Almacenes y Minería de Datos

Prácticas

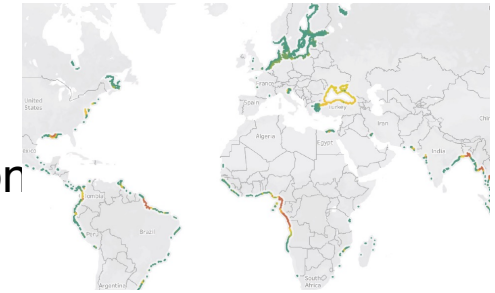
Talend- ETL incremental

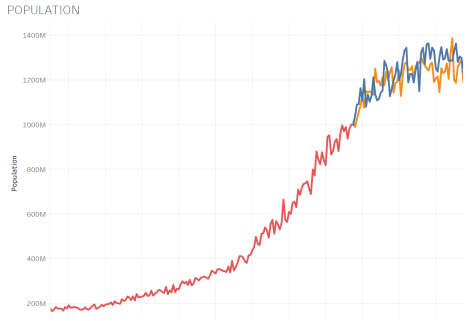
Apache Airflow

Joaquín Ángel Triñanes Fernández
Instituto de Investigacións Tecnolóxicas
Joaquin.Trinanes@usc.es Ext: 16001 Externo: 881816001

Apache Airflow

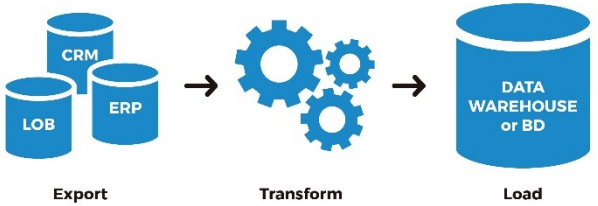
- Prácticas: Estado y dudas
- Data pipeline: Aplicación que se sitúa entre los datos crudos (fuentes de datos) y el destino de datos. Por qué las necesitamos?
  - Preparar los datos para visualizarlos
  - Migrar a otra BBDD
  - Convertir formatos
  - Para integrar datos en aplicaciones
  - Para trabajos en tiempo real
  - Proporcionar un canal de datos a multiples aplicacion





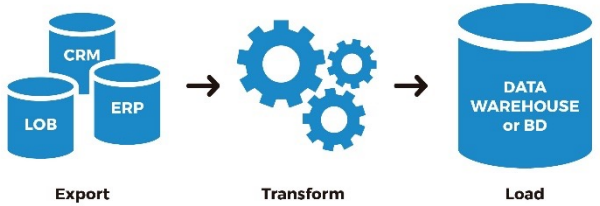
- Pipelines&Workflows:
  - Pueden operar en tiempo real, a horas específicas (batch), pueden estar alojadas en la nube, las hay de código abierto, operar en datos estructurados/no estructurados, …
  - Beneficios: procesar grandes volúmenes de datos, embeberlas en aplicaciones, capturer metadatos, components reutilizables, aplicar lógica del negocio,…
  - Procesos: unir datos de diferentes fuentes, extraer y estandarizar, corregir errores/datos perdidos, carga y análisis, automatización,…
- Necesitamos crear, ejecutar, monitorizar y calendarizar los procesos.

