



InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

# Setting up your dev environment for the program!

Module 2 - Use JPMorgan Chase frameworks and tools

**We know your first time using Python, or setting up a web development environment at work, might be daunting.**

Or feel like it uses technologies you haven't used before, or might feel like it takes too long.

# So to help you out we've created this step-by-step guide to setting up your computer for this task.

A lot of the things you do here, you will also do when you set yourself up at an in-office internship too. Look like an amazing hire when you breeze through dev environment setup!

With this guide, the approximate time to get a development environment working for you is **20 minutes**.

## To start, choose the application environment based on your device & current skill level

([Windows](#))

Setting up your dev environment for task  
2

([Mac](#))

Setting up your dev environment for task  
2

([Linux](#))

Setting up your dev environment for task  
2



InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

# Setting up your Mac for the JPMorgan Chase program

Module 2 - Use JPMorgan Chase frameworks and tools

## Local Setup (Mac)

- If your machine is running on Mac, follow this setup guide to get started.
- First you must have git installed in your system. Git is used by most programmers to collaborate with code in software projects. To do install, follow this [quick guide](#). You know you have installed successfully when you get a version output on your terminal by typing `git --version`:



```
insidesherpa — -bash — 127x34
InsideSherpas-MacBook-Pro:~ insidesherpa$ git --version
git version 2.20.1 (Apple Git-117)
InsideSherpas-MacBook-Pro:~ insidesherpa$
```

# Local Setup (Mac)

- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following commands on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-2.git
```

```
git clone https://github.com/insidesherpa/JPMC-tech-task-2-PY3.git
```

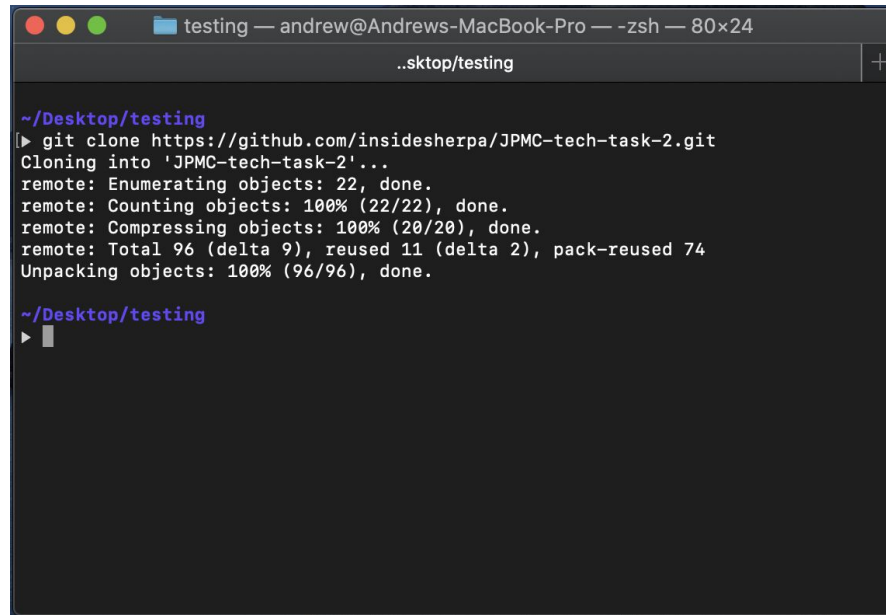
- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later

# Local Setup (Mac)

- You'll know you cloned successfully if you have the copy of the application code on your machine:

*note: the image here just does not contain the other repository but it should if you did the previous slides and execute the **ls** command. ``ls`` just lists the files/folders in the current directory*

*note: take note of the current directory your terminal is in because that is the location where you copied the repository. In the image here it's Desktop/testing*



```

testing — andrew@Andrews-MacBook-Pro — zsh — 80x24
..sktop/testing

~/Desktop/testing
└─▶ git clone https://github.com/insidesherpa/JPMC-tech-task-2.git
Cloning into 'JPMC-tech-task-2'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 96 (delta 9), reused 11 (delta 2), pack-reused 74
Unpacking objects: 100% (96/96), done.

~/Desktop/testing
└─▶ █
  
```



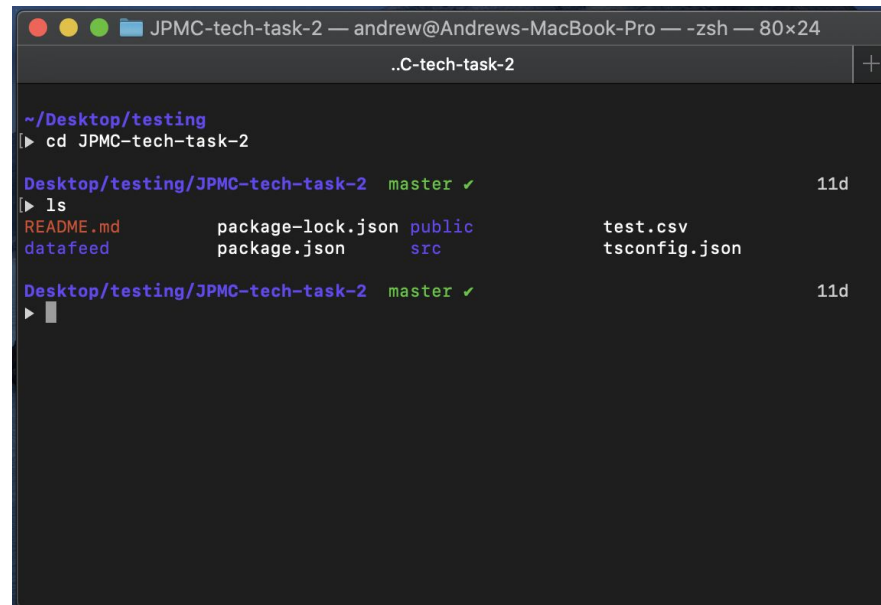
# Local Setup (Mac)

- To access the files inside from the terminal, just change directory by typing the following commands:

```
cd JPMC-tech-task-2
ls
```

*note: If you choose to work using python3 and your system has version python3.x as default instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above); **cd** changes the directory your terminal is in. Check [this](#) for more info on **cd***

```
cd JPMC-tech-task-2-py3
```



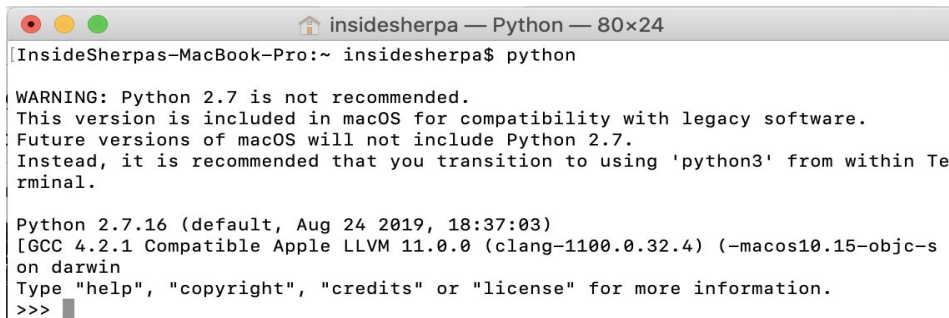
```
JPMC-tech-task-2 — andrew@Andrews-MacBook-Pro — zsh — 80x24
..C-tech-task-2
~/Desktop/testing
└─ cd JPMC-tech-task-2
Desktop/testing/JPMC-tech-task-2 master ✓ 11d
└─ ls
  README.md      package-lock.json public  test.csv
  datafeed       package.json      src    tsconfig.json
Desktop/testing/JPMC-tech-task-2 master ✓ 11d
└─
```

# Local Setup (Mac)

- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use.
- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.
- We'll discuss checking / installing Python in your system in the following slides

# Local Setup (Mac)

- Next, you'll need to have Python 2.7 **and/or** Python 3 installed on your machine. Follow the [instructions here](#) (python 2.7) and [here](#) (python 3) You can verify this on your terminal if you get a result like:



```

insidesherpa — Python — 80x24
[InsideSherpas-MacBook-Pro:~ insidesherpa$ python
WARNING: Python 2.7 is not recommended.
This version is included in macOS for compatibility with legacy software.
Future versions of macOS will not include Python 2.7.
Instead, it is recommended that you transition to using 'python3' from within Te
rminal.

