

# Máster Universitario en Ingeniería Informática

## CLOUD COMPUTING

### PRÁCTICA 2: FLUJOS DE TRABAJO EN CLOUD COMPUTING CON AIRFLOW



**UNIVERSIDAD  
DE GRANADA**

Carlos Morales Aguilera  
75925767-F  
carlos7ma@correo.ugr.es

Curso Académico 2020-2021

# Índice

<b>1. Introducción</b>	<b>2</b>
1.1. Objetivo . . . . .	2
1.2. AirFlow . . . . .	2
1.3. Despliegue . . . . .	2
<b>2. Resolución de tareas</b>	<b>4</b>
2.1. Tarea 1: Creación de carpeta . . . . .	4
2.2. Tareas 2 y 3: Descargar Temperatura y Humedad . . . . .	4
2.3. Tarea 4: Descomprimir ficheros CSV . . . . .	4
2.4. Tarea 5: Preprocesamiento de datos . . . . .	4
2.5. Tarea 6: Almacenar la información en MongoDB Atlas . . . . .	4
2.6. Tarea 7: Descargar servicios de GitHub . . . . .	5
2.6.1. Servicios . . . . .	5
2.7. Tarea 8: Testear los servicios . . . . .	5
2.8. Tarea 9: Copiar fuentes en máquina a desplegar . . . . .	6
2.9. Tarea 10: Crear el contenedor en la máquina a desplegar . . . . .	6
2.10. Tarea 11: Desplegar el contenedor . . . . .	6
2.11. Flujo de tareas . . . . .	7
<b>3. Conclusiones</b>	<b>9</b>

# 1. Introducción

## 1.1. Objetivo

En esta práctica se pretende resolver la necesidad de un modelado de flujo de trabajo completo para un servicio que sigue la filosofía Cloud Native. Para ello se van a realizar y desplegar una serie de microservicios escalables bajo esta filosofía.

El objetivo de la práctica es realizar una entrega continua de software utilizando la herramienta **AirFlow** para dar servicio sobre un pronóstico del tiempo utilizando un modelo de aprendizaje mediante análisis de datos.

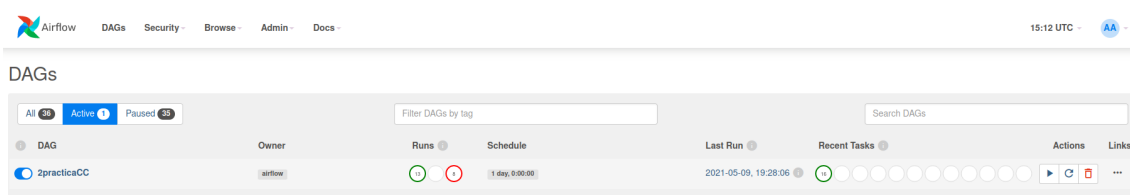
Por último cabe destacar que la práctica se encuentra alojada haciendo uso de un sistema de gestión de versiones como es **Git** y en la plataforma **GitHub**.

## 1.2. AirFlow

La práctica se encuentra alojada en el repositorio dedicado a la misma llamado Carlosma7/cc2\_airflow\_practica. Para ello he seguido la siguiente estructura:

- **dags**: Carpeta donde alojo el grafo asociado a AirFlow, con el nombre de **practica**.
- **Docker-compose.yml**: Composición utilizada, que proviene de la página oficial de AirFlow, por lo que conservo la documentación inicial.
- **requirements.txt**: Requisitos de instalación para AirFlow.

Para poder desplegar **AirFlow** por lo tanto he realizado la composición con **docker-compose up -d**, y una vez levantado, accedo a AirFlow:



Cabe destacar que he instalado la herramienta de **Git** y **Unzip** para la realización de la práctica.

## 1.3. Despliegue

Para la realización del despliegue, he utilizado un **droplet** propio de **DigitalOcean**, abriendo los puertos **2020** y **2021** siendo estos los utilizados por las versiones 1 y 2 de los servicios, respectivamente.

# Firewalls

 MyFirewall

## Inbound

Type	Protocol	Port Range	Sources	
SSH	TCP	22	All IPv4	All IPv6
Custom	TCP	2020	All IPv4	All IPv6
Custom	TCP	2021	All IPv4	All IPv6

## 2. Resolución de tareas

Antes de hablar de los diferentes servicios, cabe destacar que se encuentran en diferentes repositorios (para facilitar su descarga mediante **git**) siendo estos `Carlosma7/cc2_airflow_arima_v1` y `Carlosma7/cc2_airflow_arima_v2` para las versiones, respectivamente.

