New York Institute of Technology

College of Engineering and Computing Sciences

Big Data Project Report

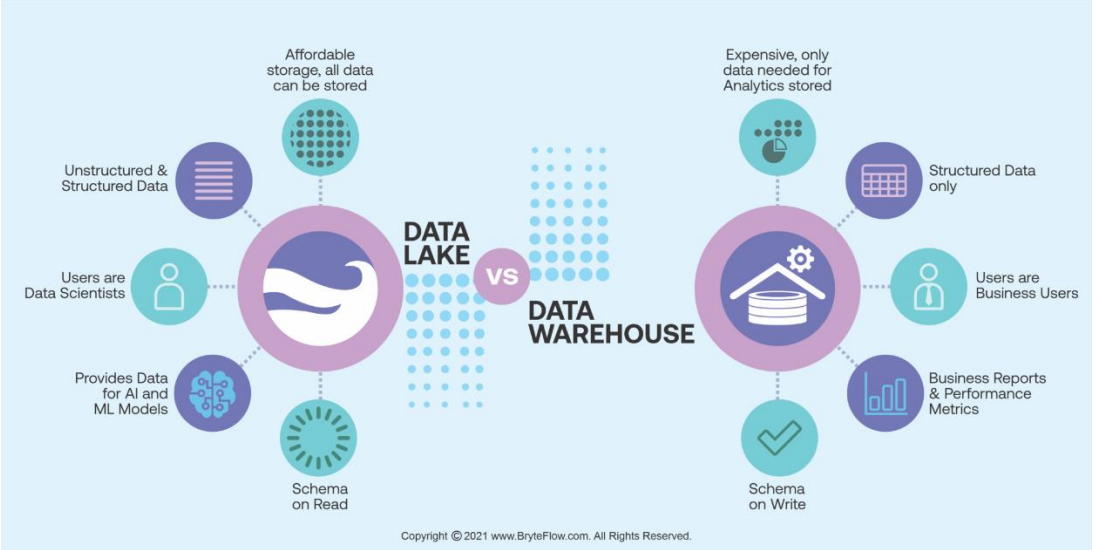
Professor: Liangwen Wu

GAJJELA HARIKA| TEJASWI VENUMULA| RAGHURAM BOBBA | NITHIN RAO DUGYALA

Aim: The objective of the project is to generate visual representations of an analysis on New York City parking tickets. This involves conducting diverse aggregations and showcasing a variety of statistics obtained from the dataset.

Data Warehouse vs Data Lake:

- **Data Warehouse:** A data warehouse is a centralized and organized storage system for a company's information. It acts like a comprehensive library, bringing together data from different parts of the business into one accessible place. This makes it easier for people in the company to analyze and understand the data, aiding in better decision-making.
- **Data Lake:** A data lake is like a vast, unstructured reservoir for storing all types of data—structured, semi-structured, and unstructured—without the need for predefined formats. It's a flexible storage space where businesses can collect and keep large amounts of raw data from various sources. Unlike a data warehouse, which organizes data neatly, a data lake allows for more flexibility and exploration, making it possible to analyze diverse and evolving datasets for insights and decision-making.
- A data warehouse and a data lake are both storage architectures designed for handling and analyzing large volumes of data,
- In our Project, we used Data Warehouse, as it specially designed for structured data, it also ensures data consistency, accuracy along with that it helps in performing complex queries and analysis on aggregated data.



Pictorial Representation of Data Lake vs Data Lake

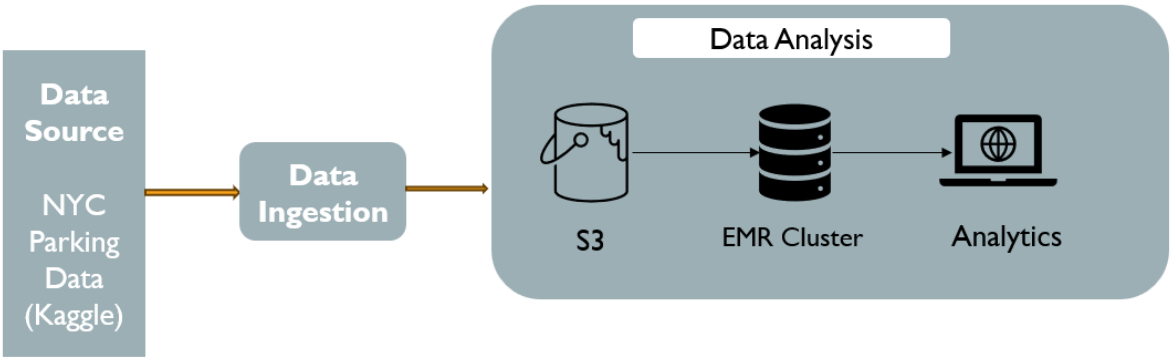
Data Collection:

In this Project we have taken dataset from Kaggle, from the below link.
https://www.kaggle.com/datasets/new-york-city/nyc-parking-tickets/data

Data Pipelining of the Project:

A data pipeline is a set of processes and tools that move and transform data from one or multiple sources to a destination for analysis, it involves the extraction, transformation, and loading (ETL) of data, ensuring that it flows smoothly from source to destination.

In this Project, we commonly used Amazon Simple Storage Service (S3) and Amazon Elastic MapReduce (EMR) are two services offered by Amazon Web Services (AWS). Where S3 is Amazon Simple Storage Service (S3) and Amazon Elastic MapReduce (EMR) are two services offered by Amazon Web Services (AWS) and Amazon EMR (Elastic MapReduce) is a cloud-based big data platform that utilizes Apache Hadoop and Apache Spark to process and analyze large datasets. EMR clusters can be part of the data processing steps in a data pipeline. Data from S3 can be loaded into an EMR cluster for processing and analysis using tools like Apache Spark or Apache Hive. The results of these computations can then be stored back in S3 or sent to other destinations.



AWS Pipeline

Analysis of the Project:

S3 Setup:

- **Sign in to AWS Console:** Log in to AWS Management console, once logged in, find and click on "S3" under storage.
- **Create S3 bucket:** Click the "Create bucket" button > Choose a globally unique bucket name and select the region for the bucket > Configure other settings like versioning, logging, and tags>Click "Create bucket" to finish.
- **Configure bucket permissions:** Click on the newly created bucket > Navigate to the "Permissions" tab > Adjust the bucket policy and access control list (ACL) settings.
- **Document Upload:** Click on the "Upload" button > Click "Add files" or "Add folder" to select the document to upload > Configure optional settings like permissions, metadata, and storage class > Click "Upload" to upload the document.
- **Document URL:** Once the document is uploaded, select it in the S3 bucket > In the details pane, find the "Object URL" or "Link" and get the URL Path of the document.

EMR Cluster setup:

- **Sign in to AWS Console:** Log in to AWS Management console, once logged in,

find and click on "EMR" under the "Analytics" section in the AWS Management Console.

- **Creating a Cluster:** Click the "Create cluster" button > Choose the "Go to advanced options" to customize the cluster.
- **Configure cluster settings:** Enter name for the cluster > Configure logging options.
- **Choose Hardware and software:** Select the EMR release version > Choose the spark application or framework > Choose 1 primary node and 5 core nodes.
- **Security Configuration:** Configure EC2 key pair, IAM role.
  Launch Cluster: click on "Create cluster" to launch EMR cluster.

Data Processing:

- **EMR Cluster Creation:** Once the EMR Cluster is created, the cluster status will transition to “Waiting”.

