



Práctica Orquestación

Apache Airflow

Username: airflow
Password: airflow



The screenshot shows the Apache Airflow web interface. At the top, there is a teal header bar with the Airflow logo on the left, the date and time '2020-05-03, 15:19:11 UTC' in the center, and a 'Login' link on the right. Below the header, there is a 'Sign In' modal box. Inside the modal, it says 'Enter your login and password below:'. There are two input fields: 'Username:' and 'Password:'. The 'Username:' field has a small icon of a person to its left. The 'Password:' field has a small icon of a key to its left. Below the input fields is a teal 'Sign In' button.

<http://<direccion ip>:8010>



Apache Airflow - creación pipeline

Veremos cómo crear un pipeline a través de un DAG, creando las tareas:

- Ingest
- Transform
- Load

Apache Airflow - creación pipeline

Ingest (/home/hadoop/scripts/ingest.sh)

##borro todo del directorio landing

```
rm -f /home/hadoop/landing/*
```

##obtengo los files

```
wget -P /home/hadoop/landing https://<url>/yellow_tripdata_2021-01.parquet
```

##borro el contenido del directorio de HDFS /ingest

```
/home/hadoop/hadoop/bin/hdfs dfs -rm -f /ingest/*
```

##copio todo el contenido del directorio /landing en /ingest

```
/home/hadoop/hadoop/bin/hdfs dfs -put /home/hadoop/landing/* /ingest
```

Apache Airflow - creación pipeline

```
GNU nano 4.8 /home/hadoop/scripts/ingest.sh #
#borro todo del directorio landing
rm -f /home/hadoop/landing/*

##obtengo los files
wget -P /home/hadoop/landing https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2021-01.>
##borro el contenido del directorio de HDFS /ingest
/home/hadoop/hadoop/bin/hdfs dfs -rm -f /ingest/*

##copio todo el contenido del directorio /landing en /ingest
/home/hadoop/hadoop/bin/hdfs dfs -put /home/hadoop/landing/* /ingest
```

Apache Airflow - creación pipeline

Transform & Load (/home/hadoop/scripts/transformation.py)

```
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import HiveContext
sc = SparkContext('local')
spark = SparkSession(sc)
hc = HiveContext(sc)
```

##leo el file desde HDFS y lo cargo en un dataframe

```
df = spark.read.option("header",
"true").parquet("hdfs://<direccionIP>:9000/ingest/<file>
```

Apache Airflow - creación pipeline

Transform & Load

##creamos una vista del DF

```
df.createOrReplaceTempView("tripdata_vista")
```

##Filtramos el DF quedándonos solamente con aquellos viajes que viajaron un solo pasajero y tuvieron una distancia mayor a 5 millas

```
new_df = spark.sql("select * from tripdata_vista where passenger_count = 1  
and trip_distance > 5")
```

##Creamos una vista con la data filtrada###

```
new_df.createOrReplaceTempView("tripdata_vista_filtrada")
```

##insertamos el DF filtrado en la tabla tripdata_table

```
spark.sql("insert into tripdata.tripdata_table select * from  
tripdata_vista_filtrada;")
```

Apache Airflow - creación pipeline

```
GNU nano 4.8 /home/hadoop/scripts/transformation.py
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql import HiveContext
sc = SparkContext('local')
spark = SparkSession(sc)
hc = HiveContext(sc)

##leo el csv desde HDFS y lo cargo en un dataframe
df = spark.read.option("header", "true").parquet("hdfs://172.20.0.2:9000/ingest/yellow_tripdata_2021-0>

##creamos una vista del DF
df.createOrReplaceTempView("tripdata_vista")

##Filtramos el DF quedandonos solamente con aquellos viejes que viajaron un solo pasajero y tuvieron u>
new_df = spark.sql("select * from tripdata_vista where passenger_count = 1 and trip_distance > 5")

##Creamos una vista con la data filtrada###
new_df.createOrReplaceTempView("tripdata_vista_filtrada")

##insertamos el DF filtrado en la tabla tripdata_table
spark.sql("insert into tripdata.tripdata_table select * from tripdata_vista_filtrada;")
```


Apache Airflow - creación pipeline

Creación DAG

```
from datetime import timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.dummy import DummyOperator
from airflow.utils.dates import days_ago
```

```
args = {
    'owner': 'airflow',
}
```

