

Proyecto 3 Big Data

Pablo Moreno Quintero

Juan Sebastián Camacho Palacio

Samuel Salazar Salazar

Escuela de Ciencias e Ingeniería, Universidad EAFIT

Pregrado en Ingeniería de Sistemas

Edwin Nelson Montoya Munera

23 de noviembre de 2024

1. Configuraciones previas
 - a. Configurar un nuevo clúster EMR (o clonarlo)
 - b. Crear 3 nuevos buckets (en S3)
 - Bucket 1: Recibe los datos crudos
 - Bucket 2: Almacena datos procesados
 - Bucket 3: Almacena los resultados del análisis

Buckets de uso general (5) Información Todas las regiones de AWS

Los buckets son contenedores de datos almacenados en S3.

Nombre	Región de AWS	Analizador de acceso de IAM
<input type="radio"/> aws-logs-637423661635-us-east-1	EE. UU. Este (Norte de Virginia) us-east-1	Ver analizador para us-east-1
<input type="radio"/> bucket-rawdata-p3-telematica	EE. UU. Este (Norte de Virginia) us-east-1	Ver analizador para us-east-1
<input type="radio"/> bucket-refineddata-p3-telematica	EE. UU. Este (Norte de Virginia) us-east-1	Ver analizador para us-east-1
<input type="radio"/> bucket-trusteddata-p3-telematica	EE. UU. Este (Norte de Virginia) us-east-1	Ver analizador para us-east-1
<input type="radio"/> labsbucket-pmorenoq	EE. UU. Este (Norte de Virginia) us-east-1	Ver analizador para us-east-1

2. Enviar información a los buckets
 - a. Acceder al servicio de jupyterhub

EMR en EC2: Clústeres > My cluster PMORENOQ

Propiedades Acciones de arranque Instancias (hardware) Pasos Aplicaciones Configuraciones Monitorización Eventos

Interfaces de usuario de aplicaciones Información
Las aplicaciones instaladas en el clúster de Amazon EMR publican interfaces de usuario (IU) como sitios web. Puede utilizarlas para supervisar la actividad del clúster.

☒ **IU de la aplicación en el clúster**
Las IU en el clúster solo están disponibles mientras se está ejecutando el clúster. Utilice los siguientes enlaces para comenzar. Para obtener acceso a todas las IU de la aplicación, configure el túnel de SSH.

☐ **IU de aplicación persistente**
Las IU persistentes no requieren el túnel de SSH, ya que se alojan fuera del clúster y están disponibles durante 30 días después de que finalice la aplicación.

IU de la aplicación activas
Estas IU de aplicaciones en clúster están disponibles sin el túnel de SSH.
IU de la aplicación [\[?\]](#)
[IU del servidor de historial de Spark](#)

IU de la aplicación en el nodo principal
Estas requieren que el túnel de SSH esté habilitado. [Habilitar una conexión SSH](#)

Aplicación	URL de la IU [?]
Administrador de recursos	http://ec2-18-205-6-42.compute-1.amazonaws.com:8088/
JupyterHub	https://ec2-18-205-6-42.compute-1.amazonaws.com:9443/
Livy	http://ec2-18-205-6-42.compute-1.amazonaws.com:8998/
Nodo del nombre de HDFS	http://ec2-18-205-6-42.compute-1.amazonaws.com:9870/
Servidor de historial de Spark	http://ec2-18-205-6-42.compute-1.amazonaws.com:18080/
Tonalidad	http://ec2-18-205-6-42.compute-1.amazonaws.com:8888/
Zeppelin	http://ec2-18-205-6-42.compute-1.amazonaws.com:8890/

- b. Crear un archivo de python para acceder a la API

Files **Running** Clusters

Select items to perform actions on them.

