CMSE 202 - Fall 2020
Set-up Guide Windows

Table of Contents

- Installing Python for this Course
  - If you have Anaconda on your computer already
    - Let's make sure Anaconda is updated to the latest version
    - Also let's check to make sure that Anaconda is in your path
    - If any of the above fail, remove anaconda and reinstall. To remove Anaconda follow these steps
  - If you don't have a fully functioning up-to-date installation of Anaconda
- Setting up a terminal
  - If you have Windows 10
    - Install WSL via Windows features
    - Install Ubuntu onto our WSL
    - Setup Ubuntu on WSL
    - Set-up Windows Anaconda on WSL
    - (Optional) Helpful Commands and Shortcuts
      - WSL Home Directory Shortcut
      - WSL to Windows 10 User Folder
      - Using Windows Terminal as main Terminal
      - Useful Commands
  - If you do not have Windows 10 or WSL installation did not work
    - Installing Git-Bash and Git
    - Adding Anaconda functionality to Git-Bash
    - Changing startup location to Home directory
    - (Optional) Helpful Commands and Shortcuts
- Setting up Spyder (Text Editor/Python IDE)
- MSU's JupyterHub Interface
  - Connecting to the engineering JupyterHub server
  - Getting Jupyter notebook files into JupyterHub
  - Making a copy of Jupyter notebooks from JupyterHub and turning them in
- Course Communication with Slack
  - Slack usage rules

As this is a course in computational modeling and data science, you will be completing all of your assignments using your computer! However, in order to do so there are a number of things you need to set up before the course starts. If you run into issues during this setup process make sure to document the error you encountered and send an email to your Professor to let them know that you ran into a problem.

MAKE SURE TO COMPLETE ALL OF THE SECTIONS LISTED IN THIS DOCUMENT BEFORE YOU COME TO CLASS

Installing Python for this Course

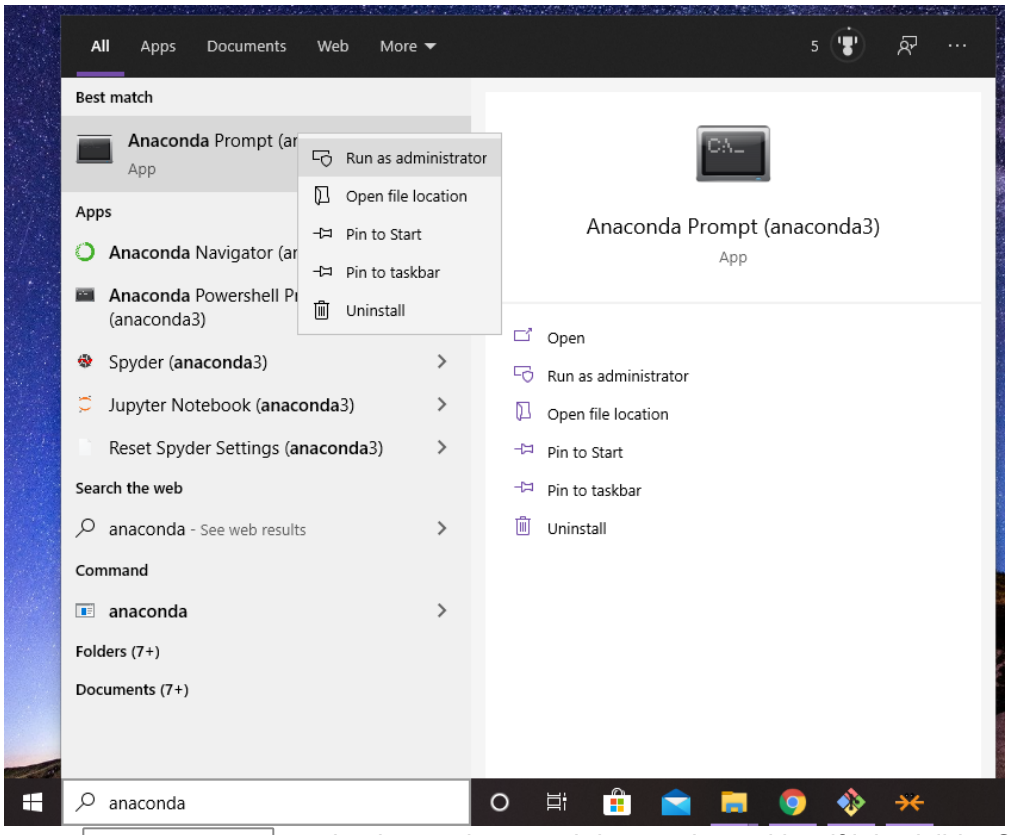
You need to have a functioning and **current** Anaconda Python installation on your computer for this course. If you have a past installation, you are expected to make sure it is up-to-date. In addition to making sure your installation is updated, you should also ensure that the Anaconda installation is in your default path.

If you have Anaconda on your computer already

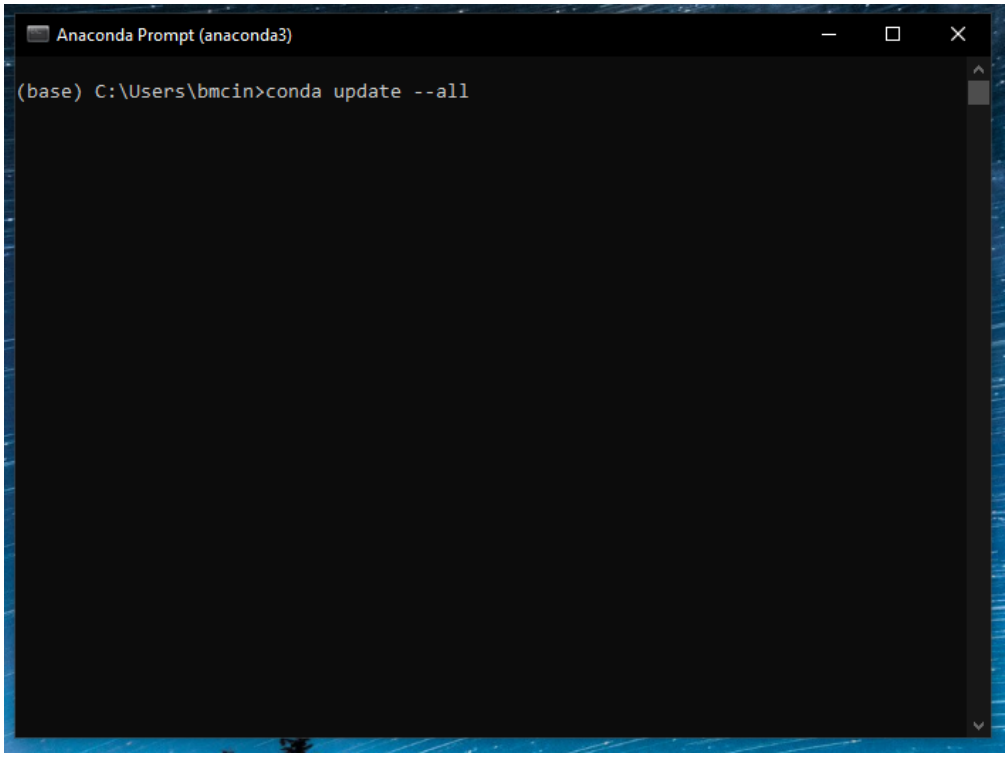
(If you do not have Anaconda click here)

Let's make sure Anaconda is updated to the latest version:

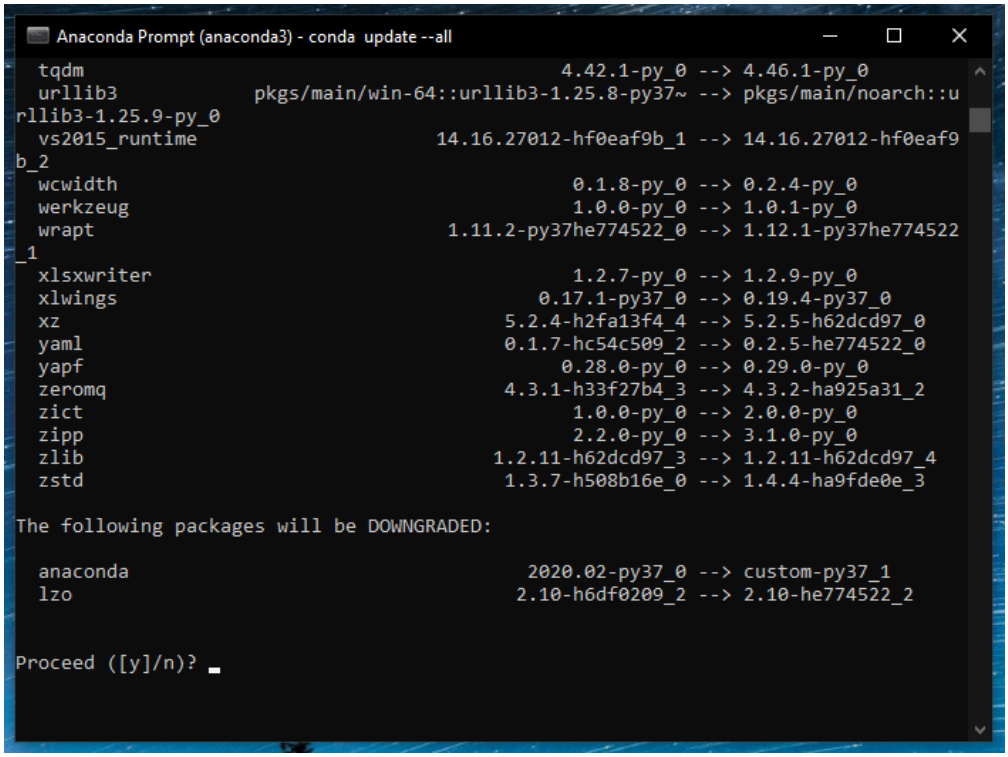
1. Be connected to the internet
2. Find your Anaconda prompt and update Anaconda.



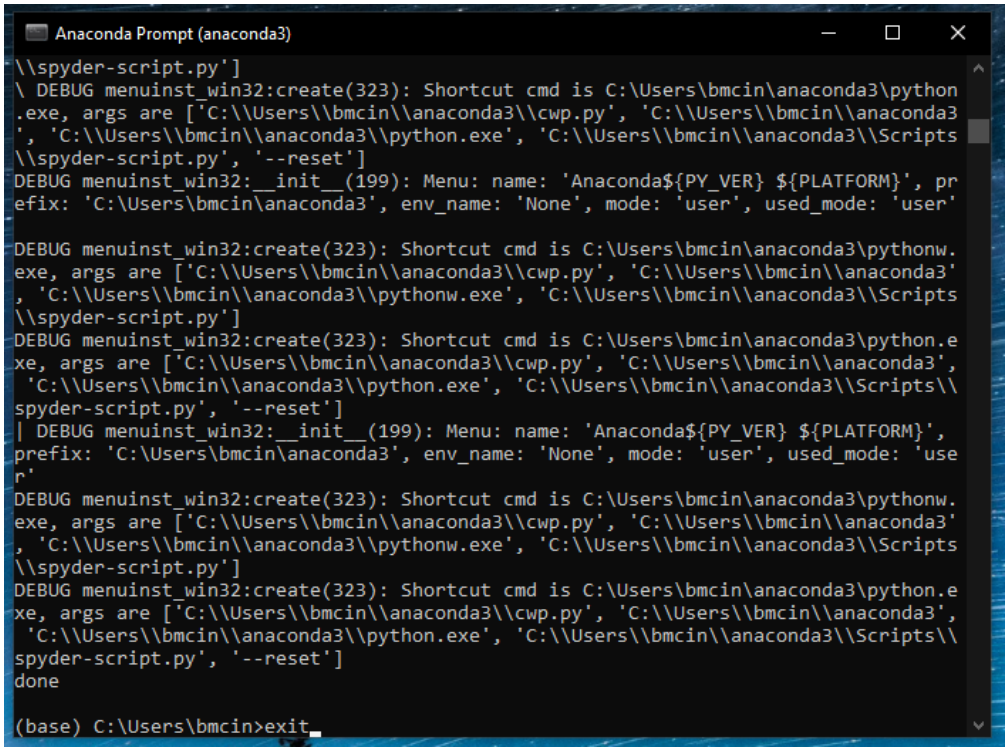
On keyboard press `Windows-key` or simply use the search bar on the taskbar if it is visible. Search `Anaconda Prompt` and right-click on the search result and select "Run as administrator".



Type in the command `conda update --all` and press `Enter`. This command will update anaconda.



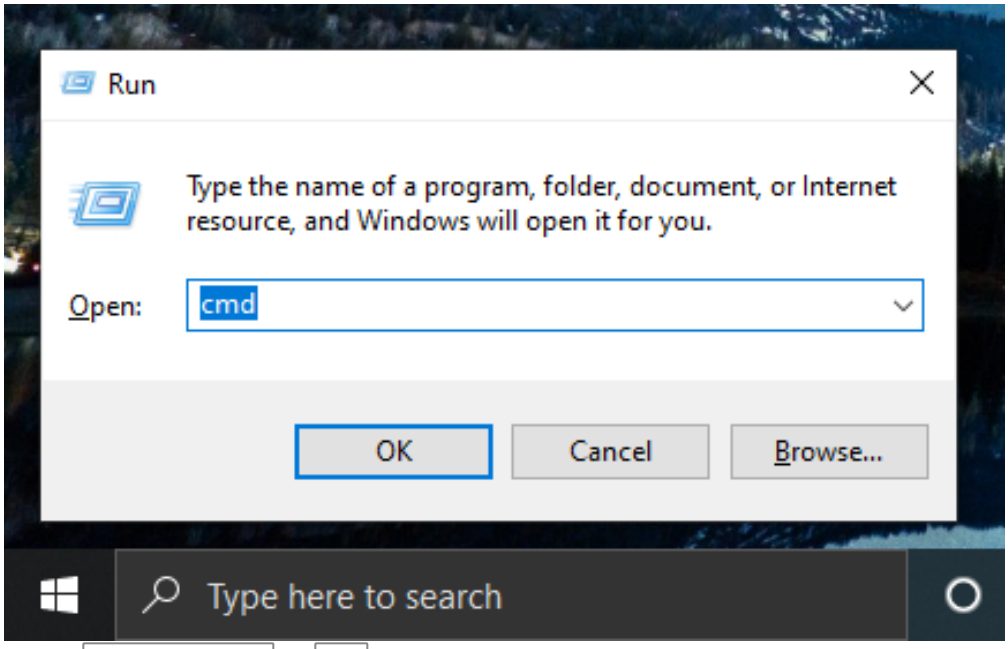
To continue type `y` and press enter.



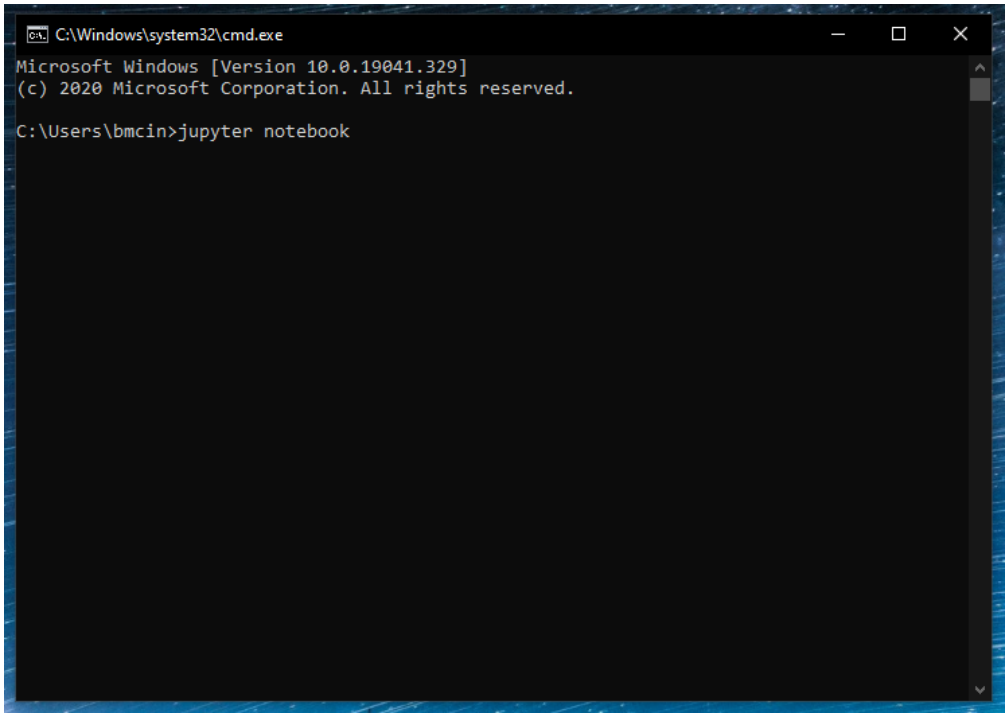
If all goes well you should be all updated. To close out of the terminal type `exit` and press enter.

Also let's check to make sure that Anaconda is in your path.

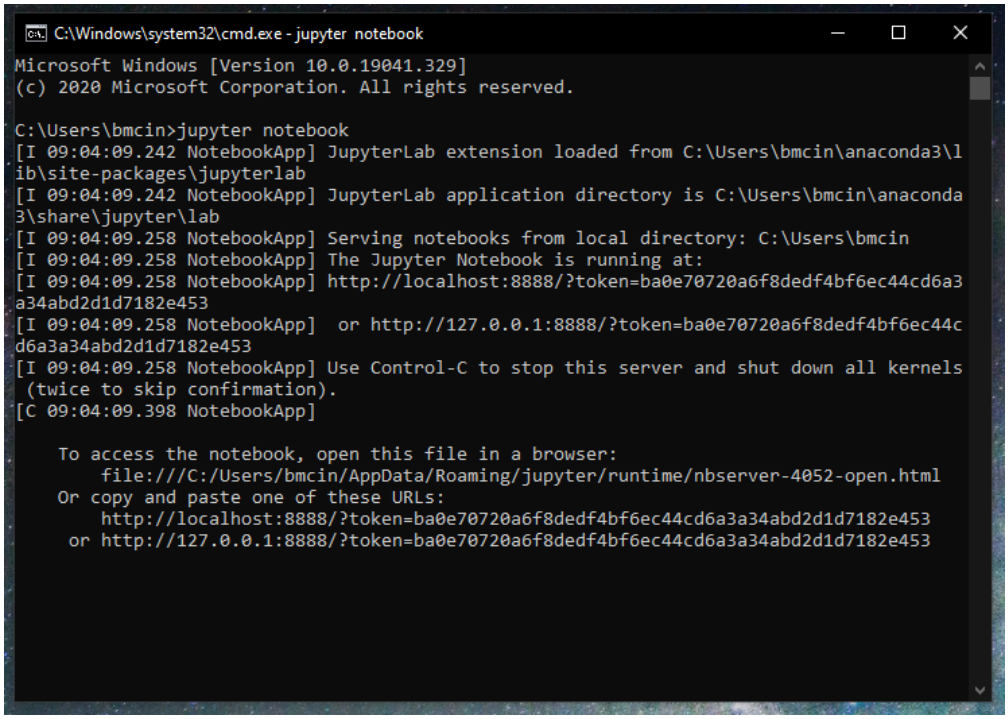
1. Have anaconda installed on system.
2. Open up any terminal besides anconda prompt and run Jupyter Notebook.

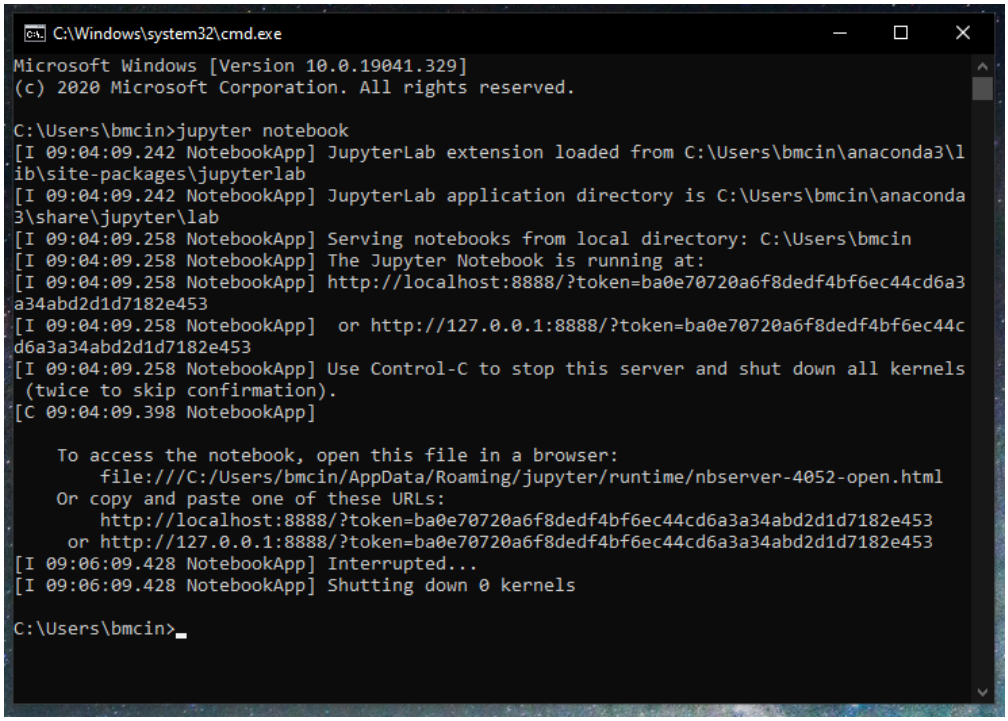


On keyboard press `Windows-key` + `r` or simply use the search bar on the taskbar if it is visible. Enter `cmd` and press enter. This will open up the Windows Command terminal.



Type in `jupyter notebook` and press `Enter`. This should launch a jupyter notebook tab on a web browser.

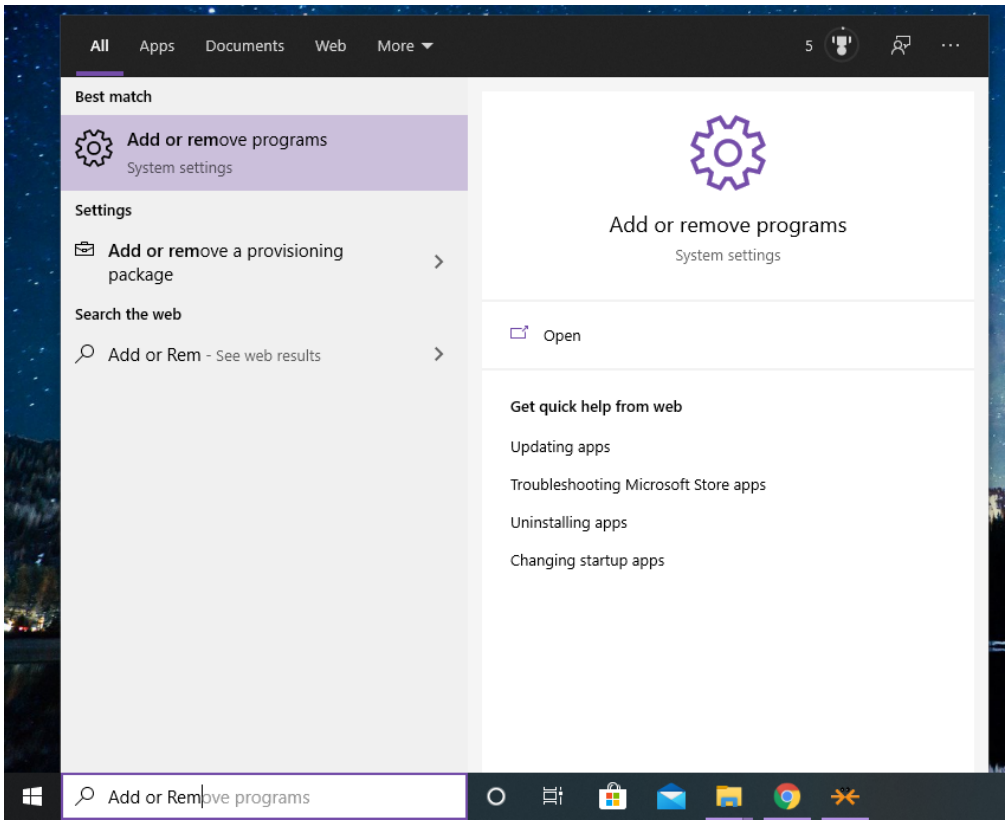




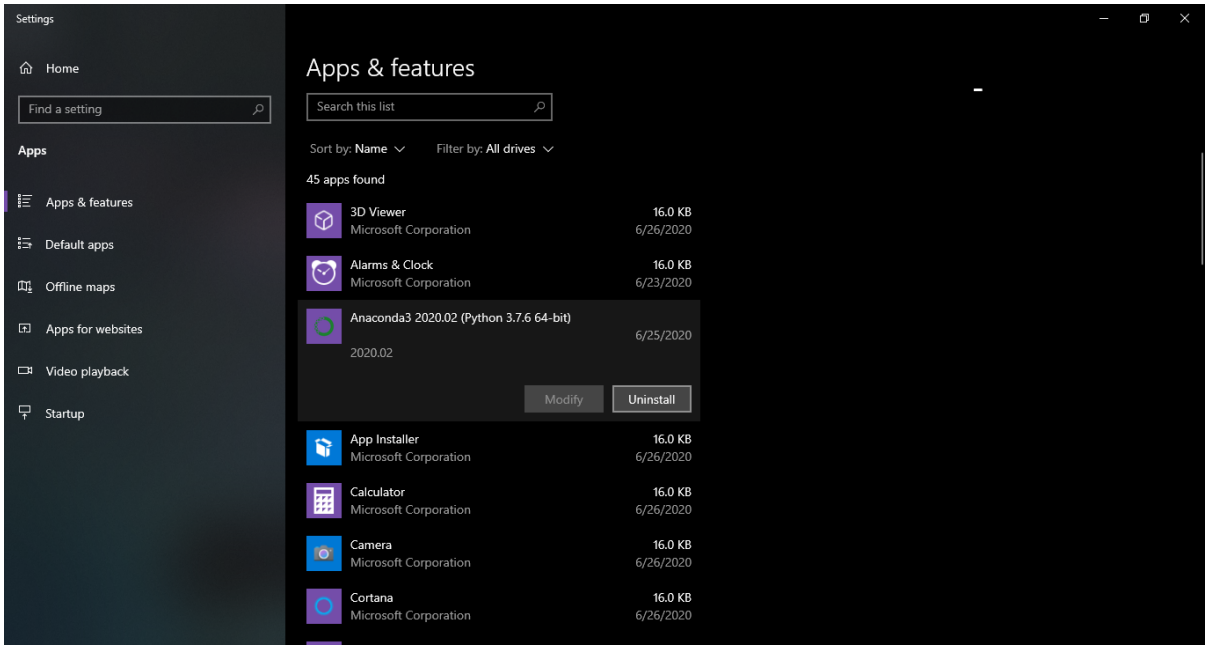
To exit jupyter notebook close the tab on the web browser, and go to the terminal window and type `Ctrl` + `c` twice in a row.

If any of the above fail, remove anaconda and reinstall. To remove Anaconda follow these steps.

1. On keyboard press `Windows-key` or simply use the search bar on the taskbar if it is visible.
2. Search `Add or remove programs` (Windows 8 & 10) or `Programs and Features` (Windows 7 or before) and click on the search result. This should take you to a list of programs installed on your machine.



3. Look for a listing that says "Anaconda" in the title and click on the listing.

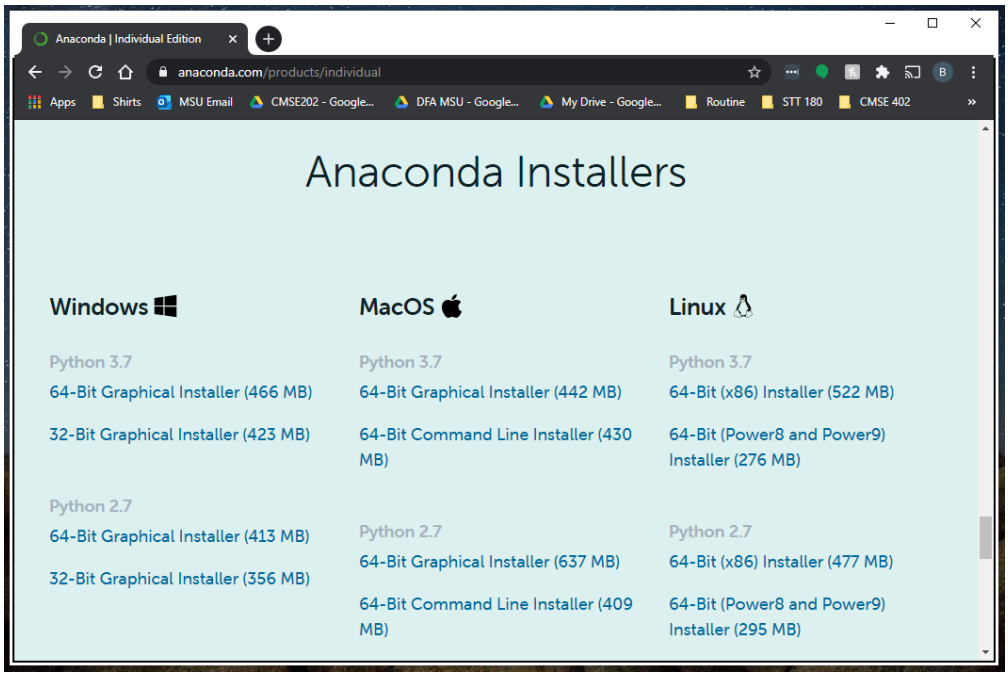


4. Press the button that says `Uninstall` and follow the prompts. The uninstall process should be straight forward.

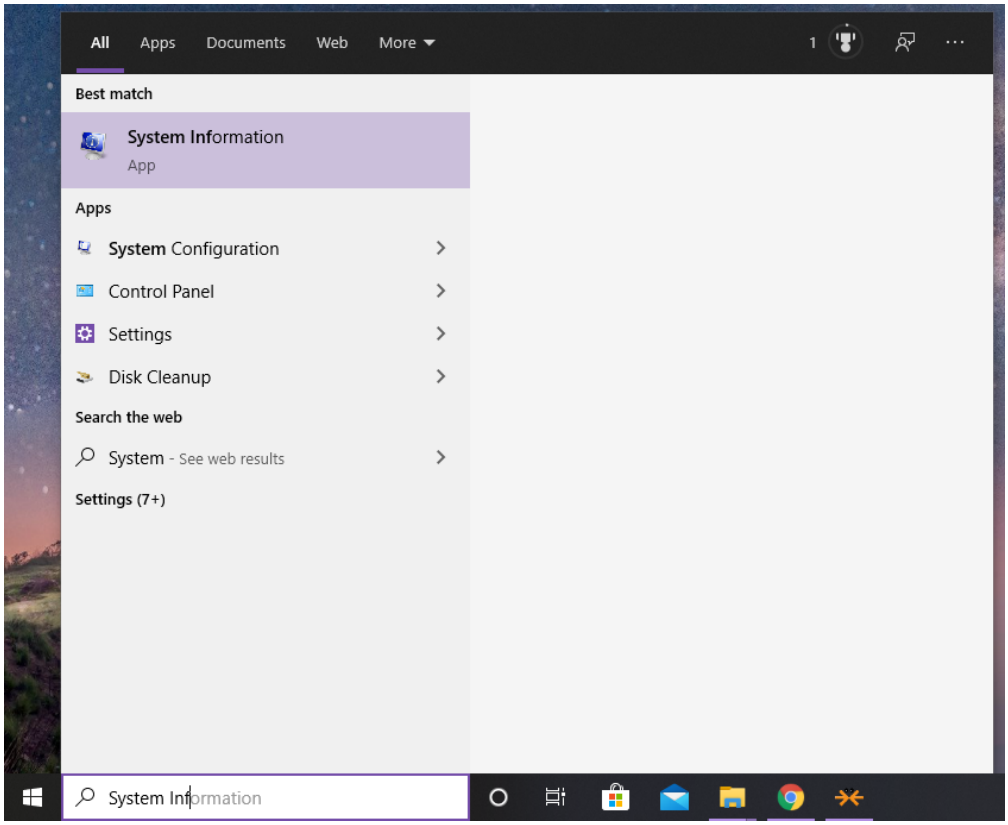
If you don't have a fully functioning up-to-date installation of Anaconda

Instructions for downloading Anaconda (Python 3.7.x):

1. Go to the [Anaconda Download webpage: (https://www.anaconda.com/download/)](https://www.anaconda.com/download/)
2. Use the `download` button under the `Your data science toolkit` (or just scroll until you see `Anaconda Installers`)

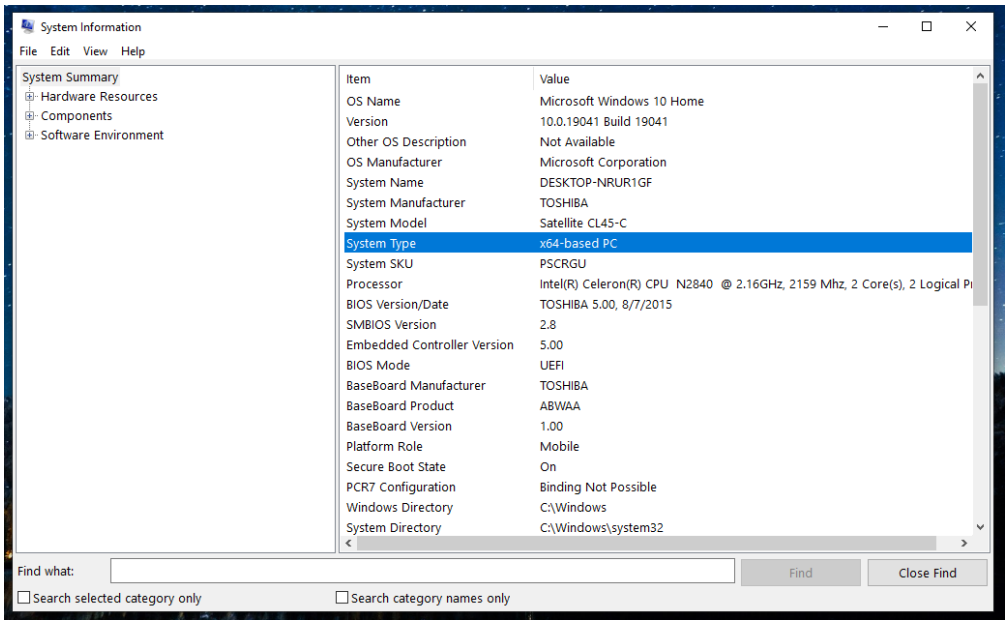


3. Download the Python 3.7 version, you'll notice there is a 32-bit and 64-bit version. If you are unsure which you should download, follow the instructions below.
   a. On keyboard press `Windows-key` or simply use the search bar on the taskbar if it is visible.
   b. Search `System Information` and click on the search result.