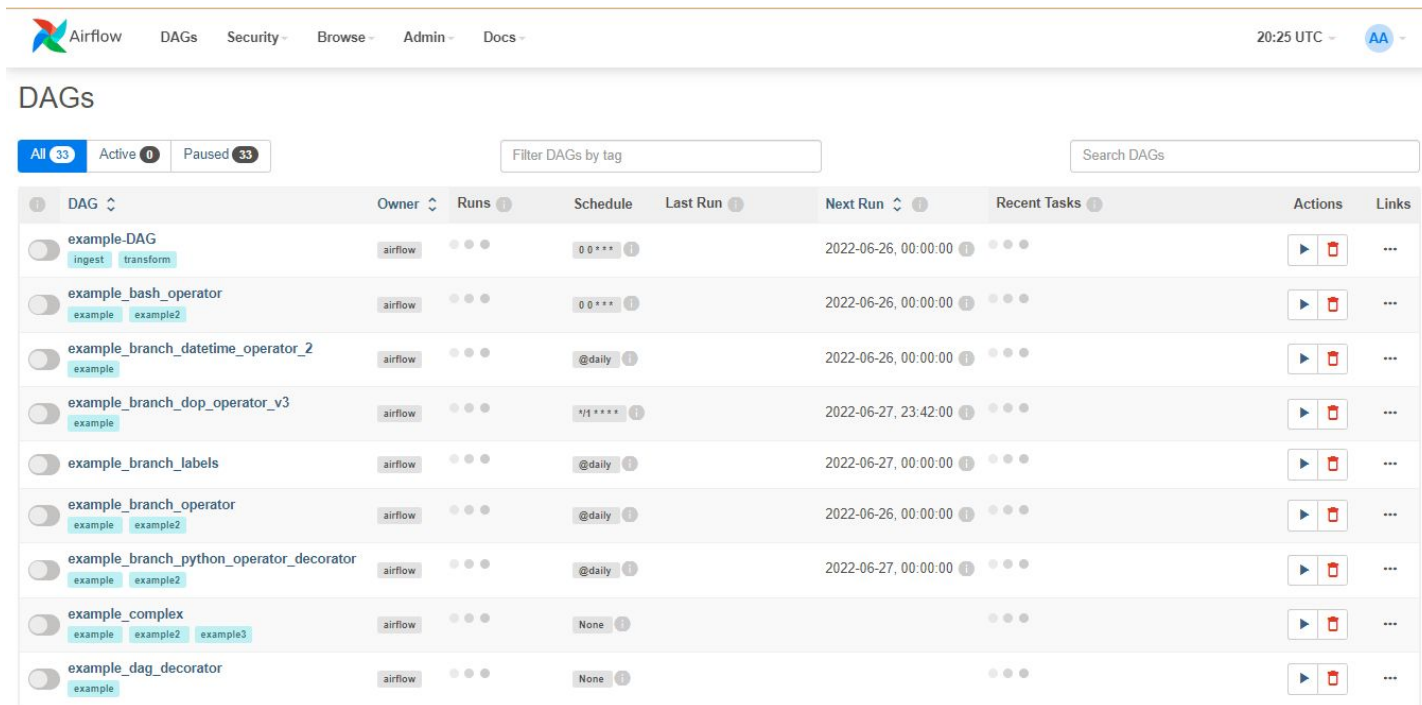
```
with DAG(
    dag_id='ingest-transform',
    default_args=args,
    schedule_interval='0 0 * * *',
    start_date=days_ago(2),
    dagrun_timeout=timedelta(minutes=60),
    tags=['ingest', 'transform'],
    params={"example_key": "example_value"},
) as dag:
```

Apache Airflow - creación pipeline

Creación DAG

```
finaliza_proceso = DummyOperator(  
    task_id='finaliza_proceso',  
)  
  
ingest = BashOperator(  
    task_id='ingest',  
    bash_command='/usr/bin/sh /home/hadoop/scripts/ingest.sh ',  
)  
  
transform = BashOperator(  
    task_id='transform',  
    bash_command='ssh hadoop@172.20.0.2 /home/hadoop/spark/bin/spark-submit --files  
/home/hadoop/hive/conf/hive-site.xml /home/hadoop/scripts/transformation.py ',  
)  
  
ingest >> transform >> finaliza_proceso  
  
if __name__ == "__main__":  
    dag.cli()
```

Apache Airflow



The screenshot displays the Apache Airflow web interface. At the top, there is a navigation bar with the Airflow logo and links for DAGs, Security, Browse, Admin, and Docs. The current time is 20:25 UTC, and the user is logged in as 'AA'. Below the navigation bar, the 'DAGs' section is active, showing a list of DAGs. The interface includes filters for 'All' (33), 'Active' (0), and 'Paused' (33) DAGs, along with a search bar and a 'Filter DAGs by tag' input field. The DAGs are listed in a table with columns for DAG name, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, Actions, and Links. Each DAG entry includes a toggle switch to enable or disable it, a list of tags, and a status indicator (e.g., 'airflow').

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
<input type="checkbox"/> example-DAG ingest transform	airflow	...	0 0 * * *		2022-06-26, 00:00:00	...	▶ 🗑️	...
<input type="checkbox"/> example_bash_operator example example2	airflow	...	0 0 * * *		2022-06-26, 00:00:00	...	▶ 🗑️	...
<input type="checkbox"/> example_branch_datetime_operator_2 example	airflow	...	@daily		2022-06-26, 00:00:00	...	▶ 🗑️	...
<input type="checkbox"/> example_branch_dop_operator_v3 example	airflow	...	* * * * *		2022-06-27, 23:42:00	...	▶ 🗑️	...
<input type="checkbox"/> example_branch_labels	airflow	...	@daily		2022-06-27, 00:00:00	...	▶ 🗑️	...
<input type="checkbox"/> example_branch_operator example example2	airflow	...	@daily		2022-06-26, 00:00:00	...	▶ 🗑️	...
<input type="checkbox"/> example_branch_python_operator_decorator example example2	airflow	...	@daily		2022-06-27, 00:00:00	...	▶ 🗑️	...
<input type="checkbox"/> example_complex example example2 example3	airflow	...	None			...	▶ 🗑️	...
<input type="checkbox"/> example_dag_decorator example	airflow	...	None			...	▶ 🗑️	...