Name	State	Size
RawDataToS3.ipynb	Running	
jupyterhub-proxy.pid		
jupyterhub.sqlite		
jupyterhub_cookie_secret		

[Upload](#) [New](#) [Refresh](#)

Notebook: [Python 3](#) [PySpark](#) [Spark](#) [Other:](#) [Text File](#) [Folder](#)

c. Crear el archivo para llevar los datos de la API al bucket RawData

*Instalar las librerías necesarias previo a correr el programa y reiniciar el kernel

RawDataToS3
Last Checkpoint: hace 8 minutos (autosaved)

[Logout](#) [Control Panel](#)

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run

```

In [2]: import requests
import boto3

# Descargar archivo desde la API y guardarlo en S3
def download_covid_data_to_s3():
    url = 'https://www.datos.gov.co/resource/gt2j-8ykr.json'
    response = requests.get(url)
    s3 = boto3.client('s3')
    s3.put_object(Bucket='bucket-rawdata-p3-telematica', Key='covid_data.json', Body=response.content)

download_covid_data_to_s3()

```

```

import requests
import boto3

# Descargar archivo desde la API y guardarlo en S3
def download_covid_data_to_s3():
    url = 'https://www.datos.gov.co/resource/gt2j-8ykr.json'
    response = requests.get(url)
    s3 = boto3.client('s3')
    s3.put_object(Bucket='bucket-rawdata-p3-telematica', Key='covid_data.json',
Body=response.content)

download_covid_data_to_s3()

```

d. Corremos el programa y verificamos en nuestro bucket o en HUE que se haya generado el archivo covid_data.json

bucket-rawdata-p3-telematica Información

Objetos | Propiedades | Permisos | Métricas | Administración | Puntos de acceso

Objetos (1) Información

Copiar URI de S3 Copiar URL Descargar Abrir Eliminar **Acciones** Crear carpeta Cargar

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	covid_data.json	json	24 Nov 2024 9:46:42 AM -05	522.8 KB	Estándar

