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

By Joseph DeArce 04/29/2020

The Windows Subsystem for Linux (WSL) is a new part of the Windows OS and has been around for several years now. The steps below come from a Microsoft document from the site listed here ‘**https://docs.microsoft.com/en-us/windows/wsl/install-win10**’ to install WSL. Below that is the section on installation and configuration of Airflow on Ubuntu (WSL)

## Install the Windows Subsystem for Linux

Before installing any Linux distros for WSL, you must ensure that the "Windows Subsystem for Linux" optional feature is enabled:

1. Open PowerShell as Administrator and run:

PowerShell Copy

Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux

1. Restart your computer when prompted.

## Install your Linux Distribution of Choice

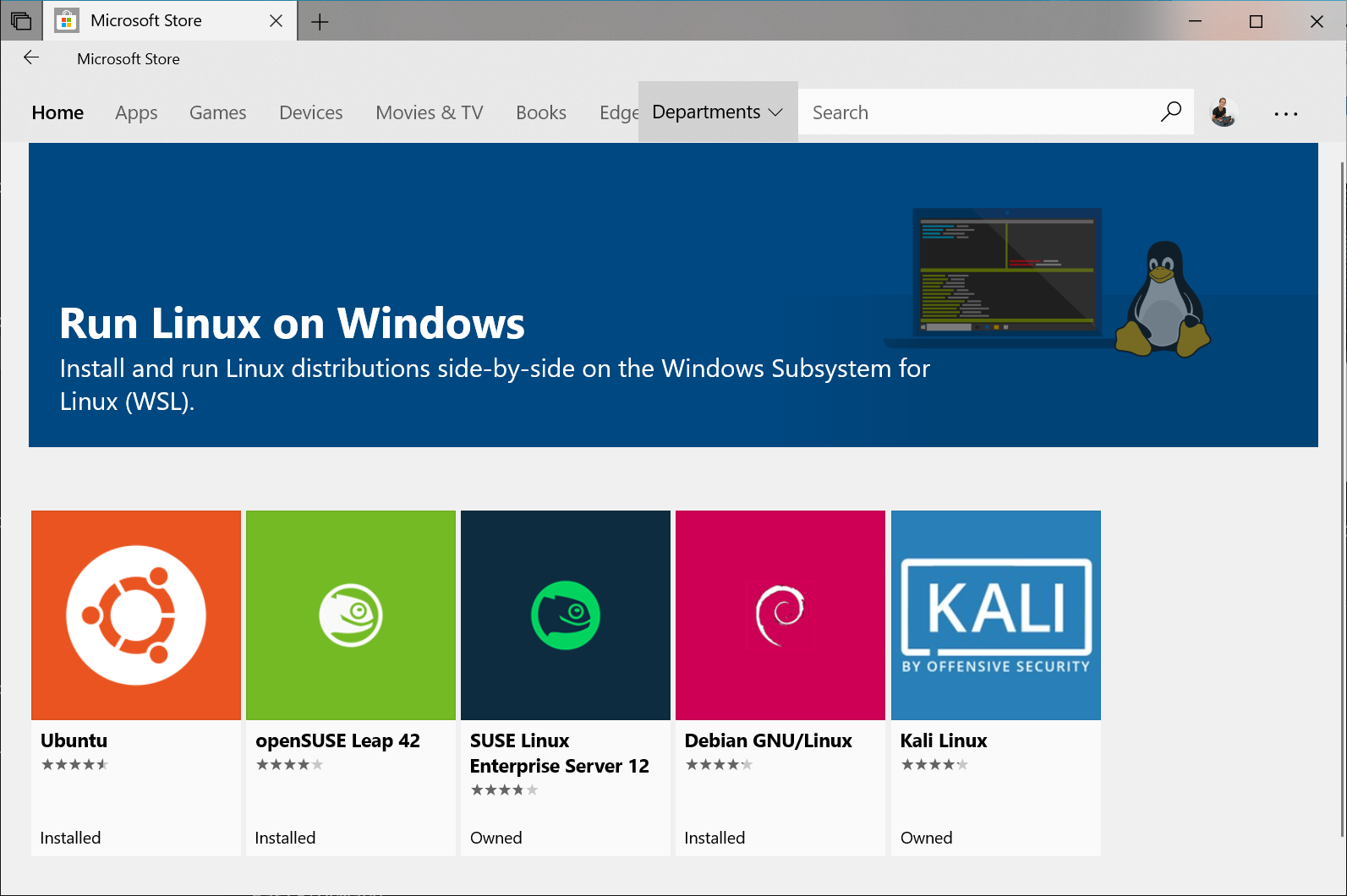
To download and install your preferred distro(s), you have three choices:

* Download and install from the Microsoft Store (see below)
* Download and install from the Command-Line/Script ([read the manual installation instructions](https://docs.microsoft.com/en-us/windows/wsl/install-manual))
* Download and manually unpack and install (for Windows Server - [instructions here](https://docs.microsoft.com/en-us/windows/wsl/install-on-server))

### Windows 10 Fall Creators Update and later: Install from the Microsoft Store

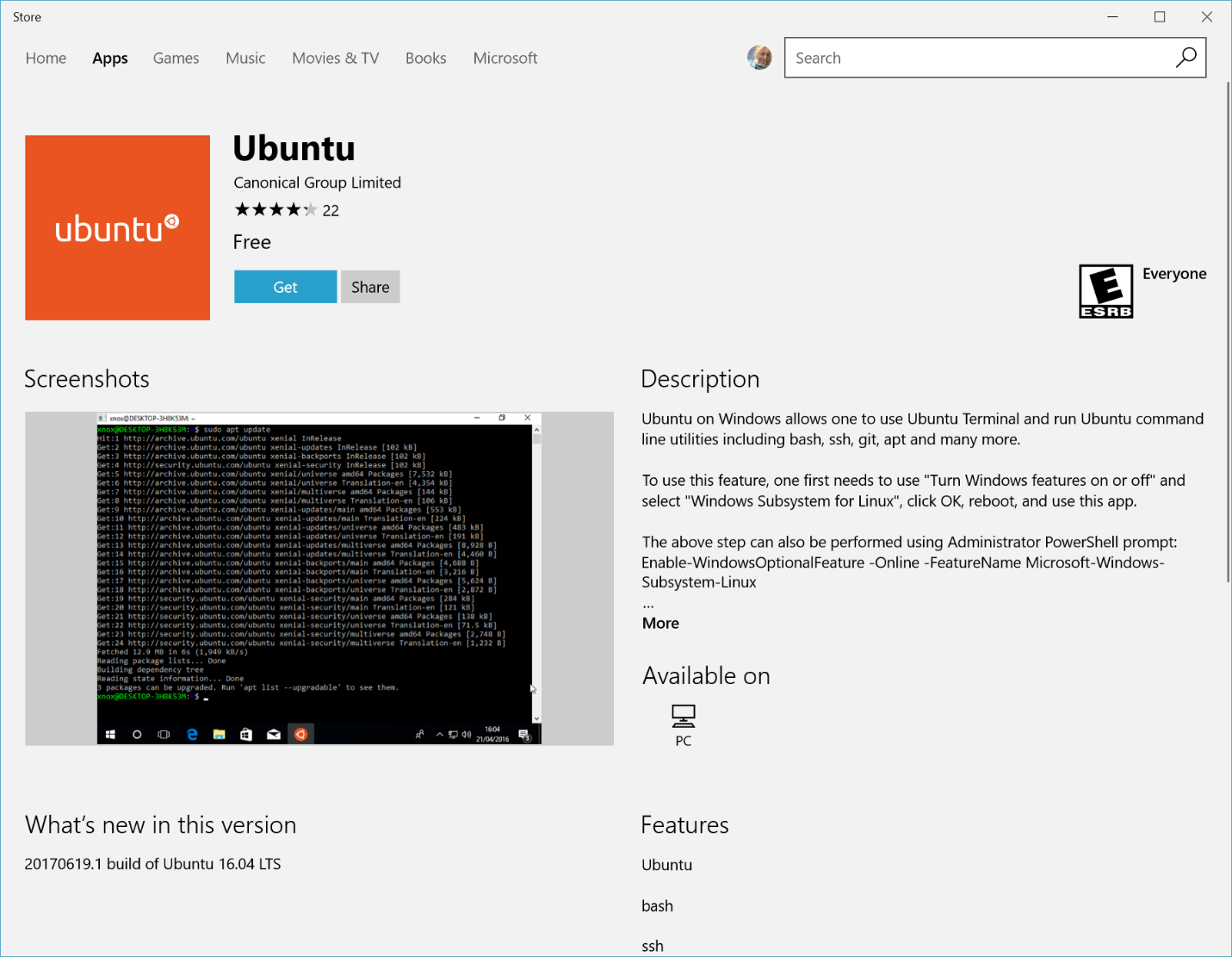
This section is for Windows build 16215 or later. Follow these steps to [check your build](https://docs.microsoft.com/en-us/windows/wsl/troubleshooting#check-your-build-number).

