**Esquisse canevas workshop**

1. mettre a jour notre systeme ubuntu

# *apt update*

2. installer java (jdk et jre)

# *apt install openjdk-8-jdk*

# *sudo apt install default-jdk*

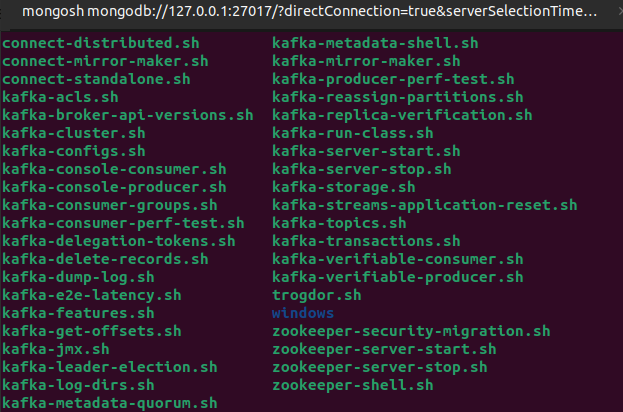
3. installer apache-kafka

# wget <https://downloads.apache.org/kafka/3.5.0/kafka_2.13-3.5.0.tgz>

dezipper et renommer

# tar -xzf [kafka\_2.13-3.5.0.tgz](https://downloads.apache.org/kafka/3.5.0/kafka_2.13-3.5.0.tgz)

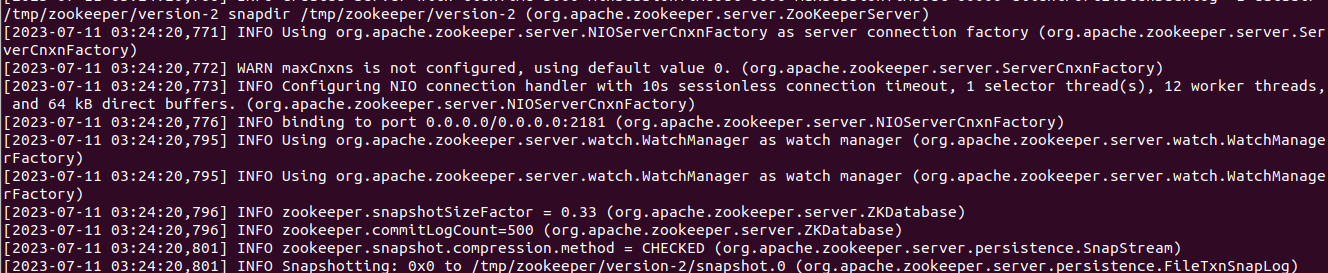
# mv [kafka\_2.13-3.5.0.](https://downloads.apache.org/kafka/3.5.0/kafka_2.13-3.5.0.tgz) kafka

Nous pouvons visualiser les commandes d’administration se trouvant dans kafka/bin

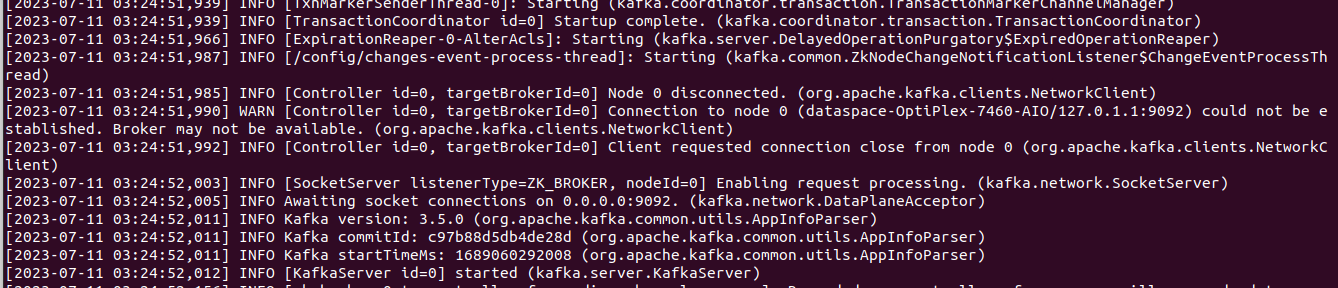
rien qu’en entrant dans la racine du sous-dossier bin

kafka est maintenat installe nous pouvons lancer nos deux serveurs zookeeper et kafka-server

