Guía Completa

Ejecución

1. Ubicarse en la raíz del repo

```
cd /ruta/al/repo/Proyecto-EXTRACCION-Y-GESTION-DE-DATOS-MASIVOS
```

---

2. Verificar dataset

```
ls -lh data/Individual_Incident_2020.csv
```

Debe pesar aproximadamente **1.5 GB**.

---

3. Arrancar Spark con el script incluido

```
sudo bash scripts/start_spark_cluster.sh #La contraseña es admin123
```

¿Qué hace?

- Ejecuta `docker-compose up -d`
- Aplica permisos en `/usr/local/spark`
- Inicia master y 2 workers conectados a:

```
spark://spark-master:7077
```

---

4. Ejecutar experimento (Logistic Regression)

```
docker exec spark-master /usr/local/spark/bin/spark-submit /opt/spark-
code/etapa3_mllib_clasificacion.py
```

Tiempo estimado: **13 minutos**

---

5. Generar reportes

```
docker exec spark-master python /opt/spark-code/etapa3_report_viz.py
```

---

6. Revisar resultados

Métricas obtenidas

```
cd data/reports
nano etapa3_report.txt
```

7. Limpieza opcional

```
docker-compose down
```

8. Herramientas utilziadas

Para la ejecución del experimento se utilizó Proxmox VE como plataforma de virtualización, donde se creó y administró una máquina virtual Ubuntu. En Proxmox se emplearon herramientas de gestión de VM, snapshots, consola integrada, monitoreo de recursos y administración de almacenamiento mediante LVM. Dentro del sistema operativo Ubuntu se utilizaron herramientas como APT, Bash, y utilidades del sistema para ampliar el volumen (lvextend, resize2fs).

Sobre este entorno se instaló Docker Engine y Docker Compose, los cuales permitieron desplegar un cluster distribuido de Apache Spark compuesto por un Spark Master y dos Spark Workers. Para gestionar el cluster se utilizaron los scripts de Spark (start-master.sh, start-worker.sh) además de un script personalizado (start_spark_cluster.sh). Finalmente, el análisis de datos y el experimento de machine learning fueron ejecutados con PySpark dentro de contenedores basados en la imagen jupyter/pyspark-notebook.