

Using Git and Github

Why use git? (summarized)

Version control – all changes to your code are documented and can be backtracked.

Collaboration – each change is attributed to a developer and the changes of multiple developers can be *merged* to allow working on multiple aspects of a project simultaneously.

Branching – working in a branch allows you to experiment on a project safely without breaking anything.

Setting up Git


1. Go to <https://github.com/> and set up an account. Choose the free option when given the choice.
2. Verify your email address and click “Start a project”

Create a new repository


A repository contains all the files for your project, including the revision history.

Owner

Repository name


 jaapdd ▾

 /


SophEE 

Great repository names are short and memorable. Need inspiration? How about **animated-guacamole**.

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**


You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

 |

Add a license: **None** ▾ 

Create repository

3. Give your project (or “repository”, “repo”) a name and check the box to create a README.
4. From here on, we’re going to be using command line tools, so the instructions will differ between Windows and OSX.

Windows Installation (Up-to-date Windows 10 only): For windows, we will be enabling the “Windows Subsystem for Linux” that will allow us to use bash and Linux command line tools in Windows. I am a BIG fan of this as it allows me to do everything I love in Linux on a Windows machine, without the downsides of a virtual machine.



- a. Open the windows menu and type “powershell”. Right click the program and click Run as Administrator.
- b. Run the following command

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

- c. Restart your computer when prompted.
- d. Open the Microsoft store and install Ubuntu

Store

Home Apps Games Music Movies & TV Books Microsoft

Search

Ubuntu
Canonical Group Limited
★★★★☆ 22
Free
Get Share

Everyone

Screenshots

Description

Ubuntu on Windows allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

To use this feature, one first needs to use "Turn Windows features on or off" and select "Windows Subsystem for Linux", click OK, reboot, and use this app.

The above step can also be performed using Administrator PowerShell prompt: Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux

More

Available on

PC

Features

Ubuntu
bash
ssh

What's new in this version

20170619.1 build of Ubuntu 16.04 LTS

- e. Once it's done click Launch.
- f. Enter a username and password.
- g. Awesome, now you're running a Ubuntu terminal. Type `sudo apt update` and press enter. Wait for it to finish and type `sudo apt upgrade`

and again, press enter and respond to the prompts.

These two instructions will check for updates and install them; a good idea to do before installing anything new.

h. Now let's install Git. Type

```
sudo apt install git-all
```

and press enter. When it's done, continue to [Using the Command Line](#)

Mac OSX Installation:

- a. Open your Applications folder, then the Utilities folder and open "Terminal"
- b. Type `git --version`. If you don't already have git installed it will prompt you to install it. If this doesn't work you can try the following link <https://git-scm.com/download/mac>

Using the Command Line

1. If you don't already have it open now, open a Bash shell. On Windows this means opening the "Ubuntu" window and in Mac open "Terminal".
2. A "Bash shell" is a Command-line interface. In short, this means you can type commands and the computer will do it. The alternative to this is a Graphical User Interface (GUI) which is what you're probably used to using when you click the files and folders and whatnot on your computer.
3. Try typing `echo "hello world"` and press enter. The computer should literally echo "hello world" back to you.

The first word you type is the command or program that you're running and anything that follows are arguments passed to the program.

4. Here are the basic commands you should know:
 - a. `ls` - lists all the files in your current folder (directory)
 - b. `cd` - "change directory," will change your current directory to whatever comes after `cd`. For example, `cd Documents` will move you into the directory "Documents". Press tab to autocomplete the directory name you're typing.
`cd ..` will move you up one directory, to the one that contains your current directory.
 - c. `mkdir` - Make a new directory
 - d. `cp` - Copy a file
 - e. `mv` - Move a file
 - f. `rm` - Delete a file

