TRABAJO PRÁCTICO PROGRAMACIÓN AVANZADA PARA GRANDES VOLÚMENES DE DATOS

Integrantes: Martina De Angelis y Santiago Lares Harbin

Como primer paso, configuramos IAM en AWS para asegurar que los servicios pudieran interactuar adecuadamente entre sí. Posteriormente, preparamos los datos crudos necesarios para nuestro DAG y los almacenamos en un bucket de S3.  
  
Luego, procedimos a instalar Airflow en nuestra consola dentro de un entorno virtual específico. Para verificar el correcto funcionamiento del DAG, realizamos pruebas locales antes de su implementación en una instancia EC2. Durante este proceso, enfrentamos varios retos relacionados con la instalación y configuración de Airflow, así como con la ejecución eficiente de los DAGs en el entorno de Airflow. Tras resolver estos problemas y confirmar su funcionamiento local, migramos el sistema a la instancia EC2. Esto implicó cambiar la versión preinstalada de Python por una versión compatible con Airflow, copiar la carpeta de DAGs y reinstalar y reconfigurar Airflow en EC2. Además, tuvimos que ajustar la configuración de EC2 para permitir el tráfico a través del puerto 8080, utilizado por Airflow.  
  
Finalmente, configuramos una API para visualizar los resultados. En comparación con Airflow, la configuración local de la API fue relativamente más sencilla, ya que no requería múltiples comandos. Además, el desarrollo del código para la API fue más directo, dado que las consultas SQL empleadas no requerían una codificación extensa.  
  
La API fue diseñada con una arquitectura modular para asemejarse lo más posible a un entorno de producción real, uno de los objetivos clave de la materia. Después de desarrollar la aplicación y verificar su correcto funcionamiento localmente, se procedió a instalar Docker Desktop y los paquetes necesarios.  
  
Se creó un archivo Dockerfile con los comandos necesarios, listando todas las dependencias del entorno virtual en un archivo app\_requirements.txt. A continuación, se generó la imagen de la aplicación en un contenedor Docker. Sin embargo, surgió un problema con las dependencias, ya que inicialmente se utilizó una copia del entorno virtual (venv). Esto requirió desinstalar todos los paquetes del venv e instalar las dependencias necesarias una por una. Superado este inconveniente, se ejecutó la imagen localmente y, tras varios intentos, se verificó que funcionaba correctamente.  
  
Posteriormente, se creó una instancia de ECS y un repositorio de ECR. La configuración de estas instancias presentó dificultades debido a las configuraciones de claves que los usuarios IAM debían tener. Las políticas de acceso predeterminadas no funcionaban correctamente, lo que obligó a especificar manualmente en un archivo JSON los accesos necesarios de la política del usuario. Una vez solucionado esto, se creó el repositorio de ECR, se obtuvo el token de acceso temporal y se subió la imagen al ECR para que pudiera ejecutarse en App Runner. Finalmente, se levantó la instancia de App Runner.  
  
Adicionalmente, se encontró un problema debido a que la imagen se había construido por defecto en una arquitectura que no permitía que Uvicorn funcionara. Este problema ocasionó varios intentos fallidos hasta que se descubrió la causa y se resolvió adecuadamente.  
  
Los pasos que llevamos a cabo fueron:  
  
1. Construir la imagen de Docker.  
2. Crear el repositorio de ECR.  
3. Subir la imagen a ECR.  
4. Levantar la imagen en App Runner.  
  
Para concluir, el trabajo práctico requirió una inversión significativa de tiempo y una revisión exhaustiva de todos los temas vistos en clase. Fue una experiencia enriquecedora, ya que implicó construir todo el proceso desde cero, lo que nos obligó a investigar y enfrentarnos a errores comunes en situaciones reales. Este proceso nos proporcionó una valiosa oportunidad para aplicar los conocimientos teóricos en un entorno práctico y desarrollar habilidades esenciales para la resolución de problemas en el ámbito profesional.