**Airflow on Windows 10 Ubuntu or Native Ubuntu 18.04 Next Steps**

By Joseph DeArce 03/17/2020

Airflow is a top-level Apache project used for orchestrating workflows and creating data pipelines. It’s becoming a popular choice for organizations of all sizes and all industries. Airflow is built upon Python, but contains some libraries that will only work on Linux, so workarounds using virtual machines or Docker are required for a fully functional version on different platforms such as Windows 10

This article is about setting up Airflow on WSL (Windows Subsystem for Linux) or a Native Linux install, for an out of the box for access that uses both Multi-threading and parallel processing and then configuring it for quasi-cluster access. In this article we will use a step by step approach to configure Airflow on Windows Ubuntu 18.04 on WLS with very little overhead and this will also work on native Ubuntu. Just to understand that this is still not for a production system, but can be used for development and testing. This is .true for several reasons, we are running on only one system not a cluster, two we are using the Local executor not the Celery Executor.

I chose to install Ubuntu 18.0.4.3 as my WSL subsystem and had no problems, but the steps listed here will also work on a native Ubuntu installation with one change, which we will point out later on in this article.

**Installing PostgreSQL on Ubuntu WSL**

A very good site for is that of Michael Treat on the GitHub repository on the Windows Sub-system for Linux and on installing PostgreSQL on WSL, these steps are about the same as on a native Linux install with some minor changes.

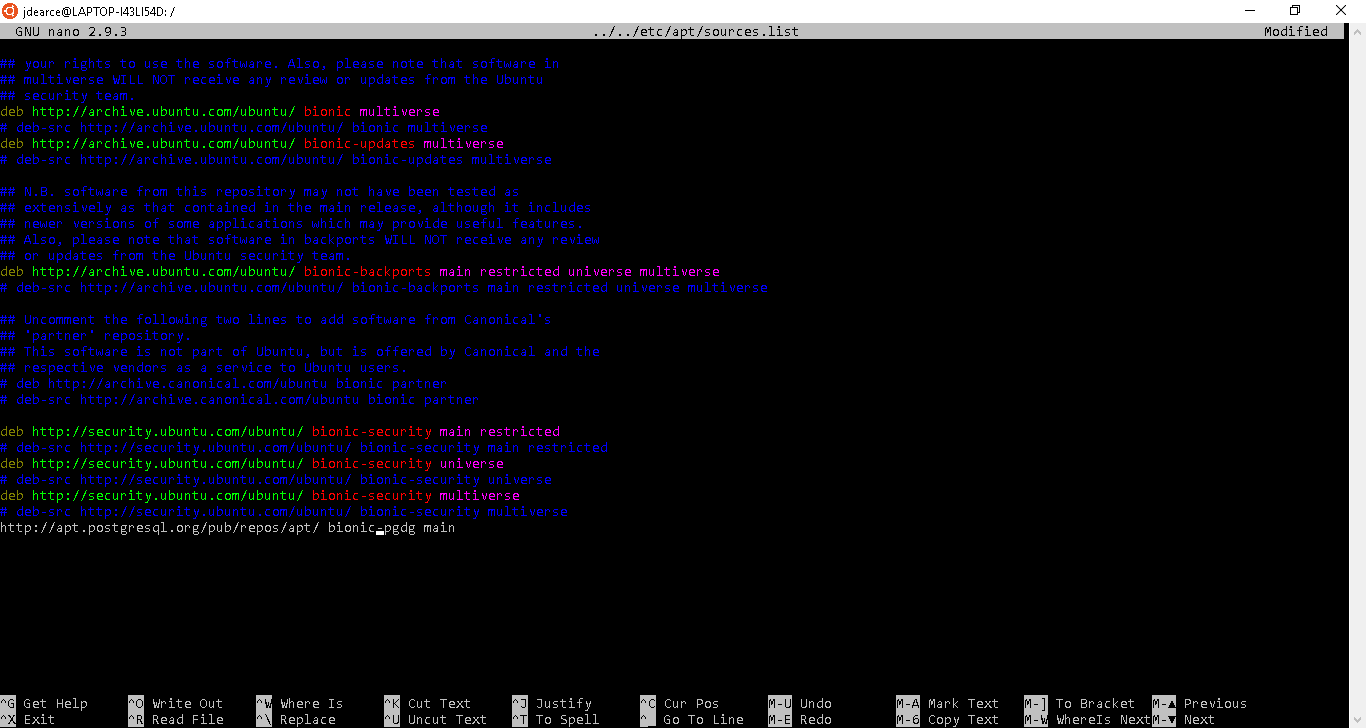
**https://github.com/michaeltreat/Windows-Subsystem-For-Linux-Setup-Guide/blob/master/readmes/installs/PostgreSQL.md**

We will now install PostgreSQL 10 under Windows WSL Ubuntu 18.04. We are installing this through a terminal running under Ubuntu 18.04 as a subsystem to Windows 10. The PostgreSQL for Linux installation docs <https://www.postgresql.org/download/linux/ubuntu/>

**Install**

Open a terminal and then go to the root of the Ubuntu Subsystem by typing cd /. This will place you on the root here you can use any editor Vim, Vi, Emacs, Atom, or nano. Since Nano comes preinstalled I used that.

Type **sudo nano ../../etc/apt/sources.list**. This will open up the sources.list file of all Ubuntu repositories in the Nano editor.