1. Open the Microsoft Store and choose your favorite Linux distribution.



The following links will open the Microsoft store page for each distribution:

* + [Ubuntu 16.04 LTS](https://www.microsoft.com/store/apps/9pjn388hp8c9)
  + [Ubuntu 18.04 LTS](https://www.microsoft.com/store/apps/9N9TNGVNDL3Q)
  + [OpenSUSE Leap 15](https://www.microsoft.com/store/apps/9n1tb6fpvj8c)
  + [OpenSUSE Leap 42](https://www.microsoft.com/store/apps/9njvjts82tjx)
  + [SUSE Linux Enterprise Server 12](https://www.microsoft.com/store/apps/9p32mwbh6cns)
  + [SUSE Linux Enterprise Server 15](https://www.microsoft.com/store/apps/9pmw35d7fnlx)
  + [Kali Linux](https://www.microsoft.com/store/apps/9PKR34TNCV07)
  + [Debian GNU/Linux](https://www.microsoft.com/store/apps/9MSVKQC78PK6)
  + [Fedora Remix for WSL](https://www.microsoft.com/store/apps/9n6gdm4k2hnc)
  + [Pengwin](https://www.microsoft.com/store/apps/9NV1GV1PXZ6P)
  + [Pengwin Enterprise](https://www.microsoft.com/store/apps/9N8LP0X93VCP)
  + [Alpine WSL](https://www.microsoft.com/store/apps/9p804crf0395)
* From the distro's page, select "Get"



## Complete initialization of your distro

Once your distro has been downloaded and installed, you'll need to complete initialization of the new distro:

## Launch a distro

To complete the initialization of your newly installed distro, launch a new instance. You can do this by clicking the "**launch**" button in the Microsoft Store app, or launching the distro from the **Start menu**:

Tip: You might want to pin your most frequently used distros to your Start menu, and/or to your taskbar!



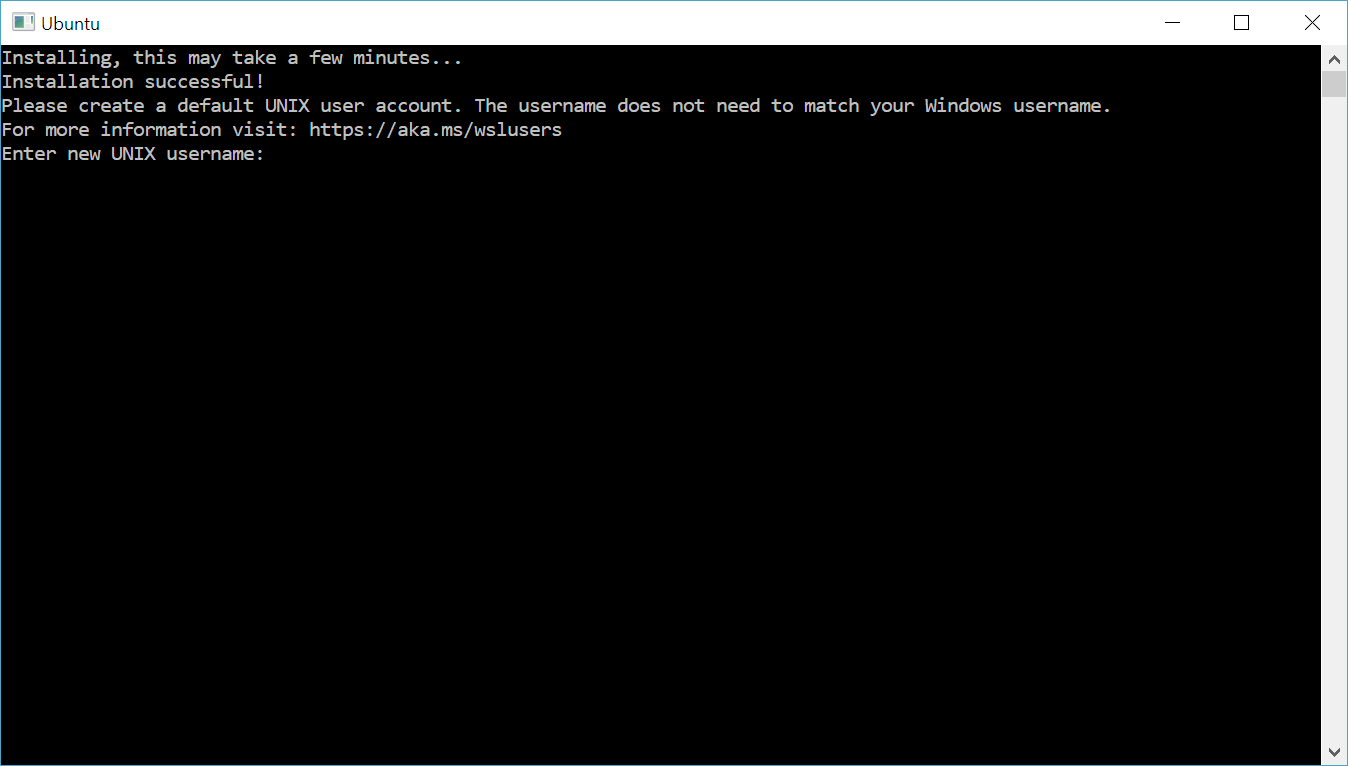
On Windows Server, you can launch your distro's launcher executable <distro>.exe from the distro installation folder.

The first time a newly installed distro runs, a Console window will open, and you'll be asked to wait for a minute or two for the installation to complete.

During this final stage of installation, the distro's files are de-compressed and stored on your PC, ready for use. This may take around a minute or more depending on the performance of your PC's storage devices. This initial installation phase is only required when a distro is clean-installed - all future launches should take less than a second.

## Setting up a new Linux user account

Once installation is complete, you will be prompted to create a new user account (and its password).



This user account is for the normal non-admin user that you'll be logged-in as by default when launching a distro.

You can choose any username and password you wish - they have no bearing on your Windows username.

When you open a new distro instance, you won't be prompted for your password, but **if you elevate a process using sudo, you will need to enter your password**, so make sure you choose a password you can easily remember! See the [User Support](https://docs.microsoft.com/en-us/windows/wsl/user-support) page for more info.

## Update & upgrade your distro's packages

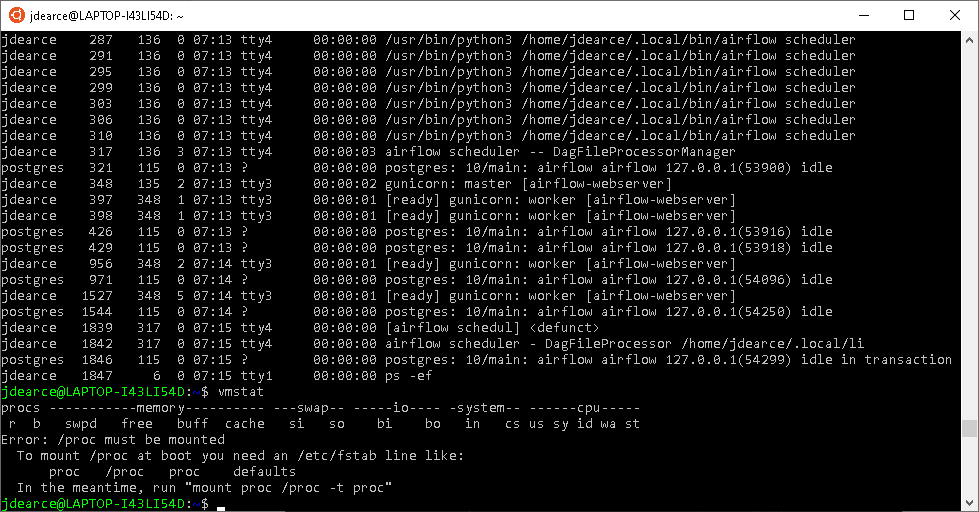
Most distros ship with an empty/minimal package catalog. We strongly recommend regularly updating your package catalog, and upgrading your installed packages using your distro's preferred package manager. On Debian/Ubuntu, you use apt:

bashCopy

sudo apt update && sudo apt upgrade

Windows does not automatically update or upgrade your Linux distro(s): This is a task that the Linux users prefer to control themselves.

You're done! Enjoy using your new Linux distro on WSL! To learn more about WSL, review the other [WSL docs](https://aka.ms/wsldocs), or the [WSL learning resources page](https://aka.ms/learnwsl).



## Troubleshooting

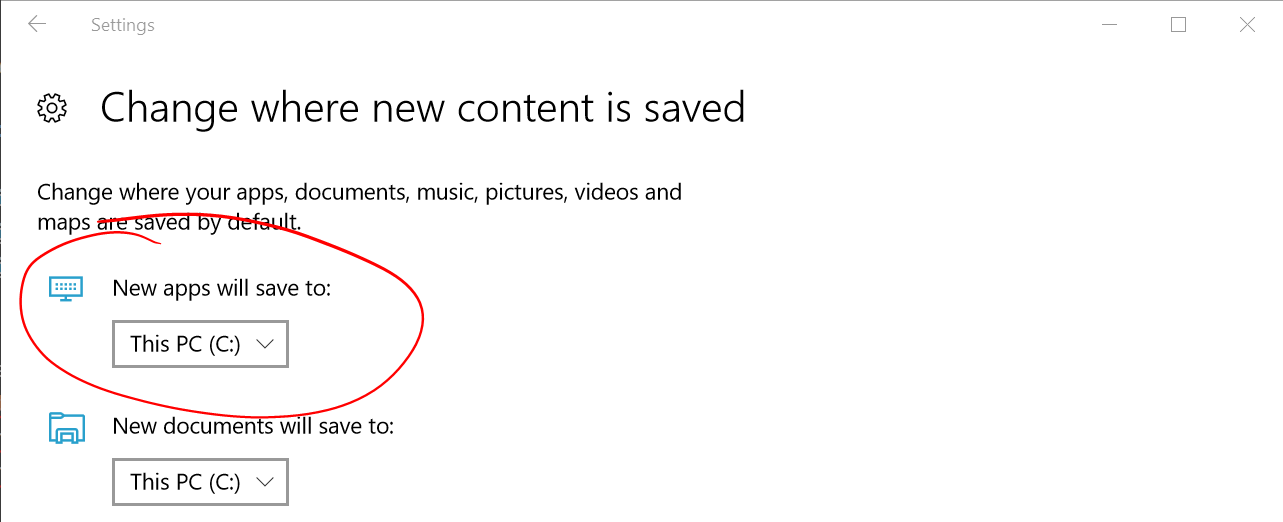
Below are related errors and suggested fixes. Refer to the [WSL troubleshooting page](https://docs.microsoft.com/en-us/windows/wsl/troubleshooting) for other common errors and their solutions.

