V 1.1

How to set up Linux Ubuntu on a PC with WSL

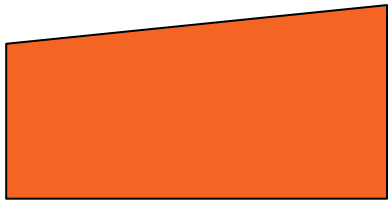
This guide is for installing Linux Ubuntu 18.04 LTS in Windows 10, using Windows Subsystem for Linux (WSL). Some additional info is provided to install a complete development environment with Visual Studio Code, GitHub desktop, DBeaver, PostgreSQL, Postman and NodeJS.
Please note that some of the modifications could potentially brick your machine or necessitate a new install of Windows if done without precautions. Proceed at your own risk, and backup your data/OS.

Why Ubuntu?

Linux is more or less just the OS Kernel. Different orgs or companies use this kernel to put together a full OS. Some different flavors, called distributions, are: Debian, Red Hat, Mint, Linux Lite and a lot of others.

Linux is extensively used by deloppers and tech companies. It is the OS of reference for supercomputers and servers. A lot of cell phones and IoT devices run on Linux.

Ubuntu is distributed and maintained by the company Cannonical. It is arguably one of the two or three most popular distributions of Linux. This means that bugs are fixed promptly, updates are regular and frequent and, more importantly, most of the applications found on a Mac running OSX or a PC/Win will exist on Ubuntu/Linux. Sometimes it is an equivalent.



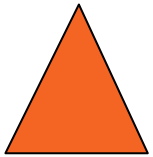
Windows Sub-System for Linux

In short: WSL.

It allows the installation and use of Ubuntu 18.04 within Win10. It is essentially a dedicated local VM. It makes it possible to run Visual Studio Code, NodeJS and Github Desktop, ESlint (not global) and Testem (not global either).

It may encounter some difficulties running global packages. PostgreSQL can run in the WSL “bubble” with some constraints.

It is a nice environnement, and it is rather easy to set up. Main caveat is it is slower than a native Linux environment. Some global installations will not work, though a lot of libraries and resources are best installed and used on a per-project basis.



Windows requirements

To operate Windows Sub-system for Linux, the machine must run a Windows 10 version different from the Home version.

A latest build is also better. It is advised to run Fall Creator or later (Build 16215 and above).

First steps

What you need:

1 USB drive 32GB,

DVDs or some more USB drives.

Before any action, prepare a Windows 10 system image with the 32GB USB drive.

Backup your files and documents, so you have them in case something goes wrong. Use some DVDs or other USB drives.

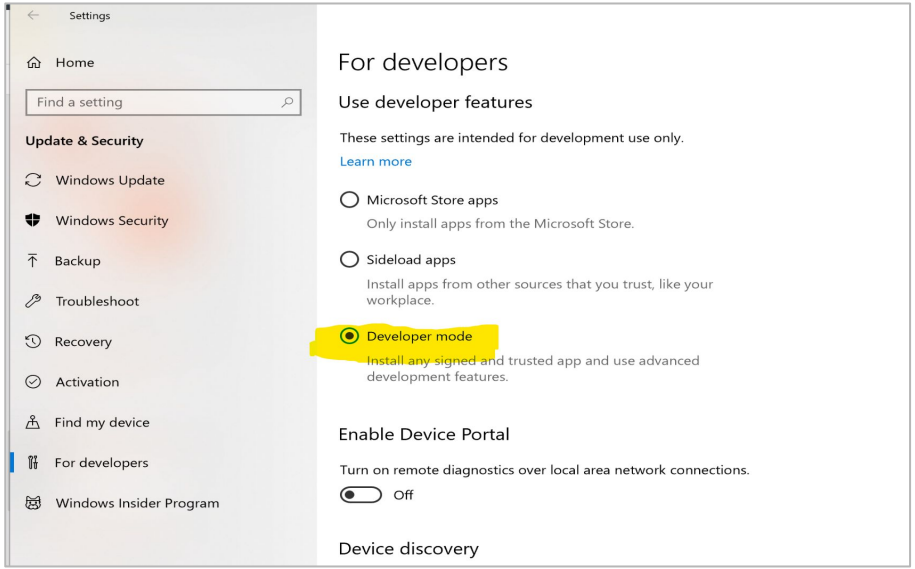
Initializing WSL

Useful links:

[Microsoft WSL Documentation]

First, enable the Developer mode in the settings:

Settings => Update and Security => For Developers => Click Developer mode ON.



