Introduction to Git

Julia Beni

version January 17, 2018

Data Science Friday 2 Aims

Today's Data Science Friday will introduce you to GitHub and Linux commands to navigate and manage files and directory structures. We will be using GitHub as a platform to store materials disseminated for class and we will create a dedicated repo for your MICB425 final portfolio work (more information about this to come).

Preliminary check: Is git installed on your computer?

Let's check this simply by typing `git` in the terminal window and then pressing enter.

Note, text formatted as "git" was above are commands that can be copy-pasted into a terminal window

This should populate the terminal window with git commands.

Configure Git command line for your GitHub account

We will link your GitHub account to work done on the command line using the following configurations settings. This configuration will be useful later on when you are pushing documents to your own repository. In the terminal, please type:

```
git config --global user.name "Your Name"
```

```
git config --global user.email "youremail@email.com"
```

This needs to be the same email that you used to set up your GitHub account

Introductory Linux commands and setting up a directory for MICB 425 GitHub work

The first commands that we will use for this tutorial represent some of the basic commands used on the Linux command line. First, we want to navigate within our file structure to where we will keep our cloned copy of the MICB425 GitHub repository. Let's keep the clone in your Documents. To get there, on the command line we will use the change directory command like this:

```
cd ~/Documents
```

Note the use of ~ in the above command; this is a shortcut key to denote the Home directory. Note that "directory" is a synonym for "folder".

It is often very useful to know where you are located within a directory structure on the Linux terminal. Ensure that you are located in the Documents directory by running the print working directory command. *(In English, the working directory is the folder you are currently located in.)* This is achieved as follows:

```
pwd
```

If you are a native Windows user, this command should output /C/Users/your_computer_name/Documents/. Mac or Linux users will see /Users/your_computer_name/Documents/.

You need to be located in the Documents directory before proceeding.

Clone the github repository for MICB425 from the web

Now we will clone the MICB425 repo located on the GitHub to your local computer. You will clone the repo as it currently exists on the EDUCE GitHub and rename the repository in one step as follows:

```
git clone https://github.com/EDUCE-UBC/MICB425 MICB425_materials
```

Once the download is complete, investigate the cloned file structure of the directory you have downloaded. First, change directories into the directory where you cloned the MICB425 repo:

```
cd MICB425_materials
```

Now you can see the files in the new directory using the list command as such:

```
ls
```

At this point you have downloaded all the current materials for MICB425 Module 01

Version control: identifying changes to the master repo and retrieving them

One of the primary strengths of GitHub is for version control of documents. Git will automatically track your documents and determine if any changes have been made to either your cloned repo, or the master. Enter the following command to verify that your cloned repo is identical to the master located on the GitHub site:

```
git status
```

When your local, cloned copy is up-to-date (identical) with the master repo, this is the message you should see after running `git status`:

```
dhcp-206-12-198-54:MICB425_materials Julia$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

“Your branch is up to date with ‘origin/master’.” is the important message here.

Now we will demonstrate what happens when there is a delta between your clone and the master repo. In order to update our local git connections, we need to retrieve them from the staging space on GitHub. This is done by the fetch command in the terminal:

```
git fetch
```

Below is the message you should see after running git fetch:

```
dhcp-206-12-198-54:MICB425_materials Julia$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/EDUCE-UBC/MICB425
   4da3619..b3a531f  master     -> origin/master
```

```
dhcp-206-12-198-54:MICB425_materials Julia$ git status
On branch master
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
```

Figure 1:

```
dhcp-206-12-198-54:MICB425_materials Julia$ git pull
Updating 4da3619..b3a531f
Fast-forward
 SURPRISE.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 SURPRISE.txt
dhcp-206-12-198-54:MICB425_materials Julia$
```

Figure 2:

Now, if there are any changes between your local cloned repository and the master, they will be recorded. In order to see if there are changes, run `git status` once more in the terminal.

The status command has noticed that your local cloned directory is one document behind the master. It also suggests how to remedy this situation - by using the pull command. Let's do that (Figure 1):

```
git pull
```

The result of the pull command looks like this (Figure 2):

You can see from the command output that the document SURPRISE.txt was added to your clone.

Git fetch and git status are important commands. Please check the status of your cloned repository regularly as we will be updating the course materials available on the master GitHub repo often. When you discover that the master repo has been updated, ensure that you pull down any changes and keep your cloned repo up to date.

Create and populate your own repository to hold assignments and course portfolio (eventually)

Now we'd like to create a repository where you will keep the course materials you are turning in for credit for this course. To start this, you will first create a directory on your local computer, then we will push that directory to your GitHub account as a new repository. We should still be located in the directory where you cloned the MCIB425 repository. Let's check that by running

```
pwd
```

This should output the directory location ~/Documents/MICB425_materials. You need to create a directory in Documents outside of MICB425_materials to be added to your own distinct repository. You can move out of the MICB425_materials directory by typing the following command:

```
cd ..
```

Note: the "." is used as a shortcut to refer to the current working directory. ".." starts you in that current directory and then moves you up one directory level.

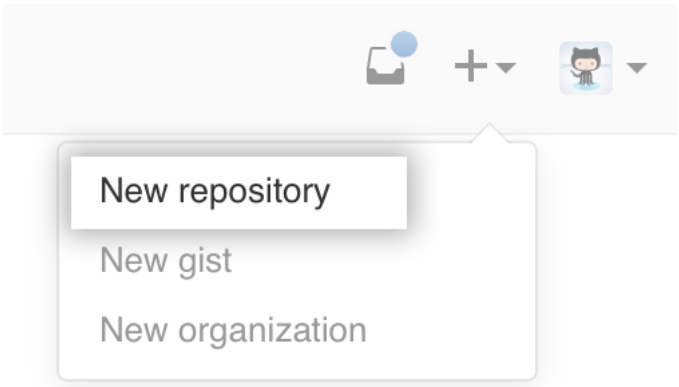


Figure 3:

You should now be located in the ~/Documents directory. Verify this using the `pwd` command.

Next we will create a new directory with the make directory command. Let's title this directory MICB425_portfolio, and create it using the following command syntax:

```
mkdir MICB425_portfolio
```

Finally, you will populate your MICB425_portfolio folder with an empty txt file titled ID.txt. We will use this file in the future.

To start, change directories into the directory you just created.

```
cd MICB425_portfolio
```

Once located in the correct directory, write the empty .txt file this way:

```
touch ID.txt
```

If you are uncertain whether the above command worked, you can use 'ls' to verify if ID.txt is in the MICB425_portfolio folder.

Creating your own repo and pushing a local directory to it

Now you'll create a repository on your GitHub account and push a local directory to it. Start this process by signing into your GitHub account on a web browser. Here we're going to use the GitHub GUI for part of the process and finish the repo push using the command line. In reality there are many ways to skin this repo-pushing-cat, but this tutorial shows you one way. In the upper right-hand corner of any GitHub page (once you've signed in), you will see an image that looks like this:

Create a new repository titled MICB425_portfolio. Do **not** initialize the repo with README, gitignore, or license files for now. **Do not navigate away from this webpage.** We will use information found on it shortly.

Now we will return to the terminal. **Ensure that your working directory is still the MICB425_portfolio directory** we just finished creating. From this directory we will initialize this directory as a new GitHub repo:

```
git init
```

The next step after initializing the repo is to add the files into the GitHub index. Think of the index like a file staging area. Adding files to the index loads up the files we want to push to our repo without finally committing them to our account. For our purposes this may seem unnecessarily cautious, but this feature is helpful when many people are collaborating on one single project and the stakes of breaking the code are high. So let's add:

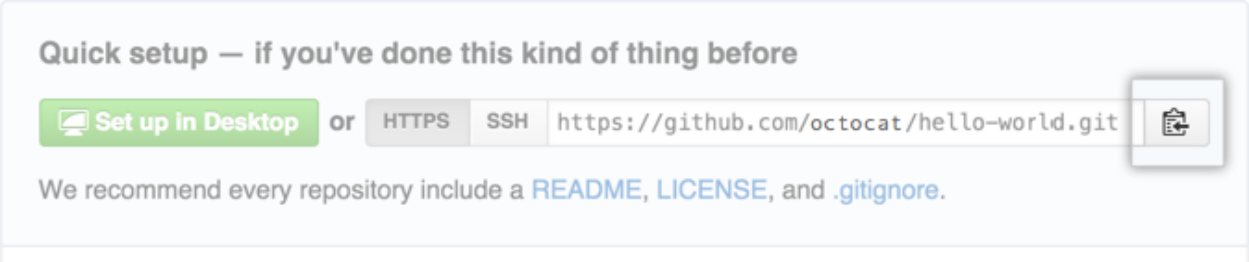
```
git add .
```

Now that our files are in the staging area, let's commit them to our repo. We do this using this command.

Note the "-m" on the command. This option is to annotate the commit with the comment "First commit" Comments can be useful when adding in new repos.

```
git commit -m "First commit"
```

Next, switch back to the GitHub GUI webpage where you started the repo. You should see information that looks like this



giving the URL for the repo you've just initiated. Include that URL in the command below:

```
git remote add origin https://remote_repository_URL
```

It is useful to verify the web URL for the new remote repo. You will do that with the following command:

```
git remote -v
```

And lastly, push your repo up to GitHub:

```
git push -u origin master
```

Verify on your GitHub page that the repository you created was successfully pushed.

Congratulations! You now have a repository started to house your MICB425 portfolio!

A bunch of other resources for learning git

There are many resources available online for learning to use the git command line, but here are several that I found useful:

- https://git-scm.com/book/en/v2
- https://gist.github.com/davfre/8313299
- https://stevebennett.me/2012/02/24/10-things-i-hate-about-git/
- https://git-scm.com/docs/gittutorial
- http://rogerdudler.github.io/git-guide/
- https://code.tutsplus.com/tutorials/quick-tip-how-to-work-with-github-and-multiple-accounts--net-22574

- https://software-carpentry.org/lessons/