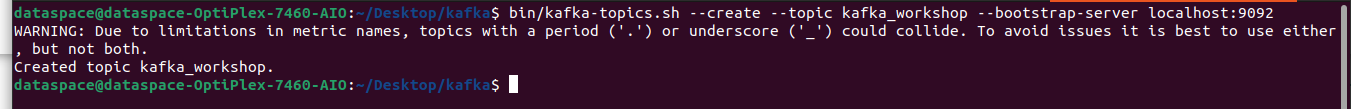
- zookeeper : *bin/zookeeper-server-start.sh config/zookeeper.properties*



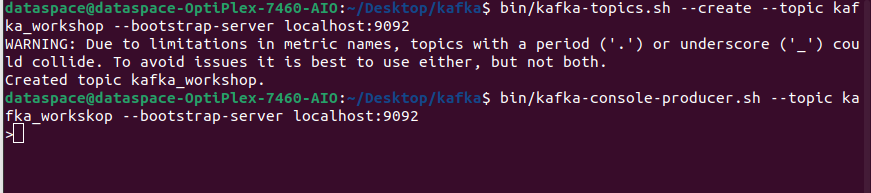
- server-kafka : *bin/kafka-server-start.sh config/server.properties*



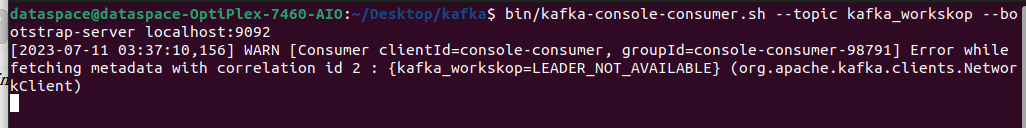
4. Creer le topics *bin/kafka-topics.sh --create --topic kafka\_workskop --bootstrap-server localhost:9092*



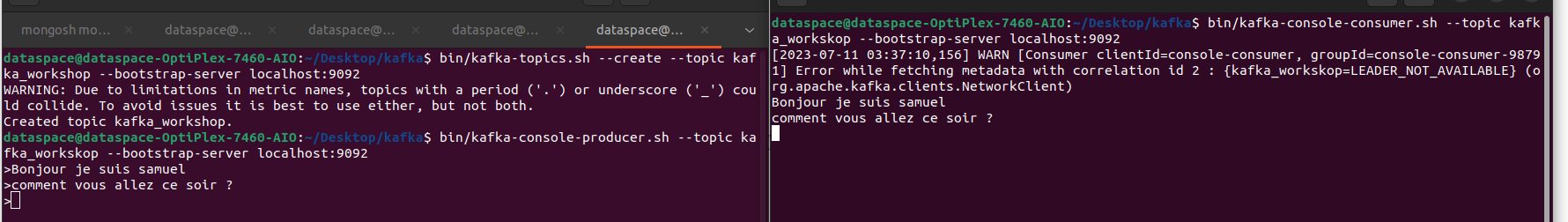
5. Cree un producer *bin/kafka-console-producer.sh --topic kafka\_workskop --bootstrap-server localhost:9092*

