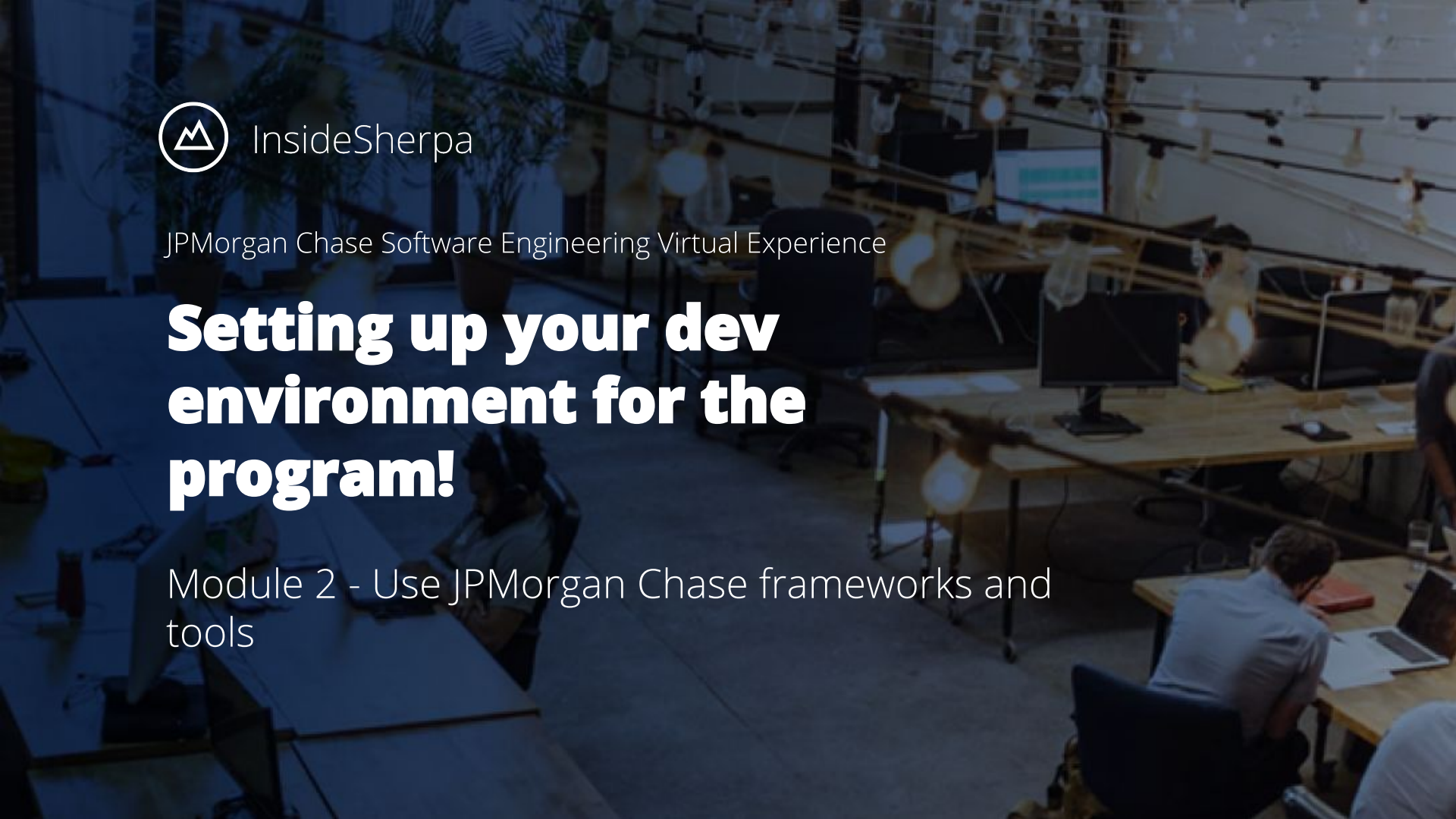


InsideSherpa

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

(Mac)
Setting up your dev environment for task 2

(Windows)
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. To do this, 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 — 127×34
InsideSherpas-MacBook-Pro:~ insidesherpa$ git --version
git version 2.20.1 (Apple Git-117)
InsideSherpas-MacBook-Pro:~ insidesherpa$
```

Local Setup (Mac)

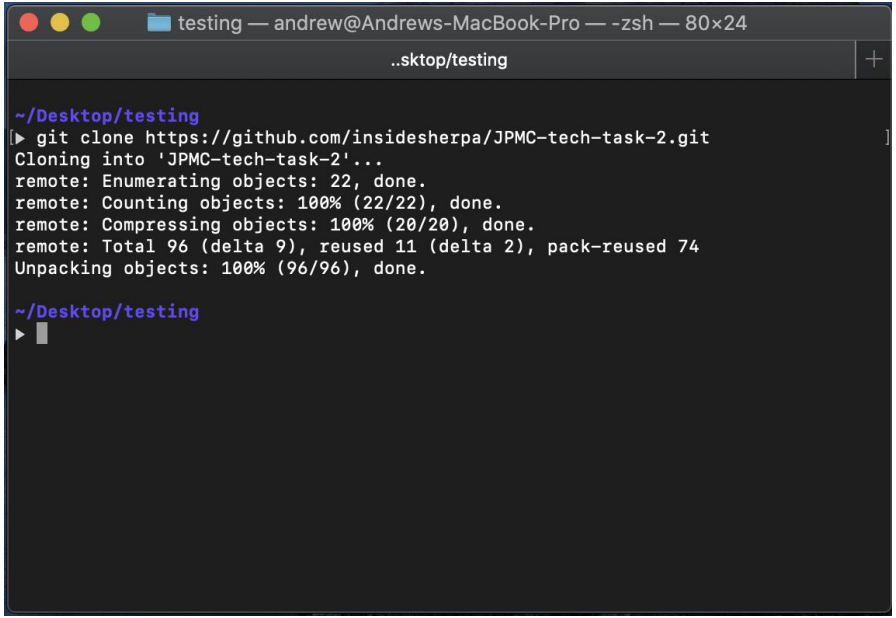
- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following commands on your terminal:

```
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*

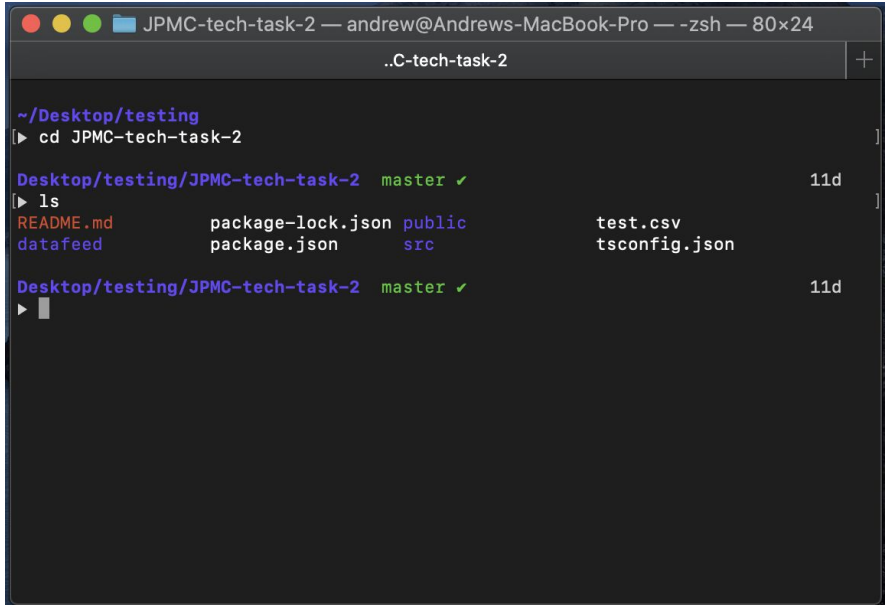
Local Setup (Mac)

- 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 or above 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*

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



Local Setup (Mac)

- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. <u>It all depends on what Python version you have or will choose to have/use.</u>

