

GitHub Codespaces

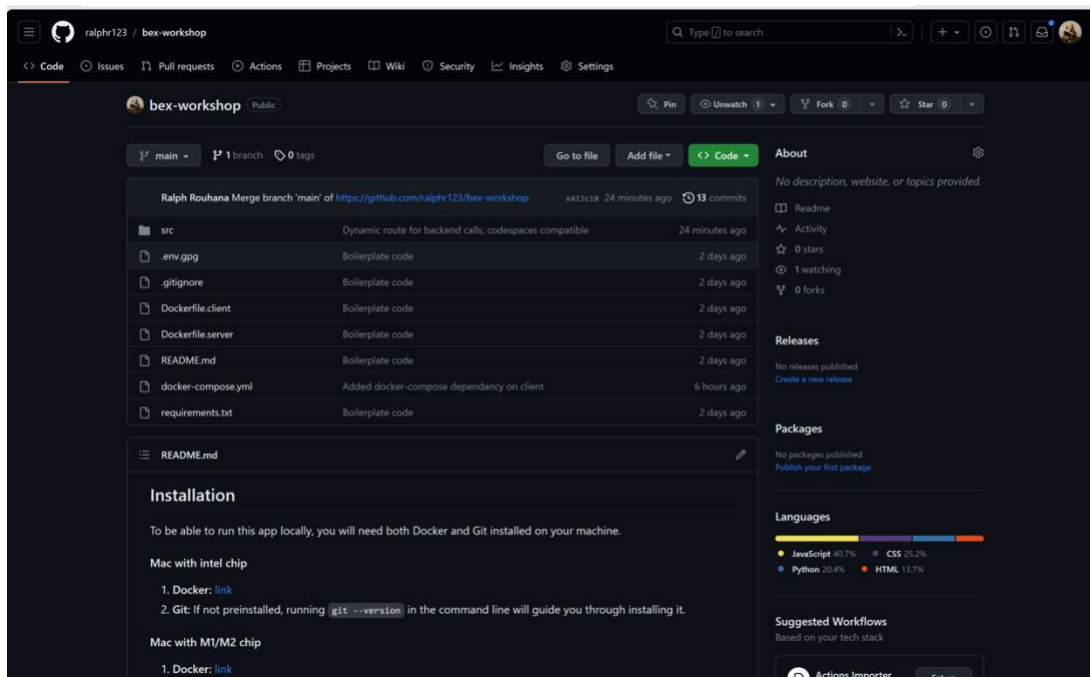
Codespaces is an online code editor for you to write and test your code on your browser. This will allow you to get setup quickly and prevent you from having to spend time installing the required services.

To use GitHub Codespaces, you will need a GitHub account. GitHub is a service that allows developers to easily store and manage their code in the cloud. If you don't already have an account, you can [sign up here](#).

Once you have an account, sign in on your preferred browser and open this link:

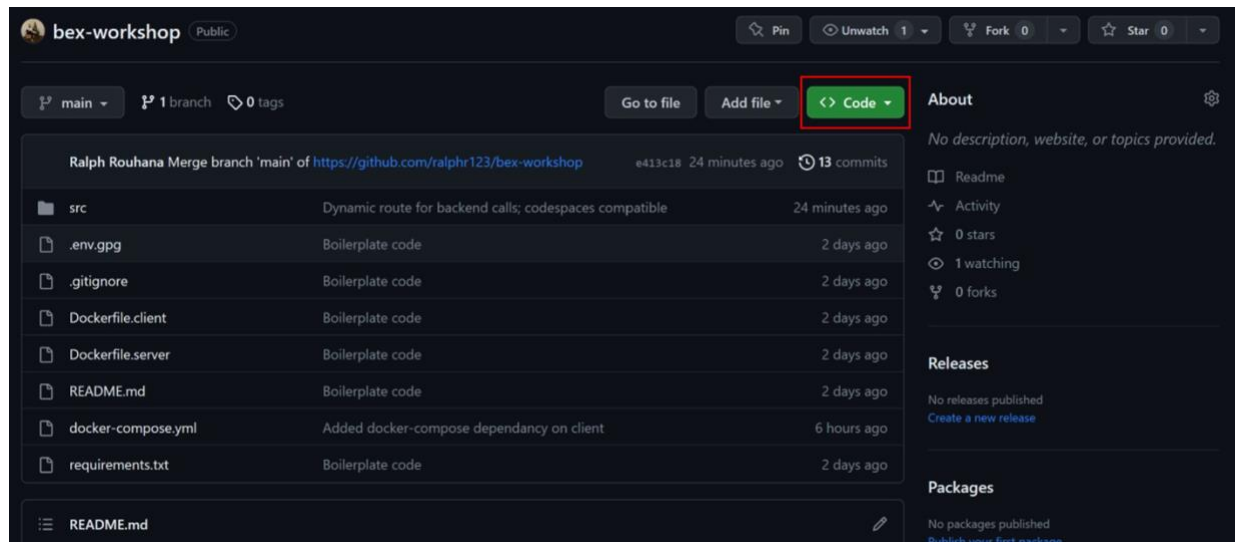
<https://github.com/ralphr123/python-openai-workshop>

You should see this page:

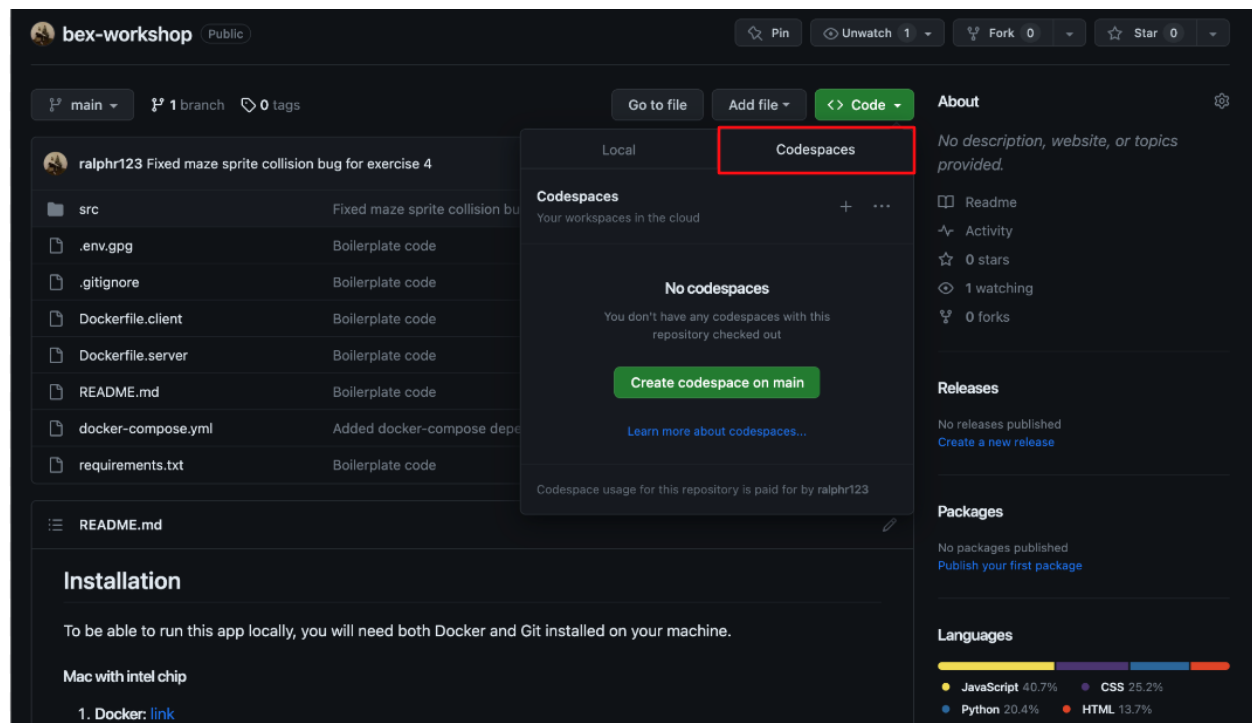


Once you're on the page and signed in, follow these steps:

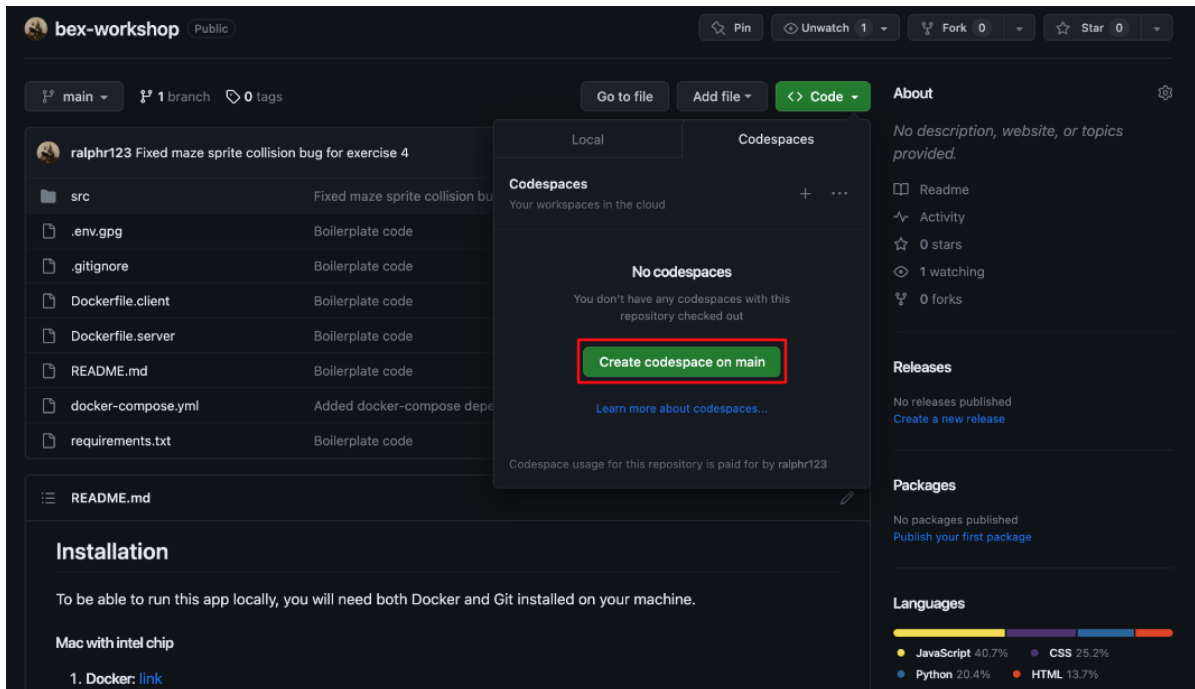
1. Click on the green *Code* button around the top right