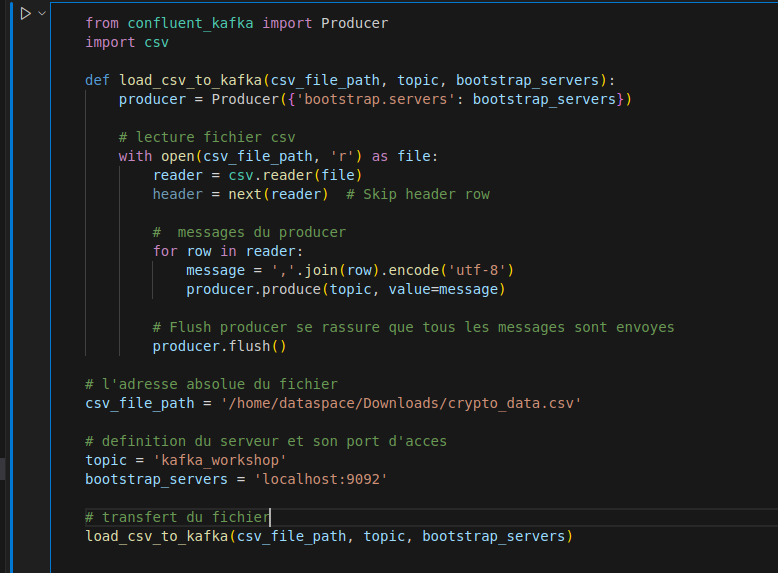


6. Cree un Consumer dans un autre terminal pour voir le transfert en temps reel *bin/kafka-console-consumer.sh --topic kafka\_workskop --bootstrap-server localhost:9092*



7. afficher le streaming des messages

8 . Creons un producer python qui ecrit un fichier csv dans le meme topic



4. installer docker pour ubuntu

- mettre a jour les packages existant

# sudo apt update

- Installation des packages pour permettre l'utilisation de dépôts via HTTPS :

# sudo apt install apt-transport-https ca-certificates curl software-properties-common

- Ajout de la clé GPG officielle de Docker :

# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

- Ajout du dépôt Docker aux sources APT

# echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

- Mise à jour des informations des packages et installation de Docker :

# sudo apt update

# sudo apt install docker-ce docker-ce-cli containerd.io

5. install mongodb

6. creer un environnement virtuel et l’activer

# python3 -m venv workshop

# source bin\activate

7. installer les packages python

- pandas, requests, Confluent\_kafka, apache-airflow-providers-apache-kafka

# *pip install package*

*8- desactiver l’environnement virtuel*

*- # deactivate*

8. installer vs code

9. Ouvrir le dossier de l’environnement virtuel dans vs code

10. Verifier si docker et docker-compose sont installer dans votre os

# docker –version

# docker-compose –version

11. Aller dans le site officiel d’apache-airflow copie le lien ci-dessous et le coller dans votre terminal vs code pour creer le fichier yaml indispensable aux images docker

# curl -LfO '<https://airflow.apache.org/docs/apache-airflow/2.6.2/docker-compose.yaml>'

noter que le fichier yaml genere est modifiable selon les exigences de chaque projet.

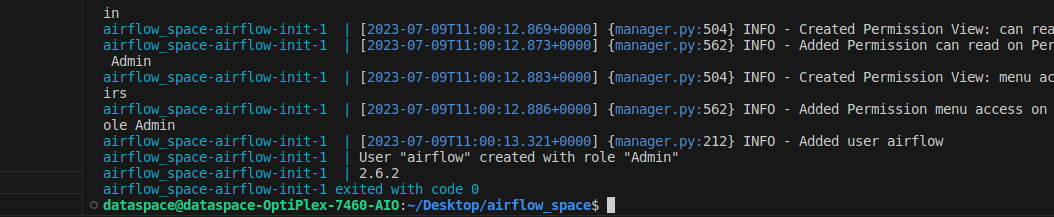
Puis executer la command suivant pour creer les sous-dossiers de configuration

# mkdir -p ./dags ./logs ./plugins ./config

12. initier la base de donnees pour une premier utilisation

# docker compose up airflow-init

cette commande va telecharger toutes les dependance necessaire a airflow et peut prendre un peu plus de temps selon les contraintes exterieures (puissance de la machine, qualite des equipement reseau et le debit de votre internet)

