Proyecto 3

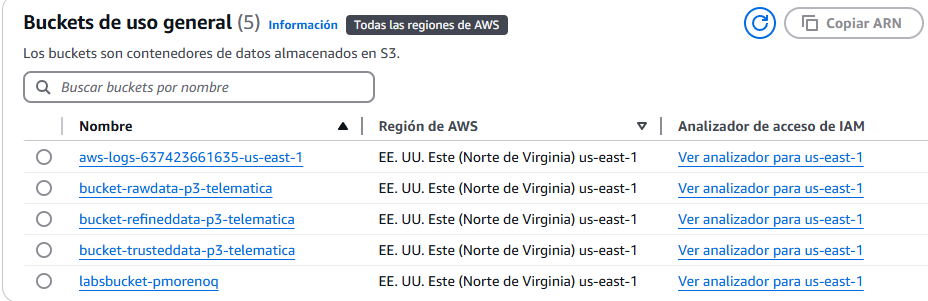
Pablo Moreno Quintero

Juan Sebastían Camacho Palacio

Samuel Salazar Salazar

1. Configuraciones previas
2. Configurar un nuevo clúster EMR (o clonarlo)
3. Crear 3 nuevos buckets (en S3)

* Bucket 1: Recibe los datos crudos
* Bucket 2: Almacena datos procesados
* Bucket 3: Almacena los resultados del análisis