Initializing WSL

Second, enable WSL through a Powershell command (official MSFT Docs) or through Windows features (2nd tutorial).

After a restart, install the chosen version of Ubuntu, preferably the latest LTS. MSFT Store provides different versions for free, as well as different distributions. Other distros may require different options and / or utility software. Some may not support all the features required by this environment. Restart again the computer

Pin the Ubuntu client to the Start menu. Other quick links will get there in time, it is a handy place to gather all the Dev tiles (see screenshot).

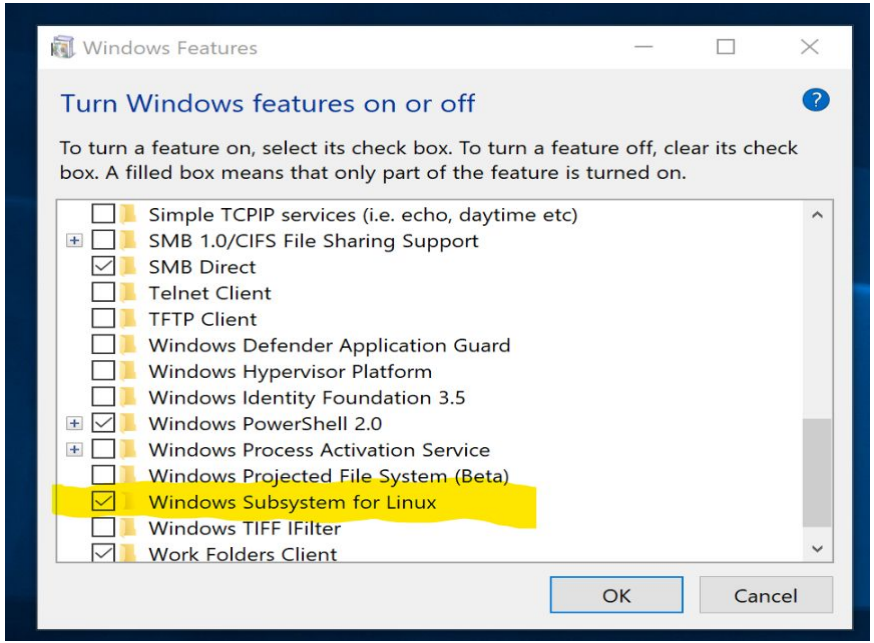
Useful links:

Microsoft WSL Documentation

WSL installation tutorial

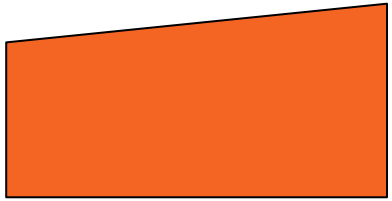
Initializing WSL

Turning WSL on through Features.

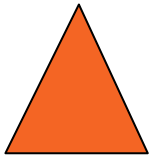


Useful links:

[Microsoft WSL Documentation]



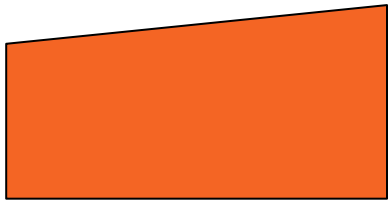
Bionic Beaver is alive!



At that point, you are pretty much done. Congratulations!

Ubuntu (or the chosen distro) is active on the Windows eco-system.

Now comes the Dev environment set-up.



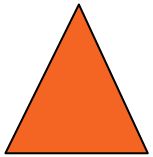
Useful links:

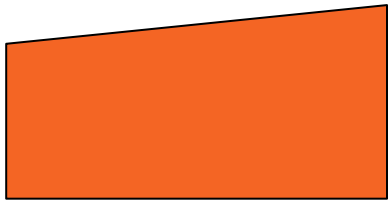
[Fullstack Academy Toolbox]

Linux for Fullstack Academy

There are some differences between the Toolbox from FSA and the implementation with WSL.

One important note: some software need to be installed in Windows, some in the Ubuntu client. **DO NOT TRY** to install Linux stuff through the PowerShell or Command Prompt terminal. Some really nasty effects will happen, with the potential of breaking down Win10.





