

# Fontys Git Manual



Last Version: 08-02-2021

## Authors:

Kristian Snel [kristian.snel@student.fontys.nl](mailto:kristian.snel@student.fontys.nl)

Tim de Laat [t.delaat@student.fontys.nl](mailto:t.delaat@student.fontys.nl)

Fontys councilor:

Ir. P.A.M. van Kollenburg



**Fontys**

**SCHOOL OF  
ENGINEERING**

# Content

1. Introduction .....	3
2. GitHub Basic .....	4
2.1. Setup GitHub .....	4
2.2. Issues .....	6
2.3. Project Boards .....	8
2.4. Milestones .....	9
2.5. Wiki .....	9
2.6. Transfer ownership .....	10
3. What is Git and how does it work .....	11
3.1. Installation .....	11
3.2. Version control with Git .....	12
3.3. Cloning an existing repository .....	13
3.4. Git workflow .....	14
3.5. Branching .....	15
3.6. .gitignore .....	16
4. For Teachers .....	17
4.1. After the project .....	17
4.2. Continuing a project .....	17
4.3. Discontinuing a project .....	17
5. Extra info .....	18
5.1. Semantic Versioning .....	18
6. Appendix .....	19
6.1. A. Installation .....	19
7. Bibliography .....	22

## 1. Introduction

GitHub is a platform for version control using Git. Besides that, GitHub can be a powerful tool for project management. This manual is divided in two parts. The first part focusses on using GitHub as a project management tool (GitHub Basic). The second part focusses on how to use GitHub for version control using Git (Git Technical). The latter is optional. This manual is relevant for all Fontys programs participating in the EXPO projects.

### About Git Version Control

What is “Git 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 recover files to their historic versions. 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 an addition to a certain module causes the system to fail and changes have to be reverted.

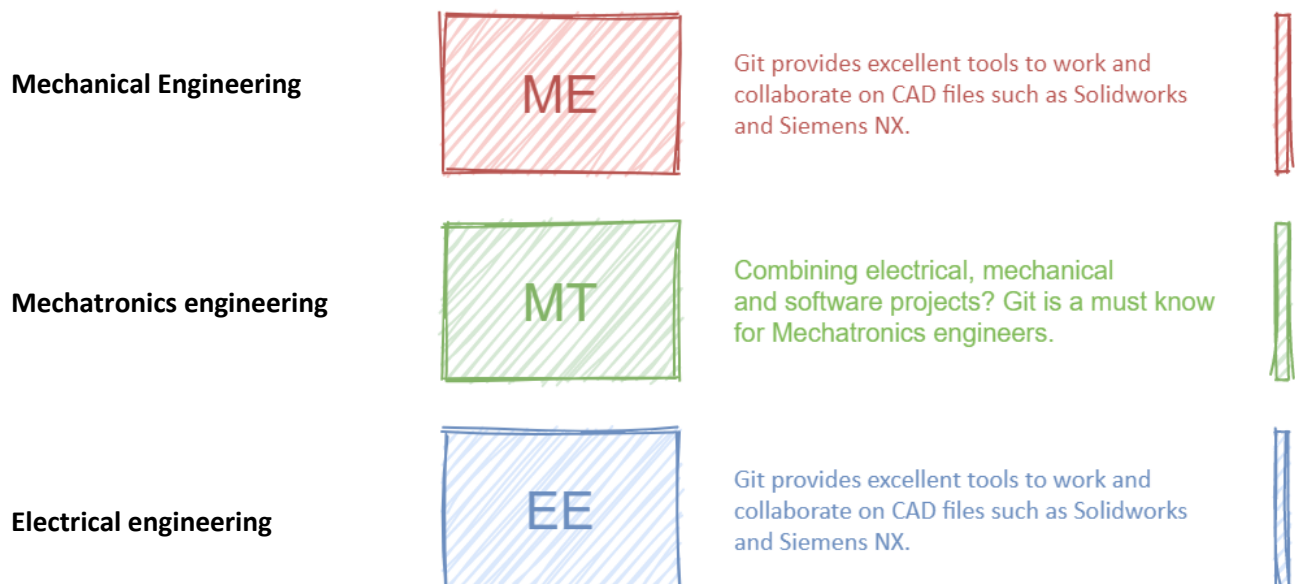
### About GitHub

GitHub is a platform for both Version Control using Git and project management. While it is mainly used for software development, it is very useful for all developers/engineers that use their computer as their computer as a development tool. As of January 2020, GitHub is hosts over 190 million projects (called repositories) and has over 40 million users.



### Department specific tips

This image gives a few examples for every study on how Git can be used to enhance their workflow.



## 2. GitHub Basic

GitHub offers project management tools that can improve the workflow of your project group. When working on a project, you (and your group) first divide the project into smaller tasks. You create a GitHub Issue for all these tasks. A GitHub Issue is like a forum post for a project task. In this Issue, group members may share information, problems and ideas related to the topic and discuss it with other members. When the task is finished, the Issue can be closed and all information about that task will be archived. To create a better overview, Issues can be labelled (i.e., Electrical/Mechanical/Mechatronics or Bug/Enhancement). Issues can also be assigned to project members, linked to a milestone, and/or linked to a project.