A continuación se enumeran y explican las tareas realizadas en la práctica:

### 2.1. Tarea 1: Creación de carpeta

En esta tarea sencillamente se crea un **BashOperator** donde crea una carpeta temporal en el sistema. Se encuentra en esta línea.

### 2.2. Tareas 2 y 3: Descargar Temperatura y Humedad

En estas tareas se descargará la información de los ficheros CSV con las temperaturas y humedades de los datos propuestos para la práctica, los cuales son descargados con la herramienta **curl** mediante un **BashOperator**. Se encuentran en esta línea la temperatura y en esta humedad.

### 2.3. Tarea 4: Descomprimir ficheros CSV

En esta tarea se utiliza la herramienta **unzip** para descomprimir ambos ficheros CSV descargados previamente, por lo tanto se utiliza un **BashOperator** para dicha tarea de forma sencilla. Se encuentra en esta línea.

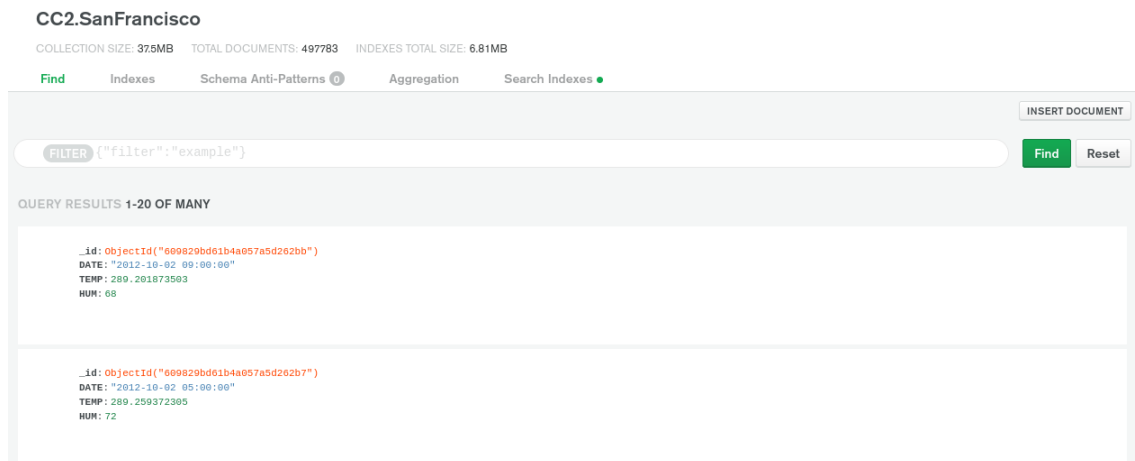
### 2.4. Tarea 5: Preprocesamiento de datos

En esta tarea, se define la obtención con **pandas** de los datos mediante **Python**. Para ello se define una función que realiza una imputación de datos omitidos y los codifica como un fichero CSV.

Esta función (se encuentra en esta línea) es posteriormente utilizada en un **PythonOperator**. Esta tarea se encuentra en esta línea.

### 2.5. Tarea 6: Almacenar la información en MongoDB Atlas

Como conocía previamente el servicio, he decidido utilizarlo en lugar de incluir una BD en una composición. Para ello he utilizado el cliente **pymongo** y la librería **dnspython** para realizar la comunicación. He creado un usuario que de acceso solo a dicha colección y se ve de la siguiente forma:



A continuación se define una función (se encuentra en esta línea) en **Python** donde utilizo **pandas** y se realiza la conexión utilizando el token de acceso con las especificaciones y el usuario limitado. Posteriormente se define la tarea con un **PythonOperator**. Se encuentra en esta línea.

## 2.6. Tarea 7: Descargar servicios de GitHub

A continuación se descargan ambos servicios desde GitHub utilizando para ello la herramienta **git** y clonando el repositorio. Esto se realiza nuevamente mediante un **BashOperator**. Se encuentra en esta línea.

### 2.6.1. Servicios

En este apartado se describen los servicios. El primer servicio utiliza **ARIMA** para realizar un modelo que permita realizar la predicción de los datos, este modelo se encuentra en el repositorio Carlosma7/cc2\_airflow\_arima\_v1.

