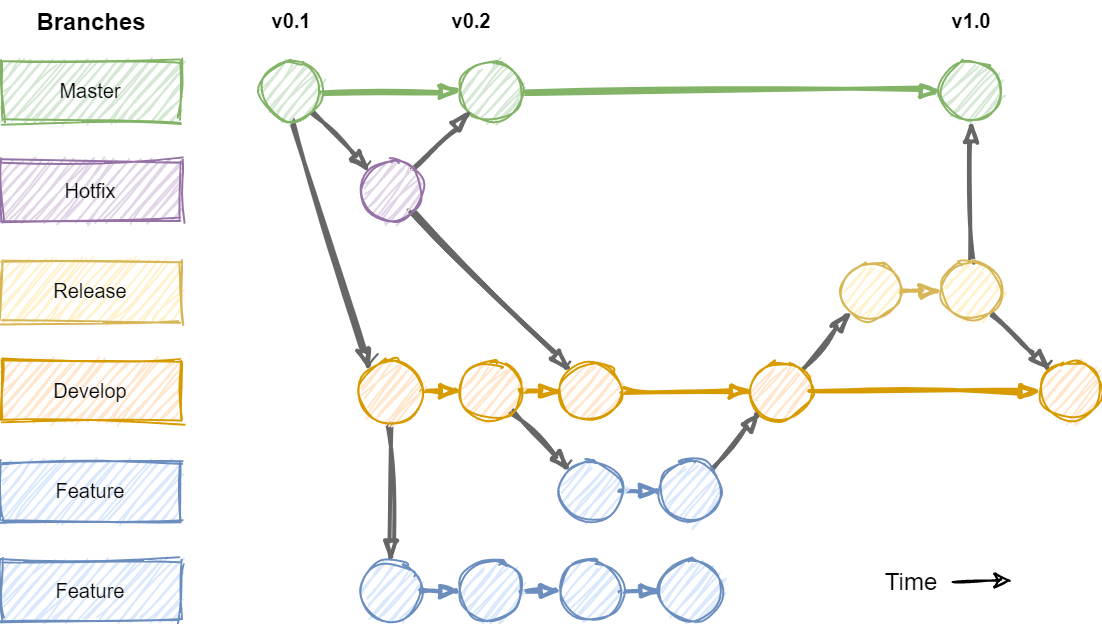
## Branching

**Merge** …

**Rebase**  …

**Merge conflict …**

## Git Issues



# Git Management

Semantic versioning

* **Major, Minor, Patch**

Tags and Releases

* **Releases**
* **Closing Issues**
* **Compatibility Matrix**

Readme.md

* **Markdown**

Project Boards

* **Project board template**
* **Link Issues**

Milestones

* **Project board template**

Insight

* **GitHub insights**

# Git Advanced

.gitignore

Git Submodules

Git Bash

# Extra info

## Issue Templates

## Release Templates

|  |
| --- |
| \_Software v2.13.0 release\_  # What's New **✨**  - Add Hello World (#152 )  - Fix old stuff (#123)  # Enhancements **🙌**  ## Add new stuff (#152 )  Added a function to print Hello world in the terminal.  Closes #152  # Bugs/issues Fixes 🐛  ## Fix old stuff (#123)  Fixed issue where the stops working after user entered a character.  Closes #123  # Features to look into 🔎  ## In progress  - Refactor Hello World(On Hold).  - Update Comments  ## Known issues  - Something is wrong. |

## Git commands

git init

git clone git pull

git config --global user.name git config --global user.email

git touch documentname

git add wildcard \*.html all .

git status

git commit -m 'description what changed' (Waar is die -m ook al weer voor?)

git push

Branching

git branch branchname

git checkin git checkout

gitmerge branchname 'describe merge'

# Appendix

## A. Installation



Run the downloaded GitHubDesktopSetup.exe

Sign in to or create a free account.





Log in to your account.



# Bibliography

|  |  |
| --- | --- |
| [1] | Backlog, "Git vs. SVN: Which version control system is right for you?," 23 June 2020. [Online]. Available: https://backlog.com/blog/git-vs-svn-version-control-system/. |
| [2] | Git, "Git - About Version Control," Git, [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. |
| [3] | Atlassian, "Git Merge," Atlassian, [Online]. Available: https://www.atlassian.com/git/tutorials/using-branches/git-merge#:~:text=Merging%20is%20Git's%20way%20of,them%20into%20a%20single%20branch.&text=The%20current%20branch%20will%20be,branch%20will%20be%20completely%20unaffected.. |



# OBSOLETE

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 