# Lab 1 - Creating a Repository

*Note: For the sake of these tutorials, we are assuming you are using a terminal capable of running Bash commands. For Windows users, the recommended terminal program for this is Git Bash, which you can download [here](https://gitforwindows.org/).*

There are a few ways to create a repository. The first is from the command line.

|  |  |
| --- | --- |
| **Command** | **Function** |
| git **init** | Initialise a directory as a Git repository |

You can then create a remote directory in GitHub with the following steps:

1. Create a new directory on your local machine and initialise it as a Git repo with the following commands:

$ mkdir my-new-repository

$ cd my-new-repository

$ git init

1. Create a new README.md file with the following command:

$ touch README.md

1. Stage and commit the new file to the repository:

$ git add README.md

$ git commit -m "Initial commit"

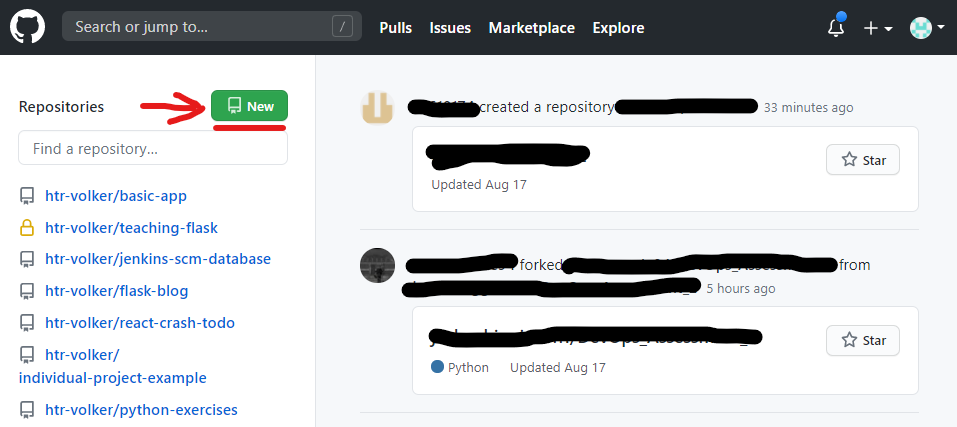
This will complain that git doesn't know who you are. So, tell git using the following command:

$ git config user.email "<your github email address>"

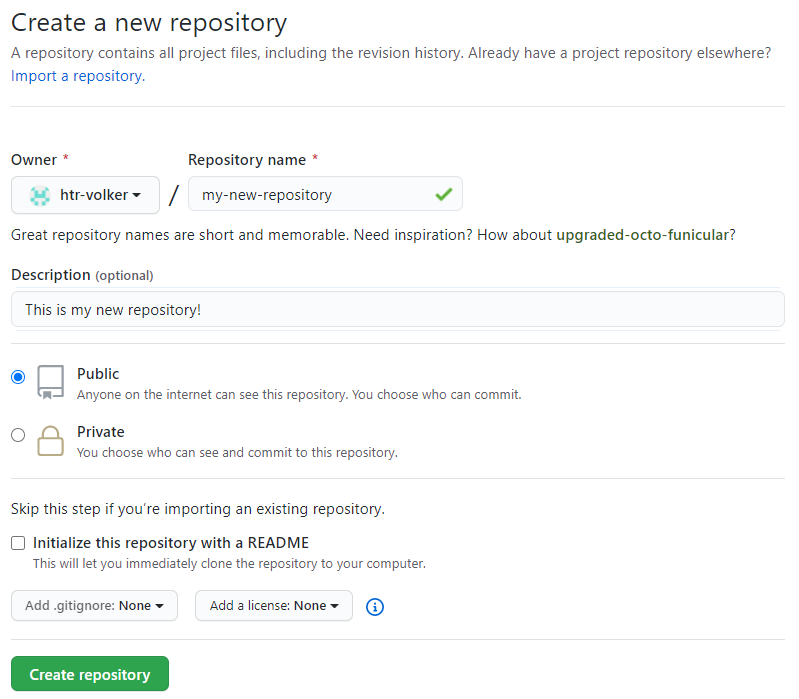
Then run the commit command again:

$ git commit -m "Initial commit"

