PHOCUS

Target locust from within

Git Guide

TU Delft

git

GitHub

iGEM

Introduction

Git is a control system that allows you to track changes throughout the process of software development. It is used when there are multiple people working on a single set of files. It helps its user maximize their efficiency by helping them collaborate. One of its biggest advantages is that it allows the user create branches.

After making changes to the code, these can be committed to the git repository. This works similar to taking "snapshot" of the code. This is something that should be done manually on a regular basis (a few times per day when working a full day). Every snapshot made is just a record of your code at that time and all these versions are stored. You can therefore always go back to previous versions of your code, which you might want to do when you accidentally break your code for instance.

Git can also be complemented with GitHub, on which you can remotely store your code. This allows multiple people to work on the same software. Everyone is able to deposit (push) or withdraw (pull) their code onto this server. You can imagine that working on the same code with multiple people can cause conflicts in the code. Aside from being a version manager, Git will present these conflicts and provide assistance in merging the two different versions. Git combined with Gitlab stops you from saving multiple versions (code_v1, code_v2, etc) and makes merging your code with your fellow developers far easier!

Git during Covid-19?

COVID-19 has had a major import on our lives, especially how we work. Many people around the world have been restricted from leaving their homes and certainly from entering the office. In general, this has made it difficult for people to collaborate nevertheless, many are still actively searching for solutions to fill this void and make sure the job gets done. Video communication platforms have played a prominent role in helping people share ideas. We believe that especially now, Git and GitHub have a key role to play in helping people share ideas and collaborate from remote locations.

Git workflow

When starting a new session, the first step is always to check the Git repository. The commands below show how this is done.

```
$ git status

On branch master

Your branch is up to date with 'origin/master'.


Changes not staged for commit:

  (use "git add <file>..." to update what will be committed)

  (use "git checkout -- <file>..." to discard changes in working
directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

When starting a new project or expanding on an existing project, you might also want to create either a new file or folder. After created on your computer they have to be added to the server. This is done using the code below. The full stop will add everything you created; you can also specify a certain file by replacing the full stop with a file path.

```
$ git add .
```

In this example, the changes are first committed (take snapshot) before they are 'pushed' to the server. After adding your files to the server, you have to commit your changes. This is done using the code below. The '-' indicates the used options and the 'a' commit all changes. The 'm' is added if you want the option for adding a message to your commit. It is advisable to always add a comment as this can help you understand the changes when looking back. In the example you can enter your personalized comment in the quotation marks.

```
$ git commit -am "commit message: My first commit" d
```

If other people have worked on the code, then it's important to 'pull' any changes made to the remote repository. In the example below, origin is a shortcut for the ssh url where you downloaded the repository. In other words, it means pull it from the master branch on the remote server is located.

```
$ git pull origin master
```

This might give rise to a merging error when multiple versions of the same code exist. Git automatically detects at which places the file was edited and if both sections that were edited are far enough apart, Git will automatically perform the merge. If, for instance, the same line of code was edited twice then Git cannot solve the merge conflict by itself and will visually tell you were (which lines) the conflict is present and ask you to choose which lines to keep before pushing the code back to the server.

```
$ git push origin master
```

Branching

When working with git, it's also possible to work in different branches. You can create a branch for each feature of your software. In essence branches allow you to continue working on the same code without changing the master, thereby prevent the master branch (final product) from being corrupted. In iGEM different branches can be used for the module that the team works on. There are some extra commands needed to create and move around branches

Whenever working with branches it's always best to grow accustomed to first check which branch you are working in. This is done using the code below.

```
$ git branch
```

The code below can be used to create a new branch. Checkout means that you want to change branch, -b means that you want to create a new branch. Branchname needs to be replaced with the name of your new branch.

```
$ git checkout -b branchname
```

Now that you know how to check which branch you are in and how to create a branch it's also important to know how to move between branches. This is done using the code below where Branchname needs to be replaced with the name of the branch that you would like to move to.

```
$ git checkout branchname
```

Branches can also be created by other people. To access these branches, you first need to 'fetch' them.

```
$ git fetch
```

Whenever you start creating new branches it's important to keep an overview of all the branches at all times! The same rules apply for each branch: first git status, git add ., git commit -am '', git pull origin branchname and finally, git push origin branchname.

When you are done working in a branch it's also important to know how to delete that (local) branch. This is done using the code below.

```
$ git branch -d branchname
```

Contact us!

iGEM TU Delft

igemtudelft

TU Delft_iGEM

iGEM TU Delft

igem@tudelft.nl

www.igemtudelft.nl

+31152785394

Building 58, Room E0.400
Van der Maasweg 9
2629 HZ Delft
The Netherlands