GitHub Tutorial



Table of Contents

[1. Converting your account to a Student Account 3](#_Toc527030540)

[2. Creating a GitHub repository 4](#_Toc527030541)

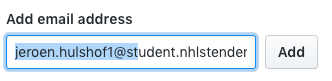
[3. Installing the repository on your computer using Git Bash or Terminal. 5](#_Toc527030542)

[4. Pull and Push explained 6](#_Toc527030543)

# Converting your account to a Student Account

GitHub has a lot of features you can’t use if you don’t pay for the service. For example: Private repositories. If you are a student you can get those features for free. Here’s how you can convert your account to a Student Account.

**How to convert your account:**



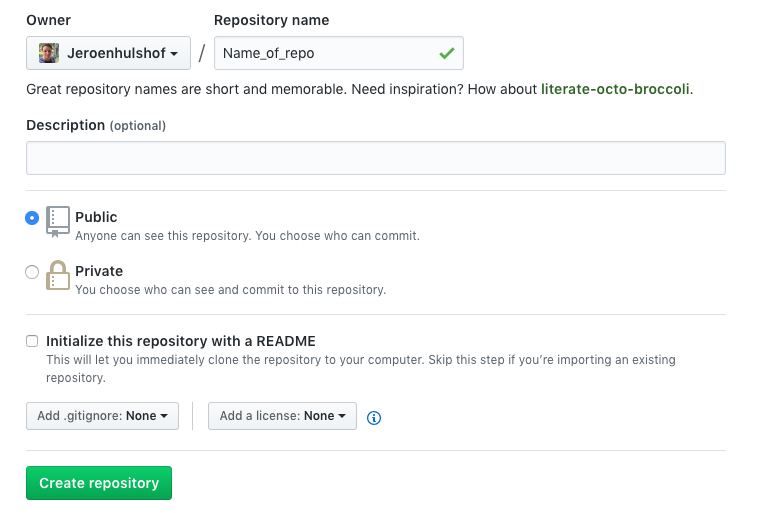
1. Log in to GitHub.
2. Go to Settings and select “Email” in the menu on the left.
3. Add your school email address to your account.
4. Go to <https://education.github.com/pack>.Follow the steps. Make sure to select the school email address you’ve just created.
5. You now have access to multiple features.

# Creating a GitHub repository

Screen%20Shot%202018-10-11%20at%2013.31.53.png

**Creating a repository:**

1. On the main page, there’s a button called “New Repository”. Click on it.
2. Fill in the repository name. Make the repository public or private.
3. You can choose to initialize this repository with a readme. For this tutorial, I will initialize this repository with a readme.



1. Click on “Create Repository”.
2. You will get redirected to the repository you’ve just created.

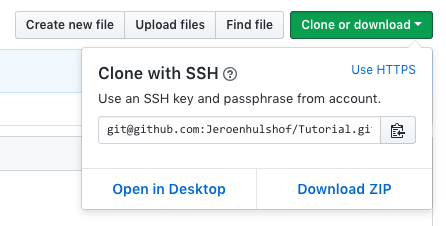
# Installing the repository on your computer using Git Bash or Terminal.

Now you’ve created a repository, you need to install it on your computer in order to use the repository.

**Install the repository on your computer:**

1. Create a folder on your computer. Remember the path to the folder.
2. Open the terminal or Git Bash.
3. Change directory to your just-created folder using the command: **cd path-to-folder**.

Screen%20Shot%202018-10-11%20at%2013.39.21.png

1. Now you’re in the right folder, you have to initialize the Git repository on your computer. We can do that using the command: **git init**.
2. Go to the GitHub website and go to your repository. On the right of the page, you’ll find a button called: “Clone or Download”. Click on it and copy the link that you see.
3. In the Terminal or Git Bash, type in the following command:

Screen%20Shot%202018-10-11%20at%2013.47.39.pnggit remote add origin (the link you’ve copied) and press enter.

1. The repository has been successfully installed on your computer. If you initialized the repository with a readme, you will have to pull the file on your computer. You can do this by using the command**: git pull origin master**. (Master is the name of the main branch that has been created automatically). We will get into branches later.
2. Go to your folder on your computer and check if the readme is in the folder.

Screen%20Shot%202018-10-11%20at%2013.52.37.png

# Pull and Push explained

**Pull:**

Pulling basically means that you are ‘pulling’ the latest version of your GitHub repository to your computer. You should always do this when you start working again on the project. This is because a co-user of the repository might have worked on the project and pushed it to the GitHub repository. **If you don’t pull the latest version to your computer and just go to work on the project, Git will reject your push since your version is not the same as the GitHub version.** You can pull the latest version by using the command: git pull origin (branch name).

**Push:**

Pushing means that you ‘push’ your work to the GitHub repository. To do this you have to do the following things. In this tutorial, I will use the readme as an example.