**Installation failed with error 0x8007007e** This error occurs when your system doesn't support Linux from the store. Make sure that:

* You're running Windows build 16215 or later. [Check your build](https://docs.microsoft.com/en-us/windows/wsl/troubleshooting#check-your-build-number).
* An other way to check the build number is Winver, see below.
* The Windows Subsystem for Linux optional component is enabled and the computer has restarted. [Make sure WSL is enabled](https://docs.microsoft.com/en-us/windows/wsl/troubleshooting#confirm-wsl-is-enabled).



Below are related errors and suggested fixes. Refer to the [WSL troubleshooting page](https://docs.microsoft.com/en-us/windows/wsl/troubleshooting) for other common errors and their solutions.

* **Installation failed with error 0x80070003**
  + The Windows Subsystem for Linux only runs on your system drive (usually this is your C: drive). Make sure that distros are stored on your system drive:
  + Open **Settings** -> **Storage** -> **More Storage Settings: Change where new content is saved** 
* **WslRegisterDistribution failed with error 0x8007019e**
* The Windows Subsystem for Linux optional component is not enabled:
* Open **Control Panel** -> **Programs and Features** -> **Turn Windows Feature on or off** -> Check **Windows Subsystem for Linux** or using the PowerShell cmdlet mentioned at the beginning of this article.

**Installation and configuration of Apache Airflow**

Windows has come a long way in a short time by accepting and promoting open source software and developers in an increasing way. They have also joined many Opensource organizations and contributed to many development efforts. Nothing illustrates this more than their development of WSL (**Windows Subsystem for Linux**), which allows you to install a Linux distribution on your PC alongside Windows without having to worry about VMs or containers. This is great for developers that work with tools that only run in Linux, such as Apache Airflow and has low overhead.

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. 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.

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

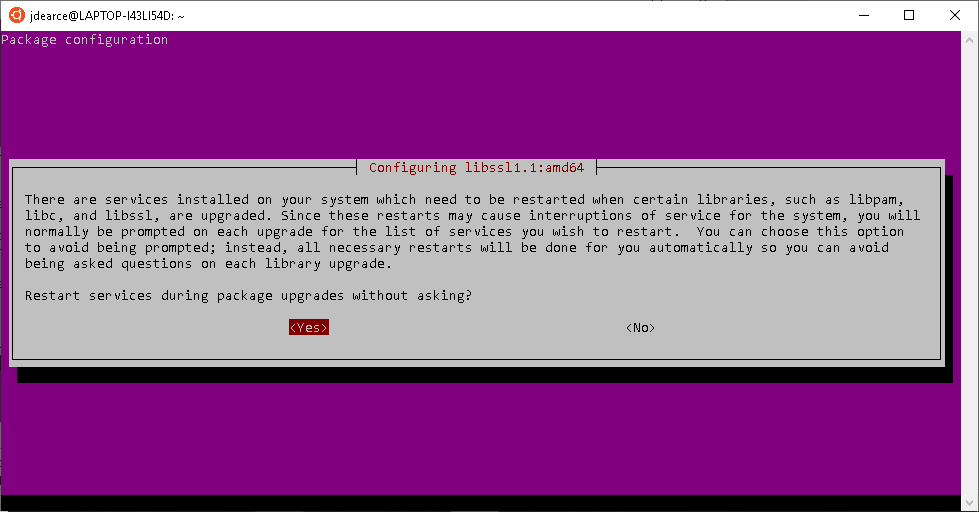
While both VMs and Docker are great options, this article will be about setting up Airflow on WSL out of the box for very simple access to Airflow with little overhead, but this is not for a production system. This is true for several reasons, we are running on only one system not a cluster, two we are using the Serial executer, three we need to upgrade the out of the box install to a real database Postgres or MySQL.

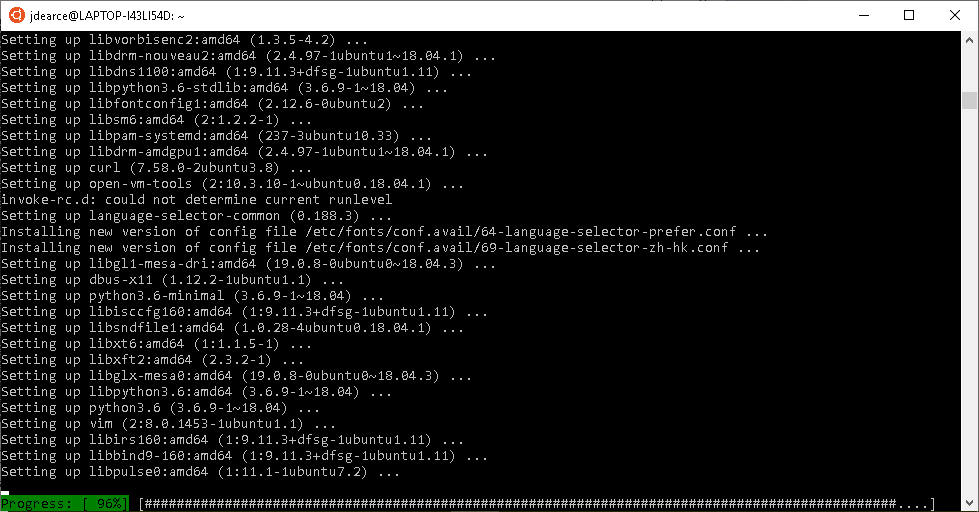
**Installing Airflow on WSL**

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 cover later in this article.

Make sure you follow the link to initialize your new distro instance. The installation will create a new Linux user with a home directory, and you will have to create a password. Every time you need to issue a privilege command that uses sudo you will need to use that password.

Also run **sudo apt update && sudo apt upgrade** to upgrade your distro to the current version for your Linux distribution.





**Warning**

Files in the **Linux file system should not be accessed from Windows**, as they can end up getting corrupted. Do not open or edit any files from processes running in Linux with Windows applications such as explorer, notepad, atom, or another IDE.

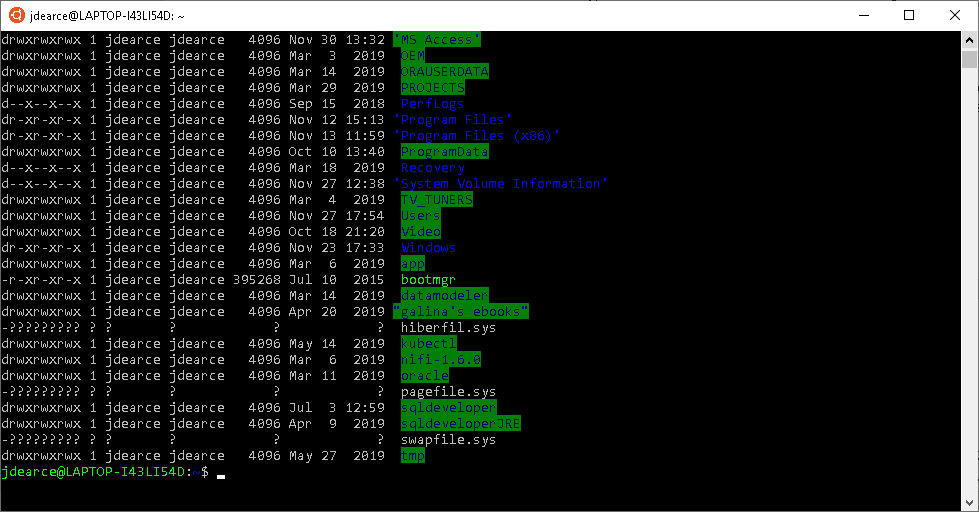
Linux can access files in the Windows file system. For example, you can create a folder in **C:/Users/your\_username/airflow** and use it as your **AIRFLOW\_HOME** directory. You can edit files in Airflow home using your favorite Windows editors (notepad, atom, PyCharm, etc.) and then use them in Airflow on WSL.

**Additional Setup**

Once you have WSL Linux installed, and updated, launch Ubuntu to which will open a terminal window into your Linux home directory and leave you on the command line. If you type **pwd** (present working directory), you should see **/home/user\_name**, indicating that you are in the Linux file system in your Linux home directory. When you use the **~** this will reference your home path.

