

# instructions\_week1

January 12, 2022

## 1 Introduction

Welcome to the Microbial Genomics Laboratory course for the Spring 2022 semester at The George Washington University! This is a computer-based lab course which will teach you how to use latest and free computational tools to analyze microbial genomes and metagenomes.

Detailed course syllabus for the course can be found [here](#).

### 1.1 About this documentation

This documentation system includes a collection of instructions on computational exercises designed for you throughout the semester. It is impossible to properly display the html pages of these instructions on Blackboard. In order to display the pages correctly, you will need to download the files through our course Github repository, which is located here:

[https://github.com/SawLabGW/MicrobialGenomicsLab\\_Spring2022](https://github.com/SawLabGW/MicrobialGenomicsLab_Spring2022)

However, in order for you to see this repository, you will need to create a Github account and let me know your username and email used to create an account there. Only then can I add you into the list of people allowed to view the pages and files there.

You cannot download the contents of this page just simply through this web address but they need to be “cloned” to your computer through a special command known as `git`. We will be setting up these necessary commandline tools during the first week of class.

## 2 Week 1

### 2.1 Agenda for (01/12/2022)

- Introduction to the course
- Setting up your computing environment
- Introduction to Unix/Linux commands
- Install Miniconda
- Install git
- Install Jupyter lab
- Install and test some bioinformatic tools
- Assignments:
  - Assignment 1
  - Reading assignments

## 2.2 Introduction to the course

This week, due to the spike in Omicron variant of the SARS-CoV-2, we will be meeting virtually on WebEx.

There will be a Powerpoint presentation given over WebEx or Zoom. I will go over the details of how the class will be taught and what platforms we will be using in the Powerpoint presentation but I outline below these for you to remember.

### 2.2.1 Classroom location

When we're not meeting virtually, we will be meeting physically in the basement of Rome Hall. The classroom is located in **Rome B104**.

### 2.2.2 Where is the virtual classroom?

We will be meeting either on **WebEx** and/or **Zoom** platforms, depending on how well they work. WebEx record time stamps of when a person enters and leaves the room so it helps me to use WebEx for recording attendance. Currently, the WebEx link for the class is shown below.

<https://gwu.webex.com/gwu/j.php?MTID=mf096d6b29554d8fdd2b17519e9ea2357>

Meeting number: 2624 258 8428

Password: ppFmVRME223

### 2.2.3 Besides the virtual classroom, what else will we be using?

Besides using either WebEx or Zoom, we will use **Slack** to communicate, discuss, and disseminate information such as reading materials, videos, code snippets, etc. You will get an invitation to join a Slack group created for this course. While Blackboard works fine, due to the cross-listed nature of the course, it becomes very difficult to organize content across both courses on Blackboard. We will use Blackboard for grading and submission of assignments but for sharing of content, we will use Slack.

We will also be using **Github** to share certain course content that cannot be easily shared on Blackboard. For example, the computational instructions you are seeing now was shared through a Github repository. Therefore, you will need to create an account on Github website and let me know your account information so that I can add you as a member of the course and you will be able to download course materials.

Computational instructions will be updated on a weekly basis and will be available on Wednesday at noon. You will need to use `git pull` command to update course repository and will be able to view the updated instructions each week.

## 3 Setting up your computing environment

Computational exercises you will be performing in this course require operating systems (OS) with Unix or Linux-like commands. This means you need to set up your environment to make sure those commands are available in a commandline environment, for example a terminal window. It is **crucial** that you set this up correctly/properly as the course heavily relies on tools that need to be run in command line environment.

The following sets of instructions will help you to set up your environment correctly to make sure that these sets of commands are available. If you are already using Linux or Unix operating system (OS), then you are all set. If not, you will want to follow the instructions to configure your environment.

### 3.1 Windows OS

Your Windows OS should already come installed with Windows Powershell (if the OS version is fairly recent), which you will need for the following instructions.