- 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 or Python 3 installed on your machine. Follow the instructions here(python 2.7) or here (python 3) You can verify this on your terminal if you get a result like:

```
insidesherpa — Python — 80×24
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 should suffice but the latest 2.7.x is recommended;*
any python 3.x is fine, latest is recommended)

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

Sometimes your system might have it as

`python3 --version`

Local Setup (Mac)

- Once you have Python 2.7 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.

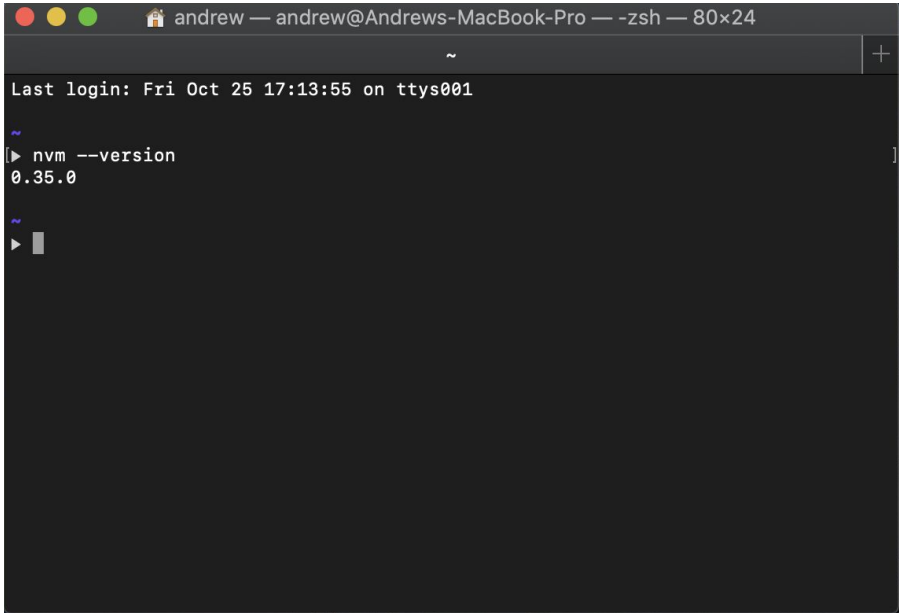
Local Setup (Mac)

- 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.

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**:



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 — 80×24

```
~
▶ 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

<u>Take note:</u> 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)

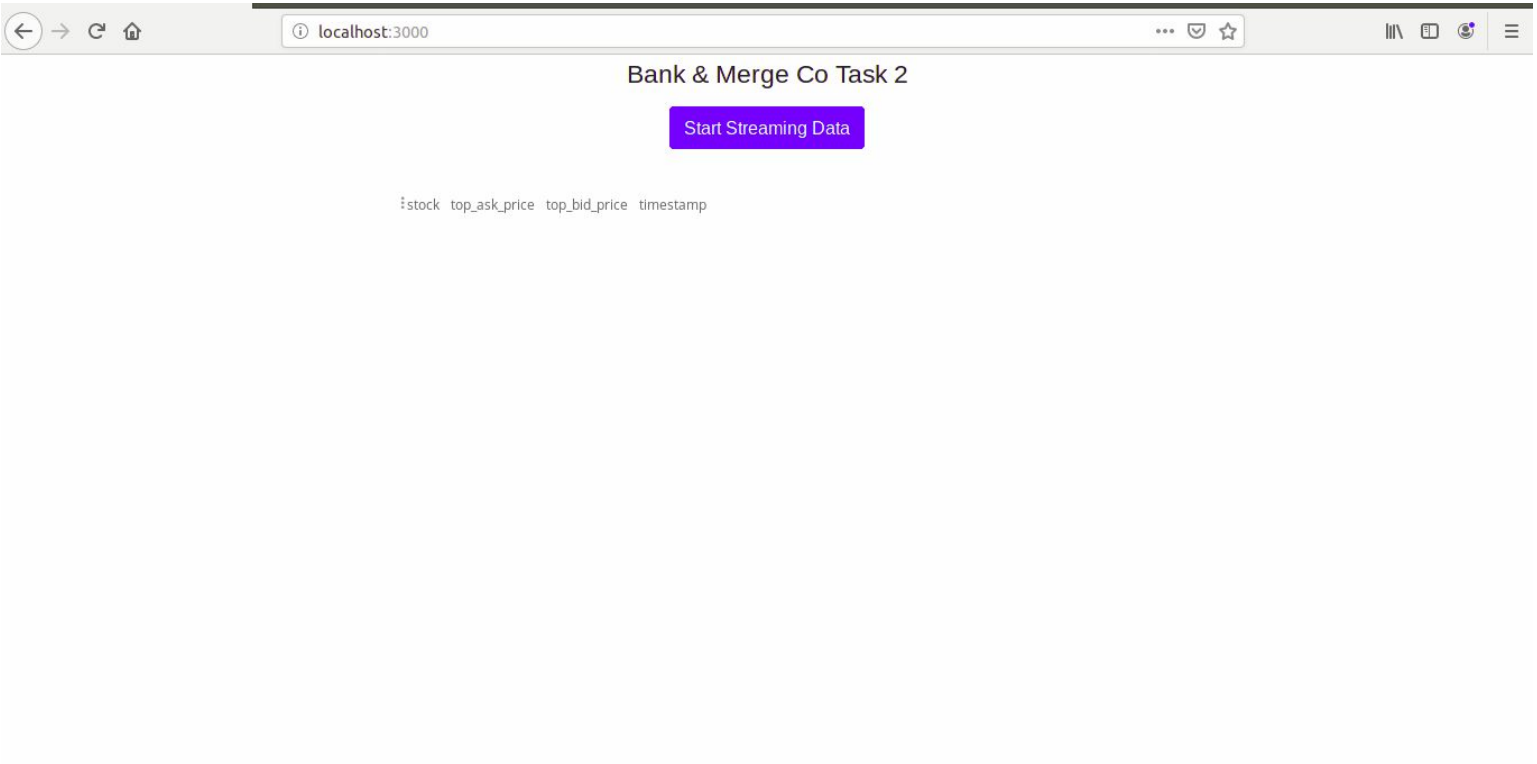
- Finally, to start the client application, all we have to do would be to run the commands below

```
npm install

npm start
```