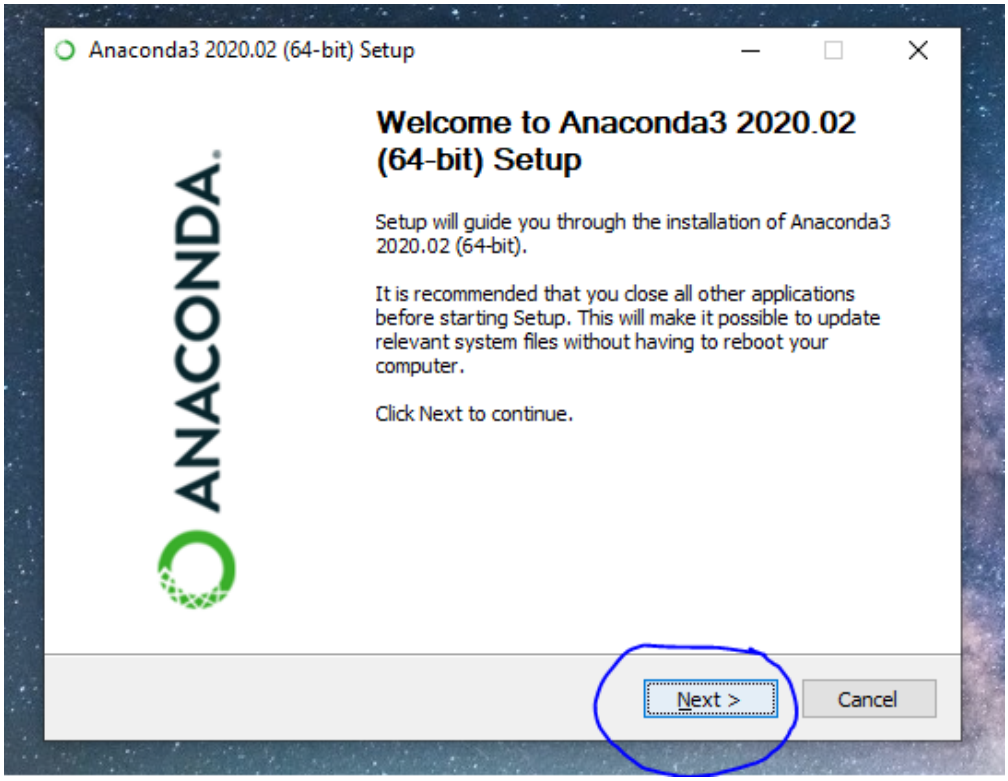


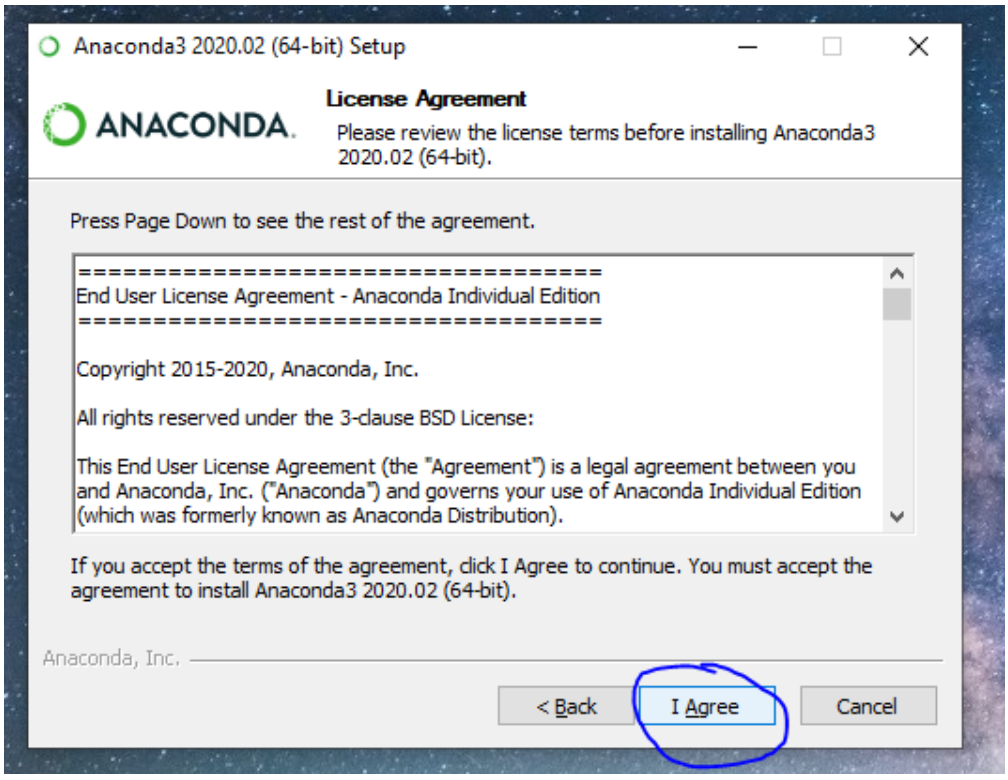
c. Look for the line called `System Type`

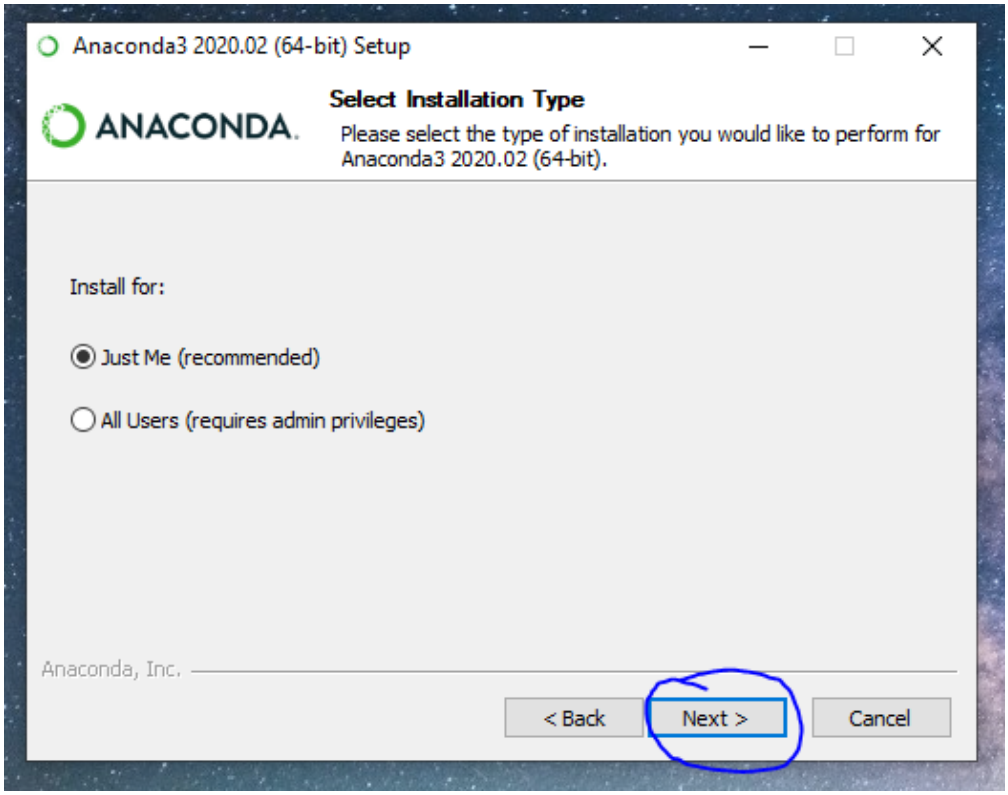
```
* If it reads `x64-based PC` you have a 64-bit system and you should download 64-bit Anaconda.
* If it reads `x86-based PC` you have a 32-bit system and you should download 32-bit Anaconda.
```



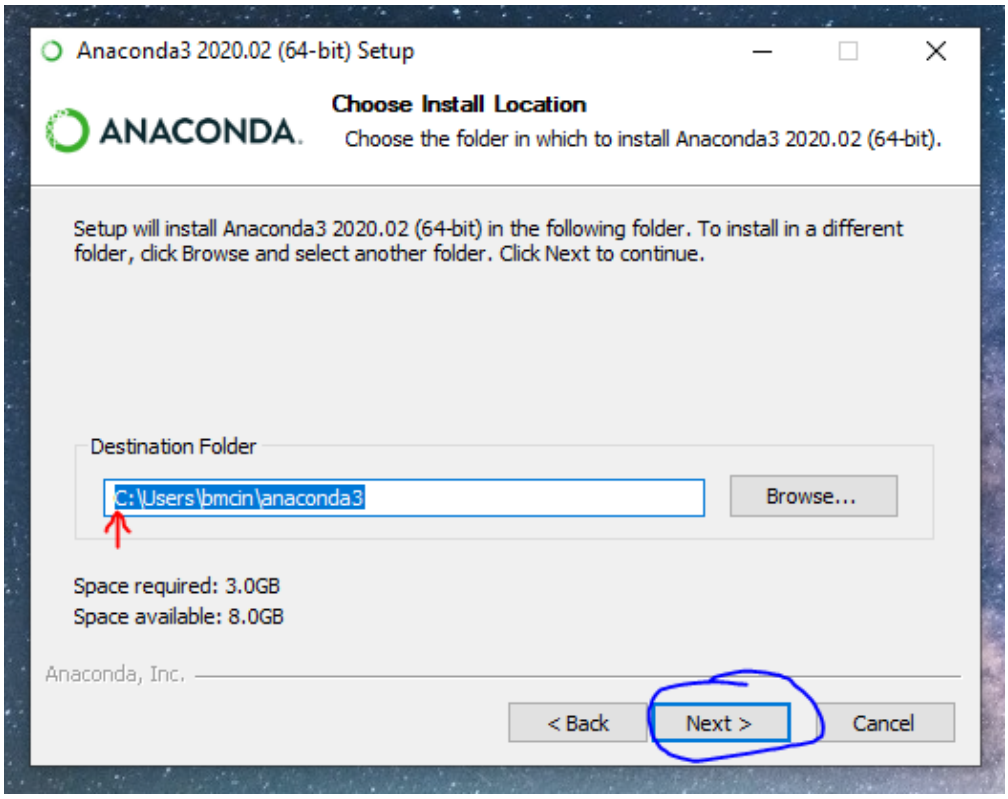
4. After downloading, run the Anaconda Installer Executable. Say `yes` to any warnings.


Anaconda3 2020.02 (64-bit) Setup
ANACONDA.
Welcome to Anaconda3 2020.02 (64-bit) Setup
Setup will guide you through the installation of Anaconda3 2020.02 (64-bit).
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

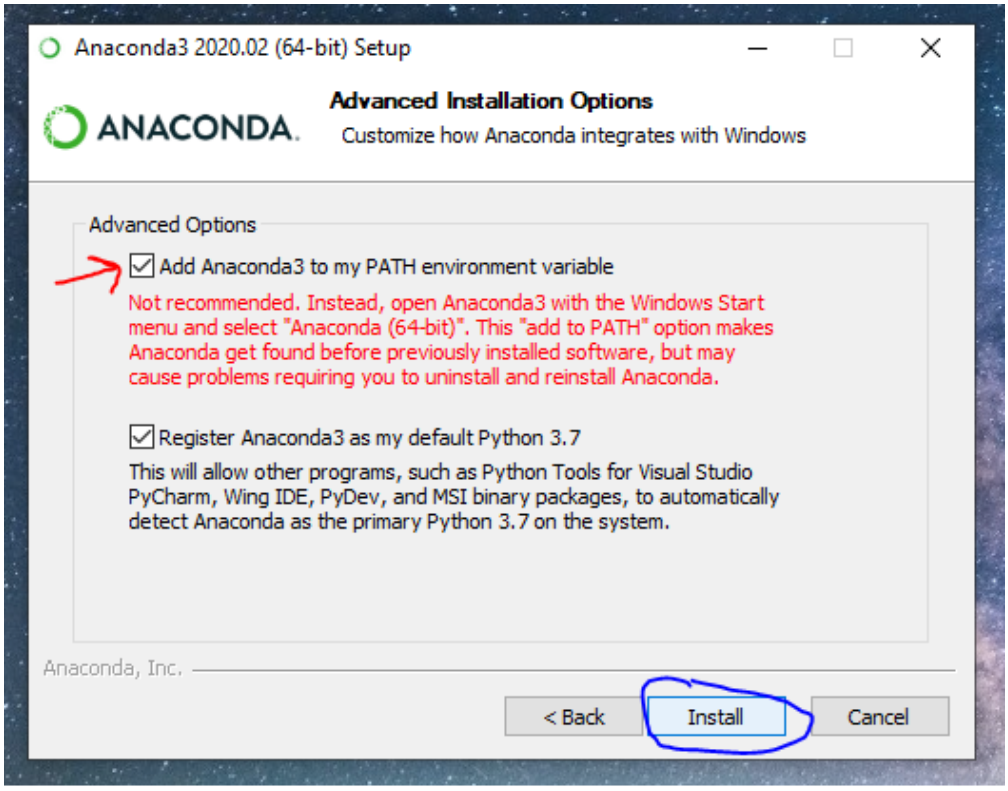

Anaconda3 2020.02 (64-bit) Setup
ANACONDA.
License Agreement
Please review the license terms before installing Anaconda3 2020.02 (64-bit).
Press Page Down to see the rest of the agreement.
End User License Agreement - Anaconda Individual Edition
Copyright 2015-2020, Anaconda, Inc.
All rights reserved under the 3-clause BSD License:
This End User License Agreement (the "Agreement") is a legal agreement between you and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Individual Edition (which was formerly known as Anaconda Distribution).
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2020.02 (64-bit).
Anaconda, Inc.
< Back
I Agree
Cancel



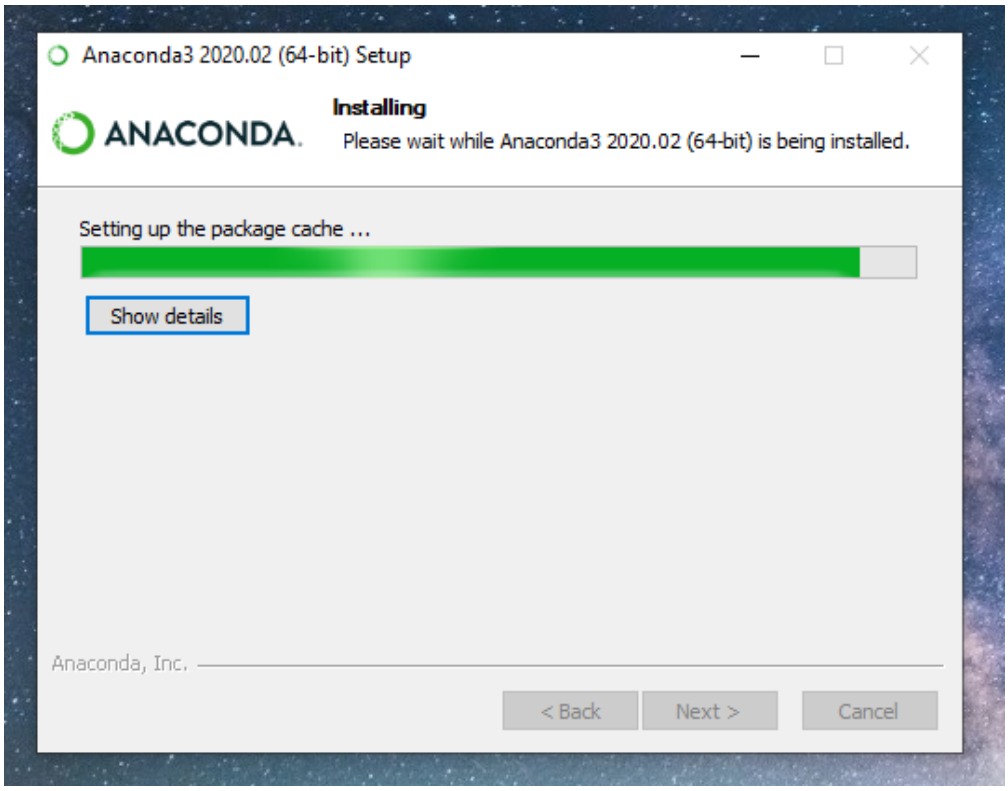
Any option here is ok, change to `All Users` if you want to install to all accounts on your PC.



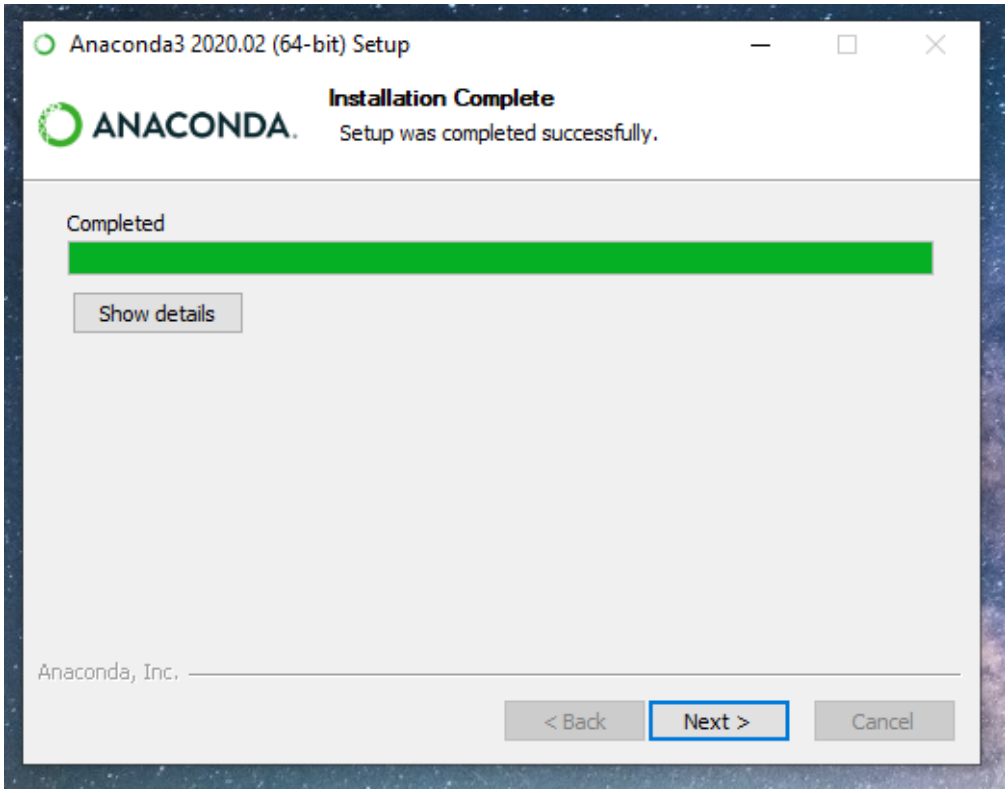
Change the Destination folder at your own risk If troubles creep up later in class with using Anaconda, this might make the issues harder to fix. If you do change location, make sure it remains on the drive your windows installation is on.

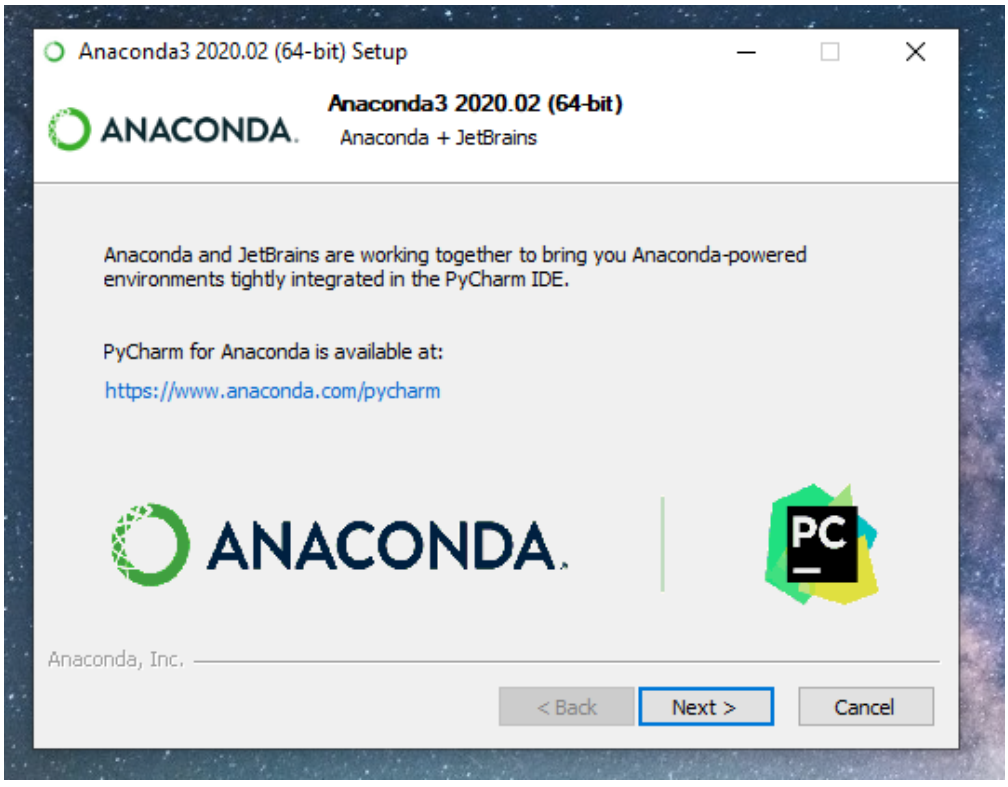


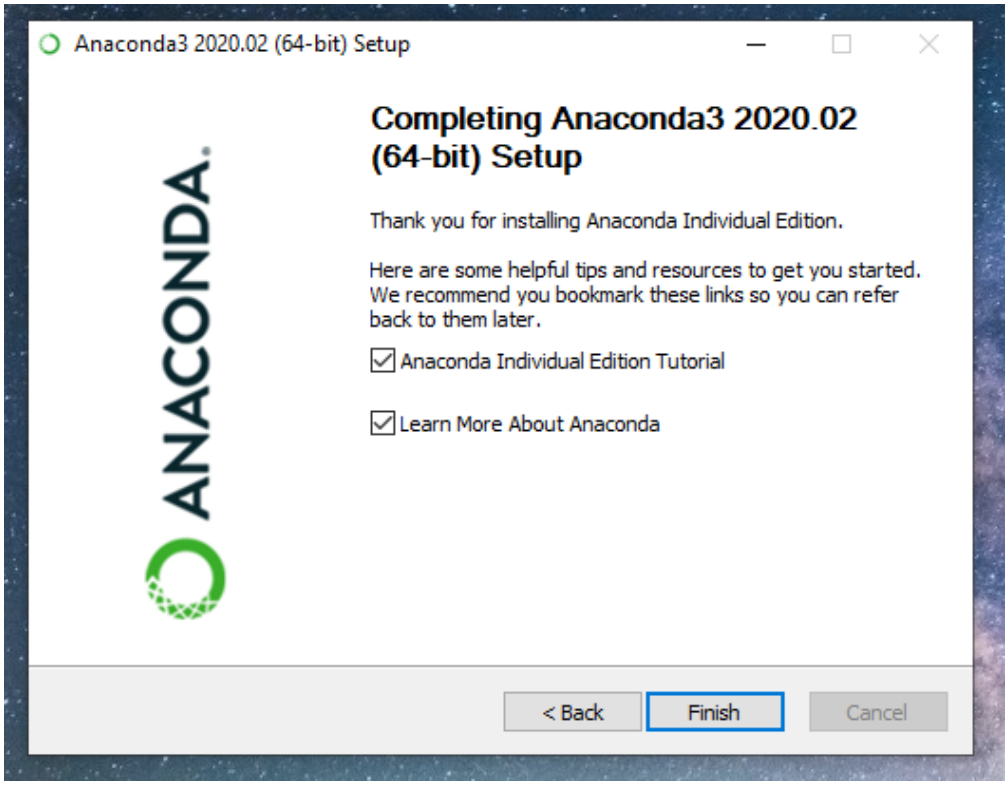
Make sure to enable this option This is required for software this class uses.



Installation may take awhile, it may stay at this screen for awhile. Be patient.

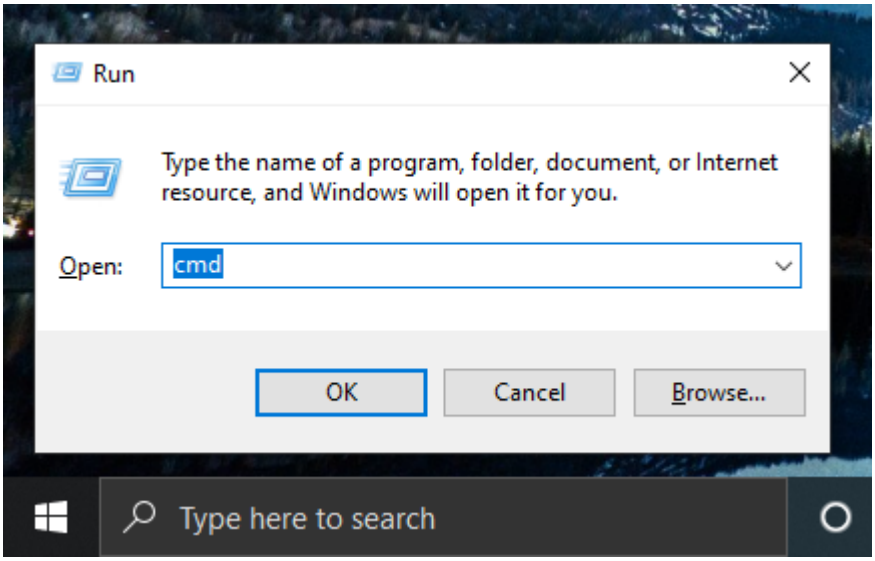

Anaconda3 2020.02 (64-bit) Setup
ANACONDA.
Installation Complete
Setup was completed successfully.
Completed
Show details
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 2020.02 (64-bit) Setup
ANACONDA.
Anaconda3 2020.02 (64-bit)
Anaconda + JetBrains
Anaconda and JetBrains are working together to bring you Anaconda-powered environments tightly integrated in the PyCharm IDE.
PyCharm for Anaconda is available at:
https://www.anaconda.com/pycharm
ANACONDA.
PC
Anaconda, Inc.
< Back
Next >
Cancel

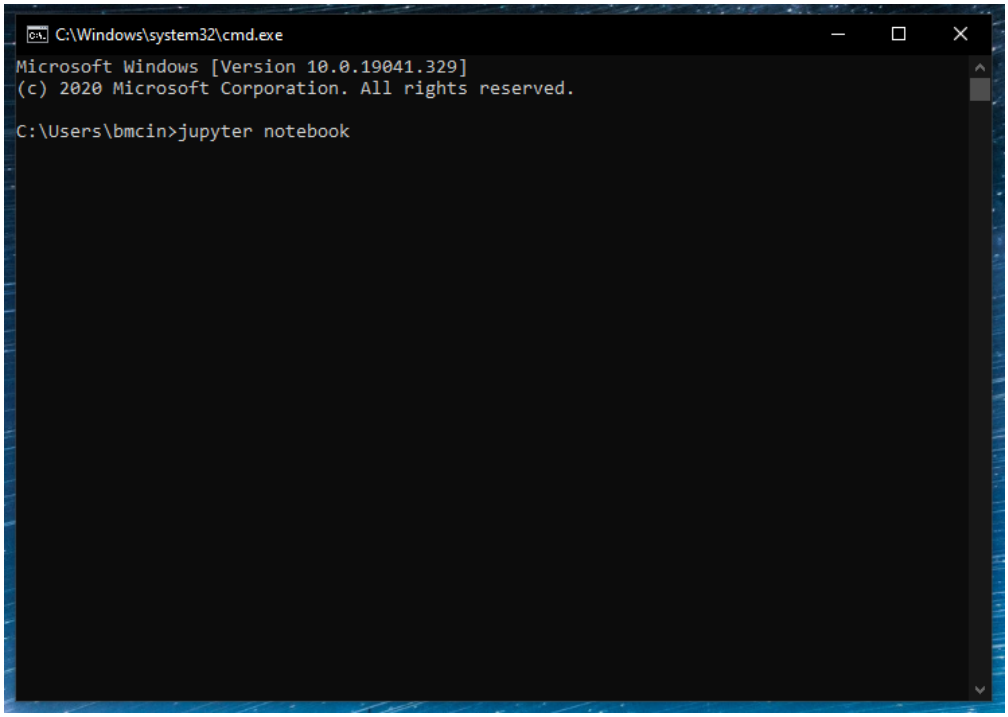


Any option here is okay, if you wanna get a feel for the things Anaconda can do, feel free to keep those checkboxes selected.

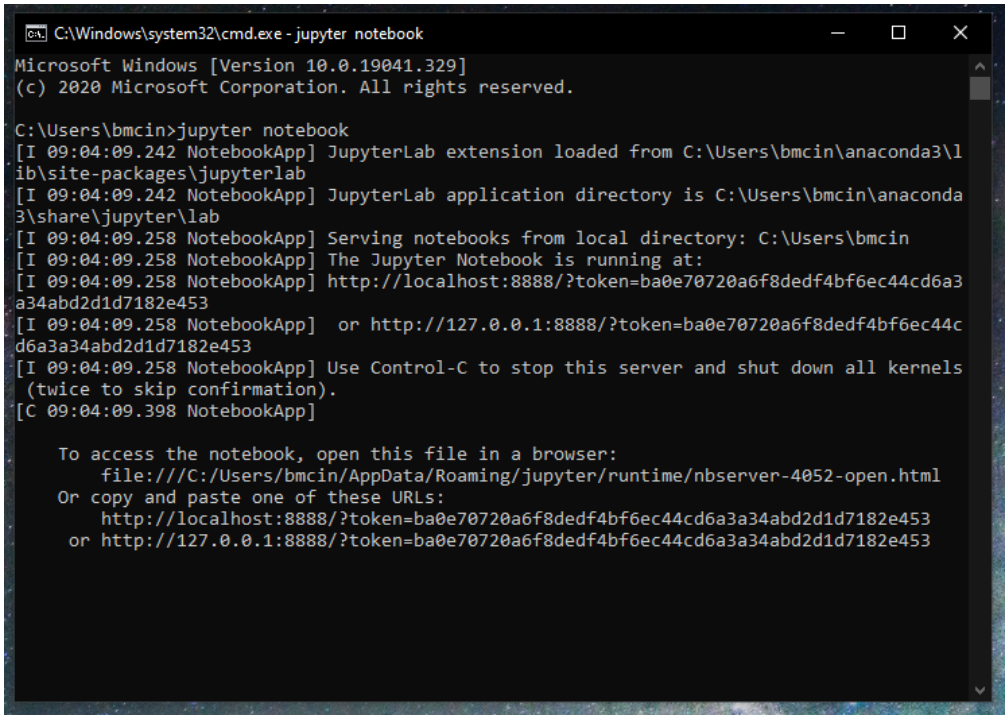
5. Open the command line program on your computer.
   a. On keyboard press `Windows-key` + `r` or simply use the search bar on the taskbar if it is visible.
   b. Enter `cmd` and press enter.



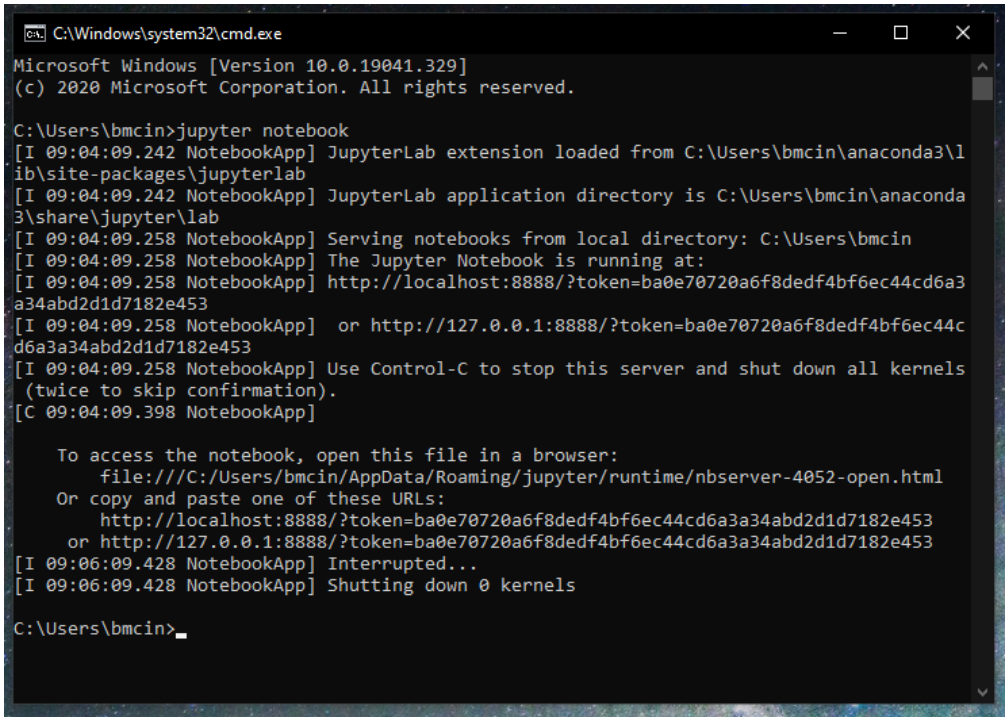
6. Type `jupyter notebook` in the command line and hit enter.



If everything goes correctly, a browser window should open up with the Jupyter interface running. If things don't work, don't worry, we will help you get started.



To exit jupyter notebook close the tab on the web browser, and go to the cmd window and type `Ctrl` + `c` twice in a row.



To close out of the terminal type `exit` and press enter.

7. *If your anaconda installation was successful* follow the instructions above under `Installing Python for this Course -> If you have Anaconda on your computer already -> Let's make sure it is updated to the latest version` to update Anaconda and all of its packages (or follow this link).

Setting up a terminal

In this course, we will learn how to perform python and data analytics outside of the Anaconda enviroment. We will also learn new tools, such as as git (a version control utility that we will learn more about later), and other useful tools powered by the command line. Using a terminal will open our abilites of what we can do, and is a great way to start opening up new avenues in computing.

We will be utilizing terminals that run what is known as *Unix-like* command line interpreter, as this will be the most compatible between all different devices (Windows, MACs, and Linux-based PCs). Different command line interpreters use different syntax on commmands, we choose Unix-like as both MAC and Linux use these interpreters naturally. On Windows, the base command line interpreter is known as MS-DOS. While it is possible to run the same commands, the syntax for MS-DOS is quite different and it will just be much easier if we use a Unix-like command line interpreter instead.

If you have Windows 10

(If you do not have Windows 10 click here)

Windows 10 has a special ability that no other windows operating system has. This is the ability to run a linux operating system natively (meaning it is not just emulated, but actually runs as Windows 10 runs on your computer). This is through a feature called Windows Subsystem for Linux, or *WSL for short*.

In this tutorial, we will be installing Ubuntu on our WSL, and be using this as our terminal and how we run Anaconda and Jupyter Notebook.

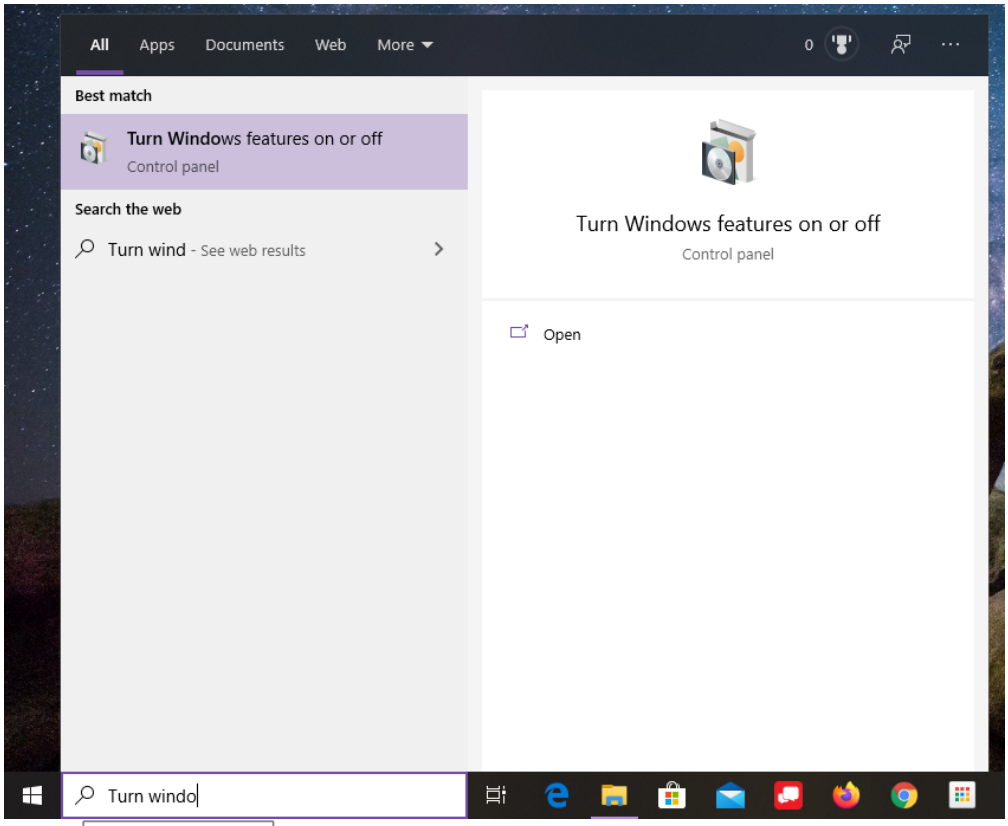
While you can install Anaconda on WSL directly, this has a lot of set-backs and techinal difficulties. For our purposes we will be setting up WSL to use the Anaconda that runs on Windows 10 instead.

Lets get started!

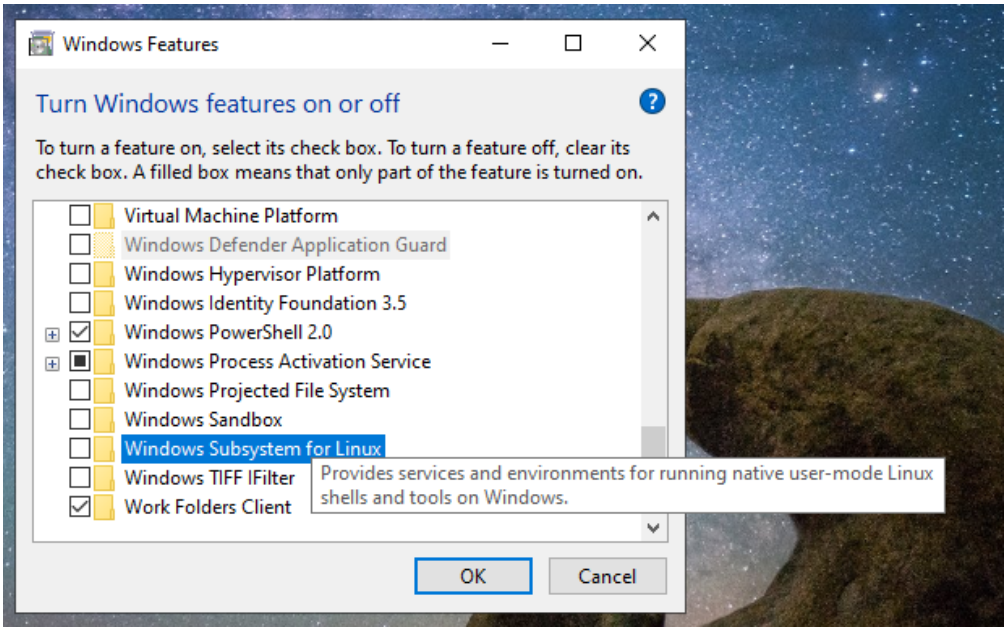
Probably wouldn't be too bad of an idea to make sure your computer is updated before we start.

Install WSL via Windows features.

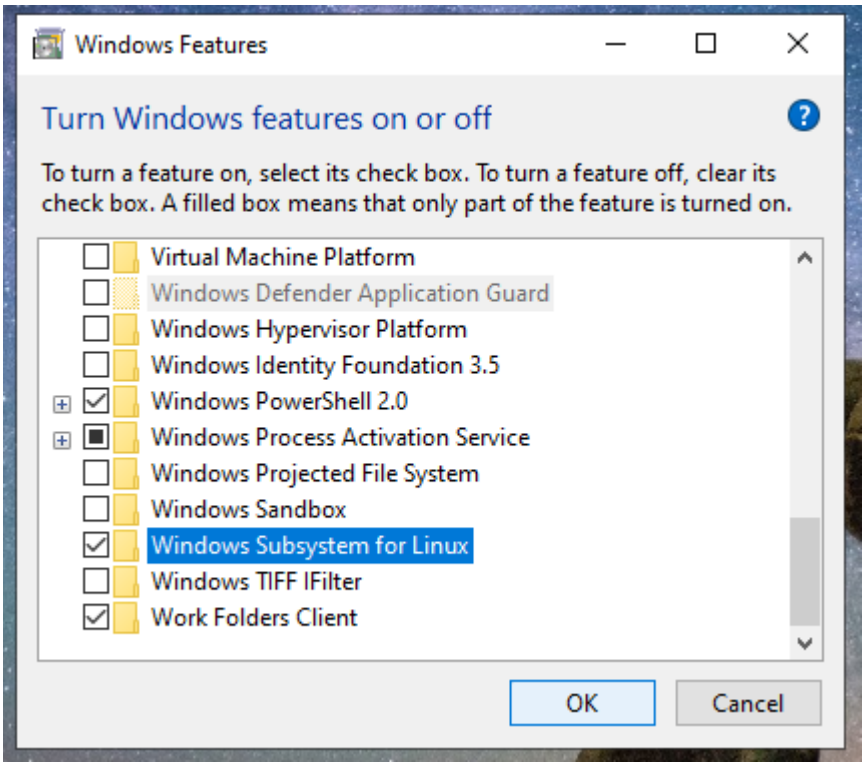
1. Update Windows and have a connection to the internet.
2. Navigate to `Turn Windows feature on or off` and enable `Windows Subsystem for Linux`



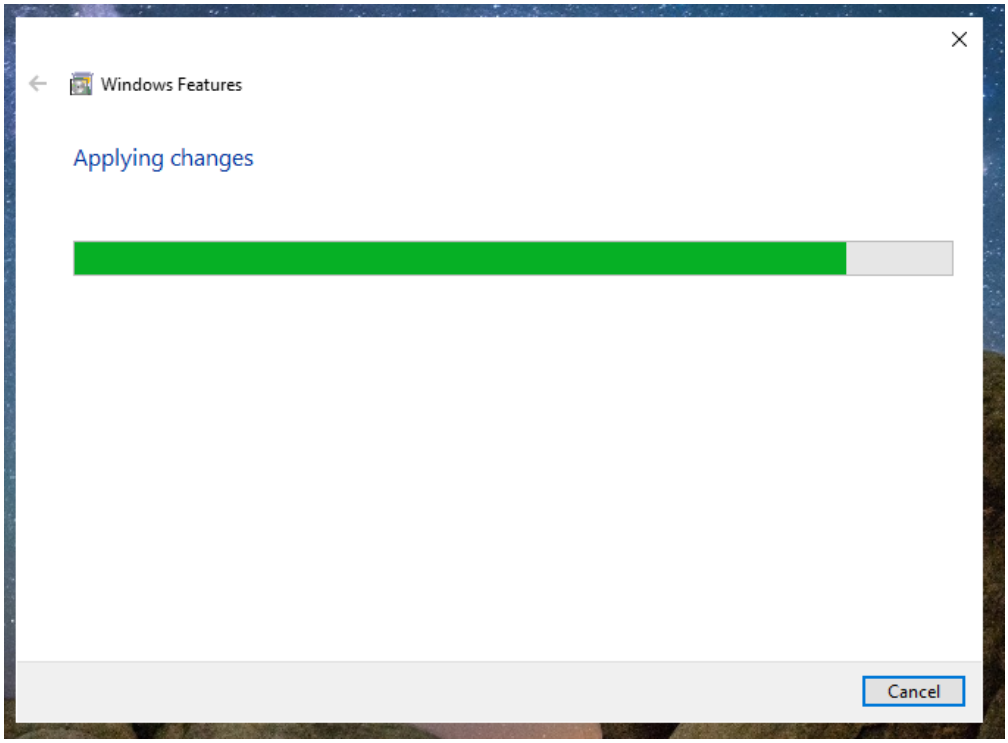
On keyboard press `Windows-key` or simply use the search bar on the taskbar if it is visible. Search `Turn Windows features on or off` and click on the result.