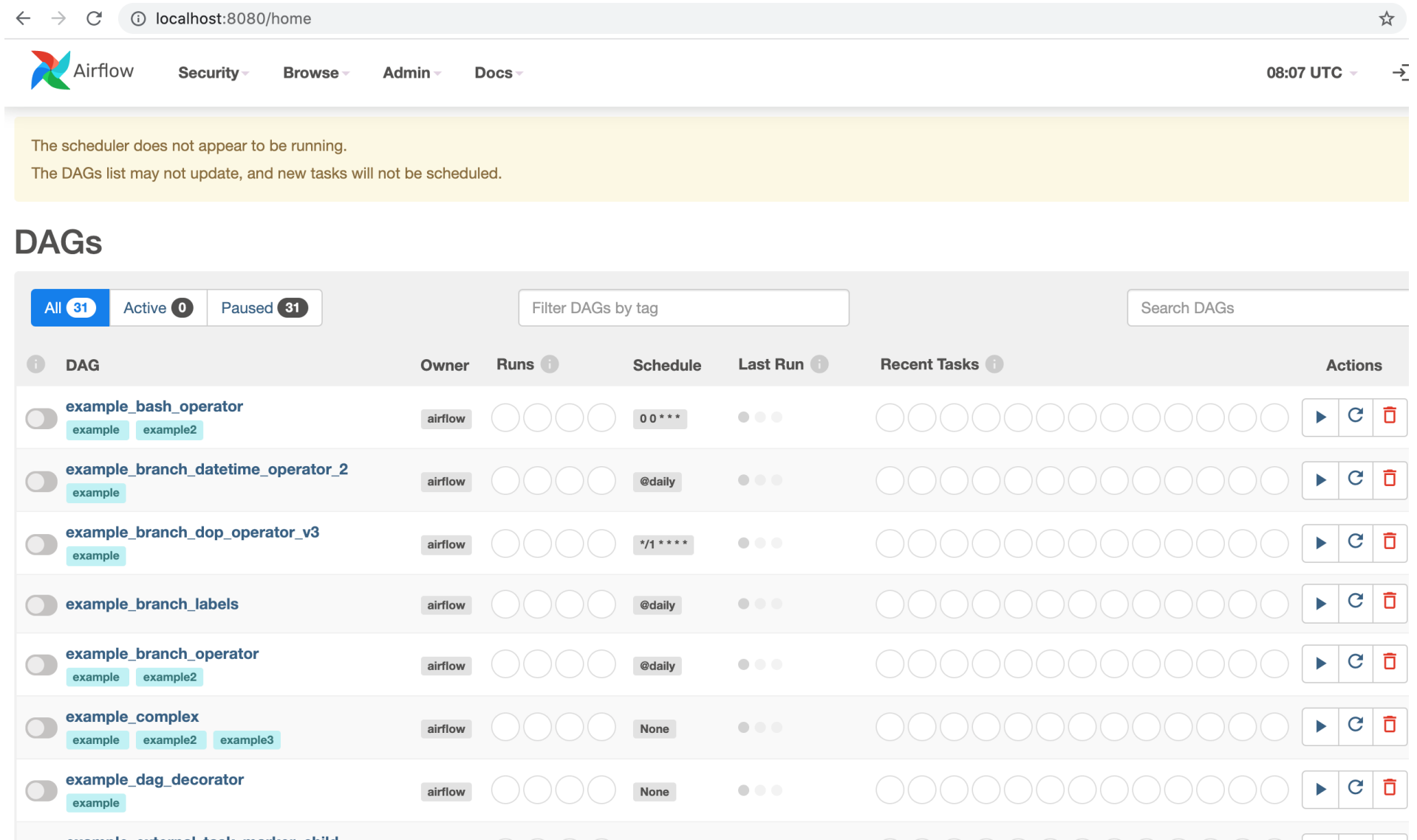
Apache Airflow



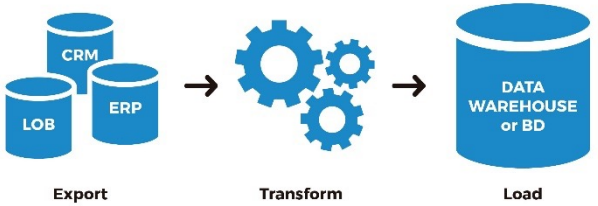
- ETL: Por qué la necesitamos?
  - Múltiples fuentes
  - No queremos modificar las aplicaciones. Separamos el proceso de transformer los datos de las aplicaciones que los usan.
- Tenemos datos de referencia, los extraemos, validamos, transformamos, movemos a la staging area, y la publicamos en el DW
- Apache Airflow (multiplataforma-python&pip. Problemas en Windows)
  - Instalar (linux):
    - export AIRFLOW_HOME=~/airflow
    - python3 –m venv env-airflow
    - source env-airflow/bin/activate
    - pip install apache-airflow (si error, export AIRFLOW_GPL_UNIDECODE=yes ; pip install otra vez o usar constraints Aquí)
    - airflow db init (inicializa la BBDD para airflow)
    - airflow webserver –p 8080 (ejecuta el servidor airflow) Navegador: localhost:8080
  - En Windows (docker o a través de wsl)
  - Para verificar que está instalado: airflow version

Apache Airflow





Apache Airflow



- Vamos a crear un hola mundo
  - cd airflow/dags
  - Hello_world.py

  Importamos paquetes

  Definimos función imprimir Hola

  Declaramos el DAG

  Definimos operador

  Definimos orden operadores (uno solo en existe caso)

  línea commandos:

  airflow list_dags

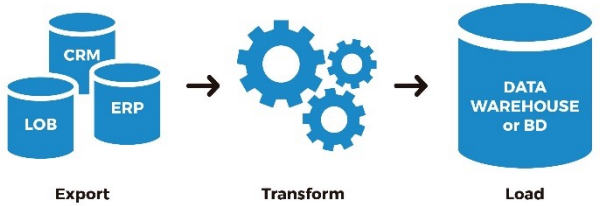
  airflow list_tags hello –tree

  airflow test …

  airflow run …

```
from datetime import datetime
from airflow import DAG
from airflow.operators.dummy_operator import
DummyOperator
from airflow.operators.python_operator import
PythonOperator

def print_hello():
   return 'Hello world from first Airflow DAG!'

dag = DAG('hello_world', description='Hello World DAG',
        schedule_interval='0 12 * * *',
        start_date=datetime(2022, 3, 20), retry_delay':
dt.timedelta(minutes=5),catchup=False)

hello_operator = PythonOperator(task_id='hello_task',
python_callable=print_hello, dag=dag)

hello_operator
```

Apache Airflow



- Ejercicio 1: Todos los días a la 1am recibimos un fichero csv con el número de casos COVID por país. A las 2am recibimos otro fichero CSV que especifica el número de fallecidos por país. Queremos, usando Apache Airflow, cargar estos ficheros, una vez merclados, en una base de datos Postgresql.
- Ejercicio 2: Repítelo con Talend.
- Ejercicio 3: En ambos casos, comentad el escenario que supone la copia incremental de datos y los retos que ésto supone.
- Ejercicio 4: Podríamos ejecutar el ejecutable de Talend desde Airflow?

Esta práctica se llevará a cabo en grupos de 2 o 3 componentes.