**

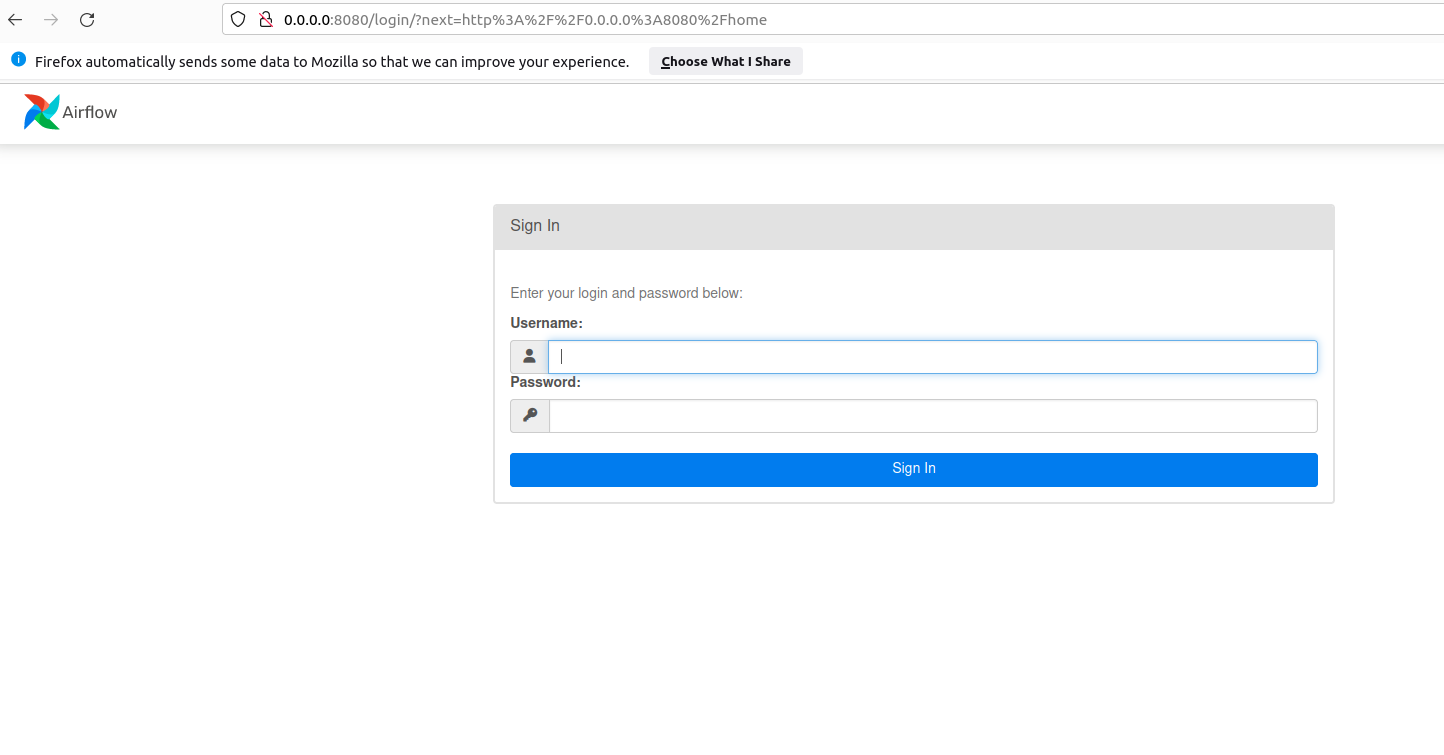
l’image ci-dessus montre que l’operation s’est deroulee avec succes avec la creation de l’utilisateur administrateur ‘airflow’ et l’initiation de la base de donnees.