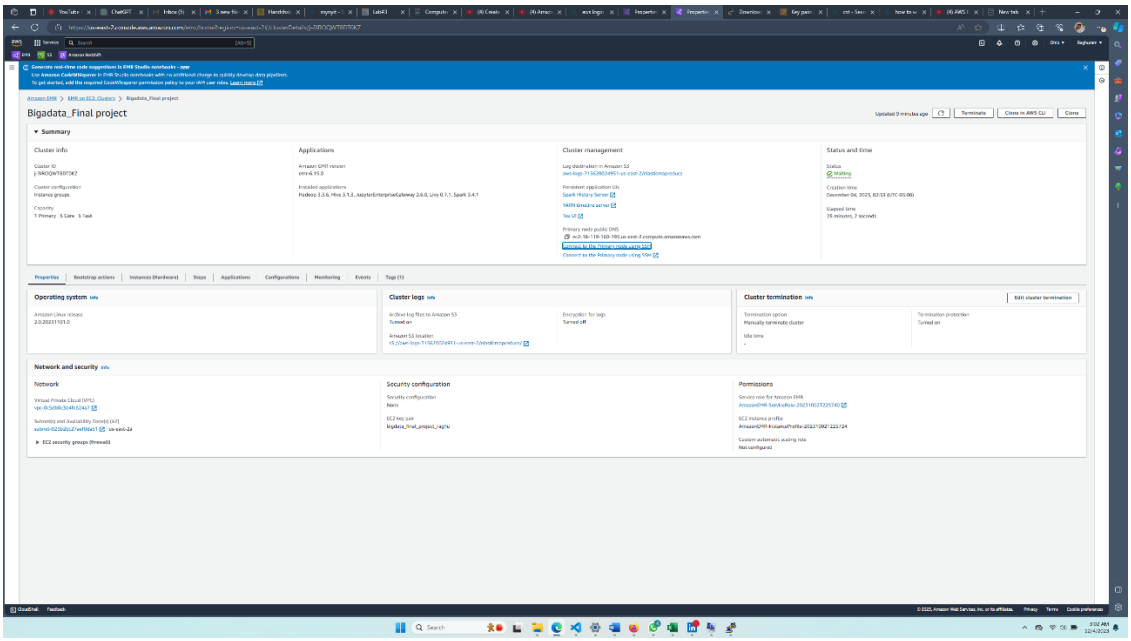


Figure 1

- **Connect to SSH:** Click on "Connect to Remote SSH" to access the remote Linux/Unix system hosted on your EMR cluster.

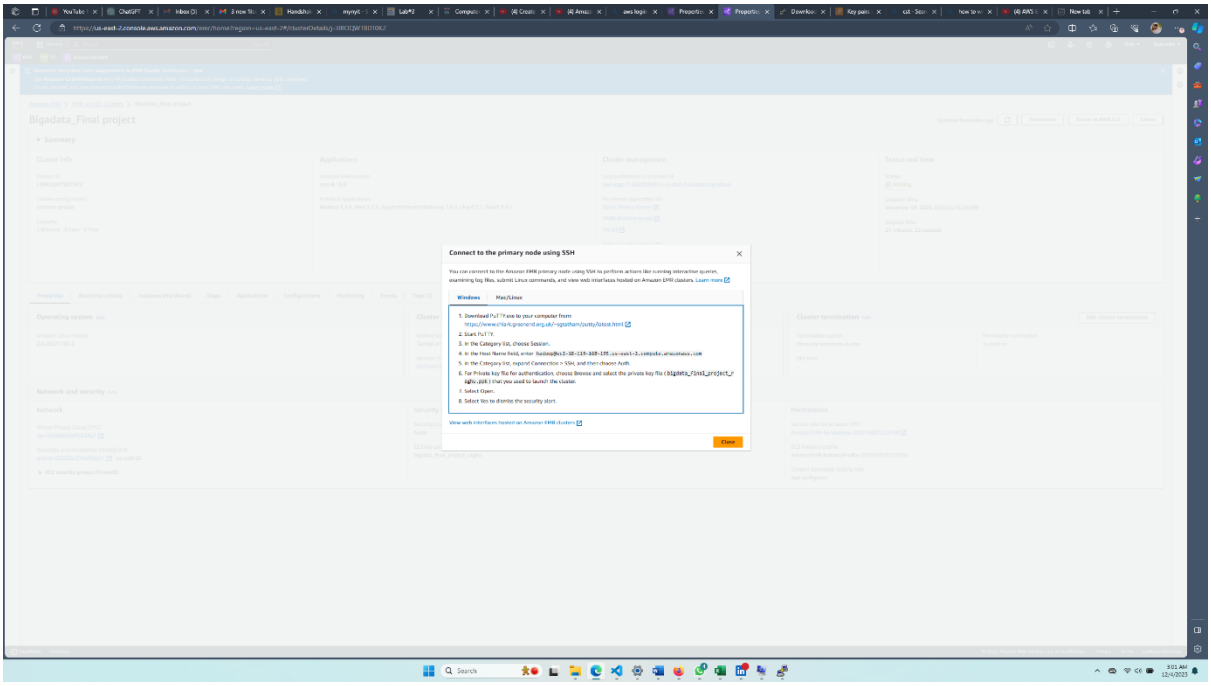


Figure 2

- **Establish Remote connection using Putty:** By using putty, we get access to connect to the remote server which helps to access the local system.