Por concretar un poco más se distingue entre model.py donde se define el modelo de entrenamiento y su exportación a **ZIP** y servidor.py donde con **Flask** defino la API del servicio.

Por otro lado, la segunda opción escogida ha sido utilizar la API oficial de **OpenWeatherMap**, la cual ofrece información actualizada, y mediante su API no necesitamos entrenar ningún modelo. Este modelo se encuentra en el repositorio Carlosma7/cc2\_airflow\_arima\_v2.

## 2.7. Tarea 8: Testear los servicios

En esta tarea se definen dos **BashOperator** los cuales simplemente lanzan **pytest** como marco de pruebas sobre test unitarios de **unittest**. Los tests se encuentran definidos respectivamente en los repositorios test V1 y test V2. La tarea se encuentra en esta línea.

## 2.8. Tarea 9: Copiar fuentes en máquina a desplegar

En esta tarea se pasan los ficheros utilizando la herramienta **scp** mediante una copia segura de estos ficheros. Esto se realiza al droplet definido previamente mediante un **BashOperator**. Esta tarea se encuentra en esta línea.

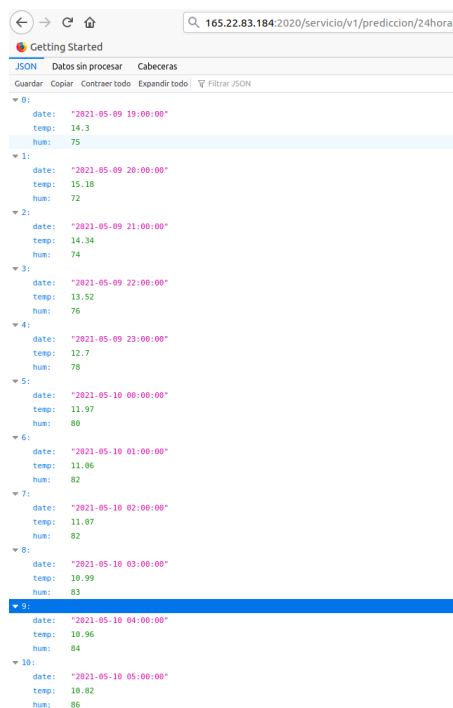
## 2.9. Tarea 10: Crear el contenedor en la máquina a desplegar

En esta tarea se ejecuta el contenedor asociado al servicio diseñado (en este caso a ambos) y se ejecuta el Dockerfile que se encuentra en ambos repositorios de los servicios Dockerfile1 y Dockerfile2 respectivamente.

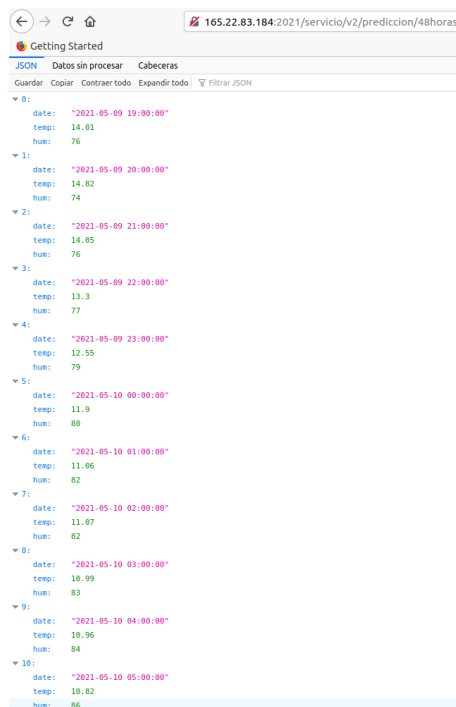
Para esta tarea se utiliza nuevamente un **BashOperator** y se encuentra en esta línea.

## 2.10. Tarea 11: Desplegar el contenedor

En esta tarea finalmente se despliega el contenedor del servicio en la máquina final, y se puede acceder. A continuación se muestra la ejecución en ambas máquinas:



	JSON	Datos sin procesar	Cabeceras
▼ 0:	date: "2021-05-09 19:00:00" temp: 14.3 hum: 75		
▼ 1:	date: "2021-05-09 20:00:00" temp: 15.18 hum: 72		
▼ 2:	date: "2021-05-09 21:00:00" temp: 14.34 hum: 74		
▼ 3:	date: "2021-05-09 22:00:00" temp: 13.52 hum: 76		
▼ 4:	date: "2021-05-09 23:00:00" temp: 12.7 hum: 78		
▼ 5:	date: "2021-05-10 00:00:00" temp: 11.97 hum: 80		
▼ 6:	date: "2021-05-10 01:00:00" temp: 11.06 hum: 82		
▼ 7:	date: "2021-05-10 02:00:00" temp: 11.67 hum: 82		
▼ 8:	date: "2021-05-10 03:00:00" temp: 10.99 hum: 83		
▼ 9:	date: "2021-05-10 04:00:00" temp: 10.56 hum: 84		
▼ 10:	date: "2021-05-10 05:00:00" temp: 10.82 hum: 86		