Setting Up Git

1. Set your username by typing:

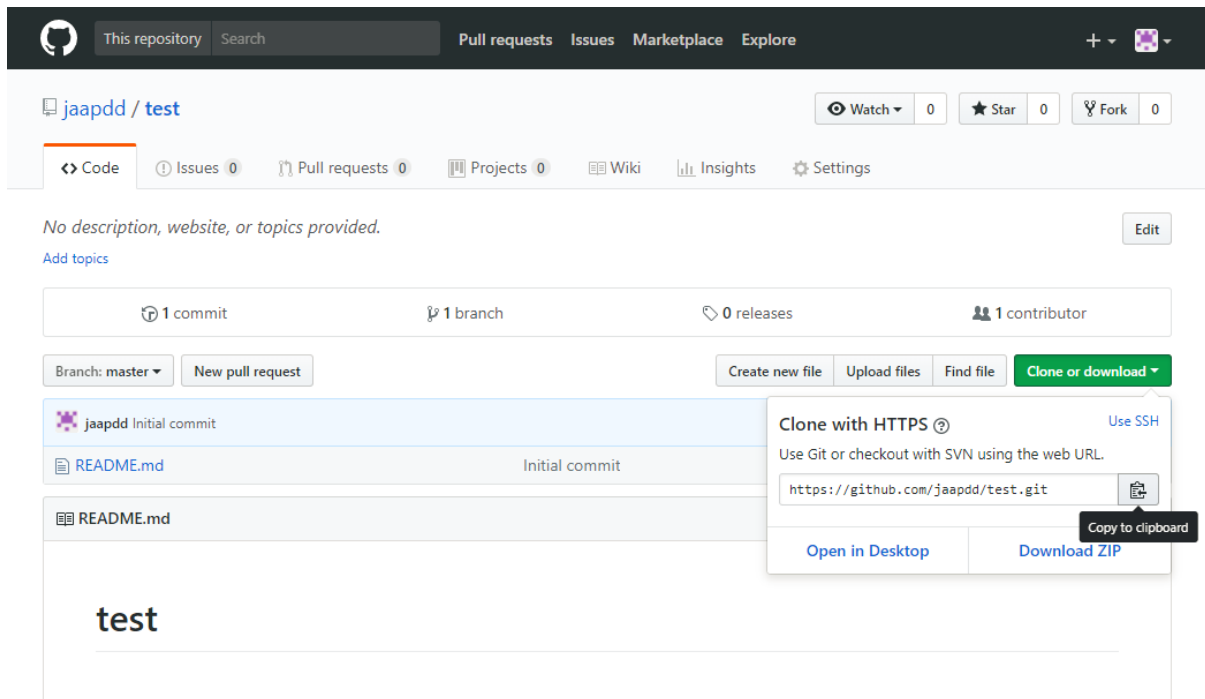
```
git config --global user.name "Enter your name here"
```

2. Connect your email address by typing:

```
git config --global user.email email@example.com
```

Make sure this is the same email address as you use on Github.

3. Go to Github in your browser and open the repository we created at the start. Click “Clone or download” and copy the address given.

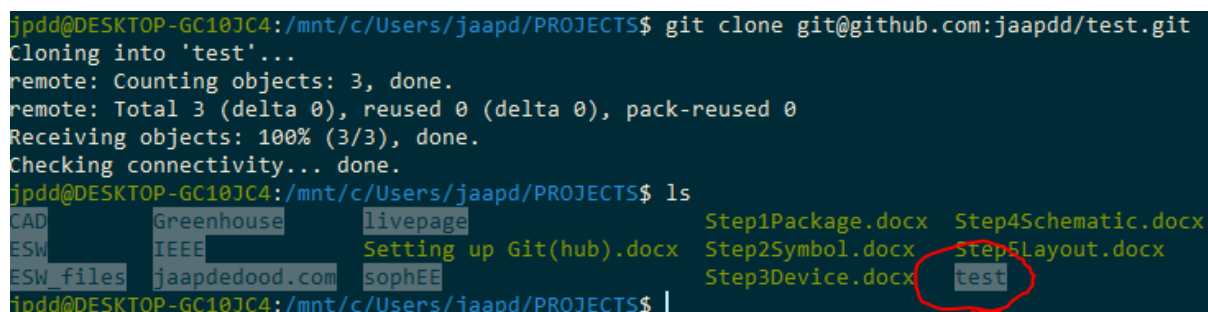


4. Go back to your bash shell window and navigate to where you want to save your files using `cd` and `ls`.

5. Type

```
git clone
```

followed by the address you just copied. Press enter. Your repo will be downloaded to your current directory. If you type `ls` you should see it there!