Linux for Fullstack Academy - Git

Useful links:

[Fullstack Academy Toolbox]

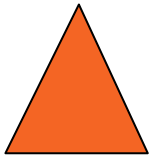
[Git Downloads]

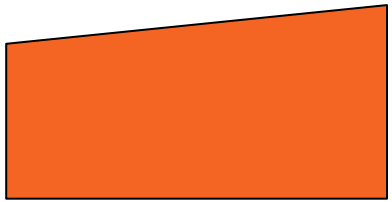
Git (version manager, hosted on GitHub) should be the first item on the list. And it is best installed on both sides, meaning in Win10 (download and install), AND in Ubuntu.

Open the brand new Ubuntu terminal and type:

```
User@pc~$    sudo apt-get install git
```

This will allow Git to run on both sides.





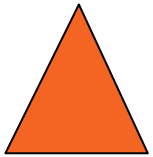
Linux for Fullstack Academy

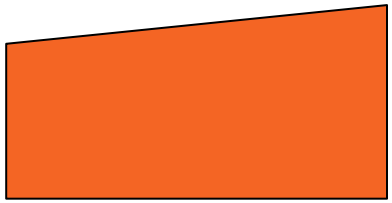
From experience, it is best to start with NVM and NodeJS, as presented in the next slide.

WSL does not allow GUI tools to run in WSL Bash. the work-around is to use Win10 tools or their equivalent and make them communicate through ports.

Useful links:

Fullstack Academy Toolbox



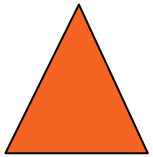


Useful links:

[Fullstack Academy Toolbox]

[NVM]

[NodeJS]

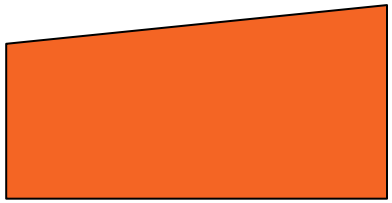


Installing NVM & NodeJS

There are a few ways to install NodeJS. For Ubuntu, and WSL, the most reliable fashion is through NVM (Node Version Manager).

Simply follow the directions in the 2 links on the left. It seems to be best to stick with a LTS version of NodeJS at the start. Having NVM will allow to manage and switch between different NodeJS versions.

Do not try to install NPM (Node Package Manager), as it comes already in with the NodeJS installation.

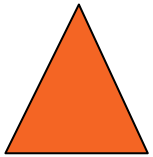


Useful links:

[Fullstack Academy Toolbox]

[Visual Studio Code - Download]

[GitHub Desktop]

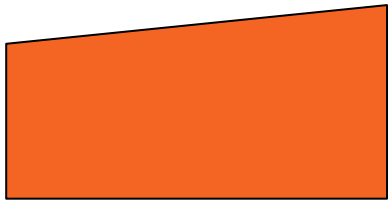


Installing Visual Studio Code

This is done on the Win10 side, not in the WSL-Ubuntu client.

Get to the download with the link, fetch the 64-bits version and install. Pin that tile to the Start Menu.

In the same way, install GitHub Desktop. It may not be necessary, but it is quite helpful when things get a bit hairy in Git.

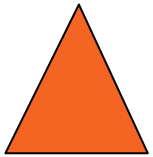


Useful links:

[Fullstack Academy Toolbox]

[Visual Studio Code - Download]

[GitHub Desktop]

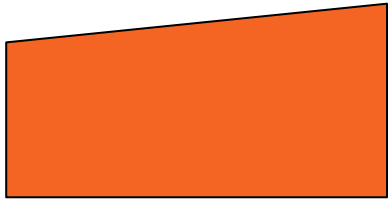


Installing Visual Studio Code

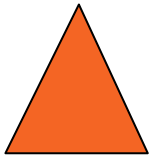
This is done on the Win10 side, not in the WSL-Ubuntu client.

Get to the download with the link, fetch the 64-bits version and install. Pin that tile to the Start Menu.

In the same way, install GitHub Desktop. It may not be necessary, but it is quite helpful when things get a bit hairy in Git.



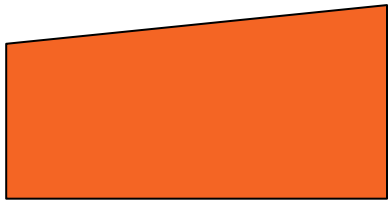
Hitting the limits in WSL



WSL, as good as it is, will not play nice with GUI tools, so no GNOME or Linux Unity.