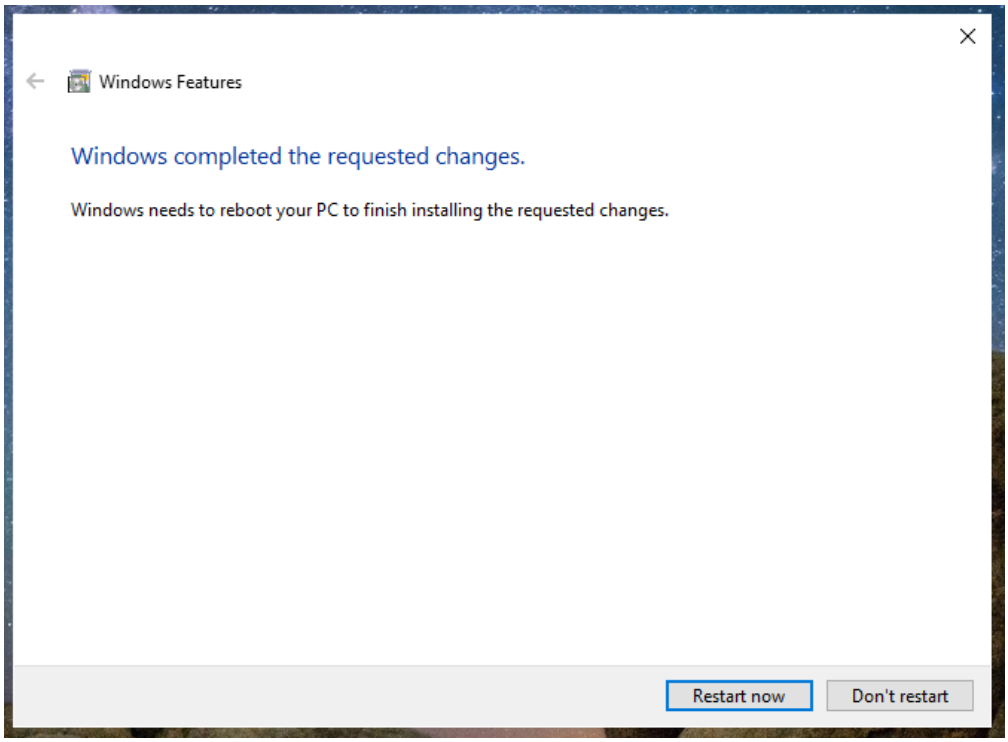


Scroll down until you find the option for `Windows Subsystem for Linux`



Click on the check box next to the feature. Then press the `OK`



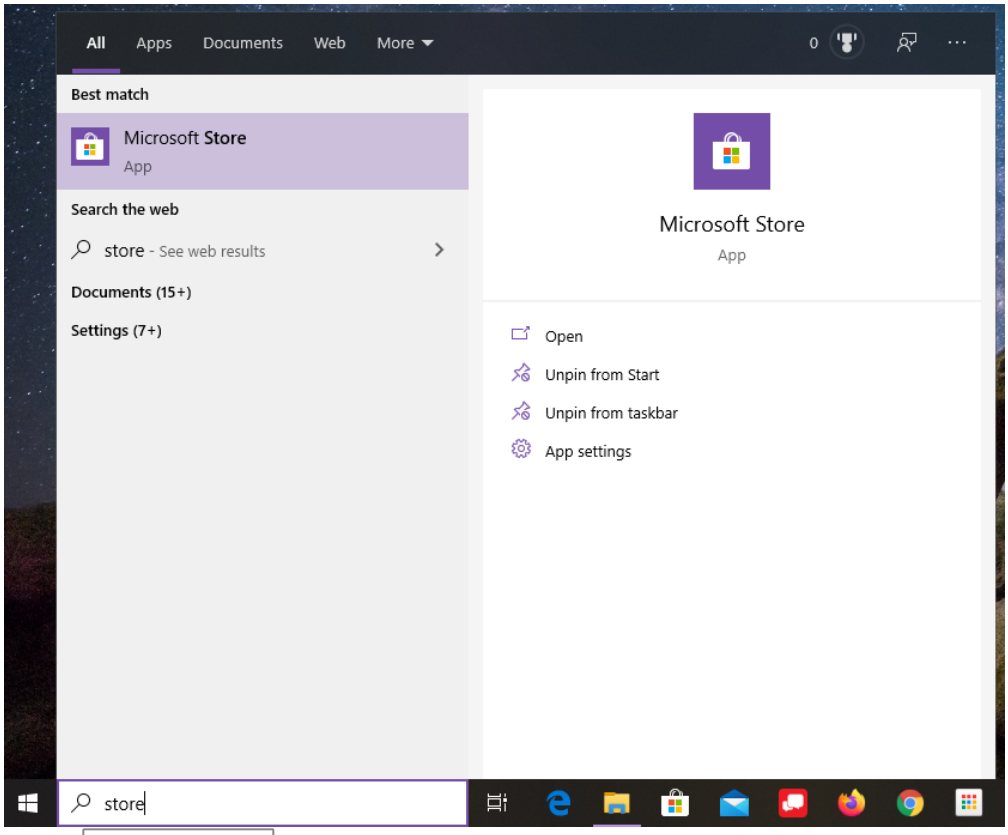


Make sure all files are saved, and your computer is ready to restart. Then, click the `Restart Now` .

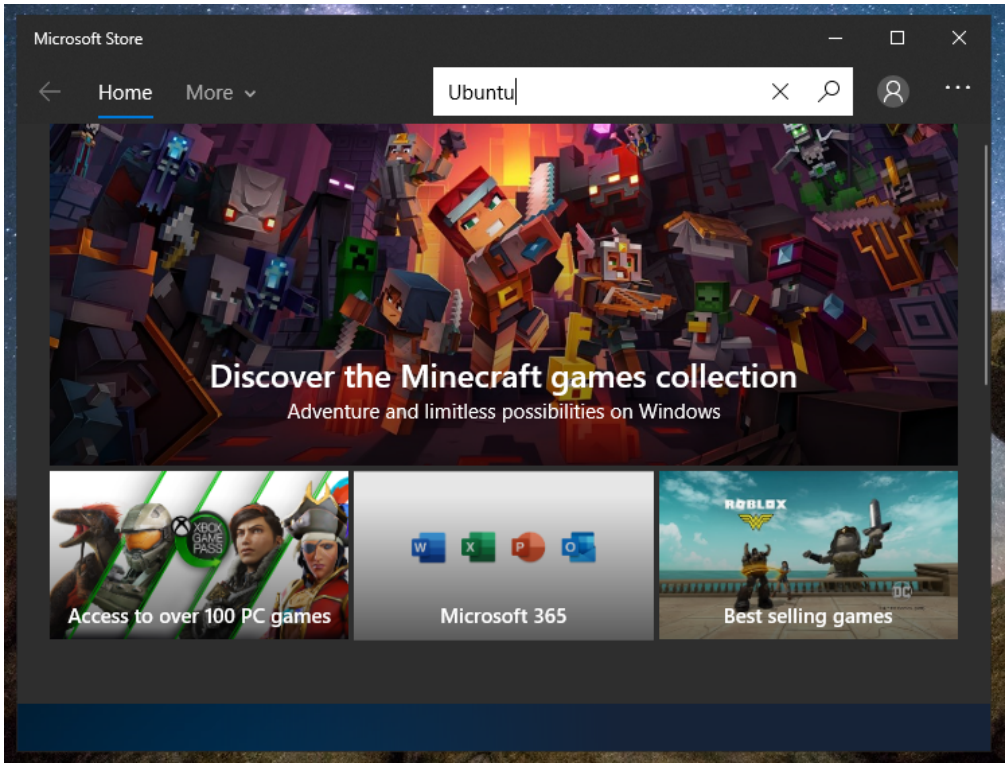
*It is possible that this did not work for you if you have an older version of Windows 10, if this is the case you will need to either update Windows or enable developer mode in the `Developer settings` . The settings can found by pressing the `Windows-key` and searching for `Developer settings` .

Install Ubuntu onto our WSL

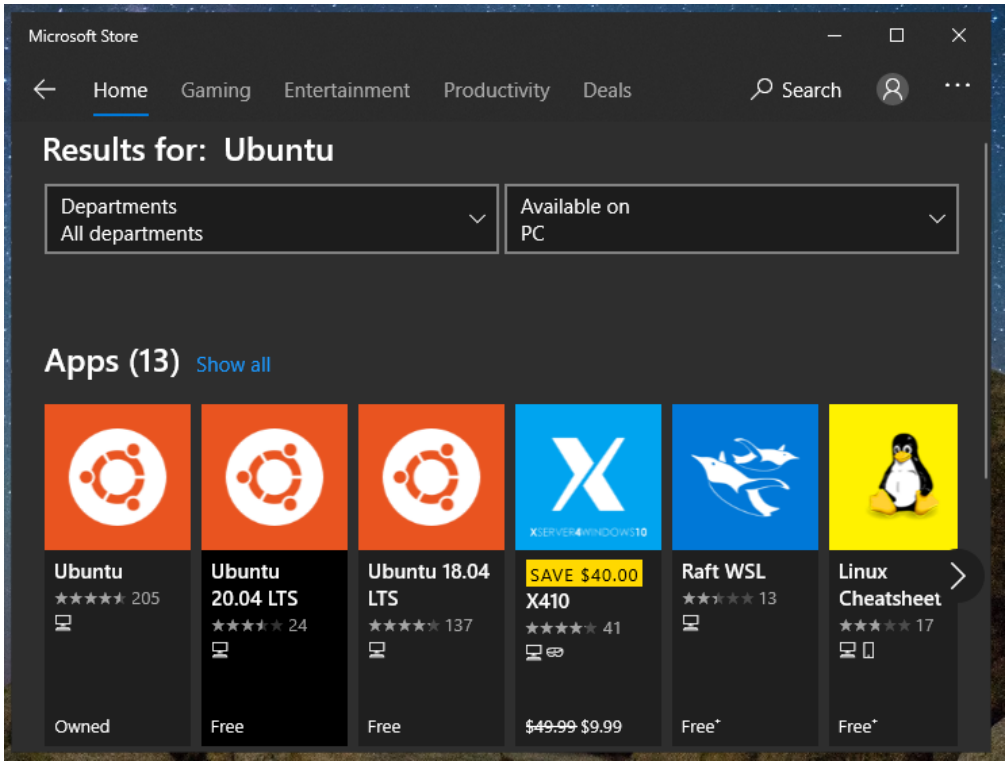
1. Have a connection to the internet
2. Naviate to the Windows Store and download `Ubuntu 20.04 LTS` . Then start up the program.



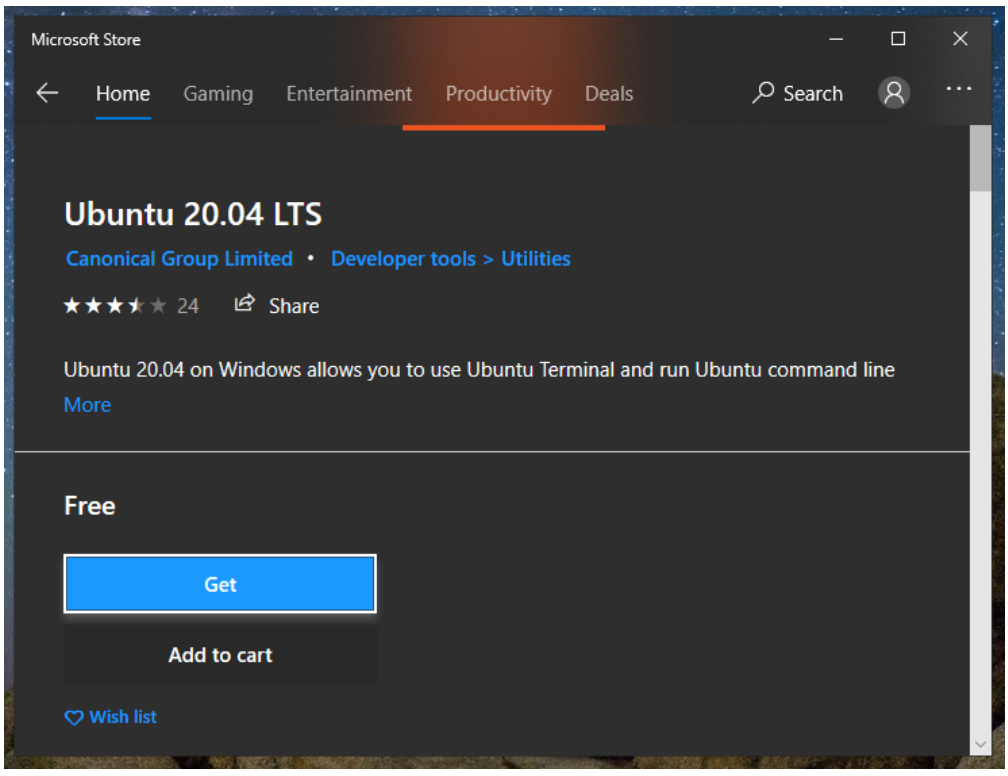
On keyboard press `Windows-key` or simply use the search bar on the taskbar if it is visible. Search `Microsoft Store` and click on the result.



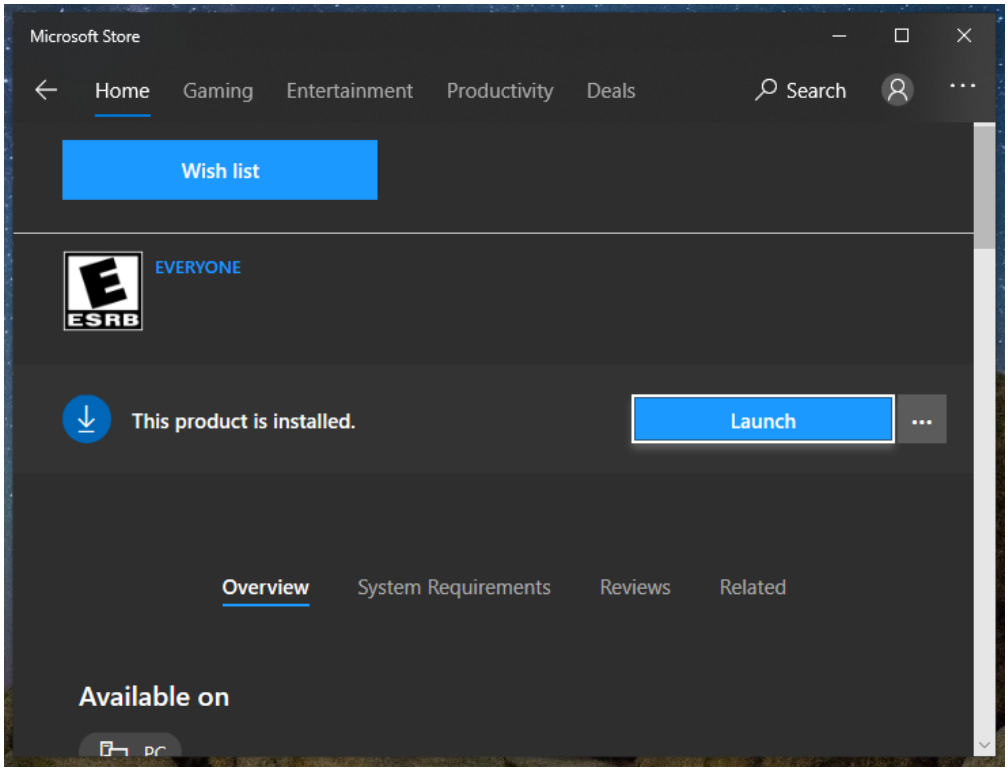
Using the search in the top right corner. Search for `Ubuntu` .



Click the "App" entitled `Ubuntu 20.04 LTS` . We will choose this one because it is the most updated and also a "Long Time Support" linux version.



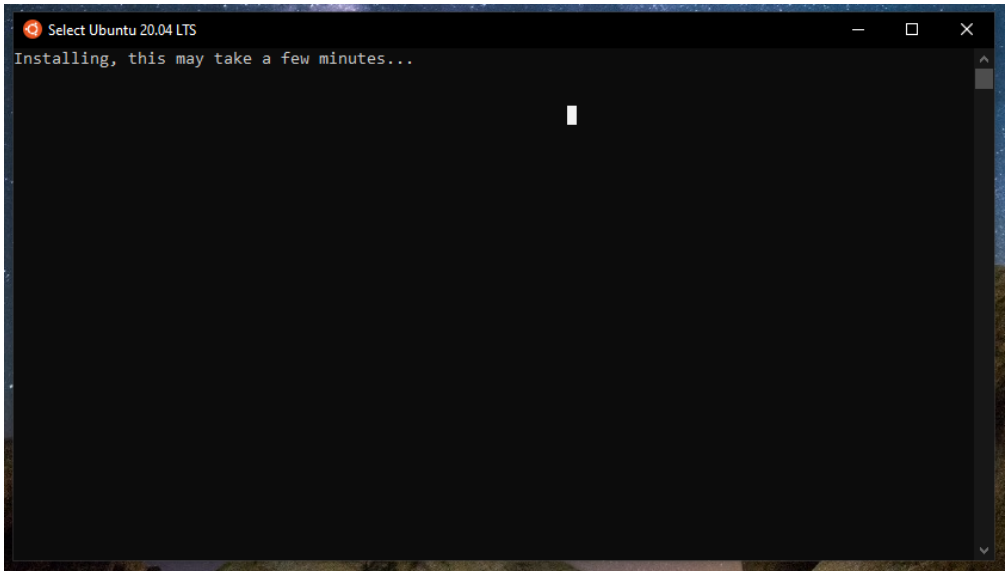
Press the `Get` button to download Ubuntu.



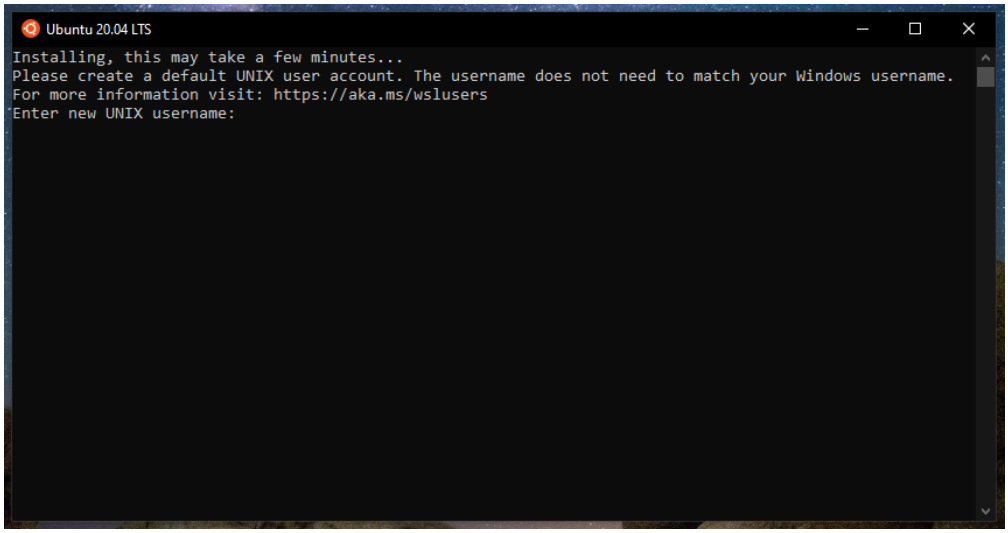
Wait until the package is installed fully. Then you can press `Launch`, however, please remember for the future that you can run Ubuntu in the future by searching `Ubuntu` using the search bar in the taskbar or `Windows-key`.

Setup Ubuntu on WSL

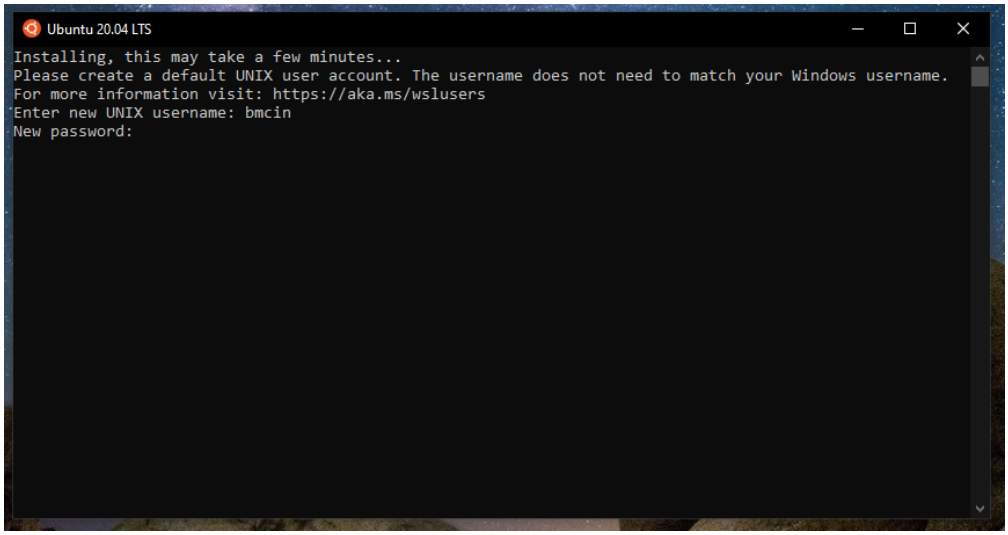
1. Set up Ubuntu with username and password.



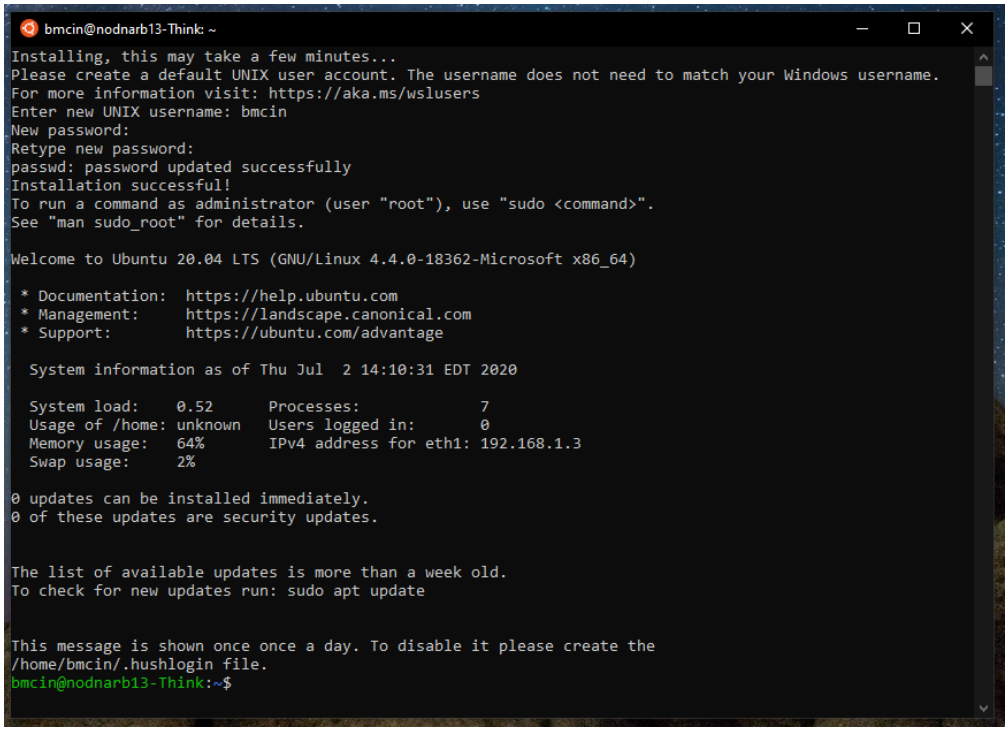
After opening Ubuntu, you should see this prompt. From this point forward in the terminal, `Ubuntu` will be reffered to as "the terminal". It may be a good idea now to "pin" the terminal to your taskbar for easy opening. You can do this by *right-clicking* the Ubuntu symbol in the task bar and selecting `Pin to taskbar`.



You may put any username you want, just **make sure there are no spaces in the name**. Once you have entered your desired username, press `Enter`.



Enter a password that you will remember. It may be a good idea to write it down somewhere as the option to "reset the password" may not be easily done. *Note: When typing in your password, you will not see anything be written in the terminal, this is a traditional saftey feature of unix-like systems (aka Ubuntu). Also, the* `Backspace` *still does work, so don't be afraid to use it if you fear you have made a mistake.* Once you have created your password (and you are sure you entered it in correctly), press `Enter`. You will prompted again to `Retype new password`. Re-enter your password again and press `Enter`.

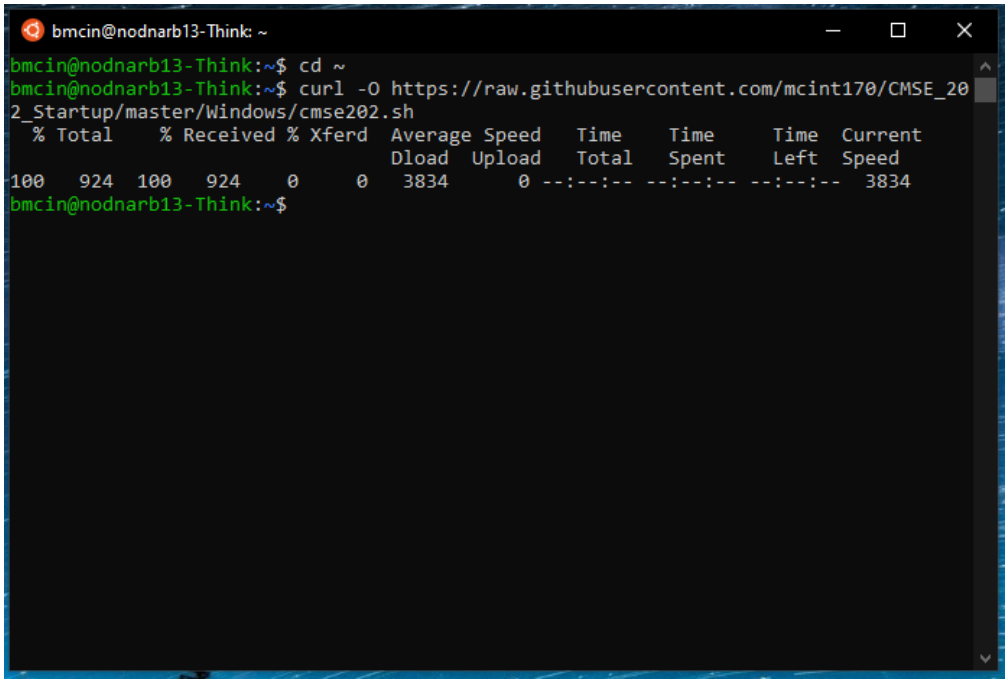


If everything went smoothly, you should be greeted with the ability to run commands. This looks like `[USERNAME]@[COMPUTER_NAME]:[LOCATION]$` . Note in the photo this looks like `bmcin@nonarb13-Think:~$` . `bmcin` being the username, `nonarb13-Think` being the computer's name, and `~` being the location in the computer we are at. In our case we should be what is known as the `home` directory, aka `~` . We will later cover the basics of where you are in your computer and the signifiance of your location.

Set-up Windows Anaconda on WSL

Though there is an Anaconda you can run directly on WSL, the support and ease is not quite there. WSL is an evolving project and someday may better support Anaconda. In this course we wil be using the Anaconda that is installed on Windows to do all of our work. In order to accomplish this we will need to set up a few shortcuts in our `.bashrc` file. These shortcuts are called *aliases*. However, fortunately we have a script (`.sh` is the file ending) that will make those edits for you. *Note: IF YOU HAVE ANACONDA INSTALLED ON UBUNTU (not shown in this tutorial), these shortcuts will not work as these alias names are already in use and will not work. You can, however, look through the script file using an editor and edit the alaises to something you will remember*

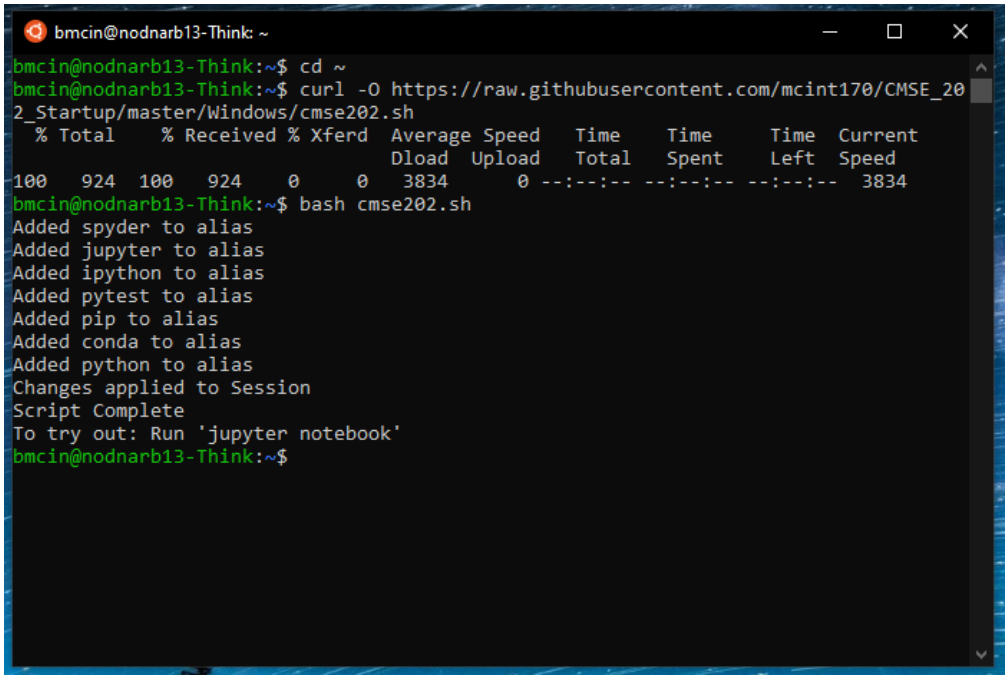
1. Have a connection to the internet.
2. Open up the terminal and download the bash script from here https://raw.githubusercontent.com/mcint170/CMSE_202_Startup/master/Windows/cmse202.sh (https://raw.githubusercontent.com/mcint170/CMSE_202_Startup/master/Windows/cmse202.sh)



Open up the Ubuntu terminal, if it is not already open, by searching "Ubuntu" in bottom left-hand corner search bar. Then, use the command `cd ~` to make sure you are in your `home` directory. Then, in the terminal copy and paste this command and press `Enter`. *Note: To copy and paste select the text by clicking and dragging over it and press* `Ctrl` + `c`. *Then, in the terminal, left-click to engage the window and then right-click to paste the content in the terminal. If this fails you can always copy the text below by hand*

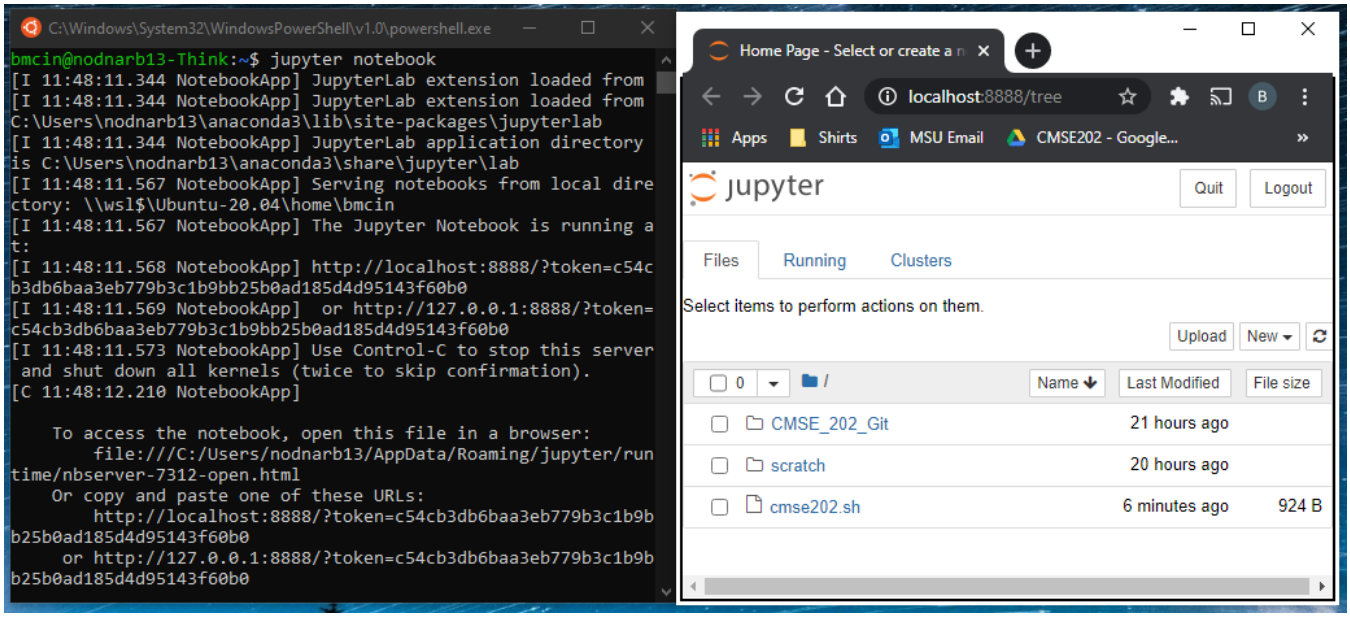
```
curl -O https://raw.githubusercontent.com/mcint170/CMSE_202_Startup/master/Windows/cmse202.sh
```

3. Execute the `cmse202.sh` script.

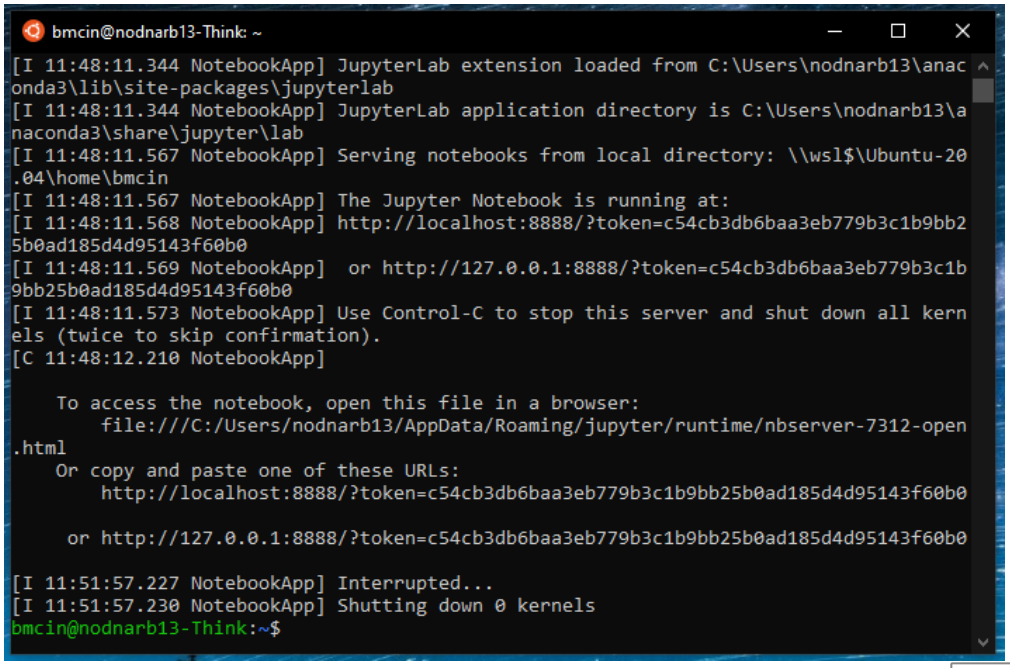


In the terminal type `bash cmse202.sh` and press `Enter`. If all goes well it should like the image above.

4. Try running Jupyter Notebook.



In the terminal type `jupyter notebook` and press `Enter`. If all goes well jupyter notebook should open up.



To close Jupyter Notebook, close the browser and then left-click in the terminal and press `Ctrl` + `c` on the keyboard. Jupyter notebooks should then stop in the terminal.

Congrats you are done with setting up a terminal! Your terminal will now run very similar to both MAC and Git-Bash users and you have everything essential to start class.

If at anypoint anything got too confusing or you were not successful, please let your instructor know during class or through email and they can work with you to resolve any issues.

If you want to learn some optional tools and shortcuts continue to the next section. However, if you want to skip the optional section, click here.

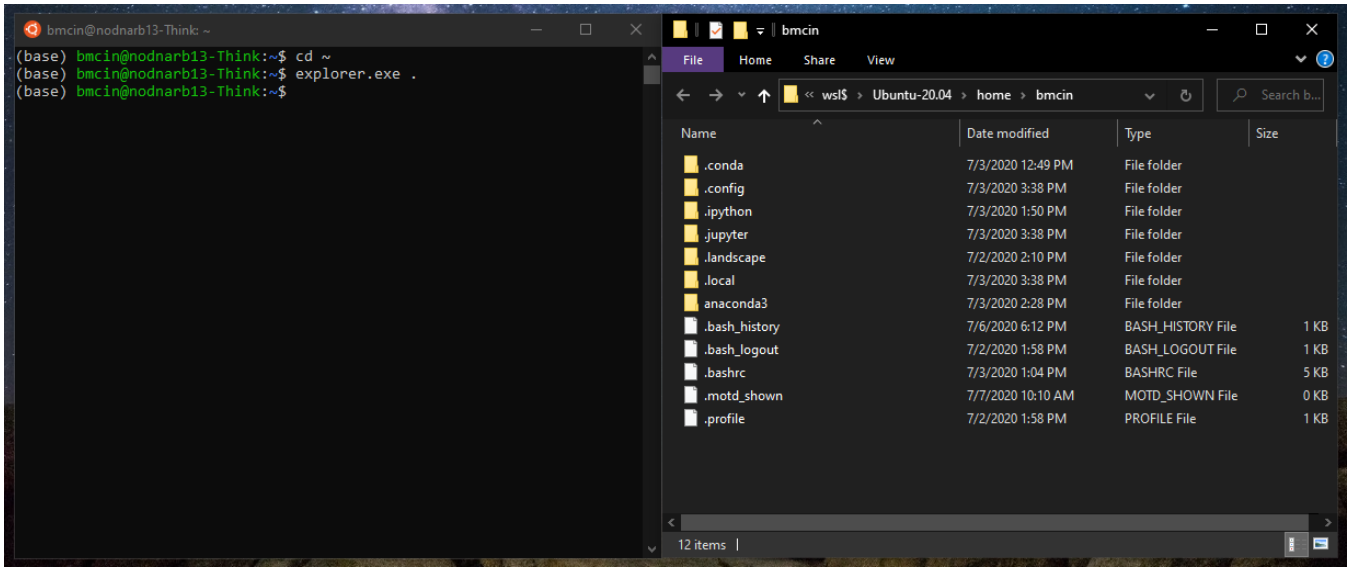
(Optional) Helpful Commands and Shortcuts

The command line is a powerful, but also complex tool. The following are some commands and shortcuts that will make using the command line on WSL easier. This is optional, but this will serve as a guide to make common actions you may need to take using WSL in this course easier.

