Fontys Git Manual



Last version: 22-11-2020

Authors:  
Tim de Laat  
Kristian Snel

Contents

[What is Git and why should you use it? 3](#_Toc57450121)

[Manual structure 3](#_Toc57450122)

[Git Basic 4](#_Toc57450123)

[Getting started with 4](#_Toc57450124)

[Installation 4](#_Toc57450125)

[Setting up a test project 4](#_Toc57450126)

[Cloning an existing repository 4](#_Toc57450127)

[Setting up a new repository 4](#_Toc57450128)

[Git workflow 4](#_Toc57450129)

[Push pull requests 4](#_Toc57450130)

[Git Technical 6](#_Toc57450131)

[Git Management 6](#_Toc57450132)

[Git Advanced 6](#_Toc57450133)

[Extra info 7](#_Toc57450134)

[Git commands 7](#_Toc57450135)

[Appendix 8](#_Toc57450136)

[A. Installation 8](#_Toc57450137)

# What is Git and why should you use it?

Large projects may require a lot of different people to work on a lot of files. Project structure may get cluttered up and files get lost. Git allows a team to keep a clear overview of the project files and the changes made to these files. Git is a so-called distributed version control system. Using a version control system allows a team to keep a log of changes made to the files in the past. Therefore, allowing every member of the team to see how the document progresses.

**Did you know?**Git was created in 2005 by Linus Torvalds. Torvalds is also famous for his creation of the Linux Kernel.

Git is not only useful for teams but also for individuals. A project may span a long amount of time. It can get difficult to keep track of project files and file history. Git provides a set of tools to see previous project versions, or test extra functionality parallel to the original project ensuring the safety of the original project.

This document features as a guideline to Git and how you as a Fontys student can incorporate Git in your standard workflow to prevent issues such as data loss and miscommunication and in general help with maintaining an organized project structure.

# Manual structure

This guide will be split into 4 different tutorials namely:

|  |  |
| --- | --- |
| Tutorial | Aimed at |
| Git basics | Everyone |
| Git technical | Project members |
| Git management | Project management |
| Git advanced | Those who want to use Git to its maximum potential |

To improve the way of working within EXPO groups all disciplines are required to take knowledge of Git. Tips only aimed at a specific discipline will from now on be color coded in the following representative colors:

**ME**

Git provides excellent tools to work and collaborate on CAD files such as Solidworks and Siemens NX.

Mechanical Engineering

Mechatronics engineering

**MT**

Combining electrical, mechanical and software projects? Git is a must know for Mechatronics engineers.

Electrical engineering

**EE**

Working on PLC software or circuit diagrams? Git provides version control to save file history.

# Git Basic

## Getting started with

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**
* **Upstream**
* **Commit**

A commit is 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 you local changes, see git Push.

* **Push**

To s

* **Pull requests**

A Pull Request (PR)

* **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.

Git also support a CLI  
For more information:   
https://git-scm.com/downloads

**MT**

**EE**

### 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.

### Setting up a test project

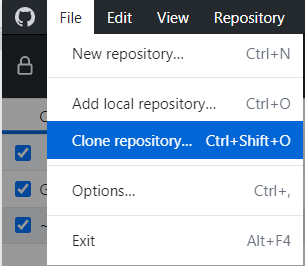


### Cloning an existing repository

To continue working on a project but create a copy of your own you may consider cloning an existing repository. To clone an existing repository, press the following button after installing the software.



Or this button under File 🡪 Clone repository ( + O)



To clone a repository, find a repository online, enter a url or select one of your own repositories.

