Taller 02 – En parejas o individual

Participantes:

- _____
- _____

Proyecto pokedex (en Flask)

- Pokenea (nacidos en Antioquia). Los pokeneas tienen la siguiente información:
    - Id
    - Nombre
    - Altura
    - Habilidad
    - Imagen
    - Frase filosófica
- Cree un sistema con 2 secciones (rutas):
    - Una ruta desplegará en un json con el id, nombre, altura, y habilidad de alguno de los Pokeneas (lo sacará de forma aleatoria del arreglo de Pokeneas -> no necesitamos una BD, con que esté en un arreglo quemado apenas es). Sugerencia: inspírese en este sitio web (https://www.pokemon.com/el/pokedex/). Y además el JSON incluirá el id del contenedor desde el cual se está corriendo la aplicación.
    - La otra ruta, mostrará por pantalla la imagen y la frase filosófica de un Pokenea de manera aleatoria. Y además mostrará el id del contenedor desde el cual se está corriendo la aplicación.
- Las imágenes de los Pokeneas estarán almacenadas en Amazon S3 -> Bucket. Deles acceso público a todos los elementos del bucket siguiendo este tutorial: https://docs.aws.amazon.com/elastictranscoder/latest/developerguide/gs-2-create-s3-buckets.html
    - **SUGERENCIAS:**
        - Para hacer los datos públicos, en la sección **Configuración de bloqueo de acceso público para este bucket** desactive el bloqueo a todo acceso público.
        - Para usar el bucket desde su aplicación de Flask, pueden usar la librería boto3 que es el Kit de Desarrollo de Software (SDK) de Amazon Web Services (AWS) para Python.
        - Al final de este documento, verán un anexo con un ejemplo de como mostrar imágenes desde AWS para que se den una idea general.
        - Las variables de entorno de acceso las pueden encontrar aquí: https://console.aws.amazon.com/iam en Users-> Security Credentials ->Create Access Key. OJO las credenciales no se pueden ver después de cerrar el diálogo así que guárdenlas!
        - S3_BUCKET es el nombre que le den al bucket.
        - Póngale la misma región al bucket que a las instancias que van a crear.

- Cree entre 7 y 10 Pokeneas.
- Mejore la arquitectura del proyecto para que no quede todo junto en un solo archivo.
- Suba el proyecto a GitHub.
- Cree un nuevo proyecto en DockerHub y conecte el proyecto GitHub a este DockerHub (con GitHub actions).

Despliegue AWS

- Cree 4 instancias en AWS con Docker (utilice un template como el definido en el tutorial).
- Una instancia será la líder (inicie Docker Swarm), y desde ahí obtenga el token para unirse como “manager”.
- Las otras 3 instancias se unirán al cluster de Docker Swarm como “managers”.
- Desde la líder, cree un nuevo servicio basado en el DockerHub del nuevo proyecto pokeneas. Cree 10 réplicas de ese servicio.

Verificación

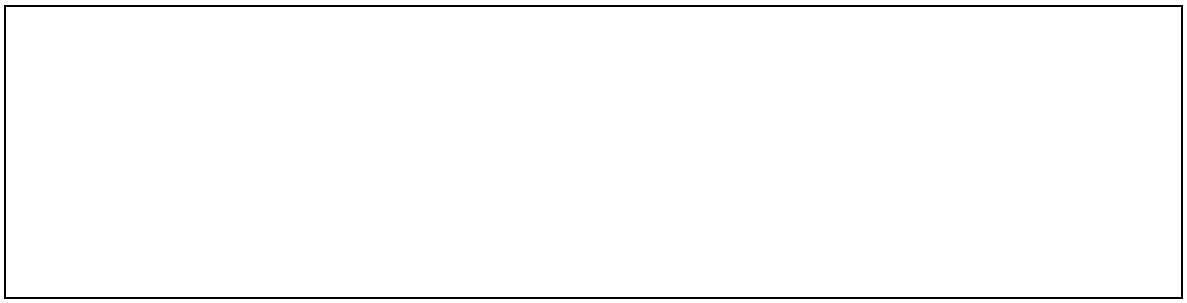
- Verifique que el servicio se ejecuta adecuadamente (verifique las 2 rutas) y que el id del contenedor varía entre los diferentes contenedores creados en las diferentes instancias de GCP.

Pantallazos

- A continuación, pegue el pantallazo de la consola donde se vean las 10 réplicas (contenedores).

- A continuación, pegue el pantallazo de la lista de imágenes almacenadas en AWS Amazon S3

- A continuación, pegue dos pantallazos de la aplicación (de la ruta donde se ven las imágenes) ejecutando (con 2 ids de contenedores diferentes).



Acceso a la APP

- **IP para que el docente pueda probar: _____**
- **Link al repositorio de Github del proyecto: _____**

ANEXOS

```python
from flask import Flask, render_template_string
import boto3
import os
from dotenv import load_dotenv

load_dotenv()

app = Flask(__name__)

s3 = boto3.client(
    "s3",
    aws_access_key_id=os.getenv("AWS_ACCESS_KEY_ID"),
    aws_secret_access_key=os.getenv("AWS_SECRET_ACCESS_KEY"),
    region_name=os.getenv("AWS_REGION"),
)

S3_BUCKET = os.getenv("S3_BUCKET")


@app.route("/imagenes")
def mostrar_imagenes():
    # Obtener la lista de objetos (archivos) del bucket
    objects = s3.list_objects_v2(Bucket=S3_BUCKET)

    # Extraer los nombres de los archivos
    image_urls = []
    for obj in objects.get("Contents", []):
        key = obj["Key"]
        url = f"https://{S3_BUCKET}.s3.amazonaws.com/{key}"
        image_urls.append(url)

    # Renderizar una plantilla HTML básica con las imágenes
    html = """
    <h1>Imágenes desde S3</h1>
    {% for url in image_urls %}
        <div>
            <img src="{{ url }}" style="max-width:300px;"><br>
            <small>{{ url }}</small>
        </div>
        <hr>
    {% endfor %}
    """
    return render_template_string(html, image_urls=image_urls)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80)

  return go(f, seed, [])
}
```