To list the files in your current directory use the command **ls -Al** to view the contents of this directory, including hidden objects. In the home directory you’ll see the file **.bashrc**. This file is a configuration file for Bash shell, which is created every time you open a terminal. You can use this file to add new settings such as environment variables that you want to apply to your bash sessions every time you start Linux.

The Windows file system is available to you in Linux and is located at **/mnt/c**. If you type ls **/mnt/c** you will see the contents of the **C:/** in Windows. Note: you will see some permissions that indicate errors, you can ignore those.



It can be a bit of a pain to do this all the time you want see a directory or file on the **C:/** drive to type **/mnt/c** to get to your Windows files.

In fact, if you end up wanting to use Docker in WSL (which you probably will), this actually won’t work at all. So let’s change how the Windows file system is mounted in Linux.

Type **sudo nano /etc/wsl.conf**. This will open the Linux **nano** text editor for editing files. Change the file structure to match the following:

**[automount]**

**root = /**

**options = "metadata"**

Enter **ctrl + s**, **ctrl + x** to save the changes and exit nano. Now sign out, then back in to windows for these changes to take effect. Open Ubuntu and type **ls /** and you should see that c is now mounted on root / instead of **/mnt/**.

**Python 3**

Confirm you have Python 3 installed with **python3 --version**.

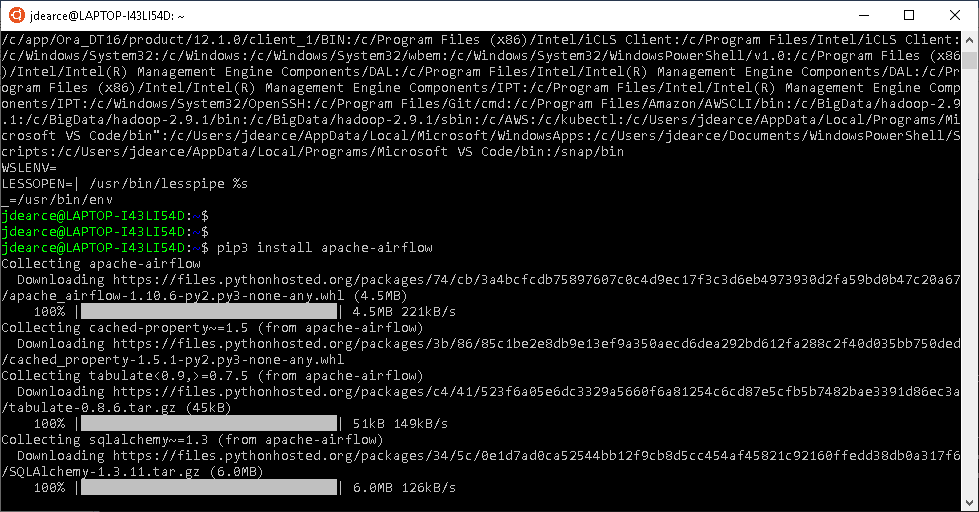
Now let's install pip: **sudo apt update**, **sudo apt install python3-pip** that should do the trick.

Run **pip3 --version** to make sure it's installed correctly.

**Installing Airflow**

There is no difference between installing Airflow in any normal **Ubuntu environmen**t. If you want to, you can include other Airflow modules such as **postrges**. Some of these may require dependencies to be installed on Ubuntu using **sudo apt** **install** [your\_dependency].

Now we will run the pip command to install Airflow **pip3 install apache-airflow**.



Now let’s set the **AIRFLOW\_HOME** (Airflow looks for this environmental variable whenever an Airflow CLI commands is run). When you run **airflow initdb** it will create all the Airflow stuff in this directory. We want the Airflow home to be in the Windows file system so you can edit all the files from Windows based tools if necessary.

Running **nano ~/.bashrc** will open up the file mentioned earlier for setting environment variables in your bash session. On a new line add the following statement **export AIRFLOW\_HOME=/c/Users/user\_name/AirflowHome** statement replacing **user\_name** with your actual windows home folder name. It should look something like this:

####

From the **.bashrc** bash configuration file add the environmental variable below the following three lines.

**# ~/.bashrc: executed by bash(1) for non-login shells.**

**# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)**

**# for examples**

**export AIRFLOW\_HOME=/c/Users/user\_name/AirflowHome**

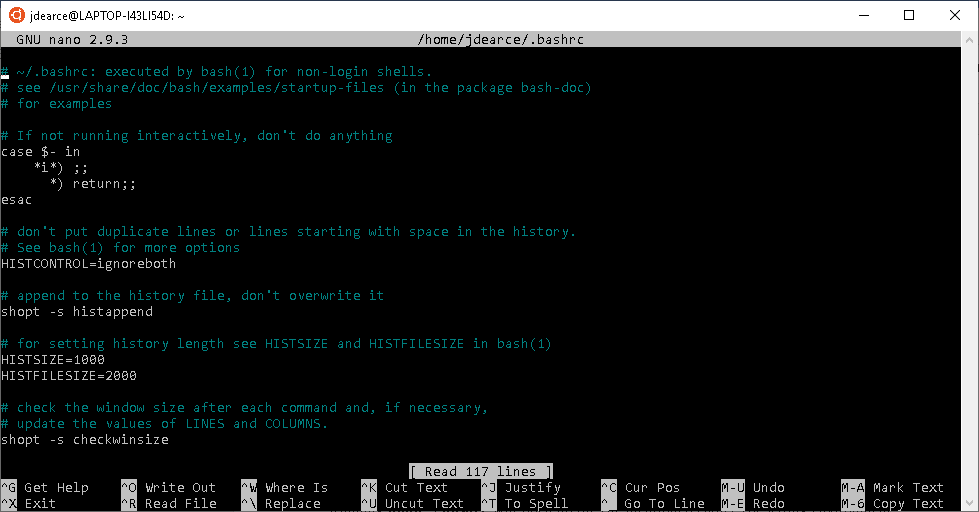
My version of the export command for Ubuntu is:

**export AIRFLOW\_HOME=/c/Users/jdearce/airflow**

For Kali it’s.

**export AIRFLOW\_HOME=/c/Users/jdearce/airflow2**

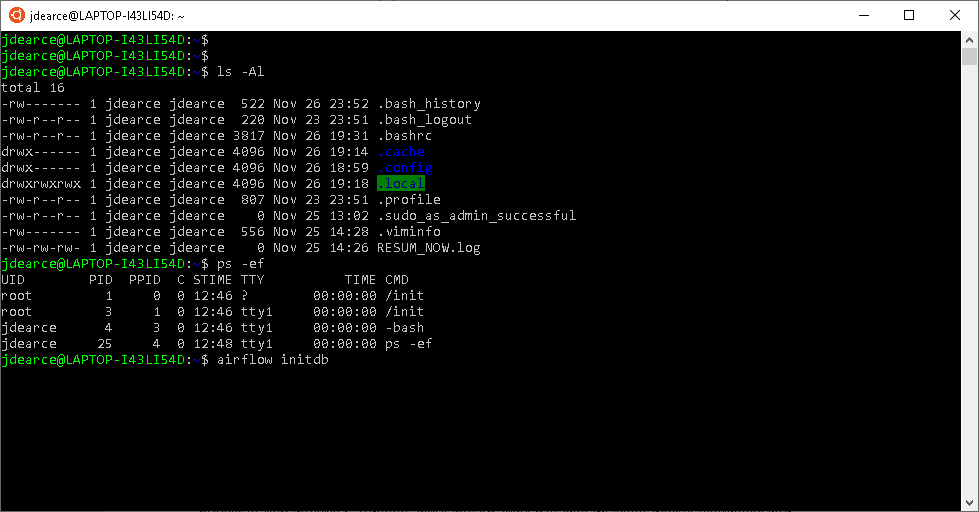
**ls -Al =/c/Users/jdearce**



Use **ctrl + s,** **ctrl + x** to save and exit nano. Now, anytime you open a bash session in Ubuntu, the **AIRFLOW\_HOME** environment variable will be set to **AirflowHome** or as in my **airflow** folder in the Windows Users directory.

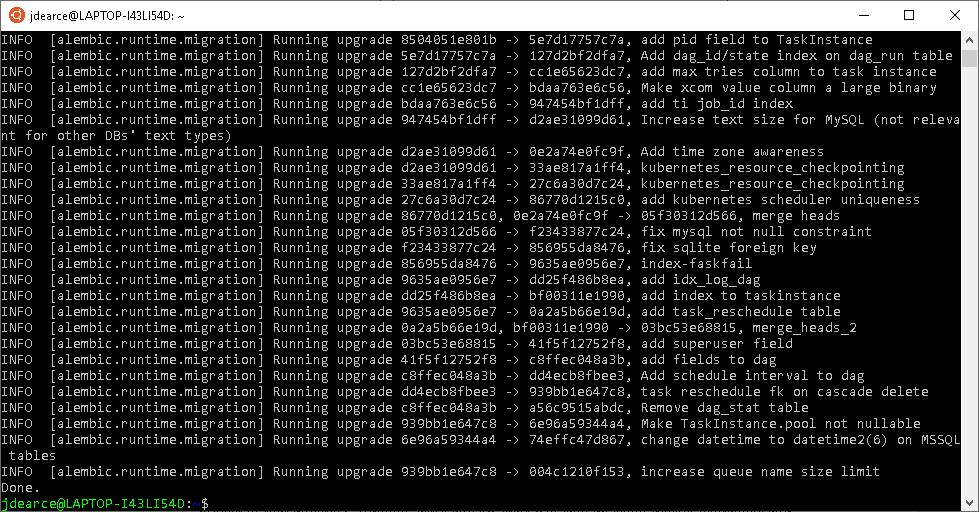
.