At the bottom of this file, paste in this line

**deb** [**http://apt**](http://apt/)**.postgresql.org/pub/repos/apt/ xenial-pgdg main**

Change the last part of the line above from **xenial**- to whichever version of Ubuntu you are running. For example, bionic- for Ubuntu 18.04.X.

Current

|  |  |  |
| --- | --- | --- |
| Version | Code name | Release |
| Ubuntu 18.04.1 LTS | Bionic Beaver | July 26, 2018 |
| **Ubuntu 18.04 LTS** | **Bionic Beaver** | **April 26, 2018** |
| Ubuntu 16.04.6 LTS | Xenial Xerus | February 28, 2019 |
| Ubuntu 16.04.5 LTS | Xenial Xerus | August 2, 2018 |

**deb** [**http://apt**](http://apt/)**.postgresql.org/pub/repos/apt/ bionic-pgdg main**

When that’s done, press **ctrl + s** to save and then **ctrl + x** to exit the file, or just **ctrl + x** and then press **y** when prompted to save your changes, and enter to finally close the editor.

Next, copy these 2 lines and paste them into your terminal:

**wget –quiet -O –** [**https://www**](https://www/)**.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add –**

**sudo apt-get update**

This will add **postgresql-10** to your repositories so you can install the latest version Also we will update the Ubuntu system to make it current..

After the updates are complete, enter in this line **sudo apt-get install postgresql-10** and press **y** when prompted. If the process aborts automatically, you may have to restart your terminal.

jdearce@LAPTOP-I43LI54D:/$ sudo apt-get install postgresql-10

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages will be upgraded:

postgresql-10

1 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.

Need to get 4946 kB of archives.

After this operation, 8503 kB of additional disk space will be used.

Get:1 [http://apt](http://apt/).postgresql.org/pub/repos/apt bionic-pgdg/main amd64 postgresql-10 amd64 10.11-1.pgdg18.04+1 [4946 kB]

Fetched 4946 kB in 2s (2753 kB/s)

Preconfiguring packages ...

(Reading database ... 36619 files and directories currently installed.)

Preparing to unpack .../postgresql-10\_10.11-1.pgdg18.04+1\_amd64.deb ...

invoke-rc.d: could not determine current runlevel

\* Stopping PostgreSQL 10 database server [ OK ]

Unpacking postgresql-10 (10.11-1.pgdg18.04+1) over (10.10-0ubuntu0.18.04.1) ...

Setting up postgresql-10 (10.11-1.pgdg18.04+1) ...

invoke-rc.d: could not determine current runlevel

Processing triggers for postgresql-common (190ubuntu0.1) ...

Building PostgreSQL dictionaries from installed myspell/hunspell packages...

Removing obsolete dictionary files:

jdearce@LAPTOP-I43LI54D:/$

**Postgres User Setup**

postgresql-10 runs under the user postgres. We need to give this user a password so that postgres can allow this user to login to the database.

To set the password for postgres, type **sudo passwd postgres**. Dj931#7&

You will get a prompt to enter in your password. It will not show you the password you are typing, but it is still registering your keystrokes. Close and reopen the terminal to finalize the changes.

**Using psql**

After your first install, and each time you restart your machine you will have to also restart the postgres service, or else you will get a Is the server running? Error.

To start the service, type **sudo service postgresql start**.

To stop the service, type **sudo service postgresql stop**.

To connect to postgres, type **sudo -u postgres psql**.

**postgres 305 1 2 07:13 ? 00:00:00 /usr/lib/postgresql/10/bin/postgres -D /var/lib/postgresql/10/main -c co**

**postgres 307 305 0 07:13 ? 00:00:00 postgres: 10/main: checkpointer process**

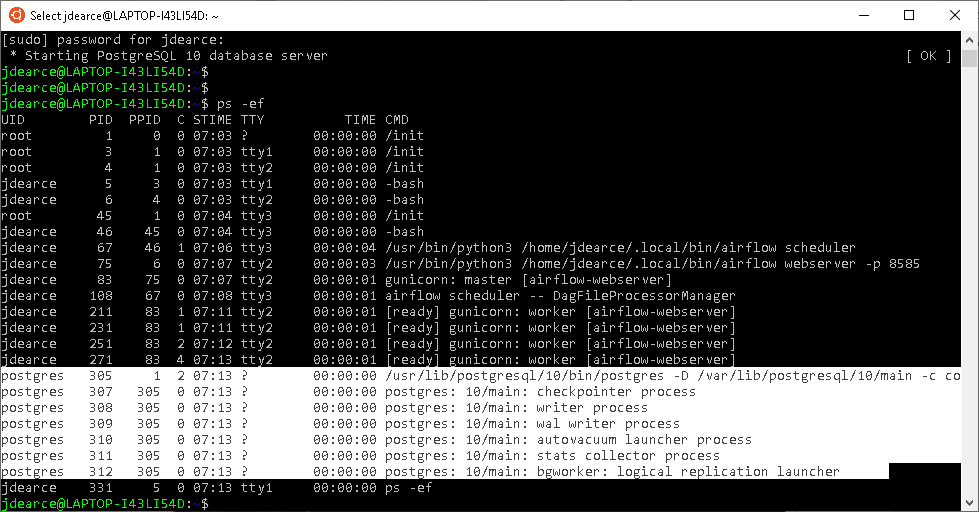
**postgres 308 305 0 07:13 ? 00:00:00 postgres: 10/main: writer process**

