



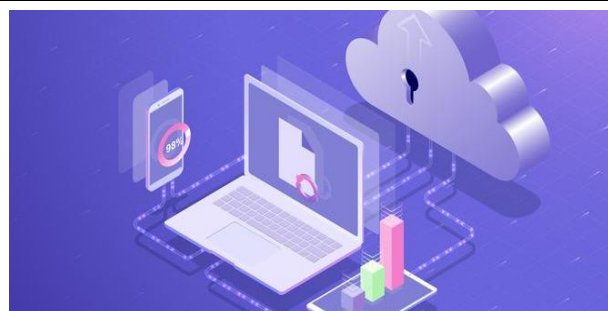
Universidad de Guadalajara

CUCEI – Centro Universitario de Ciencias Exactas e Ingenierías

Mtro. Michel Emanuel López Franco

Computación tolerante a fallas

Reporte 10



**Velasco Hernandez Victor
Manuel**

Código: 216598879

| D06 | 2022A |

| 14/Marzo/2022 |

Airflow





Universidad de Guadalajara
CUCEI
Computación tolerante a fallas

Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|14/ Marzo /2022 | |D06|



Reporte 10- Airflow

Objetivo:

Contestar preguntas y generar ejemplo

¿Qué es Apache Airflow?

es una plataforma de gestión, monitorización, planificación de flujos de trabajo de código abierto ampliamente utilizada en ingeniería de datos, usada como orquestador de servicios. [1]

Introducción:

El manejo de flujos de trabajo en un sistema de computo resultan ser una herramienta muy útil al momento de querer ejecutar una secuencia de procesos de manera programada, este tipo de herramientas hace posible que se corran procesos en segundo plano de una manera correcta, esto le brinda a los programadores potencialidad al momento de querer planificar procesos repetitivos, todo esto nos permite el estudio de estrategias para su implementación, lo cuál puede ayudar a mejorar la funcionalidad de los programas.

Desarrollo:

En esta actividad se va a crear un flujo de trabajo para realizar un proceso ETL a la API de Spotify, esto será posible gracias a un flujo de trabajo que será configurado y un script(código) que ejecutará.

Para comenzar, se desarrolla el código para el flujo de trabajo, en este caso, se le agregara al código del tutorial que a continuación se incluirá, solo para demostrar que se puede realizar muchas veces 1 o más tareas según lo agendado.

[ETL en API de Spotiy](#)



Universidad de Guadalajara
CUCEI
Computación tolerante a fallas

Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|14/ Marzo /2022 | |D06|

```
home > kali > Documents > Airflow > dags > spotify_dag.py
1 from datetime import timedelta
2 from datetime import datetime
3 from airflow import DAG
4 from airflow.operators.python_operator import PythonOperator
5 from airflow.utils.dates import days_ago
6
7 from spotify_etl import run_spotify_etl
8
9 default_args = {
10     'owner': 'kali',
11     'depends_on_past': False,
12     'start_date': datetime(2022, 4, 24),
13     'email': ['victor.velasco01@gmail.com'],
14     'email_on_failure': False,
15     'email_on_retry': False,
16     'retries': 1,
17     'retry_delay': timedelta(minutes=1)
18 }
19
20 dag = DAG(
21     'spotify_dag',
22     default_args=default_args,
23     description='Our first DAG with ETL process!',
24     schedule_interval=timedelta(days=1),
25 )
26
```

Figura 1: Código de Flujo de Trabajo

```
home > kali > Documents > Airflow > dags > spotify_etl.py
1 import sqlalchemy
2 import pandas as pd
3 from sqlalchemy.orm import sessionmaker
4 import requests
5 import json
6 from datetime import datetime
7 import datetime
8 import sqlite3
9
10
11 # Generate your token here: https://developer.spotify.com/console/get-recently-played/
12 # Note: You need a Spotify account (can be easily created for free)
13
14 def check_if_valid_data(df: pd.DataFrame) -> bool:
15     # Check if dataframe is empty
16     if df.empty:
17         print("No songs downloaded. Finishing execution")
18         return False
19
20     # Primary Key Check
21     if pd.Series(df['played_at']).is_unique:
22         pass
23     else:
24         raise Exception("Primary Key check is violated")
25
26 # Check for nulls
```