When an Issue is assigned to a project, it appears on the project's GitHub Project Board. A default Project Board will show three columns: "To Do", "In Progress" and "Done". An Issue that is linked to the project, will automatically appear in the "To Do" column. And Issues that are closed will automatically move to the "Done" list. When somebody starts working on a task, it will drag-n-drop the Issue to the "In Progress" column. This creates a good overview for the entire group (and supervisors) to what tasks are being worked on, what is done and what still left to-do. By clicking on the issue, itself, a group member can see more information about the problem itself and participate in the discussion.

How to create Issues, working with the Project Board and what structure/workflow to use will be discussed later. First, Let's create a GitHub account and create your first repository.

### 2.1. Setup GitHub

#### Create a GitHub account

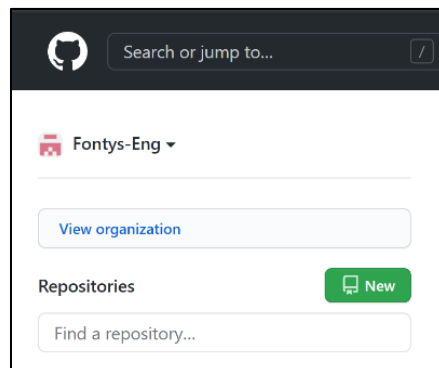
To create a GitHub account. Go to <https://GitHub.com/> and press "Sign Up" in the right top corner. Enter your email, username, password and solve the puzzle.

#### Optional: activate GitHub education pack for free!

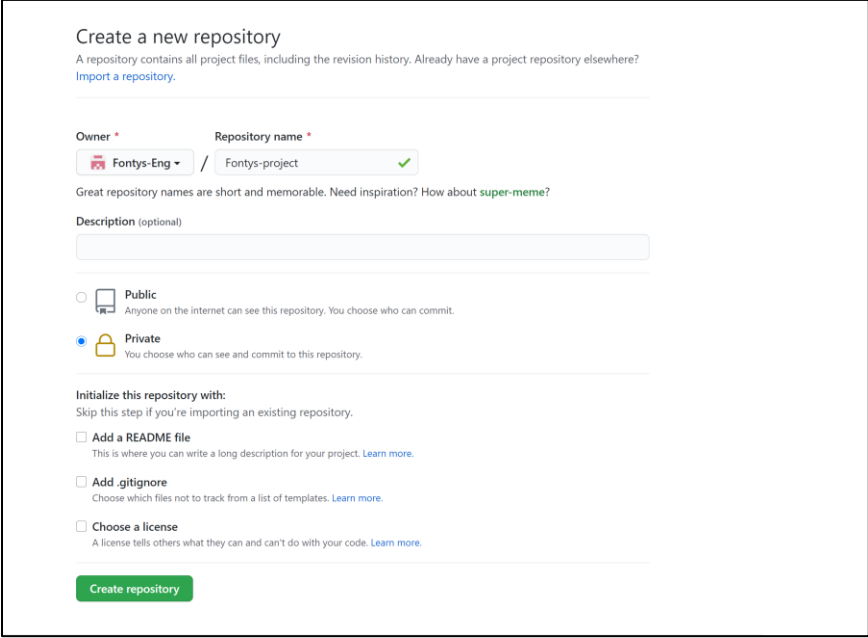
One of the benefits of being a student with GitHub is that you get software tools for free. You can sign-up via <https://education.GitHub.com/pack>. One of the primary perks is GitHub pro, which allows you to add more people to a private repository (free GitHub only allows up to three collaborators).

#### Create a repository

The next step is to create a repository. A repository is a version-controlled project folder, but this is used for now. GitHub offers a lot of project management tools with a repository. Things like projects boards, issue pages and wikis.



To create a repository, login with your GitHub account and press new in the left top of your screen. See the image below. Give the repository a name, add a description and click on repository “Private”. With a Private repository, the owner can decide who can see and work in the repository. The group team leader creates the repository. This repository is handed to your teacher at the end of the project.



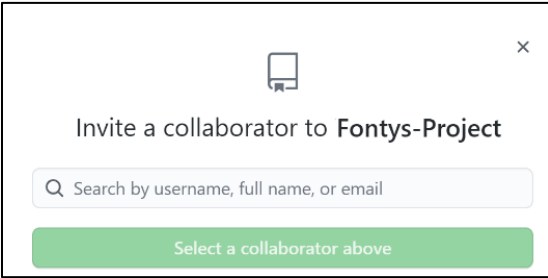
The screenshot shows the 'Create a new repository' page on GitHub. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner' with a dropdown menu showing 'Fontys-Eng' and 'Repository name' with the text 'Fontys-project' and a green checkmark. A note below says 'Great repository names are short and memorable. Need inspiration? How about [super-meme?](#)'. There is a 'Description (optional)' text area. Below that, there are two radio buttons: 'Public' (unselected) and 'Private' (selected). The 'Private' option has a lock icon and says 'You choose who can see and commit to this repository.'. Underneath, there is a section 'Initialize this repository with:' with the instruction 'Skip this step if you're importing an existing repository.'. There are three checkboxes: 'Add a README file' (unselected), 'Add .gitignore' (unselected), and 'Choose a license' (unselected). Each checkbox has a brief description and a 'Learn more' link. At the bottom, there is a green 'Create repository' button.