3. Automatización para procesos ETL

a. Volver a JupyterHub y crear un archivo de pySpark

b. Ingresar el código

```
from pyspark.sql import SparkSession

# Crear una SparkSession
spark = SparkSession.builder.appName("CovidDataETL").getOrCreate()

# Leer datos desde S3 utilizando el esquema inferido automáticamente
data_raw = spark.read.option("multiline", "true").json("s3a://bucket-rawdata-p3-
telematica/covid_data.json")

# Mostrar los primeros registros para confirmar la estructura inferida
data_raw.show(10)

# Eliminar filas que estén completamente vacías
data_cleaned = data_raw.na.drop(how='all')

# Aplicar otros filtros necesarios, por ejemplo, queremos casos confirmados si la
columna 'estado' existe
if 'estado' in data_cleaned.columns:
    data_filtered = data_cleaned.filter(data_cleaned['estado'].isin(['Leve',
'Moderado', 'Grave']))
else:
    data_filtered = data_cleaned

# Cachear los datos limpios antes de escribir
data_filtered.cache()

# Guardar los datos procesados en S3 zona Trusted
data_filtered.write.mode('overwrite').parquet("s3a://bucket-trusteddata-p3-
telematica/cleaned_covid_data")

# Leer los datos procesados desde el bucket Trusted
data_trusted = spark.read.parquet("s3a://bucket-trusteddata-p3-
telematica/cleaned_covid_data")

# Guardar los datos procesados en S3 zona Refined en formato CSV
data_trusted.write.mode('overwrite').option("header", "true").csv("s3a://bucket-
refineddata-p3-telematica/refined_covid_data_csv")

# Leer los datos procesados desde el bucket Refined para confirmar
data_refined = spark.read.option("header", "true").csv("s3a://bucket-refineddata-
p3-telematica/refined_covid_data_csv")

# Mostrar una muestra de los datos procesados para confirmar que están bien
data_refined.show(10)
```

```
# Contar la cantidad de registros en cada etapa
print(f"Cantidad de registros sin filtrar: {data_raw.count()}")
print(f"Cantidad de registros después de eliminar filas completamente nulas:
{data_cleaned.count()}")
print(f"Cantidad de registros en el CSV refinado: {data_refined.count()}")
```

```
In [6]: from pyspark.sql import SparkSession

# Crear una SparkSession
spark = SparkSession.builder.appName("CovidDataETL").getOrCreate()

# Leer datos desde S3 utilizando el esquema inferido automáticamente
data_raw = spark.read.option("multiline", "true").json("s3a://bucket-rawdata-p3-telematica/covid_data.json")

# Mostrar los primeros registros para confirmar la estructura inferida
data_raw.show(10)

# Eliminar filas que estén completamente vacías
data_cleaned = data_raw.na.drop(how='all')

# Aplicar otros filtros necesarios, por ejemplo, queremos casos confirmados si la columna 'estado' existe
if 'estado' in data_cleaned.columns:
    data_filtered = data_cleaned.filter(data_cleaned['estado'].isin(['Leve', 'Moderado', 'Grave']))
else:
    data_filtered = data_cleaned

# Cachear los datos limpios antes de escribir
data_filtered.cache()

# Guardar los datos procesados en S3 zona Trusted
data_filtered.write.mode('overwrite').parquet("s3a://bucket-trusteddata-p3-telematica/cleaned_covid_data")

# Leer los datos procesados desde el bucket Trusted
data_trusted = spark.read.parquet("s3a://bucket-trusteddata-p3-telematica/cleaned_covid_data")

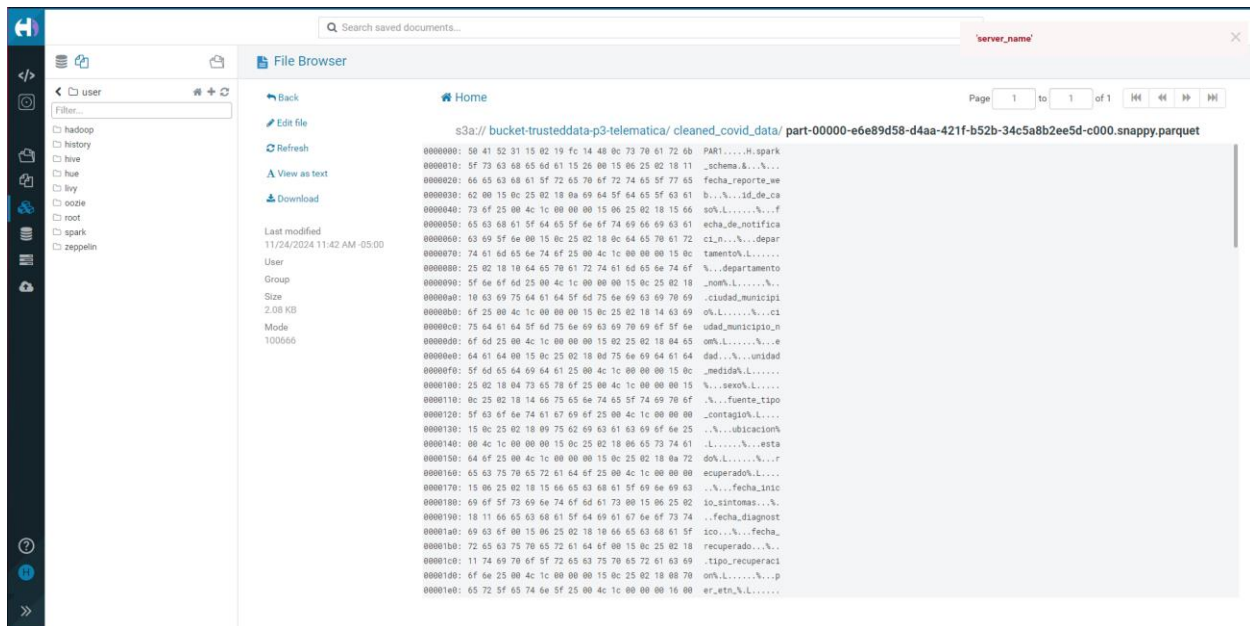
# Guardar los datos procesados en S3 zona Refined en formato CSV
data_trusted.write.mode('overwrite').option("header", "true").csv("s3a://bucket-refineddata-p3-telematica/refined_covid_data_csv")

# Leer los datos procesados desde el bucket Refined para confirmar
data_refined = spark.read.option("header", "true").csv("s3a://bucket-refineddata-p3-telematica/refined_covid_data_csv")

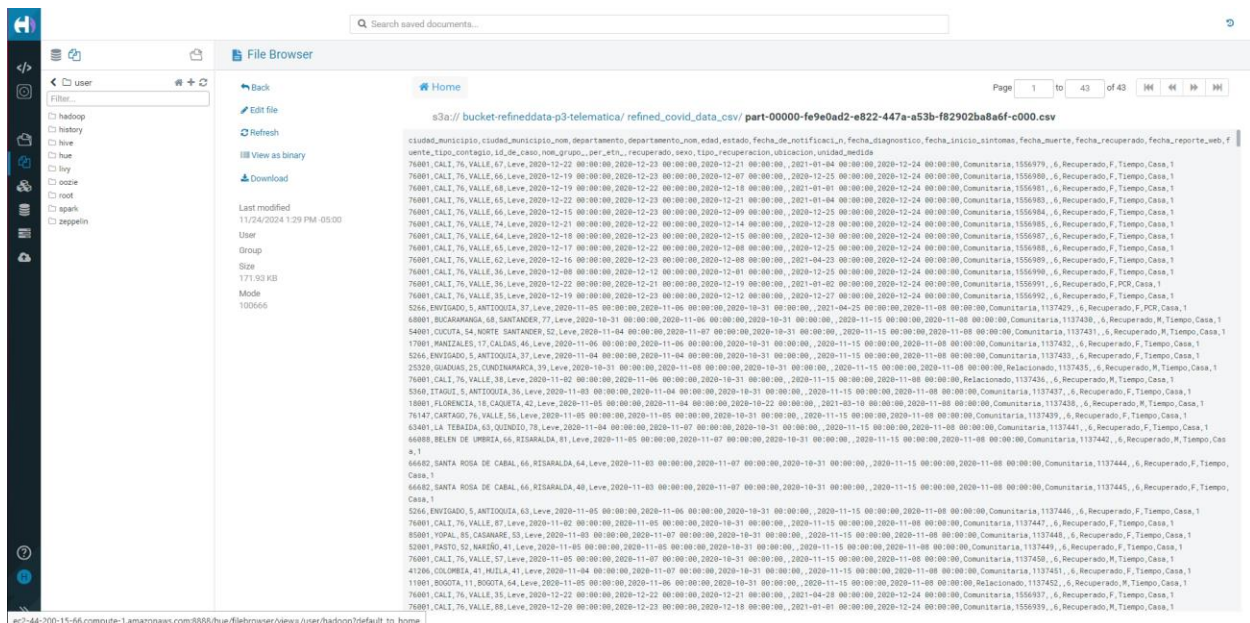
# Mostrar una muestra de los datos procesados para confirmar que están bien
data_refined.show(10)

# Contar la cantidad de registros en cada etapa
print(f"Cantidad de registros sin filtrar: {data_raw.count()}")
print(f"Cantidad de registros después de eliminar filas completamente nulas: {data_cleaned.count()}")
print(f"Cantidad de registros en el CSV refinado: {data_refined.count()}")
```