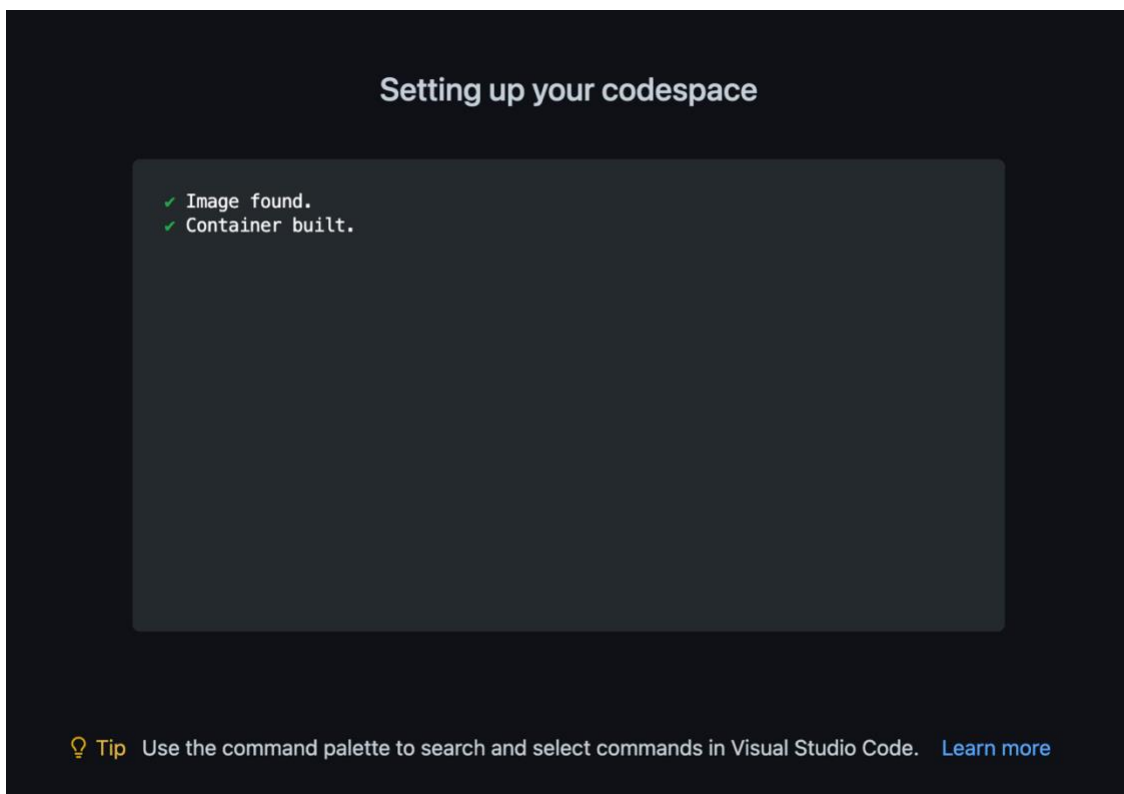
2. Click on the *Codespaces* tab



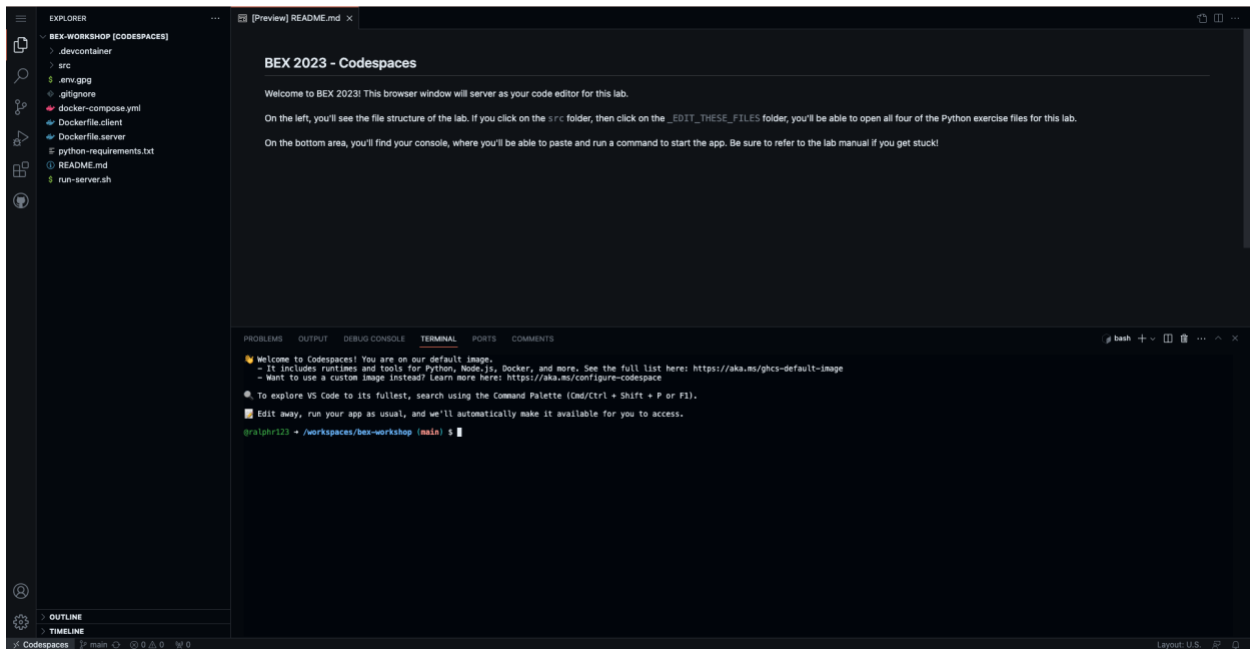
3. Click on *Create codespace on main*



4. Wait for the codespace to setup

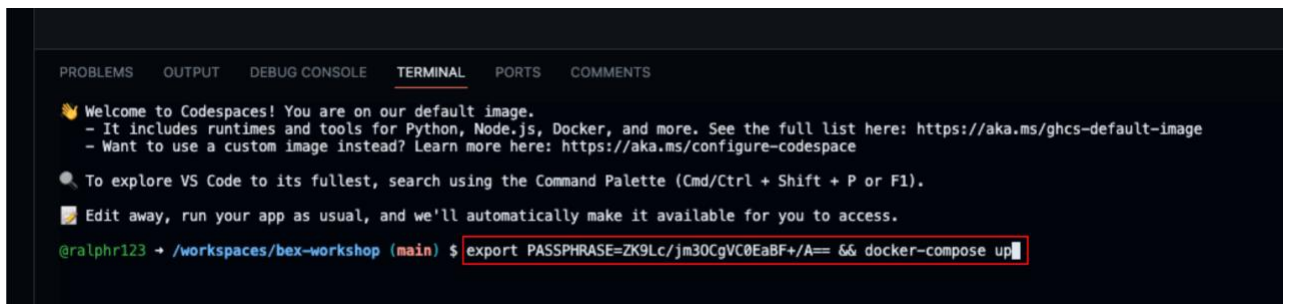


5. Once set up is complete, you should see this screen:

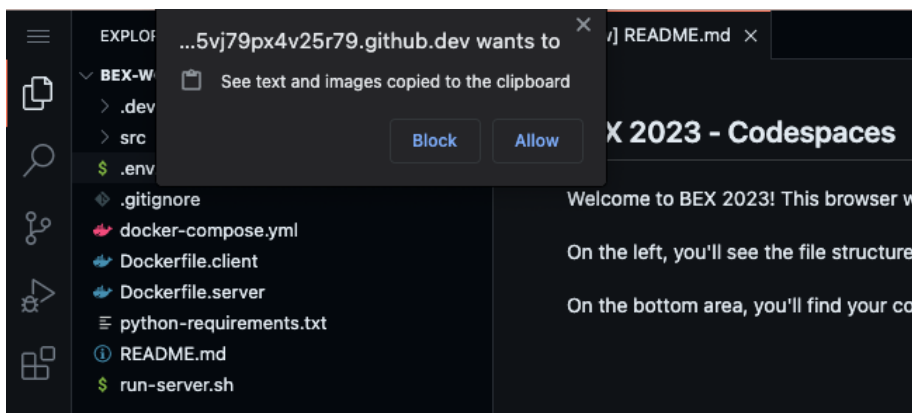


6. Paste the following command into the bottom terminal area

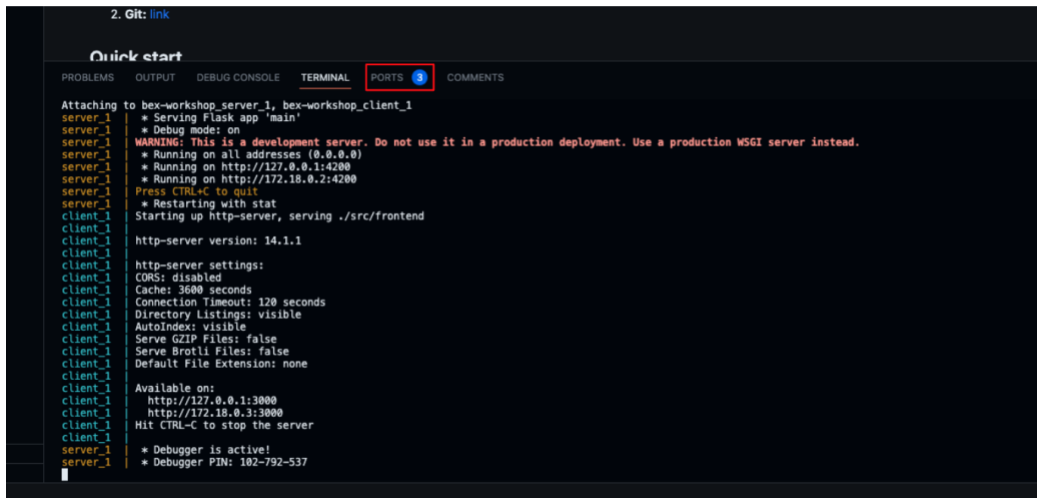
export P=<passphrase> N=<number> && docker-compose up