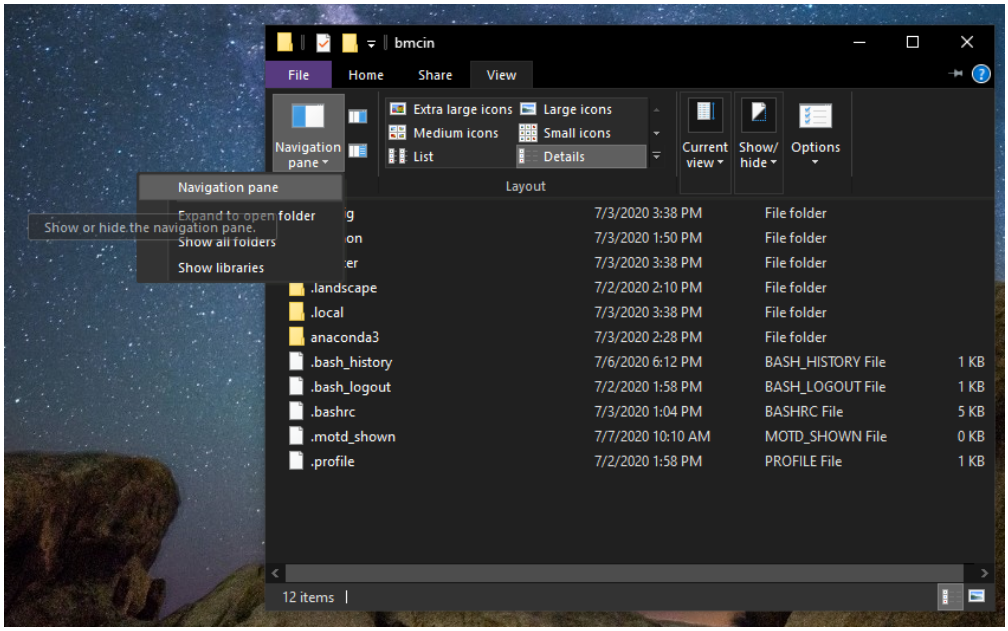
WSL Home Directory Shortcut

Throughout this course you will be downloading many notebooks and files. When it comes to the WSL, the directory you start your session in, known as the "home directory", is not easy to get to in the file explorer from Windows 10. However, thankfully common windows tools are actually baked in to WSL. We will utilize this feature to create a shortcut to the WSL home directory in your file explorer.

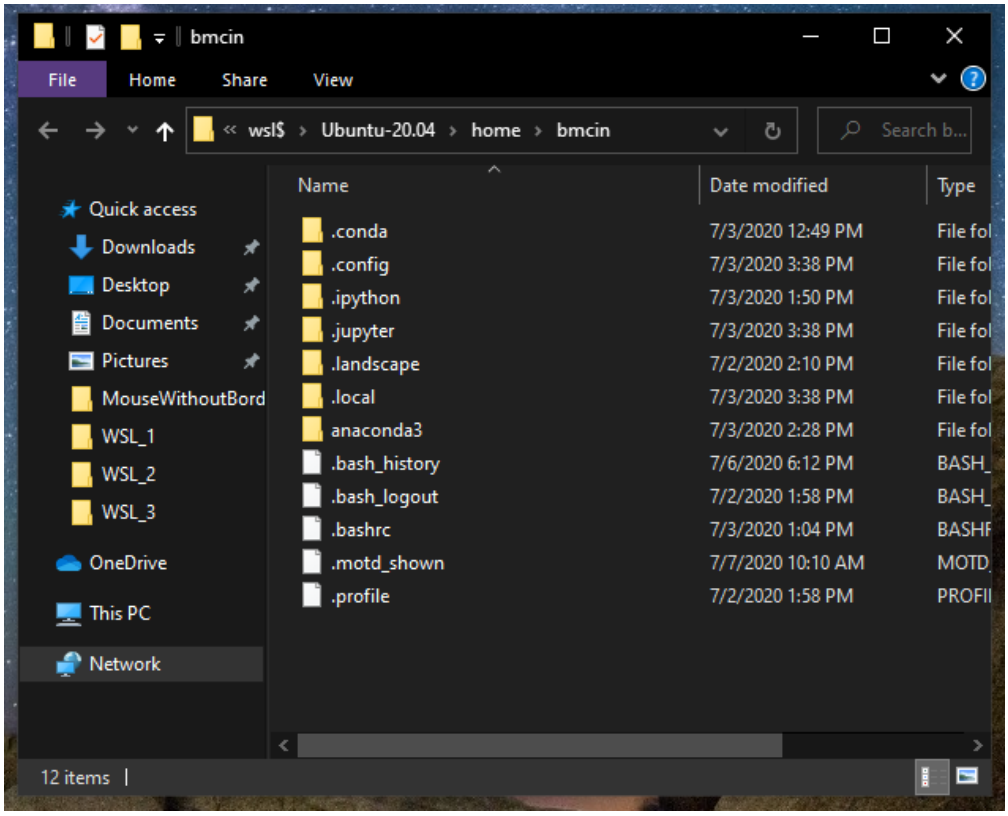
1. Open up the Ubuntu terminal, and type `cd ~` and press `Enter` to make sure you are at your home directory (~). This command will "change your directory" (`cd`) to the home directory.
2. In the terminal type `explorer.exe .` and this will open up the file explorer (`explorer.exe`) in the current directory (`.` This is the notation for current directory in Unix-like notation)



3. Turn on Navigation Pane in file explorer.

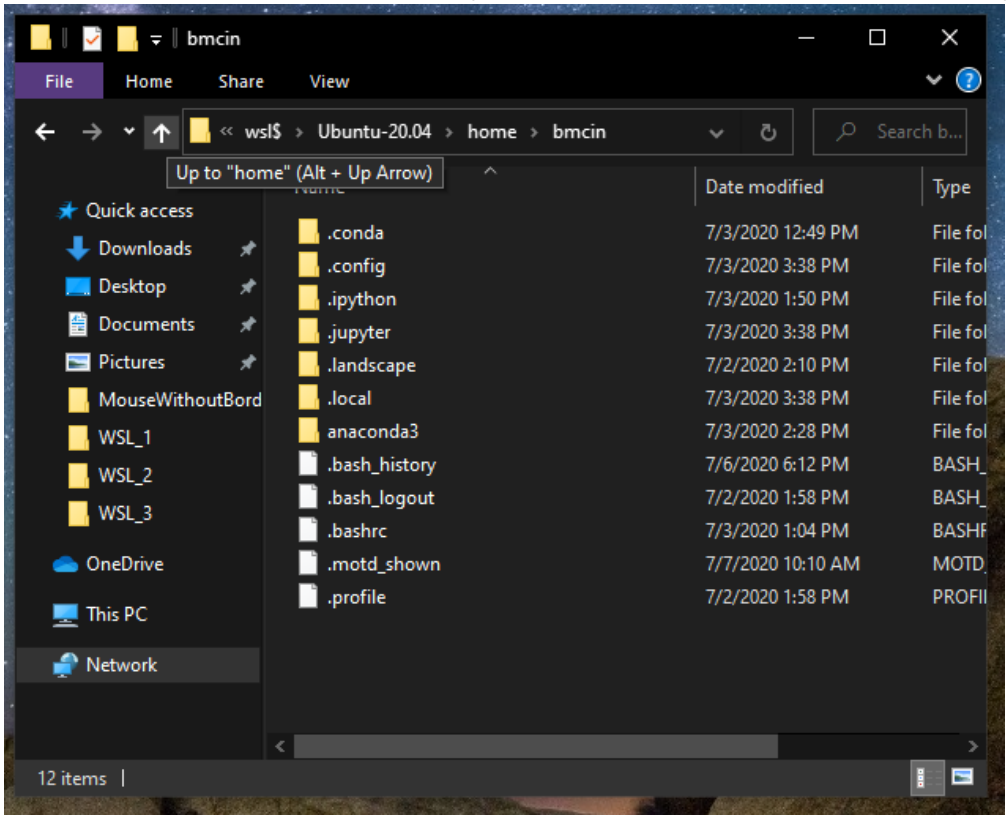


Click the `View` tab, then `Navigaton pane` , then `Navigation pane` again.



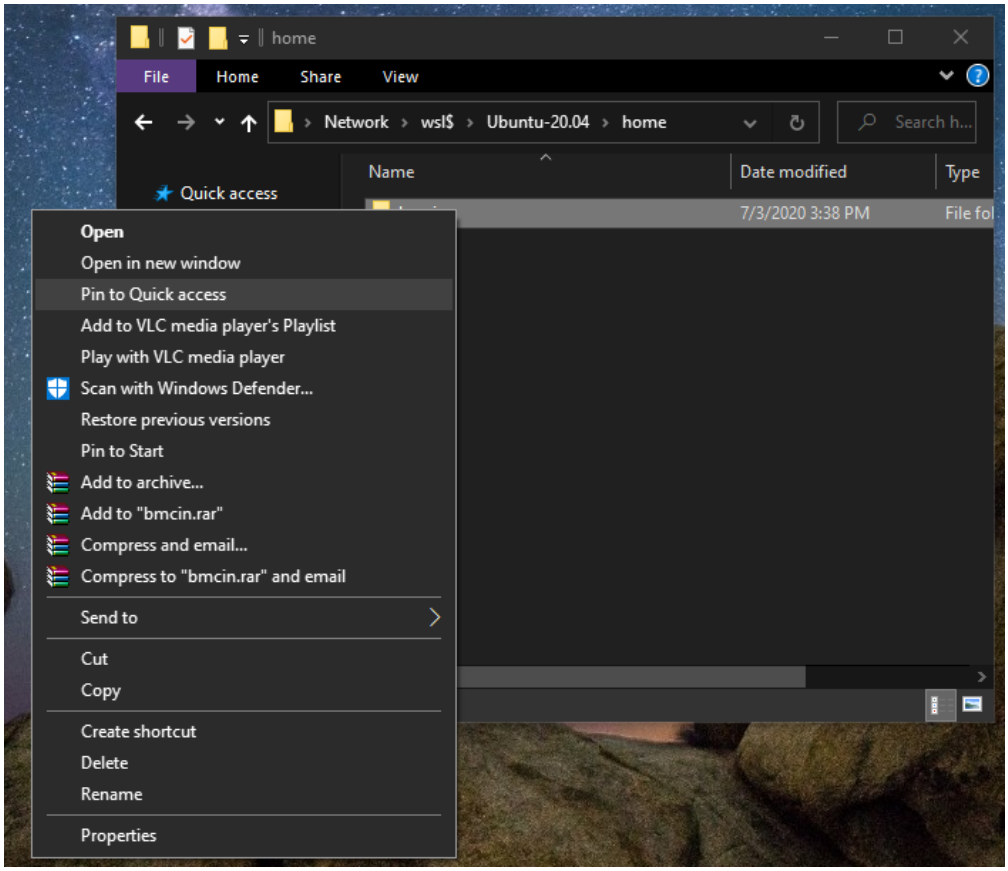
You will now notice the addition of the navigation pane in the file explorer. In the pane you will notice different sections. We will be focused on adding to the `Quick access` section.

4. Navigate to the "parent" folder of our current folder (aka one before the current folder we are in).

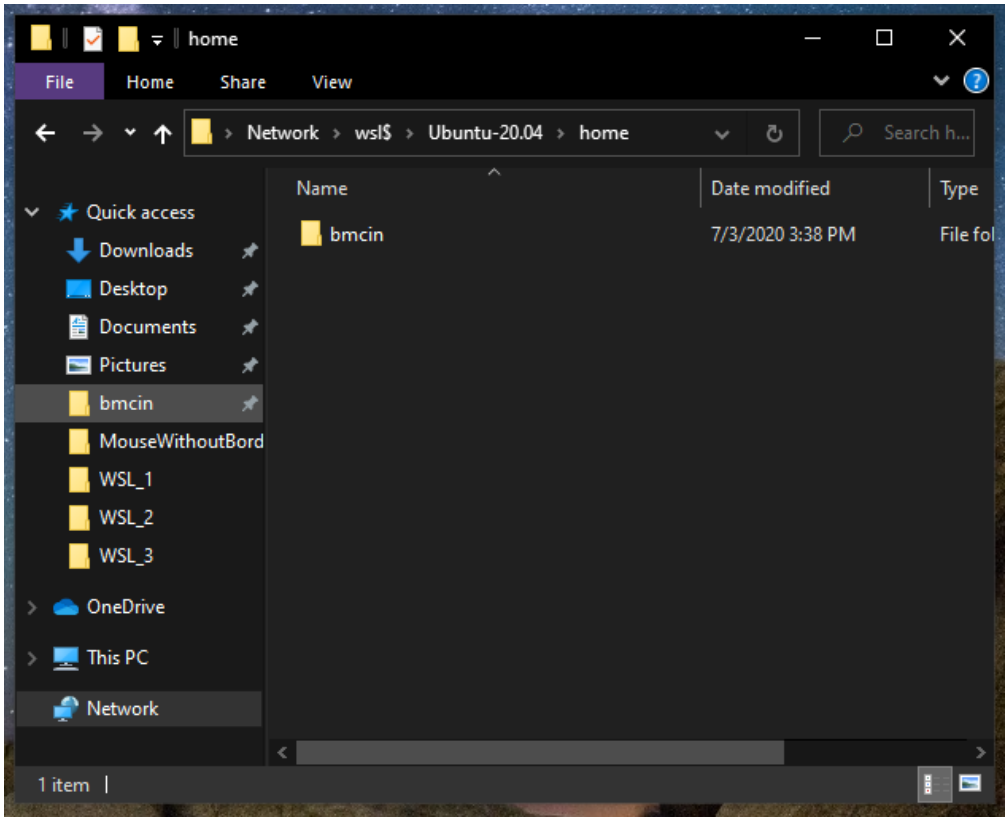


Click the arrow pointing "up" that is right next to the address bar.

5. Pin the folder we just came from to the `Quick access` .



Right-click the only folder in the file. Then click the `Pin to Quick access` option.



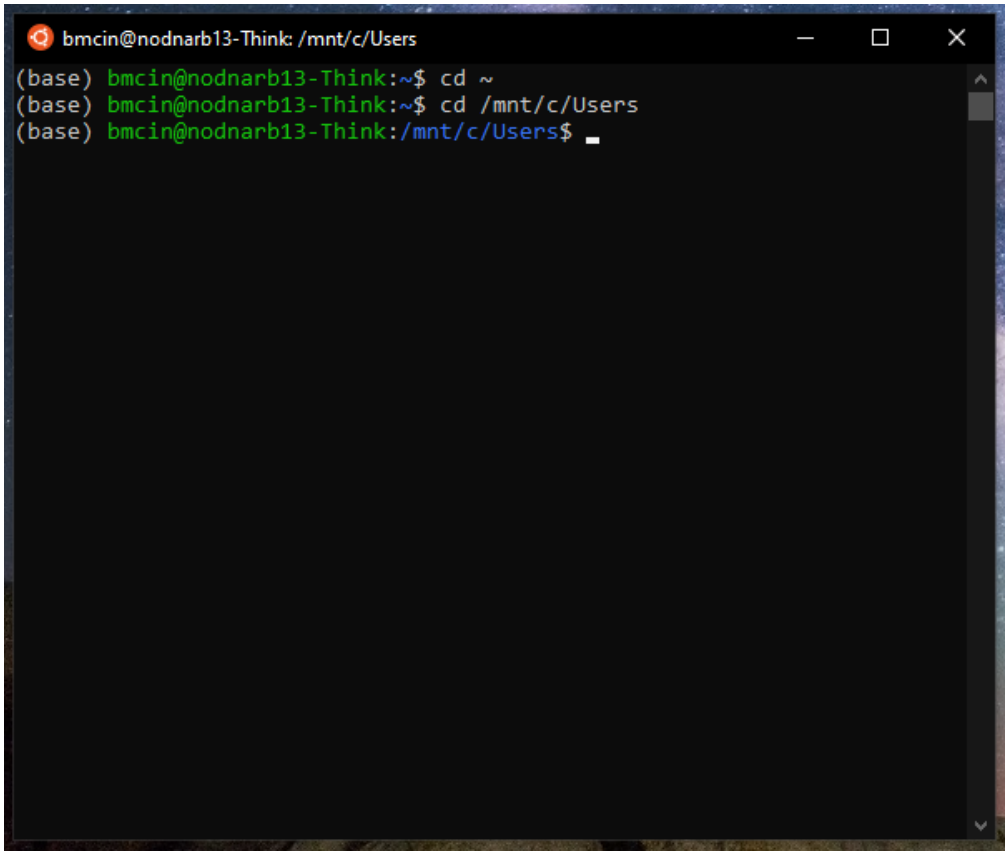
Now you should see the folder under the `Quick access` .

This will serve useful when you need to copy or place files in your home directory, or you want to access the files with the file explorer.

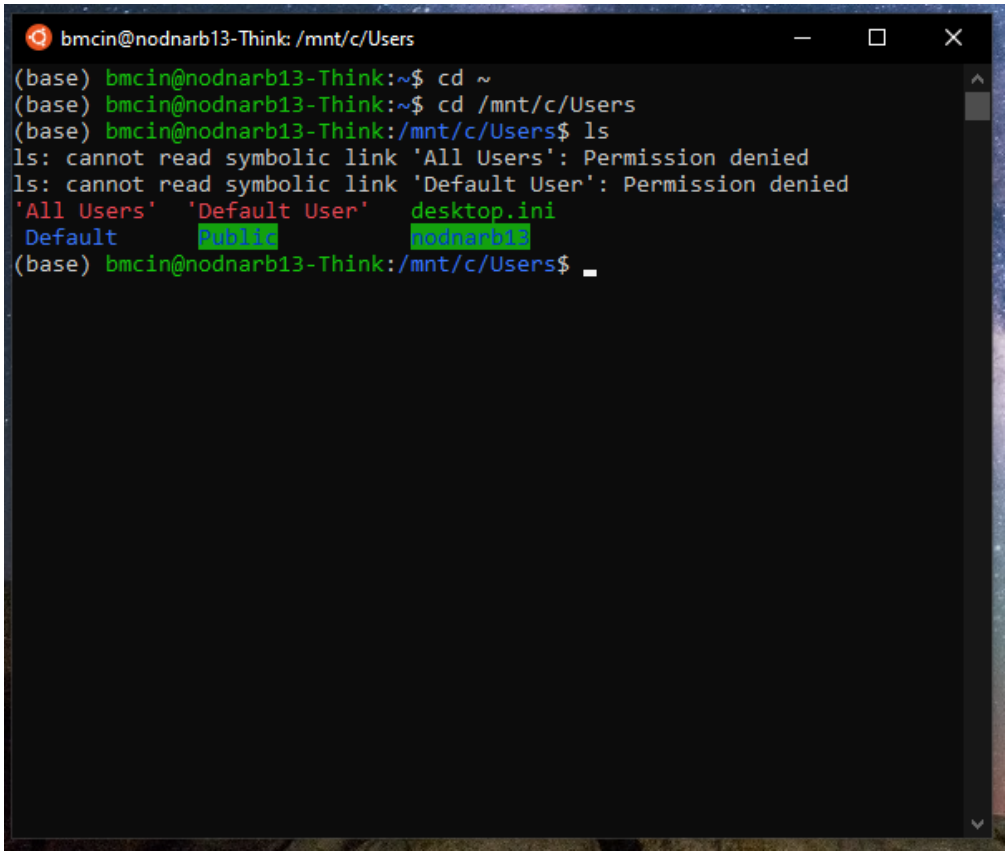
WSL to Windows 10 User Folder

When starting WSL and Ubunutu terminal, you techincally reside in part of your hard drive that is not apart of the Windows 10 file system. Above, we saw how to access this area with Windows 10, but here we will see how to access Windows 10 files from WSL. Particually, we will set up easy access to the Windows 10 folder that our friends with Git Bash (those that do not have windows 10) would call there "home" directory. This directory/folder is an important folder in windows 10 as it gives access to your desktop, downloads folder, documents folder, and many more folders you may use. To do this we will create what is known as an `alias` in your `.bashrc` file that links to this folder. An `alias` is a fancy name for a "text shortcut" that can be used in the terminal. The `.bashrc` is a settings file that your terminal uses to customize your terminal experince.

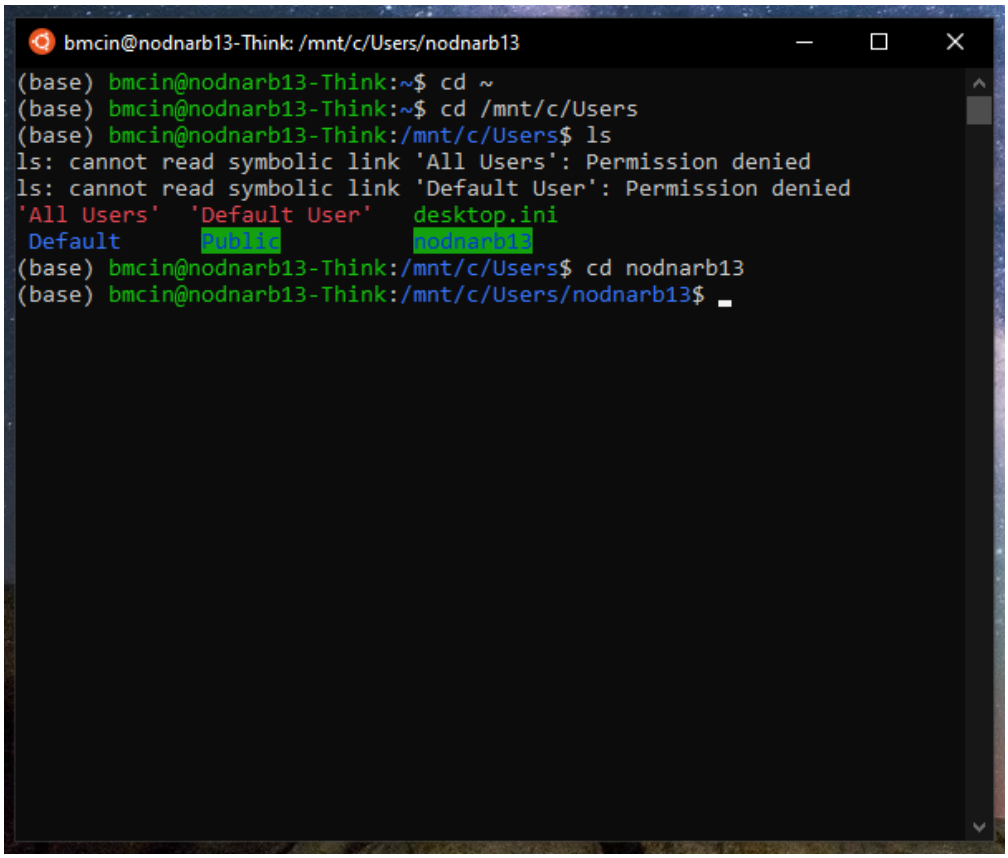
1. Open up the Ubuntu terminal, and type `cd ~` and press `Enter` to make sure you are at your home directory (~). This command will "change your directory" (`cd`) to the home directory.
2. Navigate to the C drive and go to the `Users` Directory.



   Type the command `cd /mnt/c/Users` and press `Enter` . This "changes directory" (`cd`) to the `Users` directory/folder. Notice the `/mnt/c` means that you are now in the `c` drive. The `c` drive is the drive that Windows 10 is installed, meaning that from this point on you will be accessing files that are accessible on Windows 10.
3. Print the directory/folder contents and navigate to the folder that is your Username.

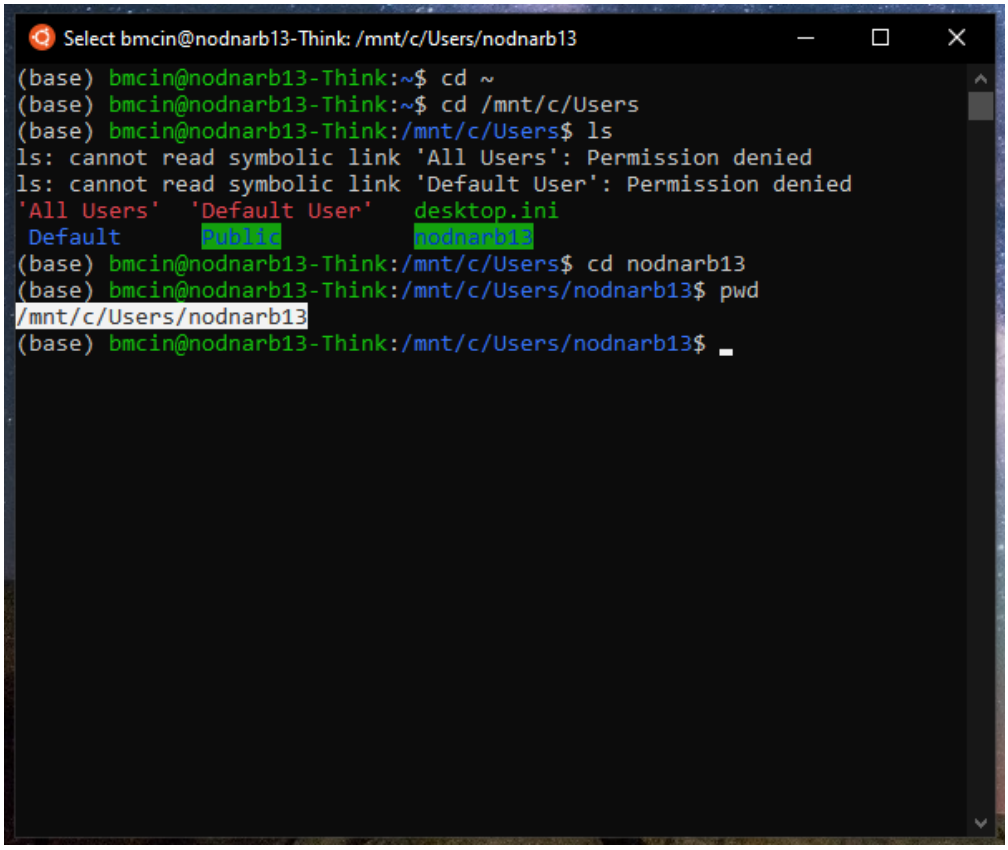


Type `ls` and press `Enter`. In the terminal you will see a list of folders and files. Notice that there is a folder that has your Username for Windows. In my case this is `nodnarb13`.

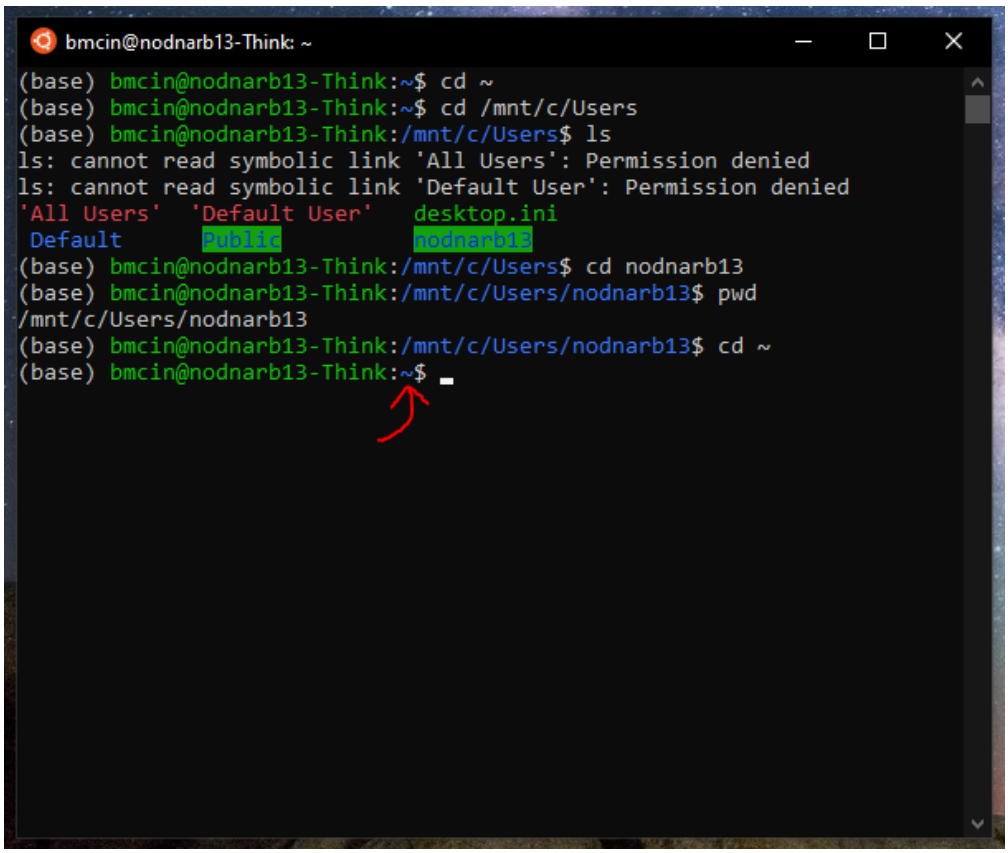


Using your username you will need to type `cd [USERNAME]` where in place of `[USERNAME]` you put the username you determined for last step. In the case of my computer this would be `cd nodnarb13`. Note as well you have a space in your username there may be `''` around your name.

4. Save the path to the User directory, and navigate back to the Home directory.

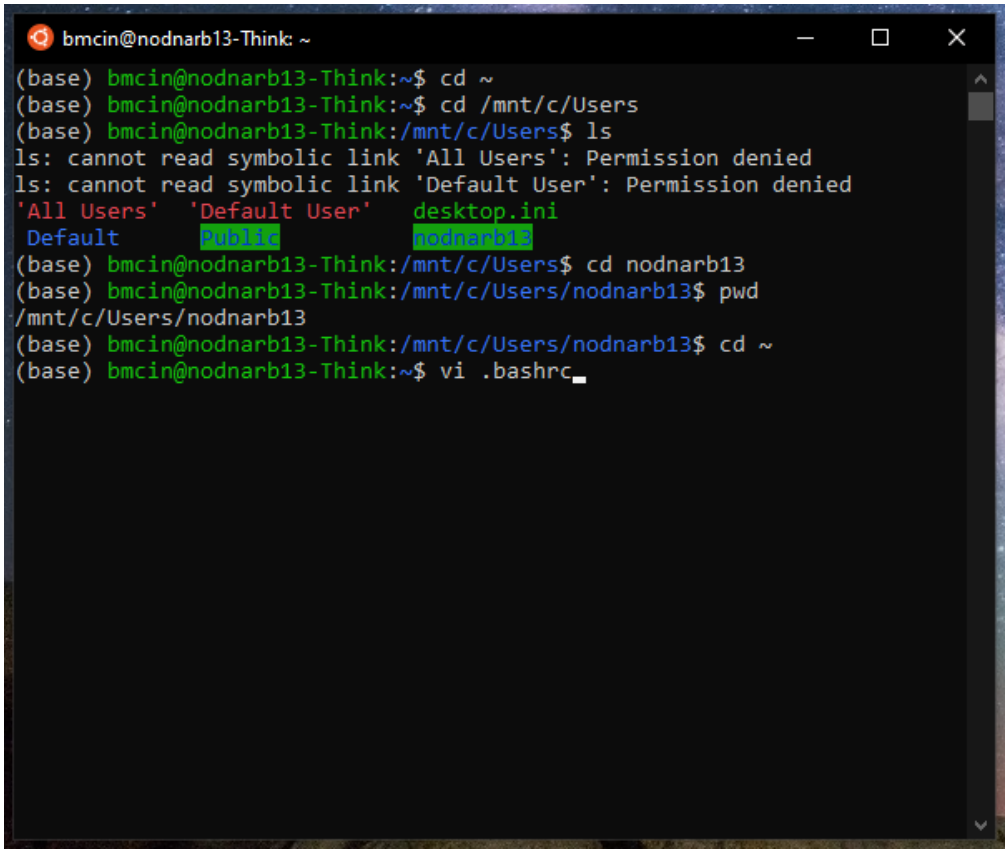


In the terminal type `pwd`. Then copy the results of the command by clicking and dragging over the text and pressing `Ctrl` + `c` to copy the highlighted text. If successful the text should become un-highlighted.

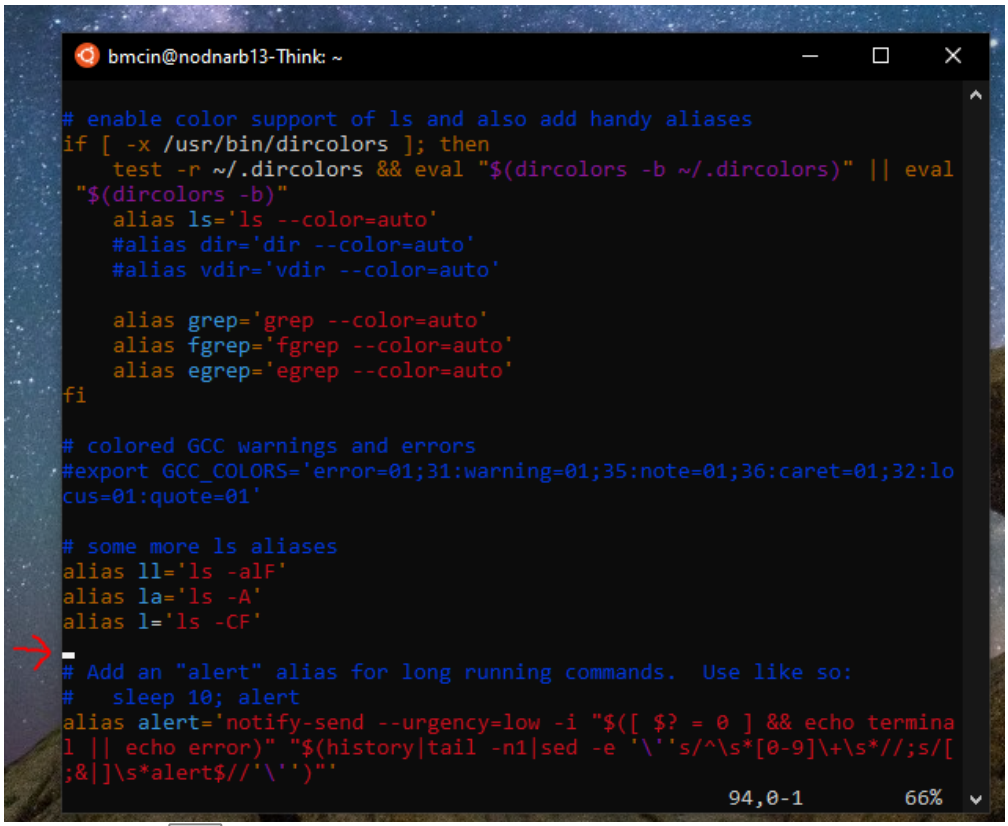


Go back to the Home directory by typing `cd ~` and pressing `Enter`. This will take you back to the home directory, which you can verify checking your location is at `~`.

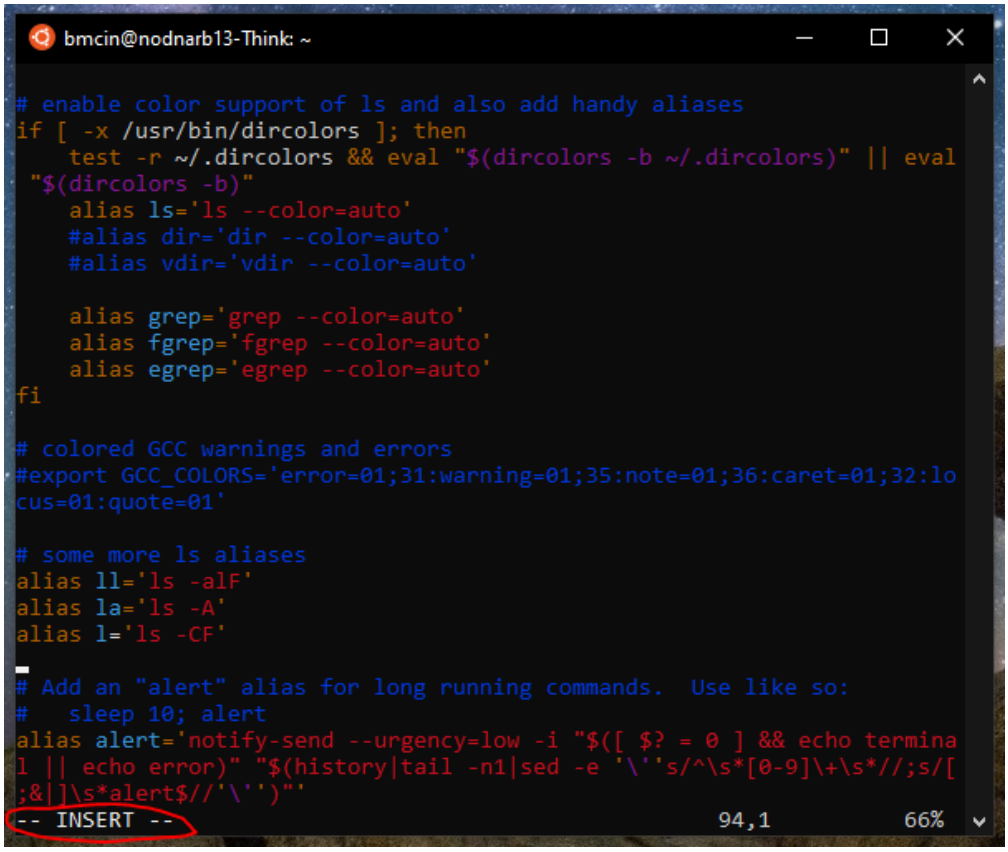
5. Add an `alias` to `.bashrc` for the User Folder path we just copied.