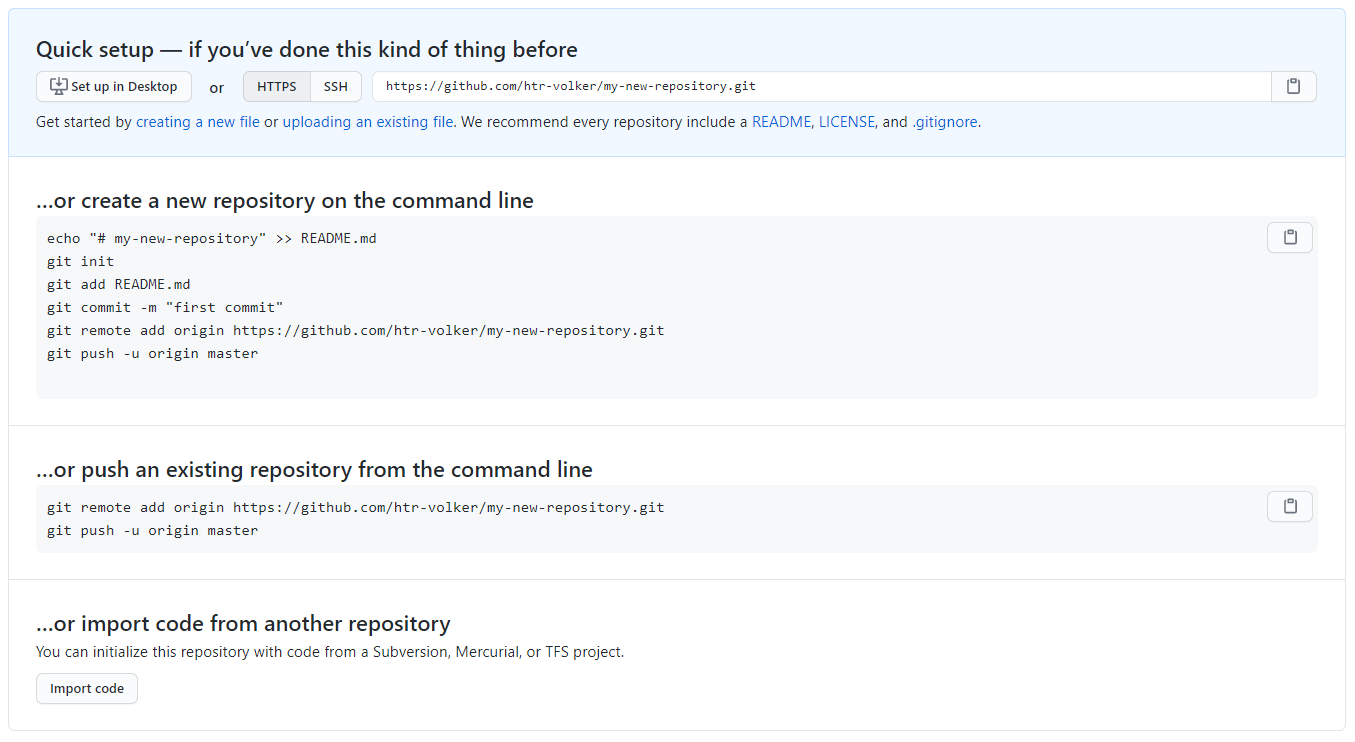
1. Navigate to GitHub on your browser and log in
2. Click on the green **New** button to create a new repository.



1. Enter the name of your new repository and press the **Create repository** button (keep the **Initialize this repository with a README** box unchecked)



1. You should see a page like this, with some options for working with your repository. Copy the URL (with or without the .git extension, it doesn't matter) for the repo as specified at the top of the page:



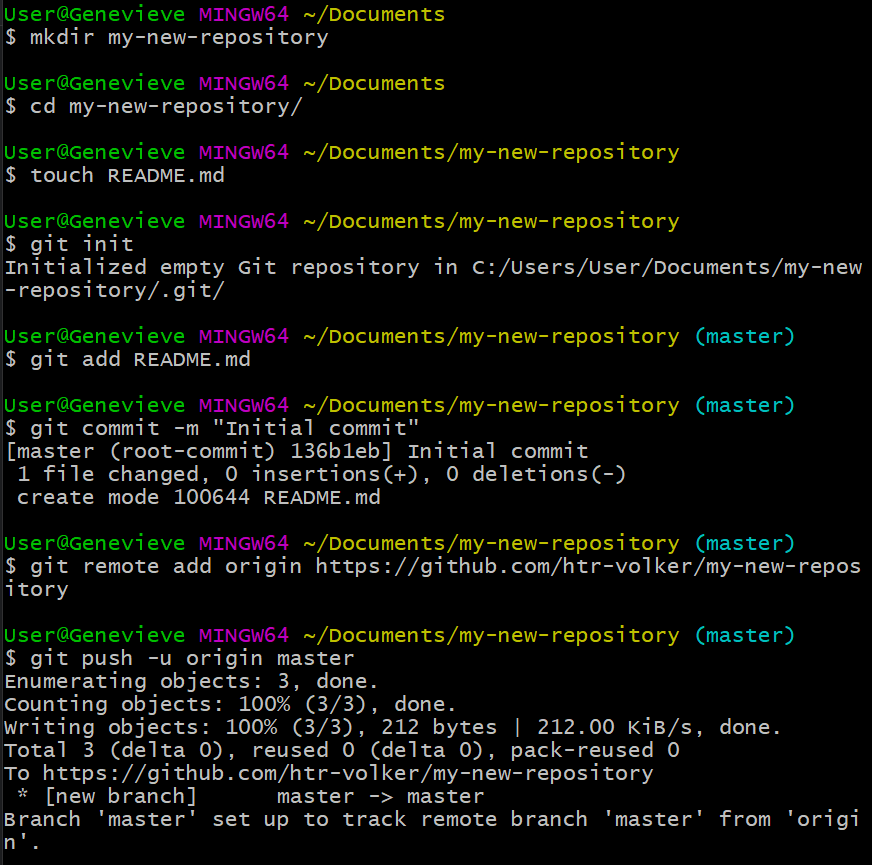
1. Finally, fun the following commands in your Bash terminal, replacing the URL for that of your new repository:

$ git remote add origin https://github.com/[YOUR\_USERNAME]/[YOUR\_REPOSITORY]

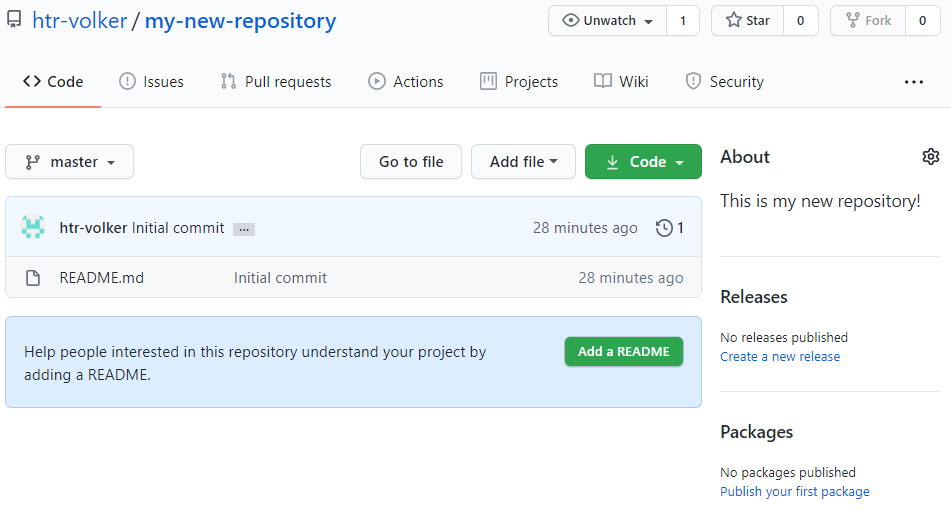
$ git push -u origin master

You might be asked for your GitHub credentials at this point, so that your PC can authenticate against the remote GitHub website. So, if necessary, enter your GitHub username and password.

Your terminal output should look similar to this:

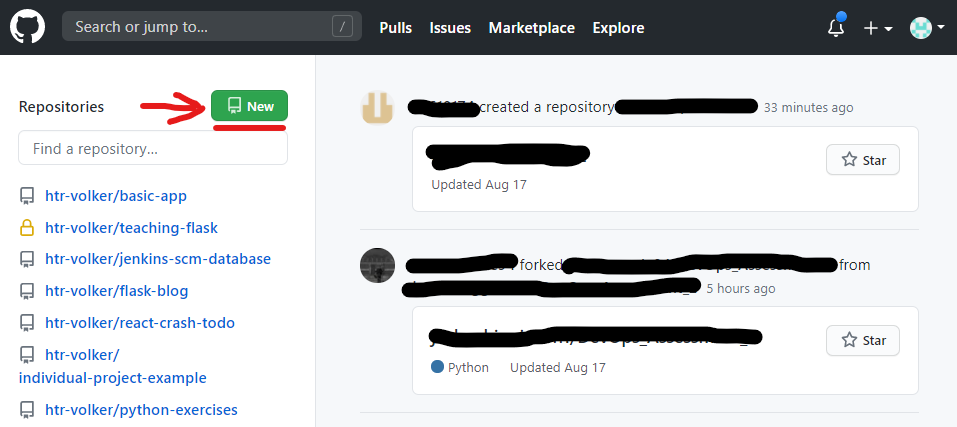


1. Refresh your GitHub repository page on your browser to see your README.md file

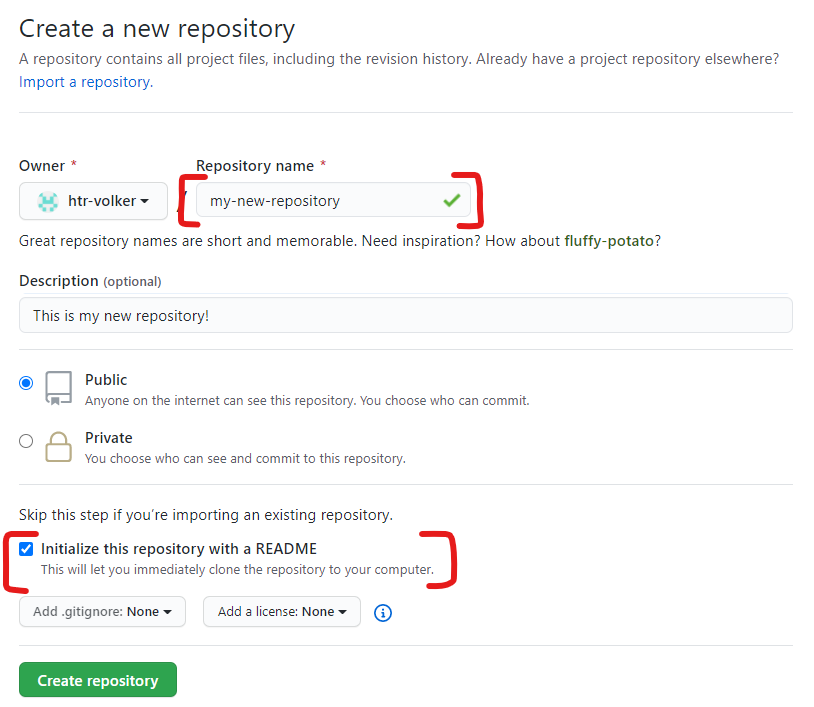


Alternatively, you can create the repository in a remote location, such as on GitHub.

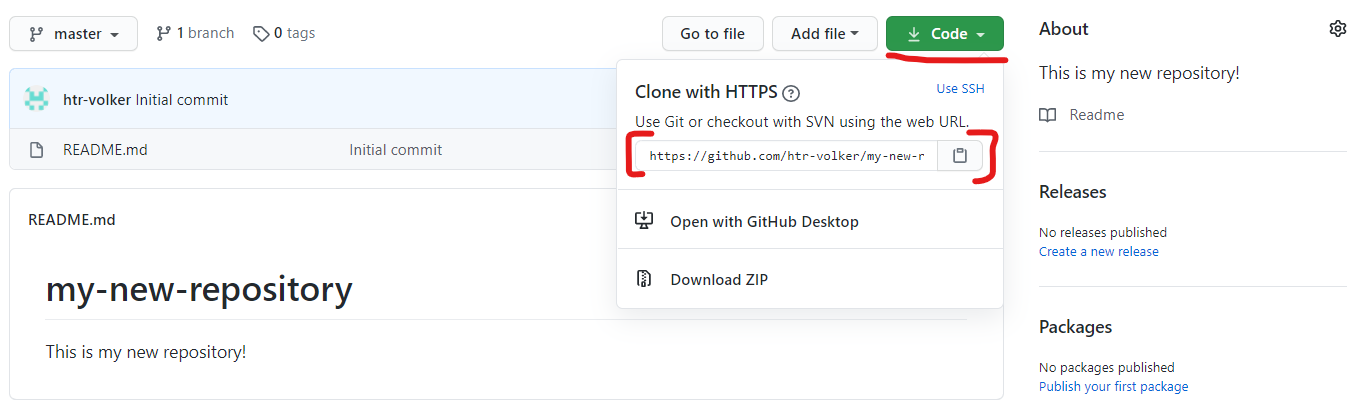
1. Click on the green **New** button to create a new repository.