It is also not too good at global installations. NodeJS passes muster, not ESLint or Nodemon , for instance. They will need to be installed on a per-project basis.

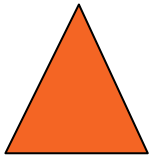
Finally, the Ubuntu 18.04 provided works around a Linux kernel v. 4.04, not the most recent version of it. It is best left alone, no need to update.



Useful links:

[Fullstack Academy Toolbox]

[ConEmu]

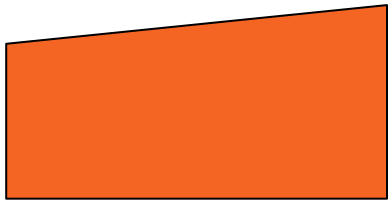


Installing some Tools

From previously, Homebrew, or more exactly its little brother for Linux (Linuxbrew), will not work well in WSL, so skip it.

ZSH and Oh-My-Zsh will work, if prefered. A terminal manager like ConEmu can also be used, it's just not completely necessary.

ESlint and Prettier can be installed, and Prettier will work globally, not ESlint. But it is important to install the dependencies and peer dependencies globally.



Useful links:

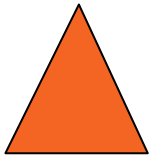
[Fullstack Academy Toolbox]

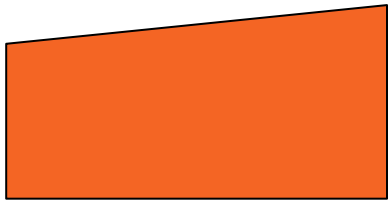
Configuring Visual Studio Code

The directions in the Toolbox are complete and should be followed. The 2 add-ons to VSC User Settings are:

```
...
"editor.formatOnSave": true,
"terminal.integrated.shell.windows": "C:\\WINDOWS\\System32\\wsl.exe"
```

The second rule will allow the WSL client to be the default terminal in VSC.





Useful links:

[Fullstack Academy Toolbox]

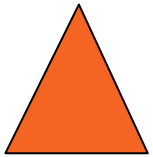
Configuring ESLint and Prettier

Install ESlint and the rest with:

```
Sudo npm install -g eslint eslint-config-fullstack eslint-plugin-react babel-eslint
```

Create ./eslintrc.json and prettierrc.js in the root of the Ubuntu-WSL terminal, as instructed in the Toolbox.

Once done, close the WSL window, restart and reopen.



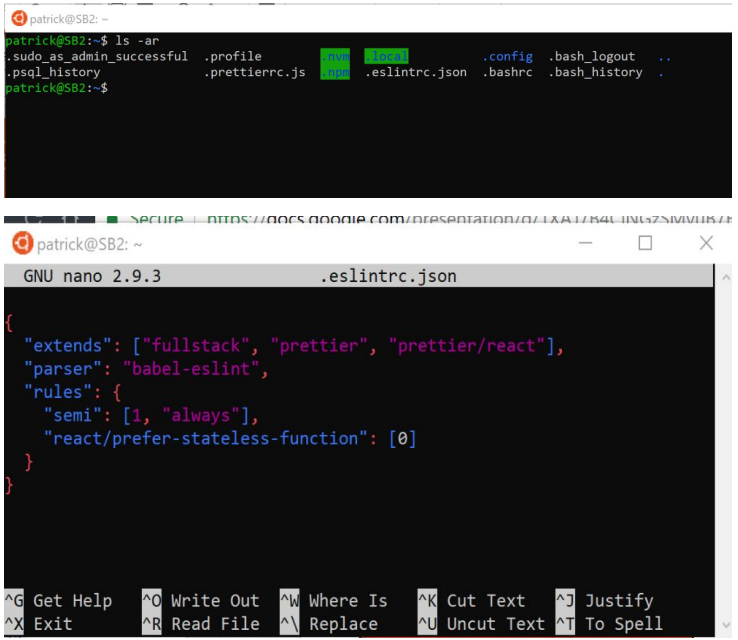
Configuring ESLint

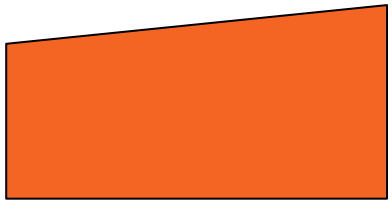
Useful links:

[Fullstack Academy Toolbox]

[Installing Postman]

This shows a complete .json file for ESLint:





Installing Postman

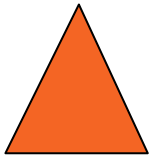
Install Postman, a really great tool for back-end development.