You will need to turn on Windows Subsystem for Linux (WSL) feature on in order to enable and run Unix/Linux commands natively on Windows operating system. You can follow instructions on this page on how to set up and configure your Windows OS to enable Unix commands:

<https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/>  
and here:

<https://docs.microsoft.com/en-us/windows/wsl/install-win10>

Please follow the instructions shown in both pages I have referred to above. WSL would allow you to use `bash` and Unix commands natively in Windows OS. You will also have an opportunity to install a Linux distribution as shown in both pages. I would highly recommend that you choose Ubuntu as the distribution as it is more widely used and has more user support than other distributions.

Briefly, the steps described in those two websites involve the following:

- enabling WSL on Windows Features
- installing a Linux distribution system (Ubuntu, for example) on Windows

Some instructions require that you run the Windows Powershell in administrator mode. To do that, you need to right click on the Windows Powershell icon to display other options and select the option that says “run as Admin” or something similar.

Setting up the command line environment for Windows can be a bit challenging compared to Mac OS but it is not impossible.

### 3.2 Mac OS

Setting up the command line environment in Mac is a lot easier as Mac OS evolved from BSD, which is an offshoot of earlier Unix operating systems. Consequently, there are more Unix-like tools already built into the operating system. Nevertheless, it is not quite the usual GNU Unix/Linux type of commands you might want to use. In order to set up the environment correctly, follow the instructions below.

This website has pretty good instructions (although a bit old) on how to set up the command line environment without having to install the whole Xcode, which is usually a precursor needed to install command line tools and requires a large space.

<https://mac-how-to.gadgethacks.com/how-to/install-command-line-developer-tools-without-xcode-0168115/>

Basically, you would first:

1. Open your terminal

2. Then type this: `xcode-select --install`
3. Then follow the instructions from the pop-up windows and it should install the necessary tools for the command line environment.

A few additional websites on how to set up the command line environment on Mac OS are shown below:

<https://osxdaily.com/2014/02/12/install-command-line-tools-mac-os-x/>

<https://www.maketecheasier.com/install-command-line-tools-without-xcode/>

Once the steps shown in these pages are followed, your command line environment should be ready for this course.

### 3.2.1 Install Homebrew

Next, you want to install Homebrew, which allows you to use the `brew` command on your Mac OS terminal. It is a package managing system that allows you to install basic utilities and even some bioinformatics tools on your Mac without needing admin privileges.

- First, go here and follow the instructions: (<https://docs.brew.sh/Installation>)
- Open your terminal
- Then type:

```
cd
```

```
mkdir homebrew
```

```
curl -L https://github.com/Homebrew/brew/tarball/master | tar xz --strip 1 -C homebrew
```

```
brew install coreutils findutils gnu-tar gnu-sed gawk gnutls gnu-indent gnu-getopt grep
```

The first `cd` commands takes you back to your home directory where you want to be installing Homebrew. The rest of the commands shown will install GNU Unix commands, which are different from original Unix-like commands that come installed with Mac OS.

Once Homebrew is installed, it will be very easy to install other tools such as `git`, which you will need to check out the course Github code repository and see the instructions. Remember, `brew` command is your best friend on any Mac OS to install and configure tools you would need for pretty much anything.

## 4 Unix and commandline environment

### 4.1 Introduction

You will be learning how to use bioinformatic tools that run in command line environment. Part of being able to run these tools require at least some basic knowledge of Unix (or Linux) commands. In this Wiki page, I will explain how to use some of these commands but not all of it.

### 4.2 Useful list of commands

This website has a handy list of Unix/Linux commands you will be using mostly.