**postgres 309 305 0 07:13 ? 00:00:00 postgres: 10/main: wal writer process**

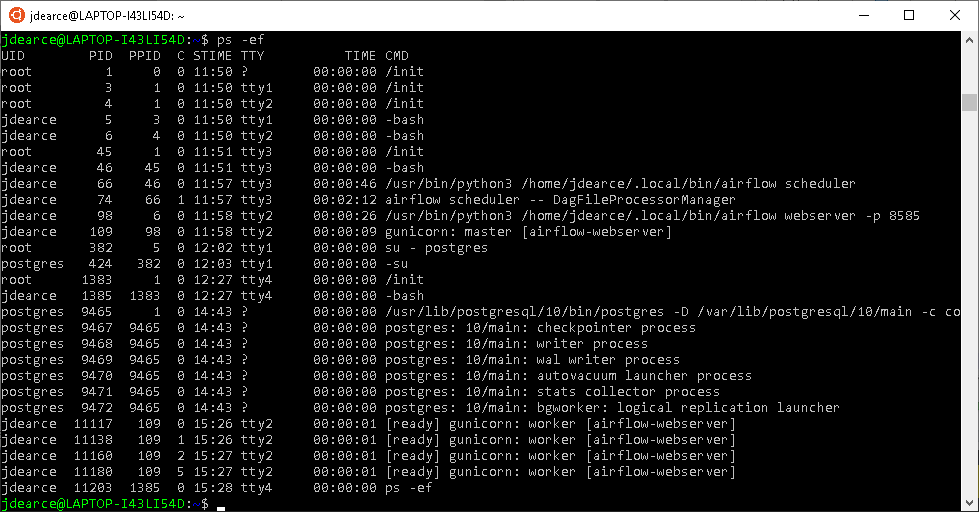
**postgres 310 305 0 07:13 ? 00:00:00 postgres: 10/main: autovacuum launcher process**

**postgres 311 305 0 07:13 ? 00:00:00 postgres: 10/main: stats collector process**

**postgres 312 305 0 07:13 ? 00:00:00 postgres: 10/main: bgworker: logical replication launcher**



You should get a prompt asking for your password. If this doesn’t work, then you can try the second option listed below.



Switch users to postgres by typing **su – postgres**.

**Type psql.**

When this is successful you will see the command line change to look like this postgres=#

**jdearce@LAPTOP-I43LI54D:~$ su – postgres**

**Password:**

**postgres@LAPTOP-I43LI54D:~$**

**postgres@LAPTOP-I43LI54D:~$**

**postgres@LAPTOP-I43LI54D:~$ psql**

**psql (10.10 (Ubuntu 10.10-0ubuntu0.18.04.1), server 10.11 (Ubuntu 10.11-1.pgdg18.04+1))**

**Type “help” for help.**

**Postgres=#**

**postgres=# help**

You are using psql, the command-line interface to PostgreSQL.

**.**

**postgres@LAPTOP-I43LI54D:~$**

**postgres@LAPTOP-I43LI54D:~$ psql**

**psql (10.10 (Ubuntu 10.10-0ubuntu0.18.04.1), server 10.11 (Ubuntu 10.11-1.pgdg18.04+1))**

**Type “help” for help.**

**Postgres=# help**

**You are using psql, the command-line interface to PostgreSQL.**

**Type: \copyright for distribution terms**

**\h for help with SQL commands**

**\? For help with psql commands**

**\g or terminate with semicolon to execute query**

**\q to quit**

**postgres=# select version();**

**postgres=# CREATE DATABASE AIRFLOWDB;**

**WARNING: could not flush dirty data: Function not implemented**

**CREATE DATABASE**

**postgres=# \l**

**List of databases**

**Name | Owner | Encoding | Collate | Ctype | Access privileges**

**-----------+----------+----------+---------+---------+-----------------------**

**airflowdb | postgres | UTF8 | C.UTF-8 | C.UTF-8 |**

**postgres | postgres | UTF8 | C.UTF-8 | C.UTF-8 |**

**template0 | postgres | UTF8 | C.UTF-8 | C.UTF-8 | =c/postgres +**

**| | | | | postgres=CTc/postgres**

**template1 | postgres | UTF8 | C.UTF-8 | C.UTF-8 | =c/postgres +**

**| | | | | postgres=CTc/postgres**

**(4 rows)**

To install PostgreSQL on Airflow using Ubuntu 18.04 use the following commands:

**sudo apt-get install postgresql postgresql-contrib**

Database driver for PostgreSQL is Psycopg2 and we will need to install it under WSL I had no problems but under Ubuntu 18.04 this did not work.

**pip install psycopg2**

jdearce@LAPTOP-I43LI54D:~$ pip3 install psycopg2

Collecting psycopg2

Downloading https://files.pythonhosted.org/packages/84/d7/6a93c99b5ba4d4d22daa3928b983cec66df4536ca50b22ce5dcac65e4e71/psycopg2-2.8.4.tar.gz (377kB)

100% |████████████████████████████████| 378kB 1.8MB/s

Complete output from command python setup.py egg\_info:

running egg\_info

creating pip-egg-info/psycopg2.egg-info

writing pip-egg-info/psycopg2.egg-info/PKG-INFO

writing dependency\_links to pip-egg-info/psycopg2.egg-info/dependency\_links.txt

writing top-level names to pip-egg-info/psycopg2.egg-info/top\_level.txt

writing manifest file 'pip-egg-info/psycopg2.egg-info/SOURCES.txt'

