Installation and Setup for Windows

For Internet Computer Web3 Application Development

Requirements

- **Windows 10 or higher (version 2004 or higher). Build 19041.xxx or higher.**
- **64-bit machine (System type x64 based PC)**

Steps

WSL

1. Find the Windows **PowerShell** in your Start menu and run it as the **Administrator**.

2. WSL is the Windows Subsystem for Linux and it will allow us to run command line commands in Windows. Here's more info from Microsoft:

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

3. As described in the docs above, we need to paste this command into **PowerShell** and hit enter:

```
wsl --install
```

4. Once that's done, you'll need to **restart** your computer.

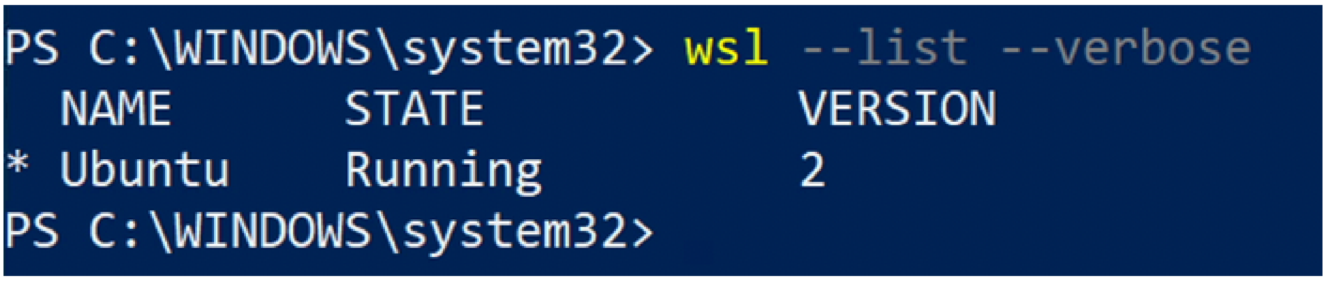
5. Upon restart you will be prompted to setup an ubuntu **username** and **password** and then you will have successfully installed WSL. (Keep a note of both of these pieces of information, you'll need it later on).

Note: when you type your password it will not show up, just make sure you know what you're typing!

6. To confirm that everything worked correctly, type the following command into PowerShell:

```
wsl --list --verbose
```

7. You should see it output something like this:



```
PS C:\WINDOWS\system32> wsl --list --verbose
  NAME      STATE           VERSION
* Ubuntu    Running         2
PS C:\WINDOWS\system32>
```

VSCode

8. Download and install the latest version of **VSCode** from here:

   https://code.visualstudio.com/

9. Install the **Motoko** language extension in VSCode (make sure it's from the **Dfinity** team, or just use the link below).

   https://marketplace.visualstudio.com/items?itemName=dfinity-foundation.vscode-motoko

10. Install the **Remote WSL** extension.

   https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl

Node

11. Search and open up **Ubuntu** from the Start menu.

12. Type the following command to install homebrew (Alternatively copy it from the homebrew website https://brew.sh/):

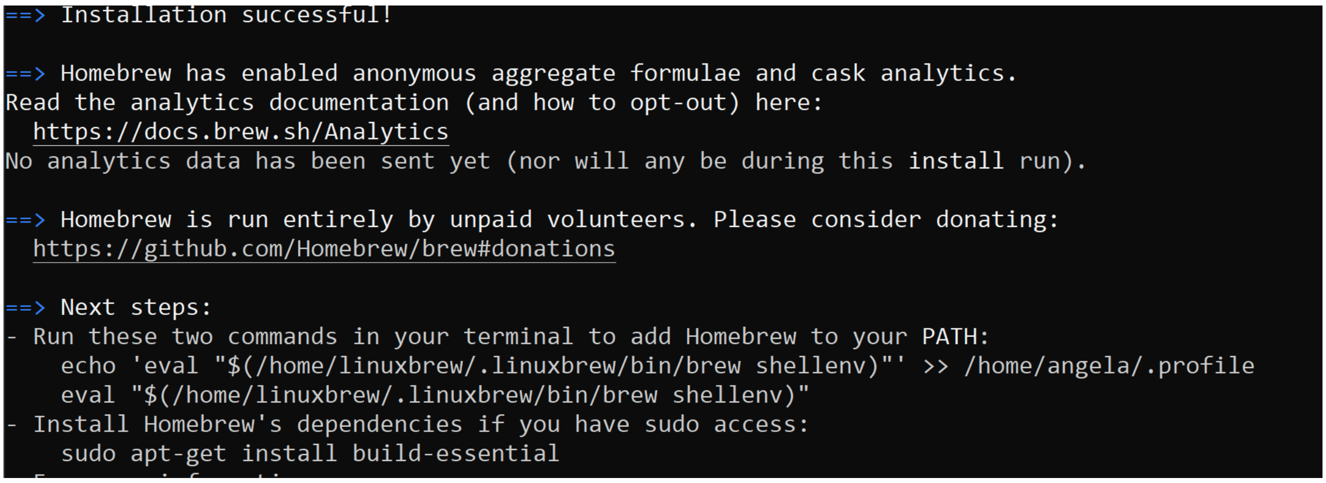
```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/in
stall.sh)"
```

