Installing Airflow

Created By	
Stakeholders	
Status	
Type	
Created	@February 23, 2022 6:05 PM
Last Edited Time	@March 7, 2022 2:14 PM
Last Edited By	

This install is for Ubuntu 20.04 OS *(Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-1029-gcp x86_64))*, it may also be applicable to previous Ubuntu version, but it is not tested.

1. Open a VM instance in GCP Compute Engine, select Ubuntu 20.04 as the disk. Don't forget to allow HTTP and HTTPS traffic, otherwise you can't connect to Airflow webserver remotely.

2. Once we connect to the VM, we run

```
# switch to root user
sudo su
```

command to act as the root user and run the following commands to update system packages:

```
sudo apt update && sudo apt upgrade -y
# apt list --upgradable # see packages that can be updated
```

3. Ubuntu comes with Python installed in it. In 20.04, we can verify this by typing

```
# run python
python3 --version
```

and we will see Python 3.8.10 in this specific Ubuntu OS. We can check the Airflow dependencies in Airflow documentation here -> https://airflow.apache.org/docs/apache-airflow/stable/installation/prerequisites.html

Airflow ≥2.1.2 versions work with Python 3.8.

4. However, pip package manager is not available in the VM, so we need to install it by typing

```
# install pip
apt install python3-pip

# check pip version
pip --version # or pip3 --version -> both pip and pip3 directs to the same python installation
```

We will need pip package manager to install airflow and some other packages that are required.

5. Now, it is time to install the Airflow system dependencies that are needed for Linux system. With the below command, we can install them at once. We can find the dependencies in this link → https://airflow.apache.org/docs/apache-airflow/stable/installation/dependencies.html

```
sudo apt-get install -y --no-install-recommends \
        freetds-bin \
        krb5-user \
        ldap-utils \
        libsasl2-2 \
        libsasl2-modules \
        libssl1.1 \
        locales  \
        lsb-release \
        sasl2-bin \
        sqlite3 \
        unixodbc
```

Notice that we didn't include "libffi6" in the installation, because if we try

```
sudo apt-get install libffi6
```

we will see "E: Unable to locate package libffi6" error. It doesn't cause any problem not to install it, but in case it does, here is a link for a possible solution → https://forums.linuxmint.com/viewtopic.php?t=186690

6. Next, let's install MySQL and verify the root user using the command below. This installs the latest MySQL release, which is 8.0 for this tutorial.

https://cloud.google.com/architecture/setup-mysql#ubuntu

```
# install mysql-server
sudo apt-get -y install mysql-server

# verifies root user authenticated with auth_socket plugin
echo "SELECT user, authentication_string, plugin, host
        FROM mysql.user WHERE user='root' ;" \
     | sudo mysql -t -u root
```

7. Then, we should create a database for Airflow to use. After activating mysql, we create a database and a user with the following commands. We can find detailed explanation here → https://airflow.apache.org/docs/apache-airflow/stable/howto/set-up-database.html and here → https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql

```
# activate mysql
mysql

# create database, create user and grant all privileges
mysql> CREATE DATABASE airflow_db;
mysql> CREATE USER 'airflow_user' IDENTIFIED BY 'airflow_pass';
mysql> GRANT ALL PRIVILEGES ON airflow_db.* TO 'airflow_user';

# see if the user has been created
mysql> use mysql;
mysql> select * from user;
```

8. We need to install PyMySQL as the driver for MySQL. That's why we should install it using:

```
# install pymysql
pip install pymysql
```

9. Now, let's start installing Airflow. It is best to install airflow using constaint files. We will install from PyPI using constraint files. We will also include celery executor within the Airflow installation. We know that the Python version is 3.8, that's why we will install the latest Airflow version (2.2.3) using the constraint files for Python 3.8. Below is the command to do these and here is the link for details → https://airflow.apache.org/docs/apache-airflow/stable/installation/installing-from-pypi.html

```
# install airflow
# https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt
pip install "apache-airflow[celery]==2.2.3" --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.2.3/constraints-3.
```

10. As we installed Airflow 2.2.3 with constraints for Python 3.8. Now is time to configure the airflow.cfg file to connect our MySQL database with airflow, but before that, we should install one more thing which is RabbitMQ Server. It is the message broker the celery executor is going to use to queue the tasks. Details are here → https://insaid.medium.com/set-up-celery-flower-and-rabbitmq-for-airflow-1ed552fe131f

