Test

Setup

To make everything as easy as possible we need to follow a few simple steps. Start by generating a ssh key for the computer you work on. This is unique to every computer you work on.

```
ssh-keygen -t rsa -b 4096 -C "your_name@youremail.com"
```

(If there already is a ssh key stored you can either overwrite it by pressing *y*. If you know you need the other ssh key for something else please look at the more advanced github tutorial). It will now ask you where to save the file. Chose default by pressing `enter`. It will then ask you for a passphrase, we will leave this blank. To set this press `enter` twice. Now we add this ssh key to the ssh-agent by

```
ssh-add ~/.ssh/id_rsa
```

Now we need to do some stuff on your github account. - Go online and login to your account. - Press the small icon with your avatar in the top right corner and chose settings. - Now go to *SSH and GPG keys*. - Click *New SSH key* - In the *Title* box type in the name of the computer you are adding in (Ex: Amadeus, Icenine, My_macbook) - Keep the browser window open in the background and get back to the terminal.

We want to copy the content of the ssh key to Github.

```
cat ~/.ssh/id_rsa.pub
```

This will type out the content of the ssh key in the terminal, a really long key with weird characters. Copy the content, including `ssh-rsa` all the way to your email adress at the end. Now past this into the `Key` box on your github account in the browser that is in the background. Finish by pressing `Add SSH key`.

What we have done here allows you to pull and push data from private Github repos without typing in your password every time.

Clone your first repo

Open the terminal and navigate to where you want to keep your scripts. To get started and download a repo you use the `clone` command. Example with T2 analysis

```
> git clone -b develop git@github.com/ubcmri/T2_MWI.git
```

This have created a new folder in your current directory. Navigate to is

```
> cd T2_MWI
```

To see which branches currently exist write

```
> git branch
* develop
```

As you can see, you only have one branch here. This is good, we want everyone to work on the develop branch.

Now make some edits on your files. Then add and commit the changes

```
> git add myfile.txt
> git commit -m 'Swedish translation'
```

At this stage, all changes you have made are local, let's get them to the central repo as well!

```
> git push
```

Now your changes have been pushed to the main repository.

The next time we start working on this we need to make sure we have the most current updates. To do this we `pull` the latest changes.

```
> git pull
```

Create your own feature branch

Let's say you want to get started on your own feature locally, it is good to start your own branch. To do this type

```
> git branch my_feature
> git checkout my_feature
```

Now you are on your new feautre branch. The `checkout` command moves you to the new branch. Make some edits here, add and commit. Now we want to go back to the develop branch and merge our edits

```
> git checkout develop
> git merge my_feature
```