# Running Airflow Webserver and Scheduler

Add a user group by running the command

```
sudo addgroup airflow
```

Use *usermod* to add new user *airflow* into the *sudo* group.

```
sudo usermod -aG sudo airflow
```

In a typical use case, Airflow needs two components that must be **constantly** running, webserver and scheduler. The former is the Web UI that is used to manage and monitor the workflows, and the latter is responsible for triggering the workflows at pre-defined timestamp correctly.

However, it is not a good idea to run these two components using the provided commands in Linux, such as follows:

```
$ airflow scheduler
```

The drawbacks are obvious:

- If any of the components crashed, the service will then terminated without any notifications.
- The logs of these services are printed in the stdout, which will be lost if crush, so it will be challenging to find out what happened
- When the Linux system restarted, it will not automatically up running.

To solve these problems, we need to run Apache Airflow as Daemon.

#### Step I: Create a Service

First, create a file named airflow-scheduler.service using the below command

```
sudo touch /etc/systemd/system/airflow-scheduler.service
cd /etc/systemd/system/
```

#### Step II: Write Service Configuration

```
sudo nano airflow-scheduler.service
```

Now, open the airflow-scheduler.service file using nano editor and paste the below lines in it

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# “License”); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied. See the License for the
# specific language governing permissions and limitations
# under the License.
[Unit]
Description=Airflow scheduler daemon
After=network.target postgresql.service mysql.service
Wants=postgresql.service mysql.service
[Service]
EnvironmentFile=/etc/default/airflow
#Environment=PATH=/home/biadmin/bin:/usr/local/sbin:/usr/local/bin:/usr
/sbin:/usr/bin:/sbin:/bin
#Environment="PATH=/home/biadmin/bin:/usr/local/sbin:/usr/local/bin:/usr
/sbin:/usr/bin:/sbin:/bin"
#EnvironmentFile=/etc/sysconfig/airflow
User=biadmin
Group=airflow
Type=simple
ExecStart=/usr/local/bin/airflow scheduler
Restart=always
RestartSec=5s
[Install]
WantedBy=multi-user.target
```

**EnvironmentFile :** it specifies the file where service will find its environment variables in ubuntu its /etc/default

Let’s reload the service first by using below command:

```
sudo systemctl daemon-reload
```

As the last step of the configuration of the services, we need to enable these services before we can run them.

```
sudo systemctl enable airflow-scheduler
```

#### Step III: Start Service

```
sudo systemctl start airflow-scheduler.service
```

or

```
sudo service airflow-scheduler start
```

#### Step IV: Check Status

```
sudo systemctl status airflow-scheduler.service
```

or

```
sudo service airflow-scheduler status
```

---

You might want to reboot the machine by running

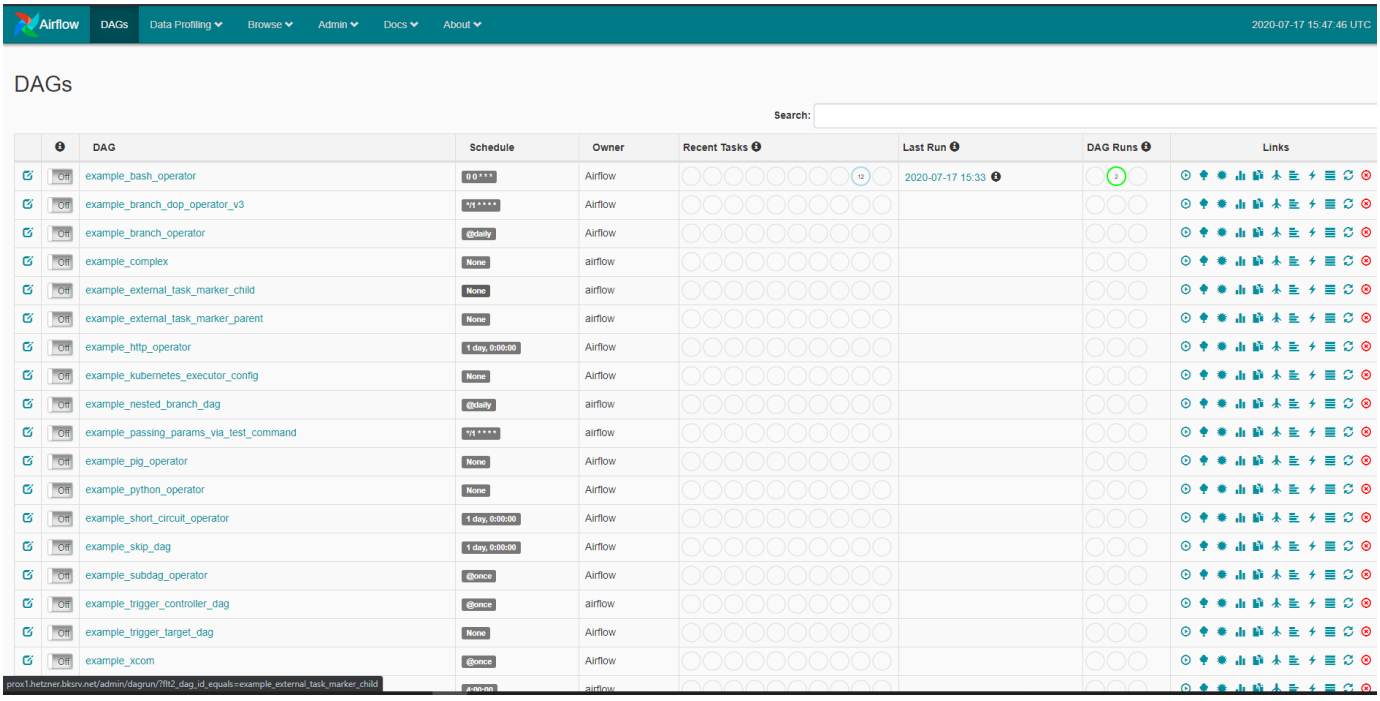
```
sudo reboot
```

After checking the airflow-scheduler.service status again you should see the following:

```
biadmin@bi-airflow:/etc/systemd/system$ sudo service airflow-scheduler status
sudo: setrlimit(RLIMIT_CORE): Operation not permitted
● airflow-scheduler.service - Airflow scheduler daemon
     Loaded: loaded (/etc/systemd/system/airflow-scheduler.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2020-07-17 15:29:35 UTC; 16min ago
   Main PID: 313 (/usr/bin/python)
      Tasks: 71 (limit: 4915)
     Memory: 186.8M
     CGroup: /system.slice/airflow-scheduler.service
             ├─313 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─326 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─334 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─335 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─336 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─337 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─338 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─339 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─340 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─341 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─342 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─343 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─345 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─346 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─347 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─349 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─351 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─354 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─357 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─360 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─371 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─373 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─374 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─376 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─380 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─382 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─385 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─388 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─389 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─391 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─393 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─395 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─397 /usr/bin/python3 /usr/local/bin/airflow scheduler
             ├─398 /usr/bin/python3 /usr/local/bin/airflow scheduler
             └─419 airflow scheduler -- DagFileProcessorManager

Jul 17 15:29:36 bi-airflow airflow[313]:  _/_/  |_/_/  /_/    /_/    /_/  \____/____/|__/
Jul 17 15:29:36 bi-airflow airflow[313]: [2020-07-17 15:29:36,074] {__init__.py:50} INFO - Using executor LocalExecutor
Jul 17 15:29:36 bi-airflow airflow[313]: [2020-07-17 15:29:36,079] {scheduler_job.py:1349} INFO - Starting the scheduler
Jul 17 15:29:36 bi-airflow airflow[313]: [2020-07-17 15:29:36,080] {scheduler_job.py:1357} INFO - Running execute loop for -1 seconds
Jul 17 15:29:36 bi-airflow airflow[313]: [2020-07-17 15:29:36,081] {scheduler_job.py:1358} INFO - Processing each file at most -1 times
Jul 17 15:29:36 bi-airflow airflow[313]: [2020-07-17 15:29:36,081] {scheduler_job.py:1361} INFO - Searching for files in /home/biadmin/airflow/dags
Jul 17 15:29:36 bi-airflow airflow[313]: [2020-07-17 15:29:36,086] {scheduler_job.py:1363} INFO - There are 25 files in /home/biadmin/airflow/dags
Jul 17 15:29:36 bi-airflow airflow[313]: [2020-07-17 15:29:36,144] {scheduler_job.py:1420} INFO - Resetting orphaned tasks for active dag runs
Jul 17 15:29:36 bi-airflow airflow[313]: [2020-07-17 15:29:36,157] {dag_processing.py:562} INFO - Launched DagFileProcessorManager with pid: 419
Jul 17 15:29:36 bi-airflow airflow[419]: [2020-07-17 15:29:36,178] {settings.py:55} INFO - Configured default timezone <Timezone [UTC]>
biadmin@bi-airflow:/etc/systemd/system$
```

To double check the scheduler status launch the webserver:




In order to run apache-airflow as service, we need to create two service entries one for airflow-webserver and another for airflow-scheduler:

Repeat all steps for airflow-webserver

> after making some changes in airflow.cfd don`t forget to reset airflow with `airflow resetdb`

> From your local machine the airflow would be available at http://prox1.hetzner.bksrv.net/admin/
>
> From remote (hetzner) server: http://192.168.100.31:8080/admin/

There is a number of articles providing sort of tutorial/documentation on how to get this done:

1. https://medium.com/@shahbaz.ali03/run-apache-airflow-as-a-service-on-ubuntu-18-04-server-b637c03f4722
2. https://towardsdatascience.com/how-to-run-apache-airflow-as-daemon-using-linux-systemd-63a1d85f9702.
3. https://www.ryanmerlin.com/2019/07/apache-airflow-installation-on-ubuntu-18-04-18-10/.
4. https://janakiev.com/blog/apache-airflow-systemd/
5. https://medium.com/@vando/airflow-inside-a-virtual-enviroment-and-integrated-with-systemd-3b6427bd6430
6. https://stackoverflow.com/questions/39383429/how-to-run-airflow-scheduler-as-a-daemon-process
7. https://github.com/apache/airflow/tree/master/scripts/systemd
8. https://stackoverflow.com/questions/52292591/trying-to-run-apache-airflow-on-ubuntu-server-with-systemd
9. https://medium.com/@taufiq_ibrahim/apache-airflow-installation-on-ubuntu-ddc087482c14
10. https://axdlog.com/2019/running-apache-airflow-with-systemd-via-miniconda-on-gnu-linux/