<https://files.fooswire.com/2007/08/fwunixref.pdf>

I will display it below for a quick preview.

```
[8]: from IPython.display import IFrame

IFrame("https://files.fooswire.com/2007/08/fwunixref.pdf", width=800,
      ↪height=800)
```

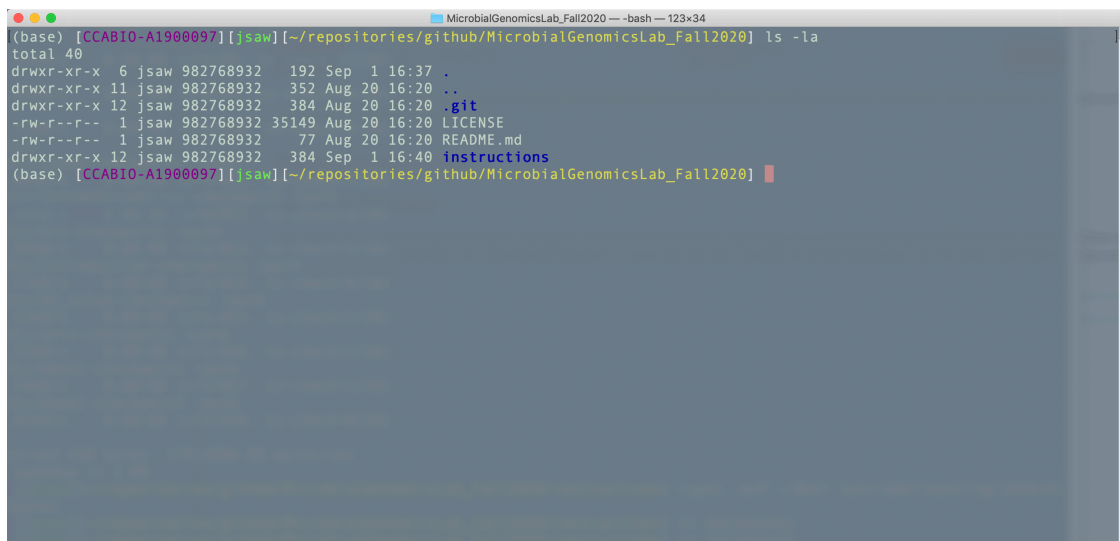
```
[8]: <IPython.lib.display.IFrame at 0x104824c90>
```

As you can see, it categorized commands by the type of activity such as working with files or checking processes. Please learn these commands as they are some of the most basic commands any bioinformaticians can use.

### 4.3 Basic Unix commands

First, before you can start using these sets of commands, you must have set up your computing environment to install basic utilities that allow you to use these commands. Make sure you refer to the section on “Setting up your computing environment” to make sure you have correctly set up the environment. You will need to use a terminal or Jupyter-lab to be able to use Unix commands.

Mac OS typically comes with a terminal application installed and you will need to look through your list of applications. On Windows, you will need to use the Linux Subsystem with an Ubuntu (or a different Linux) distribution installed in order for you to use a terminal. Terminal on Mac OS looks something like this:



```
MicrobialGenomicsLab_Fall2020 — -bash — 123x34
(base) [CCAB10-A1900097][jsaw] [~/repositories/github/MicrobialGenomicsLab_Fall2020] ls -la
total 40
drwxr-xr-x  6 jsaw 982768932  192 Sep  1 16:37 .
drwxr-xr-x 11 jsaw 982768932  352 Aug 20 16:20 ..
drwxr-xr-x 12 jsaw 982768932  384 Aug 20 16:20 .git
-rw-r--r--  1 jsaw 982768932 35149 Aug 20 16:20 LICENSE
-rw-r--r--  1 jsaw 982768932   77 Aug 20 16:20 README.md
drwxr-xr-x 12 jsaw 982768932  384 Sep  1 16:40 instructions
(base) [CCAB10-A1900097][jsaw] [~/repositories/github/MicrobialGenomicsLab_Fall2020]
```