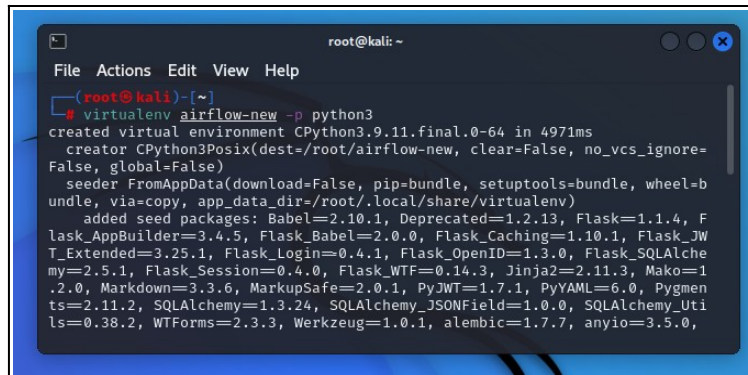
Figura 2: Script de proceso ETL en API de Spotify

Tras haber codificado la configuración, es pertinente configurar Airflow para su funcionamiento.



Primeramente, en nuestra terminal de Linux, introducimos el comando, para crear en ambiente virtual(en caso de que el programa requiera librerías con versiones distintas, se adaptaran en ese ambiente sin afectar globalmente el sistema:

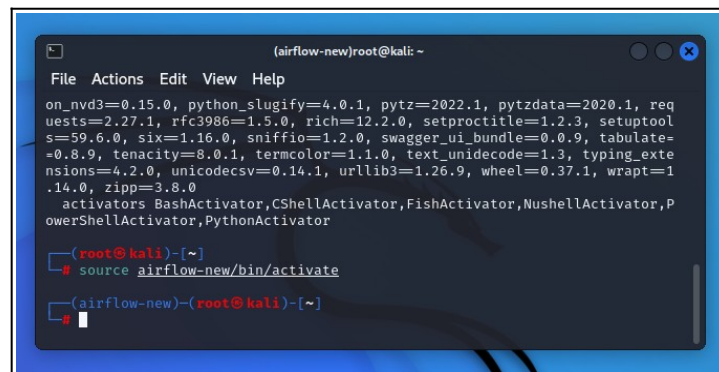
`virtualenv airflow-new -p python3`



```
root@kali: ~  
File Actions Edit View Help  
(root@kali)-[~]  
# virtualenv airflow-new -p python3  
created virtual environment CPython3.9.11.final.0-64 in 4971ms  
creator CPython3Posix(dest=/root/airflow-new, clear=False, no_vcs_ignore=False, global=False)  
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/root/.local/share/virtualenv)  
added seed packages: Babel=2.10.1, Deprecated=1.2.13, Flask=1.1.4, Flask_AppBuilder=3.4.5, Flask_Babel=2.0.0, Flask_Caching=1.10.1, Flask_JWT_Extended=3.25.1, Flask_Login=0.4.1, Flask_OpenID=1.3.0, Flask_SQLAlchemy=2.5.1, Flask_Session=0.4.0, Flask_WTF=0.14.3, Jinja2=2.11.3, Mako=1.2.0, Markdown=3.3.6, MarkupSafe=2.0.1, PyJWT=1.7.1, PyYAML=6.0, Pygments=2.11.2, SQLAlchemy=1.3.24, SQLAlchemy_JSONField=1.0.0, SQLAlchemy_Utils=0.38.2, WTForms=2.3.3, Werkzeug=1.0.1, alembic=1.7.7, anyio=3.5.0,
```

Y después usamos el comando, para entrar el ambiente virtual:

`source airflow-new/bin/activate`



```
(airflow-new)root@kali: ~  
File Actions Edit View Help  
on_nvd3=0.15.0, python_slugify=4.0.1, pytz=2022.1, pytzdata=2020.1, requests=2.27.1, rfc3986=1.5.0, rich=12.2.0, setproctitle=1.2.3, setuptools=59.6.0, six=1.16.0, sniffio=1.2.0, swagger_ui_bundle=0.0.9, tabulate=0.8.9, tenacity=8.0.1, termcolor=1.1.0, text_unidecode=1.3, typing_extensions=4.2.0, unicodcsv=0.14.1, urllib3=1.26.9, wheel=0.37.1, wrapt=1.14.0, zipp=3.8.0  
activators BashActivator,CSHELLActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator  
(root@kali)-[~]  
# source airflow-new/bin/activate  
(airflow-new)-(root@kali)-[~]  
#
```