Here, the Win10 version will be installed, from the Win10 side, NOT the WSL-Ubuntu side. It works off the port served by the app, so it needs to simply plug in the same port.

Useful links:

Fullstack Academy Toolbox

Installing Postman

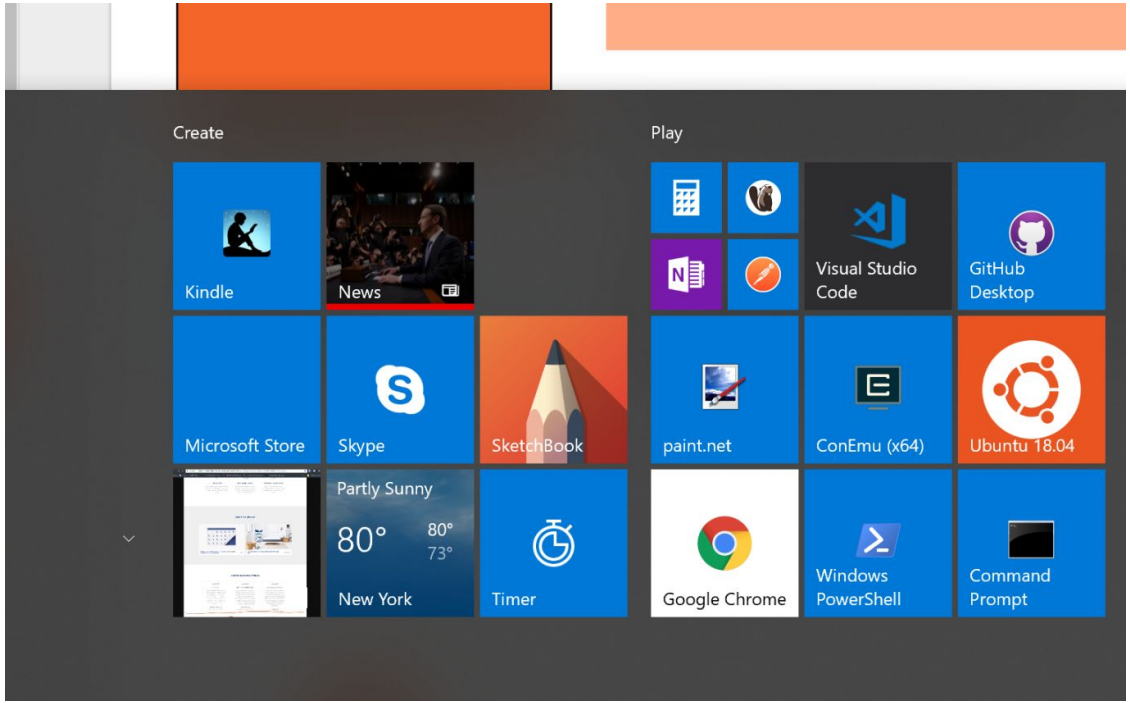


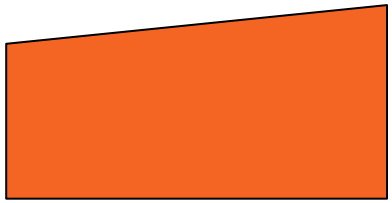
How the Start menu can look

Useful links:

[Fullstack Academy Toolbox]

[Installing Postman]

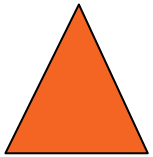




Useful links:

[Fullstack Academy Toolbox]

[PostgreSQL Install FSA]



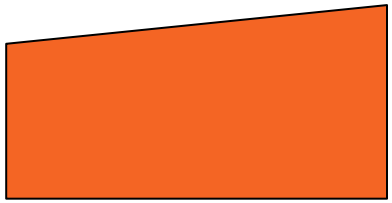
Installing PostgreSQL - part 1

This is a necessary item, though not needed before some time into Junior phase. There are 2 options, both with trade-offs. One is to install PostgreSQL for Win10, but then the PostgreSQL server will start with Win10. It may not be something needed or wanted.

The other is to install PostgreSQL on the WSL side so it is sandboxed. On the flip side, the server needs to be restarted each time a WSL session is started. In some instances, the database needs to be created before the app scripts sync the app DB part to the database.

