Descripción de implementación de Arquitectura BATCH en AWS y Azure: Cambio climático en Medellín

David Alejandro López Atehortúa

Camilo Alejandro Vélez Medina

Juan Sebastián Ávila Arias

David Armendáriz Peña

Universidad EAFIT

Maestría en Ciencia de Datos y Analítica

Edwin Nelson Montoya Munera

Medellín, Colombia

Septiembre de 2023

Introducción.

En la actualidad, el cambio climático ha resaltado como uno de los desafíos más apremiantes para la humanidad. Ante esta problemática, la aplicación de enfoques tecnológicos y científicos se vuelve esencial para comprender, mitigar y adaptarse a los efectos del cambio climático.

En este contexto, diseñamos e implementamos una Arquitectura de Big Data tipo BATCH que permite la ingesta, el almacenamiento y el procesamiento de datos ambientales con una frecuencia diaria, provenientes del Instituto de Hidrología, Meteorología y Estudios Ambientales (IDEAM), que proporciona una fuente confiable y actualizada de información para Colombia de variables climáticas clave como temperatura, precipitación y humedad relativa.

Descripción de la arquitectura

Para este proyecto se propone una arquitectura BATCH. Esta arquitectura fue diseñada e implementada para las nubes de AWS y de Azure. Con cada nube se diseñó un flujo independiente, que son representados por el siguiente diagrama:

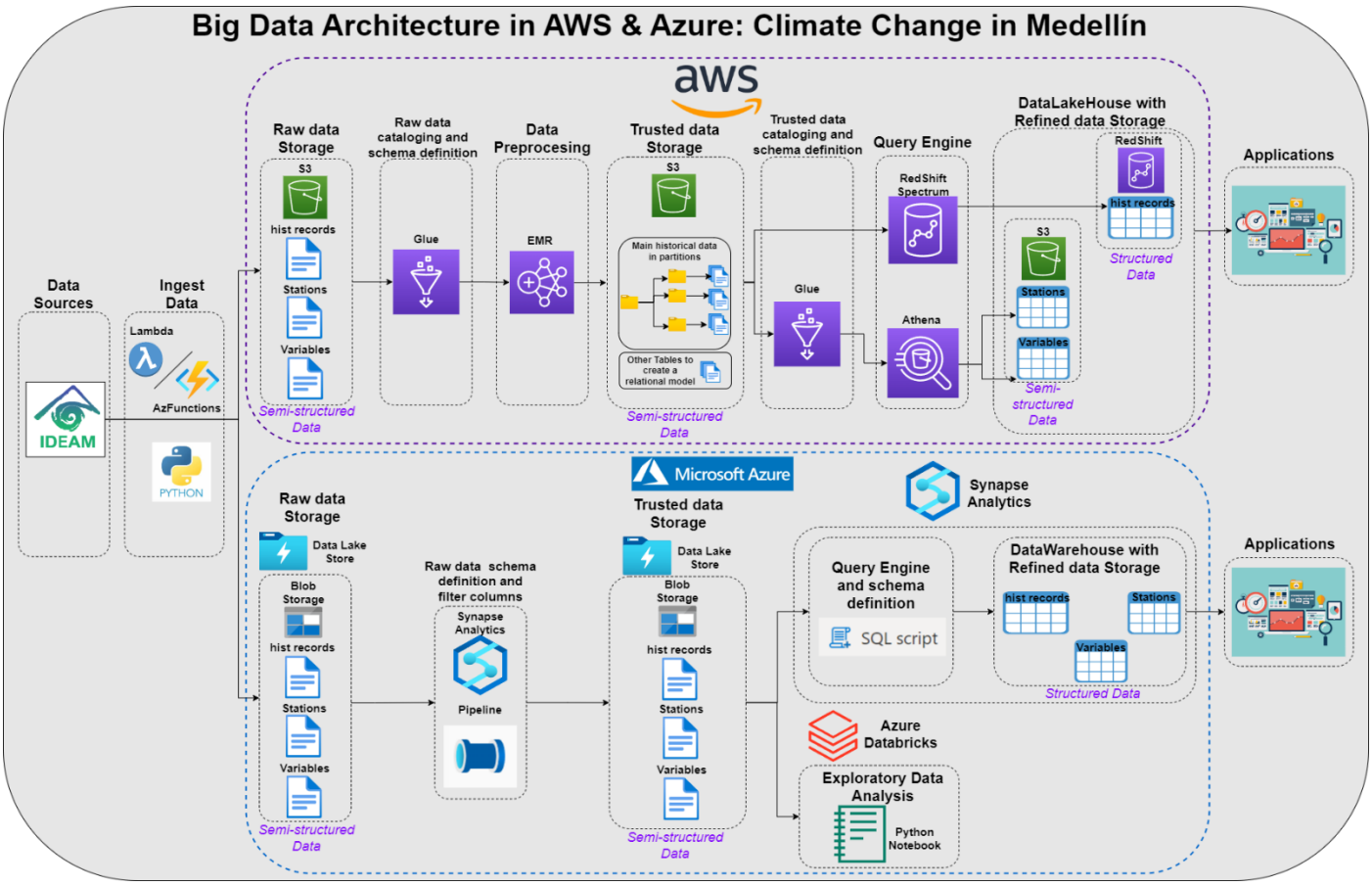

Figura 1: Arquitectura BATCH en AWS y Azure – Cambio climático en Medellín

Cada componente del diagrama juega un papel indispensable en el ciclo de vida de los datos y del DataLakeHouse. A continuación, se describe la implementación de la arquitectura, partiendo de los puntos generales (son iguales para ambos flujos de nube) y, posteriormente dando detalles para cada uno de los proveedores de nube.