Y después procedemos a instalar el programa de Apache Airflow en el ambiente virtual:

`pip install apache-airflow`



Universidad de Guadalajara
CUCEI
Computación tolerante a fallas

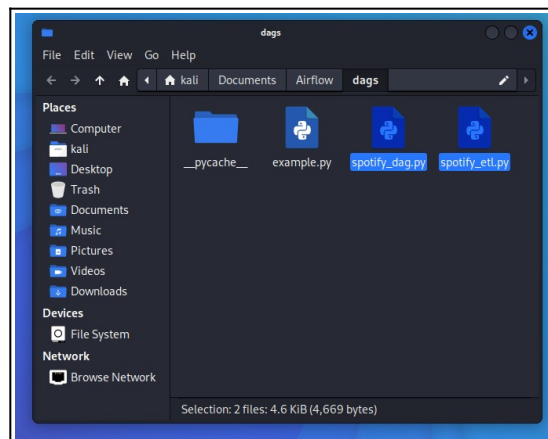
Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|14/ Marzo /2022 | |D06|

```
(airflow-new)root@kali: ~  
File Actions Edit View Help  
(airflow-new)-(root@kali)-[~]  
└─$ pip install apache-airflow
```

Después de que el programa haya sido instalado, procedemos a configurar un usuario:

```
(airflow-new)root@kali: ~  
File Actions Edit View Help  
ls  
airflow airflow-new env_airflow  
(airflow-new)-(root@kali)-[~]  
└─$ airflow users create --username kali --password kali --firstname Victor  
--lastname Velasco --role Admin --email victor.velasco01@gmail.com
```

Antes de iniciar nuestro programa, es importante que nuestro flujo de trabajo (DAG), se encuentre guardado en la carpeta asignada para estos:





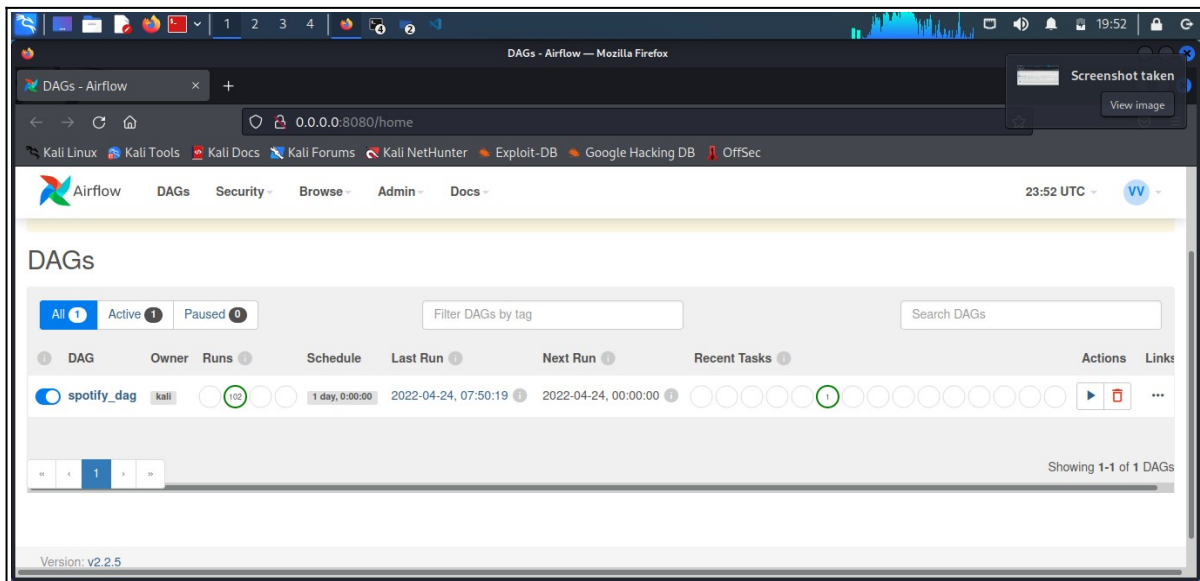
Universidad de Guadalajara
CUCEI
Computación tolerante a fallas

