F1 EDA - Big Data Final Term Project

Enrique Ulises Báez Gómez Tagle
Sara Rocío Miranda Mateos

June 1, 2024

Contents

1 Infrastructure / Architecture

- **Data Lakehouse Configuration:** Central to our project is a Data Lakehouse architecture designed for scalability and resilience within an AWS environment. This setup includes multiple layers of data storage and processing capabilities, ensuring efficient data management and analysis.
- **Storage Layers:**
  - **Raw Data Storage (RAW Layer):** Utilizes Amazon S3 to store unprocessed, raw data ingested from various sources.
  - **Cleansed Data Storage (Cleansed Layer):** Stores cleaned and transformed data in Amazon S3, ready for further analysis.
  - **Data Warehouse Storage (Data Warehouse Layer):** Contains structured and optimized data in Amazon S3, prepared for querying and visualization.
- **Data Ingestion:**
  - **Real-Time Data Ingestion (AWS Lambda):** Handles the real-time ingestion of data, storing it in the RAW Layer.
  - **Batch Data Ingestion (Static Dataset):** Manages the batch ingestion of static datasets, such as CSV files, into the RAW Layer.
- **Data Transformation:**
  - **ETL Processing Jobs (AWS Glue):** Performs ETL (Extract, Transform, Load) operations to clean and transform data from the RAW Layer to the Cleansed Layer.
- **Data Analysis:**
  - **Data Analytics and Processing (AWS EMR):** Utilizes Amazon EMR to run advanced data processing and analytical tasks on data from the Cleansed Layer, producing structured results stored in the Data Warehouse Layer.
- **Data Cataloging:**
  - **Metadata Management (AWS Glue Data Catalog):** Catalogs the metadata of datasets in the Cleansed Layer, facilitating efficient data discovery and access.
- **Data Visualization:**
  - **Visualization and Analysis (RStudio):** Processes the curated datasets in the Data Warehouse Layer locally to create insightful visualizations and analysis, supporting decision-making.
- **Deployment Process:**

1. We initiated our deployment by setting up the AWS CLI, enabling us to interact with AWS services seamlessly.
2. Amazon S3 buckets were created to host our data layers, carefully chosen to be in proximity to our compute resources to minimize latency.
3. For infrastructure provisioning, we utilized both AWS CloudFormation and Terraform. AWS CloudFormation was used for managing AWS-specific resources, ensuring consistent and repeatable deployments within the AWS ecosystem. Terraform was employed for managing multi-cloud resources and ensuring infrastructure as code principles across various environments.
4. AWS Glue was configured to automate our ETL processes, facilitating efficient data transformation and loading into our cleansed and data warehouse layers.
5. Amazon EMR clusters were set up for big data processing, with configurations tailored to optimize performance and cost.
6. Continuous Integration and Continuous Deployment (CI/CD) pipelines were implemented using AWS CodePipeline and Jenkins, automating the deployment process and ensuring that updates to our infrastructure and applications could be rolled out smoothly.
7. Monitoring and logging were established using Amazon CloudWatch and AWS CloudTrail, providing real-time insights and audit trails for all activities within our environment.
8. Security best practices were followed, including the use of IAM roles and policies to control access, as well as encryption of data at rest and in transit to protect our sensitive information.

This infrastructure is pivotal to our project, allowing us to effectively utilize big data technologies to derive valuable insights from the Formula 1 dataset. The design prioritizes scalability, efficiency, and cost-effectiveness, ensuring it meets the high demands of large-scale data processing and analysis. Ultimately, this setup empowers us to make data-driven decisions that enhance our understanding and strategic planning within the context of Formula 1.