    As RabbitMQ Server is based on erlang programming language, we first need to install Erlang programming language on our system and then we will instal rabbitmq-server using:

    ```
    # install erlang first
    apt-get install erlang

    # then install rabbitmq-server
    apt-get install rabbitmq-server
    ```

    Next, we should start the rabbitmq-server and check if it is available as a running service:

    ```
    # start rabbitmq-server
    service rabbitmq-server start

    # see if it is running
    service --status-all
    # Below is the output
    '
    [ + ]  apparmor
    [ + ]  apport
    [ + ]  atd
    [ + ]  avahi-daemon
    [ - ]  bluetooth
    [ + ]  chrony
    [ - ]  console-setup.sh
    [ + ]  cron
    [ - ]  cryptdisks
    [ - ]  cryptdisks-early
    [ + ]  dbus
    [ - ]  gdm3
    [ - ]  grub-common
    [ - ]  hwclock.sh
    [ - ]  iscsid
    [ - ]  keyboard-setup.sh
    [ + ]  kmod
    [ - ]  lvm2
    [ - ]  lvm2-lvmpolld
    [ + ]  multipath-tools
    [ + ]  mysql
    [ + ]  network-manager
    [ - ]  open-iscsi
    [ - ]  open-vm-tools
    [ - ]  plymouth
    [ - ]  plymouth-log
    [ - ]  pppd-dns
    [ + ]  procps
    [ - ]  pulseaudio-enable-autospawn
    [ + ]  rabbitmq-server # !!! HERE IT IS RUNNING !!!
    [ + ]  redis-server
    [ - ]  rsync
    [ + ]  rsyslog
    [ - ]  saned
    [ + ]  saslauthd
    [ - ]  screen-cleanup
    [ + ]  ssh
    ```

```
[ + ]  udev
[ + ]  ufw
[ + ]  unattended-upgrades
[ + ]  uuidd
[ - ]  x11-common
'
```

Then, we should enable some required plugins for RabbitMQ Management Dashboard:

```
# enable rabbitmq-management plugin
rabbitmq-plugins enable rabbitmq_management
```

Finally, we can open the web UI for the RabbitMQ, the default listening port is 15672. We can open the login page by typing http://localhost:15672/ or http://{ip for the host machine}:15672/ on the browser. The default user is "**guest**" with the password "**guest**". If we try to open the UI remotely, we will not be able to login with the guest user. We need to create an admin user with all the privileges granted. Let's create a new user named "***admin***" with the password "***admin***" by typing the following command:

```
# create a user with username "admin" and password "admin"
rabbitmqctl add_user admin admin

# set "admin" user as an administrator
rabbitmqctl set_user_tags admin administrator

# give full permission to "admin" user to read/write data
rabbitmqctl set_permissions -p / admin ".*" ".*" ".*"
```

Now, we can login to the web UI with the "admin" user.

---

11. Next, let's check our workers. We already installed celery bundle within the Airflow installation by typing "apache-airflow[celery]". Celery and Flower come as a bundle in Airflow. That's why we don't need to make an additional installation for the Celery Executor. So, let's check Flower at this step. **Flower** is a web based tool for monitoring and administrating **Celery** clusters. With the below commandi we can open the UI:

    ```
    # open flower ui
    airflow celery flower
    ```

    By default, it is assigned to port number 5555, so you can go to http://localhost:5555/ or http://{ip for the host machine}:5555/ and you should be able to see the web UI.

---

12. Next, let's configure some airflow.cfg options to:

    a. Link MySQL database for airflow backend

    b. Set CeleryExecutor as the executor type

    c. Link RabbitMQ broker to Airflow Celery Executor

    We will do these changes in the airflow.cfg file which is located in the directory where Airflow has been installed.

