Proyecto 3

Automatización del Proceso de Captura, Ingesta, Procesamiento y Salida de Datos COVID-19 en Colombia

Luis Miguel Torres Villegas
Jerónimo Acosta Acevedo
Juan José Restrepo Higuita

Universidad EAFIT
ST0263 – Sistemas Distribuidos
Edwin Nelson Montoya Múnera

Semestre 2025-2

1. Introducción

Este proyecto implementa un *pipeline* batch de datos para el análisis de casos de COVID-19 en Colombia, utilizando servicios de cómputo distribuido y almacenamiento en la nube sobre Amazon Web Services (AWS). El flujo cubre las etapas de captura, ingesta, procesamiento, generación de resultados analíticos y exposición de estos resultados a través de una API HTTP.

El *dataset* principal corresponde a los casos positivos de COVID-19 reportados en Colombia mediante datos abiertos, complementados con información demográfica y de capacidad hospitalaria simulada, organizada lógicamente como si proviniera de una base de datos relacional.

2. Objetivos

2.1. Objetivo general

Diseñar e implementar un *pipeline* batch automatizado que tome datos de COVID-19 como fuente principal, los ingrese a un almacenamiento en la nube organizado por zonas, los procese de manera distribuida y produzca salidas refinadas listas para análisis y consumo programático.

2.2. Objetivos específicos

- Capturar y almacenar el *dataset* de casos de COVID-19 de Colombia en una zona *raw* en Amazon S3.
- Emular una fuente relacional (tipo Amazon RDS) mediante archivos CSV con datos demográficos y de capacidad hospitalaria.
- Implementar procesos ETL en EMR/Spark para limpiar, normalizar y enriquecer los datos, generando una zona *trusted*.
- Calcular métricas agregadas por departamento y generar indicadores en una zona *refined*.
- Construir vistas analíticas específicas para consumo por API (resumen nacional diario).
- Exponer los resultados a través de Amazon Athena y de un endpoint HTTP basado en AWS Lambda y Amazon API Gateway.
- Documentar la arquitectura, las decisiones de diseño y los pasos de ejecución del *pipeline*.

3. Descripción de datos y modelo

3.1. Dataset principal de COVID-19

El *dataset* principal se obtiene de Datos Abiertos Colombia (`datos.gov.co`):

- Nombre: *Casos positivos de COVID-19 en Colombia*.
- Identificador: `gt2j-8ykr`.
- Endpoint CSV (Socrata): `https://www.datos.gov.co/resource/gt2j-8ykr.csv`.

Campos relevantes utilizados en el proyecto:

- Fechas:
  - `fecha_reporte_web`
  - `fecha_de_notificaci_n`
  - `fecha_inicio_sintomas`
  - `fecha_diagnostico`
- Ubicación:
  - `departamento`, `departamento_nom`
  - `ciudad_municipio`, `ciudad_municipio_nom`
- Características:
  - `edad`, `unidad_medida`, `sexo`
  - `fuente_tipo_contagio`
- Estado:
  - `ubicacion`, `estado`, `recuperado`
  - `fecha_muerte`, `fecha_recuperado`

3.2. Datos relacionales emulados

Se define un modelo relacional lógico con dos tablas:

Tabla `departamento_demografia`

- `codigo_departamento` (PK)
- `nombre_departamento`
- `region`
- `poblacion`
- `anio_corte`

Tabla `departamento_capacidad_hospitalaria`

- `id_capacidad` (PK)
- `fecha`
- `codigo_departamento` (FK a `departamento_demografia`)
- `camas_uci_totales`
- `camas_uci_ocupadas`
- `fuente`

Por restricciones de permisos en el entorno de AWS Academy no se crea una instancia real de Amazon RDS, sino que estas tablas se materializan como archivos CSV locales:

- `data/rds/departamento_demografia.csv`
- `data/rds/departamento_capacidad_hospitalaria.csv`

Estos archivos se suben a S3 bajo:

- `s3://st0263-proyecto3-covid19/raw/rds/`

y son leídos por Spark como extractos de una base de datos relacional.

3.3. Organización en S3

El *bucket* principal del proyecto es:

- `st0263-proyecto3-covid19`

Se definen tres zonas lógicas:

- **`raw/`**:
  - `raw/covid/` → CSV descargados desde Datos Abiertos.
  - `raw/rds/` → CSV que emulan las tablas relacionales.
- **`trusted/`**:
  - `trusted/covid/` → casos de COVID limpios y enriquecidos, en formato Parquet particionado por año, mes y día.
  - `trusted/demografia/` y `trusted/capacidad_hospitalaria/` (planificados para extensión).