c. Accedemos al servicio HUE y revisamos que el archivo este en el bucket trusted



d. Y revisamos el contenido del bucket refined, quedaría algo así



4. Realización de búsquedas

a. Mediante Hive se pueden realizar consultas de la siguiente forma

```

-- Crear la tabla en Hive
CREATE EXTERNAL TABLE IF NOT EXISTS covid_data_refined (
    ciudad_municipio STRING,
    ciudad_municipio_nom STRING,
    departamento STRING,
    departamento_nom STRING,
    edad INT,
    estado STRING,
    fecha_de_notificaci_n STRING,
    fecha_diagnostico STRING,
    fecha_inicio_sintomas STRING,
    fecha_muerte STRING,
    fecha_recuperado STRING,
    fecha_reporte_web STRING,
    fuente_tipo_contagio STRING,
    id_de_caso STRING,
    nom_grupo_ STRING,
    per_etn_ STRING,
    recuperado STRING,
    sexo STRING,
    tipo_recuperacion STRING,
    ubicacion STRING,
    unidad_medida INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 's3://bucket-refineddata-p3-telematica/refined_covid_data_csv/';

-- Consulta 1: Conteo de casos por departamento
SELECT departamento_nom, COUNT(*) AS total_casos
FROM covid_data_refined
GROUP BY departamento_nom
ORDER BY total_casos DESC;

-- Consulta 2: Número de casos activos, recuperados y fallecidos
SELECT estado, COUNT(*) AS total_casos
FROM covid_data_refined
GROUP BY estado;

-- Consulta 3: Número de casos por género
SELECT sexo, COUNT(*) AS total_casos
FROM covid_data_refined
GROUP BY sexo;