1. Give your new repo a name, select the tickbox to **Initialize this repository with a README** and click **Create repository**



1. Click on the **Code** button and copy the URL provided under the **Clone with HTTPS** section



1. Enter the following command in your Bash terminal:

$ cd ..

$ git clone https://github.com/[YOUR\_USERNAME]/[YOUR\_NEW\_REPO]

**Lab 2 - Getting to Grips with Git**

*Note: For the sake of these tutorials, we are assuming you are using a terminal capable of running Bash commands. For Windows users, the recommended terminal program for this is Git Bash, which you can download [here](https://gitforwindows.org/).*

Let’s put some of these commands to use. Referring back to the commands we’ve covered in this document, do the following:

1. Create a new Git repository in GitHub called **git-practice** (refer to Intro to Source Control for a refresher - include a README.md file)
2. Clone the repository to your home directory and make it the current working directory with the following commands:

$ cd ~

$ git clone https://github.com/[YOUR\_USERNAME]/git-practice

$ cd ~/git-practice

1. Configure the user.name and user.email properties.
2. Run the following command to create some files in the repository:

$ touch file.java file.cs file1.txt file2.txt

1. Stage the java file and commit it
2. Stage all text files and commit them
3. Push the changes to your remote repository
4. Add a new file called file.py:

$ touch file.py

1. Stage all files and commit them
2. Push the changes to your remote repository

## Lab 3 - Branching

*Note: For the sake of these tutorials, we are assuming you are using a terminal capable of running Bash commands. For Windows users, the recommended terminal program for this is Git Bash, which you can download [here](https://gitforwindows.org/).*

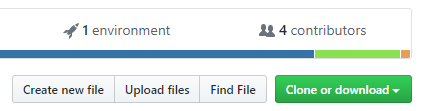
Follow along with these tasks to get familiar with branching in Git.

1. Run the command cd ~
2. Run the command git clone https://github.com/jordan-grindrod/scripts.git
3. Run the command cd ./scripts
4. See which branches are currently configured for that repository
5. Create a new branch called develop
6. From develop, checkout to a new branch called issue-1
7. Delete the issue-1 and develop branches

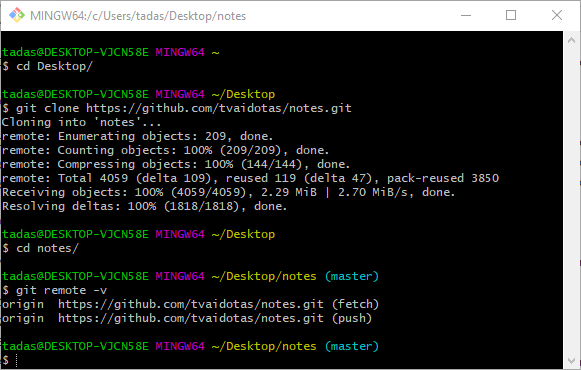
## Lab 4a - Forking a repository

*Note: For the sake of these tutorials, we are assuming you are using a terminal capable of running Bash commands. For Windows users, the recommended terminal program for this is Git Bash, which you can download [here](https://gitforwindows.org/).*

This practice gets you to fork a remote repository.

1. Make sure you're logged into your GitHub account
2. Go to any public repository available. [This one](https://github.com/tvaidotas/FlaskAppBasic) for example:  
   https://github.com/qa-apprenticeships/FlaskAppBasic
3. Click on **Fork** in the top right of the page for your chosen repository. This will create a copy of the repository under your account  
     
   Fork >
4. You will be redirected to your account's version of the repository
5. Click on the green **Clone or download** button, and copy the URL  
     
   
6. Open a Bash terminal in the location you want to clone your project
7. Run the command git clone [URL], but replace [URL] with the repository URL you copied in the previous step
8. Change directory to the project you just cloned
9. Run the following command, to confirm that you have cloned the correct repository:

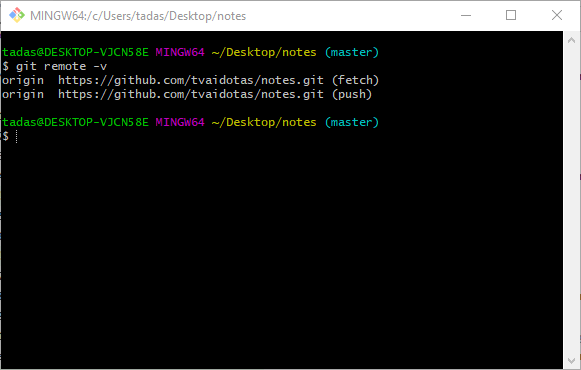
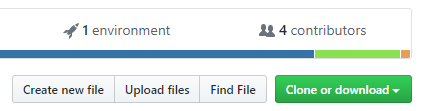
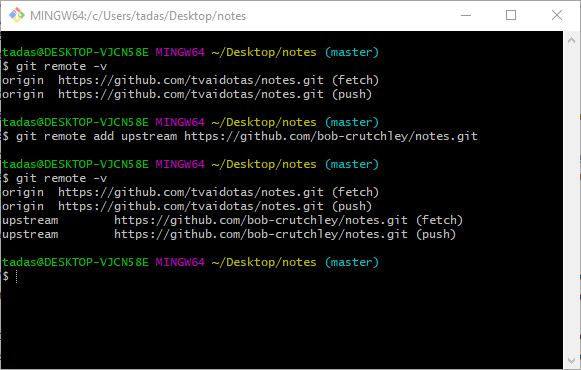
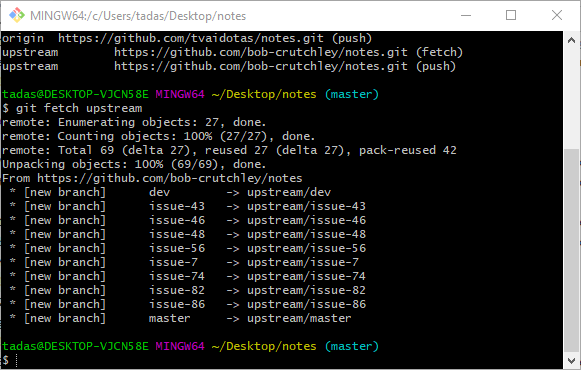
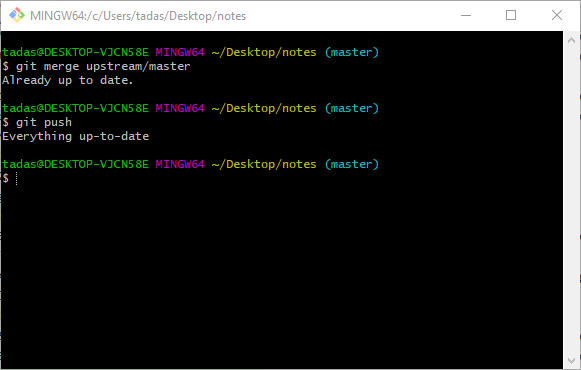
$ git remote -v

You should get a similar output for fetch and push, but the URL will be pointing to your repository.  
  


## 

## Lab 4b - Updating forked repository from original

Now you will set up your local project to receive updates from the original repository. This is required, so that when the owner of the original repository adds new functionality, bug fixes etc., you will be able to get these changes as well.

1. Open your Bash terminal, in the root directory of your project. You should be able to see that you're on the master branch. Next, you want to execute git remote -v to make sure that the upstream to the original repository is not set up yet.  
   You should see a similar output to this.  
     
   
2. In your browser, go to the original git repository and copy the repository URL. You can do this by clicking on the green **Clone or download** button and copying the URL  
     
   
3. Execute the following command in your Bash terminal, but replace the [URL] with the one you just copied:  
    git remote add upstream [URL]
4. Next, you want to check that this has worked, which you can do by executing the git remote -v command.  
   The output should look similar to this:  
     
   
5. Now you will pull the changes from the original repository into yours, which you can do by executing the git fetch upstream command in your Bash terminal. The output will depend on whether there are new changes or not:  
     
   
6. You will want to update the master branch, so we will merge the *upstream/master* into the *local origin/master*. You can do this  
   by executing the git merge upstream/master command. Keep in mind that there may be merge conflicts that you will need to resolve. If you have resolved the merge conflicts, or if there were none, you should *push* to update your *origin/master* branch.  
     
   

## Lab 5 - Creating a Pull Request

*The following practice can be performed entirely on the GitHub website. Refer to the Pull Request section of the Learner Guide to see how to do a Pull Request on GitHub.*

Create a new feature branch on a GitHub project you’ve already made (or create a new project, if necessary). Make some changes on this feature branch, then create a pull request to have the new feature merged into the project.

Next, fork a repository from someone in your class (any repository will do. Make some changes in this forked repository and create a pull request to have your feature merged into the original project.