Figure 3

- **Access to the remote Linux/Unix server:** Upon successful connection, gain access to the remote Linux/Unix system from your local environment. This establishes a seamless bridge between your local system and the EMR cluster.

- **Python file creation:** Generate a Python file (e.g., filename.py) to develop the code for data processing on the EMR cluster.

- **Initiate the spark code:** Save the Python file and initiate the Spark job using the following command:

  spark-submit filename.py.

- Once the spark job has been submitted, it calls python file and execute the code and displays the respective results.

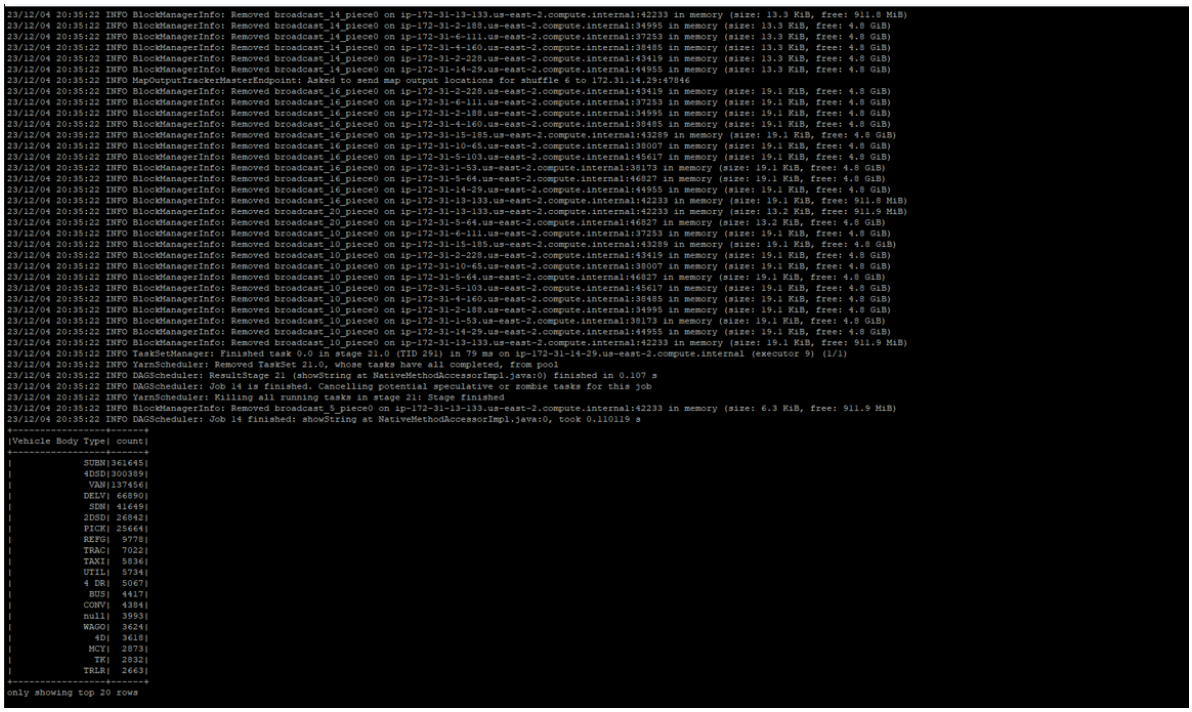


Figure 4

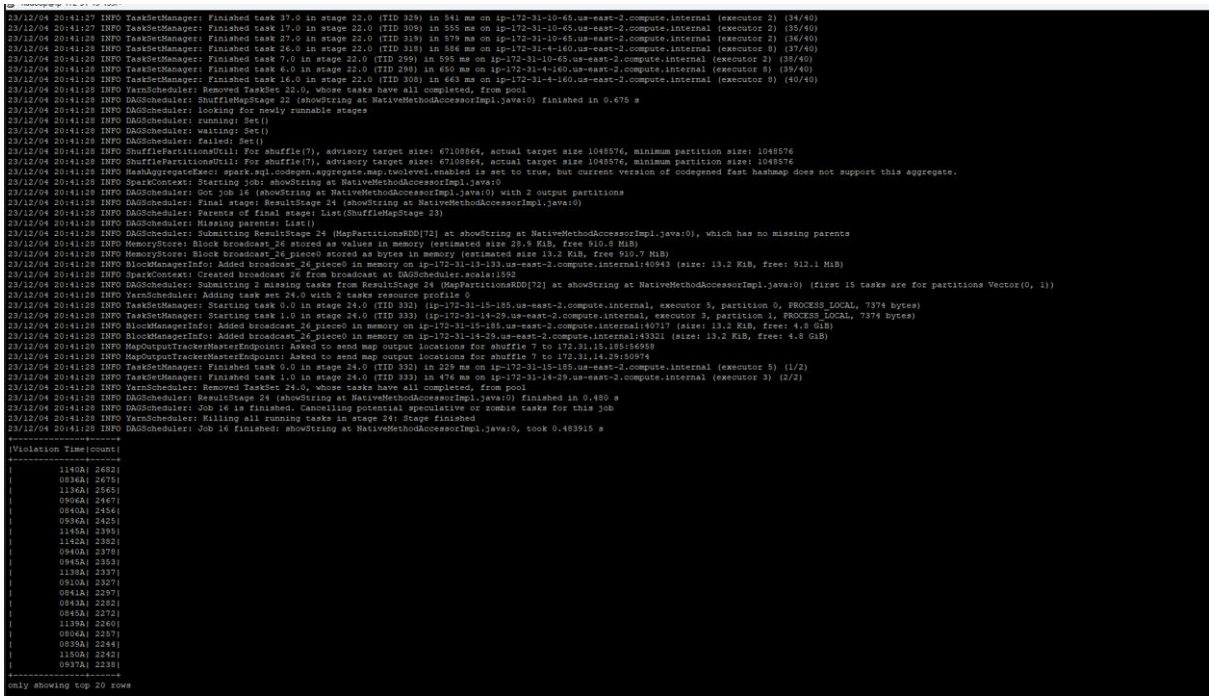


Figure 5

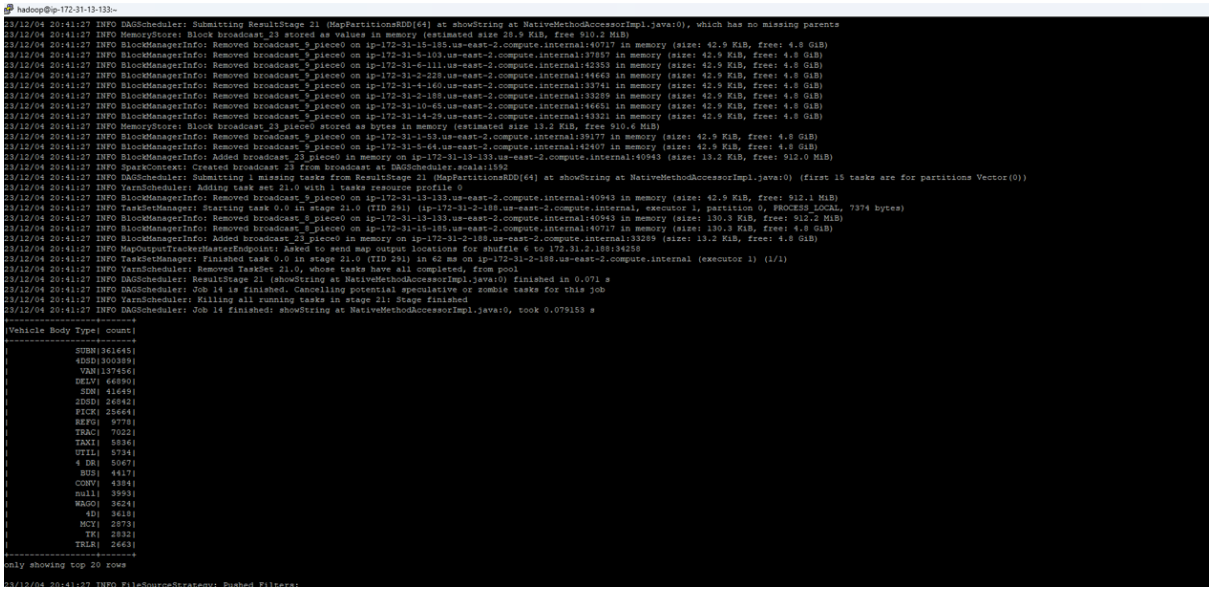


Figure 6

Monitor Hardware Instances:

- Navigate to the AWS Console to monitor hardware instances associated with the EMR cluster. This provides an overview of the infrastructure utilized during the data processing task.

