Airflow Installation

The ubuntu server for airflow is available via "ssh biadmin@prox1.hetzner.bksrv.net -p 3122" (can use PutTTy for this purpose).
Internal ip is 192.168.100.31 if you want to connect from one of the windows remote servers.

Once the airflow is up and running, it should work from the windows remote server from a browser on http://prox1.hetzner.bksrv.net/.

> The host url is prox1.hetzner.bksrv.net. JD has forwared port 80 on the host to 8080 on the airflow machine It doesn't really matter on which port airflow runs, incoming http request --> prox1.hetzner.bksrv.net:80 --> 192.168.100.31:<*WhateverPortYouWant*>

Start by updating the package list using the following command:

```
sudo apt update
```

Also run

```
sudo apt upgrade
```

to make sure everything is up to date.

Use the following command to install pip for Python 3:

```
sudo apt install python3-pip
```

The command above will also install all the dependencies required for building Python modules.

Once the installation is complete, verify the installation by checking the pip version:

```
pip3 --version
```

The version number may vary, but it will look something like this:

```
pip 20.0.2 from /usr/lib/python3/dist-packages/pip (python 3.8)
```

Airflow comes with sqlite database backend, this database system will not be able to run data pipeline on webUI. SQLite is able to run the tasks only sequentially and is good for just testing purposes. We would require to have more powerful database system like PostgreSQL, that comes with robust feature set, data integrity and extensibility. We will install PostgreSQL and configure it to use with Airflow.

```
sudo apt-get install postgresql postgresql-contrib
```

As we have already installed postgresql database using above mentioned command. We will now create a database for airflow and grant access to a sudo user.

Creating new Linux user *airflow*.

```
sudo adduser airflow
```

Install python package for postgres

```
sudo apt-get install python3-psycopg2
```

Lets access to psql, a command line tool for Postgres.

```
sudo -u postgres psql
```

After logging in successfully, we will get psql prompt (postgres=#). We will create a new user and provide privileges to it.

```
CREATE ROLE biadmin;
CREATE DATABASE airflow_db;
GRANT ALL PRIVILEGES on database airflow_db to biadmin;
ALTER ROLE biadmin SUPERUSER;
ALTER ROLE biadmin CREATEDB;
ALTER USER biadmin WITH PASSWORD 'ubuntu';
ALTER ROLE biadmin WITH LOGIN;
GRANT ALL PRIVILEGES ON DATABASE airflow_db to biadmin;
```

Commands:

```
\du           to list all users
\list         to list all existing databases
\dt           to list all tables
```

Now connect to airflow database and get connection information.

```
postgres-# \c airflow_db
```

After successful connection, prompt will be changed to airflow-#. We will verify this by fetching connection info

```
airflow=# \conninfo
```

\conninfo command output:

```
You are connected to database "airflow_db" as user "postgres" via
socket in "/var/run/postgresql" at port "5432".
```

Restart the PostgreSQL service:

```
sudo service postgresql restart
```

Check if you have access to created db as user biadmin:

```
psql -d mydb -U myuser
psql -d airflow_db -U biadmin
```

If you decide to use **Postgres**, we recommend using the `psycopg2` driver and specifying it in your SqlAlchemy connection string. Also note that since SqlAlchemy does not expose a way to target a specific schema in the Postgres connection URI, you may want to set a default schema for your role with a command similar to `ALTER ROLE username SET search_path = airflow, foobar;`

Install Airflow

As PostgreSQL is already installed and configured. Next, We will install Airflow and configure it.

Set AIRFLOW_HOME environment variable to ~/airflow.

```
export AIRFLOW_HOME=~/airflow
```

On a typical installation this should install to the user's home directory. In this case it is located at **/home/ubuntu/airflow**

Install Airflow and its packages.

```
sudo pip3 install apache-airflow
# pip3 install apache-airflow[postgres, mssql, celery, rabbitmq]
```

for other subpackages like celery, async, crypto, rabbitmq etc., you can check apache airflow installation page.

We can then trigger the initialization of its back-end database with the command:

```
airflow initdb
```

This will create a default configuration file and initialize an empty back-end DB. In the freshly created configuration file `${AIRFLOW_HOME}/airflow.cfg`, you'll find all the configuration parameters that specify how Airflow will run. There are a few important settings that we need to change in the **airflow.cfg** file to get the engines started.

Now airflow.cfg file should be generated in airflow home directory, we will tweak some configuration here to get better airflow functionality. We might want to edit the executor and connection string in airflow.cfg. The executor class that airflow should use. Choices include SequentialExecutor, LocalExecutor, CeleryExecutor

executor = LocalExecutor
The LocalExecutor can parallelize task instances locally.

Out of the box, Airflow uses a sqlite database, which you should outgrow fairly quickly since no parallelization is possible using this database backend. It works in conjunction with the `airflow.executors.sequential_executor.SequentialExecutor` which will only run task instances sequentially. While this is very limiting, it allows you to get up and running quickly and take a tour of the UI and the command line utilities. So Airflow uses the `SequentialExecutor` by default, but we'll change that to use the more useful `LocalExecutor`.

Once you've setup your database to host Airflow, you'll need to alter the SqlAlchemy connection string located in your configuration file `$AIRFLOW_HOME/airflow.cfg`. You should then also change the "executor" setting to use "LocalExecutor", an executor that can parallelize task instances locally.

We will be using LocalExecutor instead of SequentialExecutor which come by default with airflow. Change

```
executor = LocalExecutor
```

Change the connection for your specified database in the airflow.cfg:

```
# The SqlAlchemy connection string to the metadata database.
# SqlAlchemy supports many different database engines
# The format is:
sql_alchemy_conn =
#postgresql+psycopg2://$your_db_user:$your_db_password@$your_postgre#s_d
b_host:$postgres_port/$db_name
in our case it would be
sql_alchemy_conn = postgresql+psycopg2://ubuntu:ubuntu@localhost
/airflow_db
```

Running Airflow Webserver and Scheduler