- If all goes well (and it should), you should end up with a similar result in the next slide

Local Setup (Mac)



Local Setup (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.

- 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. Make sure you install pip for the right Python version you're working with in this project.

Local Setup (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 (Mac)

- Then afterwards, you can run the following command on your terminal to install the dependency:

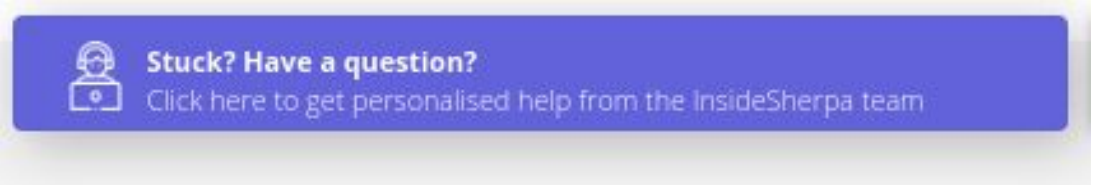
```
pip install python-dateutil
```

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

Local Setup (Mac)

- 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)



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. To do this, 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\[redacted]>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\j[redacted]>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*

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 or above 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*

```
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. <u>It all depends on what Python version you have or will choose to have/use.</u>

- 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 or Python 3+ installed on your machine. Follow the instructions here (for python2), or here (for python3) You can verify this on your command line (cmd) if you get a result like:

```
C:\Users\[redacted]>python -V
Python 2.7.13
```

```
C:\Users\[redacted]>python3 --version
Python 3.8.0
```

(any python 2.7.x should suffice but the latest 2.7.x is recommended;
any python 3.x is fine, latest is recommended)

Execute the command below to verify what version you have:

`python --version`

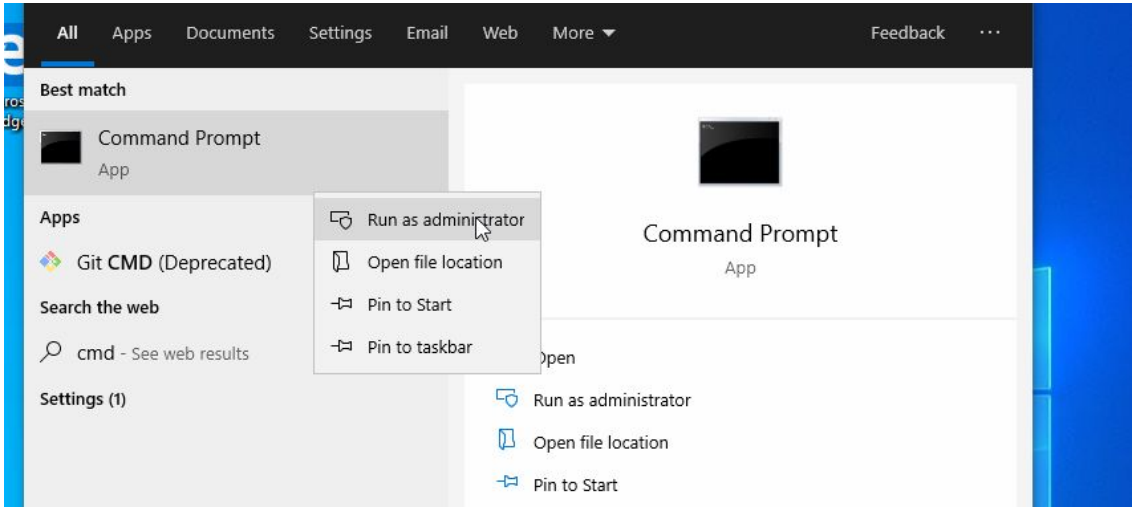
Note: the image here is only of 2.7 but it should be similar if you check for python3

Sometimes your system might have it as

`python3 --version`

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 (*see image in the next slide*). Ensure that the command line wherein you run the server app is on Administrator mode:



Local Setup (Windows)

- Once you have Python 2.7 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
```

