# **Veri Mühendisliğinde Güçlü Open Source Orkestrasyon Aracı: Apache Airflow**

# **Airflow Nedir?**

Apache Airflow, iş akışlarını sistematik olarak oluşturmanıza, zamanlamanıza ve izlemenize olanak tanıyan bir platformdur. Karmaşık iş akışlarını ve veri hatlarını (data pipelines) kod olarak tanımlamanıza olanak tanır.

# **Airflow’da Kullanılan Temel Terimler**

## **DAG’lar (Directed Acyclic Graphs)**

Airflow’da DAG, bağımlılıklar ve ilişkilerle düzenlenmiş görevlerin bir koleksiyonudur. Her bir görevi (A, B, C ve D gibi) belirli bir senaryoyu temsil eden bir akış şeması olarak düşünebiliriz. DAG, bu görevlerin hangi sırayla çalışması gerektiğini ve hangi görevlerin diğerlerine bağımlı olduğunu tanımlar. Örneğin, Görev A tamamlandıktan sonra Görev B başlayabilir. 💡

## **Görevler (Tasks)**

Görevler, Airflow’daki iş akışlarınızın yapı taşlarıdır. Belirli aksiyonları temsil ederler. Görevler iki türde olabilir:

****Operatörler****: Operatörler, önceden tanımlanmış görev şablonlarıdır. DAG’larınızda hızla görevler oluşturmanıza olanak tanır. Örnek olarak en çok kullanılan iki operatörden bahsedelim:

* ****PythonOperator****: Python scriptlerinde tanımladığımız fonksiyonları çalıştırmak için kullanılır.
* ****BashOperator****: Bash komutlarını çalıştırmak için kullanılır. Operatörler parametrelerle özelleştirilebilir.

****Sensörler****: Sensörler, bir operatör türüdür. Bir şeyin gerçekleşmesini beklerler. Bu, zaman tabanlı olabilir, bir dosya beklemek veya harici bir olay olabilir. Koşul karşılandığında sensörler başarılı olur ve ardından gelen görevlerin çalışmasına izin verir. Örnek olarak en çok kullanılan sensörlerden bahsedelim:

* ****HttpSensor****: Belirli bir URL’nin belirli bir durumu (örneğin, belirli bir yanıt kodu veya içerik) döndürüp döndürmediğini periyodik olarak kontrol eder. İş akışlarınızda dış sistemlerin belirli bir duruma veya olaya ulaşmasını beklemek için mükemmel kullanışlıdır.
* ****SqlSensor****: Belirli bir SQL sorgusunun sonucunu bekleyen bir sensördür. Bu sensör, SQL sorgusunun belirli bir durumu döndürüp döndürmediğini periyodik olarak kontrol eder ve bu koşul karşılandığında başarılı olur.

Buraya kadar anlatmaya çalıştığımız Operatörlerin ve Sensörlerin kullanımını toplu olarak aşağıdaki scriptte inceleyebilirsiniz

from airflow import DAG  
from airflow.operators.python\_operator import PythonOperator  
from airflow.operators.bash\_operator import BashOperator  
from datetime import datetime, timedelta  
  
  
def my\_python\_function():  
 # Özelleştirilmiş python scriptimizi buraya yazıyoruz  
 print("Merhaba, Airflow!")  
  
default\_args = {  
 'owner': 'your\_name',  
 'depends\_on\_past': False,  
 'start\_date': datetime(2024, 8, 1),  
 'emaelay': timedelta(minutes=5),  
}  
#dagımızın tanımını yapıyoruz dag\_id ,dagın çalışma aralığını vs  
dag = DAG(  
 'bash\_operator\_and\_python\_operator\_example',  
 default\_args=default\_args,  
 description='A simple tutorial DAG',  
 schedule\_interval="15 8 \* \* \*", # Her gün 08:15'te   
   
)  
# burada da görevlerin tanımını yapıyoruz   
python\_task = PythonOperator(  
 task\_id='my\_python\_task',  
 python\_callable=my\_python\_function,  
 dag=dag,  
)  
  
bash\_task = BashOperator(  
 task\_id='list\_files',  
 bash\_command='ls -l',  
 dag=dag,  
)  
  
wait\_for\_api\_data = HttpSensor(  
 task\_id='wait\_for\_api\_data',  
 http\_conn\_id='my\_api\_connection', # Airflow bağlantı id'si  
 endpoint='/data', # Kontrol edilecek URL'nin son noktası  
 request\_params={'param\_key': 'param\_value'}, # İsteğe bağlı parametreler  
 response\_check=lambda response: 'expected\_data' in response.text, # Yanıt kontrolü  
 poke\_interval=60, # 60 saniyede bir kontrol eder  
 timeout=600, # 10 dakika sonra zaman aşımına uğrar  
 dag=dag,  
)  
  