#### 4.3.1 Navigating between directories

To navigate between folders, the most basic command you can use is `cd`, which means to change directory to something. First, before you do anything, to see where you are located, type `pwd`. `pwd` stands for print working directory. For example, if I were to type this in my terminal (using Jupyter-lab as an interface), you would see something like this (ignore the first line that says `%%bash`. It is for Jupyter-lab environment, which I will talk about a little later):

```
[16]: %%%bash
cd
pwd
```

/Users/jsaw

First, typing the `cd` command without any parameters brings you to your “Home” directory in any Unix-like environment. Here, my home directory is “jsaw” under “Users”. If I want to navigate into a folder under “jsaw” named “tools”, then I would type:

```
cd tools
```

```
[17]: %%%bash
cd
cd tools
pwd
```

/Users/jsaw/tools

Now, I’m in a folder named “tools”. If I want to go back up one folder into “jsaw”, then I would type:

```
cd ..
```

The two dots mean I want to go back up to a higher ranking folder than current one.

#### 4.3.2 Creating and removing directories

So, first thing for you to do is to navigate to your home directory and create 4 folders named as `tools`, `repositories`, `data`, and `exercises`.

Type in your terminal:

```
cd
mkdir tools repositories data exercises
ls
```

The `mkdir` command means “make directory” and it will create a directory after the `mkdir` command.

The `ls` command means list the contents of a directory. When you type this command in your home folder, you will see a bunch of folders such as “Documents”, “Downloads”, etc, in addition to the 4 folders you have just created. You will be using these folders for the following purposes:

- `tools` (any software tools that I might make you download and install prior to exercises)
- `repositories` (this is where you will “clone” the course repository into, or any other public repositories of interest)
- `data` (this is where you will store data that we will be using in the class)
- `exercises` (this is a folder for you to do your course exercises and assignments)

**Warning:** Unix/Linux commands are case-sensitive, which means you need to type exactly as shown. If you type `Ls` instead of `ls`, the command is not going to work. Next, you also want to make sure that files and folder names do not have *any* spaces between them. This will avoid a lot of headaches down the line. For example, if you were to create a folder name “My Folder” by

typing `mkdir My Folder`, it will create two separate folders named “My” and “Folder” instead of what you have intended.

### 4.3.3 Copying, renaming, and moving files/folders

To copy one file to another location, you can type something like this:

```
cp file.txt some_folder/
```

And this command will copy the `file.txt` file into the `some_folder` folder. If you want to rename the `file.txt` into `file2.txt`, then type:

```
mv file.txt file2.txt
```

You can use the same syntax to rename folders. For example if you want to rename the `some_folder` into `my_folder`, then type:

```
mv some_folder my_folder
```

### 4.3.4 Removing/deleting files/folders

If you want to delete a file or folder, you type something like this:

```
rm file.txt
```

The `rm` command will remove the file. To remove a directory, you can either type `rmdir` or `rm` commands depending on whether the folder is empty or not. If the folder is not empty (has some files in it), you will need to type something like this:

```
rm -rf non_empty_directory
```

**Warning:** Be very careful with the `rm -rf` command. It will not ask you for confirmation on whether you *really* want to remove something. It will just silently carry out the task. You do not want to accidentally remove anything important using this command. A lot of people have learned this the hard way and they may shout and scream and pull their hair out after realizing what they have done.