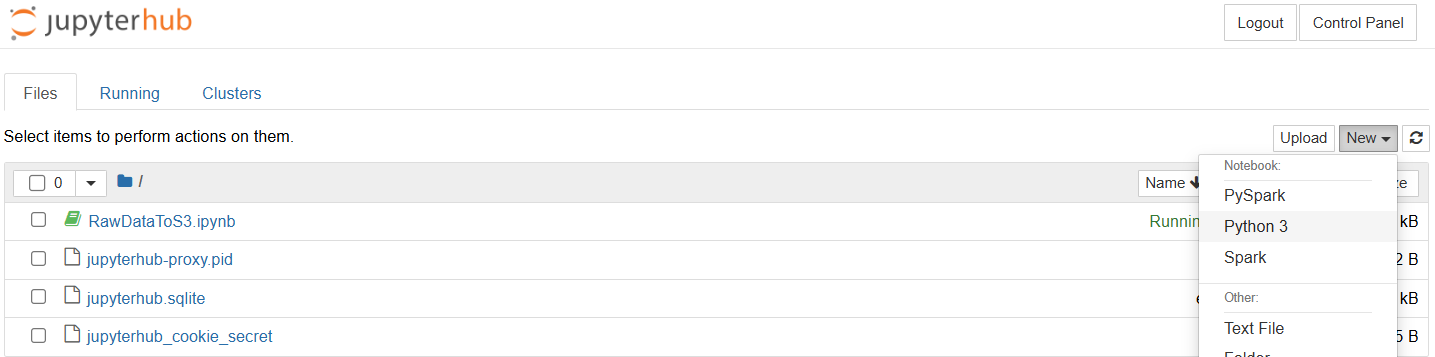


1. Enviar información a los buckets
   1. Acceder al servicio de jupyterhub

A screenshot of a computer

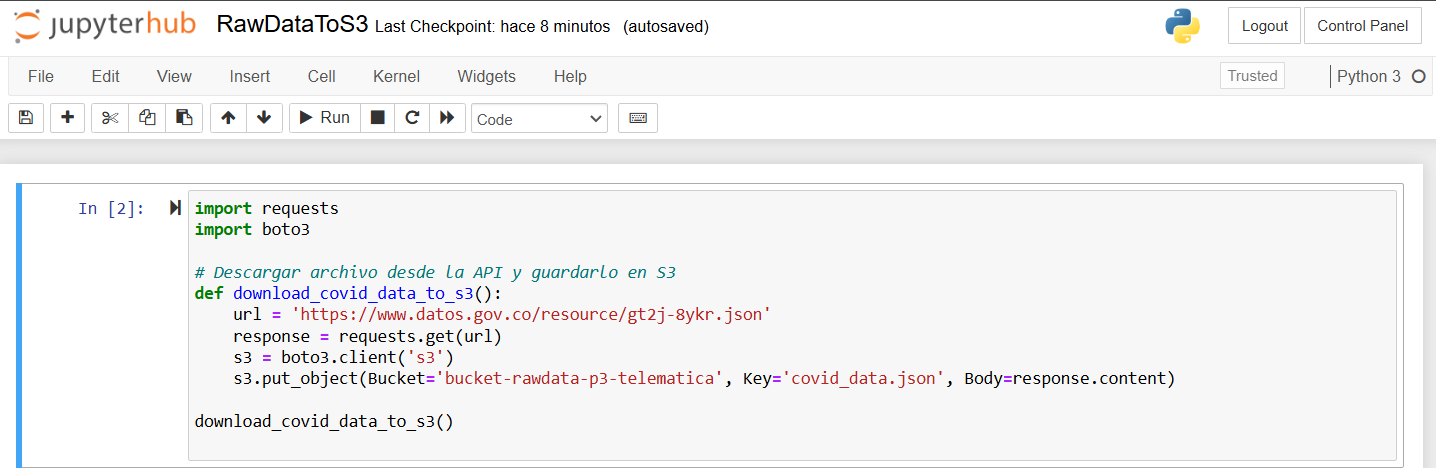
Description automatically generated

* 1. Crear un archive de python para acceder a la API



* 1. 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



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'

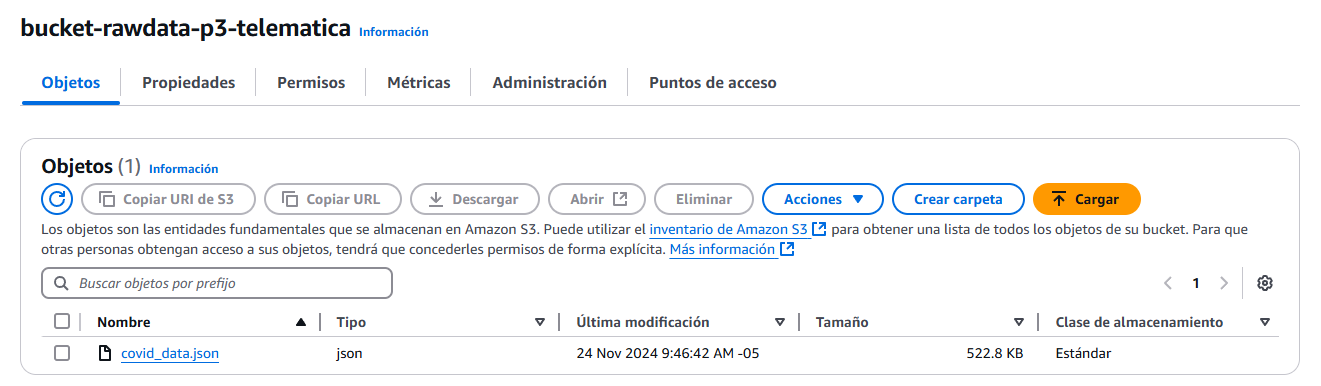
    response = requests.get(url)