Database driver for PostgreSQL is Psycopg2, under Ubuntu 18.04 I was not able to install it using the regular package see below.

jdearce@datatree17:~$ pip3 install psycopg2

Collecting psycopg2

Using cached https://files.pythonhosted.org/packages/84/d7/6a93c99b5ba4d4d22daa3928b983cec66df4536ca50b22ce5dcac65e4e71/psycopg2-2.8.4.tar.gz

Complete output from command python setup.py egg\_info:

running egg\_info

creating pip-egg-info/psycopg2.egg-info

writing pip-egg-info/psycopg2.egg-info/PKG-INFO

writing dependency\_links to pip-egg-info/psycopg2.egg-info/dependency\_links.txt

writing top-level names to pip-egg-info/psycopg2.egg-info/top\_level.txt

writing manifest file 'pip-egg-info/psycopg2.egg-info/SOURCES.txt'

Error: b'You need to install postgresql-server-dev-X.Y for building a server-side extension or libpq-dev for building a client-side application.\n'

----------------------------------------

Command "python setup.py egg\_info" failed with error code 1 in /tmp/pip-install-ygmcrs0i/psycopg2/

You are using pip version 18.1, however version 20.0.2 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.

jdearce@datatree17:~$ pip3 install --upgrade pip3

Collecting pip3

Could not find a version that satisfies the requirement pip3 (from versions: )

No matching distribution found for pip3

You are using pip version 18.1, however version 20.0.2 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.

jdearce@datatree17:~$ pip3 install --upgrade pip

Collecting pip

Downloading https://files.pythonhosted.org/packages/54/0c/d01aa759fdc501a58f431eb594a17495f15b88da142ce14b5845662c13f3/pip-20.0.2-py2.py3-none-any.whl (1.4MB)

100% |████████████████████████████████| 1.4MB 3.1MB/s

Installing collected packages: pip

Found existing installation: pip 18.1

Uninstalling pip-18.1:

Successfully uninstalled pip-18.1

Successfully installed pip-20.0.2

jdearce@datatree17:~$ pip3 install psycopg2

Collecting psycopg2

Using cached psycopg2-2.8.4.tar.gz (377 kB)

ERROR: Command errored out with exit status 1:

command: /home/jdearce/anaconda3/bin/python -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'"'/tmp/pip-install-272ycoxd/psycopg2/setup.py'"'"'; \_\_file\_\_='"'"'/tmp/pip-install-272ycoxd/psycopg2/setup.py'"'"';f=getattr(tokenize, '"'"'open'"'"', open)(\_\_file\_\_);code=f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, \_\_file\_\_, '"'"'exec'"'"'))' egg\_info --egg-base /tmp/pip-install-272ycoxd/psycopg2/pip-egg-info

cwd: /tmp/pip-install-272ycoxd/psycopg2/

Complete output (7 lines):

running egg\_info

creating /tmp/pip-install-272ycoxd/psycopg2/pip-egg-info/psycopg2.egg-info

writing /tmp/pip-install-272ycoxd/psycopg2/pip-egg-info/psycopg2.egg-info/PKG-INFO

writing dependency\_links to /tmp/pip-install-272ycoxd/psycopg2/pip-egg-info/psycopg2.egg-info/dependency\_links.txt

writing top-level names to /tmp/pip-install-272ycoxd/psycopg2/pip-egg-info/psycopg2.egg-info/top\_level.txt

writing manifest file '/tmp/pip-install-272ycoxd/psycopg2/pip-egg-info/psycopg2.egg-info/SOURCES.txt'

Error: b'You need to install postgresql-server-dev-X.Y for building a server-side extension or libpq-dev for building a client-side application.\n'

----------------------------------------

**ERROR:** Command errored out with exit status 1: python setup.py egg\_info Check the logs for full command output.

jdearce@datatree17:~$

Under Ubuntu 18.04 I had to use psycopg2-binary and this installed correctly.

jdearce@datatree17:~$ pip3 install psycopg2-binary

Collecting psycopg2-binary

Downloading psycopg2\_binary-2.8.4-cp37-cp37m-manylinux1\_x86\_64.whl (2.9 MB)

|████████████████████████████████| 2.9 MB 372 kB/s

Installing collected packages: psycopg2-binary

Successfully installed psycopg2-binary-2.8.4

jdearce@datatree17:~$

Then to create a user for airflow:

**jdearce@LAPTOP-I43LI54D:~$ sudo -u postgres createuser --interactive**

**[sudo] password for jdearce:**

**Enter name of role to add: airflow**

**Shall the new role be a superuser? (y/n) n**

**Shall the new role be allowed to create databases? (y/n) n**

**Shall the new role be allowed to create more new roles? (y/n) n**

**jdearce@LAPTOP-I43LI54D:~$**

Then set the user's password and create the database:

**sudo -u postgres psql**

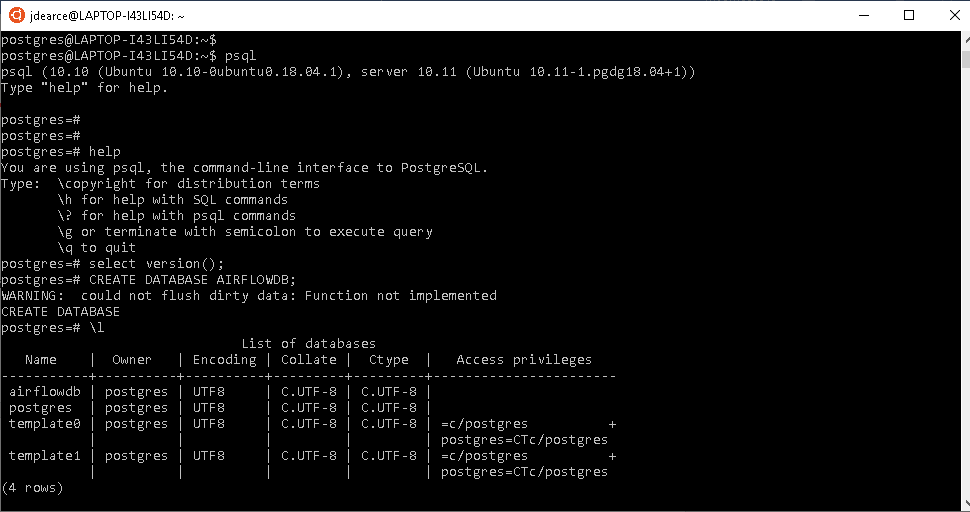
psql (10.10)

Type "help" for help.

postgres=# **ALTER USER airflow WITH PASSWORD 'airflow\_password';**

**ALTER ROLE**

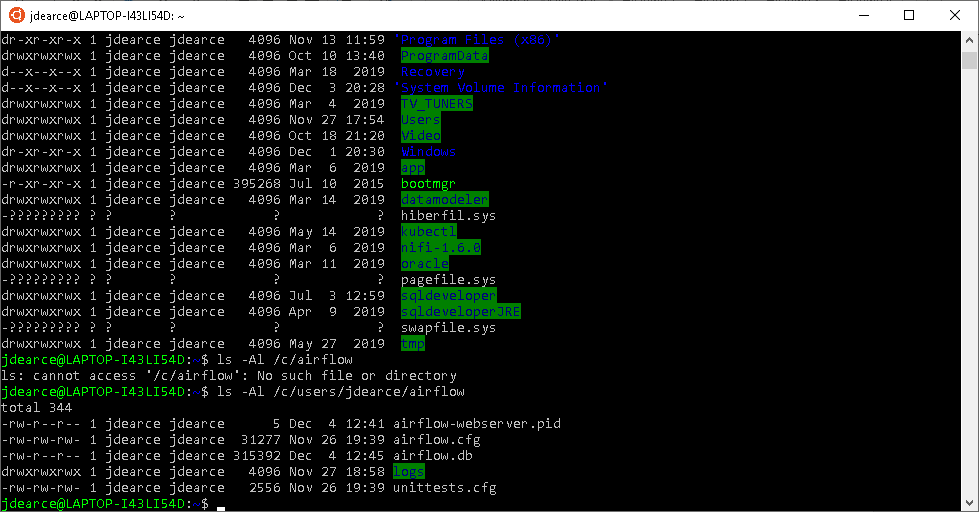
ALTER USER airflow WITH PASSWORD ‘Juq1Y8#dH!8’;

**..**

# Check for success

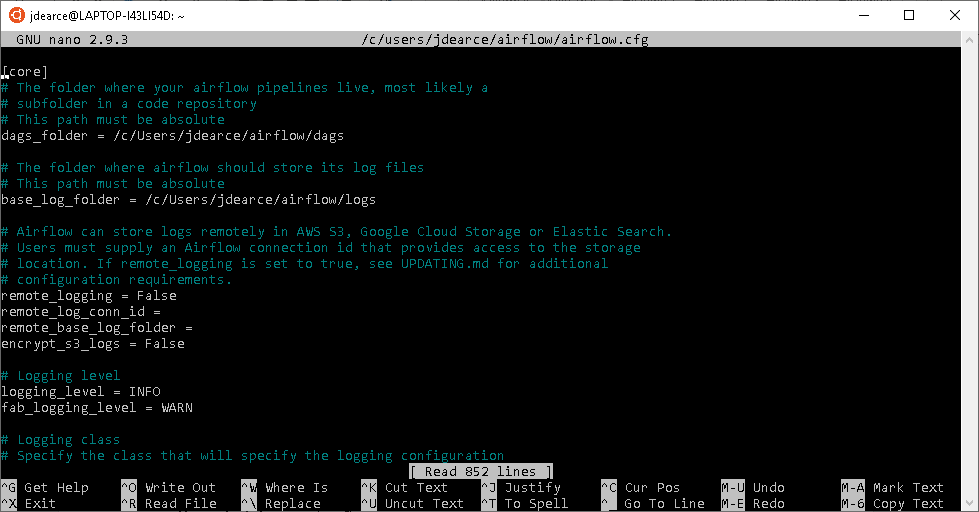
Modify the configuration in **AIRFLOW\_HOME/airflow.cfg**

To access the Airflow directory from Ubuntu home type **ls -Al /c/users/jdearce/airflow** and you will see the screen below.



We will edit the **airflow.cfg** file by typing the command **nano /c/users/jdearce/airflow/airflow.cfg**

You are now in the main Airflow configuration file for the Airflow system.



To the **airflow.cfg** file we will make two changes, see below.