Python 2.7.16 (default, Aug 24 2019, 18:37:03)
[GCC 4.2.1 Compatible Apple LLVM 11.0.0 (clang-1100.0.32.4) (-macos10.15-objc-s
on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
  
```

*(any python 2.7.x >= 2.7.16 should suffice but the latest 2.7.x is recommended (2.7.17));  
any python 3.x >= 3.6 is fine, latest is recommended (3.8.0))*

Execute the command below to verify what version you have:

**python --version**

**Note:** the image here is only of 2.7 but it should be similar if you check for python3. If you install both versions, **make sure python command in your terminal at least maps to 2.7.x**

Sometimes your system might have python3 version mapped to **python3 --version**

# Local Setup (Mac)

- Once you have Python 2.7 **and/or** Python 3 installed, you can start the server application in one terminal by just executing it:

(note: just choose to run one server; either the python 2 or python 3 version of server. Run the commands below depending on your python version)

*// If python --version = 2.7+, you must be in the JPMC-tech-task-2*

*// If python --version = 3+ , you must be in JPMC-tech-task-2-py3 directory*

```
python datafeed/server.py
```

*// If your system makes the distinction of python3 as `python3`,*

*// you must be in JPMC-tech-task-2-py3 directory*

```
python3 datafeed/server3.py
```

If ever you encounter an error when starting the server application, [see troubleshooting in this slide](#)

## Local Setup (Mac)

- If you've done the previous slide, then you should get something similar to the pic below when you ran the server.

```
HTTP server started on port 8080
```

- To be clear, the server application isn't stuck. It's behaving perfectly normal here. The reason why it's just like that for now is because it's just listening for requests
- For us to be able to make requests, we have to start the client application. The following slides will help you do that.

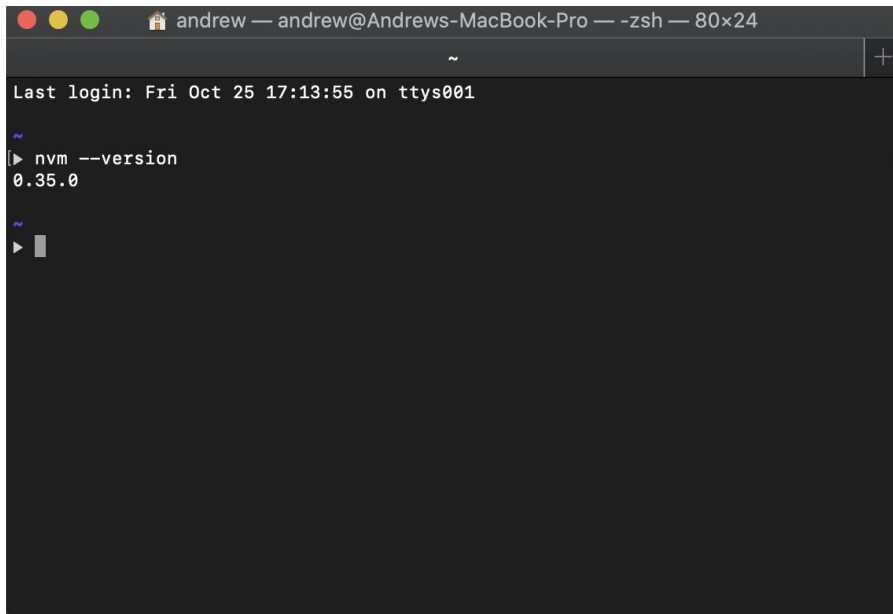
# Local Setup (Mac)

- In a separate terminal, let's install the other remaining dependencies i.e. Node and Npm. Node is a JavaScript runtime environment that executes JavaScript code outside of a browser and Npm is node's package manager that helps us to install other libraries/dependencies we'll be using in our web application app.
- To install node and npm and be able to manage versions seamlessly we will install NVM (node version manager). Once this is on your machine you can basically install any version of node and npm and switch depending on a project's needs. Follow these [instructions for to install nvm for mac](#).

note:you might need to have [Xcode](#) also installed in your system (<https://github.com/nvm-sh/nvm#important-notes>)

# Local Setup (Mac)

- You will know you've successfully installed nvm if you get a similar result below when you type the command **nvm --version**:

A screenshot of a macOS terminal window. The title bar shows 'andrew — andrew@Andrews-MacBook-Pro — -zsh — 80x24'. The terminal content shows a login message: 'Last login: Fri Oct 25 17:13:55 on ttys001'. Below that, the command 'nvm --version' is entered and executed, resulting in the output '0.35.0'. A cursor is visible on the line following the output.

```
andrew — andrew@Andrews-MacBook-Pro — -zsh — 80x24
~
Last login: Fri Oct 25 17:13:55 on ttys001

nvm
|▶ nvm --version
0.35.0

nvm
▶ |
```

## Local Setup (Mac)

- Now, we just need to install the right node version using nvm and that would consequently get us the right npm version as well. We do this by executing the commands:

```
nvm install v11.0.0
```

```
nvm use v11.0.0
```

- You should end up with a similar result in the next slide after doing the commands above



# Local Setup (Mac)

```

andrew — andrew@Andrews-MacBook-Pro — -zsh — 80x24
~
┌─▶ nvm install v11.0.0
└─▶ Downloading and installing node v11.0.0...
    Downloading https://nodejs.org/dist/v11.0.0/node-v11.0.0-darwin-x64.tar.xz...
    ##### 100.0%
    Computing checksum with shasum -a 256
    Checksums matched!
    Now using node v11.0.0 (npm v6.4.1)
    Creating default alias: default -> v11.0.0

┌─▶ nvm use v11.0.0
└─▶ Now using node v11.0.0 (npm v6.4.1)

┌─▶ npm -v
└─▶ 6.4.1
    
```

To check your node version type and enter:

**node -v**

To check your npm version type and enter:

**npm -v**

*Take note: If you open a new terminal after all this, make sure to recheck your node version and npm version. It might be the case you've switched to a different version so just execute `nvm use v11.0.0` again if ever...*

# Local Setup (Mac)

- Finally, to start the client application, all we have to do would be to run the commands below

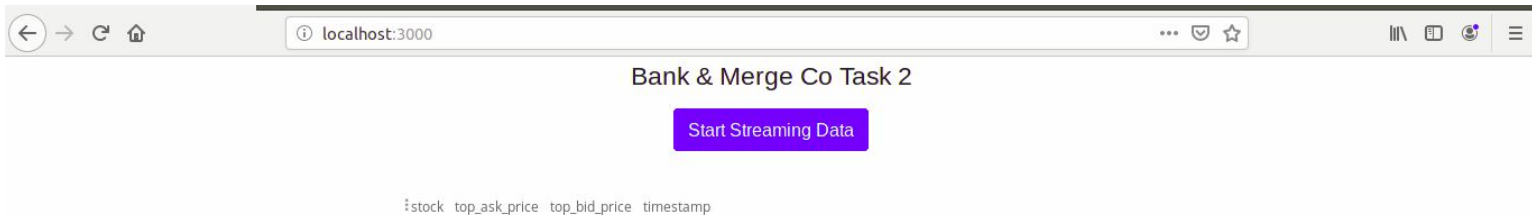
```
npm install
```

```
npm start
```

Note: If **npm install** succeeded for you (i.e. no errors) you don't have to do it again... But it always has to be successful first before you execute **npm start**

- If all goes well (and it should), you should end up with a similar result in the next slide. See [troubleshooting slides](#) if you encounter any problems with **npm install**

# Local Setup (Mac)



**Note:** This part assumes no errors came out of **npm install**. Some data should show if you click on the “Start Streaming Data” button. If nothing is showing, check the terminal where you’re running the server and see if anything printed out, like “Query received”. If there’s something like that and you’re not seeing results, then try using a different browser (e.g. Chrome/Firefox) and checking localhost:3000. If no response like “Query received” got printed in the terminal running the server, you might be experiencing [this issue](#).

# Local Setup: Troubleshooting (Mac)

- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code and eventually arrive at the desired output.
- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:
  - [dateutil dependency](#)
  - [npm install or npm start errors](#)
  - [Socket unavailable](#)

# Local Setup: Troubleshooting (Mac)

- In some cases, dependency issues might arise like when you run `server.py`:

```
Traceback (most recent call last):
  File "server.py", line 26, in <module>
    import dateutil.parser
ImportError: No module named dateutil.parser
```

In this case, you must install [pip](#) first. pip is python's package manager for installing python dependencies. Make sure you install pip for the right Python version you're working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too

## Local Setup: Troubleshooting (Mac)

- Installing pip nowadays usually involves downloading the [get-pip.py](#) script. If you followed the instructions in the last slide, it usually involves using the command:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

If you don't have curl, just install it in your system. For mac, [it's this way](#)

Then just run the script using python:

```
//if python --version = 2.7+ this will install pip for python2
```

```
//if python --version = 3+ this will install pip for python3
```

```
python get-pip.py
```

```
//if your system makes the distinction of python3 as `python3` then
```

```
//doing the command below will install pip for python3
```

```
python3 get-pip.py
```

# Local Setup: Troubleshooting (Mac)

- Then afterwards, you can run the following command on your terminal to install the [dependency](#):

```
pip install python-dateutil
```

Afterwards, you can rerun the server and then rerun the client

**Note:** For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

# Local Setup: Troubleshooting (Mac)

- In other cases, you might encounter problems after doing npm install. Errors like the following might appear on your end:

## CASE A:

```

...ts/isaacswork/jp-morgan-virtual-internship/task-2/JPMC-tech-task-2-PY3 — python3 datafeed/server3.py  ~/Documents/isaacswork/jp-morgan-virtual-internship/task-2/JPMC-tech-task-2-PY3 — -bash +
prebuild-install WARN install No prebuilt binaries found (target=11.0.0 runtime=node arch=x64 platform=darwin)
gyp ERR! configure error
gyp ERR! stack Error: Command failed: /anaconda3/bin/python -c import sys; print "%s.%s.%s" % sys.version_info[:3];
gyp ERR! stack   File "<string>", line 1
gyp ERR! stack     import sys; print "%s.%s.%s" % sys.version_info[:3];
gyp ERR! stack                                     ^
gyp ERR! stack SyntaxError: invalid syntax
gyp ERR! stack
gyp ERR! stack   at ChildProcess.exithandler (child_process.js:289:12)
gyp ERR! stack   at ChildProcess.emit (events.js:182:13)
gyp ERR! stack   at maybeClose (internal/child_process.js:962:16)
gyp ERR! stack   at Socket.stream.socket.on (internal/child_process.js:381:11)
gyp ERR! stack   at Socket.emit (events.js:182:13)
gyp ERR! stack   at Pipe._handle.close (net.js:611:12)
gyp ERR! System Darwin 18.7.0
gyp ERR! command "/Users/isaacwatson/.nvm/versions/node/v11.0.0/bin/node" "/Users/isaacwatson/.nvm/versions/node/v11.0.0/lib/node_modules/npm/node_modules/node-gyp/bin/node-gyp.js" "rebuild"
gyp ERR! cwd /Users/isaacwatson/Documents/isaacswork/jp-morgan-virtual-internship/task-2/JPMC-tech-task-2-PY3/node_modules/bufferutil
gyp ERR! node v11.0.0
    
```



# Local Setup: Troubleshooting (Mac)

- **CASE A:** If you're having a problem similar to this situation, most likely, you haven't set **python** in npm to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command npm uses to **python 2.7.x** . Here's a [reference](#) on how to do it.
- In our setup node-gyp is called by way of npm, so in order for you to avoid the error, you have to explicitly tell npm what python version to use. Thus, you have to set npm's 'python' config key to the appropriate value:
  - **npm config set python /path/to/executable/python** where /path/to/executable/python is the directory where the python2.7.x binary is, for example in /usr/bin/python usually in linux/mac or C:\\Python27 for Windows

# Local Setup: Troubleshooting (Mac)

- **CASE A (continuation):** Alternatively, you can set **python** in your system to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command to **python 2.7.x** (meaning **python --version** should output 2.7.x). [There's a discussion in our github repo on how to go about this.](#)
- It could help that you also temporarily remap your **python3.x** to the command **python3** if you also have it in your system. This is also discussed in the github thread mentioned in the earlier bullet point.

# Local Setup: Troubleshooting (Mac)

## CASE B:

NetworkError: Failed to execute 'send' on 'XMLHttpRequest': Failed to load 'http://localhost:8080/query?id=1'.

- **CASE B:** If you're having a problem similar to this situation, most likely, you aren't running the server application alongside the client app. Make sure you have another terminal open wherein you're running the server app which is run via `python datafeed/server.py` or `python datafeed/server3.py`

If running the server app errors out for you might be experiencing this other problem ([see linked slide](#))

# Local Setup: Troubleshooting (Mac)

- After trying the earlier suggestions for troubleshooting and **npm install** still errors out for you, try downloading the **node\_modules** [here](#).
- Then replace the `node_modules` inside your copy of the repo with the one you downloaded
- Afterwards, you can go ahead and just execute **npm start** (no need to run `npm install`)

# Local Setup: Troubleshooting (Mac)

- In other cases, you might encounter problems running the server app

## CASE A:

```

C:\Users\praja\JPMC-tech-task-1-py3>python server3.py
Traceback (most recent call last):
  File "server3.py", line 320, in <module>
    run(App())
  File "server3.py", line 214, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 452, in __init__
    self.server_bind()
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\http\server.py", line 137, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissions
    
```

note: the example here is from windows but a similar error might appear for mac

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the datafeed/server.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the src/DataStreamer.ts and change the line where it has 8080 to 8085

# Local Setup: Troubleshooting (Mac)

## CASE B:

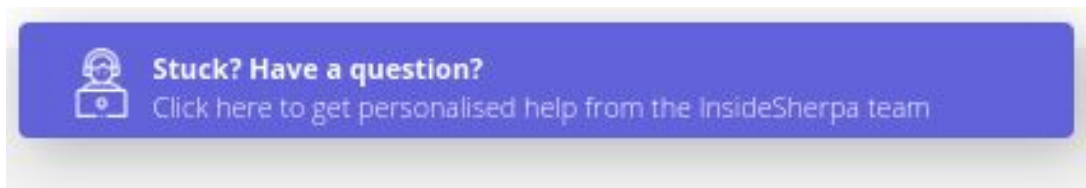
```

Traceback (most recent call last):
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 320, in <module>
    run(App())
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 213, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 420, in __init__
    self.server_bind()
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/BaseHTTPServer.py", line 108, in server_bind
    SocketServer.TCPServer.server_bind(self)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 434, in server_bind
    self.socket.bind(self.server_address)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 48] Address already in use
    
```

In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting cmd+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by [following this guide](#)

## Local Setup: Troubleshooting (Mac)

- If you did encounter any other issues not mentioned here , please post your issue/inquiry here: <https://github.com/insidesherpa/JPMC-tech-task-2/issues> or <https://github.com/insidesherpa/JPMC-tech-task-2-py3/issues> depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)
- You can also submit your query in the [module page](#)'s support modal that pops out when you click the floating element on the page (see image below)





InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

# Setting up your Windows for the JPMorgan Chase program

Module 2 - Use JPMorgan Chase frameworks and tools



# Local Setup (Windows)

- If your machine is running on Windows, follow this setup guide to get started (*the examples here are on Windows X but it should be relatively similar for other versions*)
- First you must have git installed in your system. Git is usually used by programmers to collaborate with code in a software project. To do install, follow this [quick guide](#). You know you have installed successfully when you get this output on your command line (cmd). (*any git version should suffice but the latest is recommended*)

```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\j\>git --version
git version 2.23.0.windows.1
```

# Local Setup (Windows)

- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following commands on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-2.git
```

```
git clone https://github.com/insidesherpa/JPMC-tech-task-2-PY3.git
```

- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later

## Local Setup (Windows)

- You'll know you cloned successfully if you have the copy of the application code on your machine:

```
C:\Users\jmf>git clone https://github.com/insidesherpa/JPMC-tech-task-2.git
Cloning into 'JPMC-tech-task-2'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 92 (delta 8), reused 8 (delta 2), pack-reused 74Unpacking objects: 79% (73/92)
Unpacking objects: 100% (92/92), done.
```

*note: the image here just does not contain the other repository but it should if you did the previous slides and execute the **dir** command. `dir` just lists the files/folders in the current directory*

*note: take note of the current directory your commandline is in because that is the location where you copied the repository. In the image above it's in **C:\Users\<username>** (See next slide on how to navigate inside the repository) **AVOID cloning your repository in C:\\Windows\System32***

# Local Setup (Windows)

- To access the files inside from the terminal, just change directory by typing:

```
cd JPMC-tech-task-2
```

*note: If you choose to work using python3 and your system has version python3.x as default instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above); **cd** changes the directory your terminal is in. Take note, the **cd** command assumes the directory you want to go into is inside the current directory you're in. For more info on how to use cd, check [here](#)*

```
cd JPMC-tech-task-2-py3
```

# Local Setup (Windows)

- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use.
- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.
- We'll discuss checking / installing Python in your system in the following slides

# Local Setup (Windows)

- Next, you'll need to have Python 2.7 **and/or** Python 3+ installed on your machine. Follow the [instructions here](#) (for python2), and [here](#) (for python3) You can verify this on your command line (cmd) if you get a result like:

```
C:\Users\>python -V
Python 2.7.13
```

```
C:\Users\>python3 --version
Python 3.8.0
```

Execute the command below to verify what version you have:

```
python --version
```

**Note:** the image here is only of 2.7 but it should be similar if you check for python3. If you install both versions, **make sure the command `python` maps to 2.7.x at least**

*(any python 2.7.x > = 2.7.16 should suffice but the latest 2.7.x is recommended (2.7.17); any python 3.x >= 3.6 is fine, latest is recommended (3.8.0))*