7. Click *Allow* if you get a paste permission popup

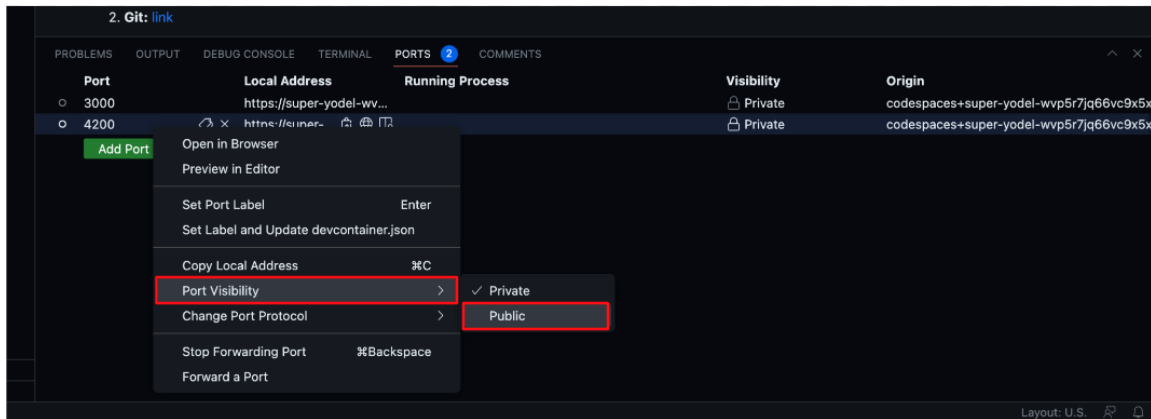


8. Once the command finishes, you will see a popup in the bottom right. In the console area, click on the *PORTS* tab:

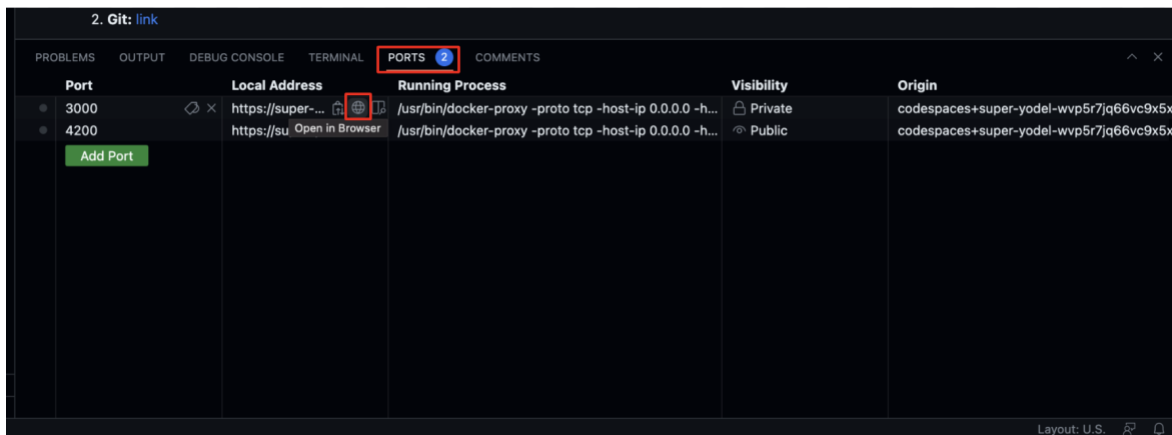


```
2. Git: link
Quick start
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
Attaching to bex-workshop_server_1, bex-workshop_client_1
server_1 * Serving Flask app 'main'
server_1 * Debug mode: on
server_1 WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
server_1 * Running on all addresses (0.0.0.0)
server_1 * Running on http://127.0.0.1:4200
server_1 * Running on http://172.18.0.2:4200
server_1 Press CTRL+C to quit
server_1 * Restarting with stat
client_1 Starting up http-server, serving ./src/frontend
client_1 http-server version: 14.1.1
client_1 http-server settings:
client_1 CORS: disabled
client_1 Cache: 3600 seconds
client_1 Connection Timeout: 120 seconds
client_1 Directory Listings: visible
client_1 AutoIndex: visible
client_1 Serve GZIP Files: false
client_1 Serve Brotli Files: false
client_1 Default File Extension: none
client_1 Available on:
client_1 http://127.0.0.1:3000
client_1 http://172.18.0.3:3000
client_1 Hit CTRL-C to stop the server
server_1 * Debugger is active!
server_1 * Debugger PIN: 102-792-537
```

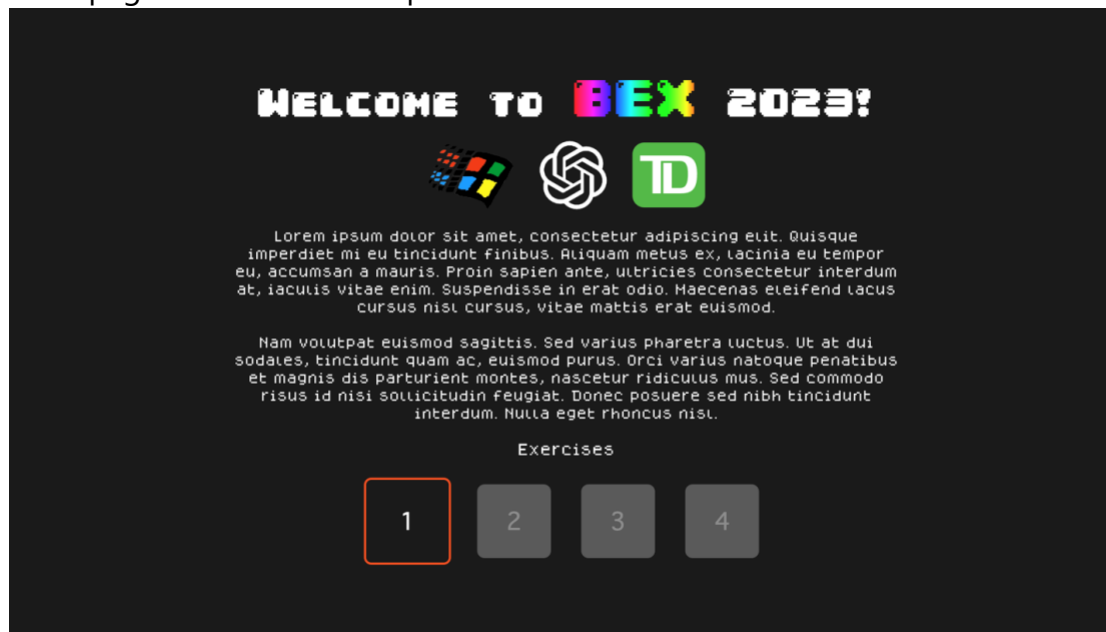
9. Right click on port 4200, click on *Port Visibility*, then click on *Public*



10. Stay on *PORTS*. Click on the globe icon for the top row (port 3000)

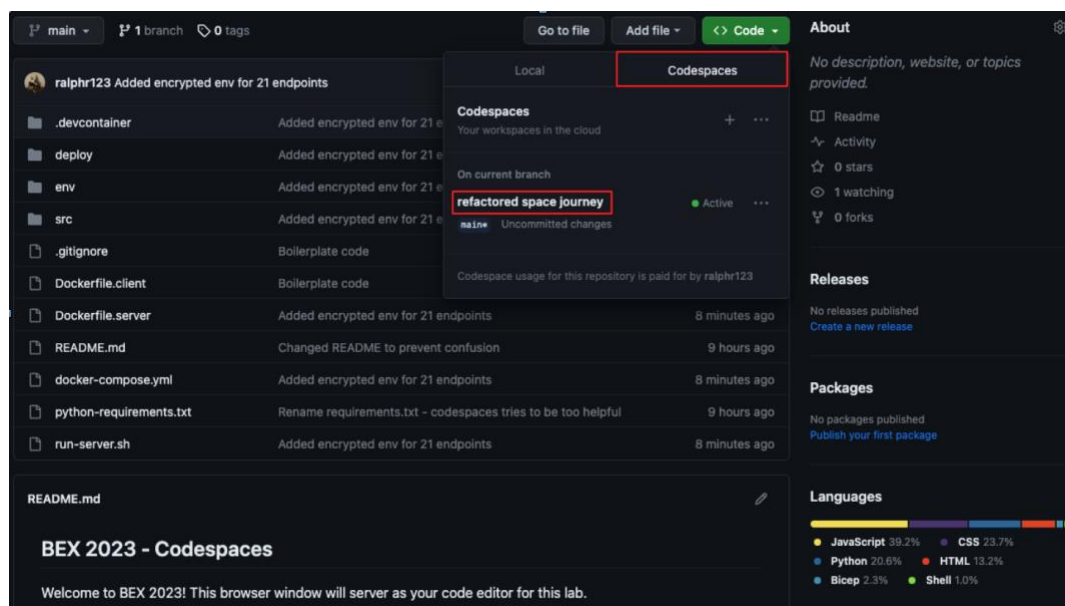


10. A page like this should open:



If so, you're all set! Keep both this tab and the code editor open throughout the lab. Otherwise, feel free to ask for help. If you accidentally close the bottom terminal area, no problem, you can open and close it with the top left menu, Ξ > View > *Terminal*.