6. Navigate into your repo folder using `cd` and type `ls`. You should see the README file here. Typing `ls -a` will list all directories including hidden ones, and you will see the `.git` directory. This is how you know you’re in a directory linked to a git repository.

```

jpdd@DESKTOP-GC10JC4:/mnt/c/Users/jaapd/PROJECTS$ cd test
jpdd@DESKTOP-GC10JC4:/mnt/c/Users/jaapd/PROJECTS/test$ ls
README.md
jpdd@DESKTOP-GC10JC4:/mnt/c/Users/jaapd/PROJECTS/test$ ls -a
.  ..  .git  README.md
jpdd@DESKTOP-GC10JC4:/mnt/c/Users/jaapd/PROJECTS/test$ |

```

Pushing to your Git Repository

1. Let's create a file in our repo and "push" it to our repository.

Type

```
nano hello.txt
```

Nano is a command line text editor that comes with most Linux operating systems. If this doesn't work on OS X try `open -e hello.txt` to use TextEdit.

2. Type some text into your file and save it by pressing Ctrl + X, answer Yes to saving the file and press enter to confirm. Type `ls` to double check the file was created.

3. Run

```
git add *
```

to add all your *changes* to the repo to an index

4. Run

```
git commit -m "My first commit"
```

to commit your changes. In the quotation marks you usually want to enter a short description of what you're changing with this commit e.g. "Fixed typo in motor function"

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAHAHAHAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT
MESSAGES GET LESS AND LESS INFORMATIVE.

Note that commit does not upload your changes to the server. You can see it as packaging your list of changes with a shipping label before sending it out.

5. Finally, run

```
git push origin master
```

to finally upload your changes to the remote git server.

```
jpdd@DESKTOP-GC10JC4:/mnt/c/Users/jaapd/PROJECTS/livepage$ git add *
jpdd@DESKTOP-GC10JC4:/mnt/c/Users/jaapd/PROJECTS/livepage$ git commit -m "sophee demonstration"
[master 5a1c734] sophee demonstration
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
jpdd@DESKTOP-GC10JC4:/mnt/c/Users/jaapd/PROJECTS/livepage$ git push origin master
Username for 'https://github.com': Jaapeddood
Password for 'https://Jaapeddood@github.com':
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 336 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
remote: This repository moved. Please use the new location:
remote:  https://github.com/Jaapeddood/livetestingpage.github.io.git
To https://github.com/Jaapeddood/livetestingpage.git
   8549932..5a1c734  master -> master
jpdd@DESKTOP-GC10JC4:/mnt/c/Users/jaapd/PROJECTS/livepage$ |
```

If all the commands ran successfully, you should be able to see hello.txt appear in your repo on the Github website!

Bookmark or save this cheat sheet

http://rogerdudler.github.io/git-guide/files/git_cheat_sheet.pdf

For future reference of the most important commands.

The most important to remember are probably the last three you ran: add, commit, push. This is what you'll run every time you want to upload a change you made to your code to the repo.

Collaborating with others using git

1. Open your repository on Github and navigate to Settings -> Collaborators and add a friend.
2. If they now clone the repo as you did before, you can both push, pull and branch in the repo.
3. Pulling is the opposite of pushing i.e. you download the latest version of your project from the server. Try having a friend push a file to the repo and then pull it to your own local version by running

```
git pull origin master
```

You should now see the file in your local version.

4. You can get by with only pushing and pulling to a single "master" branch for this project, but a popular mantra for using git is "branch early, and branch often." I would suggest becoming familiar with git a little first, just pushing and pulling, and then invest some time into learning how to successfully using and merging branches.

<https://learngitbranching.js.org/> ← Will teach you branching and merging much better and less boring than me

5. Finally, before you close this and start working on your project, I suggest you *fork* my SophEE repo so that you have all the files I want to provide to you. Forking simply means making a copy of a repo to one of your own.

Visit my repo on github at <https://github.com/Jaapdedood/sophEE> [Blueprint](#) and click “Fork”. Done. You now have a repo of your own with all the files you need. Clone this to a local location and get coding!