1. Edit the readme file that you’ve pulled.
2. Save the readme file.
3. Open the Terminal or Git Bash
4. If you are not in the right directory, change directory using the following command: **cd path-to-directory.**
5. Add the file to the push-queue using: **git add .** (Don’t forget to add the dot at the end of the command. The dot stands for “All edited files”, just like \* stands for everything in SQL.
6. The files have been added to the queue.
7. Screen%20Shot%202018-10-11%20at%2014.08.44.pngNow you have to commit the changes to the queue. You can do this by using the command: **git commit –m “Some information about the changes here”**.
8. You are ready to push the changes to the GitHub repository. Use the following command: **git push –u origin master**. Good to know: If you can’t push it for some reason, change **–u** to **–f.** This will force the push.

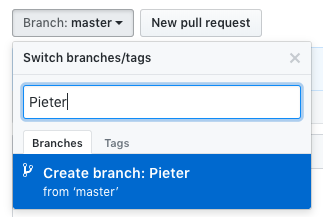
# Introducing GitHub Workflow – Setting things up

Now that you understand how Git and GitHub works, we will get into the GitHub workflow. Let me remind you that this is one of the many workflows that you can use for Git/GitHub projects. I chose this workflow because it works the best in my opinion.

**The workflow:**

A repository has branches for each individual that’s collaborating with this repository. For example, if I’m collaborating with someone called Pieter, I should have 3 branches: master, Jeroen and Pieter. Everyone is going to work within his/her own branch. That means that they are only going to push to their own branch. The Master branch will never be touched.

Let’s say you are the project leader and you’re the one who’ve created a repository on GitHub. The first things you should do is:



1. Invite all participants to the GitHub repository. You can do this by going to: **Repository** **Settings > Collaborators** and add everyone. They will receive an email when you invite them.
2. Create a specific branch for everyone that’s collaborating**.** In this example, I will add my own name, and someone called “Pieter”.
3. Check your branches. If you have multiple branches, you good.

# Getting a branch back-on-track with the master branch

Now that everyone has their own branch, they can start working in it. If the branch is not mutual with the master branch, the collaborator should pull the master branch and push it. If your or someone else his branch is not on track with the master branch, you or the collaborator will have to do the following to get it back-on-track.

1. First, open the Terminal or Git Bash, change directory to the right directory and type in the following command: **git checkout branch name**. The branch name should be the name of the person. I.e. git checkout Jeroen.
2. Now the terminal is working in the right branch.
3. Pull the content from the master branch to your computer using the command: **git pull origin master.**
4. Push the content you’ve just pulled to the GitHub repository by using the following command: **git push –u origin branch name**. NEVER use master, since you’re not working in the master branch, but in your own branch.
5. Check if the branch is on-track with the master branch. You can do this by going to your branch on GitHub.

Screen%20Shot%202018-10-11%20at%2014.57.41.png

# The GitHub Workflow

This is the workflow you have to use every day. I will explain this in a few steps:

* What do I have to do when I start working on the project (again)?
* What do I have to do when I’m finished for the day?
* What to do with the pull-requests at the end of the day?

**What do I have to do when I start working on the project (again)?**

First thing you do is check if your branch is on-track with the master branch. You can do that by following the “Getting a branch back-on-track with the master branch” step.

If your branch is on-track with the master branch, follow the following steps:

1. Open the terminal or Git Bash
2. Change directory to the right directory
3. Checkout to your branch using: **git checkout branch name**
4. Pull the master branch on your computer using: **git pull origin master**. Your computer is now on-track with the content in the master branch and you can start working on the project.

**What do I have to do when I’m finished for the day?**

Follow the following steps every-single-day when you’re done working on the project for the day:

1. Open Terminal or Git Bash
2. Create a queue using: **git add .** All your edited files have been stored in the queue.
3. Commit the changes using: **git commit –m “Some information about the changes”**.
4. Push the queue to your GitHub repository using: **git push –u origin branch name**. Branch name MUST BE your branch.
5. Go to the GitHub website
6. Go to the GitHub repository
7. Navigate to your branch
8. You should see that this branch is one (or more) step(s) ahead of the master branch, and you see a button called: “Create pull-request”. Click on it.
9. Follow the steps and commit the pull-request.
10. You’ve successfully uploaded your files to GitHub.

**What to do with the pull-requests at the end of the day?**

The project leader should look at all the pull-requests that have been made that day. He should be responsible for merging all pull-requests into the master branch.

The project leader can merge the pull-requests by following the steps:

1. Click on “Pull-requests”.
2. The project leader should see all the requests that have been created by each individual.
3. Press the button “Compare and merge” on each pull request. This will update the master branch.

**Screen%20Shot%202018-10-11%20at%2015.21.36.png**

**DO NOT FORGET TO ALWAYS PULL THE MASTER BRANCH WHEN YOU START WORKING ON THE PROJECT AGAIN. THIS IS BECAUSE THE FILES ON YOUR COMPUTER IS NOT ON-TRACK WITH THE MASTER BRANCH ON GITHUB. IF YOU WORK ON THE PROJECT WITHOUT PULLING IT FIRST, YOU WILL LOSE ALL CHANGES YOU’VE DONE THAT DAY.**