If you decide to take a break, no problem, your code will save. If you accidentally close your tab, you can reopen the code space by going to the [repo](#), clicking on *Code > Codespaces > Codespace name*. **You may need to repeat steps 6-10 exactly to rerun the app after restarting the codespace.**



Local Development

If you would prefer to run this lab locally, another suitable option is by using Docker Desktop. **This is a separate alternative from all of the steps above!** Running the lab locally is more involved and requires more steps and installations. To be able to run this app on your computer, you will need both Docker and Git installed on your machine. To check if you have each installed, run "docker" and "git" in your terminal separately. If you receive an output that includes "not found" or "not recognized", then the software needs to be installed.

You will also need a code editor of your choice to write code with. The recommended option is [Visual Studio Code](#).

Windows

WSL

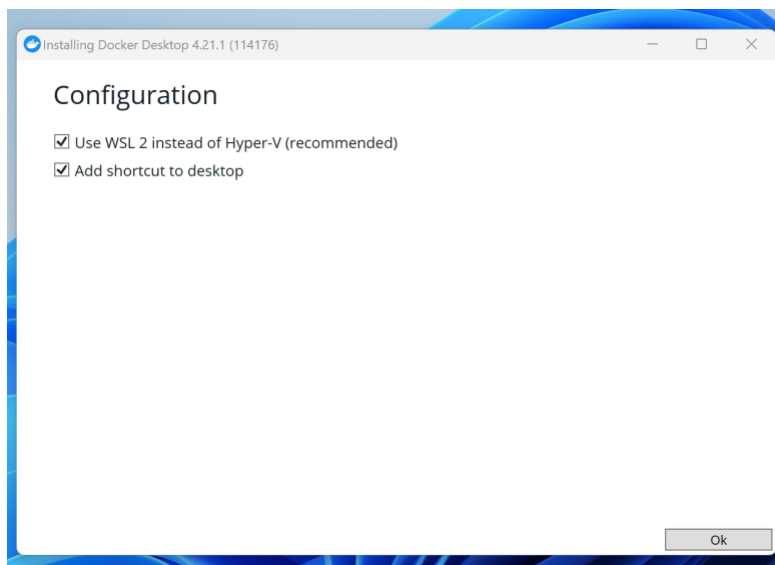
WSL (or Windows Subsystem for Linux) allows a Windows computer to behave like a Linux machine without the need for a separate device. It is required for Docker to be able to run.

- Open "command prompt"
- Run the following command: *wsl --install*
- Wait for every step to complete (reach 100%)
- Close your terminal (important)

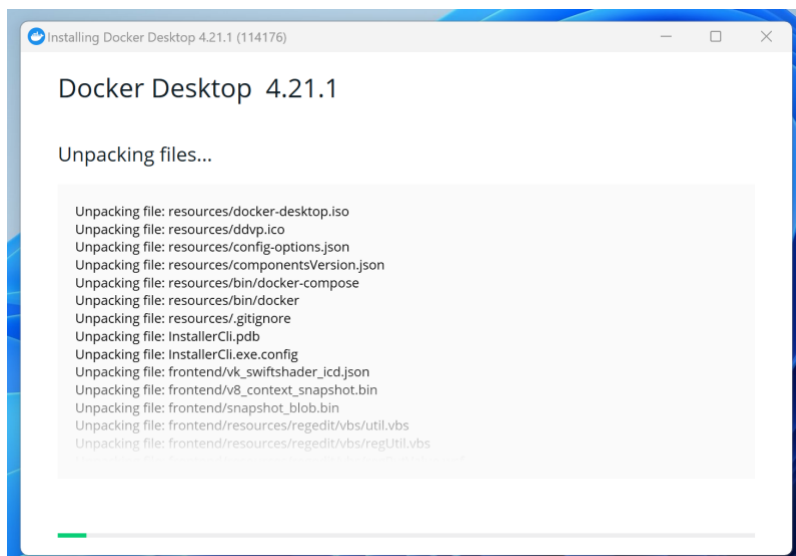
Docker

Docker is a service that allows developers to run software on containers. A container is a separate environment for software to run in, separate from your computer's files. This gives us the benefit of not worrying about different bugs and issues occurring on different computers and operating systems; the lab will behave the same way on every machine.

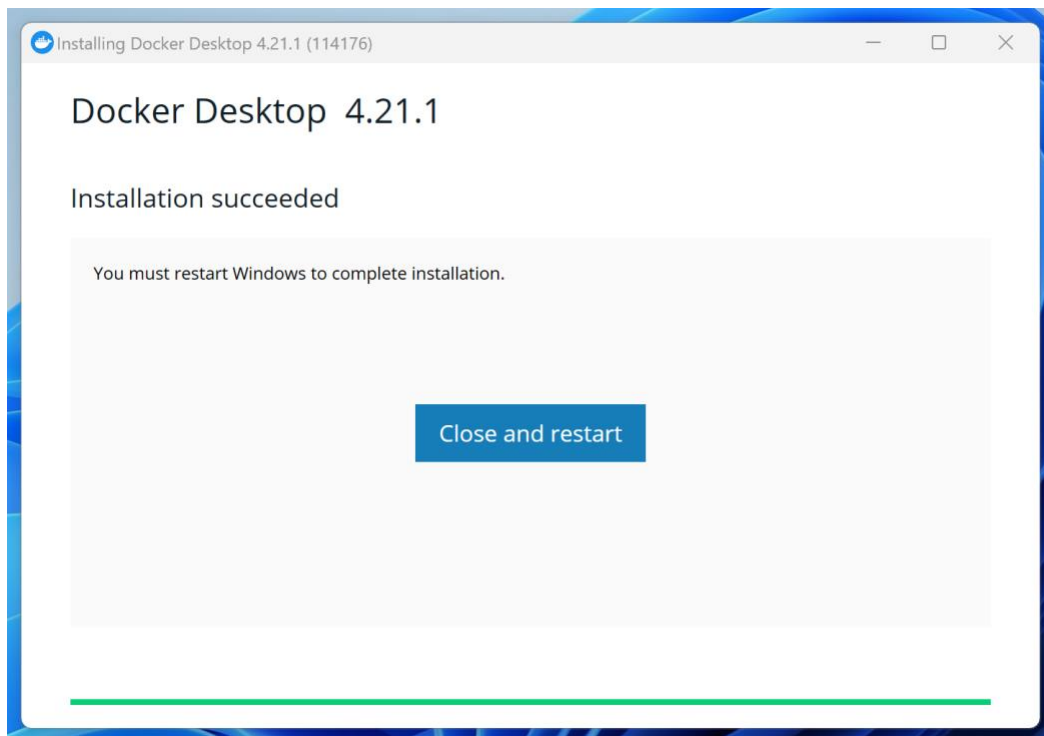
- Download Docker using [this link](#)
- Click on the downloaded file (then click yes on the popup if prompted).
- Check both boxes and click *Ok*:



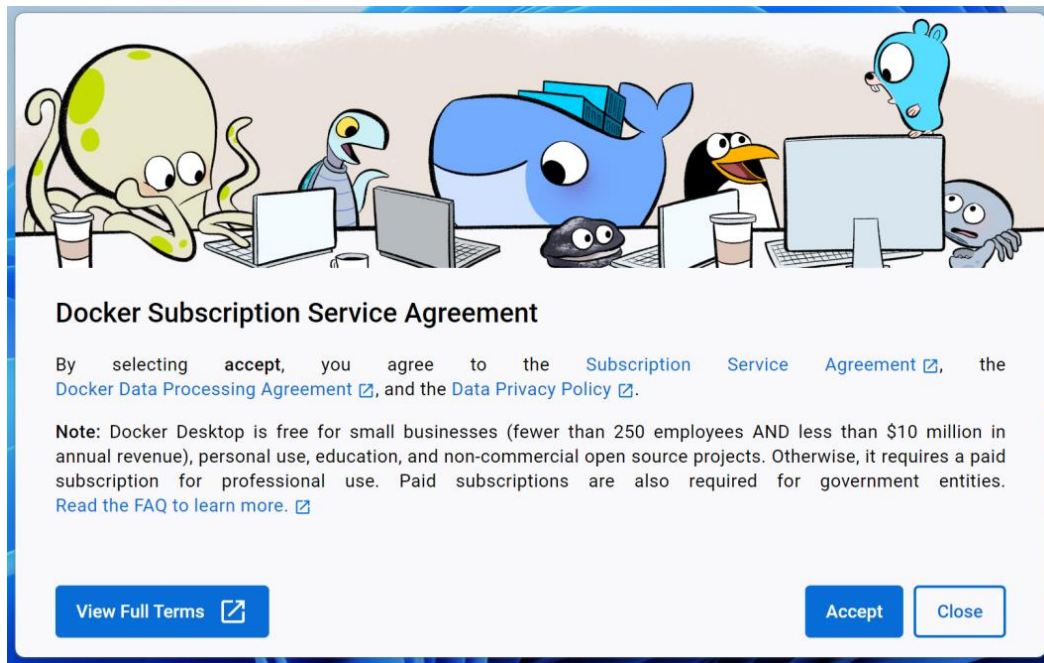
- Wait for the installation to complete



- Press close and restart, this will restart your computer



- Once you've restarted, open Docker from the start menu and click Accept

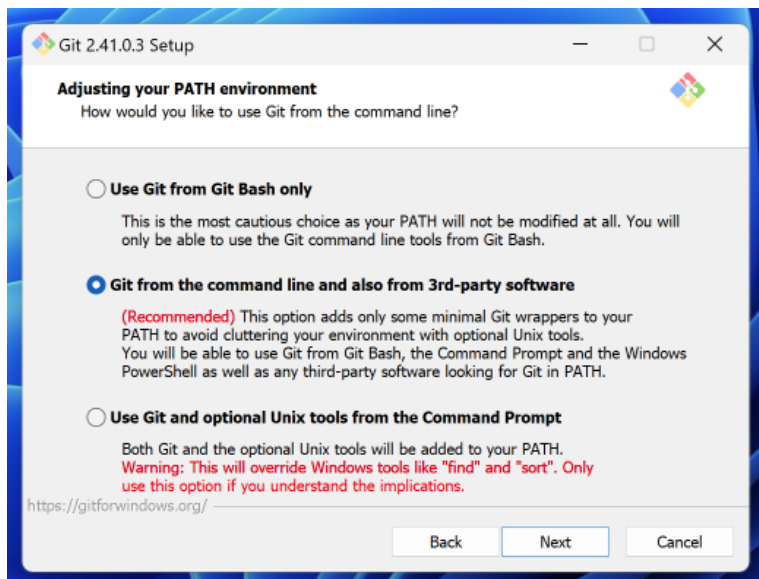


- If you're offered survey questions, press Skip
- You're done! You can minimize the window, but don't close it for the duration of the lab

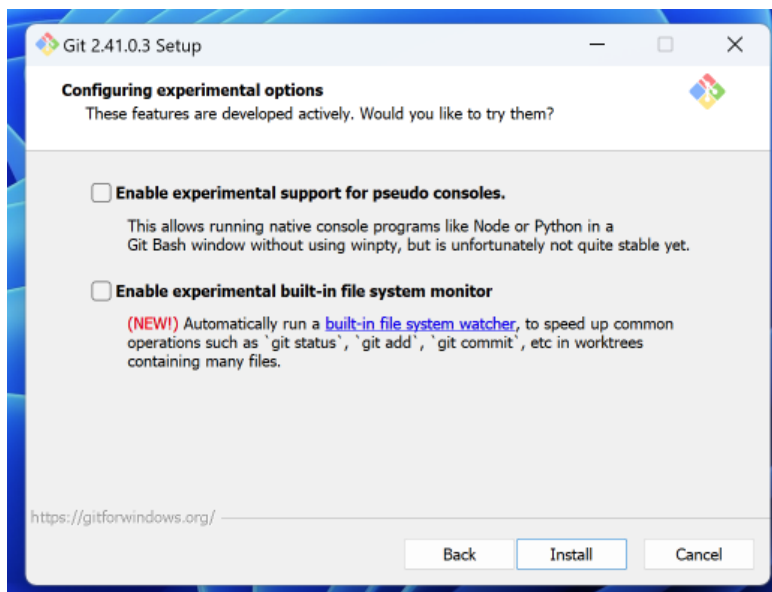
Git:

Git is a software that allows developers to collaborate on the same piece of code. It offers features such as keeping track of change history, managing different versions of the code, and more. You'll be using it to download the starter code for this lab.

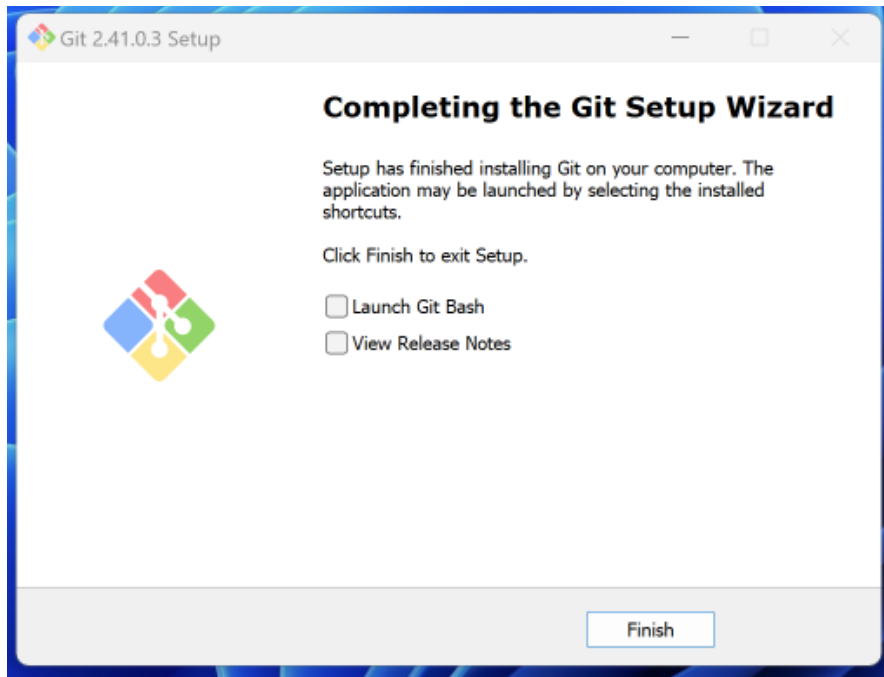
- Download git using [this link](#)
- Click on the downloaded file (then click yes on the popup if prompted).
- Keep clicking next until you reach the "Adjusting your path environment" page. Make sure "Git from the command line" is selected".