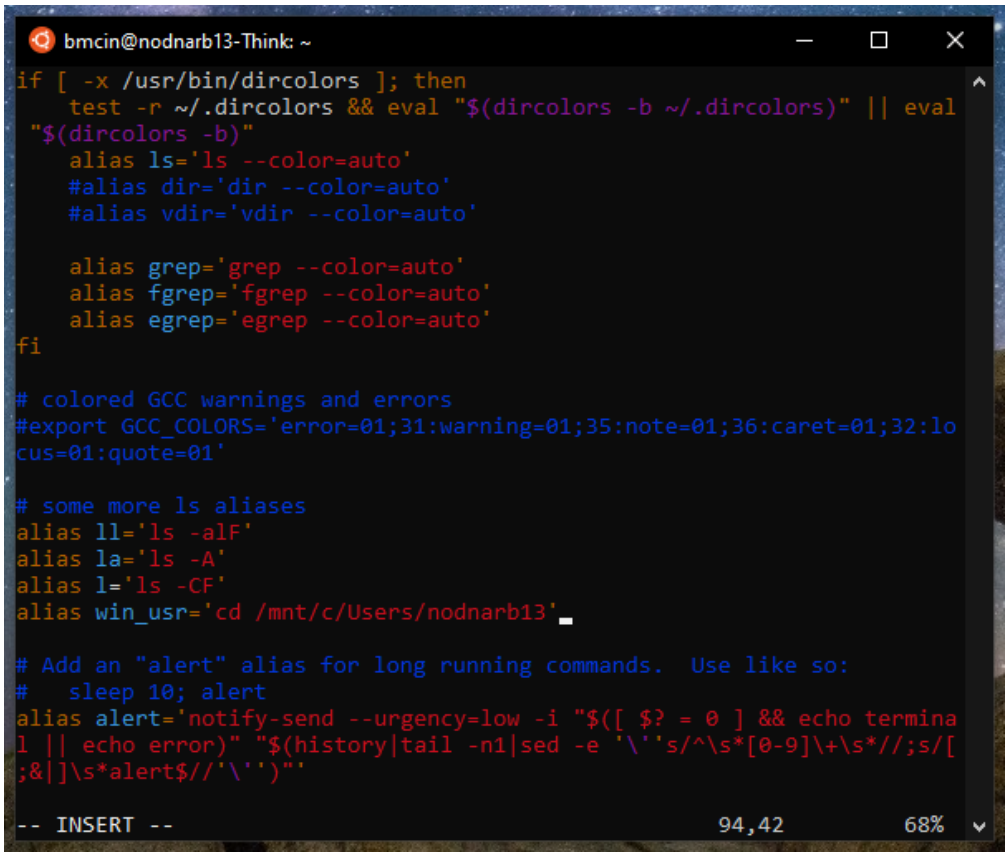
In the terminal type `vi .bashrc`. This will open up the `.bashrc` file in a command line text editor called Vim.



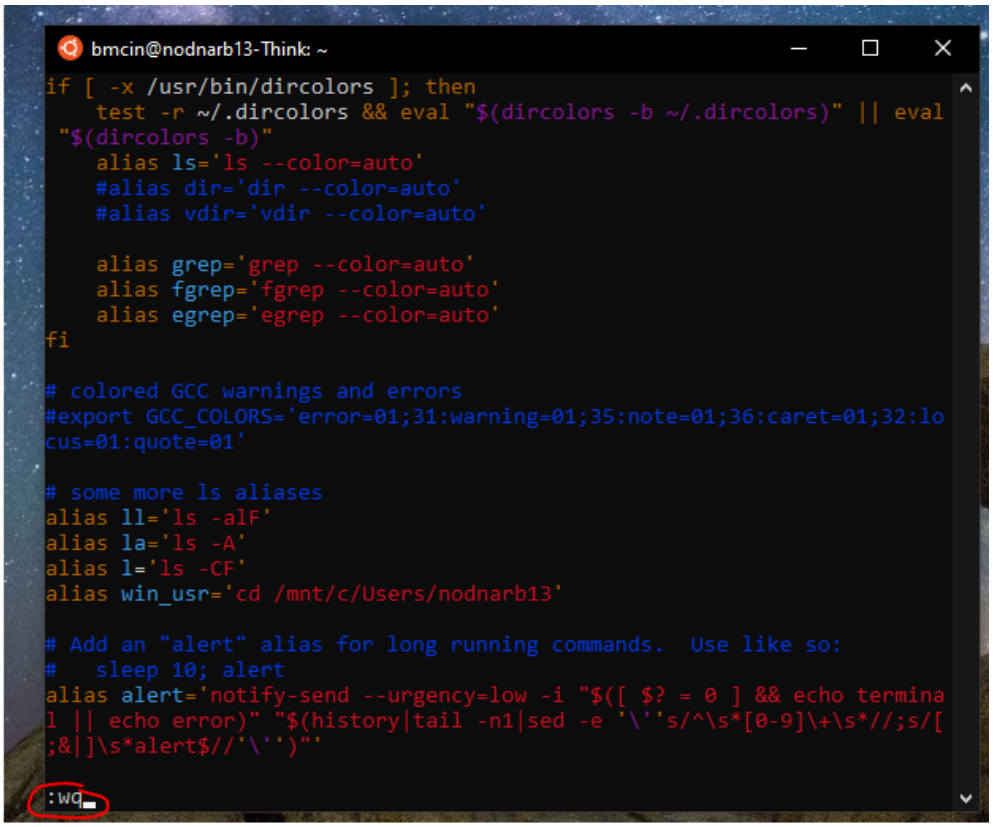
Using the down-arrow key [↓] on your keyboard. Scroll through the file until your cursor lands just after these aliases as shown.



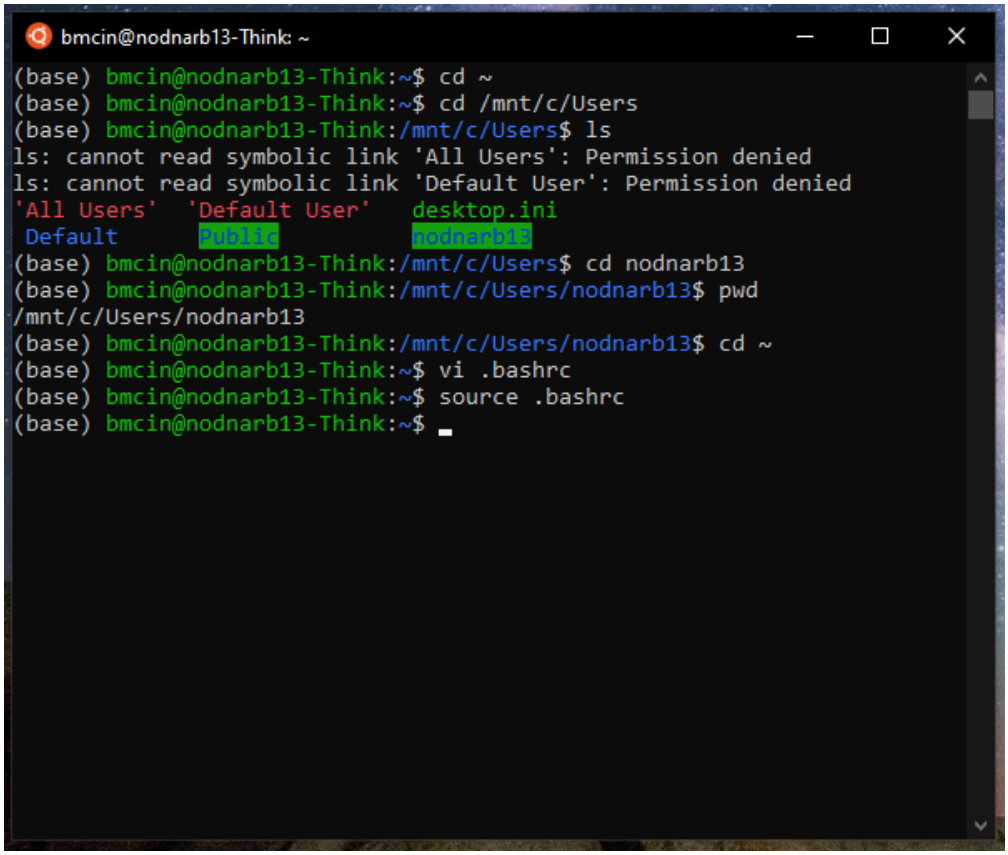
Press (lowercase) `i` . You will notice that this bottom left corner will go from *blank* to saying `-- INSERT --` . This changed the mode from "Command Mode" to "Insert Mode". Command Mode is where you can run commands such as save, exit, and the like. Insert mode is where you can type into the file and edit its contents.



On your keyboard press `Enter` to create a newline. Then on your keyboard press the up-arrow key ↑ once to move to the newly created blank line. Then type `alias win_usr='cd` . Using your mouse , in the terminal window *right-click* this should then paste the path you copied earlier to line you are typing on. Then finally type `'` . If all went well you should have `alias win_usr='cd [PATH_YOU_PASTED]'` . In my case this looks like `alias win_usr='cd /mnt/c/Users/nonarb13` . *Note: If any of the lines besides the one you just created were altered, this could break things. Make sure you did not alter any other lines before continuing. If you did happen to mess them up, press `Esc` on the keyboard and then type `:q!` and press `Enter` this will not save any of the changes you made, and you can try editing `.bashrc` again.*

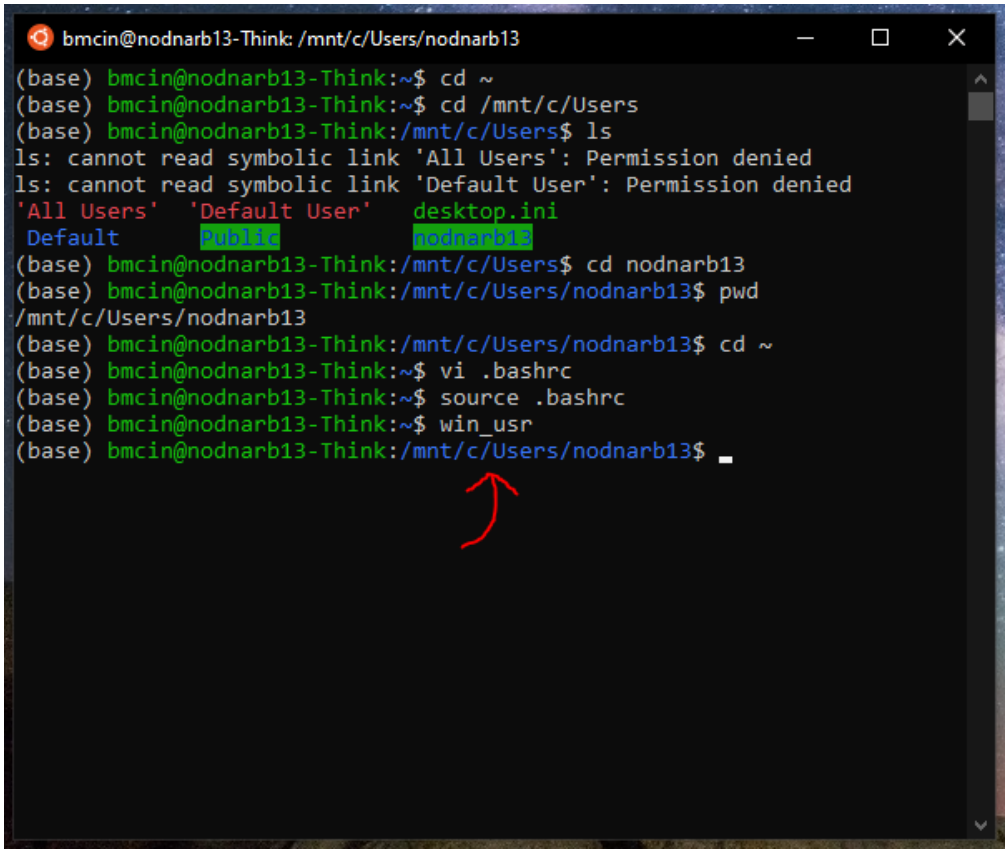


To save and close the file we will need to return back to the "Command Mode" to do this press the `Esc` on your keyboard. You will notice the bottom left-hand corner will go from `-- INSERT --` to *blank*. Then type on your keyboard `:wq` and press enter. This is the command to save (or *write* to the disk `w`) and to quit the text editor (`q`).

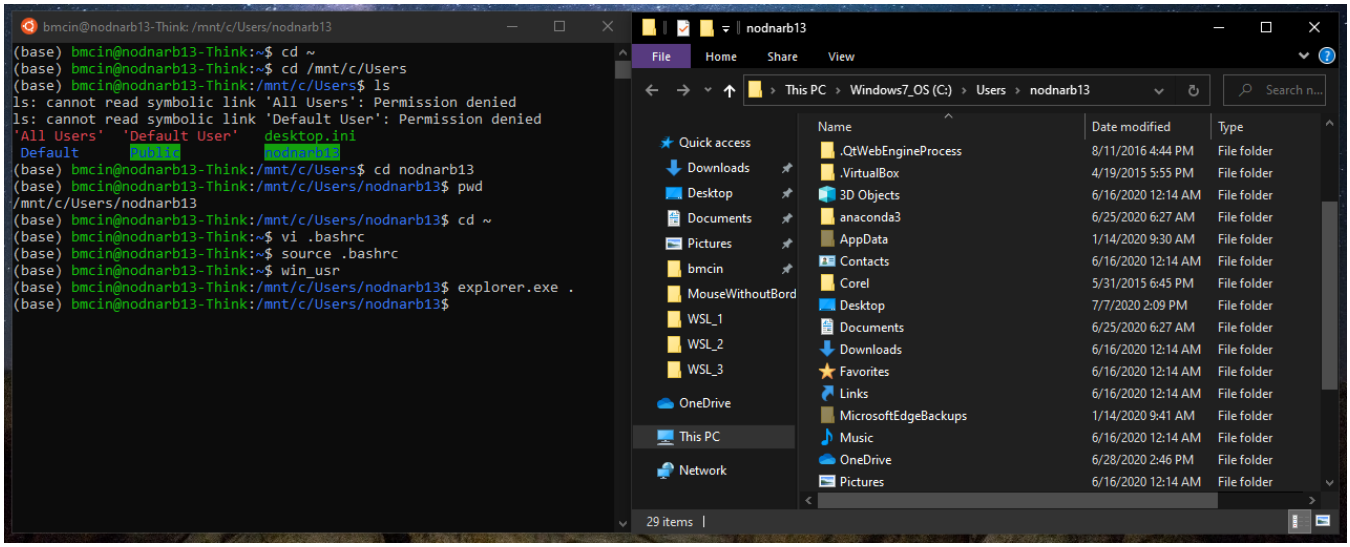


In the terminal, type `source .bashrc` and press `Enter`. This will apply the changes we made to the file. If you encounter an error it is most likely you did not close your `'` in your alias. You can reopen the file again with `vi .bashrc` and check to see if you spot an error.

6. Test the alias.



In the terminal try out the command by typing `win_usr` and pressing `Enter`. If all went well you will notice your location change to the appropiate folder.

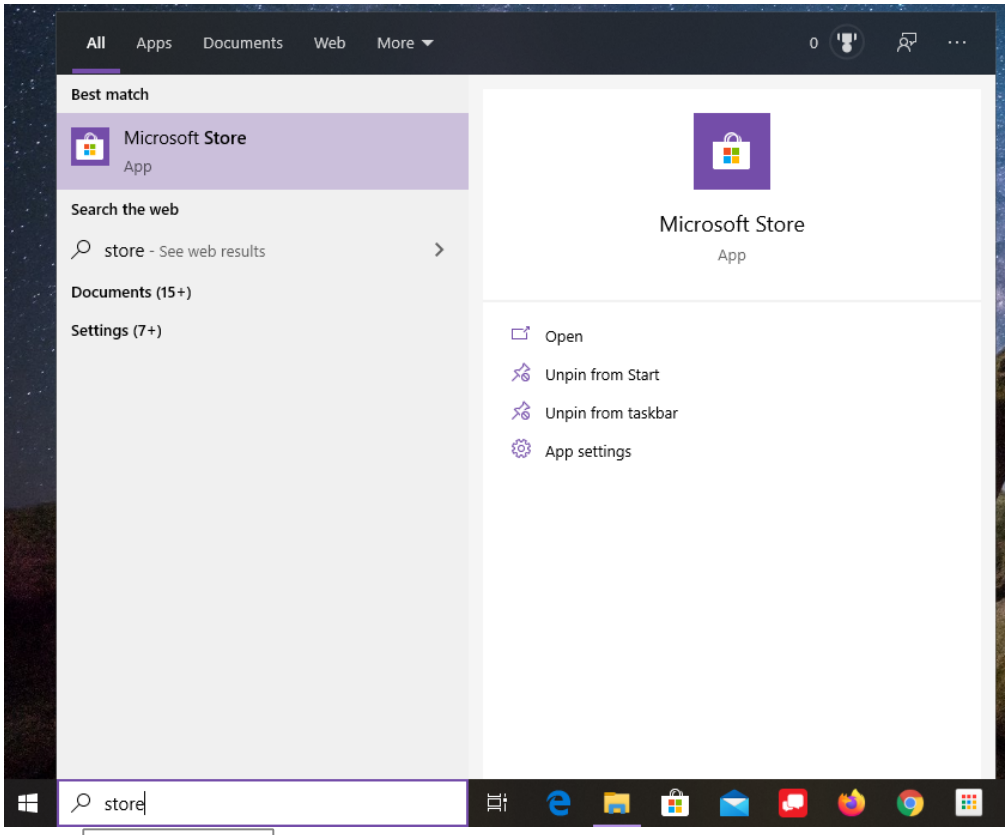


As stated earlier this folder has lots of important and useful folders in it that may come in handy later. You will learn more about navigating with the command line later, but for now you can type `explorer.exe .` to open the folder in the file explorer to see all the folders this folder gives access to.

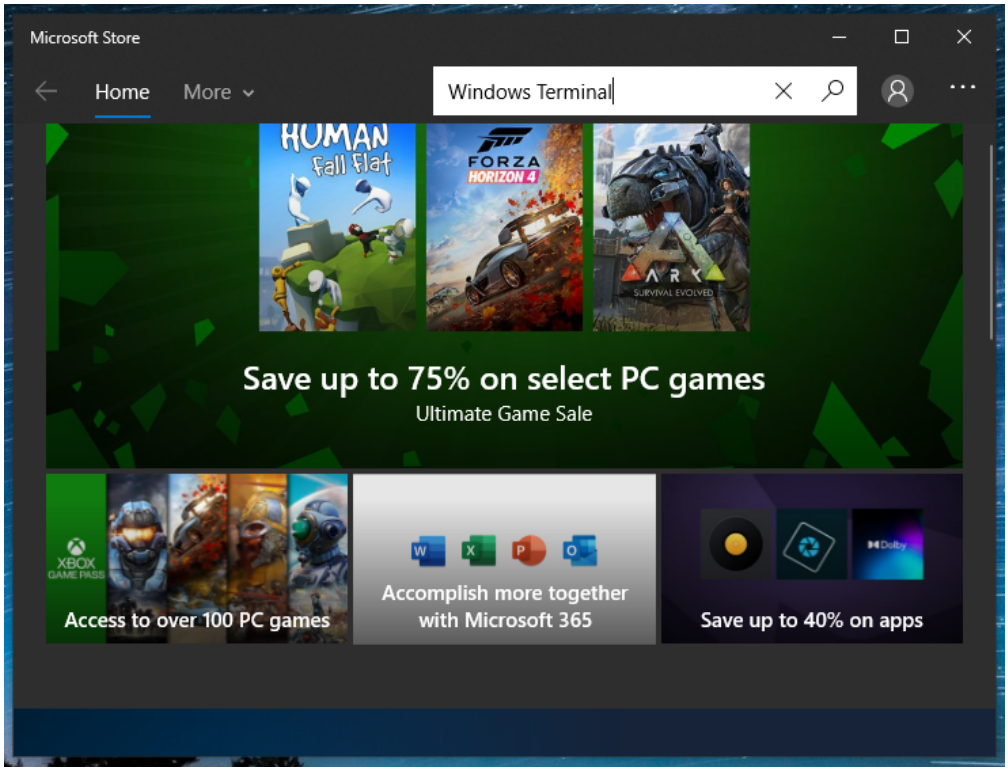
Using Windows Terminal as main Terminal

The Ubuntu Terminal works well as a terminal, however, sometimes you may want to run multiple instances (or windows) of the terminal or sometimes even run a powershell or command line terminal. Such an example is when you are a running Jupyter Notebook process (which locks up the terminal until the Jupyter Notebook process is stoped), but you also want to perform actions on the terminal such as creating and moving files. This would require you to open another Ubuntu terminal, which is not hard, but requires mutliple clicks and keystrokes and will produce yet another window you have to juggle around. When using Windows Terminal, opening another terminal is a simple as clicking one button and the terminals will become tabs making window management easy. This section will take you through how to setup Windows Terminal to work well with WSL and make terminal window management a little easier.

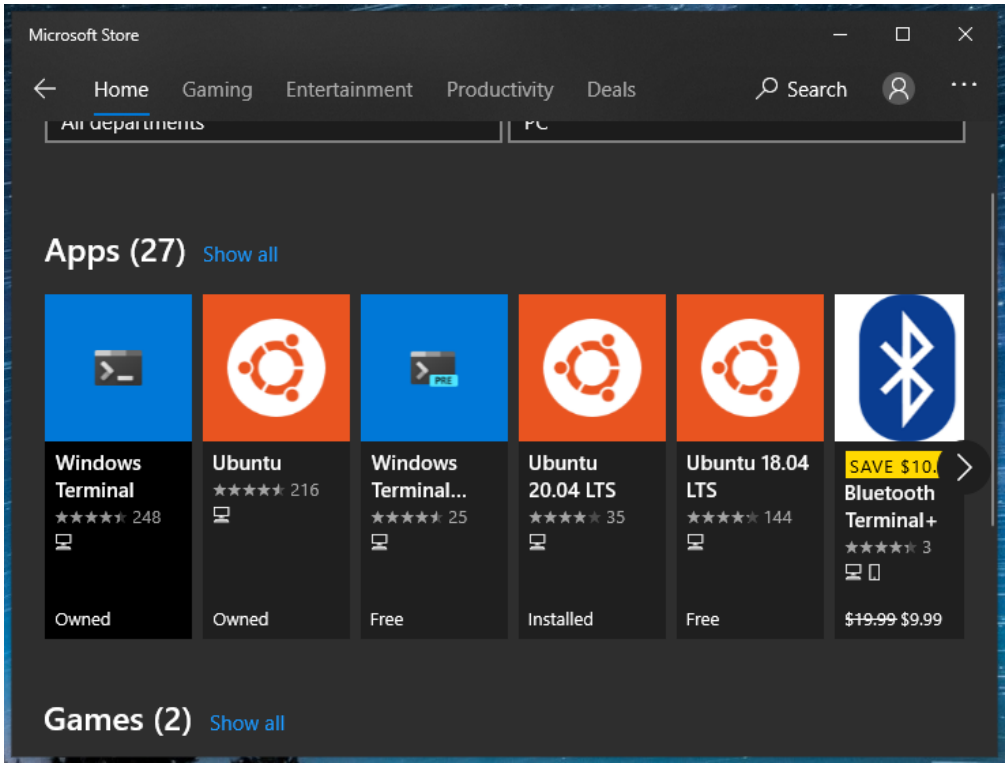
1. Have a connection to the internet
2. Naviate to the Windows Store and download `Windows Terminal`. Then start up the program.



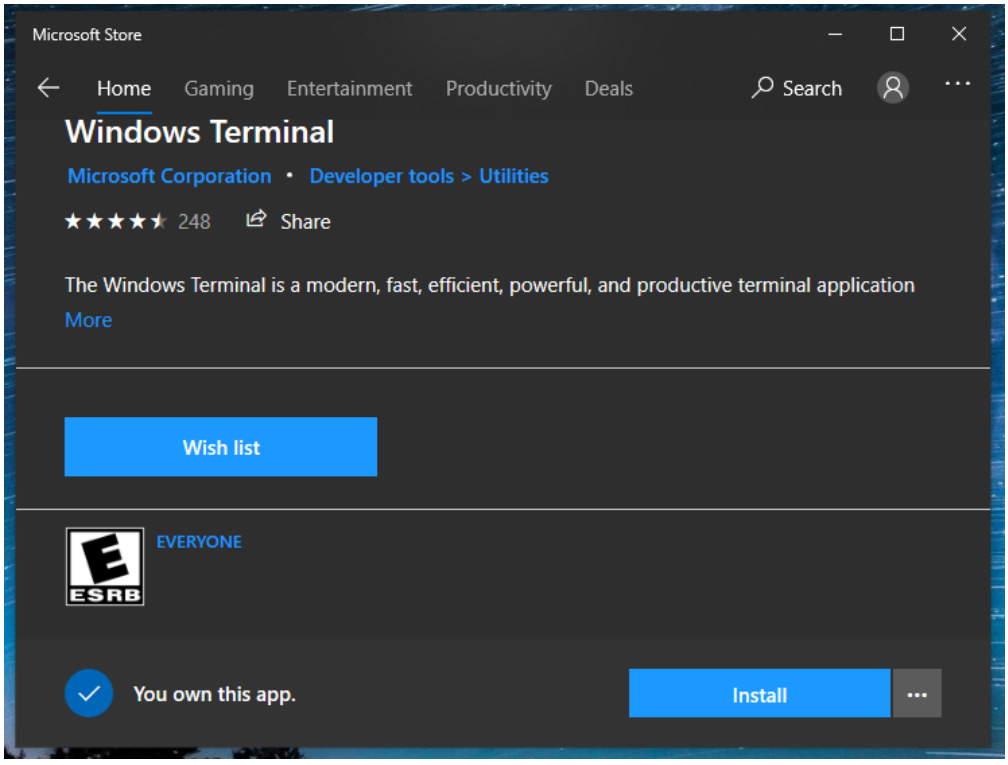
On keyboard press `Windows-key` or simply use the search bar on the taskbar if it is visible. Search `Microsoft Store` and click on the result.



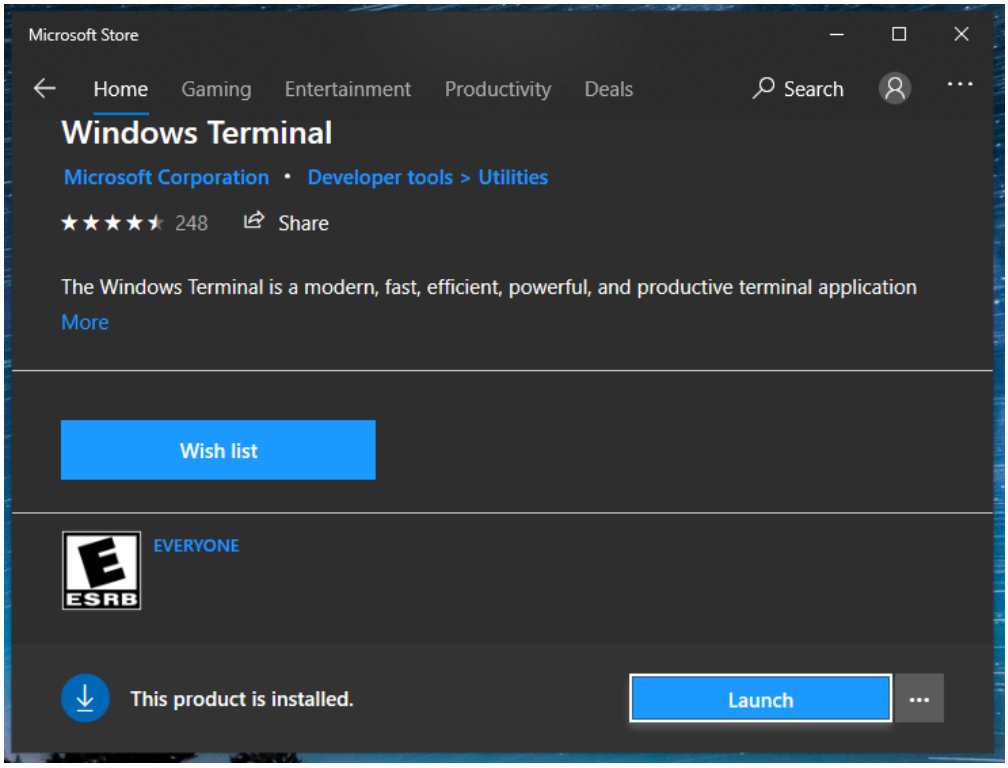
Using the search in the top right corner. Search for `Windows Terminal` .



Click the "App" entitled `Windows Terminal`.

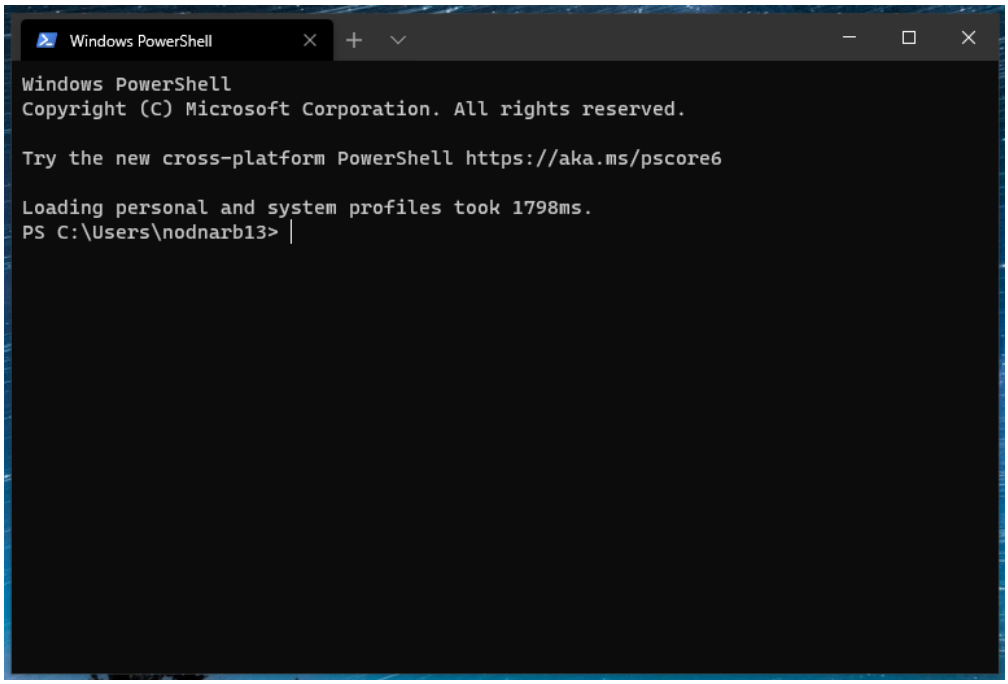


Press the `Get` button to download `Windows Terminal`. (This image has `Install` instead since this was already previously downloaded, your should say `Get` if you have never downloaded it before)

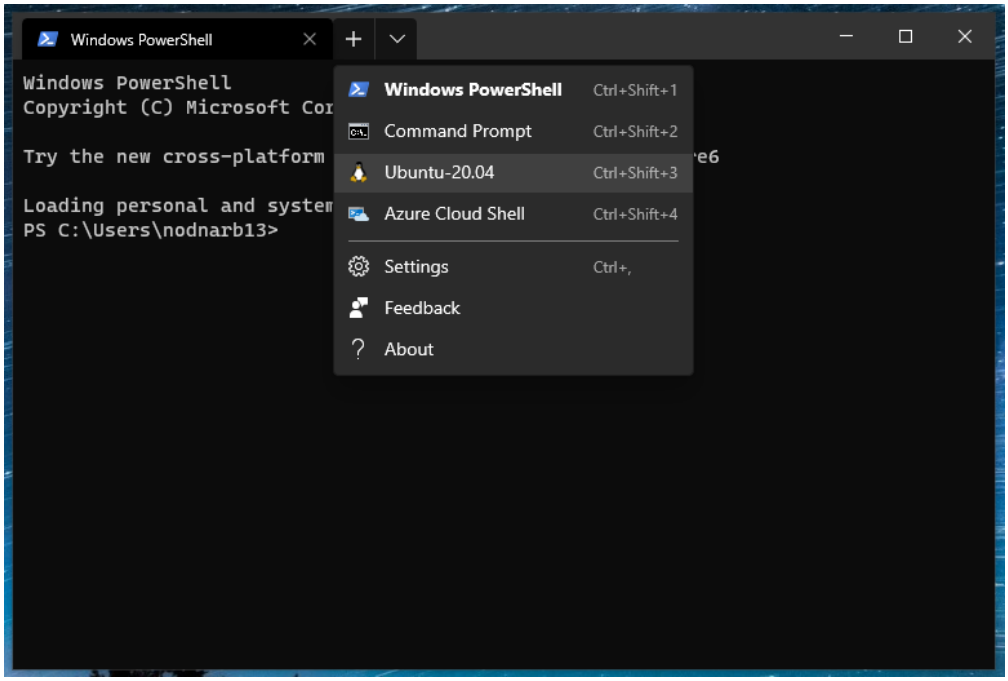


Wait until the package is installed fully. Then you can press `Launch`, however, please remember for the future that you can run Windows Terminal in the future by searching `Windows Terminal` using the search bar in the taskbar or `Windows-key`.

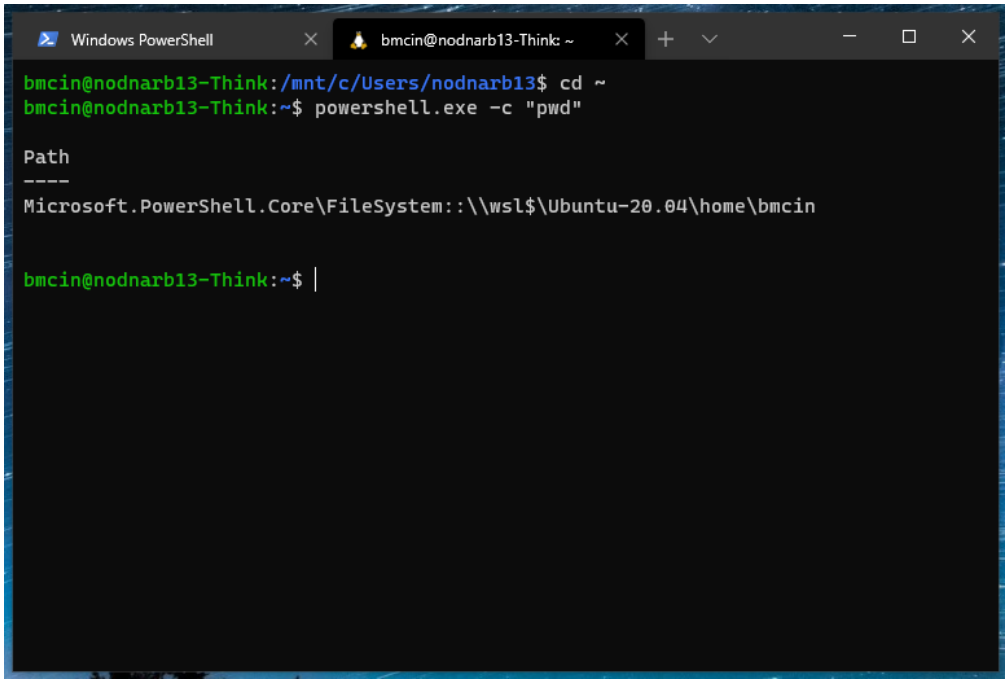
3. Change Default Startup Terminal and Startup location.



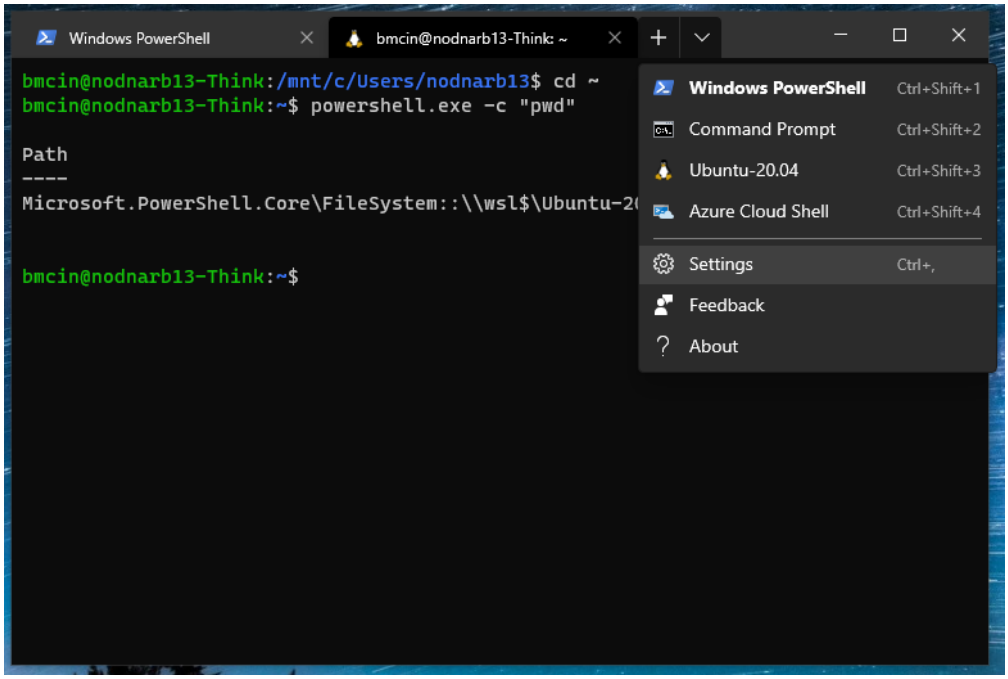
When opening Windows Terminal you should see something simialr to the image above. You will notice that the default terminal is Powershell and the starting location is in the Windows "home" directory, but not the WSL "home" directory. However, both these defaults can be changed by editing the settings.



In order to change the default start location, we are going to need to find the Windows Path to the WSL home directory. In order to find this first we are going to need to open up a Ubuntu Terminal. To do this, click the ▼ (down arrow) next to the `+` (plus sign) on the tab bar. There you will have the option to select Ubuntu. Click that option.



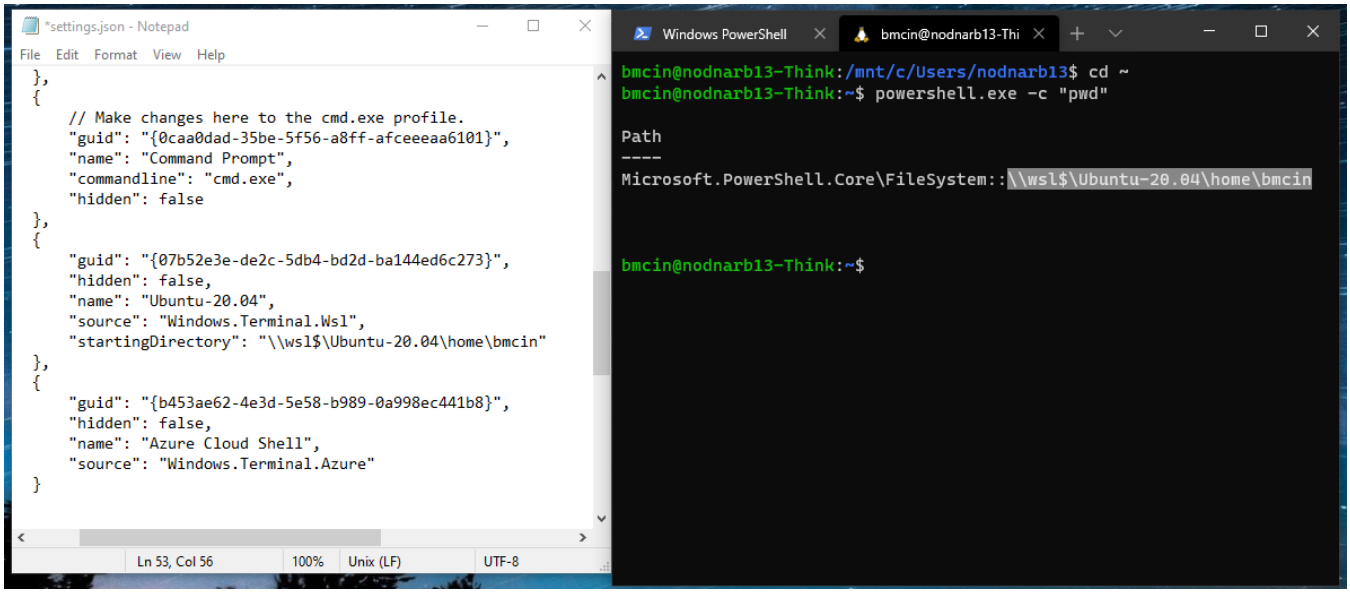
Now we can use WSL built in ability to run powershell in order to grab the Windows path that we need. In this terminal type `cd ~` go to WSL's home directory. Then type `powershell.exe -c "pwd"` to use powershell to print the home directory through the Windows path. You should get something that looks like the image above. We will use this in a moment.