## Markdown

A blank repository can be generated with a README.md file. This file is used to describe the project. When viewed on Github.com the readme file is displayed on the webpage in a Markdown format. Markdown is a lightweight Markup language which allows you to add some formatting to plain tekst.

## Manage access

The next step is to invite your other group members, tutor and teacher into the repository. To do that, open your repository. Go to settings → Manage access and press Invite a collaborator. You should see a similar prompt to this image.



The screenshot shows a dialog box titled 'Invite a collaborator to Fontys-Project'. It has a close button (X) in the top right corner. Below the title, there is a search bar with the placeholder text 'Search by username, full name, or email'. At the bottom, there is a green button with the text 'Select a collaborator above'.

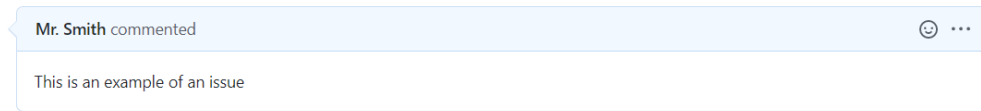
Choose to invite the new group into the repository (and remove the old group from it). possible for the teacher to hand-over the repository to the new group or let them make a new repository.

## 2.2. Issues

When starting a project, the first step is to divide project into smaller tasks. On GitHub, you place these tasks GitHub Issues. These GitHub Issues are similar to forum posts where group members share information, problems and ideas related to the task and discuss it with other group members. To create an issue, go to ⓘ Issues and press on **New issue**. There you give the Issue a descriptive title and description for the task. You can also insert pictures (including vector images, GIF and MP4 files). The result will look something like this:

### New Issue #1

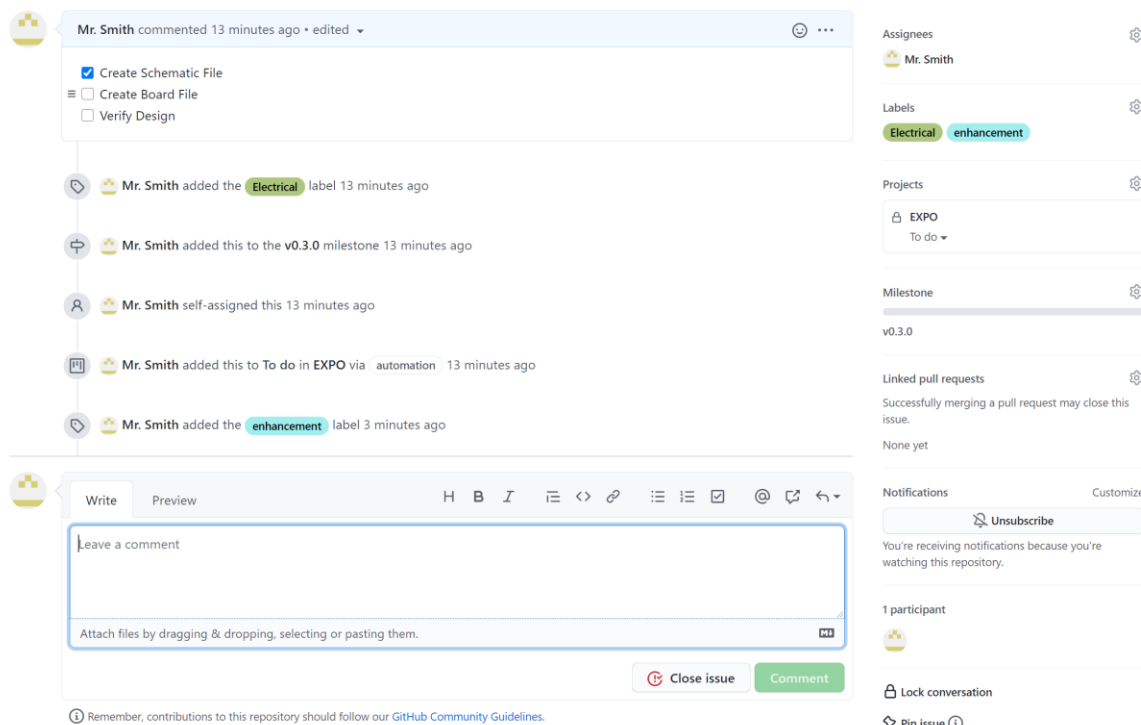
ⓘ Open Mr. Smith opened this issue 3 minutes ago · 0 comments



It is also possible (and recommended) to assign the Issue to group members (including yourself). This creates an overview of who is assigned to which task. Besides that, it is also possible to label the issue, assign it to a project or link it to a milestone. This further improves readability of all the issues. Once an issue is assigned to a Project, it will be placed on the project's Project Board (see next paragraph).

### PCB Design #11



ⓘ Open Mr. Smith opened this issue 13 minutes ago · 0 comments



*Quick Note: it is also possible to reference to other issues by using the issue number (i.e. #11). And mention your group members in the issues by using @ (i.e. @MrSmith).*

After all tasks are added, the issue list will give an overview like this.