Descripción de la implementación de la arquitectura – Puntos Generales.

- **Orígenes de datos:**

Se define como fuente de datos el IDEAM dado que permite extraer información de variables climáticas libremente sin necesidad de algún tipo de registro previo. Se utilizan tres entidades como insumo:

I. Catálogo de las **estaciones** de medición de todo el país: Corresponde a dos archivos, uno que contiene las estaciones del IDEAM y otro que contiene las estaciones operadas por externos.

II. Catálogo de las **variables** que se pueden consultar: Son tres archivos, donde cada uno contiene un subconjunto del total de variables que pertenecen a una categoría específica.

III. Detalle de los **registros históricos** de las variables y estaciones de interés: Compuesto por múltiples archivos, donde cada uno de ellos representa una variable para un rango de fechas.

Las tablas que contienen los catálogos de estaciones y de variables no se actualizan frecuentemente, pues solo lo hacen cuando se agrega una nueva estación de medición o se empieza a medir una nueva variable. Por otra parte, la tabla con los registros históricos se actualiza con nuevos datos todos los días.

Estos datos se obtienen en un formato CSV y, cada uno de ellos tiene variables con diferentes tipos de datos, por lo tanto, estos ***orígenes de datos*** se pueden considerar como **semi-esctructurados.**

- **Ingesta de datos:**

Inicialmente, se hace una descarga manual de los datos de las tres entidades mencionadas en el punto anterior. Para las entidades de ***estaciones*** y ***variables,*** no debe repetirse el proceso a no ser que el IDEAM anuncie una nueva estación de medición o la inclusión de una nueva variable en sus métricas, que sean de interés para analizar le cambio climático en Medellín.

Para los ***registros históricos***, se obtuvieron todos los registros existentes (hasta la fecha de descarga manual) de las variables a analizar (temperatura mínima diaria, temperatura máxima diaria, humedad relativa mínima diaria, humedad relativa máxima diaria y precipitación diaria) para las estaciones de medición de Medellín.

Posteriormente, para automatizar la descarga de los nuevos datos históricos de manera diaria, se desarrolló un script de Python que obtiene los datos del día anterior y los lleva de manera simultánea a un Bucket en S3 de AWS y a un Blob Storage en Data Lake Storage de Azure. Para ejecutar este script de manera autónoma, se plantea la implementación de una instancia en Lamba

de AWS o en Azure Functions de Microsoft Azure que permite agendar la ejecución del scipt sin preocuparse por administrar la disponibilidad del entorno en el que se ejecuta el código.

- **Diseño de zonas de almacenamiento:**

Para mantener los datos ordenados, se definen las siguientes zonas de almacenamiento en el Data Lakehouse: Almacenamiento de datos ***raw***, almacenamiento de datos ***trusted*** y almacenamiento de datos ***refined.*** Cada una de estas zonas se representa en S3 como un Bucket o carpeta.

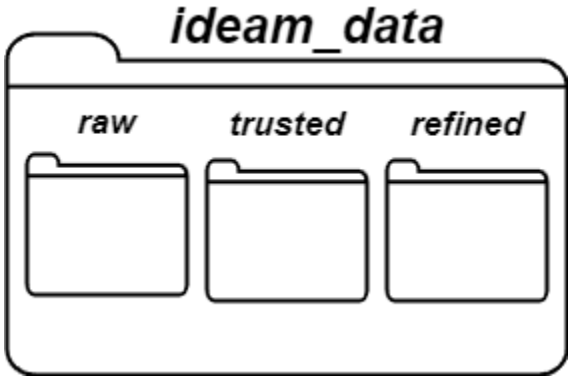

Figura 2: Zonas de almacenamiento definidas.

- **Almacenamiento de datos raw:**

Se utilizan los Buckets de S3 en AWS y los Blob Storage de Data Lake Storage en Azure para almacenar los datos. En esta zona se almacenan los datos crudos ingestados del IDEAM. Para que esta carga sea automática, el Script de Python desarrollado, no solo extrae los datos, sino que también los guarda directamente en este almacenamiento. Estos datos llegan a este almacenamiento sin estructura y en formato CSV. A continuación, se ilustra la estructura de esta zona de almacenamiento.