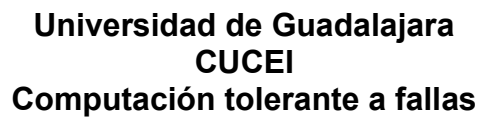
Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|14/ Marzo /2022 | |D06|

Tras haber sido guardado con una terminal usamos el comando “[airflow webserver](#)” para desplegar nuestra pagina de flujo, y en otra se usara “[airflow scheduler](#)” para inicializar los flujos de trabajo guardados.

```
(airflow-new)root@kali: ~  
File Actions Edit View Help  
~(airflow-new)-(root@kali)-[~]  
# airflow webserver  
[2022-04-24 19:50:30,556] {dagbag.py:500} INFO - Filling  
/dev/null  
[2022-04-24 19:50:30,977] {manager.py:512} WARNING - Refi  
ssion view, assoc with role exists DAG Runs.can_create Ar  
Running the Gunicorn Server with:  
Workers: 4 sync  
Host: 0.0.0.0:8080  
  
(airflow-new)root@kali: ~  
File Actions Edit View Help  
~(airflow-new)-(root@kali)-[~]  
# airflow scheduler  
[2022-04-24 19:50:46 -0400] [52020] [INFO] Starting gunicorn 20.1.0  
[2022-04-24 19:50:46 -0400] [52020] [INFO] Listening at: http://0.0.0.0:8793  
(52020)  
[2022-04-24 19:50:46 -0400] [52020] [INFO] Using worker: sync  
[2022-04-24 19:50:46,803] {scheduler_job.py:694} INFO - Starting the schedule  
r  
[2022-04-24 19:50:46 -0400] [52021] [INFO] Booting worker with pid: 52021
```

Después de ejecutarlo, procedemos a ver la pagina desplegada, en la cuál nos aparecerá nuestro DAG(flujo de trabajo):





Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|14/ Marzo /2022 | |D06|

El uso de flujos de trabajo para hacer tareas repetitivas tiene un gran potencial como lo es la realización de los procesos ETL, realizar todo esto en segundo plano



Universidad de Guadalajara
CUCEI
Computación tolerante a fallas

Victor Manuel Velasco Hernández
|Código: 216598879 | | 2022A |
|14/ Marzo /2022 | |D06|

resulta ser muy útil para realizar procesos sin preocuparnos por estarlos ejecutando cada vez que queramos, al igual que con los Workflows, con Apache Airflow se vuelve a recordar la importancia de los flujos de trabajo para realizar sistemas tolerantes a fallos, trabajar con este tipo de tecnologías resulta ser fundamental para conocer la naturaleza de este tipo de herramienta para desarrollar cada vez, mejores flujos de trabajo que garanticen la calendarización y ejecución de los procesos, lo cuál ayuda a tener una mejor comprensión de la lógica de Apache Airflow, esto hace que pensemos en ideas de aprovechamiento para el futuro.

Link de repositorio:

<https://github.com/Victor012396/ComputacionTolerante.git>

Bibliografía:

- [1] "Apache Airflow : ¿qué es y cómo se usa?"
<https://datascientest.com/es/todo-sobre-apache-airflow?msclkid=4315749dc42211eca69e9356b47bfc87> (consultado el 24 de abril de 2022).
- [2] INSAID, "Setting up Apache-Airflow in Ubuntu", Medium, el 20 de octubre de 2021. <https://insaid.medium.com/setting-up-apache-airflow-in-ubuntu-d0317a59f8a4> (consultado el 24 de abril de 2022).
- [3] Karolina Sowinska, Flujo de aire para principiantes: ¡ejecute Spotify ETL Job en 15 minutos!, (el 12 de noviembre de 2020). Consultado: el 24 de abril de 2022. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=i25ttd32-eo>