The screenshot shows the GitHub Issues list interface. At the top, there is a search bar with the text "is:issue is:open", a "Filters" dropdown, and buttons for "Labels 15" and "Milestones 1". A green "New issue" button is on the right. Below the search bar, it says "4 Open" with an exclamation mark icon and "11 Closed" with a checkmark icon. The main area displays a list of four issues, each with a checkbox, a green exclamation mark icon, a title, labels, and a version number. The issues are: "3D designs" (Mechatronics, enhancement, v0.3.0), "Software crash" (Bug, Software, v0.3.0), "Mechanical Design" (Mechanical, enhancement, v0.3.0), and "PCB Design" (Electrical, enhancement, v0.3.0). Each issue has a progress bar and a "2 of 3" or "1 of 3" indicator.

When the issue is finished, you can close this issue by pressing the  button. It is recommended to add a closing comment to give a reason for why the issue is closed. When an Issue is closed, all contents of that Issue is archived. It is possible to later reopen the issue by pressing .

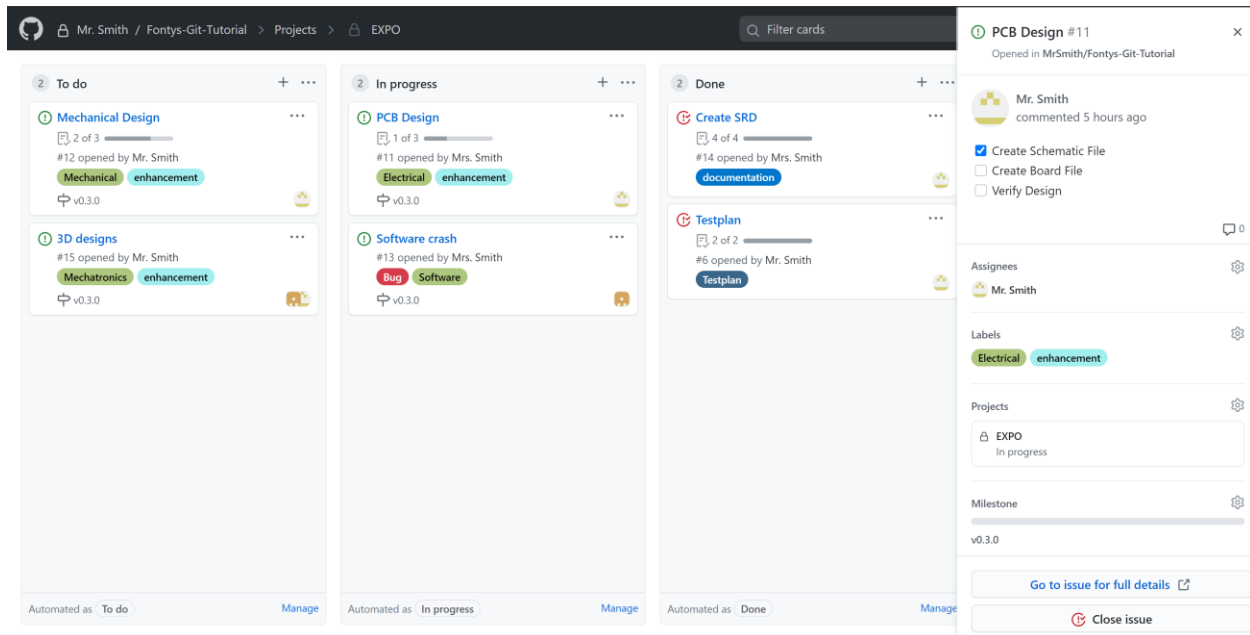
The screenshot shows the GitHub issue comment form. It has a "Write" tab and a "Preview" tab. The "Write" tab is active, showing a text area with the text "This issue is solved." Below the text area is a placeholder for attaching files. At the bottom right, there are two buttons: "Close with comment" and "Comment".

## 2.3. Project Boards

GitHub allows you to create a Project Board give an overview of the tasks and their status. These Project Boards are organized in three columns: “To do”, “in progress” and “Done” (similar to Trello). When creating a GitHub Project Board, make sure to select “Basic Kanban” under templates when creating a new project in GitHub. This template will give a good starting point and add some automation to make managing easier.

✓ **Basic kanban**  
Basic kanban-style board with columns for To do, In progress and Done.

As mentioned in chapter 2.2, when you create an issue and assign it to a project, that issue will appear on that project board (in the “To do” list). You can also see who is assigned to that project. When a group member starts working on a task, it will drag and drop the issue to the “In progress” column. When the task is finished, the group member will close the task (see chapter 2.2) and it will automatically be moved to the “Done” column. This overview will look something like this.



It is also possible to click on the one of the issues on the project board. This opens a window on the side (as shown in the image above) and gives additional information about the task. It is also possible to close the issue from here or see the go to the issue for full details.



## 2.4. Milestones

A useful way to keep track of deadlines and what tasks to finish for those deadlines can be done by creating milestones. After you have created a milestone, you can assign GitHub issues to it. The milestone will become visible in both the issue overview and on the project board. To create a milestone, go to [Issues](#) and in the right corner, you will find [Milestones](#). This brings you to the GitHub milestones, where you can create a new milestone. When you create a milestone, you give it a title, a date for the deadline (optional) and a description of what the milestone holds. The title can be either a version number (using semantic versioning) or a short descriptive title like “demo”.