# Change the executor to Local Executor

**executor = LocalExecutor**

**# Change the meta db configuration**

# Note: the postgres username and password do not matter for now, since the database server and clients are still on the same host

**# Lets create a metadata database for airflow**

CREATE DATABASE database\_name;

\l

CREATE DATABASE AIRFLOW;

\l

**#Check for success**

Modify the configuration in **AIRFLOW\_HOME/airflow.cfg**

We will edit the **airflow.cfg** file by typing in the command **nano /c/users/jdearce/airflow/airflow.cfg**

**# Change the executor to Local Executor**

**executor = LocalExecutor**

**# Change the meta db configuration**

**# Note: the postgres username and password do not matter for now, since the database server and clients are still on the same host**

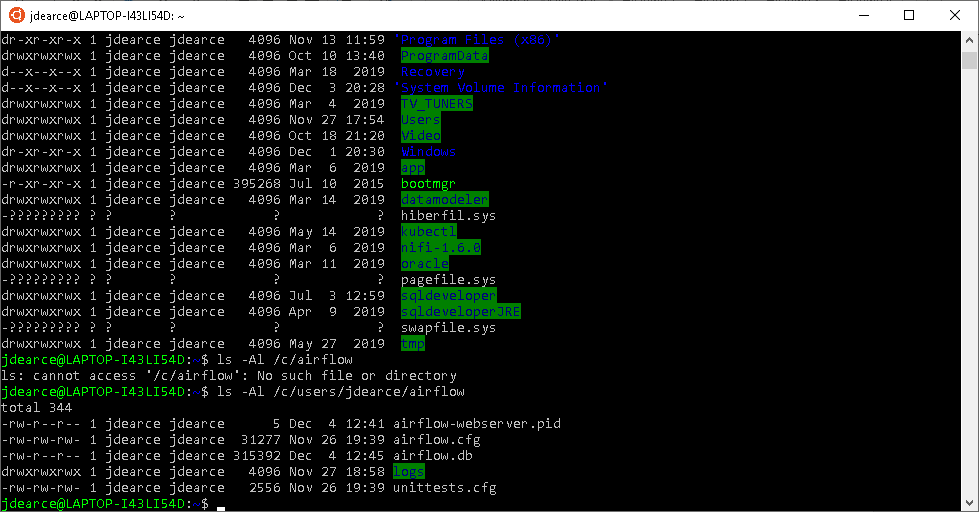
sql\_alchemy\_conn = **postgresql+psycopg2://your\_postgres\_user\_name:your\_postgres\_password@host\_name/database\_name**

My version of the connect string

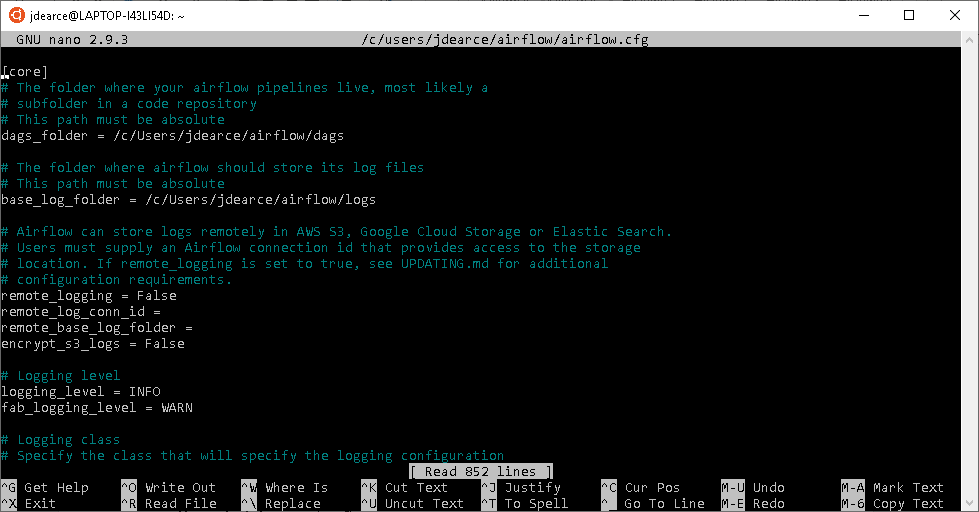
sql\_alchemy\_conn = **postgresql+psycopg2://postgres:**Fhx#29gB1!7**@localhost:5432/airflowdb**

Our first change is to change the executer option this located in the **airflow.cfg** file. We will change executer setting from **SequentialExecutor** to **LocalExecutor**.

To access the Airflow directory from Ubuntu home type **ls -Al /c/users/jdearce/airflow** and you will see the screen below.



We will edit the **airflow.cfg** file which is located in the Airflow home directory, by typing the command **nano /c/users/jdearce/airflow/airflow.cfg** at the terminal prompt. This is the main configuration file for the Apache Airflow system,



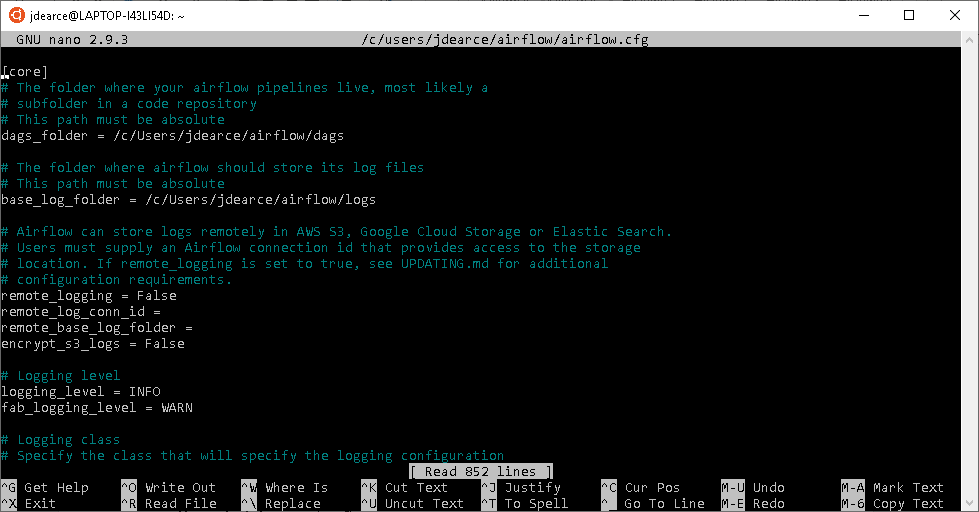
The file in the nano editor holds all the configuration settings for Airflow. We will now change line 58 which holds the executer settings from **SequentialExecutor** to **LocalExecuto**r and the **sql\_alchemy\_conn** value.