Now click again on the ▼ (down arrow) next to the + (plus sign) on the tab bar. This time, click on the `Settings` option.

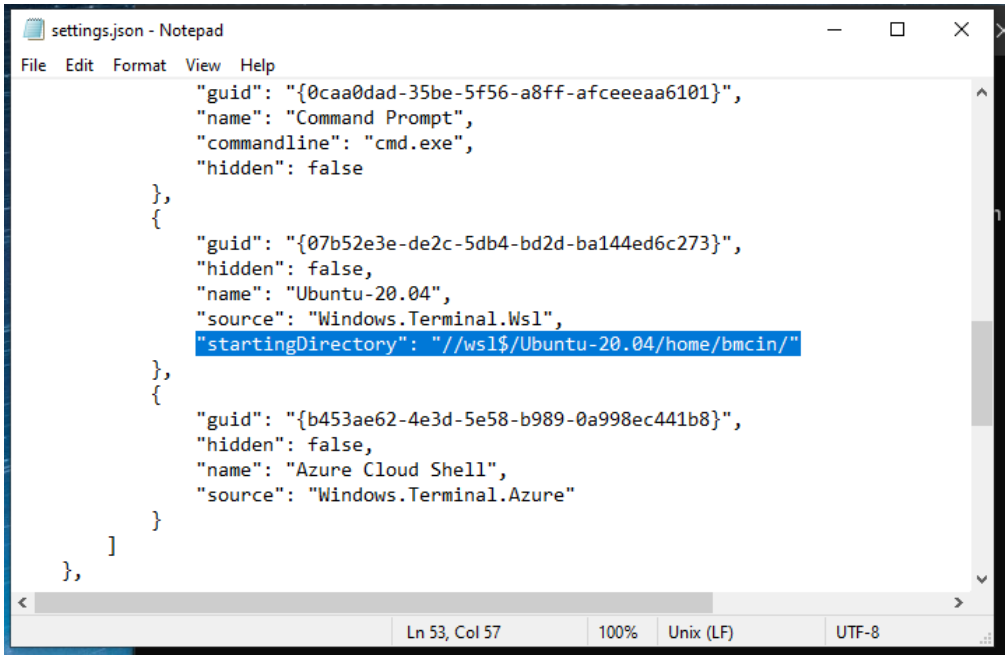
Depending on your perferences, the `settings.json` file should now open up in `Notepad`. It is possible it may open up in a different program, or it may ask you which program you would like to open it. In the file, you will need to scroll down until you see something that looks like

```
{
    "guid": "{07b52e3e-de2c-5db4-bd2d-ba144ed6c273}",
    "hidden": false,
    "name": "Ubuntu-20.04",
    "source": "Windows.Terminal.Wsl"
},
```

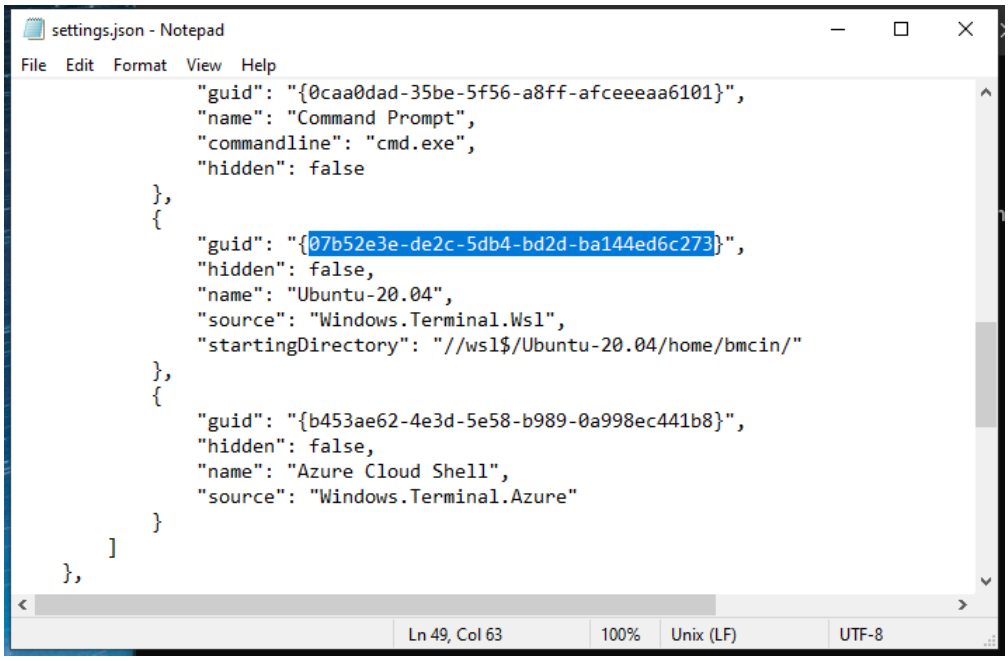


You will now need to edit this section. First, you will need to add a comma to the end of the "source" line. Then you will need to create a newline underneath that by pressing `Enter` at the end of the "source" line. You will notice that then moving to the new line it will start at the beginning of the line on the left. In order to get the correct indentation you will need to press the `Tab` key twice. Then in the new line you will need to add the following: `"startingDirectory": "[PATH]"` . Where the `[PATH]` is the Windows Path we got earlier in our Ubuntu Terminal. The Path is the highlighted part in the image above (`\\wsl$\Ubuntu-20.04\home\bmcin`). It starts with `\\wsl$` and will need to be copied exactly into the settings. For example the section should now look like this.

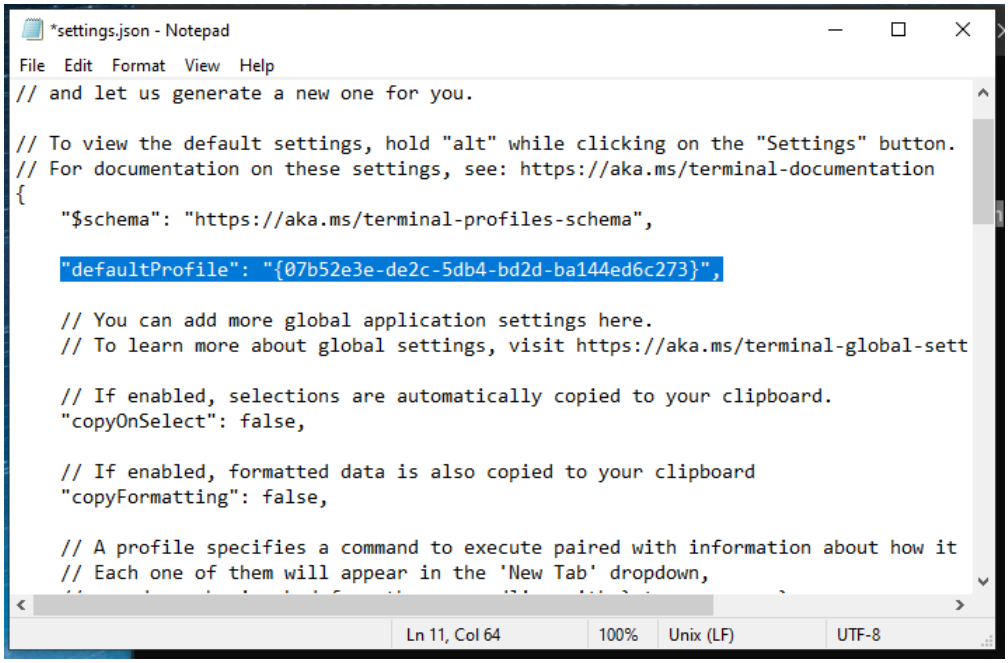
```
{
     "guid": "{07b52e3e-de2c-5db4-bd2d-ba144ed6c273}",
     "hidden": false,
     "name": "Ubuntu-20.04",
     "source": "Windows.Terminal.Wsl",
     "startingDirectory": "\\wsl$\Ubuntu-20.04\home\bmcin"
 },
```



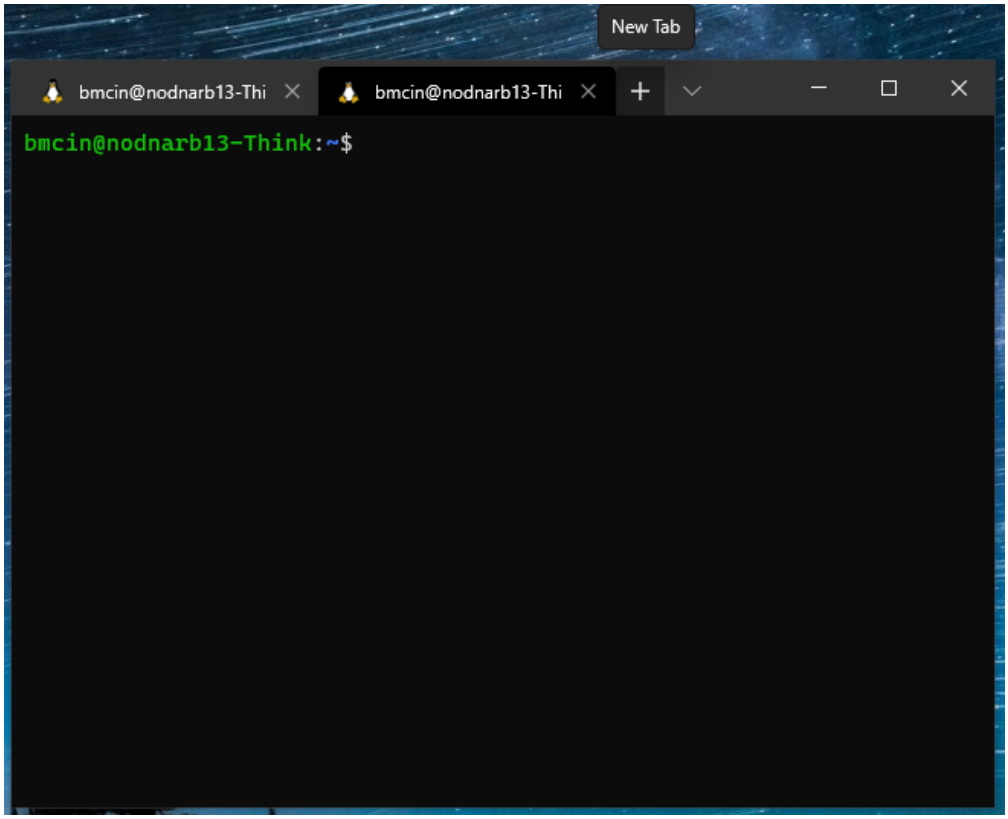
Now we are going to need manipulate the "startingDirectory" slightly in order for this setting to work. What we will need to do is change every `\` (back-slash) to a `/` (forward-slash). We will also need to put a `/` (forward-slash) at the end of the path as well. The line should now look like `"startingDirectory": "//wsl$/Ubuntu-20.04/home/bmcin/"`.



The starting location is now fixed, but to make things even easier, we will need to change the default starting terminal. In order to do that we first need to copy the `guid` for Ubuntu. That is the first line of the section we just edited. You can see what we need to copy in the image above.



Now you will need to scroll up the page until you see a line that says `"defaultProfile": "{61c54bbd-c2c6-5271-96e7-009a87ff44bf}",`. Here, you will take your recently copied `guid` and paste it into the curly brackets, replacing the guid that is already in there. You can then close and save the file.



To test it out, close the Windows Terminal and reopen it. You should now see the Ubuntu terminal is the first terminal to open, as well as the starting location is now the WSL home directory. To quickly make another ubuntu terminal/tab, you can now press the `+`. *Note: To open other terminal such as Powershell or Command Prompt you can use the ▼ (down arrow) next to the `+` (plus sign) on the tab bar and click on any terminal you would like to open*

Useful Commands

- `cd [DIRECTORY/FOLDER_NAME]` -- Will change the current directory your terminal is in to the folder specified.
- `cd ..` -- Will change the current directory your terminal is in to the the parent directory/folder of the current directory/folder.
- `explorer.exe .` -- Will open up the current directory your terminal is in, in Windows file explorer.
- `notepad.exe [FILE_NAME]` -- Will open up the specified file with the Windows notepad program
- `clear` -- If you want to "clean" up the terminal to clear past commands from view, you type this command.
- `exit` -- This will close out the terminal
- `cat [FILE_NAME]` -- This will print all the contents of the specified file into the terminal for easy viewing.
- `jupyter notebook` -> This will open up Jupyter Notebook in your current directory.

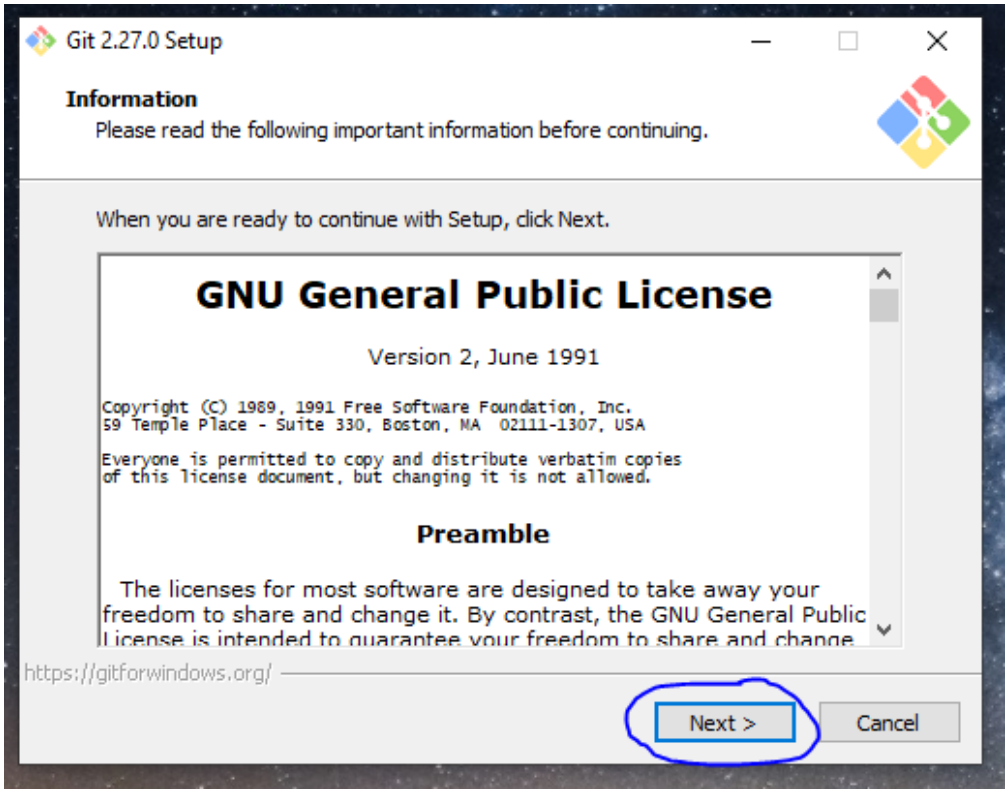
Continue to the next section by clicking here.

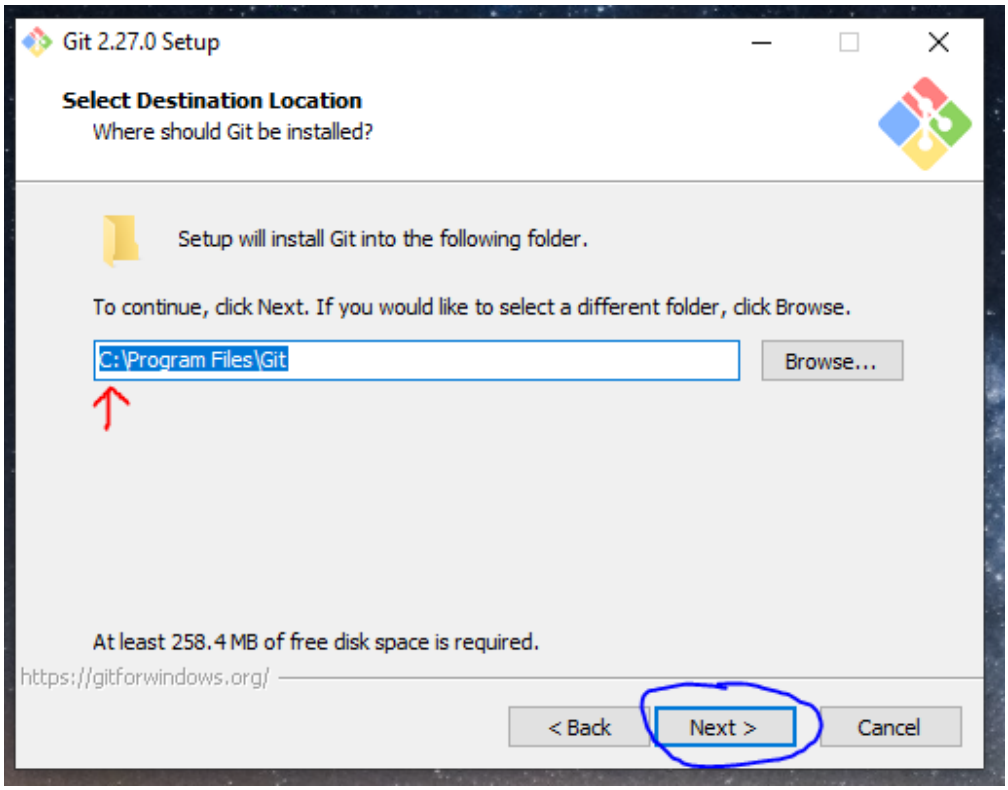
If you do not have Windows 10 or WSL installation did not work

While it is possible to use MS-DOS (Command Prompt and Powershell) as a terminal, in order to keep things simple we will download a Unix-like Command Line Interpretor Emulator to get the job done. In our case we will be utilizing a package that has both Git-Bash, a terminal, and "Git" (a version control utility, we will learn more about this later) baked in.

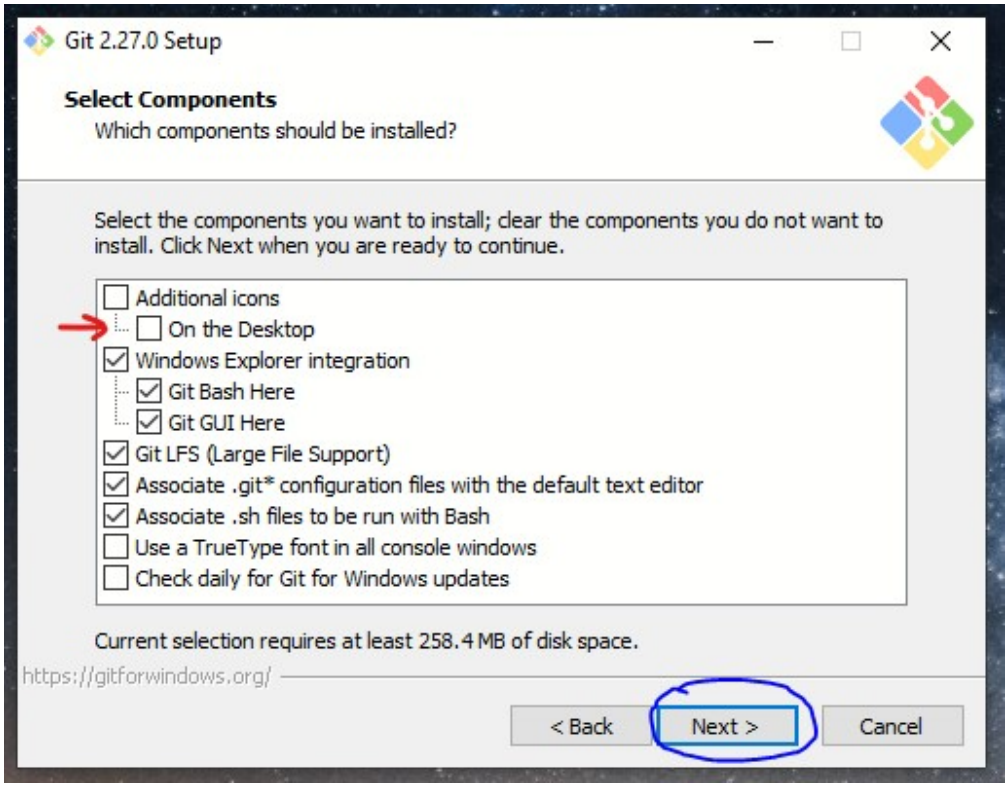
Installing Git-Bash and Git.

1. Go to the [Git Download webpage (https://git-scm.com/downloads)](https://git-scm.com/downloads), and download the Windows version.
2. After downloading, run the Git exectuable file. Say `yes` to any warnings.

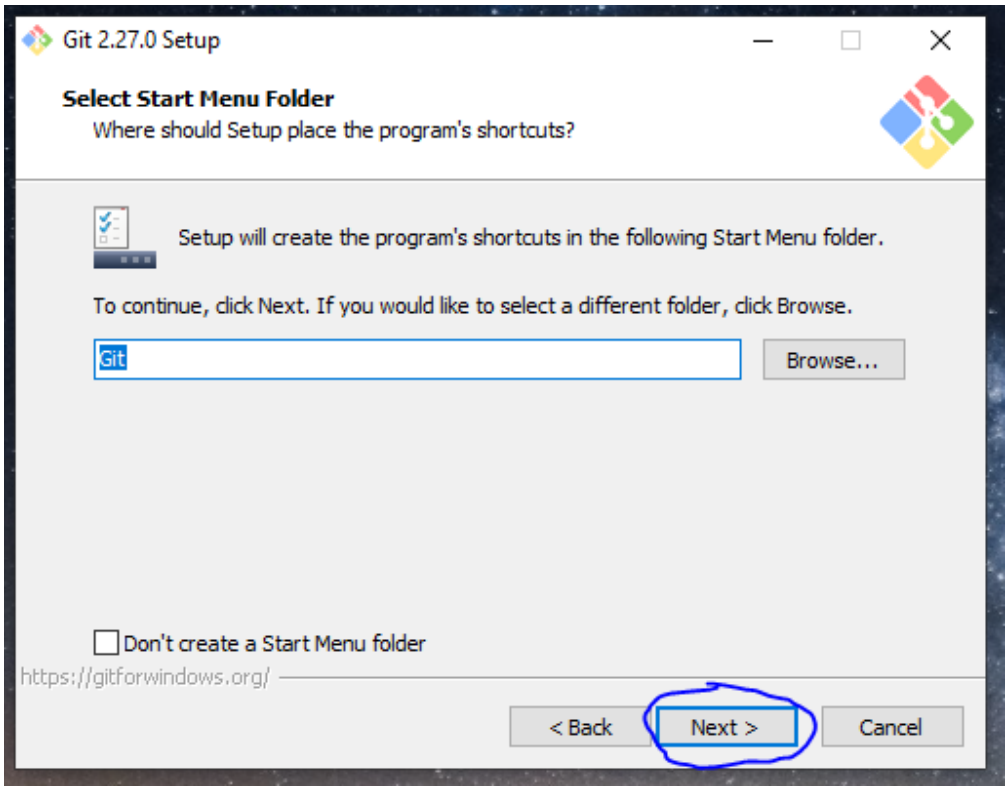


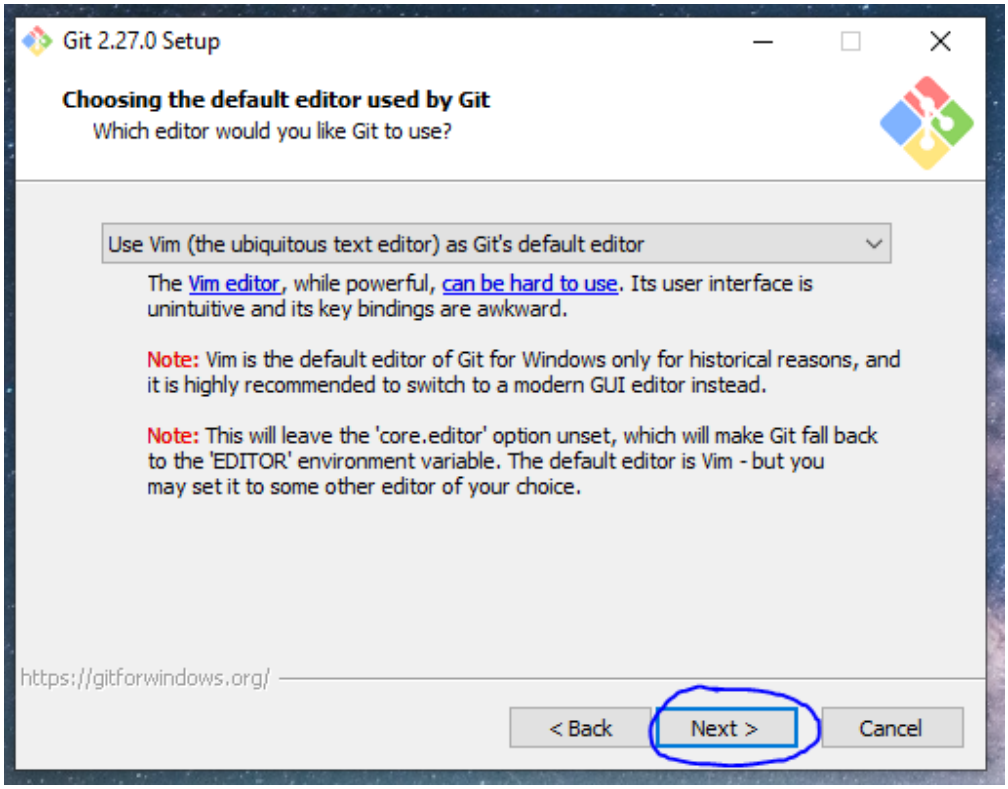


It is suggested to leave this alone, however, if you do want to change the location, make sure it is in the same drive that Anaconda is installed in.

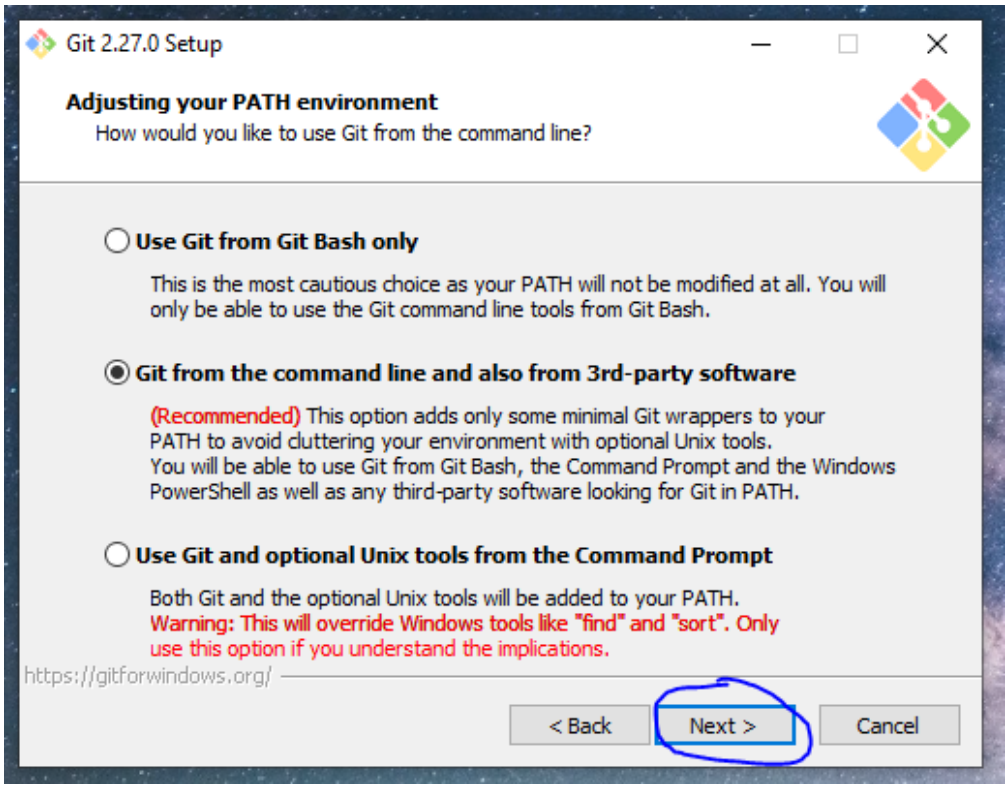


If you click this option, a shortcut linking to the terminal will be placed on your desktop. Alternatively, once the terminal is started you can pin the program to your taskbar, or search for it with `Windows-key`

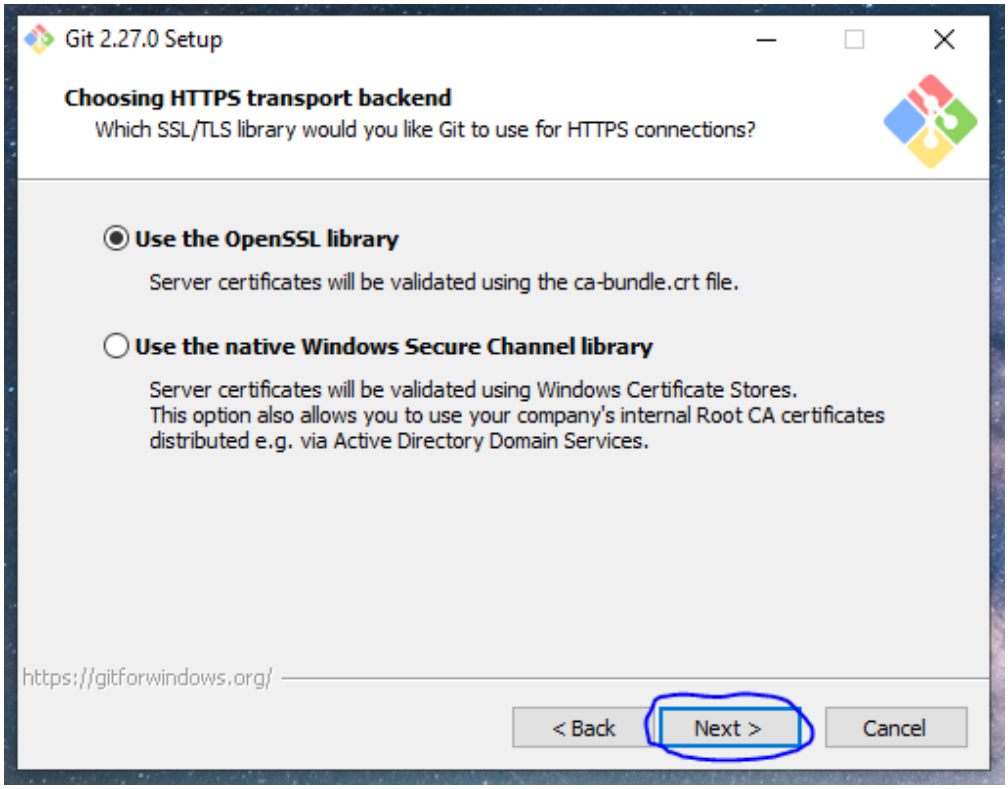


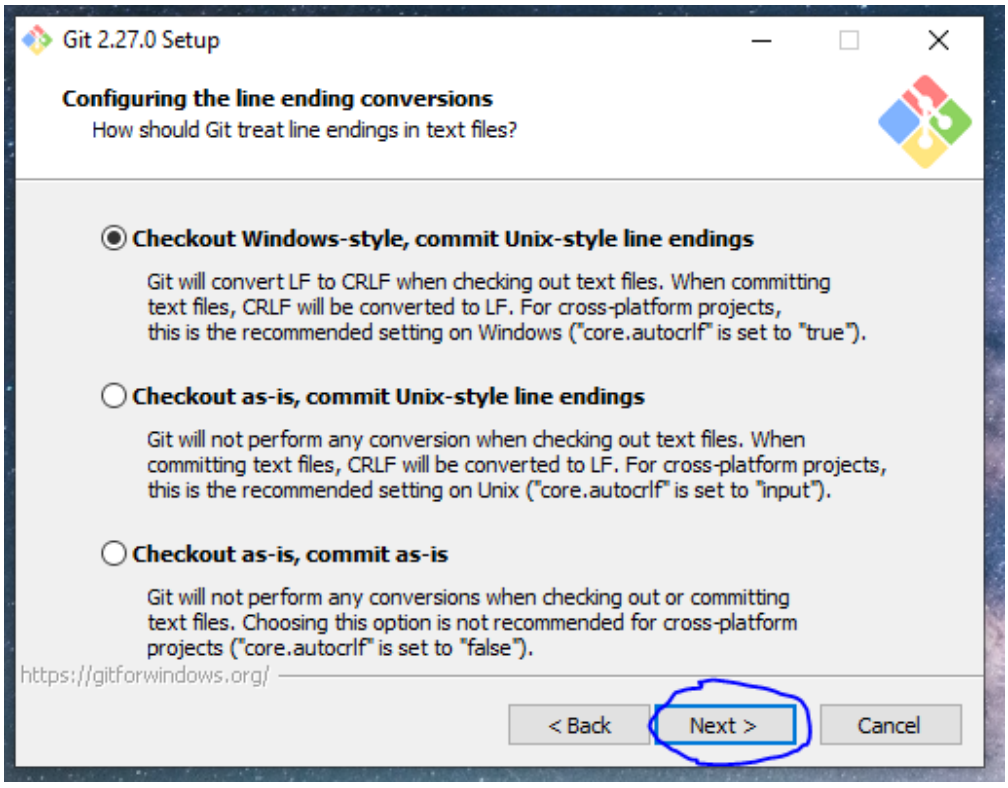


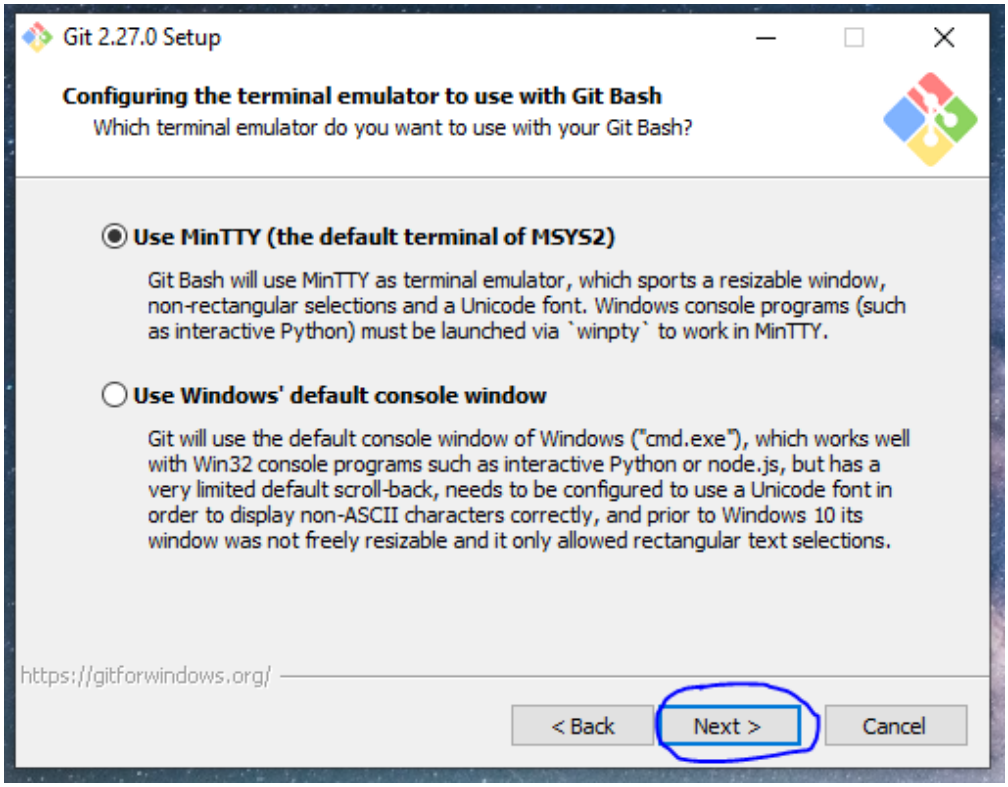
Any text editor can be used here, however, if you have never used a command line text editor just leave it to Vim for now. We will cover text editors later.

