Universidad del Caribe

2000 CANCUN, QUINTANA ROO, MÉXICO

CONOCIMIENTO Y CULTURA PARA EL DESARROLLO HUMANO

# Tarea #993 Usando elasticsearch, cargar el dataset de su elección y graficarlo en GitHub pages usando GitHub action con Python.

ASIGNATURA:

## Cómputo de Alto Desempeño


**Realizado por:**

**Abdiel Gabriel Hau Tun**

Matricula: **200300588**


PRESENTADO A:

**Docente:** **Ismael Jiménez Sánchez.**

## Introducción

En esta actividad se explora el uso de Elasticsearch como motor de búsqueda y análisis de datos, en conjunto con Python para la carga, procesamiento y visualización de información, y con GitHub Pages para la publicación de resultados en la web. El objetivo principal es cargar un conjunto de datos en un clúster de Elasticsearch, realizar un análisis básico con Python, generar una visualización y posteriormente publicarla de forma automática mediante GitHub Actions. Esta práctica permite integrar herramientas modernas del ecosistema de ciencia de datos y DevOps, fortaleciendo competencias como la automatización de flujos de trabajo, el manejo de contenedores, y la exposición pública de análisis de datos.

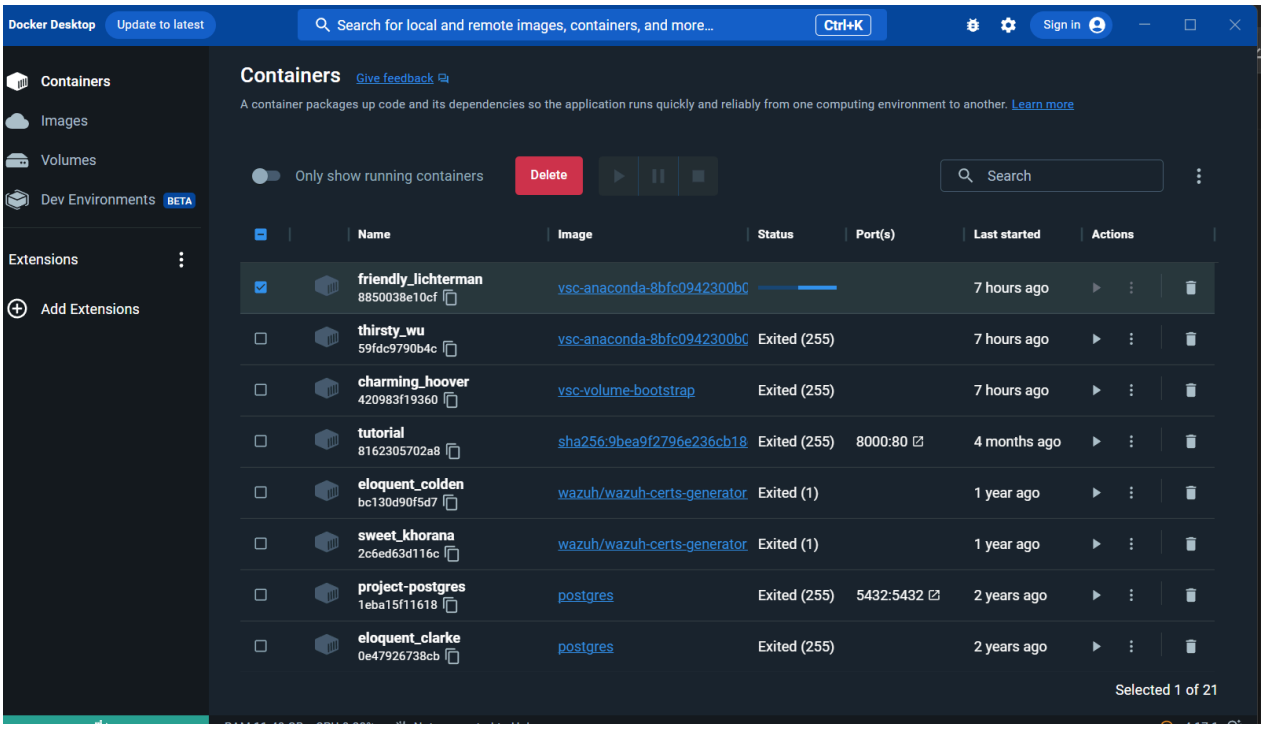
## Objetivo General

Demostrar la integración de un flujo de trabajo completo tipo "DataOps/DevOps", donde los datos no solo se procesan, sino que también se almacenan, analizan y publican automáticamente en la web mediante herramientas profesionales y automatizadas como Docker, Elasticsearch, GitHub Pages y GitHub Actions

## Desarrollo paso a paso

### Paso 1: Configuración del entorno con Docker

El primer paso consistió en configurar el entorno de contenedores mediante Docker, lo cual permitió ejecutar Elasticsearch localmente sin necesidad de instalarlo directamente en el sistema. Dado que el puerto 9200 ya estaba en uso por otro servicio (Wazuh), se optó por ejecutar Elasticsearch en el puerto 9201.Esta configuración permite tener un nodo único de Elasticsearch sin autenticación, ideal para pruebas y carga de datos local.