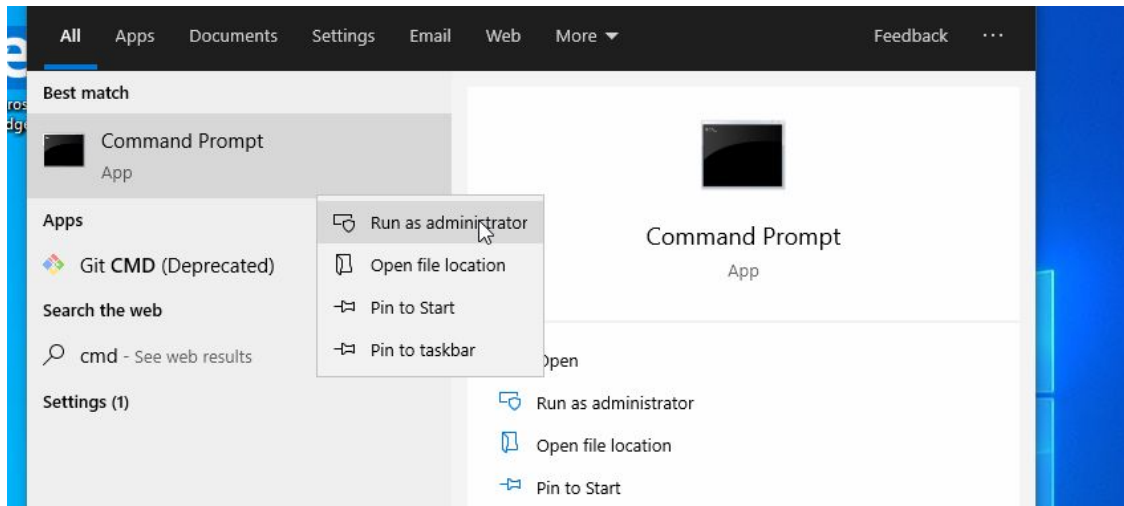
Sometimes your system might have python3 version mapped to **python3 --version**

## Local Setup (Windows)

- If something went wrong with the output you got from the previous slide, please check the [Troubleshooting slides for Windows in this guide.](#)

# Local Setup (Windows)

- Once you have python is installed, all you have to do get the server up and running is to start the server in its own cmd. Ensure that the command line wherein you run the server app is on Administrator mode:





## Local Setup (Windows)

- When you open the cmd in admin mode, you'll notice that its current directory starts from C:\Windows\System32. Assuming you cloned it your home directory you need to use the `cd` command to get there. For instance if I cloned it in C:\Users\insidesherpa, then I should do something like:

```
cd \Users\insidesherpa\JPMC-tech-task-2
```

Or

```
cd \Users\insidesherpa\JPMC-tech-task-2-py3
```

*note: the example above isn't what you're going to type on your system. You have a different user account in \Users where you probably cloned the repo. Use that instead...*

# Local Setup (Windows)

- Once you have Python 2.7 and/or Python 3 installed, you can start the server application in one terminal by just executing it:  
(note: just choose to run one server; either the python 2 or python 3 version of server. Run the commands below depending on your python version)

```
// If python --version = 2.7+, you must be in the JPMC-tech-task-2  
// If python --version = 3+ , you must be in JPMC-tech-task-2-py3 directory  
python datafeed/server.py
```

```
// If your system makes the distinction of python3 as `python3`,  
// you must be in JPMC-tech-task-2-py3 directory  
python3 datafeed/server3.py
```

If ever you encounter an error when starting the server application, see [troubleshooting in this slide](#)

## Local Setup (Windows)

- If you've done the previous slide, then you should get something similar to the pic below when you ran the server.

```
HTTP server started on port 8080
```

- To be clear, the server application isn't stuck. It's behaving perfectly normal here. The reason why it's just like that for now is because it's just listening for requests
- For us to be able to make requests, we have to start the client application. The following slides will help you do that.

## Local Setup (Windows)

- In a separate command line, let's install the other remaining dependencies i.e. Node and Npm. Node is a JavaScript runtime environment that executes JavaScript code outside of a browser and Npm is node's package manager that helps us to install other libraries/dependencies we'll be using in our web application app.
- To install node and npm and be able to manage versions seamlessly we will install NVM (node version manager). Once this is on your machine you can basically install any version of node and npm and switch depending on a project's needs. Follow these [instructions to install nvm for windows](#).

# Local Setup (Windows)

- You will know you've successfully installed nvm if you get a similar result below when you type the command `nvm -v`

```

C:\Users\>nvm -v
Running version 1.1.7.
Usage:
  nvm arch                : Show if node is running in 32 or 64 bit mode.
  nvm install <version> [arch] : The version can be a node.js version or "latest" for the latest stable version.
                                Optionally specify whether to install the 32 or 64 bit version (defaults to system arch).
                                Set [arch] to "all" to install 32 AND 64 bit versions.
                                Add --insecure to the end of this command to bypass SSL validation of the remote download server.
  nvm list [available]      : List the node.js installations. Type "available" at the end to see what can be installed. Aliased as ls.
  nvm on                    : Enable node.js version management.
  nvm off                   : Disable node.js version management.
  nvm proxy [url]          : Set a proxy to use for downloads. Leave [url] blank to see the current proxy.
                                Set [url] to "none" to remove the proxy.
  nvm node_mirror [url]    : Set the node mirror. Defaults to https://nodejs.org/dist/. Leave [url] blank to use default url.
  nvm npm_mirror [url]     : Set the npm mirror. Defaults to https://github.com/npm/cli/archive/. Leave [url] blank to default url.
  nvm uninstall <version> : The version must be a specific version.
  nvm use [version] [arch] : Switch to use the specified version. Optionally specify 32/64bit architecture.
                                nvm use <arch> will continue using the selected version, but switch to 32/64 bit mode.
  nvm root [path]         : Set the directory where nvm should store different versions of node.js.
                                If <path> is not set, the current root will be displayed.
  nvm version              : Displays the current running version of nvm for Windows. Aliased as v
    
```

## Local Setup (Windows)

- Now, we just need to install the right node version using nvm and that would consequently get us the right npm version as well. We do this by executing the commands:

```
nvm install v11.0.0
```

```
nvm use 11.0.0
```

- You should end up with a similar result in the next slide after doing the commands above

# Local Setup (Windows)

```

C:\> Command Prompt
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\> node -v
v11.0.0

C:\Users\> npm -v
6.4.1

C:\User>
    
```

To check your node version type and enter:

**node -v**

To check your npm version type and enter:

**npm -v**

*Take note: If you open a new terminal after all this, make sure to recheck your node version and npm version. It might be the case you've switched to a different version so just execute `nvm use v11.0.0` again if ever...*

# Local Setup (Windows)

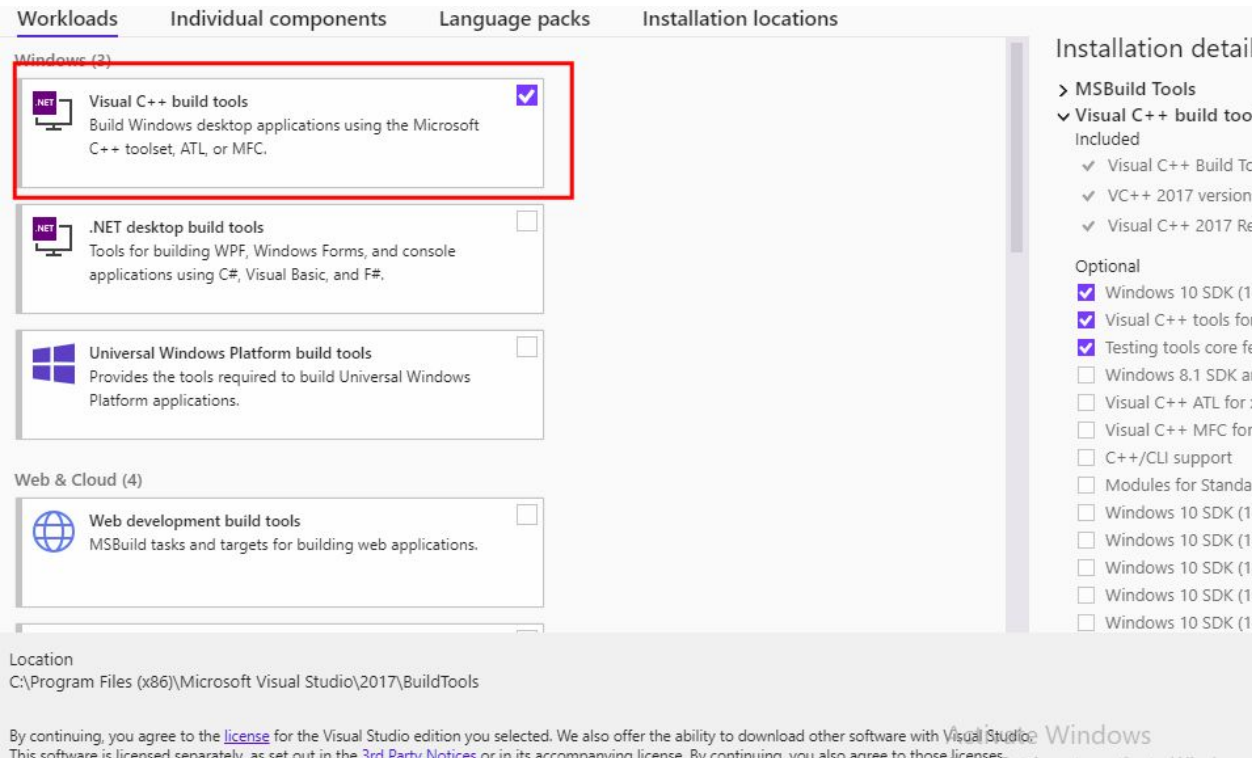
- Because windows' nvm isn't exactly like mac's or linux's, there's still a couple more dependencies we have to install in order to get this whole application to work.
- We have to install Visual C++ Build Environment via [Visual Studio Build Tools](#). Run the downloaded .exe file and make sure to have the basics installed i.e "Visual C++ Build Tools on your machine. This is actually need because of [node-gyp](#). After installing, make sure to run in your command line:

```
npm config set msvs_version 2017
```