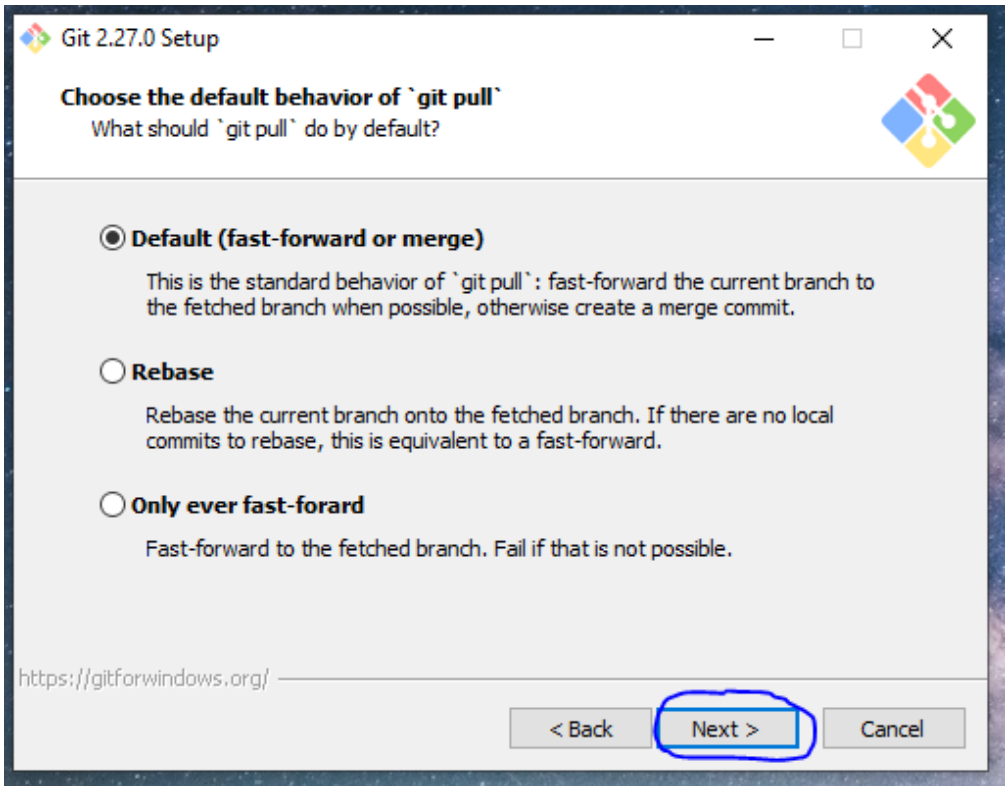


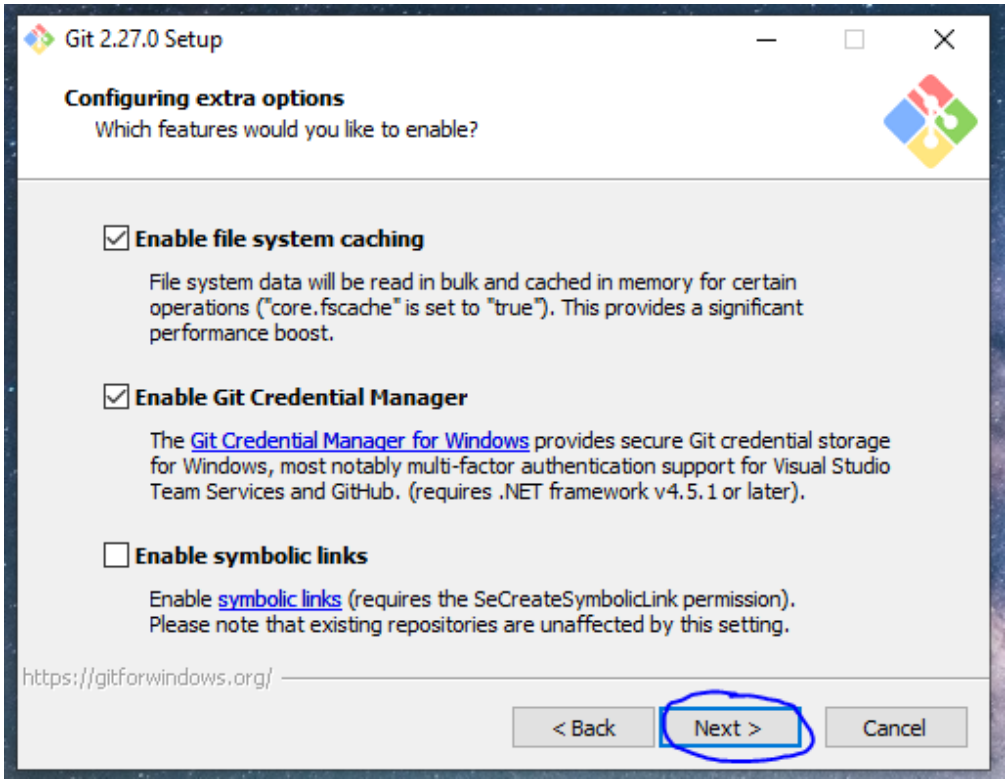
Either option 2 or 3 is ok. It is recommend to use option 2, however, if you want unix-commands to be avalible in native command prompts (like MS-DOS or powershell) you can select option 3.

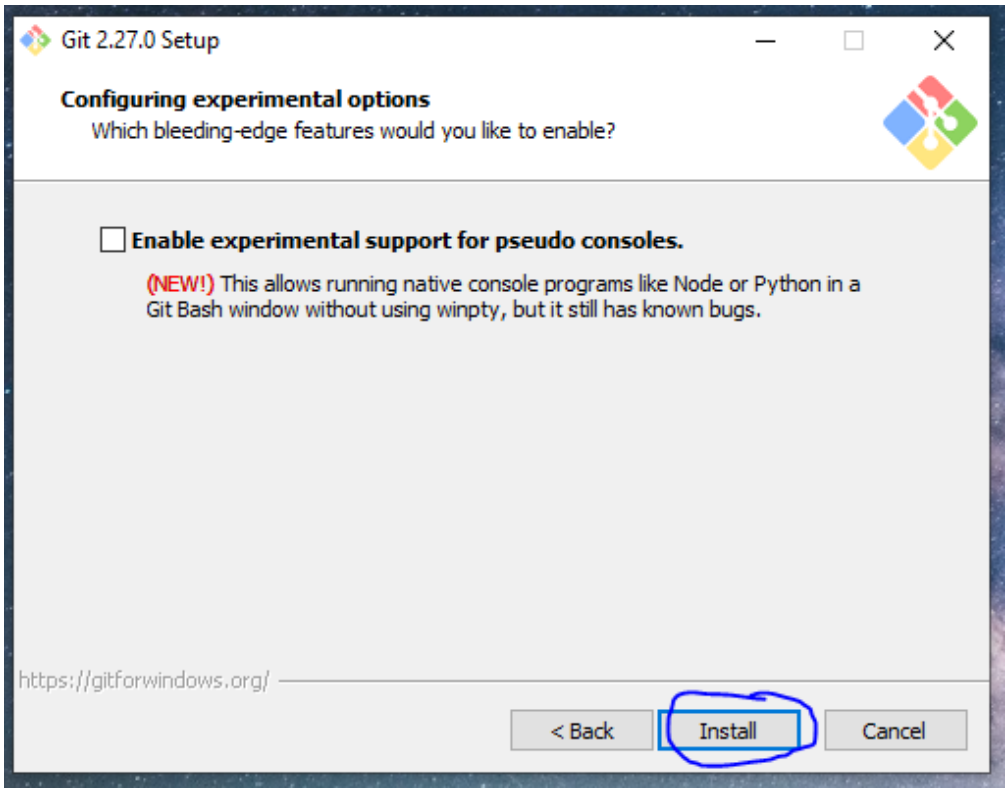

Git 2.27.0 Setup
Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?
Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.
Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates
distributed e.g. via Active Directory Domain Services.
https://gitforwindows.org/
< Back
Next >
Cancel

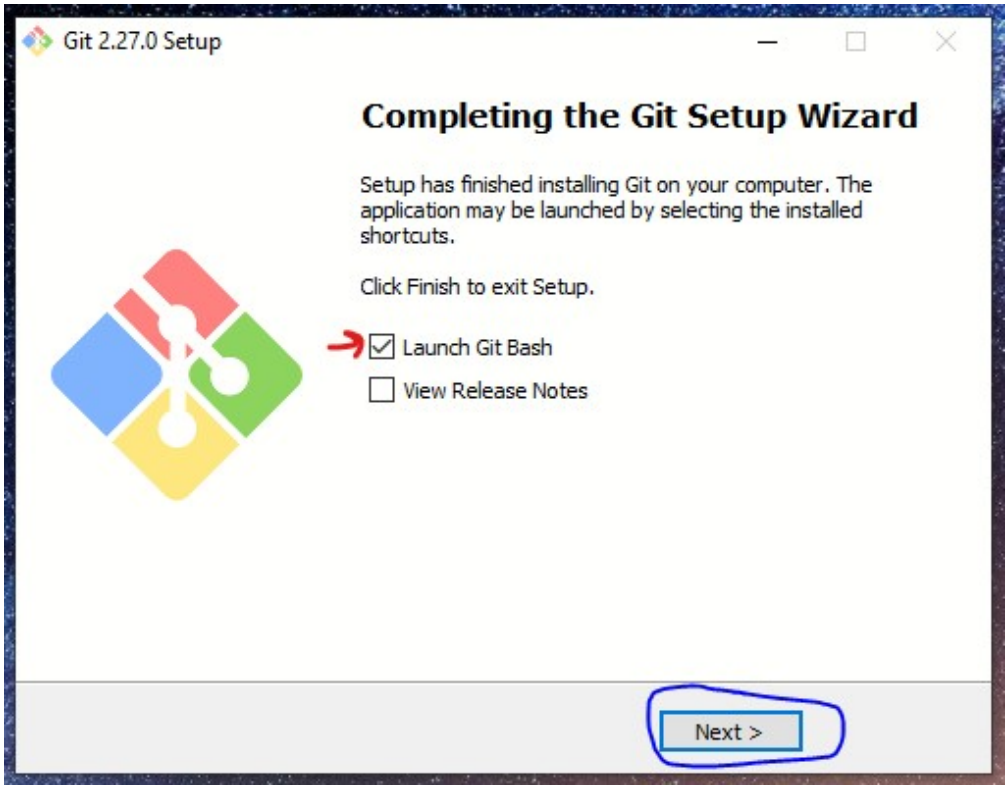

Git 2.27.0 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing
text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When
committing text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing
text files. Choosing this option is not recommended for cross-platform
projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.27.0 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.27.0 Setup
Choose the default behavior of `git pull`
What should `git pull` do by default?
Default (fast-forward or merge)
This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.
Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.
Only ever fast-forard
Fast-forward to the fetched branch. Fail if that is not possible.
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.27.0 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable Git Credential Manager
The Git Credential Manager for Windows provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or later).
Enable symbolic links
Enable symbolic links (requires the SeCreateSymbolicLink permission).
Please note that existing repositories are unaffected by this setting.
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.27.0 Setup
Configuring experimental options
Which bleeding-edge features would you like to enable?
Enable experimental support for pseudo consoles.
(NEW!) This allows running native console programs like Node or Python in a Git Bash window without using winpty, but it still has known bugs.
https://gitforwindows.org/
< Back
Install
Cancel

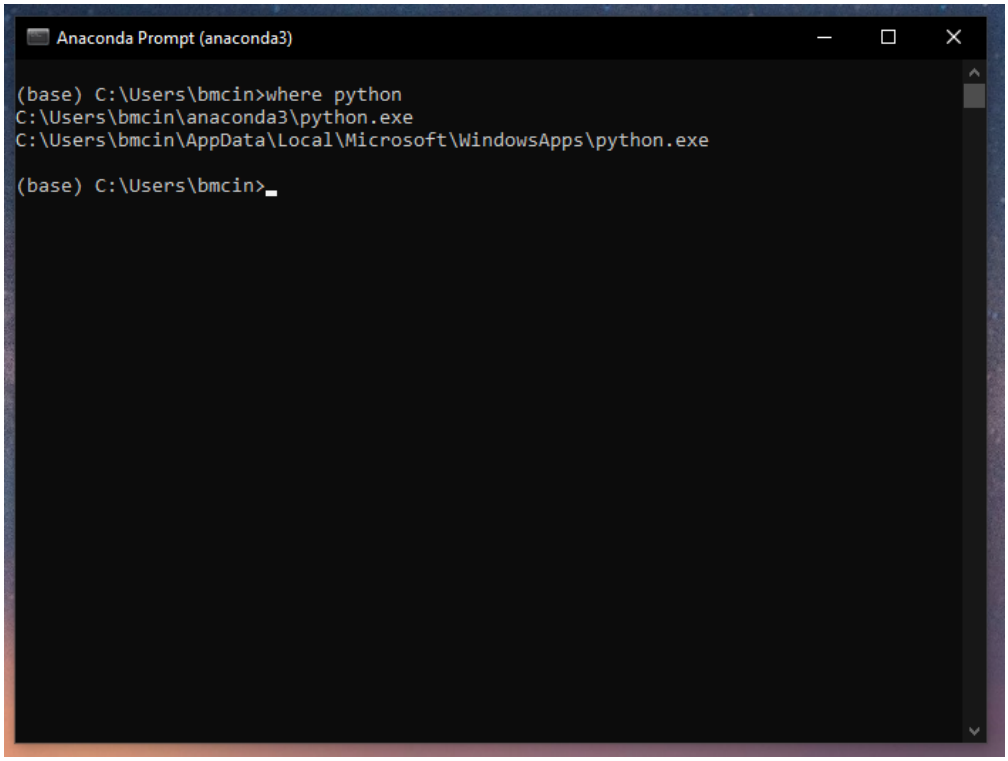


Make sure to check the box that says `Launch Git Bash` . This is helpful for the next step. If you accidently do not check the box, you can always press `Windows-key` and search for Git Bash

Adding Anaconda functionality to Git-Bash

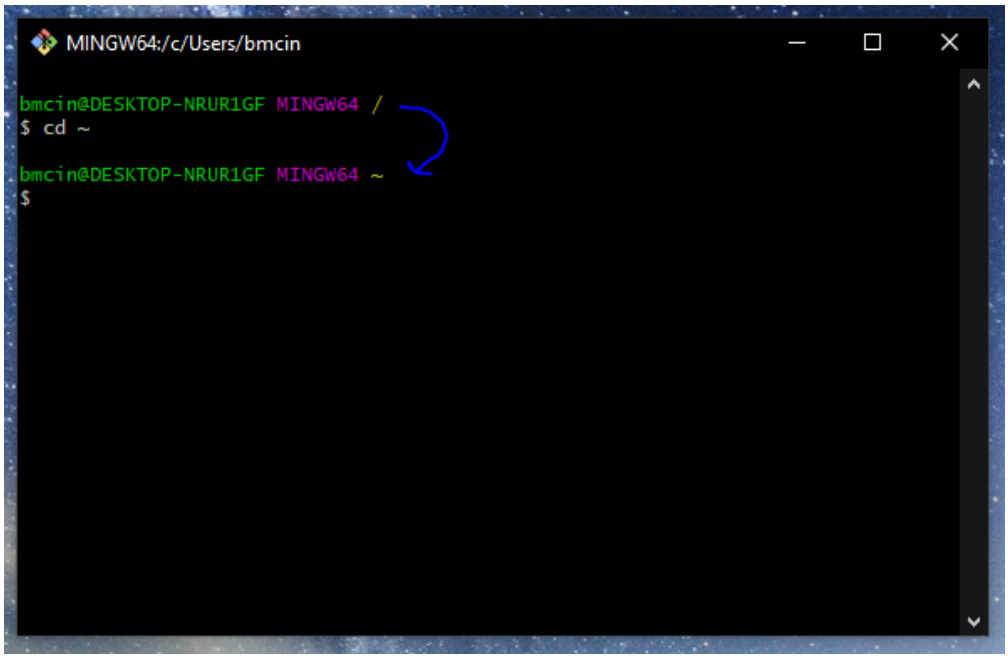
Next we are going to add Anaconda's python to our Git-bash terminal. This is crucial as this is what will allow us to run python from the terminal, and install new packages into our Andacona's python. We will first add Andaconda to our PATH, and then "create an alias" to run python from Git-bash.

1. Find your Anaconda prompt (If needed, see above for screenshots). To do this, on the keyboard press `Windows-key` or simply use the search bar on the taskbar if it is visible. Search `Anaconda Prompt` and click on the search result.
2. Type `where python` and press enter.

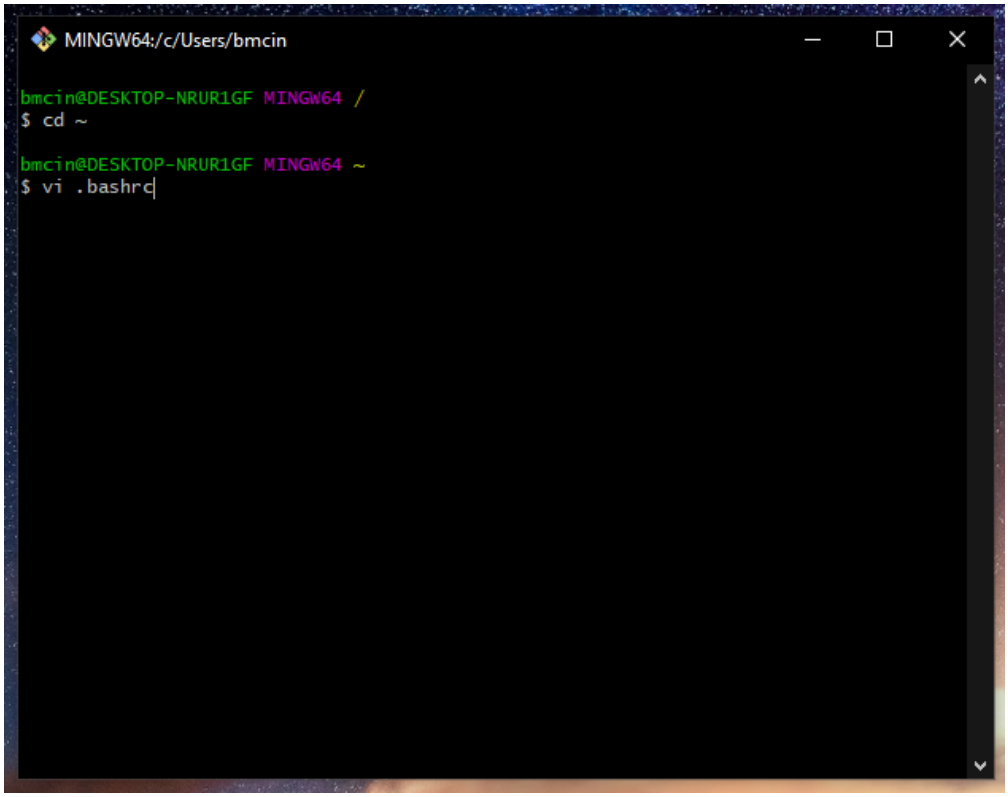


Notice that there may be two locations. We are looking for the location that has `anaconda3` in the location. Keep this window open while we switch over to using the Git Bash terminal.

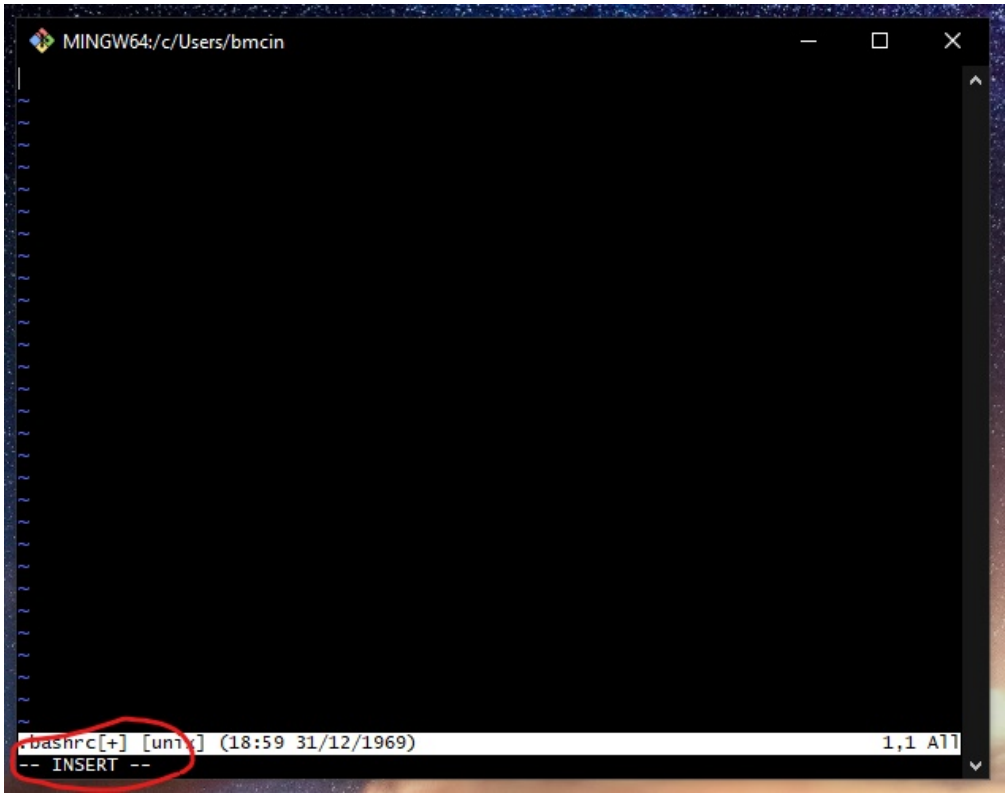
3. Using Git Bash (which should be open from our last step), navigate to what is known as your home directory by typing `cd ~` in the terminal. (Notice how the symbols changed from `/` known as the "root" directory to `~` which is our "home" directory) (It is also possible you may load into your home directory automatically as well)



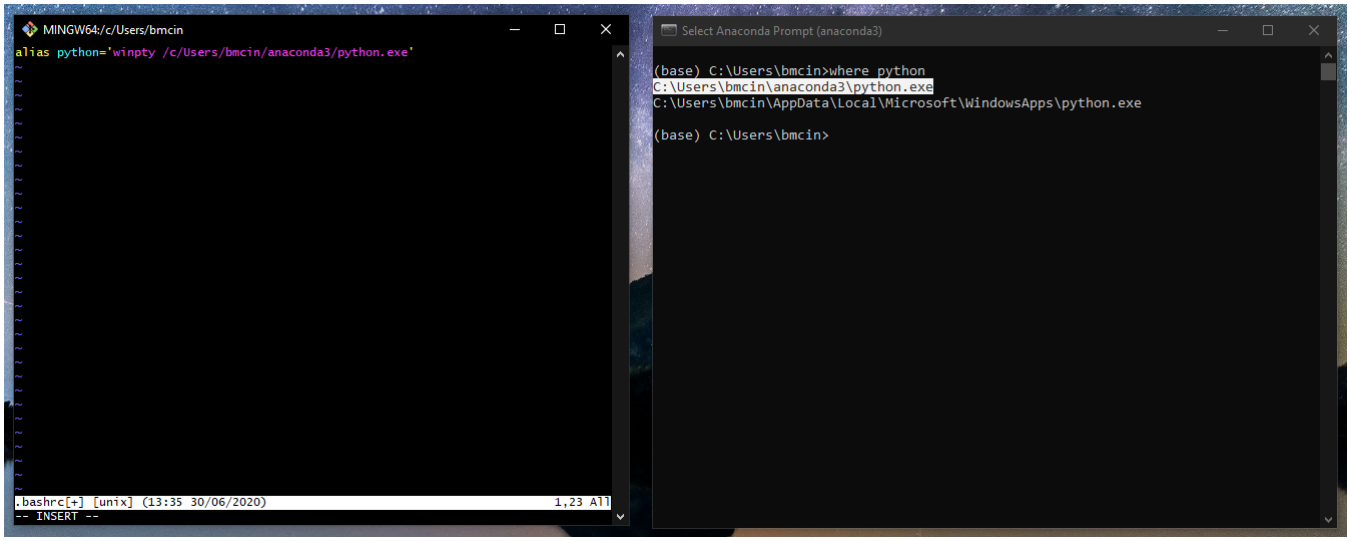
4. Using a text editor, the `.bashrc` file will need to modified to include the path to Anconda's python using the `alias` feature. It is most likely that you will not already have the `.bashrc` so the file will need to be created. (For reference the `.bashrc` file is a file that contains settings that is ran everytime the terminal is started)



Type `vi .bashrc` into the Git Bash terminal. This will open up/create the `.bashrc` with a command line text editor called Vim.



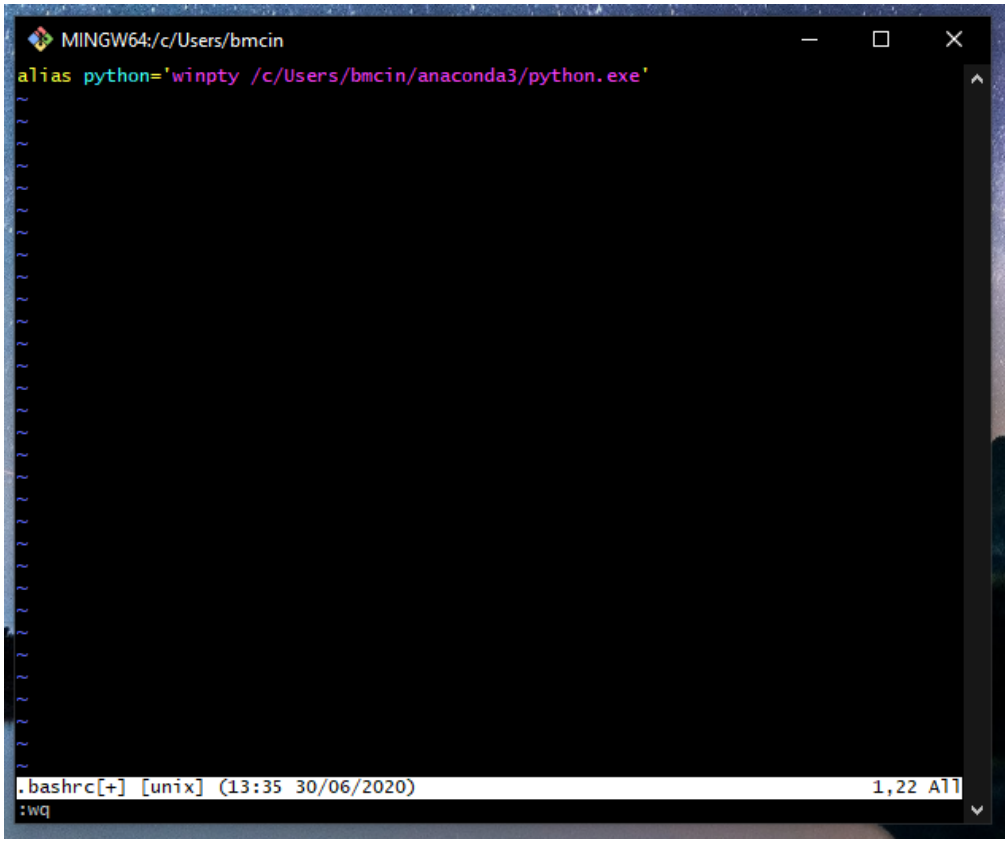
Press (lowercase) `i`. You will notice that this bottom left corner will go from *blank* to saying `-- INSERT --`. This changed the mode from "Command Mode" to "Insert Mode". Command Mode is where you can run commands such as save, exit, and the like. Insert mode is where you can type into the file and edit its contents.



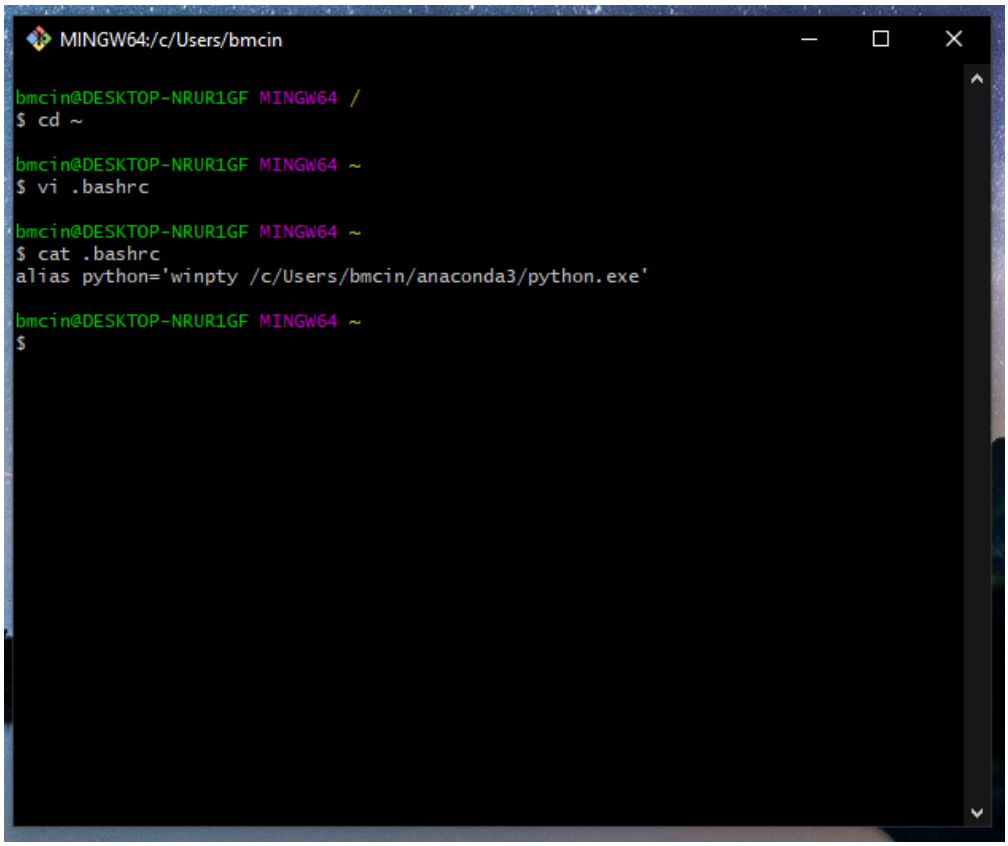
Now we will use the location found when using `where python` in Anaconda Prompt (there may be two locations, use the one with `Anconda3` in the location path) to create an alias for running python from the terminal. The general format is `alias python='winpty [PATH FROM "WHERE PYTHON"]'` .

```
    To make this a little easier would be to copy and paste from the Anconda P
rompt. First we need to type `alias python='winpty ` into the Git Bash termina
l.
    Then, to copy the "Anaconda3" python path from Anaconda Prompt, use your m
ouse to *left-click* and drag over the text in Anaconda Prompt to highlight it,
and then *right-click* on the highlighted text. This should make the highlighti
ng of the text disappear. This now means you have copied the text you just high
lighted.
    Go over to the Git Bash terminal and *right-click* into the terminal. Now
 there should be an option to `Paste`, click that. Then make sure to type ``' a
t the end of what you just copied.
    Next we are going to need to edit the location path from windows notation,
to unix-like notation. To do this use the arrow-keys to move along the text, an
d change all back-slashes `\` to `/` forward-slashes. Then change `C:` to lower
case `/c`. Then you should have the correct alias.

    **(If you cannot get this method to work, simply type out the above alias
 by hand)**
```

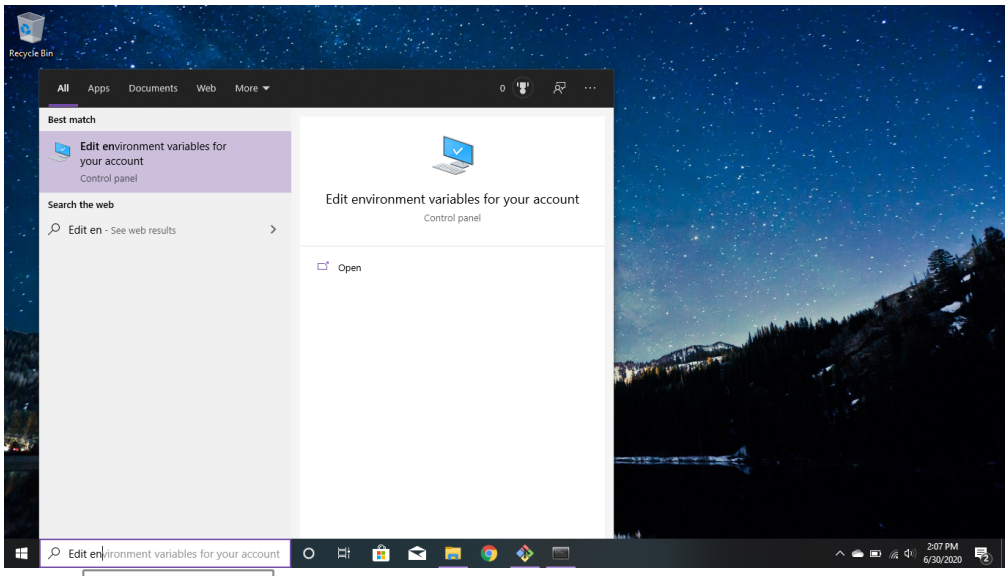


To save and close the file we will need to return back to the "Command Mode" to do this press the `Esc` on your keyboard. You will notice the bottom left-hand corner will go from `-- INSERT --` to *blank*. Then type on your keyboard `:wq` and press enter. This is the command to save (or *write* to the disk `w`) and to quit the text editor (`q`).

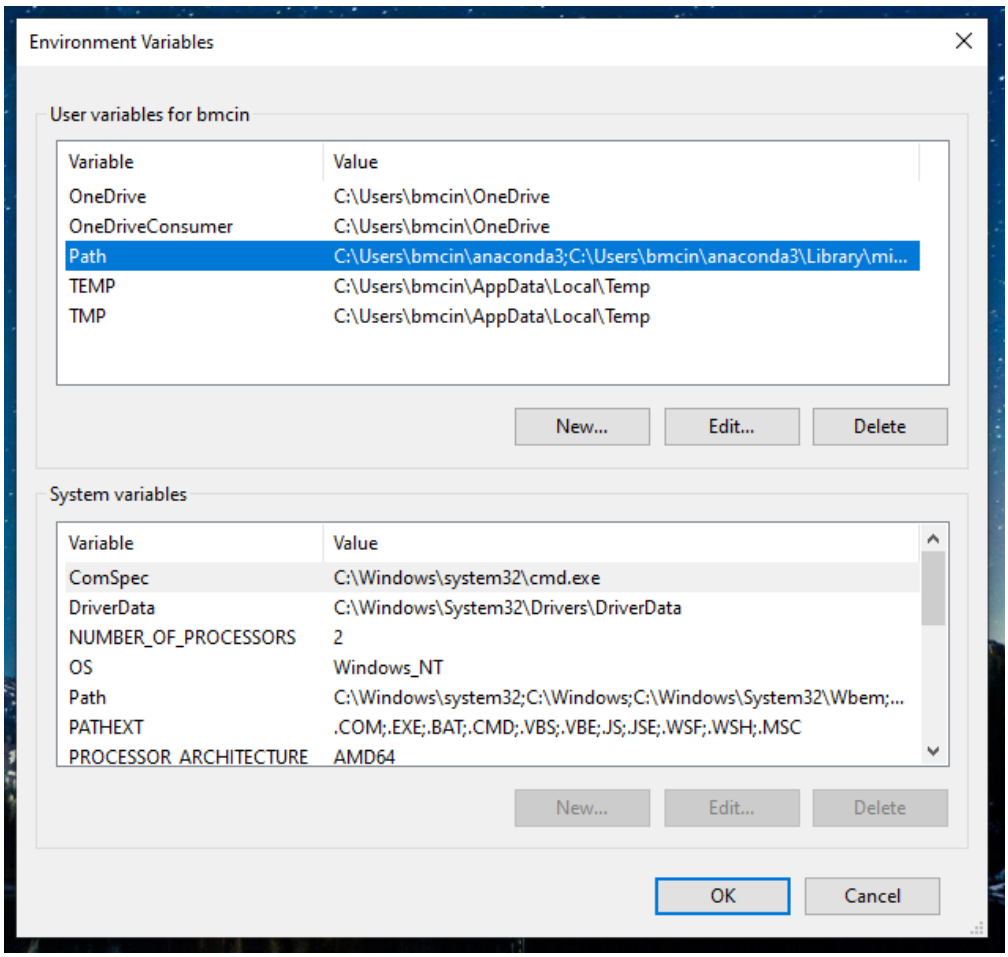


Then to check and make sure you entered the alias write, run the the command `cat .bashrc`. The `cat` command "prints" to your terminal all of the file contents. In our case this is contents of our `.bashrc` file. Keep Git Bash open for later steps.

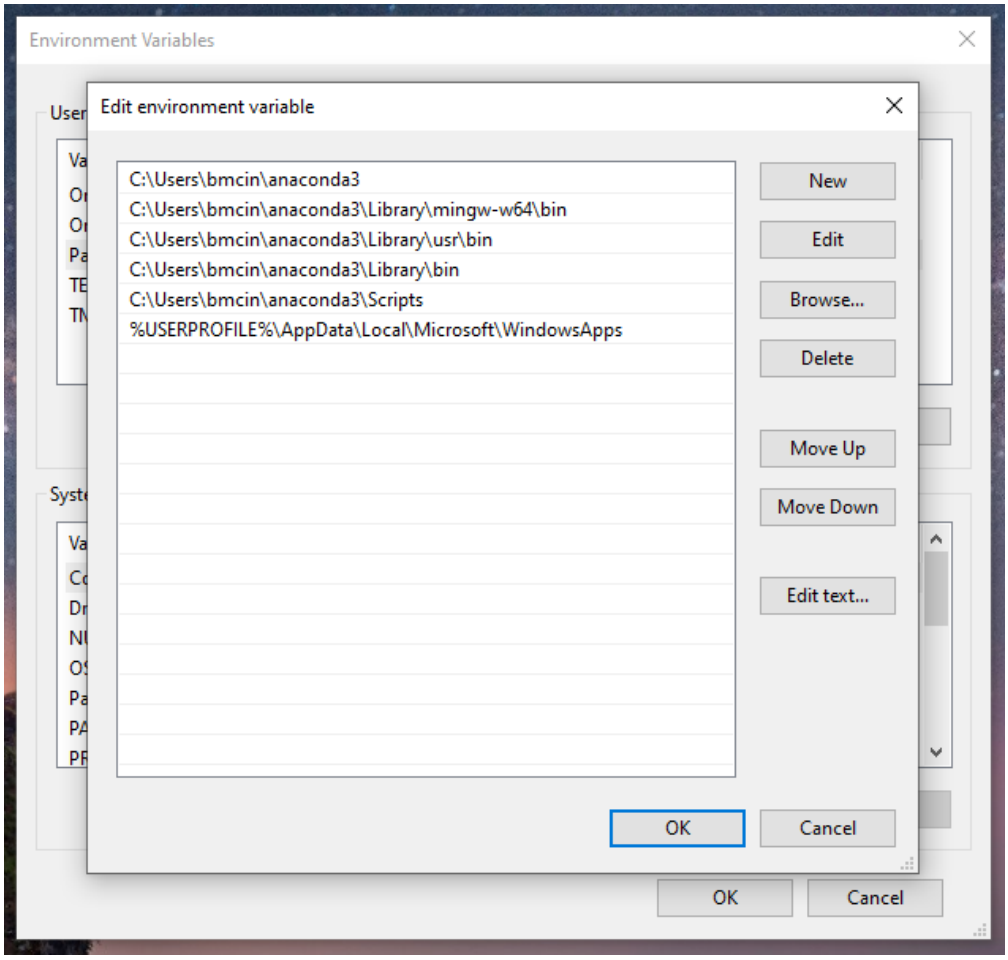
5. The PATH varibles in windows may need to altered to include the PATH to our Anaconda Installation. This will make sure to link Anaconda commands with the Git Bash terminal.



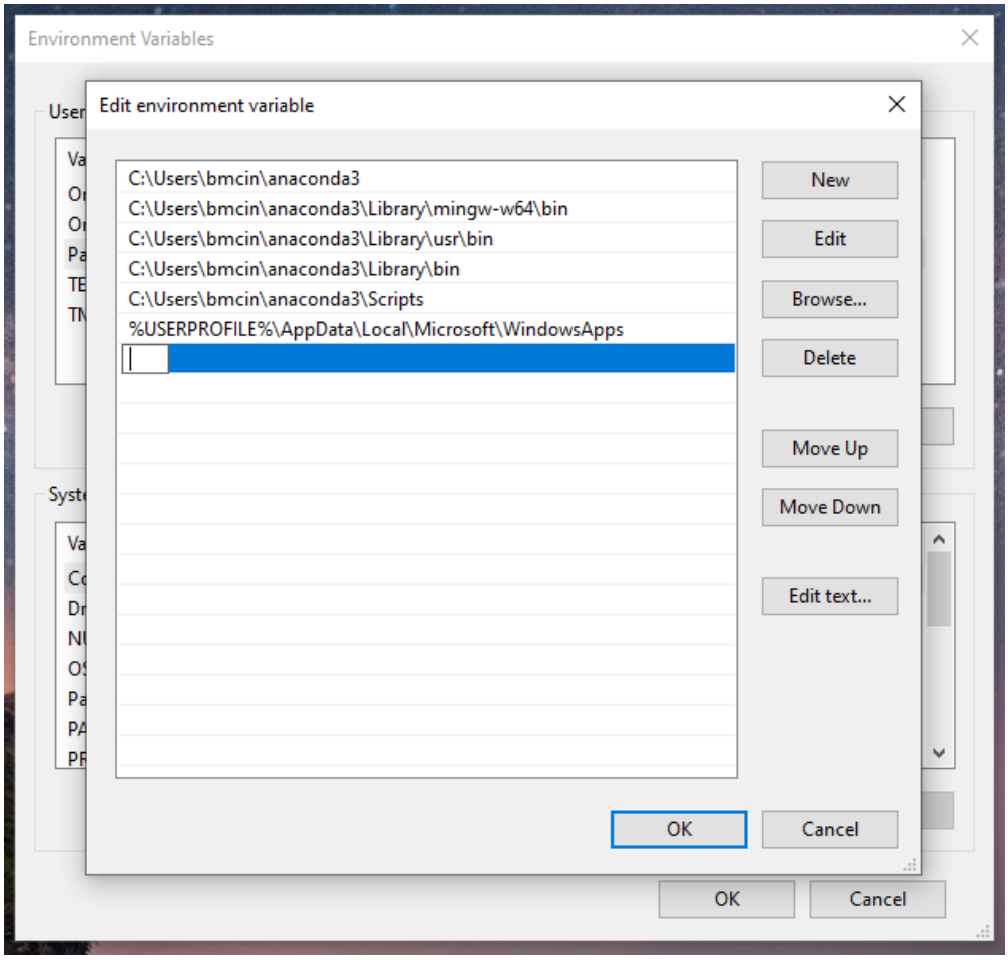
On keyboard press `Windows-key` or simply use the search bar on the taskbar if it is visible. Type in `Edit environment varibles for your account`. Click on the search result.