```



```

-- Consulta 4: Promedio de edad de los casos fallecidos
SELECT AVG(edad) AS promedio_edad
FROM covid_data_refined
WHERE estado = 'Fallecido';

-- Consulta 5: Número de casos por tipo de contagio
SELECT fuente_tipo_contagio, COUNT(*) AS total_casos
FROM covid_data_refined
GROUP BY fuente_tipo_contagio
ORDER BY total_casos DESC;

-- Consulta 6: Conteo de casos por rango de edad
SELECT
CASE
    WHEN edad < 20 THEN 'Menor de 20'
    WHEN edad BETWEEN 20 AND 39 THEN '20-39'
    WHEN edad BETWEEN 40 AND 59 THEN '40-59'
    ELSE 'Mayor de 60'
END AS rango_edad,
COUNT(*) AS total_casos
FROM covid_data_refined
GROUP BY
CASE
    WHEN edad < 20 THEN 'Menor de 20'
    WHEN edad BETWEEN 20 AND 39 THEN '20-39'
    WHEN edad BETWEEN 40 AND 59 THEN '40-59'
    ELSE 'Mayor de 60'
END
ORDER BY total_casos DESC;

-- Consulta 7: Tiempo promedio de recuperación (en días)
SELECT AVG(DATEDIFF(TO_DATE(fecha_recuperado), TO_DATE(fecha_diagnostico))) AS
tiempo_promedio_recuperacion
FROM covid_data_refined
WHERE estado = 'Recuperado' AND fecha_recuperado IS NOT NULL;

-- Consulta 8: Número de casos en el último mes
SELECT COUNT(*) AS total_casos_ultimo_mes
FROM covid_data_refined
WHERE TO_DATE(fecha_diagnostico) >= ADD_MONTHS(CURRENT_DATE(), -1);

-- Consulta 9: Distribución de casos por ubicación del paciente
SELECT ubicacion, COUNT(*) AS total_casos
FROM covid_data_refined
GROUP BY ubicacion

```

```

ORDER BY total_casos DESC;

-- Consulta 10: Comparación de casos leves vs graves por departamento
SELECT departamento_nom,
       COUNT(CASE WHEN estado = 'Leve' THEN 1 END) AS casos_leves,
       COUNT(CASE WHEN estado = 'Grave' OR estado = 'Fallecido' THEN 1 END) AS
casos_graves
FROM covid_data_refined
GROUP BY departamento_nom
ORDER BY casos_graves DESC;

-- Consulta 11: Casos por fecha de notificación (tendencia temporal)
SELECT fecha_de_notificaci_n, COUNT(*) AS total_casos
FROM covid_data_refined
GROUP BY fecha_de_notificaci_n
ORDER BY TO_DATE(fecha_de_notificaci_n);

```

	departamento_nom	total_casos
1	ANTIOQUIA	222
2	BOGOTA	168
3	VALLE	152
4	SANTANDER	108
5	BARRANQUILLA	53
6	CARTAGENA	52
7	TOLIMA	26
8	SUCRE	25
9	META	16
10	CUNDINAMARCA	15
11	STA MARTA D.E.	13
12	CALDAS	12
13	GUAJIRA	12
14	RISARALDA	12
15	CORDOBA	11
16	ATLANTICO	10
17	CAUCA	10
18	BOLIVAR	8
19	NORTE SANTANDER	7
20	HUILA	6
21	NARIÑO	4
22	CASANARE	4
23	ARAUCA	3
24	QUINDIO	3
25	CAQUETA	2
26	CESAR	2