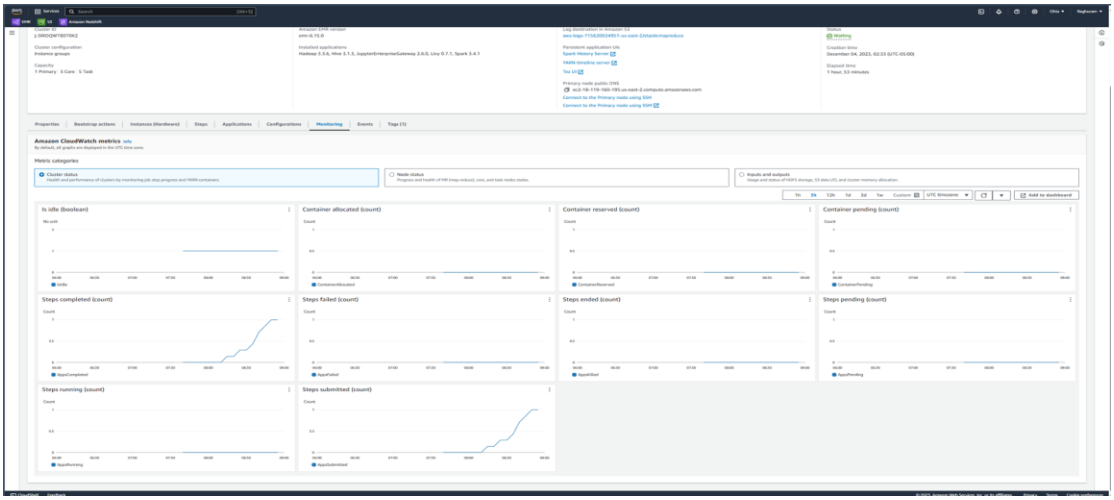


Figure 7

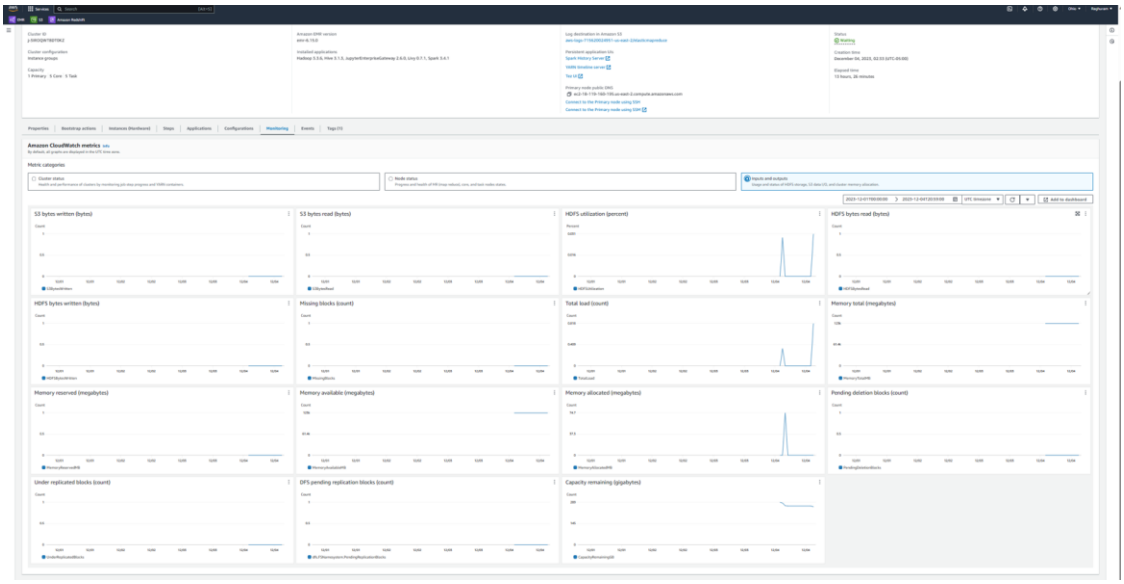


Figure 8

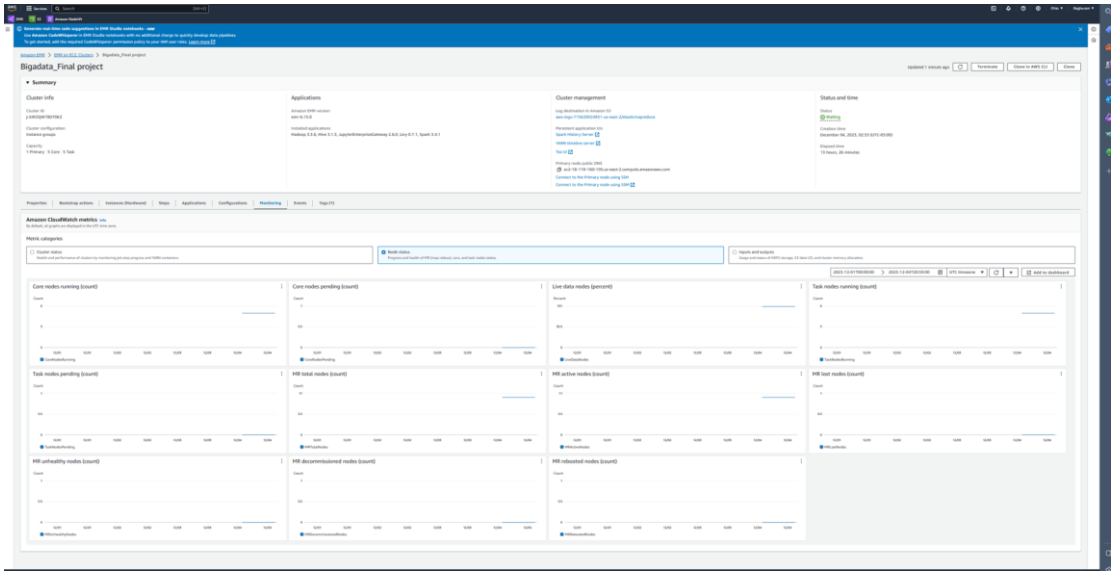


Figure 9

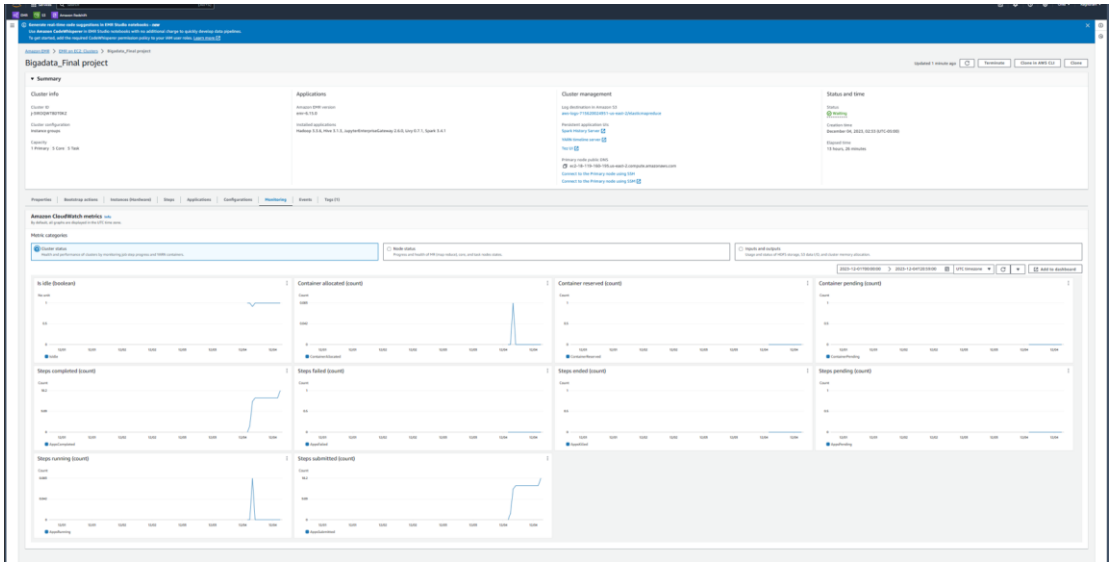


Figure 10

Conclusion:

This project successfully analyzed the NYC parking ticket data using AWS services. The initial findings offer valuable insights into parking trends and violations in the city. Further analysis can be conducted to gain deeper understanding and potentially inform policy decisions or improvements to parking infrastructure.