Under the first section "User variables for [YOUR USERNAME]" double click on the line that has `Path` under the `Varaible` column.



Here we can check and see all of the PATH variables for our terminal. You may notice Paths to Anaconda3 already entered in (except instead of my username "bmcin" you will have your own username). *If you have all of the Anaconda3 paths in this picture* (minus the one that starts with `%USERPROFILE%`) you are fine and can close out of these windows and skip to step 6. *If you do not have any/only partial of the Anaconda3 paths* (Again, minus the one that starts with `%USERPROFILE%`) then you will need to add these paths. The paths may vary by system if you saved Anaconda3 not in the default spot the Anaconda Installer suggested. Fortunately, we can use the path from the `where python` command we ran on the anaconda prompt in step 2 to get the paths we need.

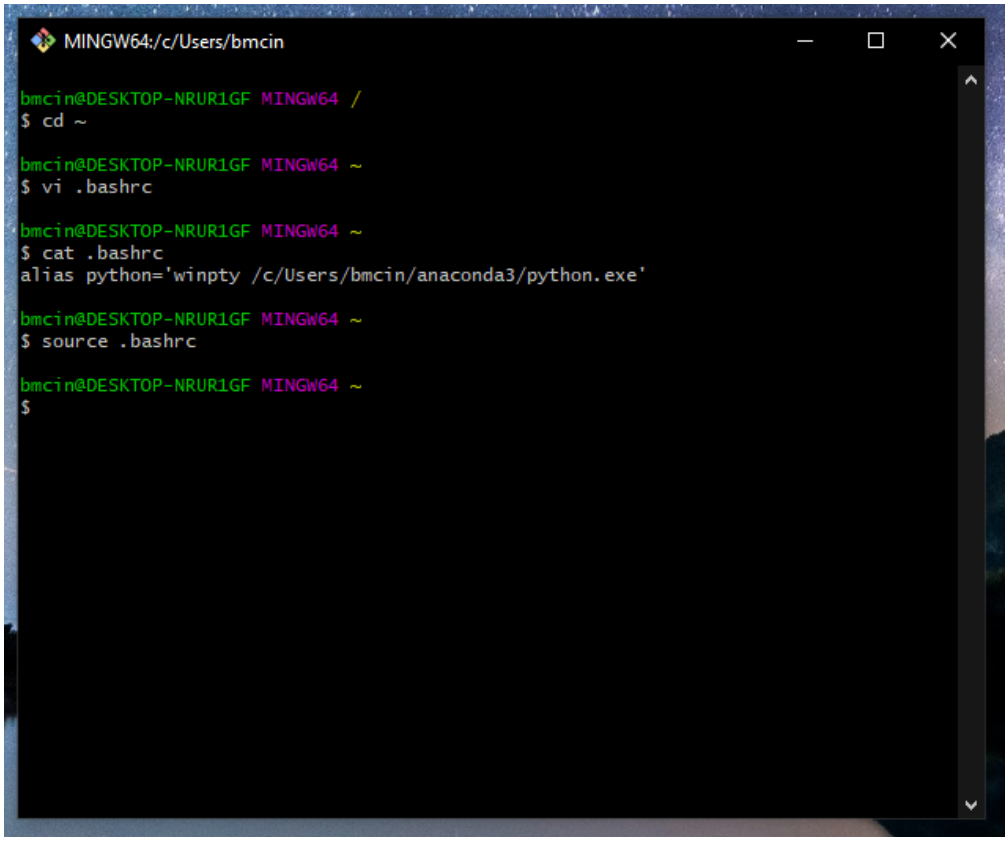


You will notice that the path from `where python` returns a path that has "Anaconda3" in the path. `C:\Users\bmcin\anaconda3\python.exe`. What you will want to do is copy everything up to "anaconda3" (`C:\Users\bmcin\anaconda3`). Then press the `New` button and then paste the path in there. Then add on whatever else is necessary to complete the path. You will need each of the following paths.*Note: [ANACONDA PATH] is the path you copied.*

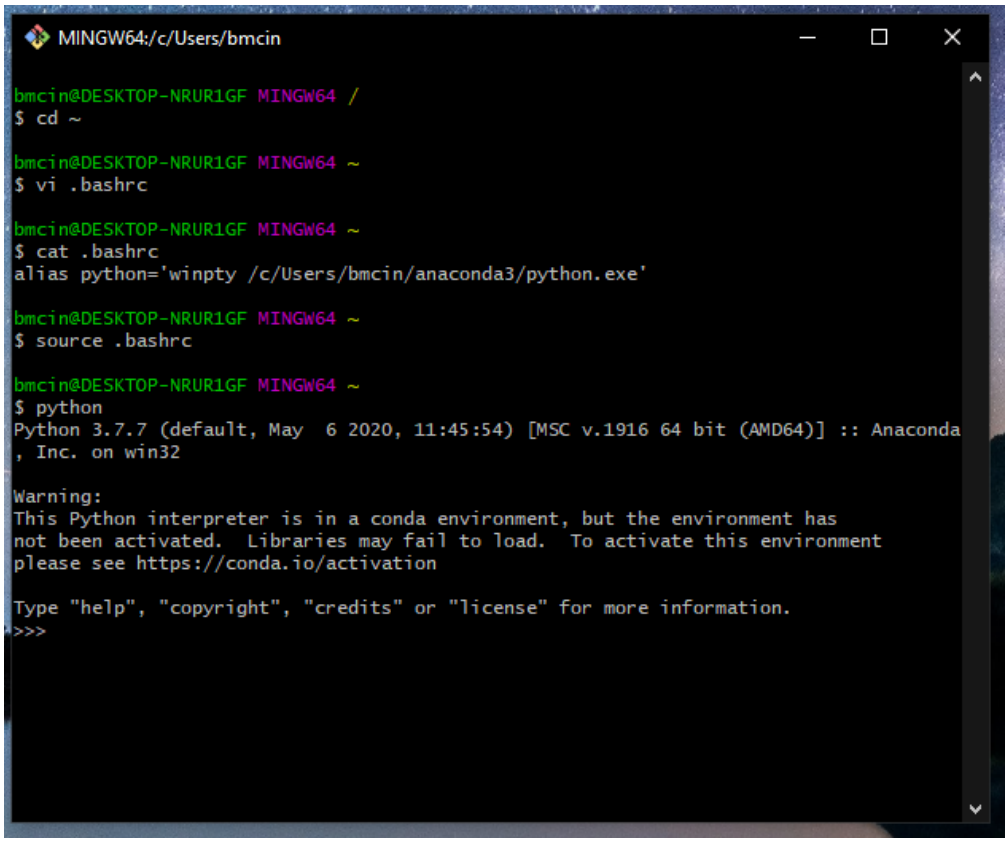
- `[ANACONDA PATH]`
- `[ANACONDA PATH]\Scripts`
- `[ANACONDA PATH]\Library\bin`
- `[ANACONDA PATH]\Library\usr\bin`
- `[ANACONDA PATH]\Library\mingw-w64\bin`

Once finished adding the paths, press `OK` on the first window to close it, and press `OK` again to close the second window.

6. To finish this out we will need to "run" the `.bashrc` file to apply our changes, and then check to make sure we are all set.



In the Git Bash terminal, type and enter `source .bashrc`. This will apply the changes we made earlier to the file. If you encounter an error in this part or the next part, it is most likely you did not close your `'` in your alias, you did not put a `/` in front of the `c`, or you spelled something wrong. Follow step 4 again.



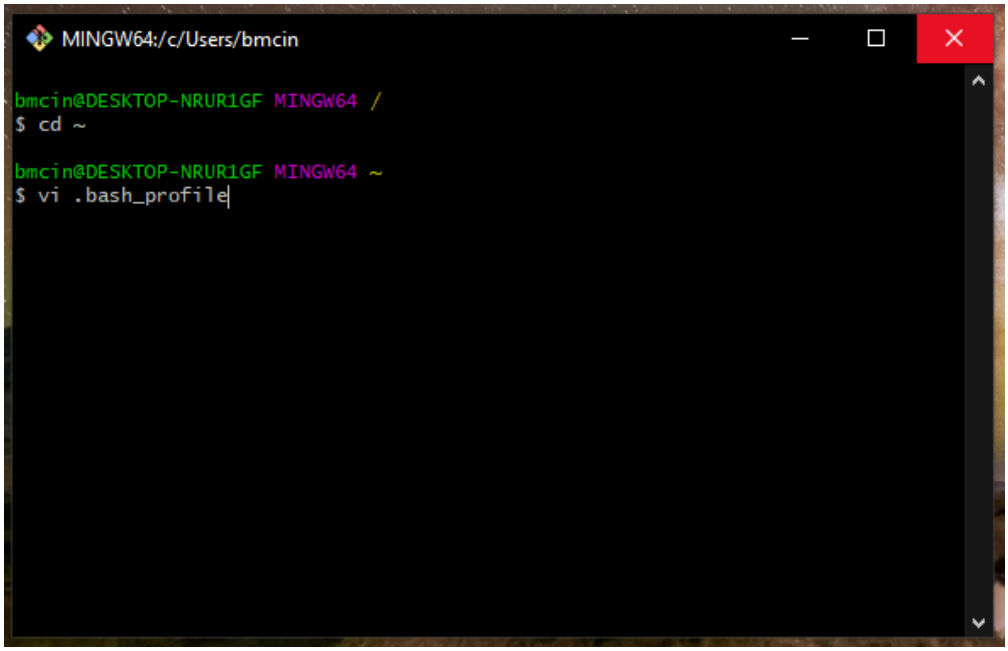
To test if everything was successful, in Git Bash type `python` and press enter, if all is well the terminal should appear as it does above. Currently you will be running the "python interpreter" which is a program that runs in your terminal.

If successful, and the terminal looks like it does above use the key combination `Ctrl` + `d` to quit out of the python interpreter. Then type in `exit` and hit enter, this will close the terminal.
If unsucessful, and nothing appears or there is an error you can simply just simply close the terminal with the `X` in the top right corner.

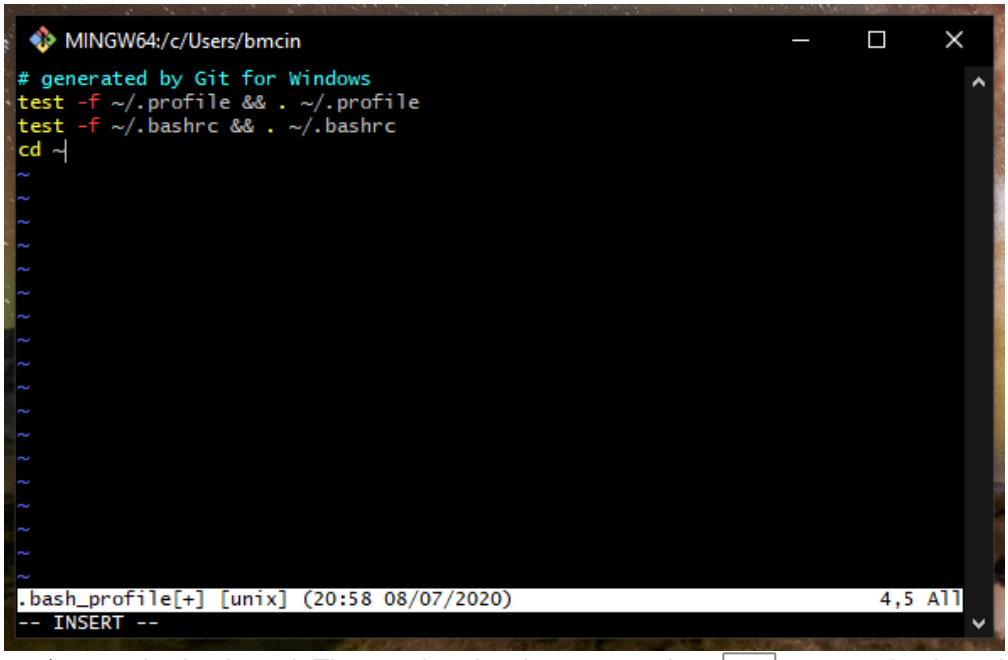
Changing startup location to Home directory

This next step is to ensure that when Git Bash starts up, you will load into the home directory (`~`) instead of the root directory (`/`). This will make it less likely for the file and folders in the root directory to be harmed, and proper file storage etiquette be followed.

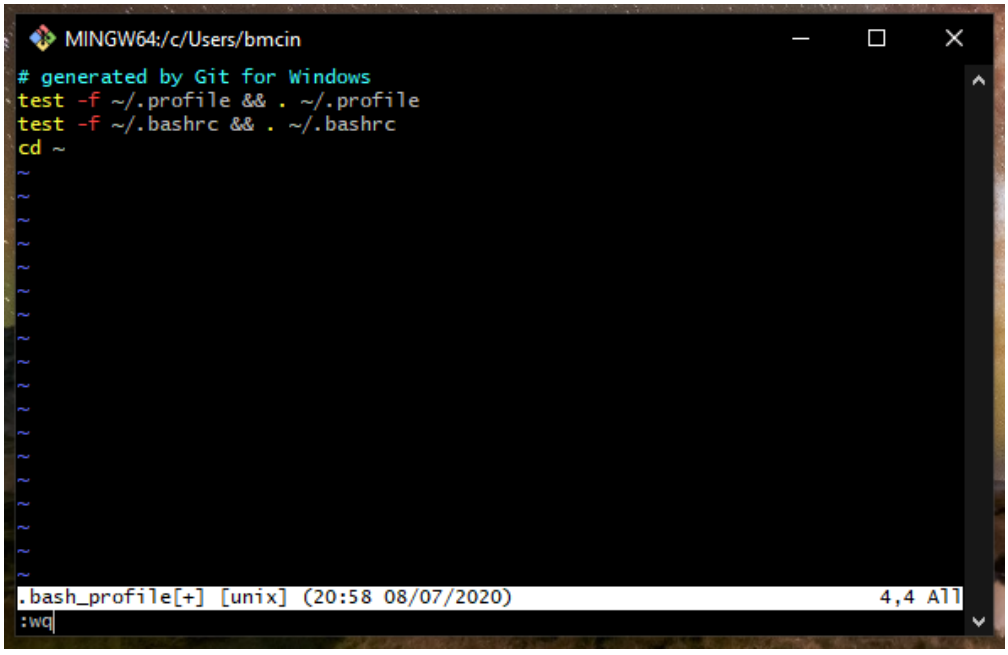
1. Close down the Git Bash terminal if open, and restart. You may notice a Warning in the terminal saying `.bash_profile` and other files are missing, you can saftely ignore this.
2. Navigate to the home directory and open `.bash_profile` in a text editor.



In the terminal type `cd ~` and press `Enter` to change your directory to the Home directory. Then type `vi .bash_profile` and press `Enter` . This will open the `.bash_profile` file that is used to run the `.bashrc` file that we edited earlier.
3. Add a line that changes the directory to home

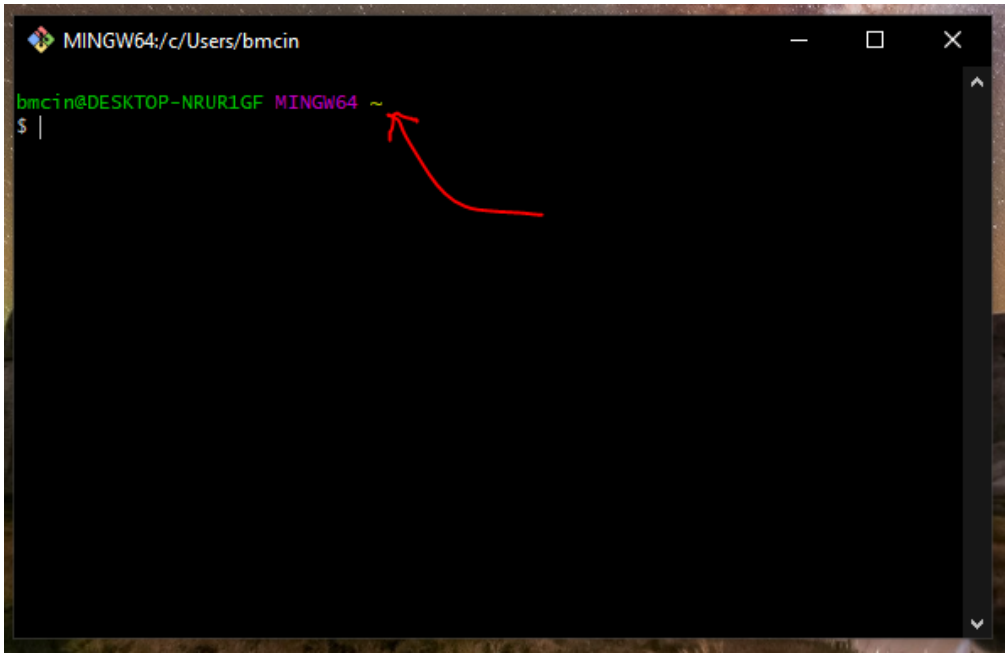


Press (lowercase) `i` on the keyboard. Then, using the down-arrow key [↓] on your keyboard move to the last possible line you can. You will notice the screen will flash if you are try pressing the down-arrow while on the last line. Then, using the right-arrow key [→] on your keyboard move to the end of the line. Again, you will notice the screen will flash if you are try pressing the right-arrow while at the end of the line. Then, on the keyboard press `Enter` and this should make a newline. Type `cd ~`. If done successfully, the file should look like this. *Note: if any of the lines above the `cd ~` were altered, this could break things. Make sure you did not alter any of those lines before continuing. If you did happen to mess them up, press `Esc` on the keyboard and then type `:q!` and press `Enter` this will not save any of the changes you made, and you can try editing it again.*



To save and close the file we will need to return back to the "Command Mode" to do this press the `Esc` on your keyboard. You will notice the bottom left-hand corner will go from `-- INSERT --` to *blank*. Then type on your keyboard `:wq` and press `Enter`. This is the command to save (or *write* to the disk `w`) and to quit the text editor (`q`).

4. Test to make sure you start up in the home directory.



First close down the Terminal. Then open it back up. If all went well you should start up in the Home directory. You can tell by looking here. If it has `~` then we are all-set.

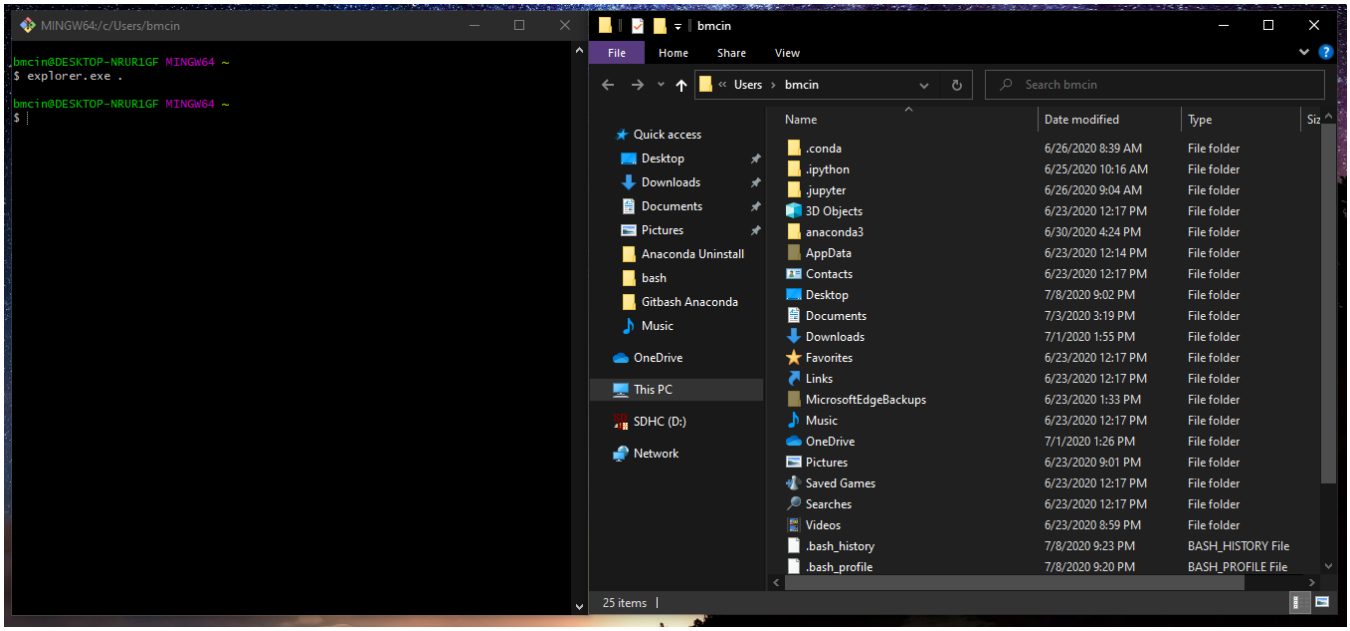
Congrats you are done with setting up a terminal!
If at anypoint anything got too confusing or you were not successful, please let your instructor know during class or through email and they can work with you to resolve any issues.

If you want to learn some optional tools and shortcuts continue to the next section. However, if you want to skip the optional section, click here.

(Optional) Helpful Commands and Shortcuts

View where Home directory is
When starting up the Git Bash terminal it is important to know where you are in the system. In this course we wil be making and using directories/folders. Making sure we know where we are at is crucial to this process. When Opening Git Bash it is likely you will start in one of two places. Either the root directory signified by `/` or your home directory signified by `~`. If you followed the instructions in the `Changing startup location to Home directory` you should always load into the Home (`~`) directory. In this course, we will never need to do anything with the root (`/`) directory as this is a where important files for Git Bash live. To see where you start up in a more familiar program, we will be utilizng the Windows file explorer command inside of the Git Bash terminal.

1. Have the terminal open and all of the above sections completed
2. From the terminal open up file explorer.



In the terminal, type `explorer.exe .` and press `Enter`. This should open up the Windows File Explorer in the current directory (this is denoted by `.` in the command `explorer.exe .`). Here you can navigate the folder and see all that you can access from your Home directory. Try to see if you can figure out how to navigate to this spot from other locations. This will be important if you need to open a file in programs such as `spyder`. If you run into issues in the future in trying to find this folder, make sure to as your Instructors or LA for help!

Useful Commands

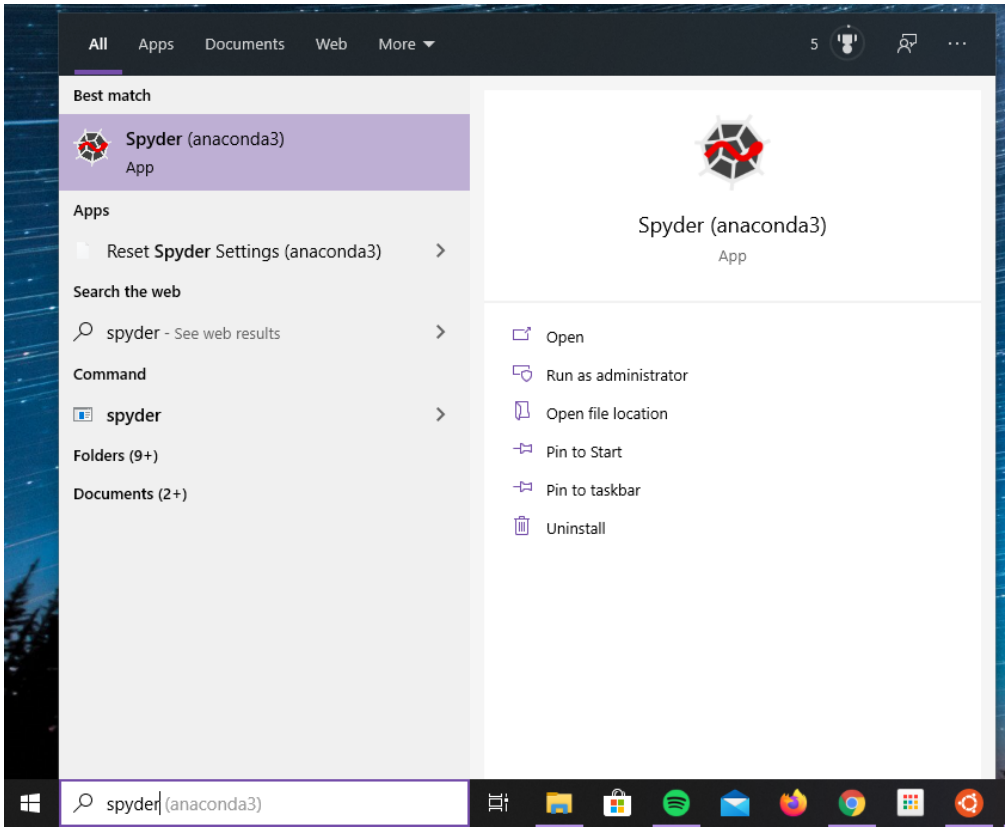
- `cd [DIRECTORY/FOLDER_NAME]` -- Will change the current directory your terminal is in to the folder specified.
- `cd ..` -- Will change the current directory your terminal is in to the the parent directory/folder of the current directory/folder.
- `explorer.exe .` -> Will open up the current directory your terminal is in, in Windows file explorer.
- `notepad.exe [FILE_NAME]` -> Will open up the specified file with the Windows notepad program
- `clear` -> If you want to "clean" up the terminal to clear past commands from view, you type this command.
- `exit` -> This will close out the terminal
- `cat [FILE_NAME]` -> This will print all the contents of the specified text file into the terminal for easy viewing.
- `jupyter notebook` -> This will open up Jupyter Notebook in your current directory.

Using Spyder (Text Editor/Python IDE)

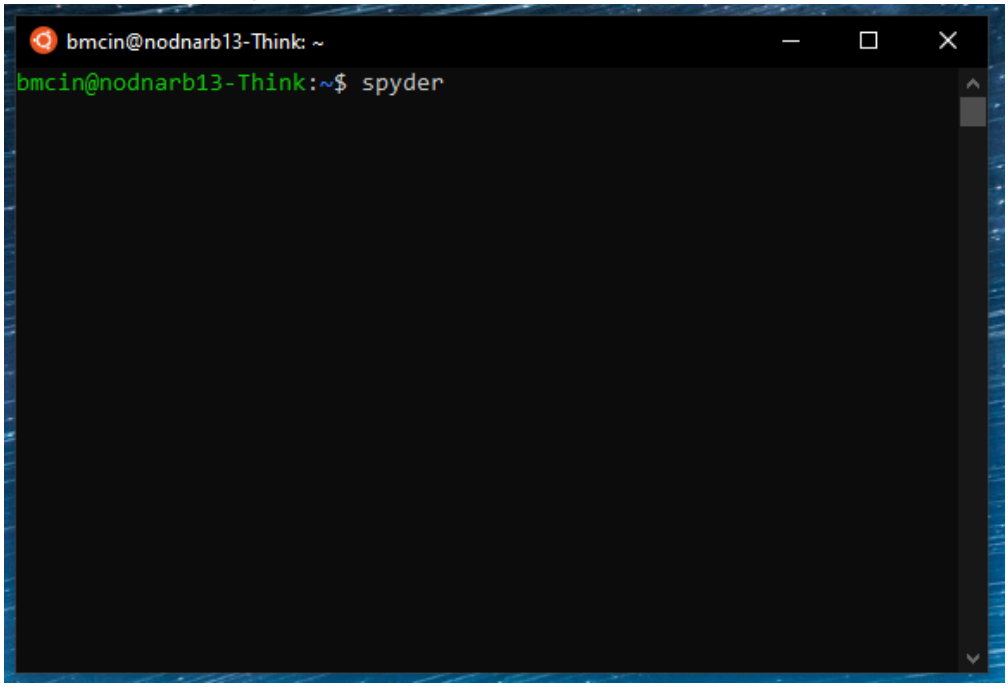
Spyder is a program apart of the Anaconda distribution. It can be used to edit text files, as well as run Python code. Since it is built into Anaconda, this will be our deafult way of editing text files. There are other solutions out there such as Atom, Sublime, and many others. You are more than welcome to explore these other options, however, we will use this one as it is already pre-installed through the Anaconda.

Spyder can easily be started from one or two ways

1. Searching for it on Windows by either using the search bar in the bottom left hand corner (if you have it) or clicking on the windows button and searching for `spyder`.

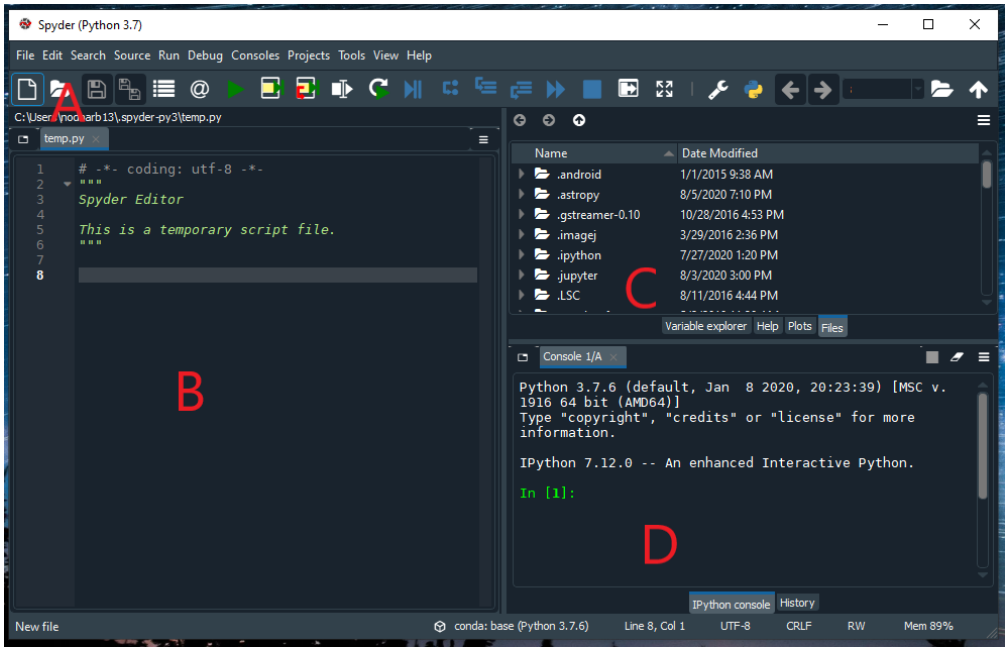


2. Using the terminal to start the program.

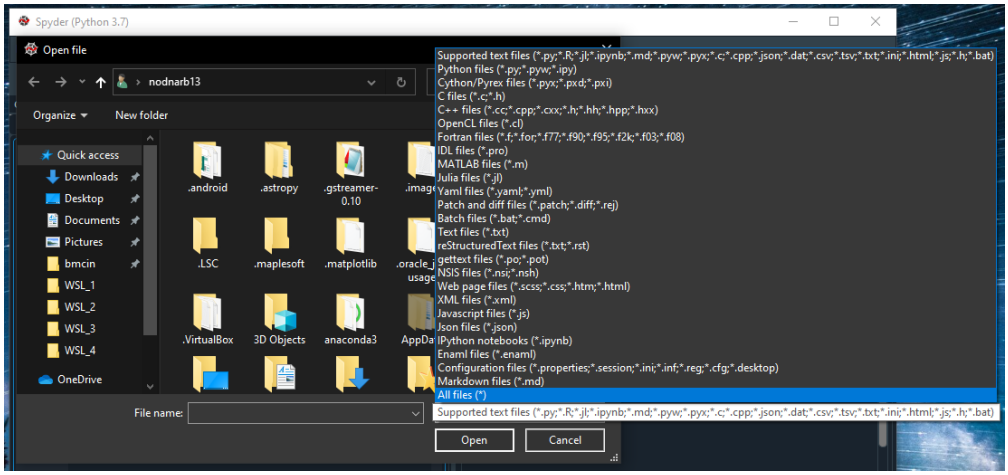


Open up your terminal (`Git-Bash` or `Ubuntu`) and type `spyder` and press enter. *Note: You can also open up files directly into spyder this way as well, you just need to include the path to the file. For example `spyder [FILE]` or `spyder [FOLDER/FILE]` .*

Once Spyder is open you should see something that looks like below. Follow the letters to learn more about these otpions



- **A.** This option you can use to open files. This is the alternative to opening files then opening them through the command line. Once you click on this icon another folder will open up.



  Notice how in the photo the options to the right of the `File name:` is clicked. Make sure before trying to navigate to the file you want to open, you click on the `All files (*)` option to be able to see/open any file. *Note: If you need to open a file that is in your home directory, and if you followed the optional guide above, you can quickly get there by using the Quick Access bar*
- **B.** This is where the file that is opened will display.
- **C.** This is place where you can look at files, or assigned varaibles from running scripts (will learn more about this later). You can change what you see by clicking the tabs.
- **D.** This is an `iPython` console where you can run python commands and test out anything you want (will come more in handy when writing python scripts)

MSU's JupyterHub Interface

From time to time, you might run into issues with your computer. When this happens, you should use the web-hosted JupyterHub server managed by MSU. It creates a virtual environment that allows you to run simple commands and host Jupyter notebooks. To make sure that you have access to this backup option, follow the

directions below. Note that there are extra steps involved that require that you upload and download you Jupyter notebooks to and from JupyterHub in order to turn them in on D2L.

Connecting to the engineering JupyterHub server

Every student enrolled in this class will be given an engineering computing account. If this is your first time using your Engineering account you will need to activate the account by going to the following website:

https://www.egr.msu.edu/decs/myaccount/?page=activate (https://www.egr.msu.edu/decs/myaccount/?page=activate)

Enter your MSU NetID. The initial password will be your APID with an @ on the end (example: A12345678@) and then you have to set a password that meets the requirements listed on the page. Verify the password. Then agree to the terms and Activate.

Once your account is activated you can access the classroom Jupyterhub server using the following instructions:

1. Open up a web browser and go to the following URL: https://jupyterhub.egr.msu.edu (https://jupyterhub.egr.msu.edu)
2. Type your engineering login name. This will be your MSU NetID.
3. Type your engineering password.

If everything is working properly you will see the main “Files” windows in the Jupyter interface.

If you ever end up working on your assignments using JupyterHub, the remaining directions should serve as a reference for how you can go about uploading and downloading Jupyter notebooks and turning them in.

Getting Jupyter notebook files into JupyterHub

IPython notebooks (also referred to as Jupyter notebooks) are files that end with the .ipynb extension. We will give you these files for all of your assignments, you will edit them and turn in the edited files in using the class Website.

You can download the ipynb assignment files from the class website (http://d2l.msu.edu (http://d2l.msu.edu)). Once you have an ipynb file you can load it into Jupyter using the “upload” button on the main “Files” tab in the JupyterHub web interface. Hitting this button will cause a file browser window to open. Just navigate to your ipynb file, select it and hit the open button.

Once you see your filename in the jupyter window you can just click on that name to start using that file.

Making a copy of Jupyter notebooks from JupyterHub and turning them in

When you are finished editing your IPython notebook and are ready to turn it in you will need to download the ipynb file from the JupyterHub interface.

1. With the notebook file open in Jupyter, go to the “File” menu, select the “Download as” menu option and then choose “iPython Notebook (.ipynb)”
2. Pick a place to save the file (The desktop is a good choice).
3. Make sure you make a copy of the .ipynb file for your own records.

4. Go to the Desire 2 Learn (http://d2l.msu.edu (http://d2l.msu.edu)) class website and upload the .ipynb file into the assignment folder.

NOTE: Video versions of these instructions are available on the CMSE YouTube channel here (https://www.youtube.com/watch?v=l7mhi4ww6tY) and here (https://www.youtube.com/watch?v=5WSQnGmz3IA).

Course Communication with Slack

We will be using Slack (http://slack.com (http://slack.com)) as our means of communicating about course content as the semester progresses. We believe that this will provide an excellent avenue to have discussions not only with course instructors, TAs, and LAs, but also between you and your fellow classmates. In order to join the Slack team that we've created for the course you should complete the following steps:

1. Go to https://cmse-courses.slack.com/signup (https://cmse-courses.slack.com/signup)
2. Once there, sign up using your @msu.edu email address. **Important**: When you create your account, use your MSU NetID as your user name. This will make it easier for your instructors to recognize you within the Slack channels.

Once you've joined the CMSE Courses Slack team, you'll need to **add yourself to two channels**. The first channel you should add yourself to is **"cmse202-f20-help"**. The second one you should add yourself to is **"cmse202-###-f20"** where "###" corresponds to the section of the course that you are enrolled in. To add yourself to these channels, click on "Channels" and search for the appropriate channel.

The "help" channel will be the place to go for any questions about assignments in the course or issues you're having with your computer or Python. We encourage you to help out other classmates when you can! The section-specific channel will be used by your instructor for important messages relevant only to your section of the course.

If you want a more streamlined experince, feel free to download slack for desktop (https://slack.com/downloads/windows).

Slack usage rules

In order to ensure that Slack is a useful tool that does not become overly time-consuming for the course instructors, TAs, or LAs, we have a list of rules for how we expect you to use Slack. They are:

1. Before you ask a question, be sure to check the other section channels to see if the question has already been answered.
2. The Slack team is primarily for you, the students, so help each other.
3. The TAs and LAs will monitor the channels, but will defer to the students to work through things. They will only enter a conversation if students are going down the wrong path and/or there are too few other students involved. However, you should not expect that the TAs or LAs will always be available. The TAs and LAs will spend a limited amount of time "logged in" to Slack and we ask that you be respectful of their time.
4. Slack is meant to be used to help you when you are stuck with a minor issue. If you are having major issues or trouble understanding the concept, go to office/helproom hours. Office/helproom hours are meant for more in-depth discussions of course content.
5. Course instructors will rarely check Slack, only to examine progress. While they may offer help, do not rely on it. Instructors will not respond to the same student twice within a 30 minute time interval.

6. Only in rare cases should you contact an instructor through a private channel. But, if you are struggling, feel free to use this option.
7. **Do not** post your solutions to out-of-class assignments directly into Slack unless prompted by an instructor.
8. Be courteous to everyone on Slack. Students who are being rude or who are excessively posting might be banned from posting on the course Slack channel