**# The executor class that airflow should use. Choices include**

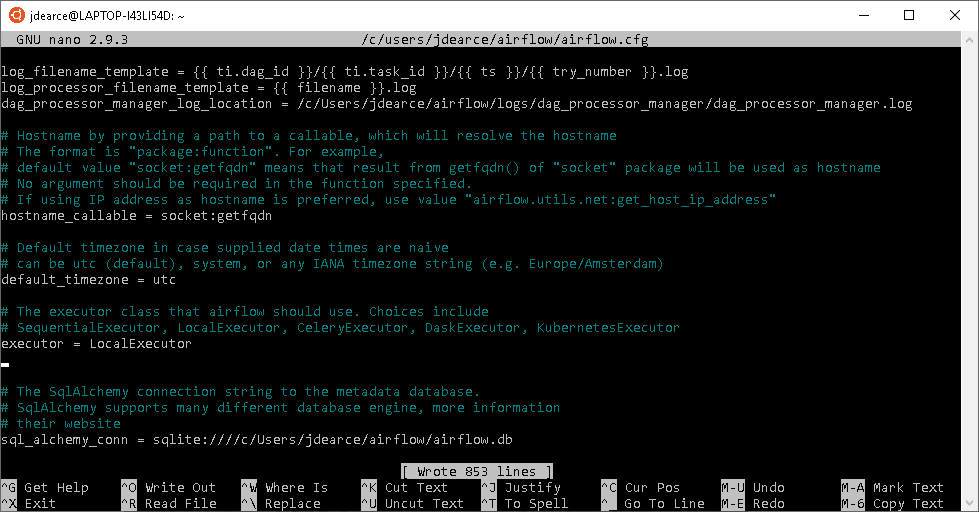
**# SequentialExecutor, LocalExecutor, CeleryExecutor, DaskExecutor, KubernetesExecutor**

**executor = SequentialExecutor**

The You are now in the main Airflow configuration file for the Airflow system.