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**

Command Prompt

```
C:\Users\[redacted]>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 downlo
ad server.
  nvm list [available]         : List the node.js installations. Type "available" at the end to see what can be installe
d. 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 def
ault 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 v11.0.0
```

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

Local Setup (Windows)

Command Prompt

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

C:\Users\[redacted]>node -v
v11.0.0

C:\Users\[redacted]>npm -v
6.4.1

C:\User[redacted]>
```

To check your node version type and enter:

node -v

To check your npm version type and enter:

npm -v

<u>Take note:</u> 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)

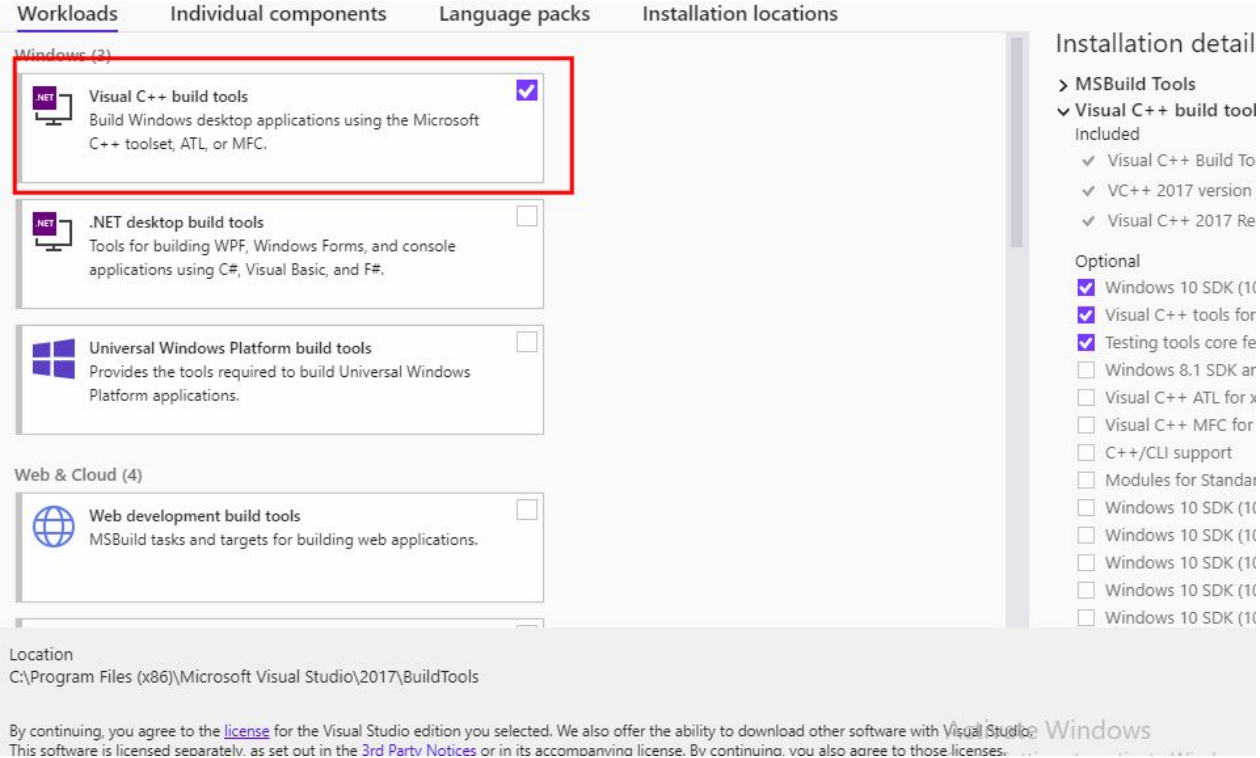
- Because windows' nvm isn't exactly like mac's or linux's, there's still a couple more dependencies we have to install in order to get this whole application to work.

- 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)



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

Local Setup (Windows)

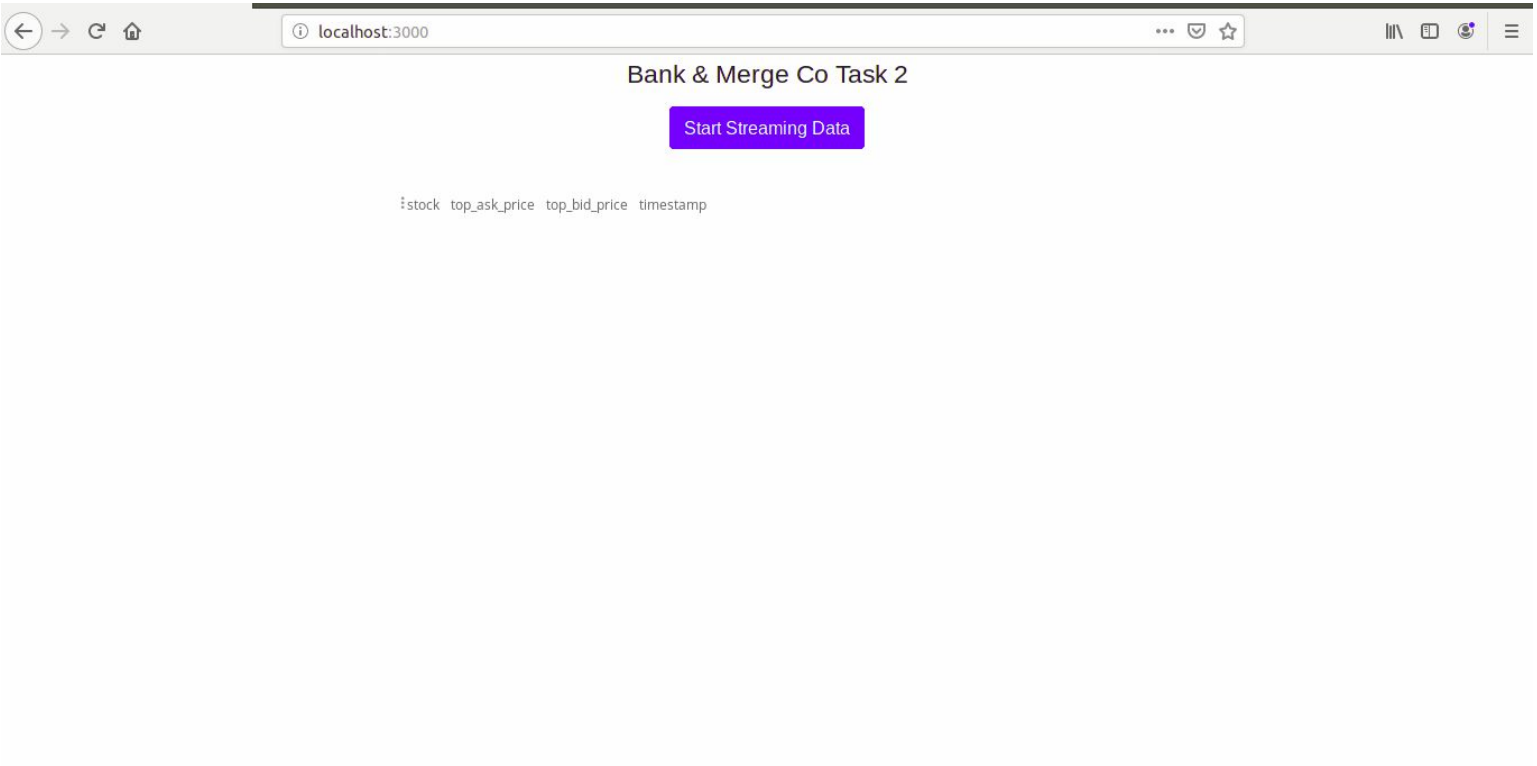
- Finally, to start the client application, all we have to do would be to run the commands below