*(see image in the next slide)*



# Local Setup (Windows)



The screenshot shows the Visual Studio Workloads Installer interface. The 'Workloads' tab is active, and the 'Visual C++ build tools' workload is selected, indicated by a red box and a checked checkbox. The 'Installation detail' pane on the right shows the following configuration:

- MSBuild Tools
  - Visual C++ build tool Included
    - Visual C++ Build To
    - VC++ 2017 version
    - Visual C++ 2017 Re
  - Optional
    - Windows 10 SDK (10
    - Visual C++ tools for
    - Testing tools core fe
    - Windows 8.1 SDK ar
    - Visual C++ ATL for x
    - Visual C++ MFC for
    - C++/CLI support
    - Modules for Standar
    - Windows 10 SDK (10
    - Windows 10 SDK (10
    - Windows 10 SDK (10
    - Windows 10 SDK (10
    - Windows 10 SDK (10
    - Windows 10 SDK (10

The location for the build tools is specified as: C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools

By continuing, you agree to the [license](#) for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

This is how installing visual C++ build environment would look like when you run the .exe file

# Local Setup (Windows)

- Finally, to start the client application, all we have to do would be to run the commands below

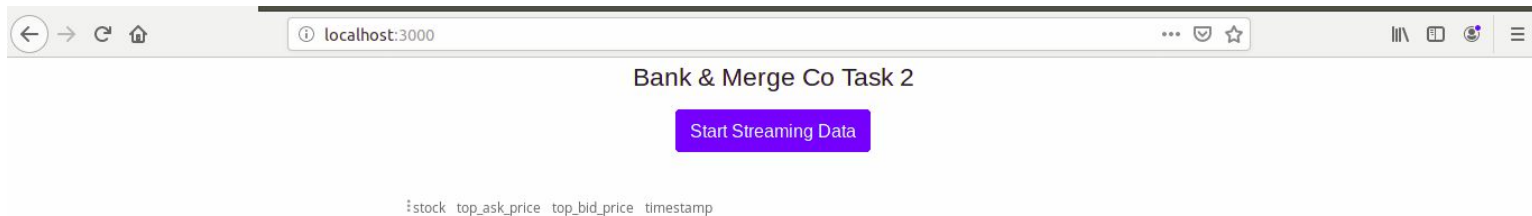
```
npm install
```

Note: If **npm install** succeeded for you (i.e. no errors) you don't have to do it again... But it always has to be successful first before you execute **npm start**

```
npm start
```

- If all goes well , you should end up with a similar result in the next slide. If you encounter problems with npm install, particularly relating to node-gyp try the other windows options of [installing here](#). If there are other issues, please refer to the [Troubleshooting slides in this guide](#).

# Local Setup (Windows)



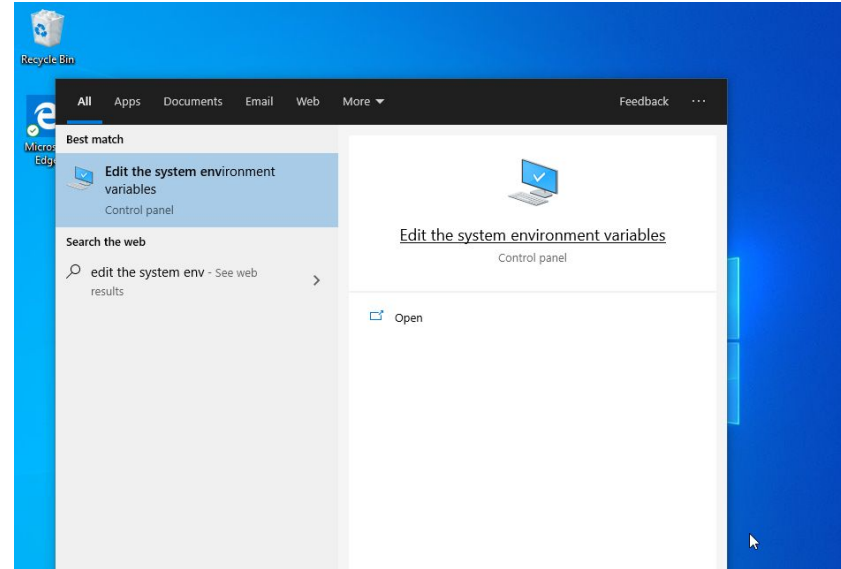
**Note:** This part assumes no errors came out of **npm install**. Some data should show if you click on the “Start Streaming Data” button. If nothing is showing, check the command line where you’re running the server and see if anything printed out, like “Query received”. If there’s something like that and you’re not seeing results, then try using a different browser (e.g. Chrome/Firefox) and checking localhost:3000. If no response like “Query received” got printed in the command line running the server, you might be experiencing [this issue](#).

# Local Setup: Troubleshooting (Windows)

- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code and eventually arrive at the desired output.
- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:
  - [python not recognized or not returning in your command line](#)
  - [dateutil dependency](#)
  - [npm install or npm start errors](#)
  - [Socket unavailable](#)

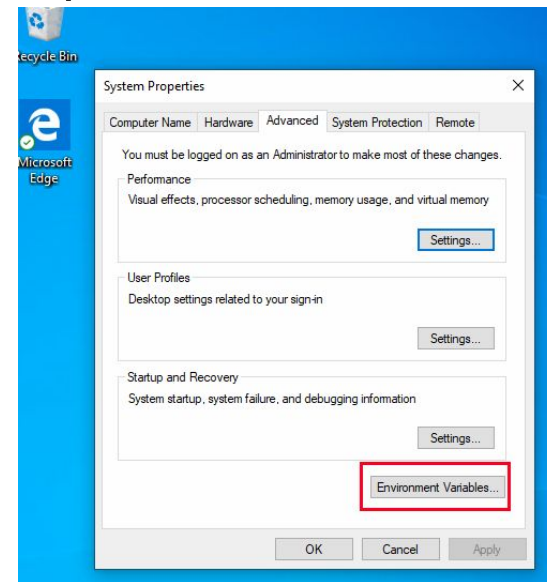
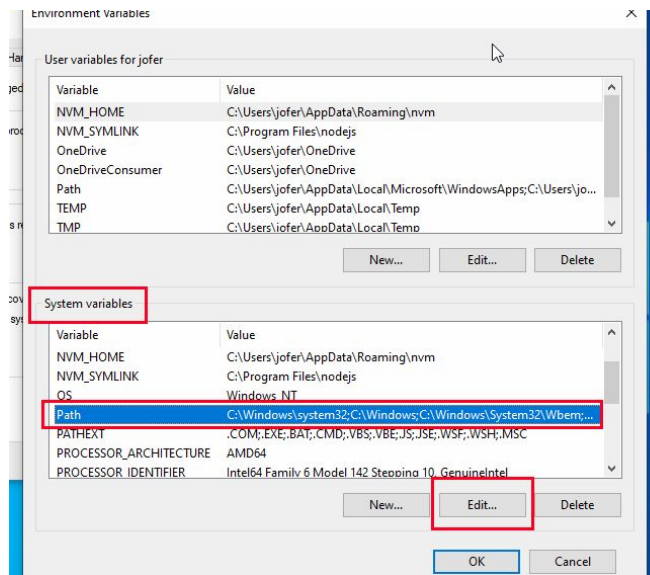
# Local Setup: Troubleshooting (Windows)

- There are some cases when you tried installing python e.g. through the usual installation process or via other software like anaconda, and when you open your command line and try executing any python command like **python** or **python --version** nothing returns. The problem here is usually because python isn't properly set in your system environment's path variable properly. To do this go to your start menu, search "edit the system environment" and you should be able to see something like the image on the right