Homebrew will make it easier for us to install other tools such as node. You might already have node installed on your windows system but because we're working with WSL, you'll need to install it on the linux system too.

13. When prompted enter the **password** for the user that you set previously in **step 5**.

14. The installer will tell you how to add brew to the **PATH**. Copy the commands they list and run them one by one in Ubuntu.

e.g.



```
==> Installation successful!

==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
  https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run).

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
  https://github.com/Homebrew/brew#donations

==> Next steps:
- Run these two commands in your terminal to add Homebrew to your PATH:
    echo 'eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"' >> /home/angela/.profile
    eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
- Install Homebrew's dependencies if you have sudo access:
    sudo apt-get install build-essential
```

15. Also run the command under the line **“Install Homebrew’s dependencies if you have sudo access”**:

```
sudo apt-get install build-essential
```

16. Check that everything worked by typing the command:

```
brew -version
```

If you see a version show up then everything was installed.

17. Install node using homebrew with the following command:

```
brew install node@16
```

18. Once it’s done check that it worked with:

```
node -version
```

NOTE: If you have another version of node installed (e.g. previous version or windows version) then you need will to link the version we just installed to homebrew (use the command: brew link node@16)

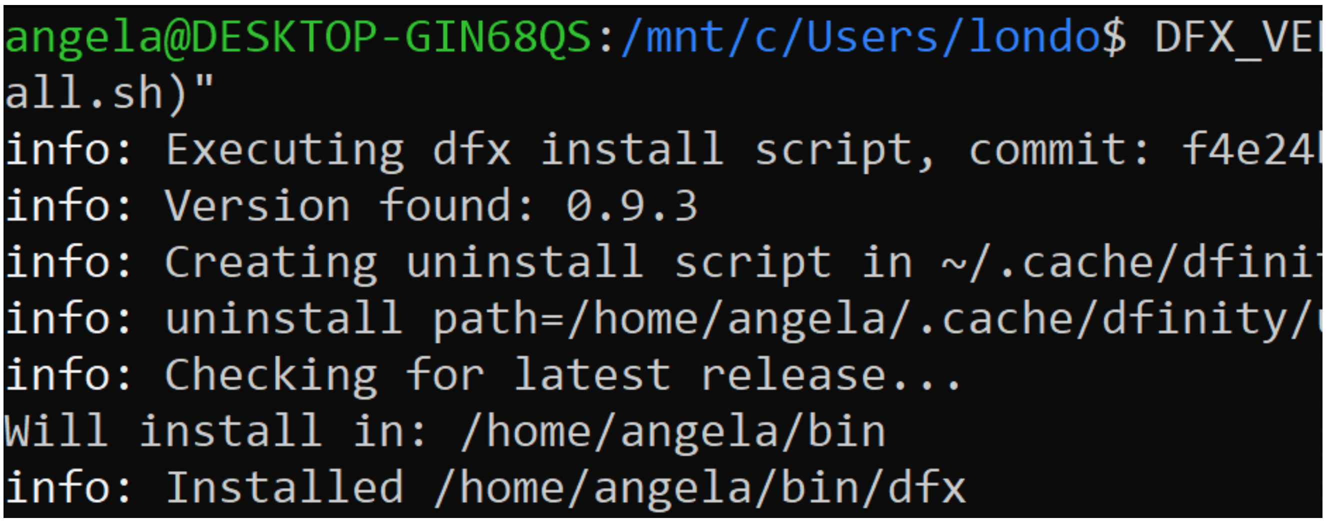
DFX

19. Open up Ubuntu from the Start menu

20. Copy the following command and paste it into your terminal and hit enter to install DFX.

```
DFX_VERSION=0.9.3 sh -ci "$(curl -fsSL
https://sdk.dfinity.org/install.sh)"
```