Problem with the **initdb** command is that **SQLite** is not a scalable database

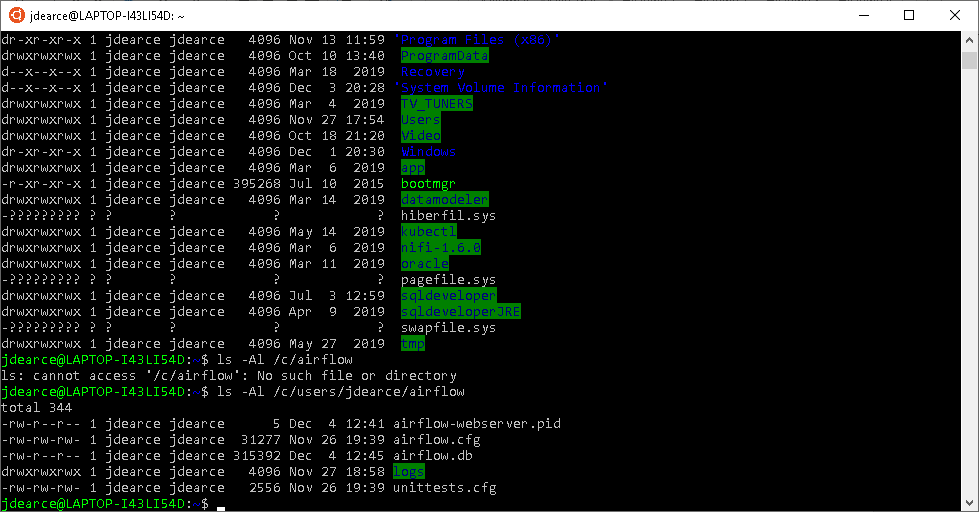


Close and reopen Ubuntu. **airflow version** should now show you the version of airflow you have

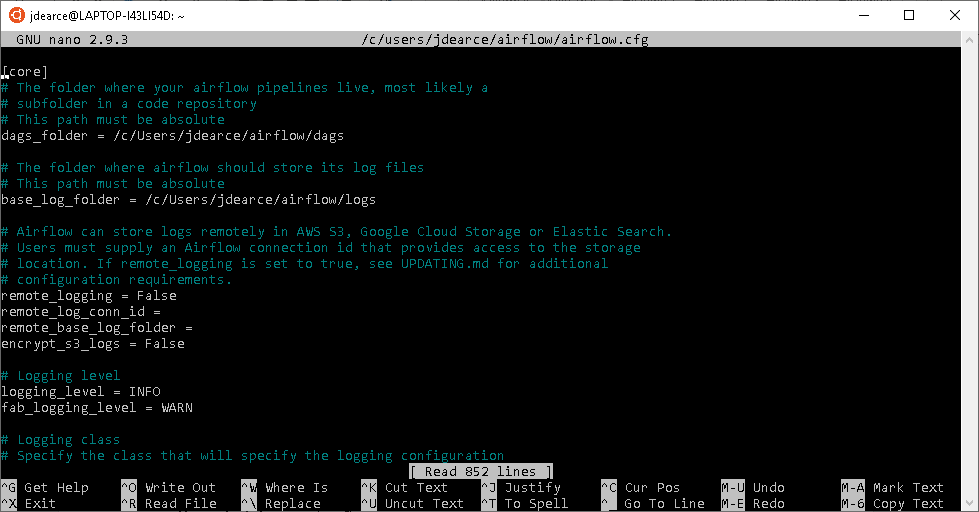
installed without any errors and running **airflow initdb** should populate your **AIRFLOW\_HOME** folder with a clean setup for Airflow. You must run the airflow scheduler first and then airflow webserver since the webserver is dependent on airflow scheduler.



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

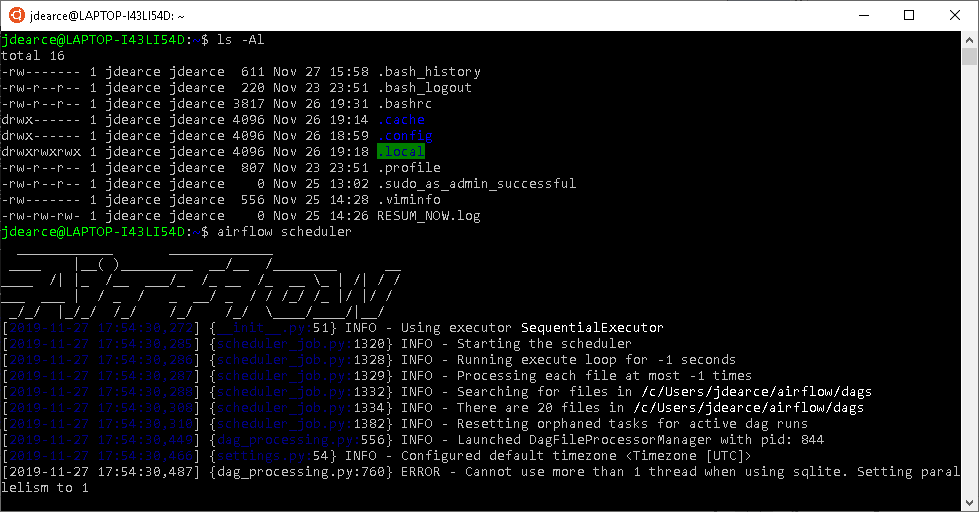


We will edit the airflow.cfg file by typing the command **nano /c/users/jdearce/airflow/airflow.cfg**

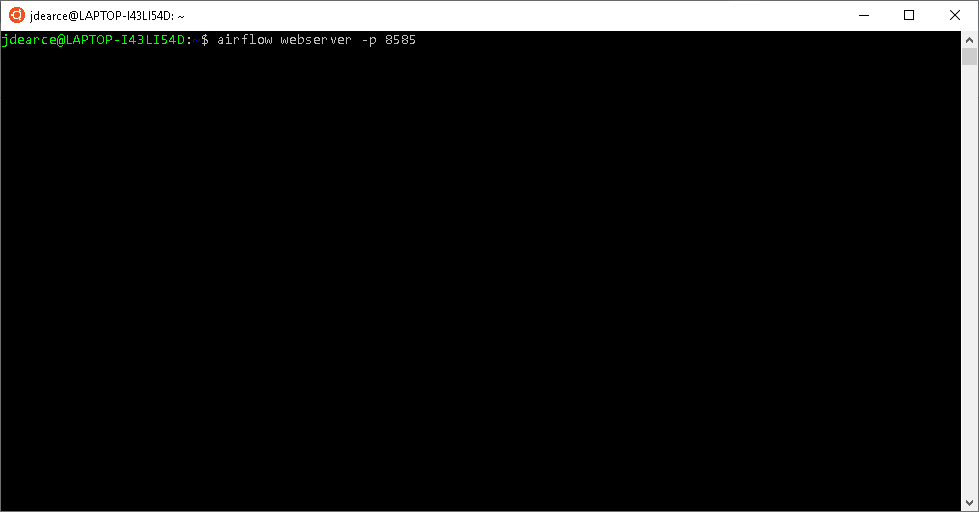


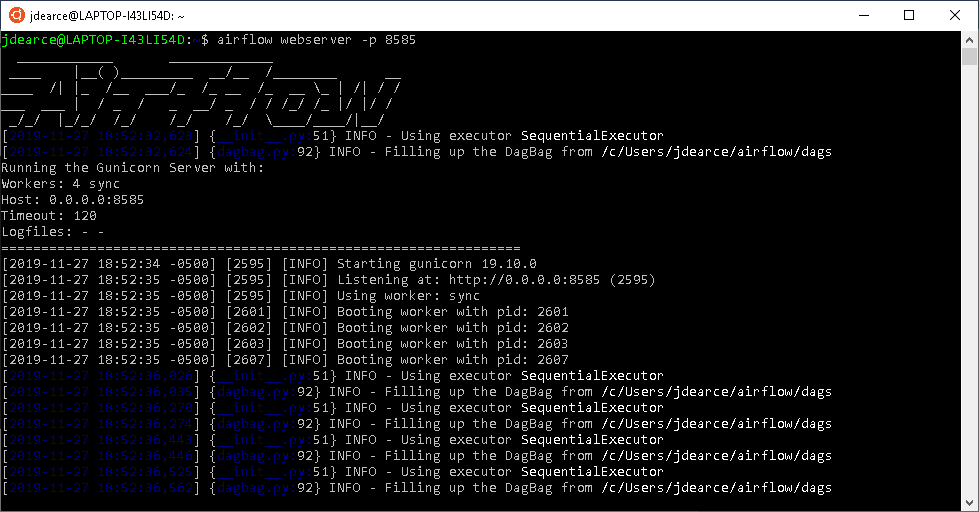
You are now in the main Airflow configuration file for the Airflow system.

Starting Airflow is two main processes the web server and the scheduler the commands below will start them. The first is run the scheduler type the command at the Ubuntu command prompt which was opened as administrator ‘**airflow scheduler**’.