```
npm install
```

npm start

- If all goes well , you should end up with a similar result in the next slide. If you you encounter problems with npm install, particularly relating to node-gyp try the other windows options of installing here

Local Setup (Windows)



Local Setup (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.

- 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. Make sure you install pip for the right Python version you're working with in this project.

Local Setup (Windows)

- 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 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 (Windows)

- Then afterwards, you can run the following command on your terminal to install the dependency:

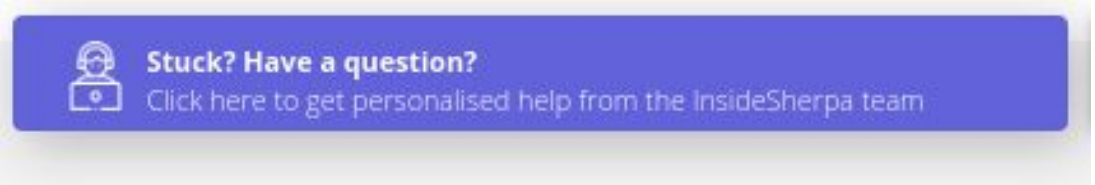
```
pip install python-dateutil
```

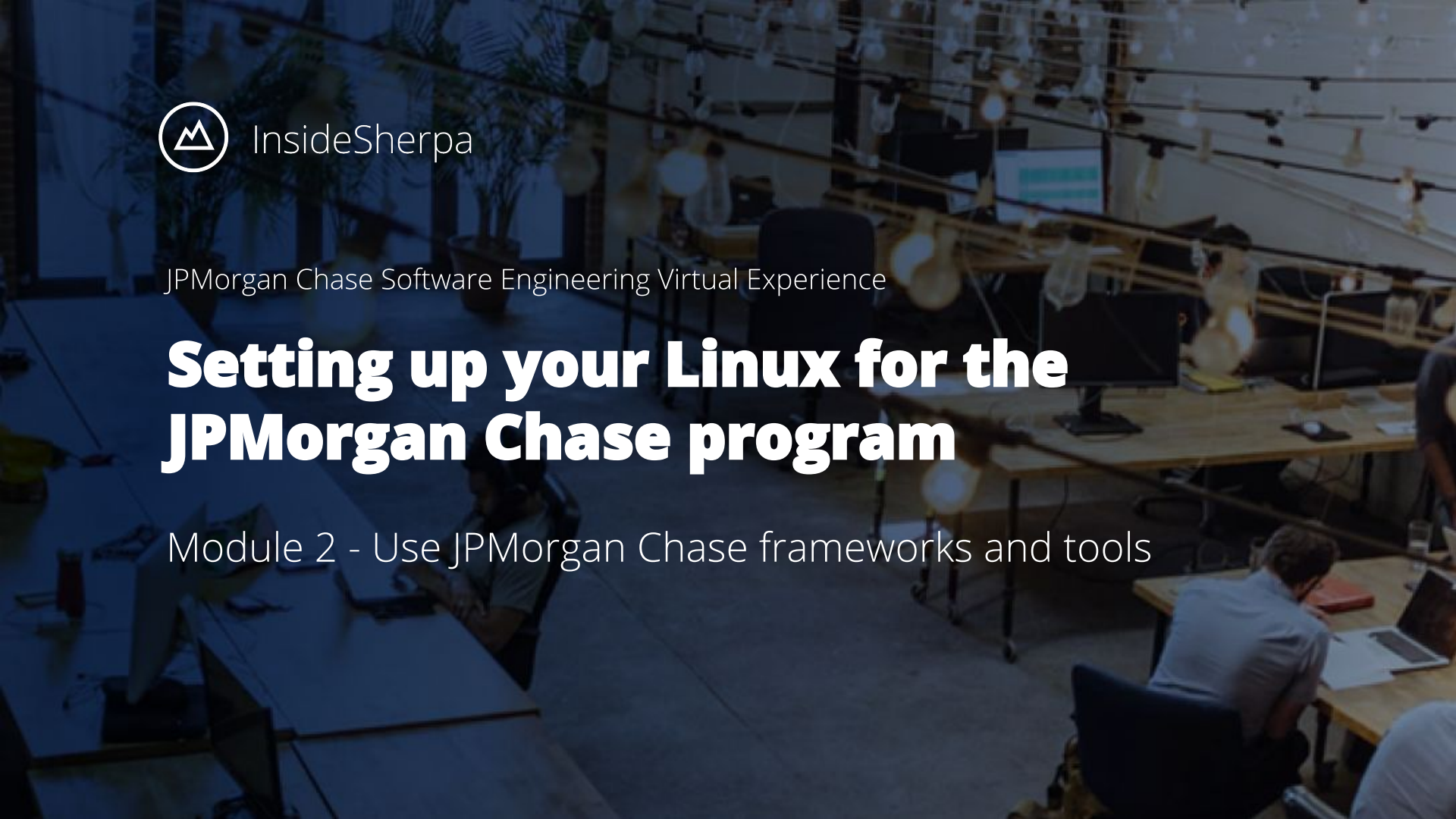
Afterwards, you can rerun the server and then rerun the client

Local Setup (Windows)

- 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)





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. You can do this 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 or above 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*

  `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. <u>It all depends on what Python version you have or will choose to have/use.</u>

- 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 or Python 3 installed on your machine. Follow the instructions here. (python2) or 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+
```

