Taxi Demand Fare

Vandit Mehta (40232414)
Master of Applied Computer Science
Concordia University
Montreal, Canada
mehtavandit2205@gmail.com

Jaiwant Singh Mahun (40270569)
Master of Applied Computer Science
Concordia University
Montreal, Canada
jaiwantmahun2000@gmail.com

Yug Kotak (40264255)
Master of Applied Computer Science
Concordia University
Montreal, Canada
yugkotak9745@gmail.com

Umang Savla (40265305)
Master of Applied Computer Science
Concordia University
Montreal, Canada
u_savla@live.concordia.ca

Jinish Vaidya (40270454)
Master of Applied Computer Science
Concordia University
Montreal, Canada
jinishvaidya@gmail.com

***Abstract*—The "Taxi Demand Fare" project analyzes Taxi and Limousine Commission (TLC) Trip Record Data effectively by utilizing Docker containerization and Amazon Web Services technologies, S3 and Elastic Kubernetes Service. With a focus on fault tolerance and scalability, the system replicates Amazon S3 buckets to provide robustness, resolves query processing issues, and improves reliability. The dataset is arranged inside the S3 bucket and contains important information from yellow and green taxi trip records. Because of Amazon S3's superior handling of large datasets and its efficiency in same region replication, its use is justified. All things considered, the project offers a scalable and robust method for organizing and evaluating large amounts of taxi trip data.**

***Index Terms*—Taxi Demand Fare, Amazon S3, Elastic Kubernetes Service, Containerization, Scalability, Fault Tolerance, Replication, System Reliability, Same Region Replication**

I. Introduction

The "Taxi Demand Fare" project is initiative to transform the management and analysis of TLC Trip Record Data, a dataset that is essential to understanding and enhancing taxi operations. This project combines the principles of fault tolerance, scalability, and containerization. Its goal is to solve the difficulties posed by processing large volumes of taxi trip records.

The project's main contribution is the introduction of a Docker-based containerization strategy that offers an effective and modular framework for deploying and managing applications. This strategy is supported by a resilient infrastructure based on AWS S3 and EKS, which guarantees the scalability needed to manage large datasets while preserving fault tolerance via replication.

The TLC Trip Record Data contains information from both yellow and green taxi trip records. It includes important details like pick-up and drop-off times, locations, distances, fares, and passenger counts. In order to take advantage of Amazon S3's superior handling of large datasets and its efficiency in same region replication—both of which are critical for real-time transportation analytics—the project deliberately uses it for dataset storage.

The methodological journey starts with the dataset organized in an Amazon S3 bucket. To enable smooth deployment, an Amazon EKS cluster is then created. Containerization is made easier with Docker images, and data processing is optimized with Python queries. Fault tolerance is guaranteed by nodes of EKS and by replication in S3, meeting the requirement for continuous system operation even in the case of node failures.

II. Components and Roles

A. Docker

Docker encapsulates applications with their dependencies and runtime environments into isolated images for portability and consistent deployment across environments[1].

B. Amazon S3

Amazon S3 is designed to store and retrieve any amount of data offering scalability, durability, secure access control, and cost-effective storage for various data types[2].

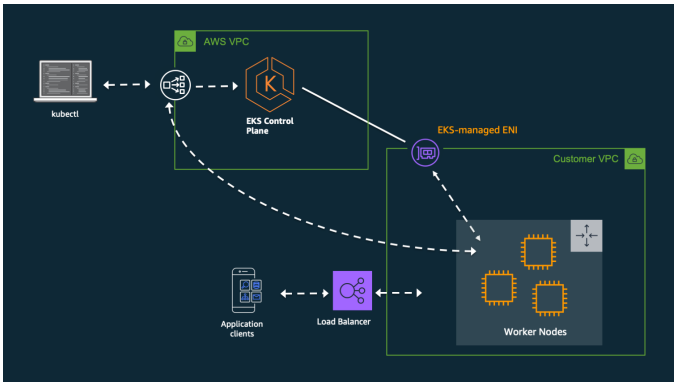

Fig. 1. Amazon EKS Architecture

C. Amazon ECR

Amazon ECR provides a secure and private registry within AWS for storing and managing Docker images used within the EKS cluster, ensuring version control and secure access[3].

D. Amazon EKS

Amazon EKS offers a managed Kubernetes service for orchestrating and scaling containerized applications within EKS clusters. Simplifies Kubernetes deployment, automates scaling, and provides high availability and fault tolerance[4].