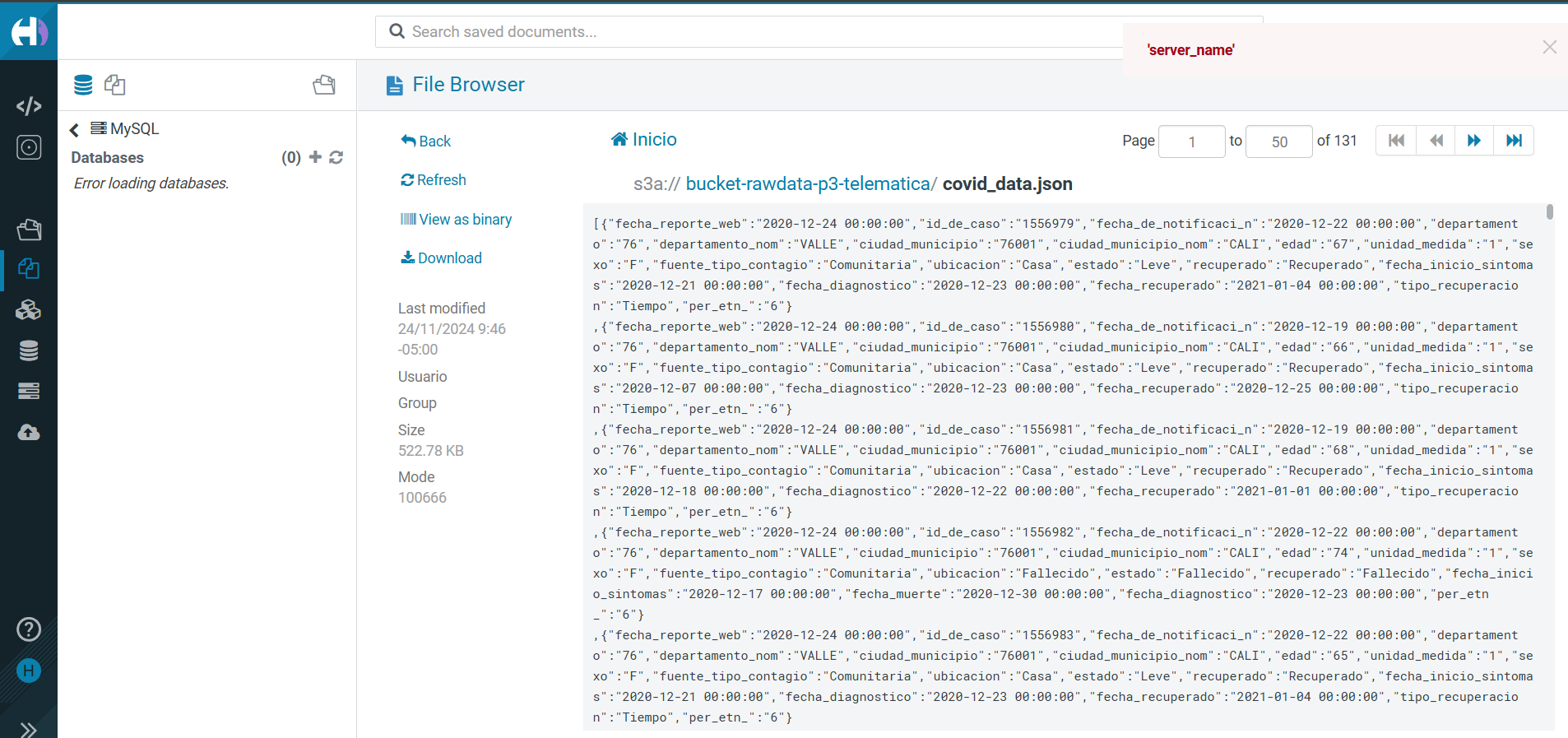
    s3 = boto3.client('s3')

    s3.put\_object(Bucket='bucket-rawdata-p3-telematica', Key='covid\_data.json', Body=response.content)

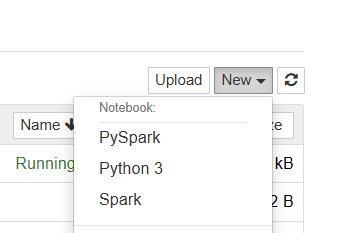
download\_covid\_data\_to\_s3()

* 1. Corremos el programa y verificamos en nuestro bucket o en HUE que se haya generado el archivo covid\_data.json





1. Automatización para procesos ETL
2. Volver a JupyterHub y crear un archivo de pySpark



1. 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()}")

A screenshot of a computer program

Description automatically generated

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

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

1. Realización de búsquedas
2. 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);

A screenshot of a computer

Description automatically generated