### 4.3.5 File permissions

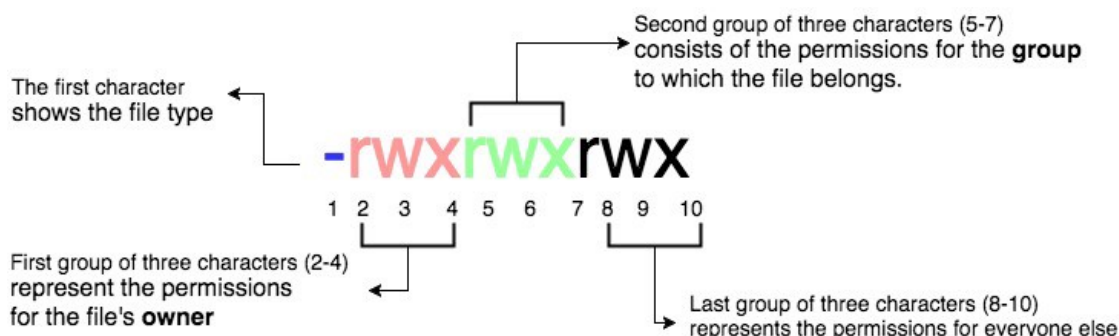
On a related note, you want to learn how to set permissions to files and folders so that other people can (or cannot) read/write/execute/delete or mess them up in anyway. For now, you are working on your own so this is not an issue, but on servers where you work with multiple people and groups, this is very important. If you type `ls -la` in your terminal, you will see something like this:

```

MicrobialGenomicsLab_Fall2020 --- -bash --- 123x34
(base) [CCABIO-A1900097][jsaw][~/repositories/github/MicrobialGenomicsLab_Fall2020] ls -la
total 40
drwxr-xr-x  6 jsaw 982768932 192 Sep  1 16:37 .
drwxr-xr-x 11 jsaw 982768932 352 Aug 20 16:20 ..
drwxr-xr-x 12 jsaw 982768932 384 Aug 20 16:20 .git
-rw-r--r--  1 jsaw 982768932 35149 Aug 20 16:20 LICENSE
-rw-r--r--  1 jsaw 982768932  77 Aug 20 16:20 README.md
drwxr-xr-x 12 jsaw 982768932 384 Sep  1 16:40 instructions
(base) [CCABIO-A1900097][jsaw][~/repositories/github/MicrobialGenomicsLab_Fall2020]

```

Here, you will notice that files (such as “README.md”) has some funny notations at the very beginning. These are file permissions. I have shown below what these mean.



For example, the first character shows the type of file. If it is a file, you will see a “-” but if it is a directory, you will see a “d”.

The `r` `w` and `x` following the first position mean read, write, and execute permissions for the owner (that is you, if you created this file). This means you have read/write/execute permissions for this file. If you’re missing a letter and instead have an “-” means you don’t have that permission. For example, if you want to remove the “read” permission of a file to others beside you and group members, you type like this:

```
chmod o-r file.txt
```

The `chmod` is the command for changing permissions. The `o-r` means you are removing (a minus sign) read permissions to others `o`. If you want to add write permission to group members, then you type something like this:

```
chmod g+w file.txt
```

The `g+w` means you are adding the write permission for group members.

These are some of the very basic Unix commands for you to learn for now. We will revisit and talk more about additional commands that will become necessary as the course progresses.



## 4.4 Install and configure Miniconda and Bioconda

Go here to download Miniconda package manager: <https://docs.conda.io/en/latest/miniconda.html>

### 4.4.1 Install Miniconda

Follow instructions for both Mac and Windows. Choose Python 3.7 version.

For **Mac**, I recommend you download the “pkg” version which is a graphical version of the installer.

For **Windows/WSL**, open your Ubuntu shell (right-click on the icon and run as Administrator). Then type:

```
cd /mnt/c
mkdir tools
cd tools
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
chmod +x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh
```

Follow the instructions along and make sure you say “yes” at the very end when it asked you if you want the installer to prepend it to path.

You can visit this page for more tips on how to install Miniconda on WSL.

<https://gist.github.com/kauffmanes/5e74916617f9993bc3479f401dfec7da>

#### After installation:

For **Mac**:

Close your terminal and open it again. Try typing `conda` to see if the command works. You will see a bunch of options you can type with the conda command.

For **Windows**:

Close your Ubuntu shell, then restart it by right-clicking on the Ubuntu icon and run it as Administrator. Try typing `conda` to see if it shows options.

### 4.4.2 Install Bioconda

After Miniconda is installed, you will need to configure Bioconda channels. Bioconda allows you to install *most* bioinformatics tools with ease without learning to use compilers or to figure out software dependencies.