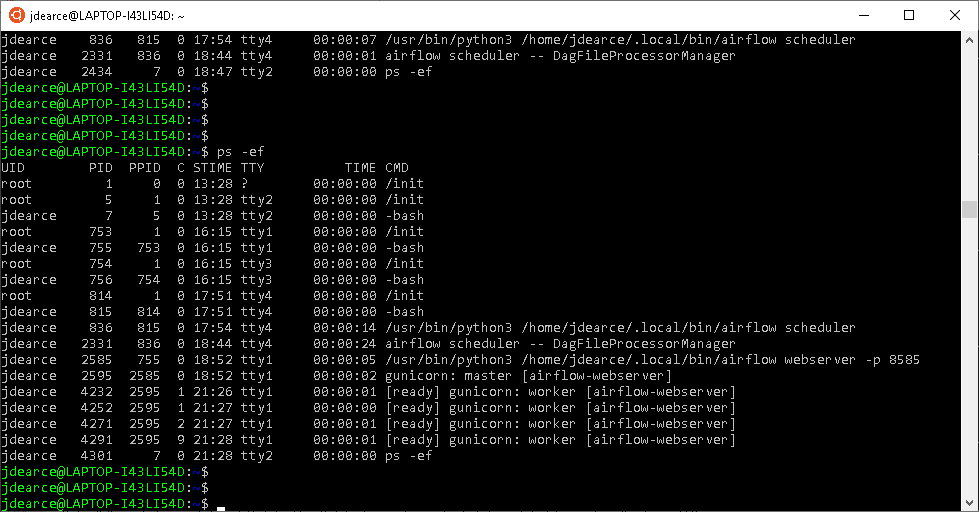


Second we will run the command at the command-line interface ‘**airflow webserver -p 8080**’ I used the port 8585 to eliminate any conflicts with Hortonworks Ambari which uses 8080.

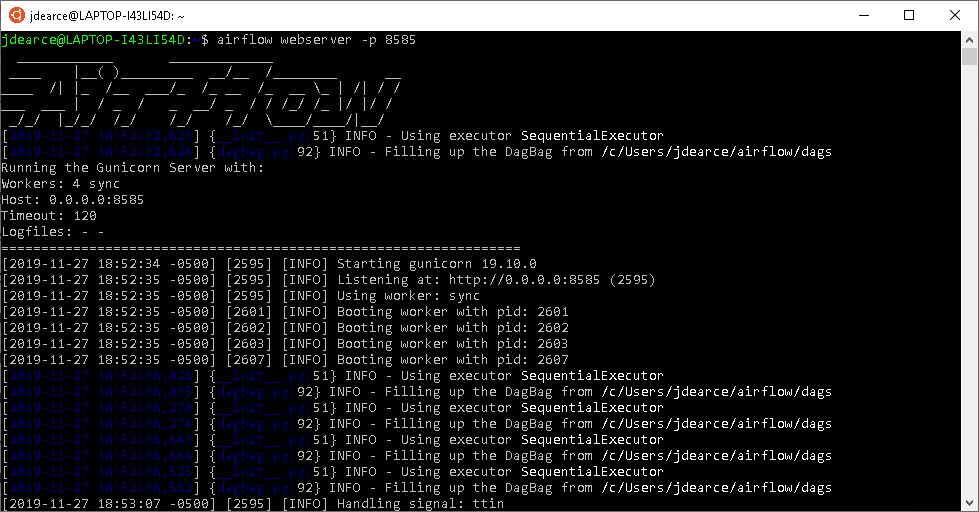




The screen below lists all the processes for Airflow on Ubuntu for the web server and the Scheduler.



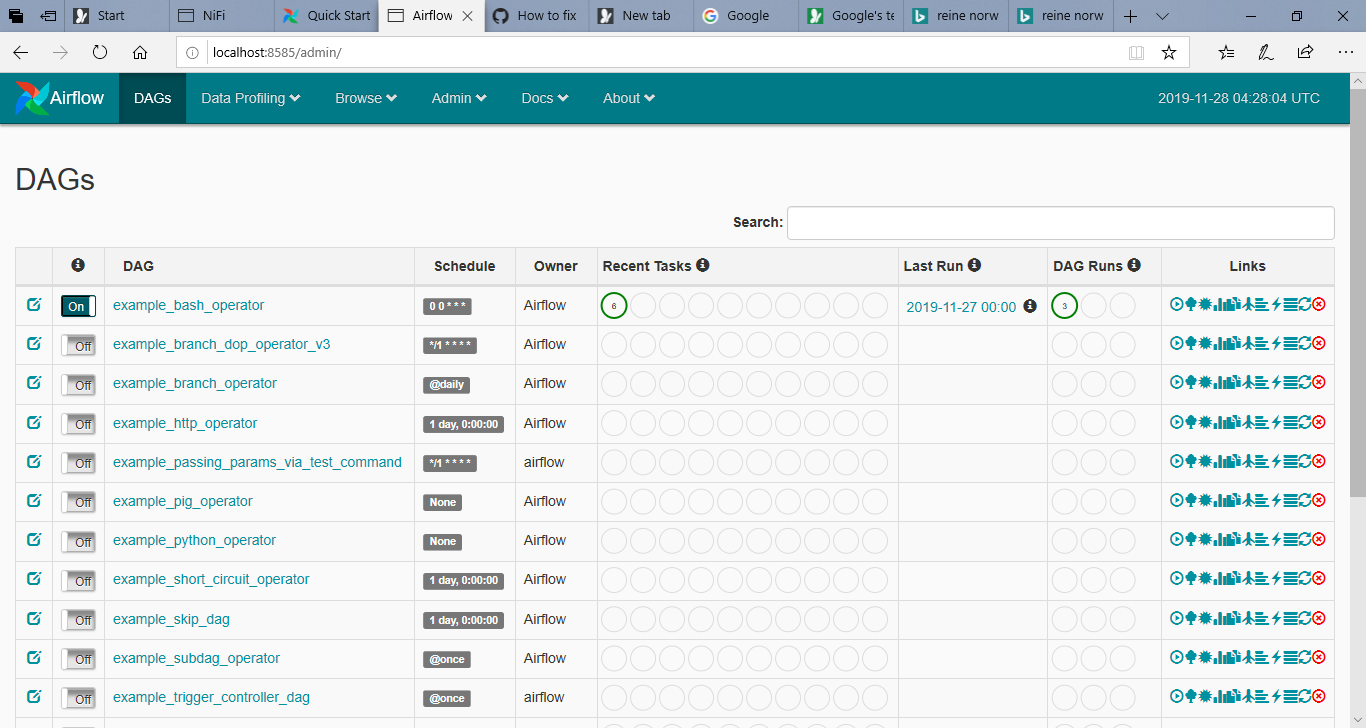
Web server log is in the window below.



Below is the webserver UI to the Airflow system where you can schedule DAG’s or add them to the system. Also the UI has many other functions which are held in the six menus drop downs.

**DAGs View**

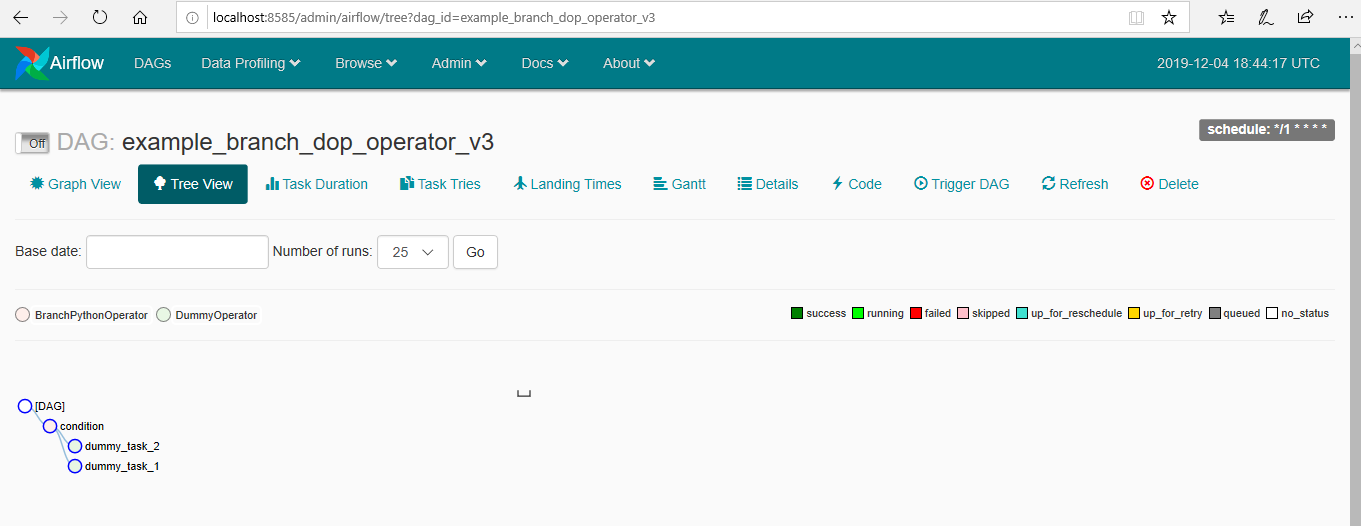
List of the DAGs in your environment, and a set of shortcuts to useful pages. You can see exactly how many tasks succeeded, failed, or are currently running at a glance.



To run the Airflow scheduler on Ubuntu you need to run Ubuntu as admin or you will get access errors shown below.