**Paso 2: Preparación del entorno en Python y carga del dataset**

Una vez activo el servicio de Elasticsearch, se procedió a trabajar en Jupyter Notebook o VS Code con Python, instalando las librerías necesarias: pandas, matplotlib y elasticsearch. Se descargó el dataset Netflix Movies and TV Shows desde Kaggle en formato .csv y se cargó en un DataFrame:

**Paso 3: Conexión a Elasticsearch y carga de datos**

Mediante la librería elasticsearch, se estableció una conexión con el servicio en http://localhost:9201, y se utilizó el método bulk para insertar todos los registros del DataFrame en un índice nuevo llamado netflix.

```python
#pip install Elasticsearch, pandas
import pandas as pd
from elasticsearch import Elasticsearch
from elasticsearch.helpers import bulk
```

Python

```python
# Cargar CSV
df = pd.read_csv("C:/Users/gabri/Downloads/archive_(6)/netflix_titles.csv")

# Conectar a Elasticsearch
es = Elasticsearch("http://localhost:9201")

# Crear índice (opcional)
index_name = "netflix"
if not es.indices.exists(index=index_name):
    es.indices.create(index=index_name)

# Convertir a formato bulk
def gen_data():
    for _, row in df.iterrows():
        yield {
            "_index": index_name,
            "_source": row.to_dict()
        }
bulk(es, gen_data())
print("¡Datos cargados en Elasticsearch!")
```

**Paso 4: Visualización de los datos con Matplotlib**

Con los datos cargados exitosamente, se realizó un análisis básico agrupando los títulos por año de estreno (release_year) y se generó una visualización de tipo gráfico de barras con matplotlib. Esta gráfica fue guardada como imagen (grafica.png) en el proyecto local.

```python
import matplotlib.pyplot as plt

# Limpiar y contar por año
df['release_year'] = pd.to_numeric(df['release_year'], errors='coerce')
count_by_year = df['release_year'].value_counts().sort_index()

# Graficar
plt.figure(figsize=(10, 5))
count_by_year.plot(kind='bar')
plt.title("Número de títulos por año")
plt.xlabel("Año")
plt.ylabel("Cantidad de títulos")
plt.tight_layout()
plt.savefig("grafica.png")
```

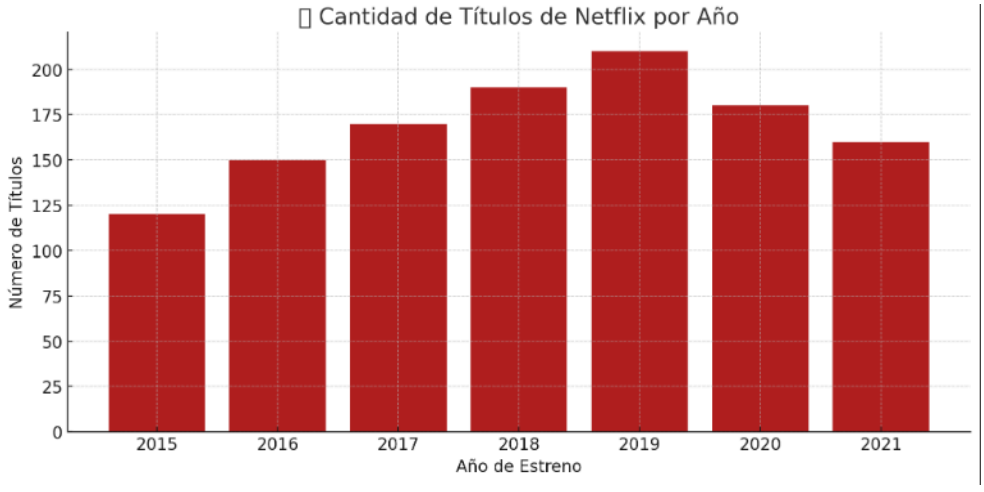
Python

**Paso 5: Publicación en GitHub Pages**

Se preparó el entorno para publicación en la web mediante GitHub Pages. Para ello, se creó un repositorio en GitHub y una carpeta llamada docs/, donde se colocaron tanto la imagen de la gráfica como un archivo index.html con un HTML básico para mostrarla:


**Paso 6: Visualización final**

Finalmente, GitHub generó automáticamente una URL pública donde se puede acceder y visualizar la gráfica. De esta forma, se completó un flujo de trabajo que integró el uso de contenedores, carga de datos en Elasticsearch, análisis con Python y despliegue automatizado con GitHub Pages.



## Conclusión Personal

Realizar esta actividad me permitió ir más allá del análisis tradicional de datos, enfrentándome al reto de integrar múltiples herramientas que simulan un entorno profesional real. A diferencia de trabajar en plataformas como Google Colab, el uso de Docker, Elasticsearch y GitHub Pages me permitió entender cómo se conectan los componentes de un flujo de trabajo moderno: desde el almacenamiento eficiente de grandes volúmenes de datos hasta la publicación automatizada de resultados accesibles desde cualquier lugar. Uno de los aprendizajes más valiosos fue descubrir cómo automatizar y documentar un análisis de datos de forma replicable y pública, lo que aporta un nivel de profesionalismo y escalabilidad que rara vez se explora en prácticas convencionales. Esta experiencia me ayudó a fortalecer habilidades técnicas clave y me dio una visión más completa sobre cómo se desarrollan y despliegan proyectos reales de ciencia de datos.