13. Vous pouvez maintenant demarrer le service dans docker

# docker-compose up -d

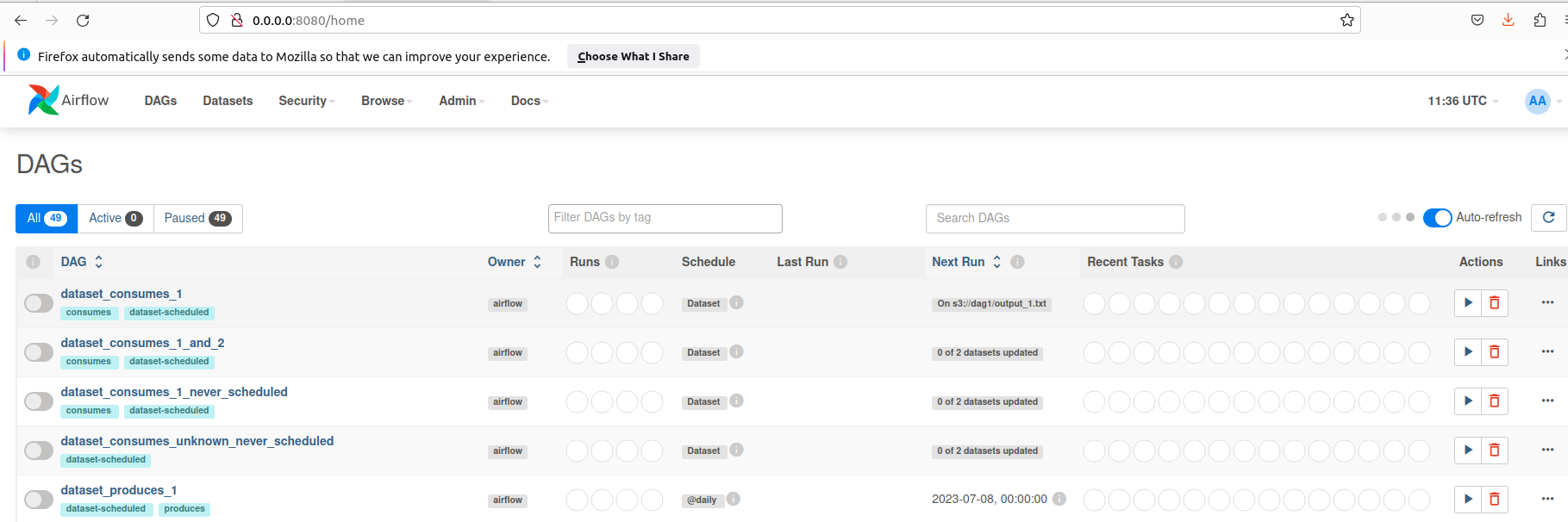
14 . lancer le navigateur et entrer la valeur airflow dans les deux champs

# 0.0.0.0:8080



username : airflow et password:airflow

15. l’ecran d’administration



16. Rentrer dans docker et ouvrir le dossier dag pour creer notre premier DAG

# sudo docker build -t cf3463b1bfc2 .

17. Ouvrir le dossier de travail “airflow\_space” dans vs code et creer un file .py dans le sous-dossier dags.

from airflow import DAG

from datetime import timedelta, datetime

from airflow.providers.http.sensors.http import HttpSensor

import json

from airflow.providers.http.operators.http import SimpleHttpOperator

from airflow.operators.python import PythonOperator

import pandas as pd

from functions import convertir\_kelvin\_fahrenheit, transformer\_donnees

#from airflow.providers.apache.kafka.sensors.kafka import KafkaTopicSensor

""" from airflow.providers.apache.kafka.operators import produce

from airflow.providers.apache.kafka.operators import consume """

default\_args = {

'owner': 'airflow',

'depends\_on\_past': False,

'start\_date': datetime(2023, 7, 10),

'email': ['myemail@domain.com'],

'email\_on\_failure': False,

'email\_on\_retry': False,

'retries': 2,

'retry\_delay': timedelta(minutes=3)