Install bioconda by typing the following commands in your terminal:

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```

These commands will add bioinformatics related tools “channels” to miniconda. Now, you should be able to search and install tools locally on your computer. Try install jupyter first.

```
conda search jupyter
conda install jupyter
```

```
conda search jupyterlab
conda install jupyterlab
```

After you have configured the bioconda channels, most of the bioinformatics tools we will use in this course should be available for installation through `conda`.

## 5 Git and Github

### 5.1 What is git?

Git is a tool for version control that allows you to track changes so that you can work collaboratively in a team environment. A lot of software developers and engineers use it but scientists also use it to maintain their software/code generated for research. Although I will cover some basics on using Git, you will need to read up on what it is and how to use it at your own time. See here for more information:

<https://guides.github.com/introduction/git-handbook/>

Here is another resource on Git. It is for Bitbucket repository but command examples are the same as what you would be using with Github.

<https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud>

### 5.2 How to install Git

First, you will try to install `git` tool through command line tools such as Homebrew on Mac.

On **Mac OS**, you will type like this in your terminal: `brew install git`

On **Windows OS** running **Linux** subsystem with **Ubuntu**, you can type something like this in your terminal: `apt-get install git`

And this will install the tool on your command line environment. You may need root privileges to install tools on Ubuntu. If this happens, you just type: `sudo apt-get install git` and it should work. You will have to type your password though.

Another way of installing `git` is to use your Miniconda package managing system. It is actually my preferred method of installing new tools in my commandline environment. If you have set up Miniconda correctly, then you can just type:

`conda install git` in either Mac or Windows OS terminal and this would install `git` on your system. Here is an example of what you will see when you use Miniconda to search if `git` is in its list of packages that you can install through `conda` command.

```
[1]: %%bash
conda search git
```

```
Loading channels: ...working... done
```

#	Name	Version	Build	Channel
	git	2.12.2	4	conda-forge
	git	2.13.3	0	conda-forge
	git	2.14.1	0	conda-forge

git	2.14.1	1	conda-forge
git	2.14.1	pl526ha558ef4_2	pkgs/main
git	2.14.1	pl526hcdafd81_2	pkgs/main
git	2.14.2	0	conda-forge
git	2.14.2	1	conda-forge
git	2.14.2	2	conda-forge
git	2.14.2	3	conda-forge
git	2.15.0	pl526h6165b5f_0	pkgs/main
git	2.16.1	h74bb3f6_0	pkgs/main
git	2.16.1	pl526h74bb3f6_1	pkgs/main
git	2.17.0	pl526h028e6c8_0	pkgs/main
git	2.17.1	0	conda-forge
git	2.17.1	pl526h028e6c8_0	pkgs/main
git	2.18.0	0	conda-forge
git	2.18.0	pl526h028e6c8_0	pkgs/main
git	2.18.0	pl526hbb17d3c_1	conda-forge
git	2.19.0	pl526hbb17d3c_0	conda-forge
git	2.19.1	pl526h028e6c8_0	pkgs/main
git	2.19.1	pl526h28b1069_1000	conda-forge
git	2.19.1	pl526h28b1069_1001	conda-forge
git	2.19.1	pl526h6951d83_0	pkgs/main
git	2.19.1	pl526hbb17d3c_0	conda-forge
git	2.19.1	pl526hbb17d3c_1	conda-forge
git	2.19.2	pl526h28b1069_1000	conda-forge
git	2.19.2	pl526hbb17d3c_0	conda-forge
git	2.20.0	pl526h28b1069_1000	conda-forge
git	2.20.0	pl526hbb17d3c_0	conda-forge
git	2.20.1	pl526h28b1069_1000	conda-forge
git	2.20.1	pl526h28b1069_1001	conda-forge
git	2.20.1	pl526h3e3e3d1_1002	conda-forge
git	2.20.1	pl526h6951d83_0	pkgs/main
git	2.20.1	pl526hbb17d3c_0	conda-forge
git	2.21.0	pl526h3e3e3d1_0	conda-forge
git	2.22.0	pl526hbdf3604_0	conda-forge
git	2.23.0	pl526h6951d83_0	pkgs/main
git	2.23.0	pl526hbdf3604_0	conda-forge
git	2.23.0	pl526hbdf3604_1	conda-forge
git	2.23.0	pl526hbdf3604_2	conda-forge
git	2.24.0	pl526hdc91d69_0	conda-forge
git	2.24.0	pl526hdc91d69_1	conda-forge
git	2.25.0	pl526hdc91d69_0	conda-forge
git	2.26.0	pl526h561ab23_0	conda-forge
git	2.26.1	pl526hcc376a2_0	conda-forge
git	2.26.2	pl526hcc376a2_0	conda-forge
git	2.27.0	pl526hcc376a2_0	conda-forge
git	2.28.0	pl526hde3ca24_0	conda-forge
git	2.28.0	pl526hde3ca24_1	conda-forge