"""  
Bu HttpSensor taskında, HttpSensor, belirli bir API'nin /data endpoint'ini kontrol eder ve  
 yanıtın 'expected\_data' içerip içermediğini belirler.   
Bu koşul sağlanana kadar belirli aralıklarla (60 saniyede bir)   
kontrol etmeye devam eder ve 10 dakika içinde beklenen yanıt alınmazsa   
görev başarısız olur.  
  
"""  
  
wait\_for\_postgres\_data = PostgresSensor(  
 task\_id='wait\_for\_data',  
 postgres\_conn\_id='my\_postgres\_connection',  
 sql='SELECT COUNT(1) FROM my\_table WHERE condition = TRUE;',  
 mode='poke',  
 poke\_interval=60, # 60 saniyede bir kontrol eder  
 timeout=600, # 10 dakika sonra zaman aşımına uğrar  
 dag=dag,  
)  
"""  
Bu PostgresSensor taskında , SqlSensor, my\_table tablosunda condition = TRUE  
 olan satırların sayısını kontrol eder.   
Eğer bu sorgu en az bir satır döndürürse sensör başarılı olur,   
aksi takdirde belirli aralıklarla (60 saniyede bir) kontrol etmeye devam eder   
ve 10 dakika içinde beklenen duruma ulaşılmazsa görev başarısız olur.  
  
"""  
  
# aşağıdaki satırda bu taskların bağımlılıklarını belirtiyoruz  
python\_task >> bash\_task >> >> wait\_for\_postgres\_data  
#burada tasklarımız sırasıyla çalışıyor eğer birinde hata alırsa sonrakini çalıştıramıyor  
  
  
  
  
# yine bu taskların bağımlıklıklarını farklı şekilde de belirtebiliriz  
python\_task >> bash\_task  
python\_task >> wait\_for\_postgres\_data  
bash\_task >> wait\_for\_api\_data  
wait\_for\_postgres\_data >> wait\_for\_api\_data  
"""  
python\_task tamamlandıktan sonra bash\_task ve wait\_for\_postgres\_data başlar  
bash\_task ve wait\_for\_postgres\_data tamamlandıktan sonra wait\_for\_api\_data   
başlar.  
"""  
  
# Airflow'da görev bağımlılıklarını bu şekilde tanımlayarak  
# iş akışlarınızı esnek ve yönetilebilir hale getirebilirsiniz

Airflow’u en çok kullanan bazı şirketler:

****Airbnb****: Airflow’un yaratıcıları olan Airbnb, veri iş akışlarını yönetmek için bu aracı yoğun bir şekilde kullanıyor.

****Google****: Google Cloud, Apache Airflow’u Google Cloud Composer’da yönetilen bir hizmet olarak sunar ve Google’ın kendi iç iş akışlarında da Airflow’u kullanmaktadır.

****Spotify****: Spotify, veri iş akışlarını ve ETL süreçlerini yönetmek için Airflow’u kullanıyor.

****Yahoo****: Yahoo, veri işlem ve analiz iş akışlarını düzenlemek için Airflow’u kullanır.

****Uber****: Uber, büyük veri işlemleri ve veri akışlarını yönetmek için Airflow’u kullanır.

****Twitter****: Twitter, veri iş akışlarını ve ETL süreçlerini yönetmek için Airflow’dan yararlanır.

****Slack****: Slack, veri iş akışlarını ve ETL süreçlerini optimize etmek için Airflow’u kullanmaktadır.

****Shopify****: Shopify, büyük veri iş akışlarını ve ETL süreçlerini yönetmek için Airflow’u tercih etmektedir.

****Dropbox****: Dropbox, veri iş akışlarını yönetmek ve ETL süreçlerini düzenlemek için Airflow’u kullanır.

****Reddit****: Reddit, veri iş akışlarını ve veri analiz süreçlerini optimize etmek için Airflow’dan yararlanır.

Bu şirketler, Airflow’un esnekliği, ölçeklenebilirliği ve geniş özellik seti sayesinde veri iş akışlarını daha etkili bir şekilde yönetmektedir.