Labels Milestones New milestone

2 Open 2 Closed Sort

**v0.4.0**  
No due date Last updated less than a minute ago  
0% complete 0 open 0 closed  
[Edit](#) [Close](#) [Delete](#)

**v0.3.0**  
Due by February 19, 2021 Last updated 3 minutes ago  
50% complete 2 open 2 closed  
[Edit](#) [Close](#) [Delete](#)

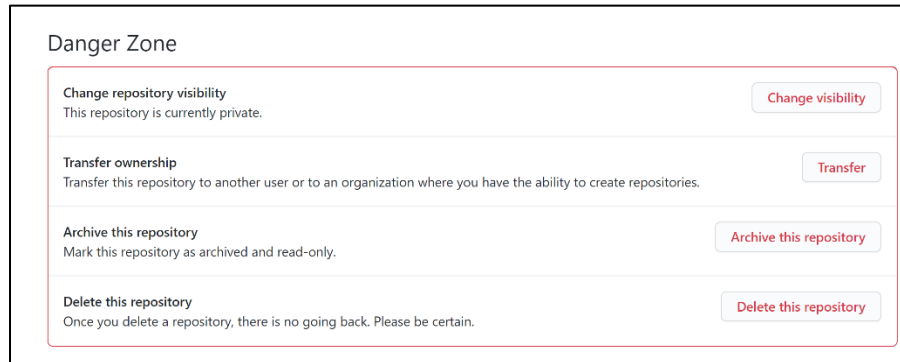
After this, you can visit the milestones menu and see the progress of the milestone. It will show how many tasks are still left to do before the milestone is completed. After the milestone is reached, you can close the milestone.

## 2.5. Wiki

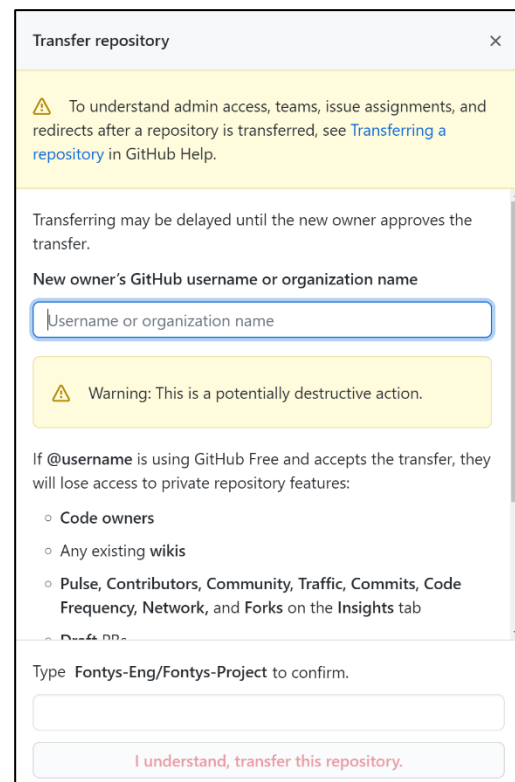
The GitHub [Wiki](#) allows you to create a knowledge base around your project. Important information can be saved here in a structured manner. GitHub Wiki also used the same Markdown language as the `readme.md` file in the root folder of your repository.

## 2.6. Transfer ownership

For continuation of the project, it is important to hand over the repository to your teacher. First you have to make sure that your teacher has a GitHub pro account. This is due to collaborator limit in private repositories with free GitHub accounts. To check this, go to Manage access and hove over the profile picture. If it is a pro user, it should show the **PRO** icon.



To transfer ownership, the owner of the repository must go to settings, and scroll all the way down to the “Danger Zone” and press “Transfer”. A prompt should show up for transferring the repository. There, enter the GitHub username of your teacher. Always double check if you have the user selected. Then follow the other instruction and press “I understand, transfer this repository.” After this is done, inform your teacher that you made a request to transfer your repository. He/she should have received an email about this to accept the transfer.



### 3. What is Git and how does it work

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

#### 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.

#### 3.1. Installation

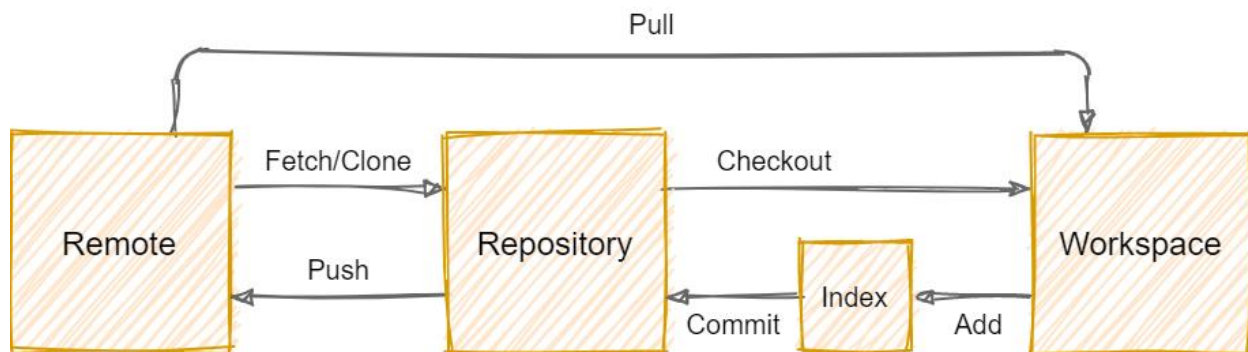
Git supports a great variety of Desktop clients. One of the most popular interfaces is GitHub Desktop.