    For detailed info about MySQL connection string → https://docs.sqlalchemy.org/en/14/dialects/mysql.html#module-sqlalchemy.dialects.mysql.mysqlconnector

```
# open airflow.cfg
# (since we installed as the root user, airflow.cfg is located in /root/airflow/airflow.cfg)
vi /root/airflow/airflow.cfg

-- inside airflow.cfg --
# press i to enter insert mode
# next find executor line and set
executor = CeleryExecutor

# next find sql_alchemy_conn line and set
```

```
sql_alchemy_conn = mysql+pymysql://airflow_user:airflow_pass@localhost/airflow_db

# next find broker_url line and set
broker_url = amqp://admin:admin@localhost/

# next find result_backend line and set
result_backend = db+mysql+pymysql://airflow_user:airflow_pass@localhost/airflow_db
```

13. Now it is time to initialize airflow. Detailed info → https://airflow.apache.org/docs/apache-airflow/stable/start/local.html

    Here are the steps to start Airflow in the correct order:

    i. Make sure MySQL and RabbitMQ servers are running as a service. We can check them and start if they are not already running:

```
# check if MySQL and RabbitMQ servers are running as a service
service --status-all

# if they are not running:
# start MySQL server
service mysql start

# start RabbitMQ server
service rabbitmq-server start
```

    ii. Let's initialize the database for airflow with the configurations we just changed. It will create the necessary tables in mysql under airflow_db database we just created. Run the following command:

```
# initialize db
airflow db init

# check if tables are created under airflow_db database im mysql
mysql

mysql> use airflow_db;
mysql> show tables;
# Below is the output
'
+-------------------------------+
| Tables_in_airflow_db          |
+-------------------------------+
| ab_permission                 |
| ab_permission_view            |
| ab_permission_view_role       |
| ab_register_user              |
| ab_role                       |
| ab_user                       |
| ab_user_role                  |
| ab_view_menu                  |
| alembic_version               |
| connection                    |
| dag                           |
| dag_code                      |
| dag_pickle                    |
| dag_run                       |
| dag_tag                       |
| import_error                  |
| job                           |
| log                           |
| rendered_task_instance_fields |
| sensor_instance               |
| serialized_dag                |
| sla_miss                      |
| slot_pool                     |
| task_fail                     |
| task_instance                 |
| task_reschedule               |
| trigger                       |
| variable                      |
| xcom                          |
+-------------------------------+
29 rows in set (0.00 sec)
'
```

iii. Create an admin user for airflow with the following command. Then we can add other users using the web UI.

```
# create airflow user
airflow users create \
    --username admin \
    --firstname Umut \
    --lastname yildiz \
    --role Admin \
    --email umut.yildiz@ace.games
# it will prompt for a password, create your password and keep it.
```

Note: Starting from the following step iv, we need to start all the processes in different terminals and they should be running at all times to be able to use Airflow. That's why it is best to use screens for each of them. There is a section about creating screens at the end of the article.

iv. Start airflow webserver using the default port 8080. In order to access the web UI, we can go to http://{ip for the host machine}:8080/ address in our web browser and login with the credentials we just created in the previous step iii.

```
# start airflow webserver -> http://{ip for the host machine}:8080/
airflow webserver --port 8080
```

v. Start airflow scheduler. This will allow scheduler to pick jobs and send them to be queued and executed afterwards.

```
# start airflow scheduler -> there is no web UI for this
airflow scheduler
```

vi. Start celery worker. These workers are going to execute the jobs.

```
# start airflow celery worker -> the web UI for this one is the Flower web UI
airflow celery worker

# If we want to see how our workers are doing, we can use Flower web UI:
# Note: we don't need to have this open all the time
airflow celery flower
# http://{ip for the host machine}:5555/ is the default address

# Also, if we want to see the queue, we can use RabbitMQ Management UI, which
# is always available at the address http://{ip for the host machine}:15672/
# It is always available because it is running as a service in the host machine
```

14. We are done with setting up Airflow. We can now use the Airflow webserver to manage our DAGs.

    For ease of use, here is a list of addresses for all the web UIs:

    - Airflow: http://{ip for the host machine}:8080/
    - RabbitMQ Management: http://{ip for the host machine}:15672/
    - Flower: http://{ip for the host machine}:5555/

_____ ********** END ********** _____

Below are some extra useful information.

- How to open screen:

```
# open a screen for airflow webserver
screen -S airflow_webserver

# -- inside screen --
# start airflow webserver
airflow webserver --port 8080
```

```
# detach from the screen and it will stay open
screen -d
```

We can apply the same procedure for airflow scheduler and airflow celery worker as well.

For further details (like killing a screen), refer to https://gist.github.com/jctosta/af918e1618682638aa82

- If we get error when starting the airflow webserver like "`Error: Already running on PID 2446`" we can find that specific process and kill it using the commands below:

```
# find process from the process id (pid)
ps -ef | grep {insert pid here}

# after seeing the process is running, you can kill it by
kill {insert pid here}
```