Para poder realizar el despliegue nuevamente se utiliza **SSH** y se ejecuta el contenedor con los servicios abriendo los puertos *2020* y *2021*, respectivamente. La tarea se encuentra en esta línea.

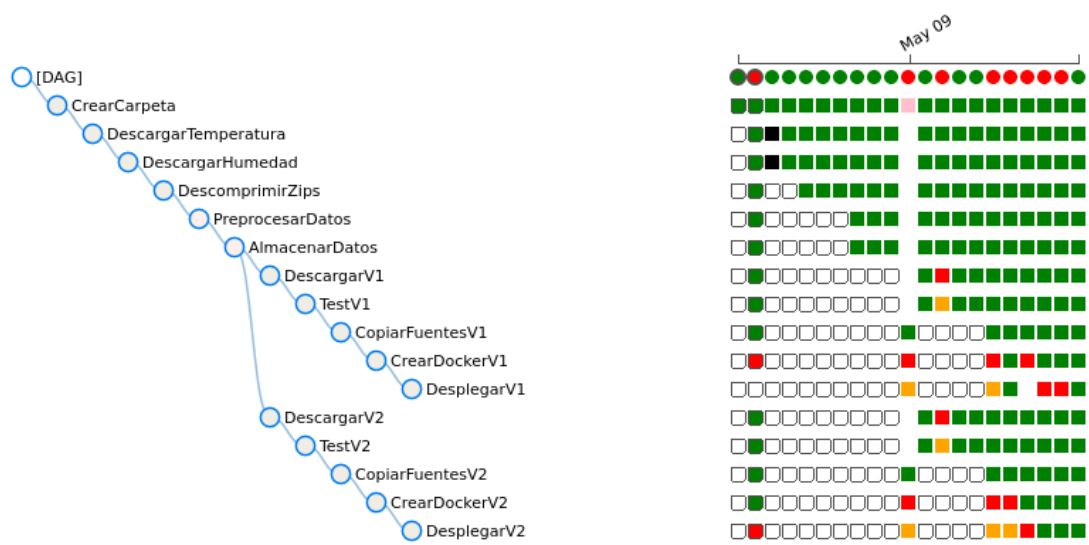
## 2.11. Flujo de tareas

A continuación en esta línea se puede observar el flujo de **AirFlow**. Este flujo se visualiza mediante un grafo de la siguiente manera:



Y mediante un árbol de la siguiente forma:





### 3. Conclusiones

Durante la realización de la práctica me he encontrado con diversos problemas que aunque finalmente he podido solucionar, no ha sido una tarea sencilla debido a que la documentación encontrada sobre **AirFlow** era escasa o con difícil comprensión. Si bien destacar que a pesar de los problemas obtenidos, he podido realizarla sin una gran complejidad.

Sobre el trabajo destacaría que a pesar de ser una tarea relativamente sencilla, la cantidad de trabajo necesaria desde el punto de formación que nos encontramos actualmente quizás es demasiado grande debido a la escasa formación recibida previamente sobre dicho tema o sobre el tema **Cloud Computing** en general. Sin embargo, no todo son pegas y considero que es no solo una metodología interesante, sino realmente práctica y que vale la pena aprender (más hoy en día).

Sobre **AirFlow** en particular destacaría que es una herramienta muy interesante, potente y flexible que permite realizar un trabajo muy escalable y realizar una programación de tareas de forma sencilla.

Además, en mi caso poseía conocimientos sobre **MongoDB Atlas** y una máquina desplegada en **DigitalOcean** por lo que me ha resultado sencillo incorporarlas en esta práctica, pero entiendo que una persona que no conozca estas herramientas quizás tenga una tarea más compleja por delante.

Considero que gracias a esta práctica he aprendido bastante aunque como pequeña reflexión creo que la formación que poseemos de cara a la realización de la misma es escasa. Espero poder utilizar **AirFlow** en el futuro.