### 5.3 Cloning course repository on Github

As a member of the Microbial Genomics Lab 2022, you will have access to our code repository.

If you log in to Github and visit this URL:

[https://github.com/SawLabGW/MicrobialGenomicsLab\\_Spring2022](https://github.com/SawLabGW/MicrobialGenomicsLab_Spring2022)

you will be able to download the code repository to your laptop. First, if you haven't already downloaded the code repository to your drive, you want to "clone" it.

First, create a folder where you would want to store this repository. For example, in your command line environment, type `cd` to go back to your home directory. Then to create a folder named "repositories".

Type: `mkdir repositories`

Then go into that folder by typing: `cd repositories`.

Then type: `git clone https://github.com/SawLabGW/MicrobialGenomicsLab_Fall2020.git`

Now, you will see the folder "MicrobialGenomicsLab\_Spring2022" in the repositories folder.

### 5.4 Committing to the repository

For example, if you have a new file named `text.txt` which created and you want to make it part of the repository, you will need to type the command `git add text.txt`. You may type this command in the folder that this file is located or from the root folder. If it is in a subfolder named `alison/notes/text.txt`, then you want to type `git add alison/notes/text.txt` in the root folder.

After adding the file, you will need to type the following command.

`git commit -a`

It will bring up the vim editor, which will prompt you to enter a message before you can commit this file to the repository. If you would rather bypass this step, you can specify in your previous git command like this:

`git commit -m "created a new file"`

Here, you have put in quotes the text you wanted to convey along with the commit command to let people know that you have created a new file and added it to the repository. This helps your teammates and collaborators keep track of what you did to the repository.

Finally, type `git push origin master` to "push" your changes to the repository.

Before committing or pushing anything to the repository, you should always type `git pull origin master` to get the latest updated version of the repository. This is because there is a built-in check for people to prevent overwriting things or missing something if someone else had made a very recent commit to the repository.

**Note:** The repository you just downloaded is read-only and you cannot add or remove files from it. We will discuss how to create your own repositories later in the semester.

## 5.5 Install Jupyter and Jupyter-lab

We will make extensive use of Jupyter-lab to document work and to take notes. You will also need to use Jupyter-lab as a notebook that you will submit at the end of the semester for your final project. To install them, type in your terminal:

```
conda install jupyter
```

and

```
conda install jupyterlab
```

You will have to type “y” when you are prompted for confirmation after typing these commands. Now these tools should be installed in your command line environment.

### 5.5.1 How to use Jupyter-lab?

I will be demonstrating its use during class.

## 5.6 Tools to install for next week

Using `conda`, you will need to install a few tools that we will be using during class next week. The tools are:

- `sra-tools`
- `fastqc`
- `bbmap`

Using examples I have shown above, search and install these three tools on your computer.