After DFX has installed it will tell you where it was installed. e.g.



```
angela@DESKTOP-GIN68QS:/mnt/c/Users/londo$ DFX_VE
all.sh)"
info: Executing dfx install script, commit: f4e24
info: Version found: 0.9.3
info: Creating uninstall script in ~/.cache/dfini
info: uninstall path=/home/angela/.cache/dfinity/
info: Checking for latest release...
Will install in: /home/angela/bin
info: Installed /home/angela/bin/dfx
```

e.g. in my case, it tells me that it has been installed in **/home/angela/bin/dfx**

21. Copy the installation path you got from the last step and replace <REPLACE WITH YOUR INSTALLATION PATH> from the command below (You can use Notepad for this):

```
export PATH=$PATH:<REPLACE WITH YOUR INSTALLATION PATH>
```

E.g. in my case it would be export PATH=$PATH:**/home/angela/bin/dfx**

22. Paste the formatted command from the previous step and hit enter.

23. Check that it has been added by running:

```
echo "${PATH//:/$'\n'}"
```

24. Check that dfx has been successfully installed with the following command:

```
dfx --version
```

Notes

- **We're going to work with dfx 9.0.3 so that we are all on the same version and you don't get any surprises. Even if it prompts you to upgrade dfx, don't do it!**

Test Everything Worked by Creating and Deploying your First DApp

Create the Default Hello DApp

1. Open up **Ubuntu** from the start menu and create a new folder called **ic-projects** using the following command:

```
mkdir ic-projects
```

2. Change directory into that folder using the command:

```
cd ic-projects
```

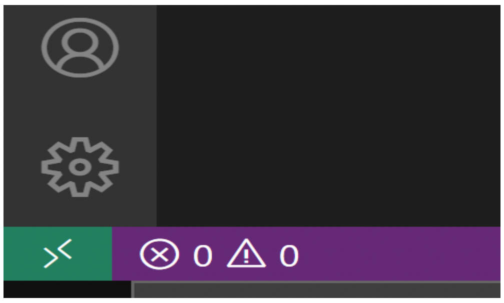
3. Inside this ic-projects folder, we're going to create our first Internet Computer DApp using the following command:

```
dfx new hello
```

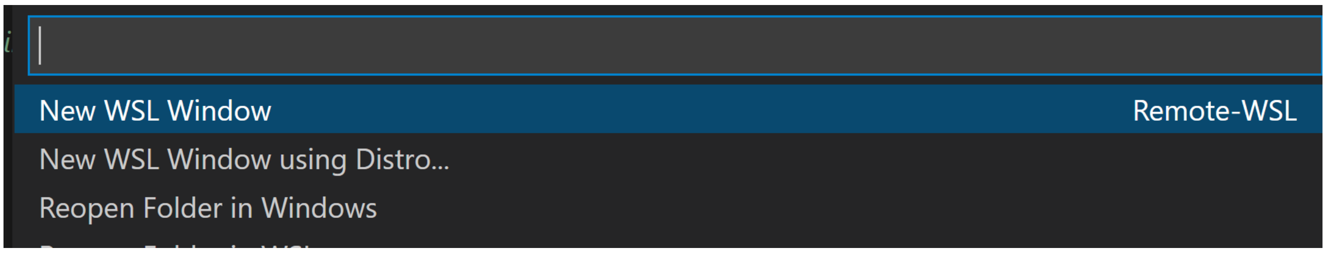
4. You can see this new project and folders by running the following command:

```
explorer.exe .
```