- `refined/`:
  - `refined/indicadores_departamento/` → métricas diarias por departamento.
  - `refined/api_views/resumen_nacional_diario/` → resumen nacional diario para consumo por API.

4. Arquitectura del pipeline

4.1. Componentes de AWS utilizados

- **Amazon S3**: almacenamiento por zonas (*raw*, *trusted*, *refined*).
- **Amazon EMR sobre EC2**: clúster con Apache Spark para procesamiento batch.
- **Amazon Athena**: consultas SQL sobre datos en S3 (zonas *refined*).
- **AWS Lambda**: función `covid_resumen_nacional` que consulta Athena.
- **Amazon API Gateway**: API HTTP que expone el resumen nacional diario.
- **AWS CLI y `boto3`**: ingesta y automatización desde scripts Python.

4.2. Flujo de datos

El flujo se organiza en las siguientes etapas:

4.2.1. Captura e ingesta

- Script `ingestion/covid_api/download_covid_to_s3.py`:
  - Descarga un CSV del dataset `gt2j-8ykr` desde Datos Abiertos.
  - Almacena el archivo en `raw/covid/` dentro del *bucket* S3.
- Script `ingestion/upload_rds_csv_to_s3.py`:
  - Sube los CSV locales de `data/rds/` a `raw/rds/`.

4.2.2. Procesamiento ETL `raw` → `trusted`

- Script de Spark: `processing/etl_trusted/covid_to_trusted.py`.
- Ejecución en EMR mediante un *step* que corre:
  - `spark-submit s3://st0263-proyecto3-covid19/scripts/etl_trusted/covid_to_trusted.py`

- Funcionalidad principal:
  - Lectura de `raw/covid/*.csv` y `raw/rds/departamento_demografia.csv`.
  - Normalización de nombres de departamento.
  - *Join* con datos demográficos.
  - Conversión de fechas y creación de columnas `anio`, `mes`, `dia`.
  - Escritura de datos limpios y enriquecidos en `trusted/covid/` en formato Parquet particionado.

4.2.3. Analítica `trusted` → `refined/indicadores_departamento`

- Script de Spark: `processing/analytics_refined/covid_indicators_refined.py`.
- *Step* en EMR que ejecuta:
  - `spark-submit s3://st0263-proyecto3-covid19/scripts/analytics_refined/covid_indicators_refined.py`
- Funcionalidad:
  - Lectura de `trusted/covid/`.
  - Agregación por fecha y departamento para obtener:
    - `casos_nuevos`
    - `casos_acumulados` (usando una ventana por departamento y fecha)
    - `casos_por_100k` (cuando hay población disponible)
  - Escritura en `refined/indicadores_departamento/` en formato Parquet particionado.

4.2.4. Vista para API `refined/indicadores_departamento` → `refined/api_views`

- Script de Spark: `processing/api_views/resumen_nacional_diario.py`.
- *Step* en EMR que ejecuta:
  - `spark-submit s3://st0263-proyecto3-covid19/scripts/api_views/resumen_nacional_diario.py`
- Funcionalidad:
  - Agregación de indicadores por departamento a nivel nacional para cada fecha.
  - Cálculo de:

- `casos_nuevos_nacional`
- `casos_acumulados_nacional`
- `poblacion_total_aprox`
- `casos_por_100k_nacional`

- Escritura de la vista en `refined/api_views/resumen_nacional_diario/` en formato Parquet particionado.

4.3. Capa de consumo: Athena y API

4.3.1. Athena

- Base de datos: `covid_analytics`.
- Tablas externas definidas:
  - `indicadores_departamento` sobre `refined/indicadores_departamento/`.
  - `resumen_nacional_diario` sobre `refined/api_views/resumen_nacional_diario/`.
- Particiones registradas mediante:
  - `MSCK REPAIR TABLE indicadores_departamento;`
  - `MSCK REPAIR TABLE resumen_nacional_diario;`