- Keep clicking next. When you reach the final step, click install.



- Wait for it to finish installing and press finish when it's done



- All done! You can now use Git from the command line.

Mac

How to know if you have an Intel or M-series chip

- Click on the Apple logo in the top left of the screen
- Click on "About This Mac"

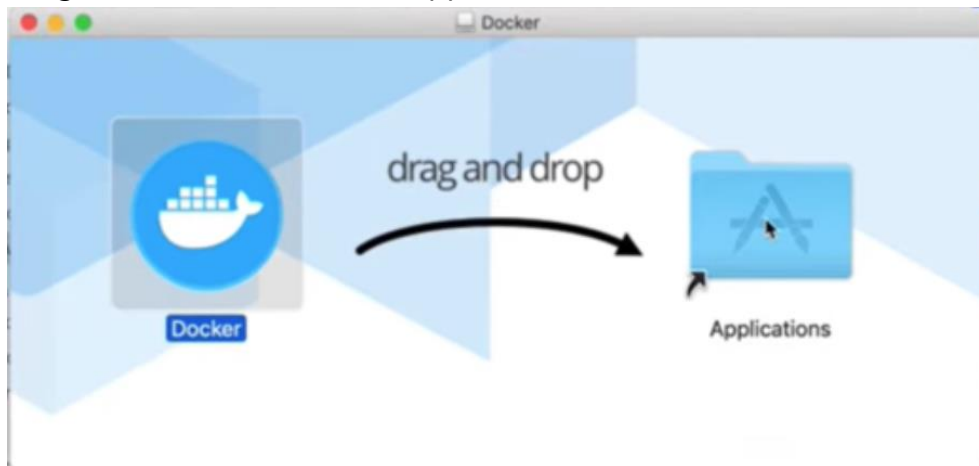


- Under the "Chip" or "Processor" label,
 - o If it includes M1/M2, then it is an M-series chip
 - o If it includes Intel Core, then it is an Intel chip



Docker

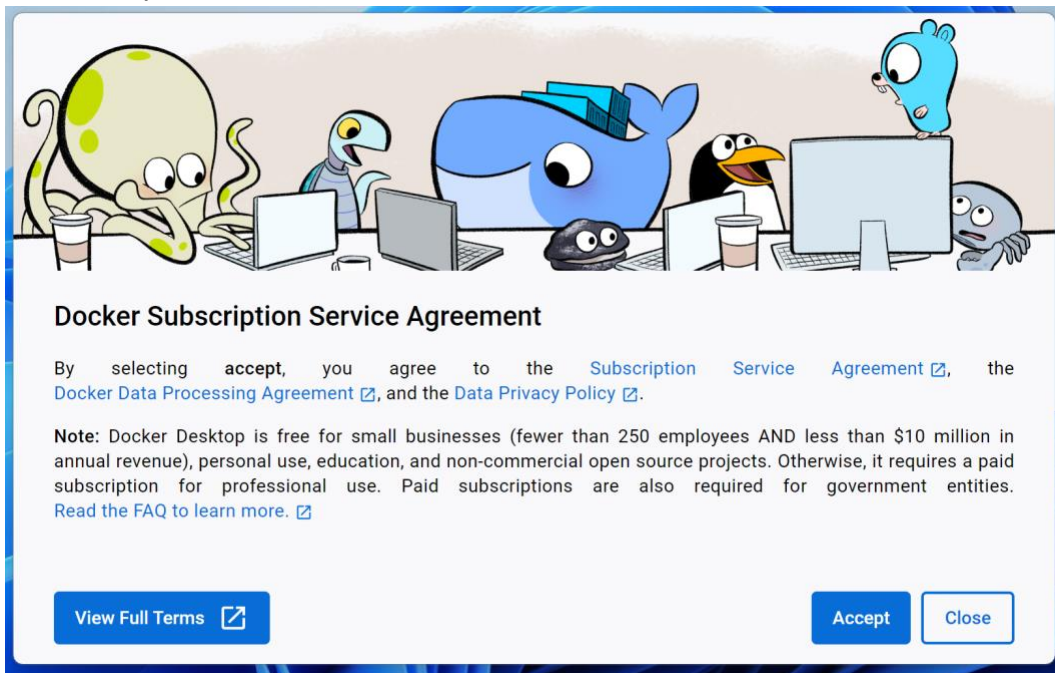
- Download Docker using:
 - o [This link](#) for Mac with an Intel chip
 - o [This link](#) for Mac with an M-series chip
- Open the installer by double-clicking the downloaded .dmg file
- Drag the Docker icon to the Applications folder and wait for it to complete



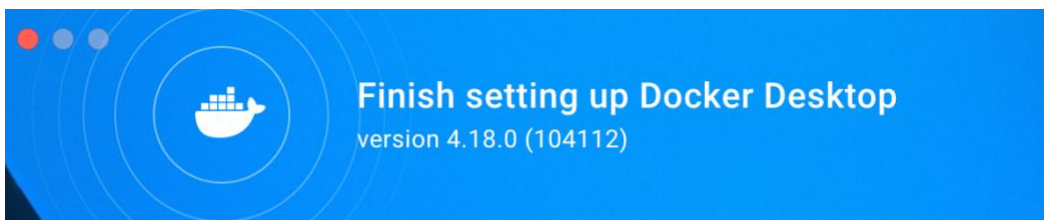
- Once complete, close the drag window and open Docker from applications
 - o You can go to your Applications folder or find it using search (cmd + space)
- If it prompts you for your password, fill it in and press Install Helper



- Click accept to the terms and conditions



- Press Finish



Complete the installation of Docker Desktop. The configurations below can be changed later in Settings.

- ☒ **Use recommended settings (Requires password)**
Docker Desktop automatically sets the necessary configurations that work for most developers.
- ☐ **Use advanced settings**
You manually set your preferred configurations.