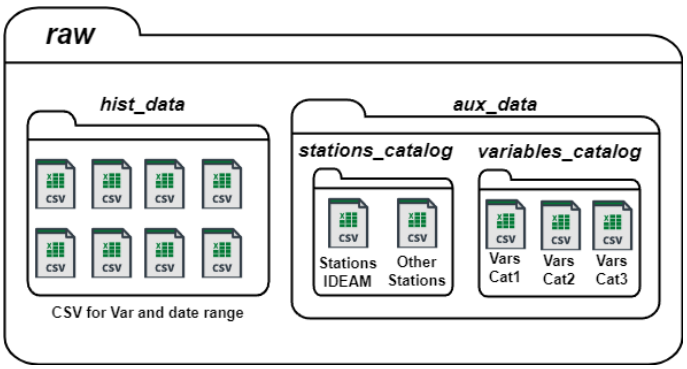

Figura 3: Estructura de almacenamiento de raw data.

Descripción de la implementación de la arquitectura – AWS.

- **Consolidación y definición de esquemas de datos raw:**

Se utiliza el servicio de GLUE para consolidar los múltiples archivos de cada entidad en una sola tabla con esquema definido. Para esto se crea un crawler por cada una de las entidades. Como resultado de este proceso quedan 3 tablas, donde cada una representa una entidad (estaciones, variables y registros históricos).

- **Preprocesamiento de datos raw y Análisis Exploratorio de Datos (EDA):**

Se utiliza el servicio de EMR, para implementar un clúster que permita realizar transformaciones por medio de Notebooks en el lenguaje de PySpark a las tablas que resultan de la catalogación de los crawlers generados en GLUE del paso anterior. Adicionalmente, en otro NoteBook con lenguaje de Python se realiza un análisis exploratorio de los datos identificando la información estadística más relevante y creando gráficos para identificar anomalías en los datos.

Se utiliza este servicio porque permite computación distribuida y paralela y, por lo tanto, es muy eficiente para procesar grandes volúmenes de datos.

- **Almacenamiento de trusted data:**

Los resultados del procesamiento de datos del paso anterior son guardados en formato CSV en la zona trusted del S3 como datos semi-estructurados. Adicionalmente, se almacenan en particiones generadas por el HDFS de EMR

- **Catalogación y definición de esquemas de datos trusted:**

Nuevamente se utiliza el servicio de GLUE de AWS para consolidar, catalogar y definir el esquema de algunas tablas del modelo de datos, específicamente, de las tablas de estaciones y de variables, esta vez tomando la información desde trusted data, Para esto se crea un crawler por cada una de las tablas.

- **Query Engine**

Se utilizan dos motores de consulta:

  - **Athena:** Se utilizó este servicio para consultar el catálogo de variables y estaciones. El resultado de la consulta es almacenado en la zona refined del almacenamiento de S3 con tipo semi-estructurado.

  - **Redshith Sprectum:** Se utiliza este servicio para crear una tabla estructurada sobre los datos históricos de las variables a analizar para las estaciones de medición de Medellín. En este caso, se leen los datos directamente desde el almacenamiento trusted, y se les crea el esquema en la definición de la tabla con SQL.

- **DataLakeHouse con almacenamiento de datos refined:**

La información Semiestructurada generada por la consulta de Athena se almacena en la zona Refined del S3. Adicional, la tabla estructurada generada por el servicio de RedShift Spectrum se almacena directamente en el motor de almacenamiento de RedShift.

- **Applications**

Estas son todas las herramientas que pueden consumir la data refinada o procesada en los pasos anteriormente mencionados, tales como; PowerBi, Tableu, Quicksight, Apis, etc.

Descripción de la implementación de la arquitectura – Azure.

- **Definición de esquemas y filtrado de columnas de datos raw:**

Se crea una serie pipelines en Synapse Analytics que ingestan los datos de los múltiples archivos del raw Blob Storage, les asigna su esquema, se eligen las columnas que se quieren conservar, y finalmente, guarda los datos obtenidos en el trusted Blob Storage como un solo archivo CSV por cada entidad.

- **Almacenamiento de datos trusted:**

Aquí se almacenan una tabla por cada entidad, que son obtenidas por los pipelines del paso anterior. Los resultados se guardan como CSV, por lo que quedan semi-estructurados.

- **Análisis Exploratorio de Datos:**

Se implementa una instancia en Azure Databricks para ejecutar Notebooks con el lenguaje de Python que permite analizar las principales estadísticas descriptivas de los datos históricos.

- **Query Engine y definición de esquemas:**

Se utilizan los scripts de SQL de Synapse Analytics para consultar y procesar los datos trusted, con el objetivo de crear unas tablas estructuradas con los resultados finales.

- **Datawarehouse con almacenamiento de datos refined:**

Aquí se almacenan las tablas creadas por las consultas del paso anterior. Las tablas refinadas se almacenan en una base de datos de Synapse Analytics. Estas tablas son estructuradas.

- **Applications**

Estas son todas las herramientas que pueden consumir la data refinada o procesada en los pasos anteriormente mencionados, tales como; PowerBi, Tableu, Quicksight, Apis, etc.