# Local Setup: Troubleshooting (Windows)

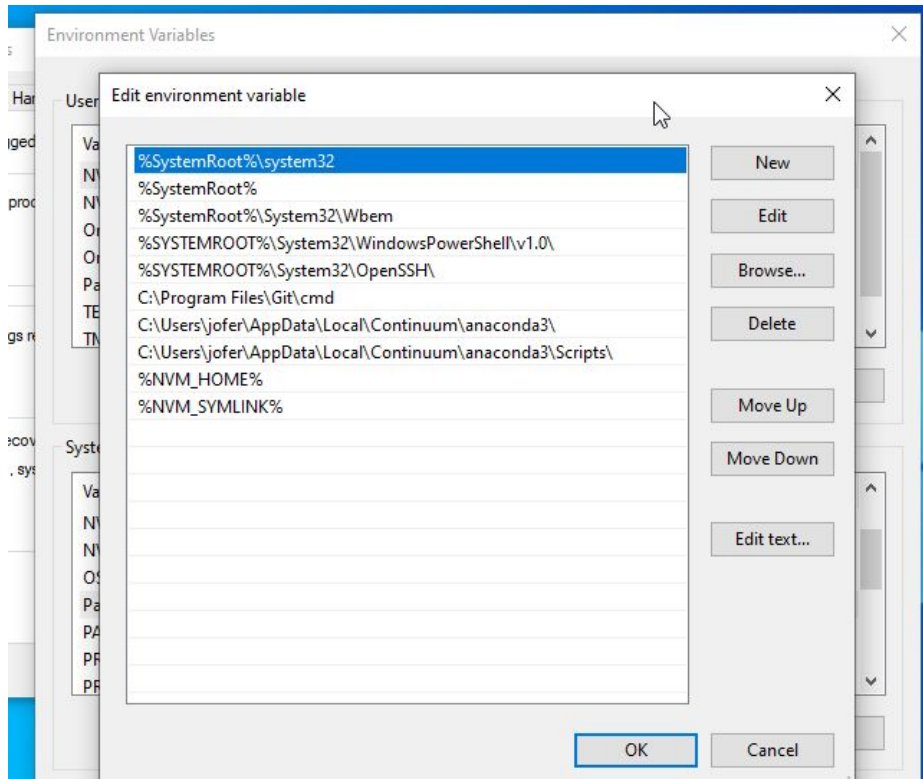
- Click that and click “Environment Variables” in the next window (image on left side)
- You should then come up with the image below:



You want to edit the **Path** under **System variables**

# Local Setup: Troubleshooting (Windows)

- You should end up with a window like this after the steps from the previous slide:



You want to make sure the directory where the **Python** executable/application you want to use is included in the list of paths.

If you followed how to install **Python2.7.x** in the earlier slides, you should have it in `C:\\Python27`. Make sure to include that in your path.

Same goes for **Python3.x** if you want to enable it. But make sure to remove your Python2.7.x path first .

If you installed python using other means, e.g. anaconda, its python executable is located elsewhere but same method of putting the path applies

Don't forget to restart your command line after setting a new path to reflect the changes...

# Local Setup: Troubleshooting (Windows)

- In some cases, dependency issues might arise like when you run `server.py`:

```
Traceback (most recent call last):
  File "server.py", line 26, in <module>
    import dateutil.parser
ImportError: No module named dateutil.parser
```

In this case, you must install [pip](#) first. pip is python's package manager for installing python dependencies. Make sure you install pip for the right Python version you're working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too



# Local Setup: Troubleshooting (Windows)

- Installing pip nowadays usually involves downloading the [get-pip.py](https://bootstrap.pypa.io/get-pip.py) script. If you followed the instructions in the last slide, it usually involves using the command:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

If you don't have curl, just copy a python file, rename it to get-pip.py and replace all the contents of the python file to what's in get-pip.py

```
//if python --version = 2.7+ this will install pip for python2
```

```
//if python --version = 3+ this will install pip for python3
```

```
python get-pip.py
```

```
//if your system makes the distinction of python3 as `python3` then
```

```
//doing the command below will install pip for python3
```

```
python3 get-pip.py
```

# Local Setup: Troubleshooting (Windows)

- After that, for pip to be a recognizable command in your terminal/command line, you need to add it in your system's path environment variable
  - On Windows, for Python2.7 it's usually in C:\\Python27\\Scripts. It would also be similar for Python3.x if you followed the installation guide for Python earlier (e.g. C:\\Python3X\\Scripts)
  - Make sure you open a new command line too and use that instead after doing this
- To edit your system path environment variable it's similar to the slides [here](#).
- Alternatively you can access it doing something like:
  - **C:\\Python27\\Scripts\\pip.exe <parameters>** (similar for python3x if it was installed in C:\\)
  - <parameters> could be something like **C:\\Python27\\Scripts\\pip.exe install python-dateutil**
  - Take note though, this assumes that you have your python installed in drive C:\\

# Local Setup: Troubleshooting (Windows)

- Then afterwards, you can run the following command on your terminal to install the [dependency](#):

```
pip install python-dateutil
```

Afterwards, you can rerun the server and then rerun the client

**Note:** For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

# Local Setup: Troubleshooting (Windows)

- In other cases, you might encounter problems after doing `npm install`. Errors like the following might appear on your end

## CASE A:

```
C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3>npm install
npm WARN deprecated fsevents@1.2.4: Way too old

> bufferutil@3.0.5 install C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3\node_modules\bufferutil
> prebuild-install || node-gyp rebuild

prebuild-install WARN install No prebuilt binaries found (target=11.0.0 runtime=node arch=x64 platform=win32)

C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3\node_modules\bufferutil>if not defined npm_config_node_gyp (node "C:\Users\Dar\Desktop\Main\InsideSherpa\JPMorgan Chase\Task 2\JPMC-tech-task-2-PY3\node_modules\npm\node_modules\npm-lifecycle\node-gyp-bin\..\..\node_modules\node-gyp\bin\node-gyp.js" rebuild ) else (node "C:\Users\Dar\AppData\Roaming\nvm\v11.0.0\node_modules\node-gyp\bin\node-gyp.js" rebuild )
gyp ERR! configure error
gyp ERR! stack Error: Command failed: C:\Users\Dar\AppData\Local\Programs\Python\Python37\python.EXE -c import sys; print "%s.%s.%s" % sys.version_info[:3];
gyp ERR! stack   File "<string>", line 1
gyp ERR! stack     import sys; print "%s.%s.%s" % sys.version_info[:3];
gyp ERR! stack                                     ^
gyp ERR! stack SyntaxError: invalid syntax
gyp ERR! stack
gyp ERR! stack   at ChildProcess.exithandler (child_process.js:289:12)
gyp ERR! stack   at ChildProcess.emit (events.js:182:13)
gyp ERR! stack   at maybeClose (internal/child_process.js:962:16)
gyp ERR! stack   at Process.ChildProcess._handle.onexit (internal/child_process.js:251:5)
gyp ERR! System Windows_NT 10.0.18362
gyp ERR! command "C:\\Program Files\\nodejs\\node.exe" "C:\\Users\\Dar\\AppData\\Roaming\\nvm\\v11.0.0\\node_modules\\npm\\node_modules\\node-gyp\\bin\\node-gyp.js" "C:\\Users\\Dar\\Desktop\\Main\\InsideSherpa\\JPMorgan Chase\\Task 2\\JPMC-tech-task-2-PY3\\node_modules\\bufferutil"
```

# Local Setup: Troubleshooting (Windows)

- **CASE A:** If you're having a problem similar to this situation, most likely, you haven't set **python** in npm to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command npm uses to **python 2.7.x**. Here's a [reference](#) on how to do it.
- In our setup node-gyp is called by way of npm, so in order for you to avoid the error, you have to explicitly tell npm what python version to use. Thus, you have to set npm's 'python' config key to the appropriate value:
  - **npm config set python /path/to/executable/python** where /path/to/executable/python is the directory where the python2.7.x binary is, for example in /usr/bin/python usually in linux/mac or C:\\Python27 for Windows

## Local Setup: Troubleshooting (Windows)

- **CASE A (continuation):** Alternatively, you can set **python** in your system to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command to **python 2.7.x** (meaning **python --version** should output 2.7.x) [Check the slides on how to set this via the system environment path variable in the earlier slides.](#)
- It could help that you also temporarily remove in that same path variable the mapping to **python3.x** if you also have it in your system. Just place it back after you **npm install** successfully and if you plan on using the **py3** version of the repository...

# Local Setup: Troubleshooting (Windows)

## CASE B:

NetworkError: Failed to execute 'send' on 'XMLHttpRequest': Failed to load 'http://localhost:8080/query?id=1'.