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>

### 3.2. Version control with Git

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



Term	Description
<b>Repository</b>	A repository (or repo) is a folder or “storage location” with a .git/ folder inside to add Git version-control. A repository records changes in the files that it contains and may support multiple timelines of a project (see Branches).
<b>Local</b>	Refers to your local repository, this is where you can make changes to project files.
<b>Origin</b>	The Origin refers to where the repository the project is cloned from (originates from). In a standard workflow, your local is your computer (on which you change the files) and the origin is the repository that you push your changes to.
<b>Remote</b>	A remote is a project repository that is not on your computer.
<b>Upstream</b>	Upstream branches define the branch tracked on the remote repository by your local remote branch.
<b>Fork</b>	A fork is a clone/copy of a repository.
<b>Commit</b>	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.
<b>Push</b>	Pushes your commits to your origin repository.
<b>Fetch</b>	Downloads the commits, files, and refs from a remote repository into your local repo.
<b>Pull</b>	Downloads the commits, files, and refs from a remote repository into your local repo and merges this with your local branches
<b>Clone</b>	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.
<b>Pull Requests</b>	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.
<b>Stash</b>	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.

Setting up a test project



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





Create a tutorial repository...

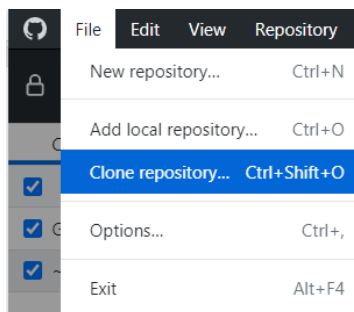
### 3.3. Cloning an existing repository

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.



Clone a repository from the Internet...

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.

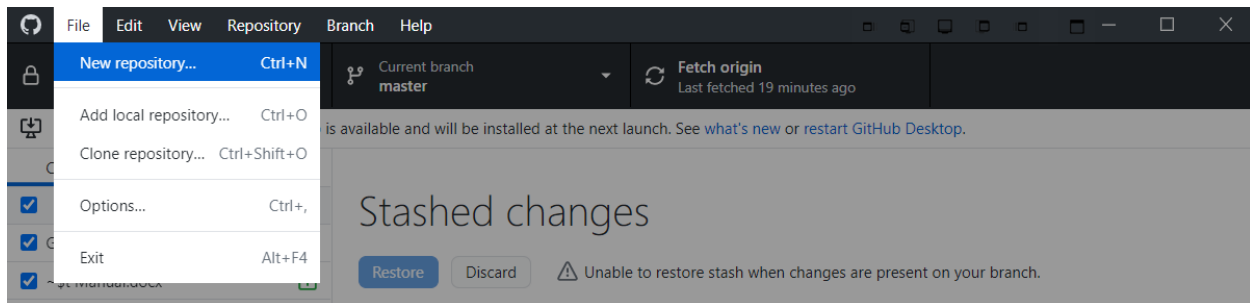
### Setting up a new repository

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



Create a New Repository on your hard drive...

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.

### 3.4. 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 group members, they could perform a so-called *Fetch request*. Therefore, A Pull request is always followed up by a Fetch request.

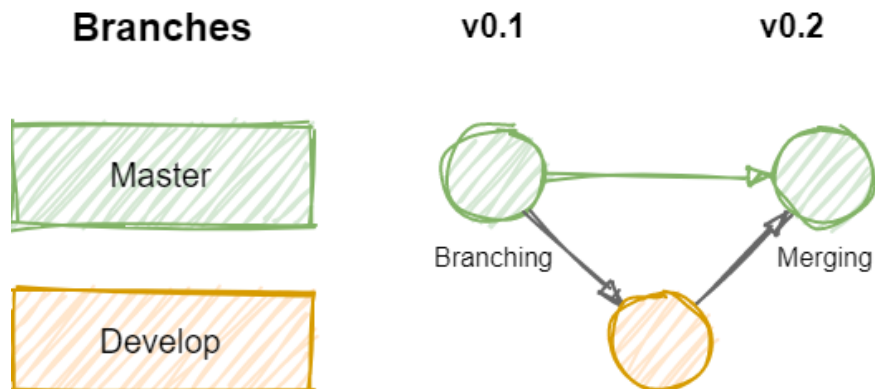


A Push request is initiated by *committing*. Committing is the process of submitting your changed files with a description about what changed.

## 3.5. 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.



Term	Description
Branch	A branch is an alternative timeline of a Repository (or project folder). This means that you may work on the same project but can also view the old unchanged files as they were before “branching”.
Merge	A merge is the combining of two branches into a single branch. Merging branches might cause conflicts which require the group to decide on what versions of files to continue with.
Merge Conflict	A merge conflict occurs when two files with the same name contain conflicting content. A merge conflict cannot be solved by Git and requires the input of a user to be solved.
Rebase	Rebasing is quite similar to Mergins as it is used to combine information from two existing branches. Rebasing is different as it merges its changes at the moment of branching off from the master. This causes the rebase to be merged into a historic version of the master. This results in a much cleaner overview but all commits to the master in the meantime will overwrite potential changes in the rebased branch.