(*any python 2.7.x should suffice but the latest 2.7.x is recommended;*
any python 3.x is fine, latest is recommended)

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

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 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
```

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

<u>Take note:</u> 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)

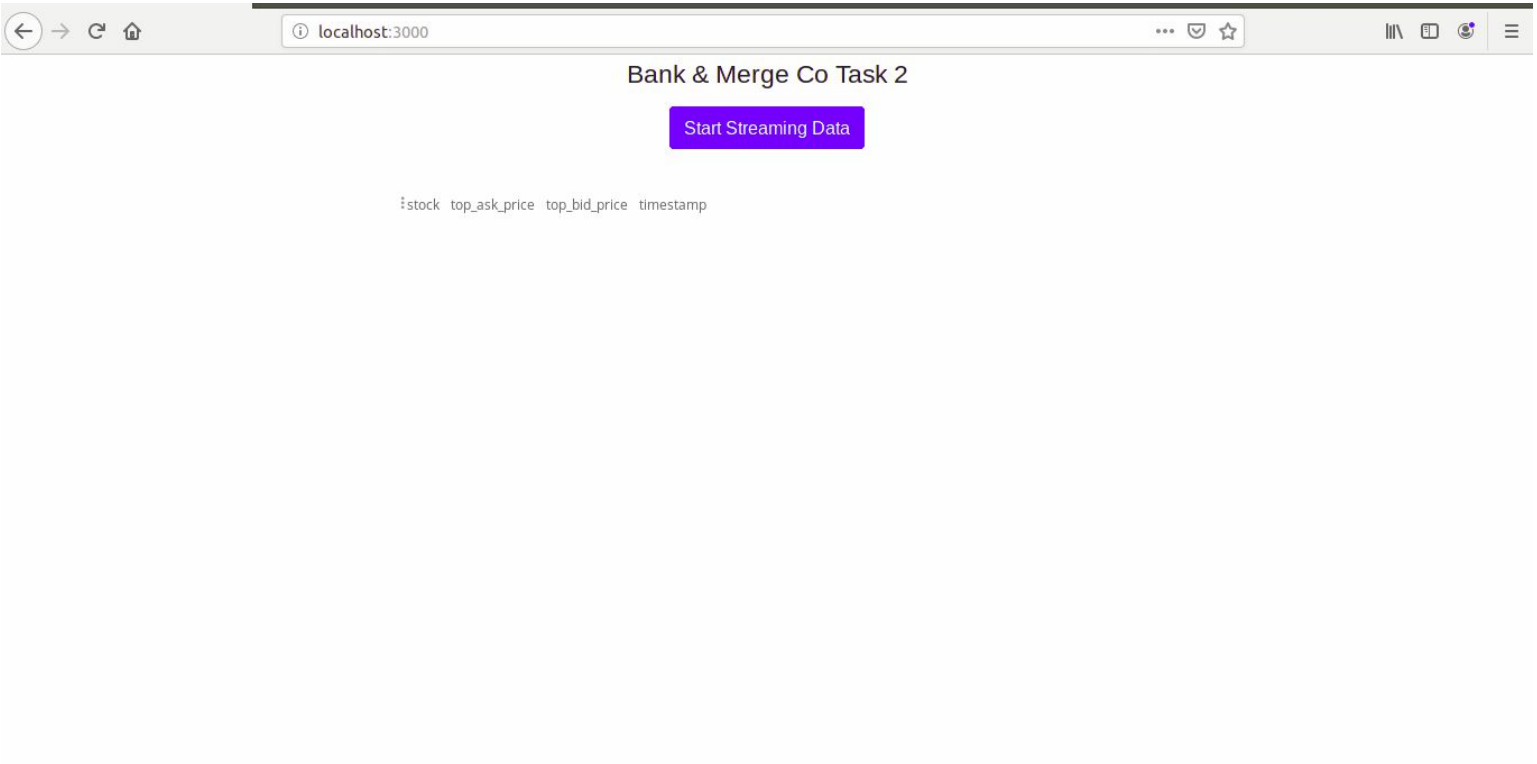
- Finally, to start the client application, all we have to do would be to run the commands below

```
npm install
```

npm start

- If all goes well (and it should), you should end up with a similar result in the next slide

Local Setup (Linux)



Local Setup (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.

- 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. Make sure you install pip for the right Python version you're working with in this project.

Local Setup (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 (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

Local Setup (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)