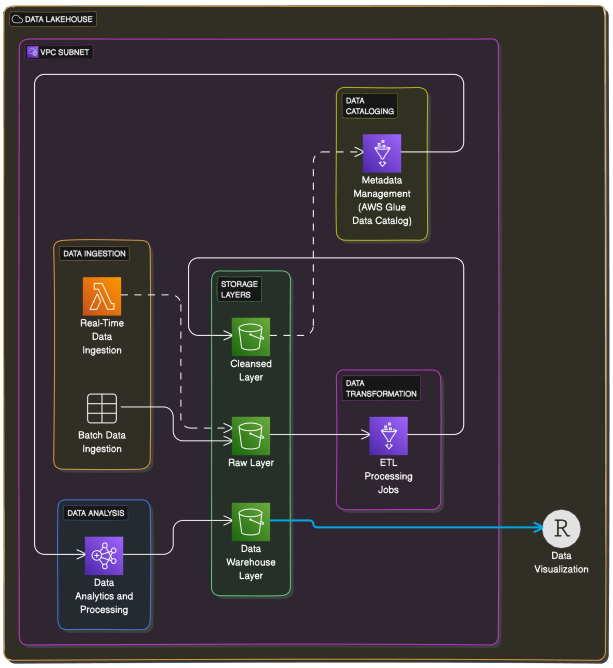

Figure 1: **AWS Data LakeHouse Architecture Diagram**

2 Data Analysis with F1 Dataset

We employed a dual dataset approach:

- **Static Dataset:** This dataset contains historical information from past Formula 1 seasons. It includes data on race results, driver standings, lap times, and other relevant metrics.
- **Dynamic Dataset:** This dataset provides real-time information for the current 2024 season. AWS Lambda is used to ingest this data by making API calls to gather the latest updates, ensuring that our analysis remains current with ongoing events.

By integrating these datasets into our Data Lakehouse architecture on AWS, we effectively manage both historical and real-time data. The static dataset allows us to analyze trends and patterns over previous seasons, while the dynamic dataset enables us to monitor and react to live data from the current season.

Our analysis aimed to uncover actionable insights that could inform race strategies and enhance the performance of both teams and drivers. We utilized SQL queries on our Amazon EMR cluster to efficiently process and analyze the substantial data volumes. These queries were meticulously crafted to explore

hypotheses regarding the relationships between various factors such as weather conditions, track characteristics, and driver performance metrics.

Upon obtaining initial results and drawing preliminary conclusions from these queries, we further refined our analysis by exporting the data to the Curated Layer in Amazon S3. These files were then imported into a dedicated workspace in RStudio for detailed visualization. This additional step allowed us to visually interpret the data more intuitively, facilitating the identification of significant patterns and trends that underpin our strategic recommendations.

2.1 Database Structure Overview

For our exploratory data analysis (EDA), we used a combination of historical and current datasets, structured into several tables with specific focuses. Each table captures unique aspects of the sport, allowing for a comprehensive analysis from multiple angles. Here's a brief guide to some tables:

- **Circuits**: Contains details about the various circuits used in the races, including location, length, and other relevant information.
- **Constructor Results**: Records the results achieved by each constructor in different races, including points and status.
- **Constructor Standings**: Summarizes the standings of constructors throughout the season, capturing their cumulative points and positions.
- **Constructors**: Lists the constructors participating in a season, along with their basic information and attributes.
- **Driver Standings**: Details the standings of drivers across the season, showing their cumulative points and positions after each race.
- **Drivers**: Contains information about the drivers, including personal details, nationality, and career statistics.
- **Lap Times**: Records the times for each lap completed during race weekends, providing timing and performance data.
- **Pit Stops**: Details the pit stops made during races, including timing and duration.
- **Qualifying**: Captures the results of the qualifying sessions, determining the starting grid for the races.
- **Races**: Provides comprehensive information about each race event, including date, location, and participating drivers and constructors.
- **Results**: Summarizes the race results for each event, including positions, points, and other performance metrics.

- **Seasons**: Lists the seasons of the championship, including the year and other relevant details.
- **Sprint Results**: Contains the results of the sprint qualifying sessions, including grid positions and points awarded.
- **Status**: Describes the status of drivers in each race, such as whether they finished or the reason for not finishing.

3 Challenges Encountered

3.1 Infrastructure-Related Challenges

Description: Initially, our project was set up on Google Cloud Platform (GCP). However, we faced a significant challenge when our GCP credits were exhausted. To continue our work without interruption, we opted to create a simplified version of a Data Lakehouse on AWS. This decision was influenced by the opportunities and services available in AWS, which allowed us to replicate the necessary functionalities we initially had on GCP. We carefully selected AWS services that closely matched those we used on GCP to ensure a smooth transition and maintain the efficiency of our data processing and analysis.

4 Project Repository

`https://github.com/enriquegomeztagle/BigData/tree/main/FinalTerm/F1-GridGuru-Project`

References