### Branching example

An example of a slightly more complex branching system is depicted in Figure 1.

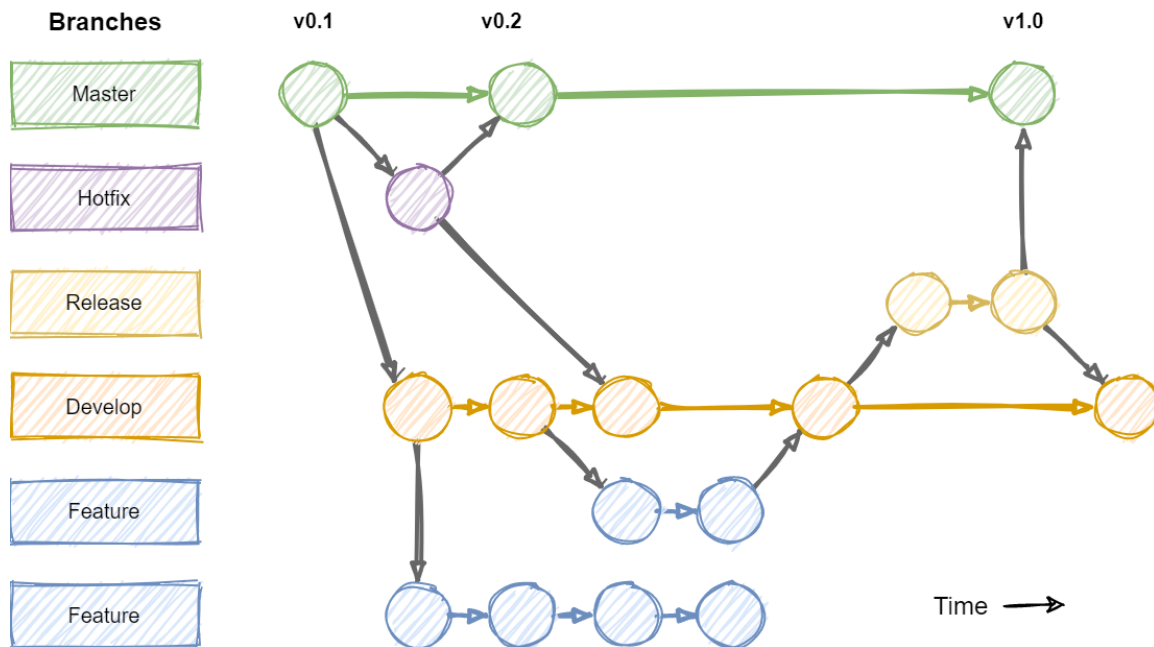


Figure 1: Example of branching.

As seen the master shows three version, v0.1, v0.2 and v1.0 (release). Projects usually stick to a company defined branching structure. Many names used for these branches might include Hotfix, Release, Develop or Feature. Depending on the branch name it may suggest what is being worked on or what is the goal of the branch. In the example above a Develop branch is used to work on new features and test in a safe development environment. While the master provides the base code. A Release branch usually holds a version that is “almost” ready for deployment.

### 3.6. .gitignore

When working on a project, you only want to keep track of and store the files that you work on. But in most cases, your program will generate extra files that are not needed for the project (i.e., binary/hex files when compiling software). Therefore Git uses a .gitignore file. This file specifies files that Git should not track (should ignore). <https://github.com/github/gitignore/tree/master/Global>



## 4. For Teachers

This chapter is meant for teachers. It gives tips on how to manage the repository that is received from the EXPO group, and how to manage it when a project continues or discontinues.

### 4.1. After the project

When the teacher receives a repository, it is recommended to remove the project members from the previous EXPO group before adding the new members. If the project member still wants to keep a copy of the project, it can choose to clone by making a Fork. This will only copy the files, not the issues, project boards and wiki documentation. For it is recommended to give the students a heads-up before they are kicked out.

### 4.2. Continuing a project

When a project continues, there are three options to deal with the

- The teacher invites the new EXPO group (and new Tutor) to the repository and maintain ownership (Recommended). The EXPO group will either create a new project board or continue using the project board of the previous group. It will have access to see both open and closed issues, their project board. This shows the progress of the previous group, as well as their achievements, struggles and design discussions (through closed issues). The group also has access to the and possible wiki documentation.
- The teacher hands over the repository to the students. This has the same benefits as discussed above, but the project is in control of the repository. This can add the risk of losing the repository. Before doing this, make sure that the student that receives the repository has GitHub pro (if not, recommend him to take activate the GitHub activation pack).
- The last option is to ask the new EXPO group to create a new repository. The new group starts with a clean slate but has no insight in the activities from the previous group (besides what is mentioned in the report).