4.3.2. Lambda y API Gateway

Lambda

- Función: `covid_resumen_nacional`.
- Rol de ejecución: `LabRole` del entorno del laboratorio.
- Lógica:
  - Recibe eventos con `queryStringParameters`.
  - Si se indica `fecha=YYYY-MM-DD`, consulta en Athena la tabla `resumen_nacional_diario` para esa fecha.
  - Si no se indica fecha, obtiene la última fecha disponible.
  - Devuelve un JSON con:
    - `fecha`
    - `casos_nuevos_nacional`
    - `casos_acumulados_nacional`
    - `poblacion_total_aprox`
    - `casos_por_100k_nacional`

API Gateway

- Tipo: HTTP API.
- Nombre: `st0263-proyecto3-covid-api`.
- *Stage*: `prod`.
- URL base:
  - `https://n0znyvjr0k.execute-api.us-east-1.amazonaws.com/prod`
- Rutas:
  - `GET /resumen-nacional`
  - `GET /resumen-nacional?fecha=YYYY-MM-DD`

5. Ejecución del pipeline

La ejecución completa del flujo, desde la captura hasta el consumo, se resume en los siguientes pasos:

1. Preparar el entorno local:
   a) Clonar el repositorio del proyecto.
   b) Crear y activar un entorno virtual de Python.
   c) Instalar `requests` y `boto3`.
   d) Configurar credenciales con `aws configure`.
2. Ejecutar la ingesta:
   a) `python ingestion/covid_api/download_covid_to_s3.py`
   b) `python ingestion/upload_rds_csv_to_s3.py`
3. Crear el clúster EMR y lanzar los *steps* de Spark:
   a) ETL `raw` → `trusted` con `covid_to_trusted.py`.
   b) `trusted` → `refined/indicadores_departamento` con `covid_indicators_refined.py`.
   c) `refined/indicadores_departamento` → `refined/api_views/resumen_nacional_diario` con `resumen_nacional_diario.py`.
4. Actualizar las particiones de las tablas externas en Athena.

5. Consumir los resultados:

   a) Ejecutar consultas SQL en Athena.

   b) Consumir la API HTTP:

   - `GET /resumen-nacional`
   - `GET /resumen-nacional?fecha=YYYY-MM-DD`

6. Limitaciones y adaptaciones al entorno

- No se crea una instancia real de Amazon RDS debido a restricciones IAM (`rds:CreateDBInstance` no permitido). Se emula la fuente relacional mediante CSV en `raw/rds/`, integrados desde Spark.
- La creación de roles IAM personalizados está limitada. Se reutiliza `LabRole` como rol de ejecución para Lambda.
- La orquestación del *pipeline* se realiza mediante ejecución manual de *steps* de EMR. En un entorno productivo podría automatizarse con servicios adicionales (por ejemplo, Step Functions o herramientas externas de orquestación), que aquí no se utilizan para ajustarse al entorno del laboratorio.

7. Resultados

- Datos de COVID-19 cargados en S3 en la zona `raw/covid`.
- Datos demográficos y de capacidad hospitalaria cargados en `raw/rds`.
- Zona `trusted/covid` con datos limpios y enriquecidos, particionados por fecha.
- Zona `refined/indicadores_departamento` con indicadores diarios por departamento.
- Vista `refined/api_views/resumen_nacional_diario` con el resumen nacional diario.
- Tablas externas en Athena que permiten analizar los indicadores por departamento, región y país, así como el comportamiento agregado nacional.
- API HTTP operativa que devuelve un resumen nacional diario en formato JSON, listo para ser consumido por aplicaciones cliente o paneles de visualización.

8. Conclusiones

- Es posible construir un *pipeline* batch completo sobre AWS combinando S3, EMR, Athena, Lambda y API Gateway, incluso con restricciones de permisos, siempre que se adapten las fuentes de datos y la forma de integración.
- La separación en zonas *raw*, *trusted* y *refined* facilita la trazabilidad, permite reprocesos y clarifica las responsabilidades de cada etapa del flujo de datos.
- El uso de Spark en EMR permite procesar volúmenes grandes de información de forma distribuida, aplicar transformaciones complejas y generar indicadores analíticos en formatos eficientes (Parquet).
- Athena ofrece una capa de consulta flexible sobre S3 sin administrar servidores adicionales, lo que simplifica la exposición de datos tanto para usuarios SQL como para servicios *serverless*.
- La combinación de Lambda y API Gateway permite exponer vistas analíticas específicas como servicios HTTP, acercando el *pipeline* de datos a aplicaciones externas y facilitando la integración con otros sistemas.