E. Kubectl

Kubectl provides a command-line tool for interacting with Kubernetes clusters. Developers and operations teams can use kubectl to manage deployments, pods, services, and other Kubernetes resources, simplifying application management within the EKS cluster[5].

III. Dataset Description

Dataset Title: NYC Taxi Trip Record Data (Yellow, Green, and FHV)

Source: NYC Taxi and Limousine Commission (TLC)[6]. Data originally collected by technology providers authorized under the Taxicab and Livery Passenger Enhancement Programs (TPEP/LPEP) for yellow and green taxis, and by FHV bases for their vehicles.

Description: This dataset captures trip information for yellow, green, and for-hire vehicles (FHVs) operating in New York City. Each record includes the following fields (adapted from [6]):

- Pick-up and drop-off dates/times: Formatted in YYYY-MM-DD HH:MM:SS (inspired by IEEE data format standard).
- Pick-up and drop-off locations: Latitude and longitude coordinates, along with taxi zone IDs.
- Trip distances: Calculated in miles.
- Itemized fares: Breakdown of fare components like metered fare, tolls, and surcharges (inspired by [6]).
- Rate types: Standard, Night, Weekend, etc.
- Payment types: Cash, credit card, etc.
- Driver-reported passenger counts: Integer values.
- Dispatching base license number (FHVs only): Unique identifier for the FHV base.

Data Collection: The TLC received the data from authorized technology providers (yellow/green taxis) and FHV bases, who are responsible for its accuracy[6].

Accuracy Disclaimer: The TLC emphasizes it did not create the data and makes no warranty regarding its accuracy or completeness, particularly for FHV trips[6].

Format: Publicly available in CSV format on Kaggle:[7].

IV. Work Flow

Fig. 2. Layered Diagram

This layered architecture depicts the combined workflow of AWS S3, Docker, EKS, ECR, and Kubectl for deploying and managing containerized applications in the cloud. Each layer represents a distinct stage in the process:

A. Developers and Docker

Role: Developers build and test containerized applications using Docker, encapsulating code, runtime environments, and dependencies within isolated images[8].

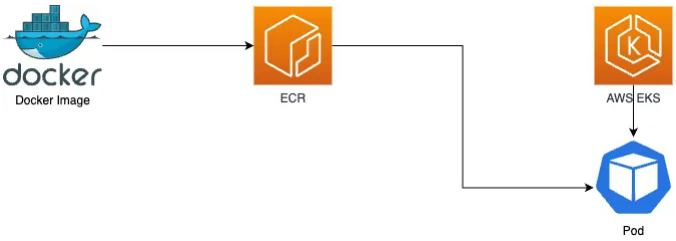

Fig. 3. Application Workflow

B. Docker Image Build and Push

Role: Docker images are built and pushed to Amazon ECR (Elastic Container Registry), a secure and private registry within AWS for version control and secure storage[9].

C. Amazon ECR (Private Registry)

Role: Stores and manages Docker images used within the EKS cluster, enabling secure access control and versioning[9].

D. Amazon S3 (Storage)

Role: Acts as a central repository for applications offering scalability, durability, secure access control, and cost-effective storage for various data types[10].

E. Amazon EKS (Cluster)

Role: Provides a managed Kubernetes service for orchestrating and scaling containerized applications within clusters. Offers simplified Kubernetes deployment, automated scaling, high availability, and fault tolerance[11].

F. Containerized Application (Pods)

Role: Containerized applications run within pods, managed by Kubernetes within the EKS cluster. Offers portability, isolation, and efficient resource utilization.

G. Application Data and Logs

Role: Application data can be stored in Amazon S3 and accessed by pods through object storage APIs or mounted volumes. Offers scalable storage and efficient access for application data and logs.

V. Methodology

The "Taxi Demand Fare" project's methodology emphasizes the use of containerization, scalability, and fault tolerance in order to manage TLC Trip Record Data in an organized and planned manner. The main elements of the project's methodology are outlined in the phases that follow:

1) Storing data in S3:
   - An Amazon S3 bucket contains the TLC Trip Record Data, which includes yellow and green taxi trip records.
   - Kept historical data versions for auditing purposes by utilizing S3's versioning functionality.
   - S3 replication was put into place to automatically replicate data to a different bucket for disaster recovery
2) Cluster Creation and Configuration
   a) EKS Cluster
      - Established an Amazon EKS cluster with the required quantity of worker nodes.
      - Set up the auto-scaling group of the cluster to automatically change the number of nodes according to resource usage.
      - S3 and other AWS services can be accessed securely with enabled IAM roles and service accounts.
   b) Network Configuration
      - The EKS cluster was given its own VPC(Virtual Private Cloud), to keep it apart from other resources.
      - Network policies and security groups were configured to limit access to the cluster and data.
3) Script Development and Containerization
   a) Development of Python Scripts
      - Developed Python script containing SQL-like queries for analyzing taxi demand and fare data.
   b) Docker Image Generation
      - Created a Docker image with all required libraries and dependencies for a Python script.
4) Deployment and Execution
   a) Docker Container Deployment
      - Containerized the Docker images and deployed them to the EKS cluster.
   b) Data Processing
      - Jobs that were sent to the EKS cluster caused the containerized scripts to start analyzing the data across nodes in parallel.

VI. Conclusion

The "Taxi Demand Fare" project effectively manages and analyzes large amounts of taxi trip data, showcasing

the possibilities of containerization, scalability, replication, and fault tolerance principles. The project uses the NYC Taxi and Limousine Commission Trip Record Data in an effective manner by utilizing Docker, AWS S3, and AWS EKS.

The project's main advantages are as follows:

- Containerization: Making use of Docker facilitates modular deployment and streamlines EKS cluster application management.
- Scalability: The project easily grows to meet data growth and processing demands because it is based on AWS EKS.
- Fault Tolerance: Auto-scaling capabilities in EKS and replication in S3 provide system dependability and uninterrupted operation even in the event of node failures.

VII. Future Enhancements

Though the project successfully handles the difficulties it faces right now, ongoing development work can further expand its potential:

- Apply analysis to additional datasets: Incorporate supplementary data sources such as climatic trends, events, or traffic circumstances to acquire more profound understanding of the variables impacting taxi demand and rates.
- Creation of Predictive models: Create more complex machine learning models to more accurately forecast future fee patterns and taxi demand, allowing for more intelligent resource allocation and dynamic pricing schemes.
- Visualization and exploration: To provide an intuitive user interface for examining the wealth of information uncovered by the study, use interactive dashboards and data visualization tools.
- Real-time application: Examine the viability of real-time fare and taxi demand prediction systems to guide passenger matching and dynamic route optimization for increased effectiveness and satisfaction of customer.
- Impact on society and economy: Evaluate the wider social and economic impacts of taxi demand and fare trends, since this may help guide legislative choices and urban planning projects.

References

[1] Merkel, D. (2014). Docker: Lightweight Linux Containers for Developers, Operators, and Cloud. IEEE Cloud Computing, 1(2), 5-14.

[2] Amazon S3 website user guide https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

[3] Amazon Web Services. (2023). Amazon Elastic Container Registry (ECR) Developer Guide. Retrieved from https://docs.aws.amazon.com/AmazonECR/latest/userguide/Registries.html: https://docs.aws.amazon.com/AmazonECR/latest/userguide/Registries.html

[4] Kumar, P., Seth, A., and Zhou, J. (2019). Amazon EKS: An Overview of a Managed Container Orchestration Service. Proceedings of the 2019 IEEE International Conference on Cloud Computing (ICCC), pp. 349-356.

[5] Kubernetes Authors. (2023). kubectl. Retrieved from https://kubernetes.io/docs/reference/kubectl/: https://kubernetes.io/docs/reference/kubectl/

[6] NYC Taxi and Limousine Commission: https://www.nyc.gov/site/tlc/index.page

[7] NYC Taxi Trip Record Data on Kaggle: https://www.kaggle.com/datasets/sohaibanwaar1203/taxidemandfarepredictiondataset?select=yellow_tripdata_2015-03.csv

[8] Merkel, D. (2014). Docker: Lightweight Linux Containers for Developers, Operators, and Cloud. IEEE Cloud Computing, 1(2), 5-14.

[9] Amazon Web Services. (2023). Amazon Elastic Container Registry (ECR) Developer Guide. Retrieved from https://docs.aws.amazon.com/AmazonECR/latest/userguide/Registries.html

[10] Dilley, J., Murillo, H., and Toutant, K. (2004). Amazon S3: A Scalable, High-Performance Data Storage Service. ACM SIGCOMM Computer Communication Review, 34(5), 307-317.

[11] Kumar, P., Seth, A., and Zhou, J. (2019). Amazon EKS: An Overview of a Managed Container Orchestration Service. Proceedings of the 2019 IEEE International Conference on Cloud Computing (ICCC), pp. 349-356.