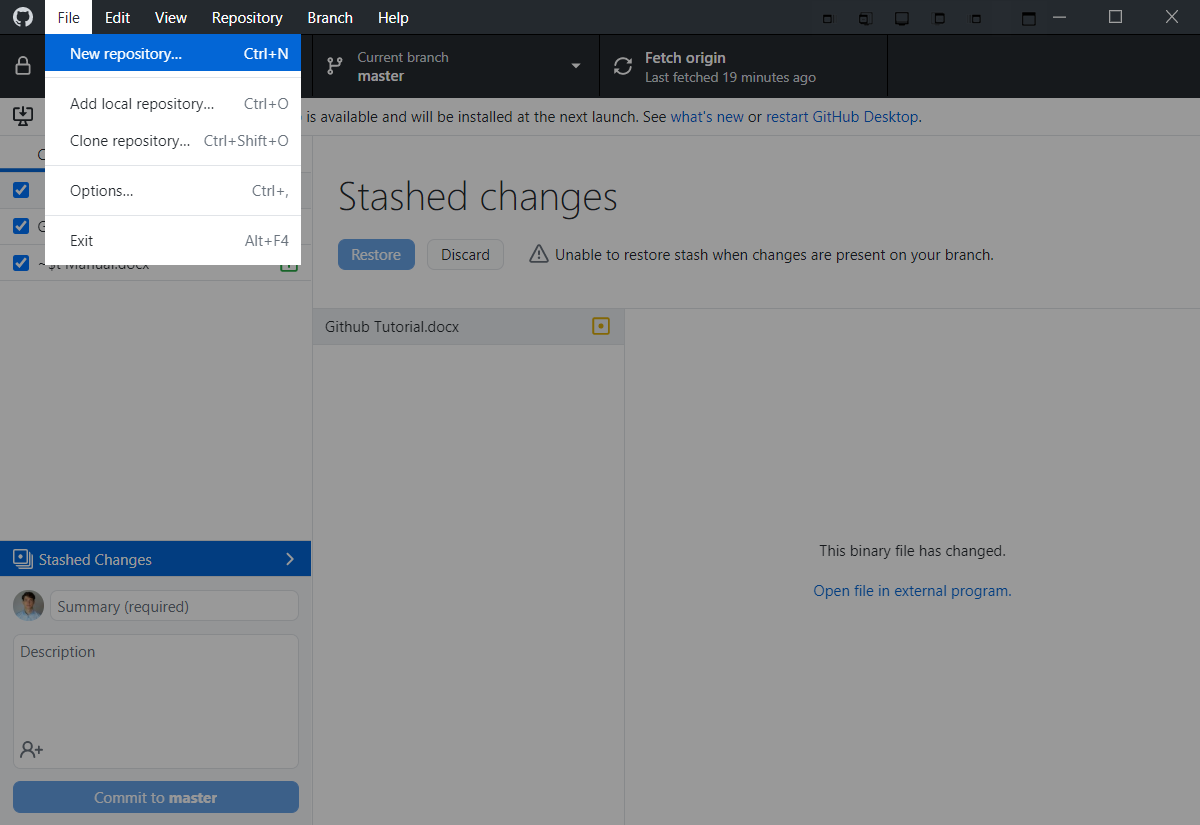
Then simply press the clone button and watch as the clone is created and the repository is opened.

### Setting up a new repository

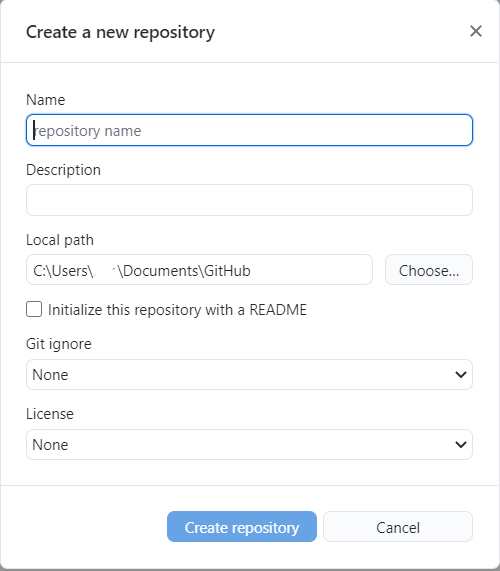
To create a new Repository right after installing Github Desktop click the button shown in the picture below.



If you have already opened a project, create a new repository under File 🡪 New repository



A new repository needs a Name and a description. By default, the new Repository will be saved locally in the Users\#YourPC#\Documents\GitHub folder.



## 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 lead 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 a fork.

# Git Technical

## 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

## 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.