Be cautious and take it slow, this part is pretty involved and can go sideways easily.

First, open a WSL window and update/upgrade the Ubuntu machine



Installing PostgreSQL - part 2

Follow the directions to install Postgres then create a super-user. The username should be the same one registered for Ubuntu. In doubt, try using **psql.** The error will show which username is needed. The password can be left blank. The last step is to put the trust in the **pg_hba.conf** file. Literally. See the view in the next slide.

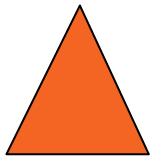
At that stage, check with **psql** that the postgreSQL can be accessed. This is pretty much it, and should allow all the labs, workshops and personal projects to interact with postgreSQL through the scripts in place. Note that it may be necessary sometimes to create the database first then proceed to the npm set-up.

NB: restart postgreSQL as indicated. It is the same command line for each time it is restarted. Or write a quick script to be executed when WSL is loaded.

Useful links:

Fullstack Academy Toolbox

PostgreSQL Install FSA



Installing PostgreSQL - part 3

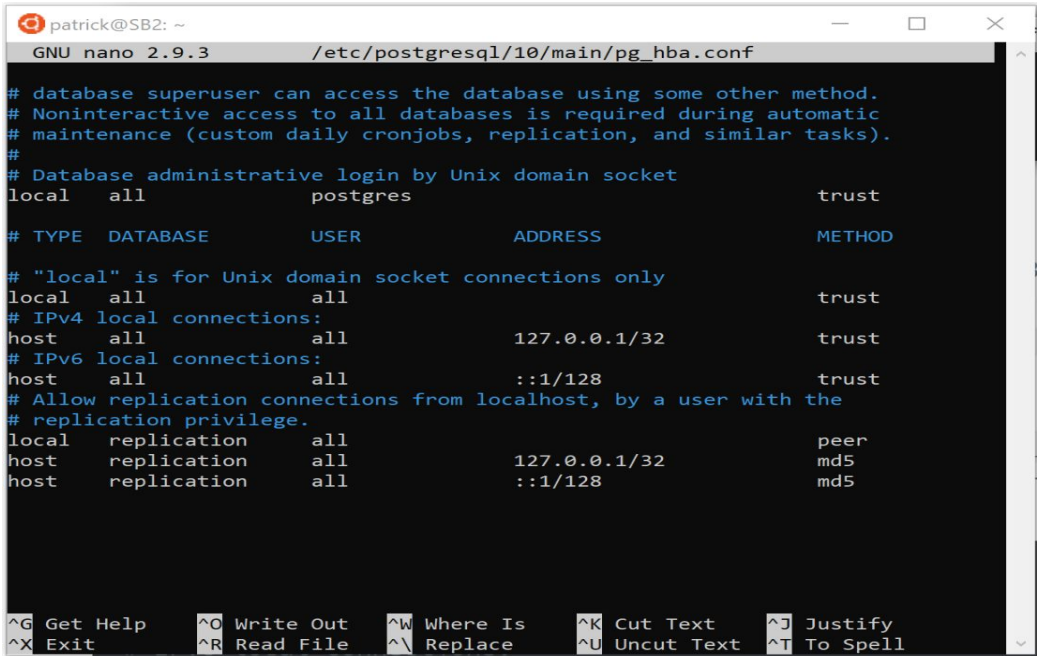
Useful links:

[Fullstack Academy Toolbox]

[PostgreSQL Install FSA]

[SQL Cheat Sheet]

`pg_hba.conf` should look like this:



Starting and stopping PostgreSQL service

The main trade-off from installing PostgreSQL in WSL-Ubuntu and not in Win10 is to restart the PostgreSQL server each time the Ubuntu client is opened. Do it with :

```
user@pc~$ sudo service postgresql start
```

The other part is that the same server needs to be shut off when leaving. The server will continue taxing the memory and the CPU because of the Unix socket it uses. This can cause the machine to heat up. On thin machines like the Surface devices, it can be an issue. Use the following command:

```
user@pc~$ sudo service postgresql stop
```