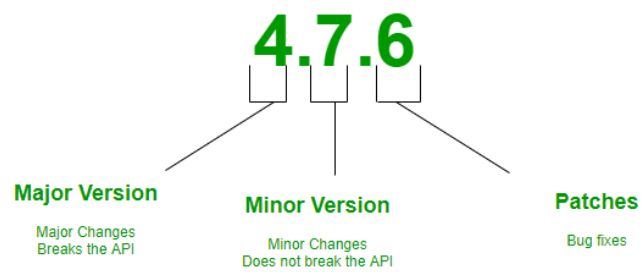
### 4.3. Discontinuing a project

When a project is discontinued, it is recommended to Archive the repository. This will make the repository read-only, preventing changes to be made to it afterwards. It is recommended by GitHub to close all active pull-requests and open issues first. To archive a report, go to settings, scroll down to the “Danger Zone”, follow the instructions and press “I understand the consequences, archive this repository”. It is possible to unarchive a repository afterwards.

## 5. Extra info

### 5.1. Semantic Versioning

Semantic Versioning is a process of assigning version numbers to different states/versions/iterations of your project. It is often used for software management, but it can also be helpful for version tracking in other engineering studies.



## 6. Appendix

### 6.1. A. Installation



Run the downloaded GitHubDesktopSetup.exe

Sign in to or create a free account.

## Welcome to GitHub Desktop

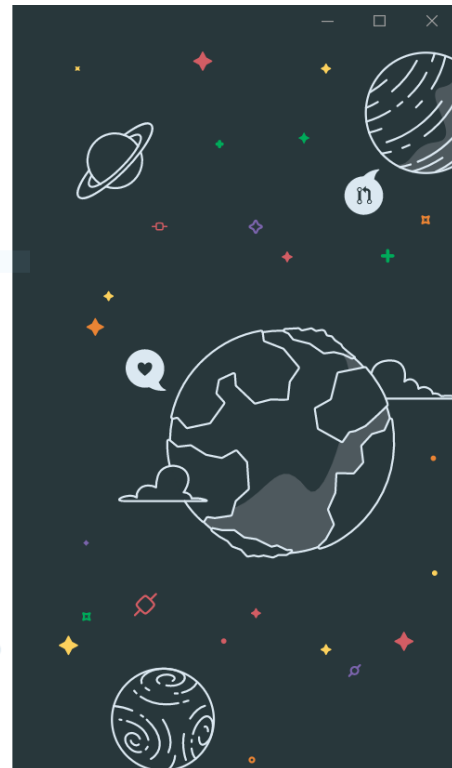
GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise Server. Sign in below to get started with your existing projects.

New to GitHub? [Create your free account.](#)

Sign in to GitHub.com

Sign in to GitHub Enterprise Server

Skip this step



## Let's get started!

Add a repository to GitHub Desktop to start collaborating

Create a tutorial repository...

Clone a repository from the Internet...

Create a New Repository on your hard drive...

Add an Existing Repository from your hard drive...


**ProTip!** You can drag & drop an existing repository folder here to add it to Desktop

Search

Sorry, I can't find any repository matching Search



Log in to your account.



## Configure Git


This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.

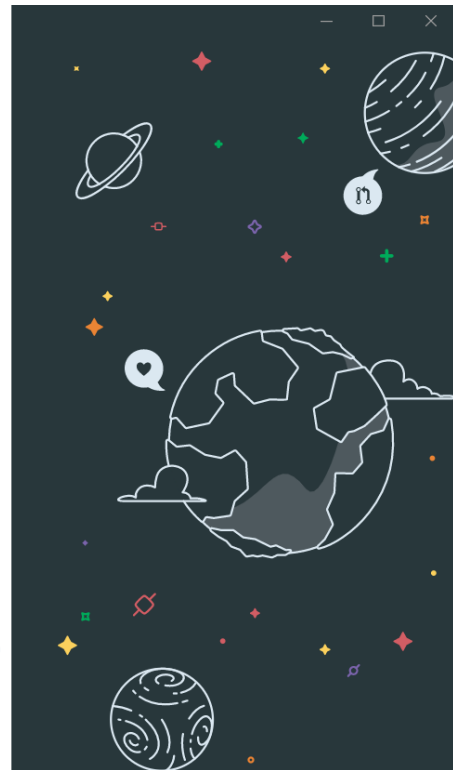
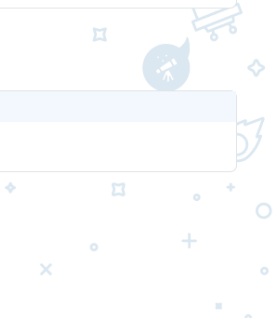
Name

Email

Example commit

**Fix all the things**

 Student • 30m



## 7. Bibliography

[ Backlog, "Git vs. SVN: Which version control system is right for you?," 23 June 2020. [Online].

1 Available: <https://backlog.com/blog/git-vs-svn-version-control-system/>.

]

[ Git, "Git - About Version Control," Git, [Online]. Available: <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>.

]

[ Atlassian, "Git Merge," Atlassian, [Online]. Available: [https://www.atlassian.com/git/tutorials/using-branches/git-](https://www.atlassian.com/git/tutorials/using-branches/git-merge)

3 merge#:~:text=Merging%20is%20Git's%20way%20of,them%20into%20a%20single%20branch.&text=The%20current%20branch%20will%20be,branch%20will%20be%20completely%20unaffected..