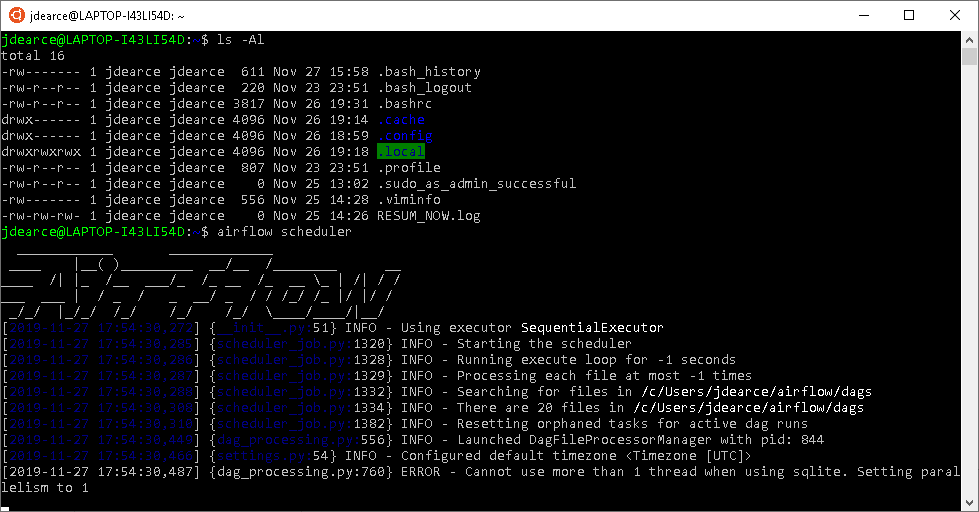


Here we are looking **nano /c/users/jdearce/airflow/airflow.cfg**

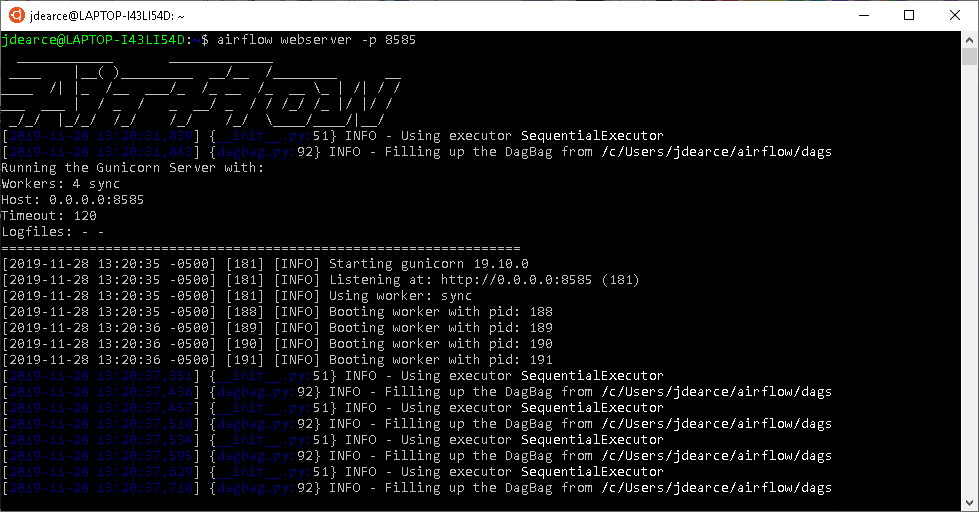


Now let’s recreate the metadata airflow database and restart airflow to test your configuration changes and the DAGs

1. airflow initdb
2. airflow scheduler command must be run from an administrator terminal window or you will get rights issues.



1. airflow webserver -p 8585



**Tips**

Since typing out **sudo service postgres start** and **sudo -u postgres psql** all the time can be odious, I would recommend you set up a couple aliases for this.

Open a terminal and type **cd ~**, then type **sudo nano .profile**. This will open your .profile which controls what your terminal does and looks like.

Add these two lines next to any other aliases that you have:

**alias pgstart='sudo service postgresql start'**

**alias pgstop='sudo service postgresql stop'**

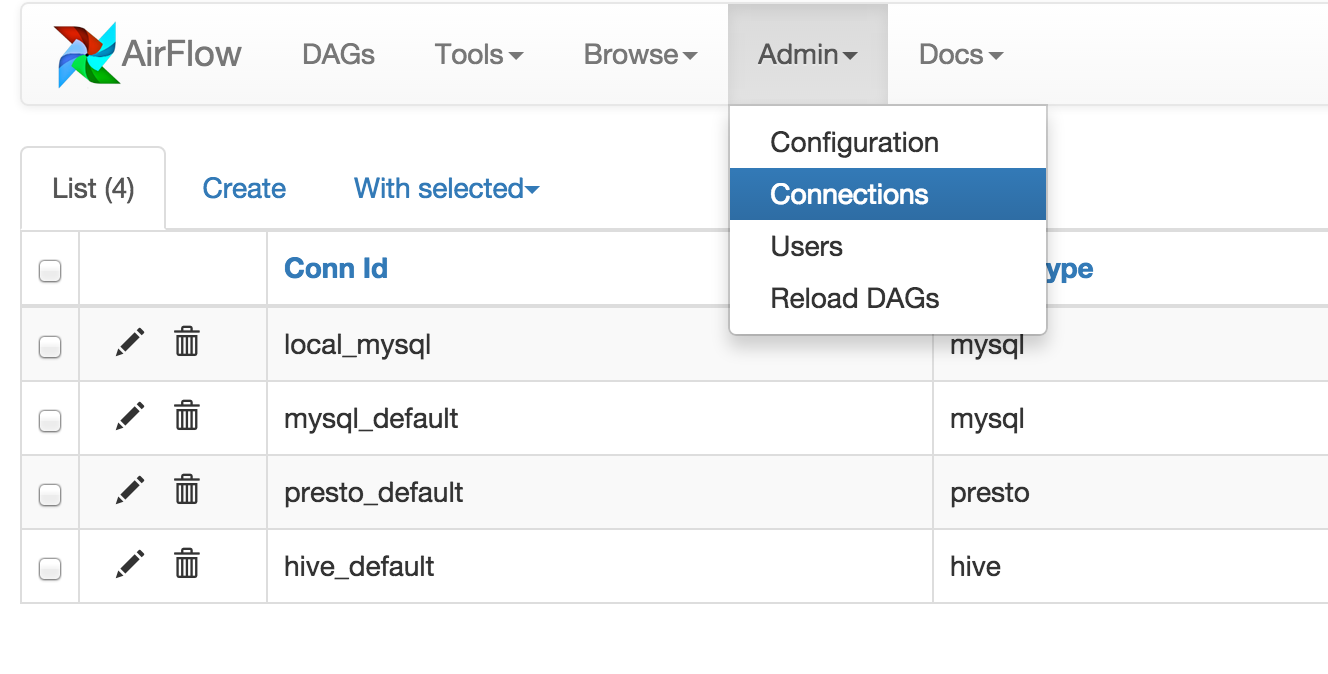
**alias ppg='sudo -u postgres psql'**

This will allow you to type **pgstart** to start running the **psql** service, and **ppg** to quickly log into the psql prompt. This is an example of a Quality of Life enhancement, something that makes your life easier and faster as a developer.

You can change pgstart and ppg to whatever you want, but just be careful you don't overwrite something that postgres might need.

**Connections**[¶](https://airflow.readthedocs.io/en/1.9.0/configuration.html#_blank)

Airflow needs to know how to connect to your environment. Information such as hostname, port, login and passwords to other systems and services is handled in the Admin->Connection section of the UI. The pipeline code you will author will reference the ‘**conn\_id’** of the Connection objects.



**Conclusion**

This intermediate configuration is Airflow on quasi-cluster mode using both the LocalExecutor and the PostgreSQL database. What this has done is create a version of Airflow on a single node that uses both multitasking and Parallel processing.

In the next installment I will cover a more advance configuration which will use encryption and different executers for specific environments.