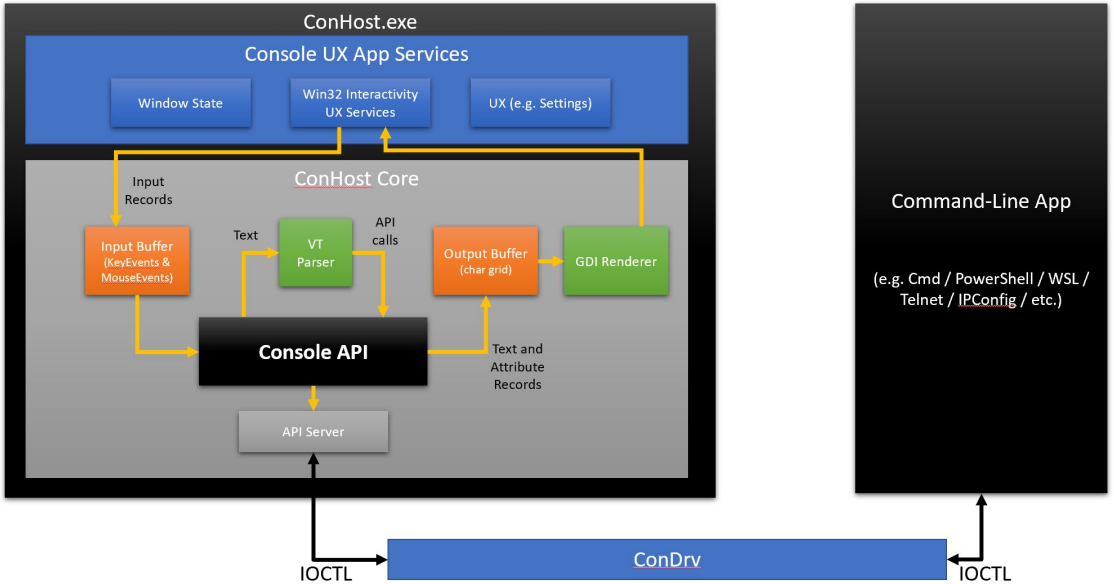
A little more on the Win10 console

Useful links:

[Microsoft Blog for Developers]

[Windows Console App]

This is a clear diagram of how the console operates:



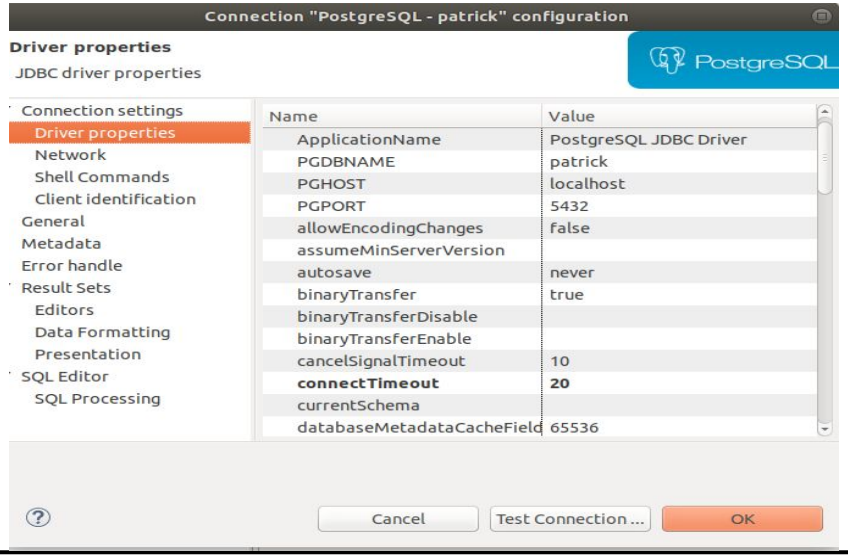
Install & config DBeaver

Useful links:

[DBeaver documentation]

[DBeaver install]

Install DBeaver on the Win10 side, as WSL does not support GUI. Setting up the connection in DBeaver can be a tad tricky. First, the JDBC driver needs to be installed, via the tab Driver.