}

with DAG('weathermap\_data',

default\_args=default\_args,

schedule\_interval = '@daily',

catchup=False) as dag:

si\_API\_disponible = HttpSensor(

task\_id ='si\_API\_disponible',

http\_conn\_id='weathermap\_api',

endpoint='/data/2.5/weather?q=Portland&APPID=115c609e4424eeb7cff7b4ae79068d88'

)

extraire\_donnees = SimpleHttpOperator(

task\_id = 'extraire\_donnees',

http\_conn\_id = 'weathermap\_api',

endpoint='/data/2.5/weather?q=Portland&APPID=115c609e4424eeb7cff7b4ae79068d88',

method = 'GET',

response\_filter= lambda r: json.loads(r.text),

log\_response=True

)

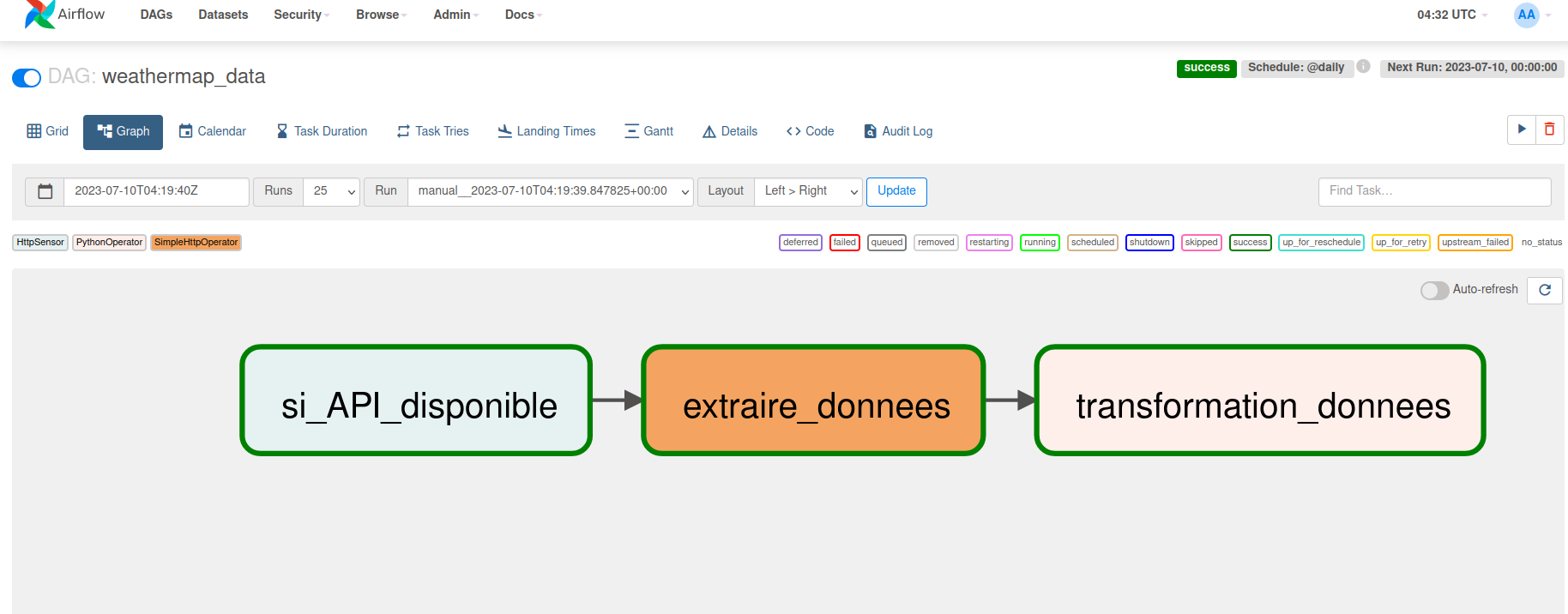
transformation\_donnees = PythonOperator(

task\_id= 'transformation\_donnees',

python\_callable=transformer\_donnees

)

si\_API\_disponible >> extraire\_donnees >> transformation\_donnees

17. Pipeline construite