- **CASE B:** If you're having a problem similar to this situation, most likely, you aren't running the server application alongside the client app. Make sure you have another terminal open wherein you're running the server app which is run via `python datafeed/server.py` or `python datafeed/server3.py`

If running the server app errors out for you might be experiencing this other problem ([see linked slide](#))

# Local Setup: Troubleshooting (Windows)

## CASE C:

```

C:\Windows\System32\JPMC-tech-task-2-PY3>npm install

> bufferutil@3.0.5 install C:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil
> prebuild-install || node-gyp rebuild

prebuild-install WARN install No prebuilt binaries found (target=11.0.0 runtime=node arch=x64 platform=win32)

C:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil>if not defined npm_config_node_gyp (node "C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\npm-lifecycle\nod
.\node_modules\node-gyp\bin\node-gyp.js" rebuild ) else (node "C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\node-gyp\bin\node-gyp.js" rebuild )
SBUILD : error MSB1009: Project file does not exist.
with: build/binding.sln
gyp ERR! build error
gyp ERR! stack Error: `C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools\MSBuild\15.0\Bin\MSBuild.exe` failed with exit code: 1
gyp ERR! stack   at ChildProcess.onExit (C:\Users\mythu\AppData\Roaming\nvm\v11.0.0\node_modules\npm\node_modules\node-gyp\lib\build.js:262:23)
gyp ERR! stack   at ChildProcess.emit (events.js:182:13)
gyp ERR! stack   at Process.ChildProcess._handle.onexit (internal/child_process.js:240:12)
gyp ERR! System Windows_NT 10.0.17763
gyp ERR! command "C:\\Program Files\\nodejs\\node.exe" "C:\\Users\\mythu\\AppData\\Roaming\\nvm\\v11.0.0\\node_modules\\npm\\node_modules\\node-gyp\\bin\\node-gyp.js" "rebuild"
gyp ERR! cwd C:\Windows\System32\JPMC-tech-task-2-PY3\node_modules\bufferutil
gyp ERR! node -v v11.0.0
    
```

**Case C:** This problem is most likely due to the fact that you're in C:\\Windows\\System32. Do not clone the repo here as mentioned. Clone in C:\\Users\\<some\_account>\\ instead.



# Local Setup: Troubleshooting (Windows)

- After trying the earlier suggestions for troubleshooting and **npm install** still errors out for you, try downloading the **node\_modules** [here](#).
- Then replace the `node_modules` inside your copy of the repo with the one you downloaded (*make sure the folder is still named `node_modules` in your repo*)
- Afterwards, you can go ahead and just execute **npm start** (no need to run `npm install`)

# Local Setup: Troubleshooting (Windows)

- In other cases, you might encounter problems running the server app

## CASE A:

```

C:\Users\praja\JPMC-tech-task-1-py3>python server3.py
Traceback (most recent call last):
  File "server3.py", line 320, in <module>
    run(App())
  File "server3.py", line 214, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 452, in __init__
    self.server_bind()
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\http\server.py", line 137, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "C:\Users\praja\AppData\Local\Programs\Python\python37\lib\socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissions
    
```

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the datafeed/server.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the src/DataStreamer.ts and change the line where it has 8080 to 8085

# Local Setup: Troubleshooting (Windows)

## CASE B:

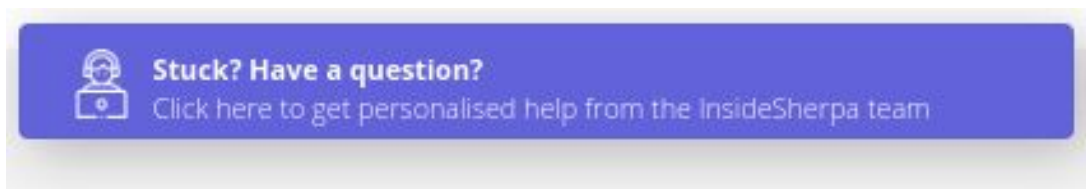
```

Traceback (most recent call last):
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 320, in <module>
    run(App())
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 213, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 420, in __init__
    self.server_bind()
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/BaseHTTPServer.py", line 108, in server_bind
    SocketServer.TCPServer.server_bind(self)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 434, in server_bind
    self.socket.bind(self.server_address)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 48] Address already in use
    
```

In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting ctrl+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by [following this guide](#)

# Local Setup: Troubleshooting (Windows)

- If you did encounter any other issues not mentioned here, please post your issue/inquiry here: <https://github.com/insidesherpa/JPMC-tech-task-2/issues> or <https://github.com/insidesherpa/JPMC-tech-task-2-py3/issues> depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)
- You can also submit your query in the [module page](#)'s support modal that pops out when you click the floating element on the page (see image below)





InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

# Setting up your Linux for the JPMorgan Chase program

Module 2 - Use JPMorgan Chase frameworks and tools

## Local Setup (Linux)

- If your machine is running on any flavor of linux, follow this setup guide to get started
- First you must have git installed in your system. Git is usually used by programmers to collaborate with code in a software project. You can do install git by simply running the command below in your terminal (ctrl+alt+t):

```
~$ sudo apt-get install git
```

- You'll know you have git if you get a similar result on your terminal:

```
~$ git version  
git version 2.17.1
```

# Local Setup (Linux)

- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following commands on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-2.git
```

```
git clone https://github.com/insidesherpa/JPMC-tech-task-2-PY3.git
```

- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later



## Local Setup (Linux)

- You'll know you cloned successfully if you have the copy of the application code on your machine:

```
→ ~ git clone https://github.com/insidesherpa/JPMC-tech-task-2.git
Cloning into 'JPMC-tech-task-2'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 92 (delta 8), reused 8 (delta 2), pack-reused 74
Unpacking objects: 100% (92/92), done.
→ ~ ls | grep JPM
JPMC-tech-task-2
```

- To access the files inside from the terminal, just change directory by typing:  
`cd JPMC-tech-task-2`

*note: If you choose to work using python3 and your system has version python3.x as default instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above); **cd** changes directory your terminal is in. Check [this](#) for more info on how to use **cd***

```
cd JPMC-tech-task-2-py3
```



# Local Setup (Linux)

- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use.
- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.
- We'll discuss checking / installing Python in your system in the following slides

# Local Setup (Linux)

- Next, you'll need to have Python 2.7 **and/or** Python 3 installed on your machine. Follow the [instructions here](#). (python2) **and** [here](#) (python3) For most cases, Linux environments already have Python 2.7. You can verify this on your terminal if you get a result like:

```
~$ python --version
Python 2.7.15+
```

Execute the command below to verify what version you have:

```
python --version
```

Note: the image here is only of 2.7 but it should be similar if you check for python3. If you installed both versions, **make sure the command `python` maps to 2.7.x at least**

*(any python 2.7.x >= 2.7.16 should suffice but the latest 2.7.x is recommended (2.7.17));*

*any python 3.x >= 3.6 is fine, latest is recommended (3.8.0))*

Sometimes your system might have it as

```
python3 --version
```

## Local Setup (Linux)

- Once you have Python installed, all you have to do get the application up and running is to start the server script in a separate terminal (see next slide). If ever you encounter an error when starting the server application, [see troubleshooting in this slide](#)

# Local Setup (Linux)

- Once you have Python 2.7 and Python 3 installed, you can start the server application in one terminal by just executing it:  
(note: just choose to run one server; either the python 2 or python 3 version of server. Run the commands below depending on your python version)

```
// If python --version = 2.7+, you must be in the JPMC-tech-task-2  
// If python --version = 3+ , you must be in JPMC-tech-task-2-py3 directory  
python datafeed/server.py
```

```
// If your system makes the distinction of python3 as `python3`,  
// you must be in JPMC-tech-task-2-py3 directory  
python3 datafeed/server3.py
```

If ever you encounter an error when starting the server application, [see troubleshooting in this slide](#)

## Local Setup (Linux)

- If you've done the previous slide, then you should get something similar to the pic below when you ran the server.

```
HTTP server started on port 8080
```

- To be clear, the server application isn't stuck. It's behaving perfectly normal here. The reason why it's just like that for now is because it's just listening for requests
- For us to be able to make requests, we have to start the client application. The following slides will help you do that.

# Local Setup (Linux)

- In a separate terminal, let's install the other remaining dependencies i.e. Node and Npm. Node is a JavaScript runtime environment that executes JavaScript code outside of a browser and Npm is node's package manager that helps us to install other libraries/dependencies we'll be using in our web application app.
- To install node and npm and be able to manage versions seamlessly we will install NVM (node version manager). Once this is on your machine you can basically install any version of node and npm and switch depending on a project's needs. Follow these [instructions to install nvm for linux](#).

# Local Setup (Linux)

- You will know you've successfully installed nvm if you get a similar result below when you type the command **nvm --version**

```
→ ~ nvm --version  
0.34.0  
→ ~
```

## Local Setup (Linux)

- Now, we just need to install the right node version using nvm and that would consequently get us the right npm version as well. We do this by executing the commands:

```
nvm install v11.0.0
```

```
nvm use v11.0.0
```

- You should end up with a similar result in the next slide after doing the commands above



# Local Setup (Linux)