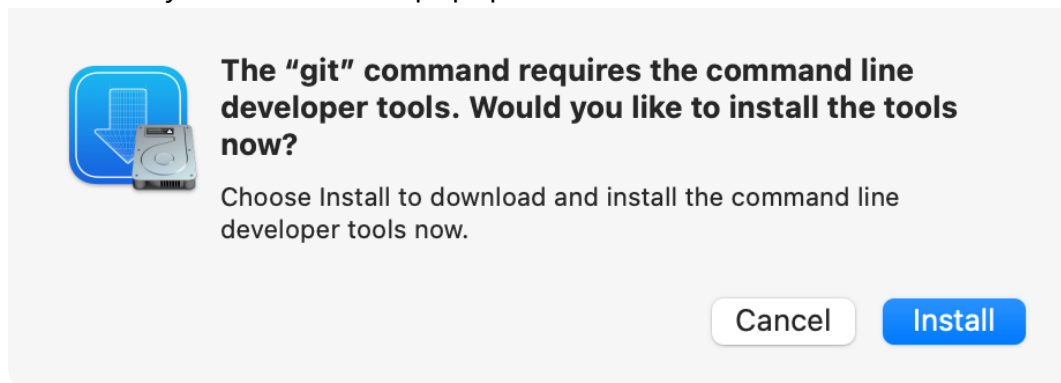
Finish

- If you're offered survey questions, press Skip
- Done! Feel free to minimize the window, but don't close it

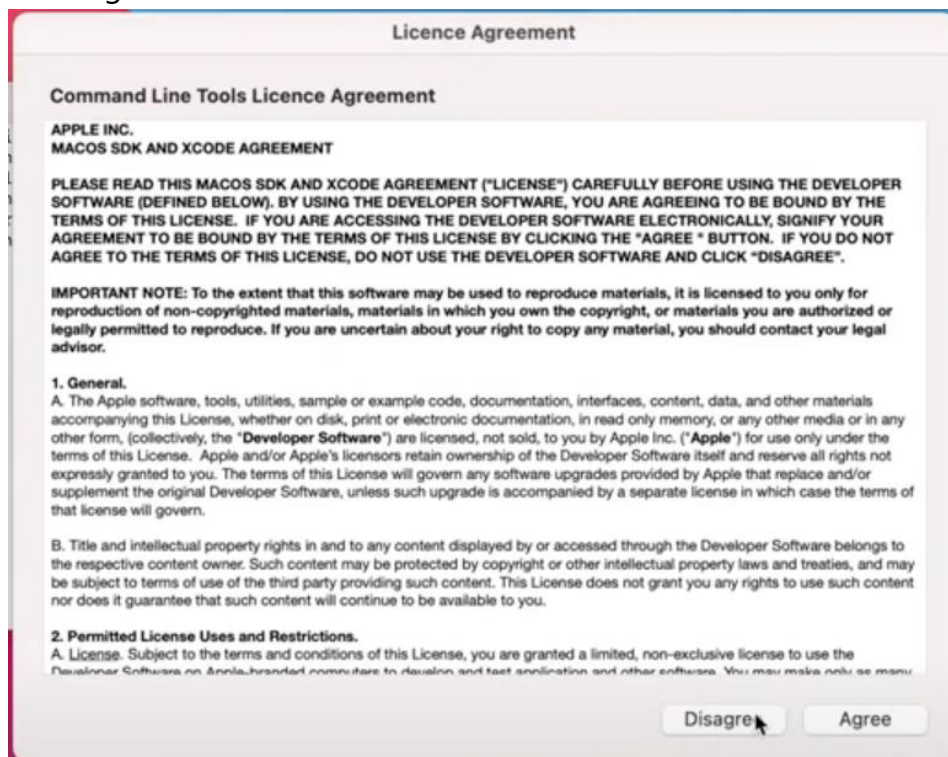
Git

The easiest way to install git on Mac is by using the command line.

- Open "Terminal" (press cmd + space and search for "terminal")
- Run `git --version` in the command line (type and press enter)
- If you receive a version number as an output, git is already installed
- Otherwise, you will receive a popup like this. Click Install.



- Click Agree



- Wait for the installation to complete, then restart the terminal.

- Restart the terminal, then run `git --version` again. If you receive a version number, you're all set! Otherwise, ask for help.

Start the app on your computer (Windows and Mac)

Once you have Git and Docker installed using the steps above, you can now run the application on your computer.

1. **Make sure Docker is open, if not then open it.** Keep it open and running in the background. You may minimize it.

2. **Clone the app (you only need to do this once):**

1. Open "terminal" on mac or "command prompt" on windows
2. Inside the terminal, run this command (paste it and press enter):

`git clone https://github.com/ralphr123/bex-workshop && cd bex-workshop`

3. **Run the app:** In the same terminal, run the following command:

1. For windows:

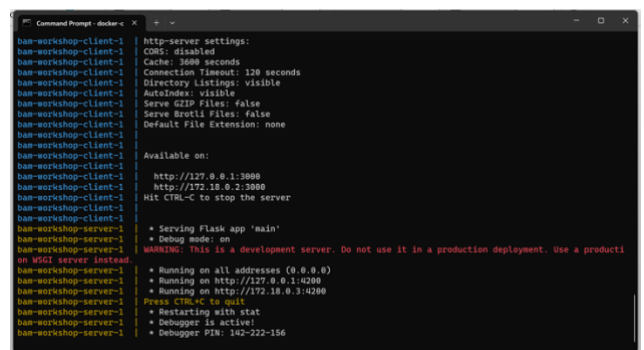
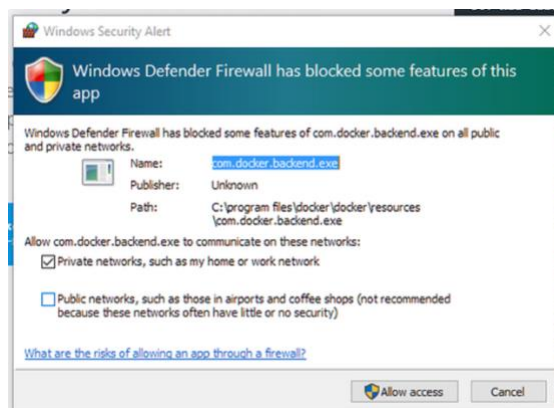
`set P=<password> && set N=<integer> && docker-compose up`

2. For Mac/Linux:

`export P=<password> N=<integer> && docker-compose up`

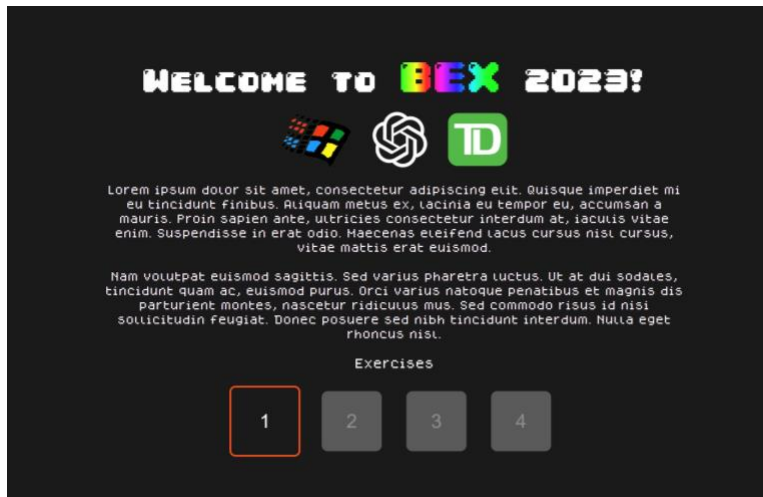
4. **Wait for the terminal to stop loading.** This may take several minutes or more.

On windows, you might get a windows defender popup. Click *Allow access*.



You will see **red text** once the app is loaded and ready:

Once its ready, open <http://localhost:3000> on your browser to access the app. If you see a page like the following, you're all set up! Otherwise, free to ask for help.



Once you see this page, all setup is complete. Any changes you make to the code will automatically be reflected on website itself, so there is no need to restart the app or refresh the page to test your changes. If you do need to stop the app, select the terminal window with your cursor and press CTRL + C. When the app is stopped, it must be rerun with step 3 to see this page again.

Once the app is running, you will need to open the folder with your code editor. If you're using Visual Studio Code, open a new window, press *File > Open Folder* on the top toolbar, and select the folder that you cloned in step 2.