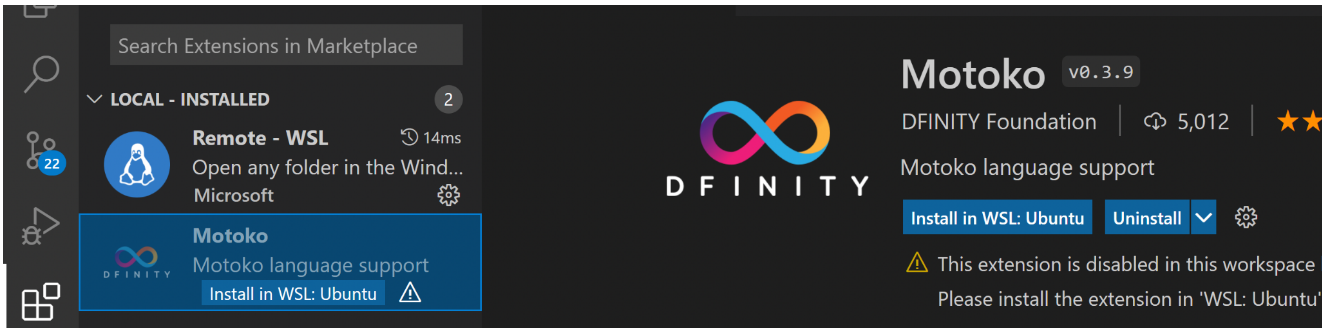
5. Open up VSCode and click on the green icon on the bottom left. It looks like this:



6. Select **New WSL Window**



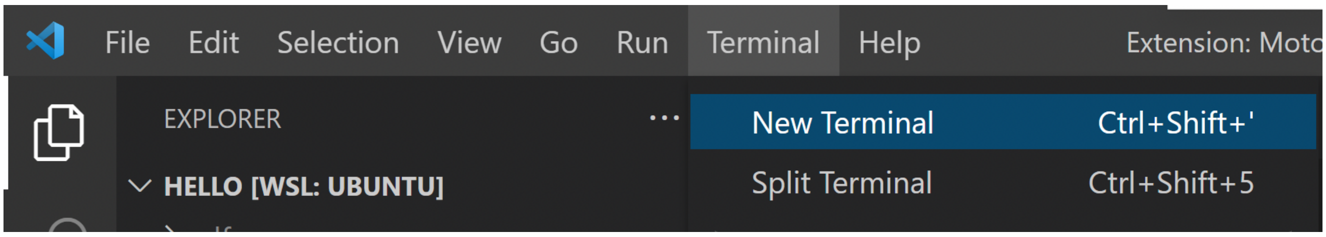
7. Inside the new window go to your **Extensions** panel and select the **Remote WSL** extension, click on **Install in WSL: Ubuntu**



8. Now take a look through the files inside the **src** folder. The main.mo is the Motoko file that we'll be writing most of our code in.

Deploy the DApp

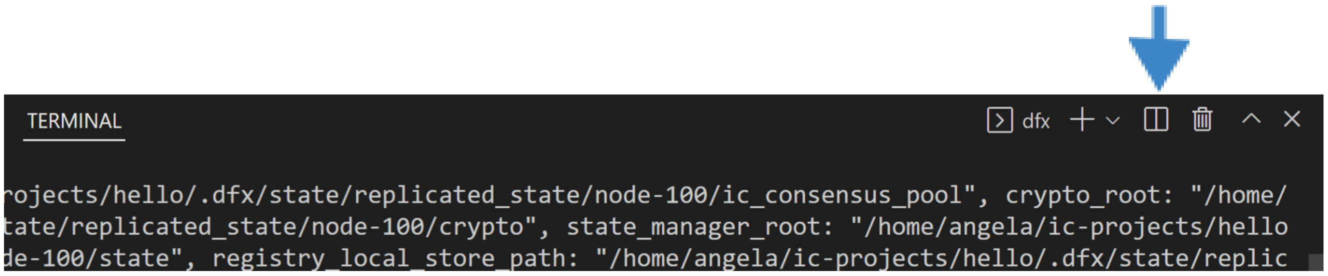
9. Go to Terminal → New Terminal



10. In the Terminal, run the following command to start the local dfx

```
dfx start
```

11. Once you see the line INFO Starting server. Listening on blah blah blah, then split out another terminal using the button shown below:



```
rojects/hello/.dfx/state/replicated_state/node-100/ic_consensus_pool", crypto_root: "/home/
tate/replicated_state/node-100/crypto", state_manager_root: "/home/angela/ic-projects/hello
de-100/state", registry_local_store_path: "/home/angela/ic-projects/hello/.dfx/state/replic
```

12. In the new terminal pane, run the following command to deploy your hello project:

```
dfx deploy
```

13. Finally, once that's done, run the following command:

```
npm start
```

14. Now you're ready to see your hello project, open up your browser and go to:

http://localhost:8080/