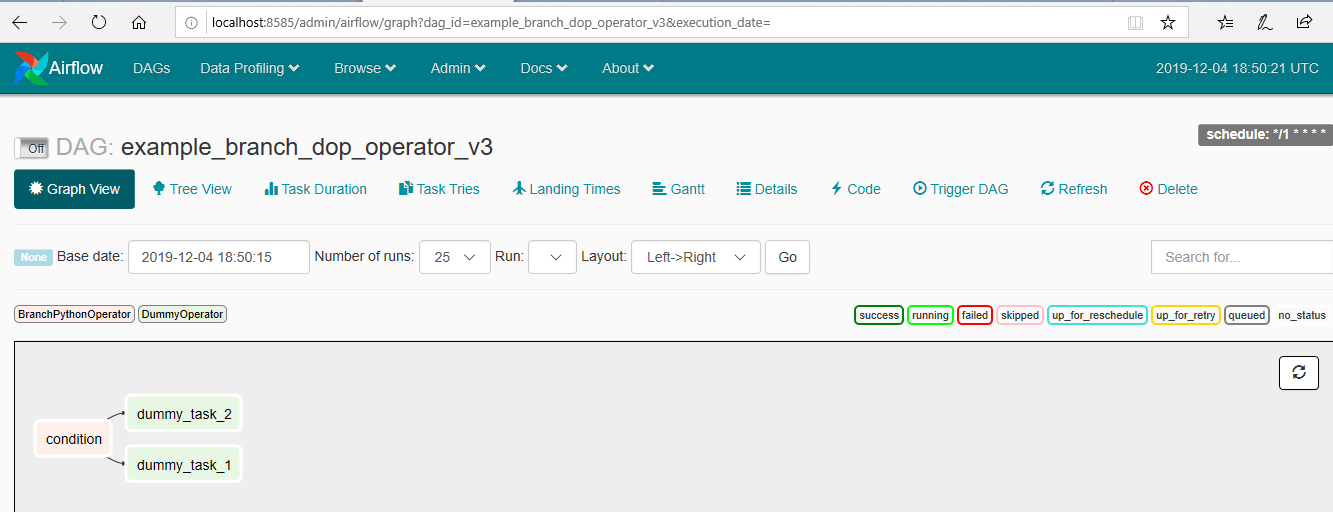
**Tree View**[¶](https://airflow.apache.org/ui.html#_blank)

A tree representation of the DAG that spans across time. If a pipeline is late, you can quickly see where the different steps are and identify the blocking ones.



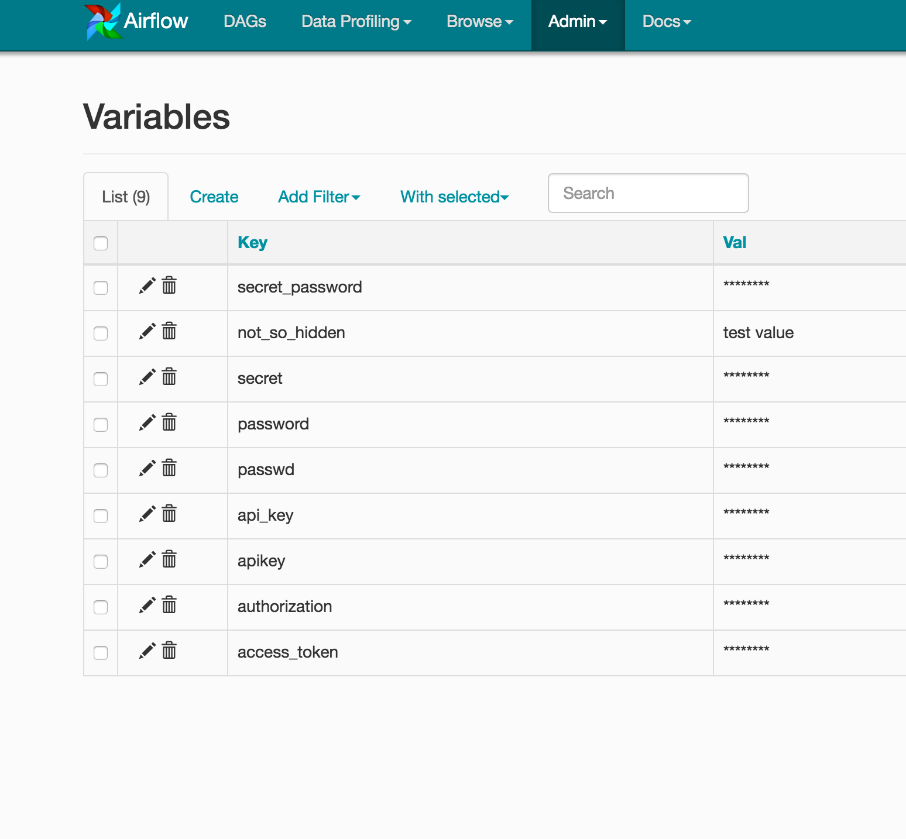
**Graph View**[¶](https://airflow.apache.org/ui.html#_blank)

The graph view is perhaps the most comprehensive. Visualize your DAG’s dependencies and their current status for a specific run.



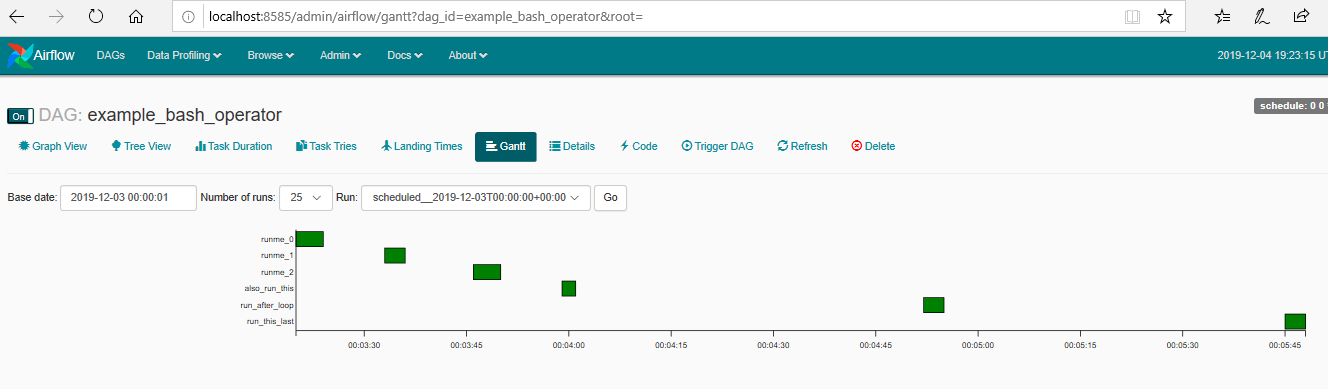
**Variable View**[¶](https://airflow.apache.org/ui.html#_blank)

The variable view allows you to list, create, edit or delete the key-value pair of a variable used during jobs. Value of a variable will be hidden if the key contains any words in (‘password’, ‘secret’, ‘passwd’, ‘authorization’, ‘api\_key’, ‘apikey’, ‘access\_token’) by default, but can be configured to show in clear-text.



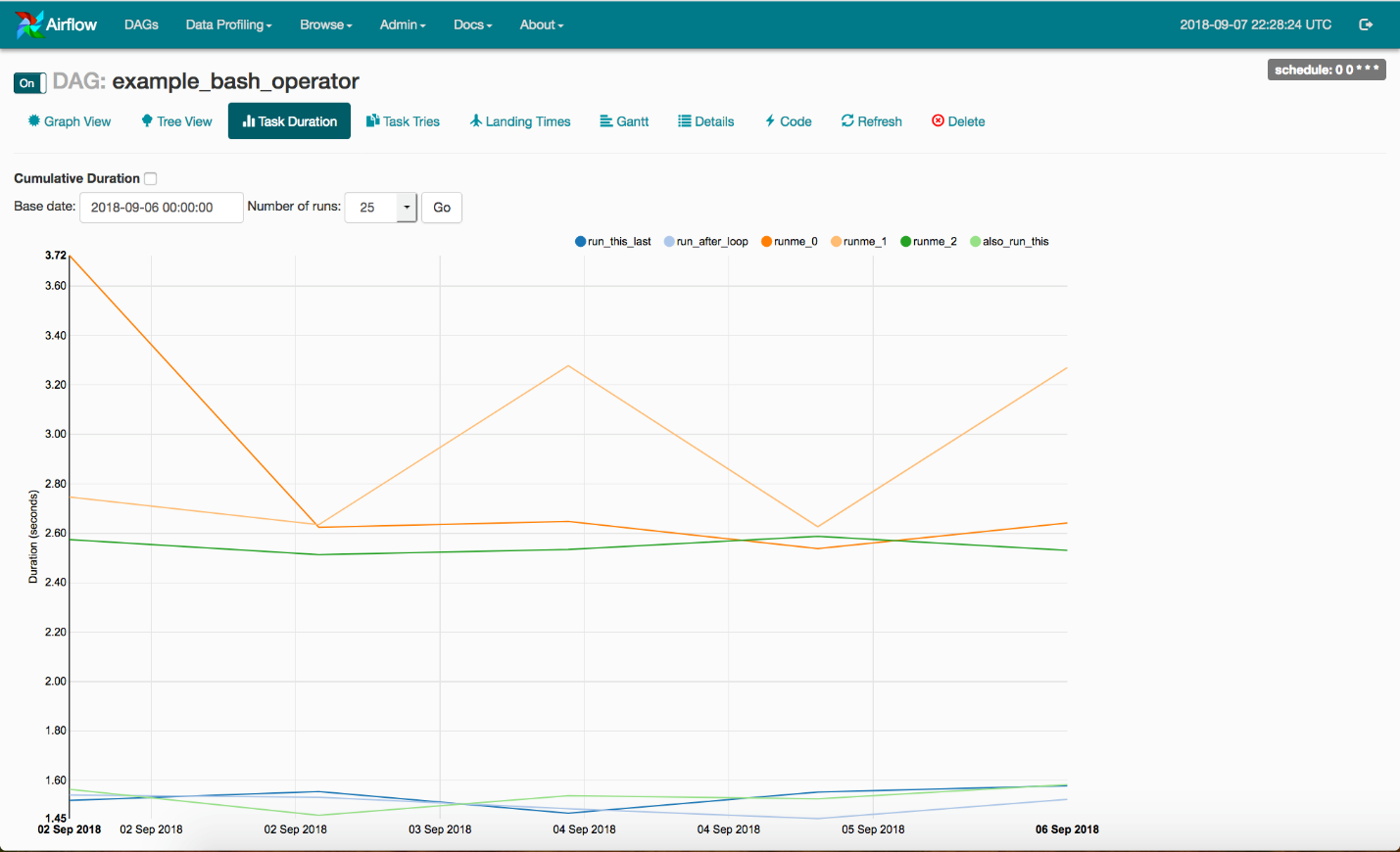
**Gantt Chart**[¶](https://airflow.apache.org/ui.html#_blank)