```
→ ~ nvm install v11.0.0
v11.0.0 is already installed.
Now using node v11.0.0 (npm v6.4.1)
→ ~ nvm use v11.0.0
Now using node v11.0.0 (npm v6.4.1)
→ ~ node -v
v11.0.0
→ ~ npm -v
6.4.1
→ ~ █
```

To check your node version type and enter:

**node -v**

To check your npm version type and enter:

**npm -v**

*Take note: If you open a new terminal after all this, make sure to recheck your node version and npm version. It might be the case you've switched to a different version so just execute `nvm use v11.0.0` again if ever...*

# Local Setup (Linux)

- Finally, to start the client application, all we have to do would be to run the commands below

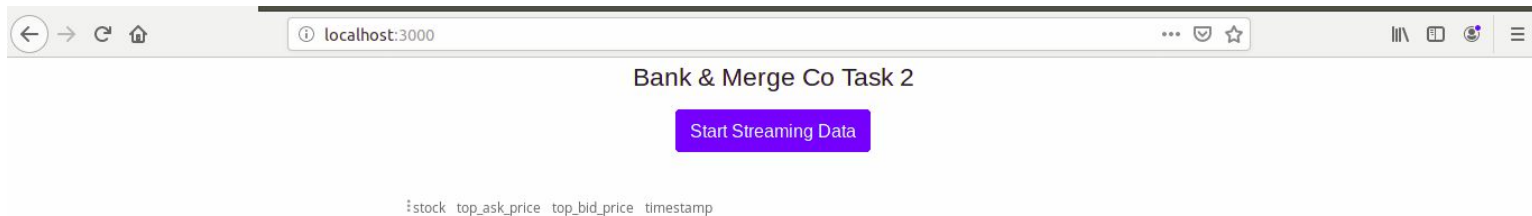
```
npm install
```

Note: If **npm install** succeeded for you (i.e. no errors) you don't have to do it again... But it always has to be successful first before you execute **npm start**

```
npm start
```

- If all goes well (and it should), you should end up with a similar result in the next slide. If there are other issues, please refer to the [Troubleshooting slides in this guide](#).

# Local Setup (Linux)



**Note:** This part assumes no errors came out of **npm install**. Some data should show if you click on the “Start Streaming Data” button. If nothing is showing, check the command line where you’re running the server and see if anything printed out, like “Query received”. If there’s something like that and you’re not seeing results, then try using a different browser (e.g. Chrome/Firefox) and checking localhost:3000. If no response like “Query received” got printed in the command line running the server, you might be experiencing [this issue](#).

# Local Setup: Troubleshooting (Linux)

- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code and eventually arrive at the desired output.
- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:
  - [dateutil dependency](#)
  - [npm install or npm start errors](#)
  - [Socket unavailable](#)

# Local Setup: Troubleshooting (Linux)

- In some cases, dependency issues might arise like when you run `server.py`:

```
Traceback (most recent call last):  
  File "server.py", line 26, in <module>  
    import dateutil.parser  
ImportError: No module named dateutil.parser
```

In this case, you must install [pip](#) first. pip is python's package manager for installing python dependencies. Make sure you install pip for the right Python version you're working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too

# Local Setup: Troubleshooting (Linux)

- Installing pip nowadays usually involves downloading the [get-pip.py](#) script. If you followed the instructions in the last slide, it usually involves using the command:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

If you don't have curl, just install it in your system. For linux, [it's this way](#)

Then just run the script using python:

```
//if python --version = 2.7+ this will install pip for python2
```

```
//if python --version = 3+ this will install pip for python3
```

```
python get-pip.py
```

```
//if your system makes the distinction of python3 as `python3` then
```

```
//doing the command below will install pip for python3
```

```
python3 get-pip.py
```

# Local Setup: Troubleshooting (Linux)

- Then afterwards, you can run the following command on your terminal to install the [dependency](#):

```
pip install python-dateutil
```

Afterwards, you can rerun the server and then rerun the client

**Note:** For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

# Local Setup: Troubleshooting (Linux)

- In other cases, you might encounter problems after doing npm install. Errors like the following might appear on your end:

**CASE A:** note: the example here is from mac but a similar error might appear for linux

```

...ts/isaacswork/jp-morgan-virtual-internship/task-2/JPMC-tech-task-2-PY3 — python3 datafeed/server3.py  ~/Documents/isaacswork/jp-morgan-virtual-internship/task-2/JPMC-tech-task-2-PY3 — -bash +
prebuild-install WARN install No prebuilt binaries found (target=11.0.0 runtime=node arch=x64 platform=darwin)
gyp ERR! configure error
gyp ERR! stack Error: Command failed: /anaconda3/bin/python -c import sys; print "%s.%s.%s" % sys.version_info[:3];
gyp ERR! stack   File "<string>", line 1
gyp ERR! stack     import sys; print "%s.%s.%s" % sys.version_info[:3];
gyp ERR! stack                                     ^
gyp ERR! stack SyntaxError: invalid syntax
gyp ERR! stack
gyp ERR! stack   at ChildProcess.exithandler (child_process.js:289:12)
gyp ERR! stack   at ChildProcess.emit (events.js:182:13)
gyp ERR! stack   at maybeClose (internal/child_process.js:962:16)
gyp ERR! stack   at Socket.stream.socket.on (internal/child_process.js:381:11)
gyp ERR! stack   at Socket.emit (events.js:182:13)
gyp ERR! stack   at Pipe._handle.close (net.js:611:12)
gyp ERR! System Darwin 18.7.0
gyp ERR! command "/Users/isaacwatson/.nvm/versions/node/v11.0.0/bin/node" "/Users/isaacwatson/.nvm/versions/node/v11.0.0/lib/node_modules/npm/node_modules/node-gyp/bin/node-gyp.js" "rebuild"
gyp ERR! cwd /Users/isaacwatson/Documents/isaacswork/jp-morgan-virtual-internship/task-2/JPMC-tech-task-2-PY3/node_modules/buffer-utl
gyp ERR! node v11.0.0
    
```



# Local Setup: Troubleshooting (Linux)

- **CASE A:** If you're having a problem similar to this situation, most likely, you haven't set **python** in npm to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command npm uses to **python 2.7.x** . Here's a [reference](#) on how to do it.
- In our setup node-gyp is called by way of npm, so in order for you to avoid the error, you have to explicitly tell npm what python version to use. Thus, you have to set npm's 'python' config key to the appropriate value:
  - **npm config set python /path/to/executable/python** where /path/to/executable/python is the directory where the python2.7.x binary is, for example in /usr/bin/python usually in linux/mac or C:\\Python27 for Windows

# Local Setup: Troubleshooting (Linux)

- **CASE A (continuation):** Alternatively, you can set **python** in your system to map to **python2.7.x**. Make sure you've installed **python 2.7.x** first and make sure you map the **python** command to **python 2.7.x** (meaning **python --version** should output 2.7.x). [There's a discussion in our github repo on how to go about this.](#) (mac solution is similar for linux)
- It could help that you also temporarily remap your **python3.x** to the command **python3** if you also have it in your system. This is also discussed in the github thread mentioned in the earlier bullet point.

# Local Setup: Troubleshooting (Linux)

## CASE B:

NetworkError: Failed to execute 'send' on 'XMLHttpRequest': Failed to load 'http://localhost:8080/query?id=1'.

- **CASE B:** If you're having a problem similar to this situation, most likely, you aren't running the server application alongside the client app. Make sure you have another terminal open wherein you're running the server app which is run via `python datafeed/server.py` or `python datafeed/server3.py`

If running the server app errors out for you might be experiencing this other problem ([see linked slide](#))

# Local Setup: Troubleshooting (Linux)

- After trying the earlier suggestions for troubleshooting and **npm install** still errors out for you, try downloading the **node\_modules** [here](#).
- Then replace the `node_modules` inside your copy of the repo with the one you downloaded
- Afterwards, you can go ahead and just execute **npm start** (no need to run `npm install`)

# Local Setup: Troubleshooting (Linux)

- In other cases, you might encounter problems running the server app

## CASE A:

```

C:\Users\praja\JPMC-tech-task-1-py3>python server3.py
Traceback (most recent call last):
  File "server3.py", line 320, in <module>
    run(App())
  File "server3.py", line 214, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 452, in __init__
    self.server_bind()
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\http\server.py", line 137, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "C:\Users\praja\AppData\Local\Programs\Python\Python37\lib\socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissions
    
```

note: the example here is from windows but a similar error might appear for linux

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the datafeed/server.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the src/DataStreamer.ts and change the line where it has 8080 to 8085

# Local Setup: Troubleshooting (Linux)

## CASE B:

```

Traceback (most recent call last):
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 320, in <module>
    run(App())
  File "/Users/oluwafeyisayoafolabi/Desktop/JPMorgan Virtual Internship/JPMC-tech-task-1/server.py", line 213, in run
    server = ThreadedHTTPServer((host, port), RequestHandler)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 420, in __init__
    self.server_bind()
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/BaseHTTPServer.py", line 108, in server_bind
    SocketServer.TCPServer.server_bind(self)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/SocketServer.py", line 434, in server_bind
    self.socket.bind(self.server_address)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 48] Address already in use
    
```

In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting ctrl+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by [following this guide](#)

# Local Setup: Troubleshooting (Linux)

- If you did encounter any issues, please post your issue/inquiry here: <https://github.com/insidesherpa/JPMC-tech-task-2/issues> or <https://github.com/insidesherpa/JPMC-tech-task-2-py3/issues> depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)
- You can also submit your query in the [module page](#)'s support modal that pops out when you click the floating element on the page (see image below)