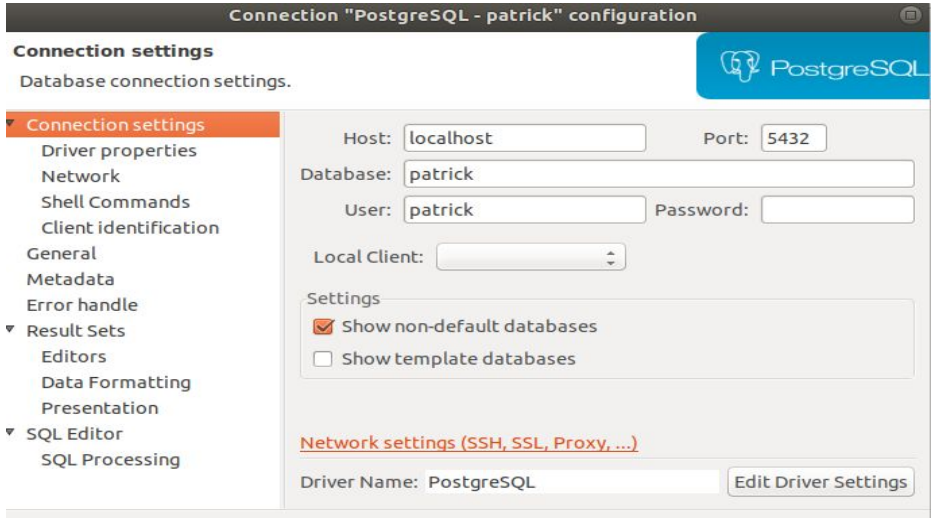
DBeaver : config

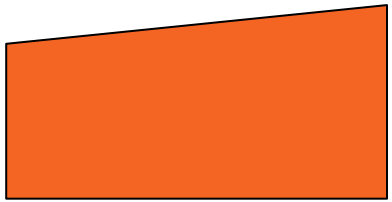
The connection parameters are mainly the user previously created, and the relevant password, if one has been defined. Check in the DB window if the <your_user> database can be accessed, even if it is presently empty.

Useful links:

[DBeaver documentation]

[DBeaver install]

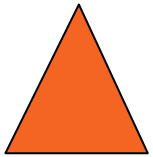


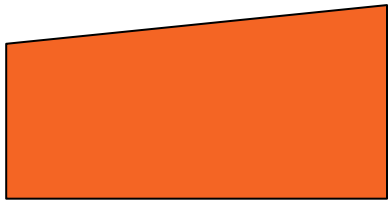


Which browser?

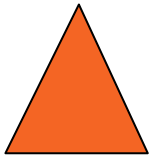
Firefox is a good option, but Chrome or its open source twin Chromium are probably better for development. Just the debugger, if it's the only aspect considered, is so worth the install. On the other hand, Firefox Inspector is quite good too. Your choice, or load them both.

It may be a good idea to add Chrome to the PATH in WSL. This allows chrome.exe to be invoked from the WSL prompt. It can also be opened from Win10, and used the same way as well.





So much to chose from...



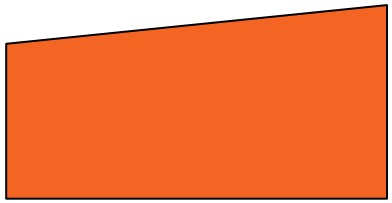
The Ubuntu and Linux community is very much alive and kicking. With a little research and some keen eyes, a lot can be mined and installed to make coding easier. Up to you!

Nevertheless, some advice is needed. Frequently, apps and software are at the experimental stage or very new. Bugs and flaws can be frequent. As it is often the case, use your best judgement and it may be better to err on the side of caution, unless one may feel comfortable with experimenting.

A few notes

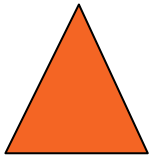
Storage: Ubuntu does not need a huge amount of disk space. First it is a very compact OS, though other Linux distributions are even more lightweight. The trick is that the disk space under Windows management can be accessed from Ubuntu. Just mount and unmount the disk, and get some folders created to use that additional storage space.

Power management: This is a minus for Linux in general, it is not good at managing the power consumption. On a desktop, not such a big deal; on a laptop the battery will be eaten up 30 to 50% faster than with Windows alone. Just be aware of it. This is even more true for Surface devices, which can get rather hot.



A few notes

Thank you to Dakota Blair at Fullstack Academy for his help on the database portion.



All those modifications have been tested on:

Custom-built desktop PC i-5 4690K overclocked, 16GB RAM, 2TB storage (SSD/2HD), GTX 1070 NVidia 8GB,

Surface Pro tablet (2017) i-7, 8GB RAM, 256GB SSD,

Surface Book 2 i-7, 16GB RAM, 1TB SSD, GTX 1060 NVidia 8GB.

All are running Windows 10 - April 2018.