The Gantt chart lets you analyze task duration and overlap. You can quickly identify bottlenecks and where the bulk of the time is spent for specific DAG runs.



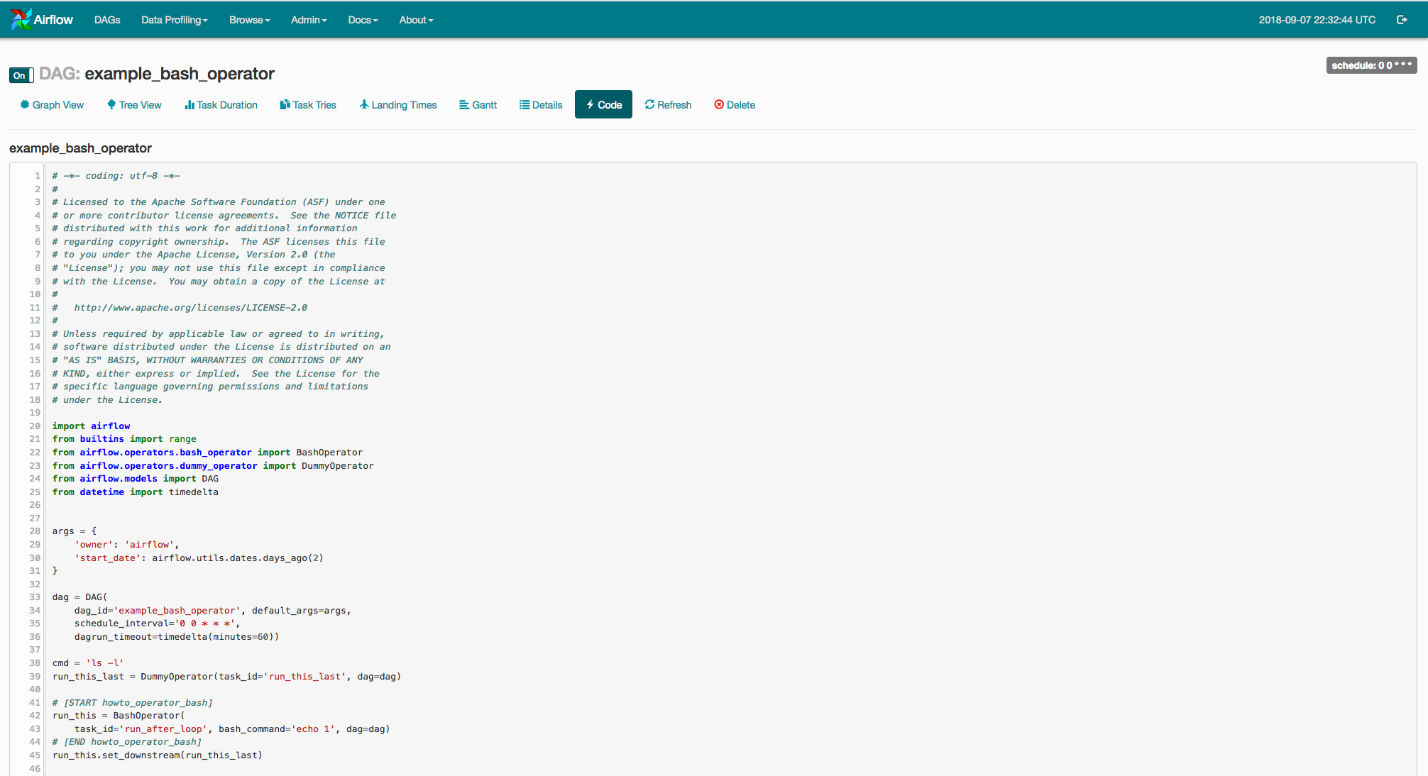
**Task Duration**[¶](https://airflow.apache.org/ui.html#_blank)

The duration of your different tasks over the past N runs. This view lets you find outliers quickly understand where the time is spent in your DAG over many runs.



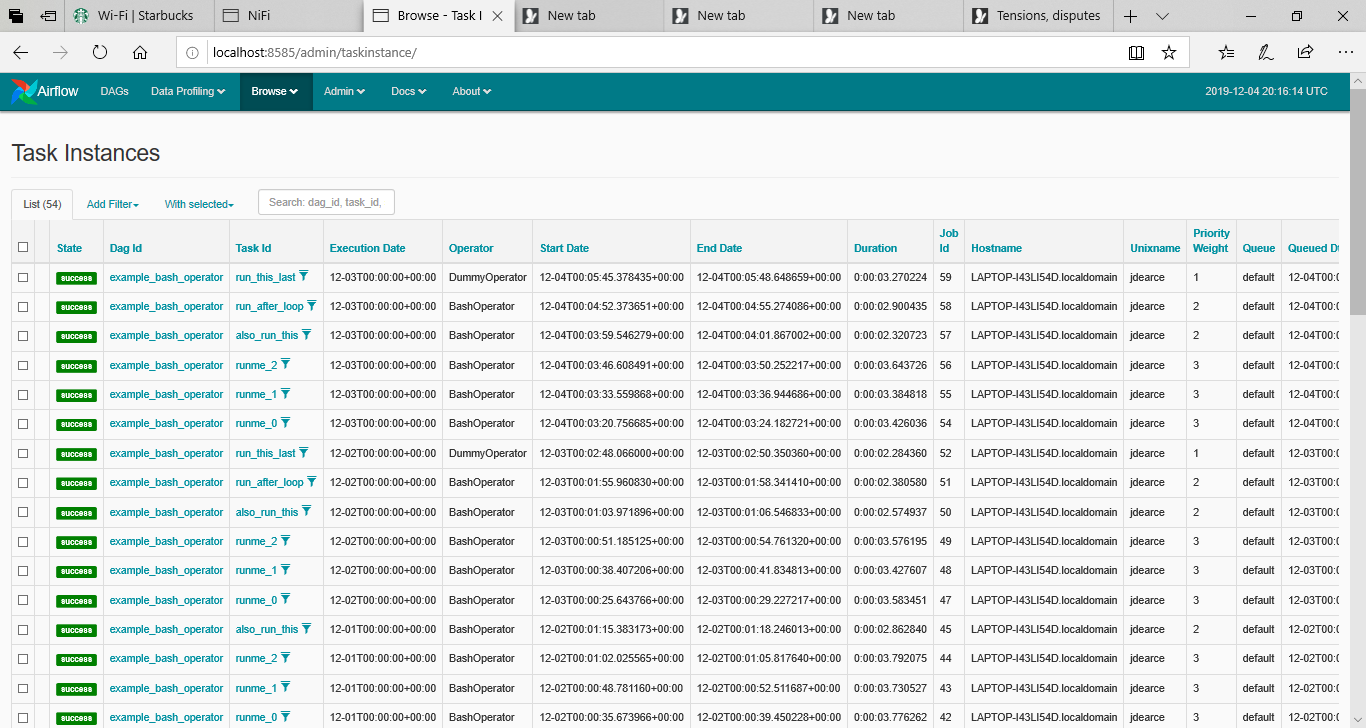
**Code View**

Transparency is everything. While the code for your pipeline is in source control, this is a quick way to get to the code that generates the DAG and provide yet more context.



**Task Instance Context Menu**[¶](https://airflow.apache.org/ui.html#_blank)

From the pages seen above (tree view, graph view, Gantt, …), it is always possible to click on a task instance, and get to this rich context menu that can take you to more detailed metadata, and perform some actions.



**Conclusion**

This basic configuration is Airflow on training wheels on a bicycle and not meant for prime-time but for the sole purpose of learning to navigate the Airflow system.

In the next installment I will cover a more advance configuration which is not hampered by **SQLite** or the **SerialExecutor**