# Architecture Design and Visualization Report

## **Objectives**

Design and implement a prototype data analysis platform that uses the New York City Taxi and Limousine Commission (TLC) Trip Record Data from AWS public datasets as source data(https://registry.opendata.aws/nyc-tlc-trip-records-pds/). The specific requirements of the data analysis platform are as follows:

* This platform needs to load source data from AWS S3, then perform ETL to clean the data and store them back to S3; finally, generate data visualization results.
* This platform need to give some statistical values, such as: travel distance, travel time, average speed, hot spots in Manhattan, trends analysis for taxi loads.
* This platform need to give a common trends analysis of such data and any relevant data you think is important to illustrate.
* This platform need to give the user some input boxes for them to define the search criteria, such as latitude/longitude boxes, time ranges, long distance (>15 miles) and etc.

## **Constraints and Assumptions**

* Due to some objective conditions, the data analysis platform can only be deployed on 3 virtual machines (named hadoop1, hadoop2, hadoop3), which run on one personal computer. And because the source code involves the configuration parameters of virtual machines, the jar package can only run on the virtual machine configured with the same parameters.
* During the platform Demo development process, I tried to write data back to S3, but I did not have an S3 data storage service account, the S3 server responded "Access Denied", so I only stored the ETL processed results in the ElasticSearch and the Hive (Based on HDFS).
* The requirements document mentions "perform streaming ETL and analyst", based on which it is inferred that the customer needs to perform ETL in real-time, so I only designed and implemented real-time data analysis framework, and the offline data analysis part can also be designed and implemented in the future.

## **Architecture**

The source data of the data analysis platform is in the AWS S3. The platform uses Hadoop as a distributed basic framework, the data transmission part adopts Kafka as the distributed message queue, Spark Streaming as the real-time computing engine, and the ETL processed data is stored in ElasticSearch to be counted, and finally Kibana visualizes the data. The platform architecture is shown in the following figures.

### Data flow

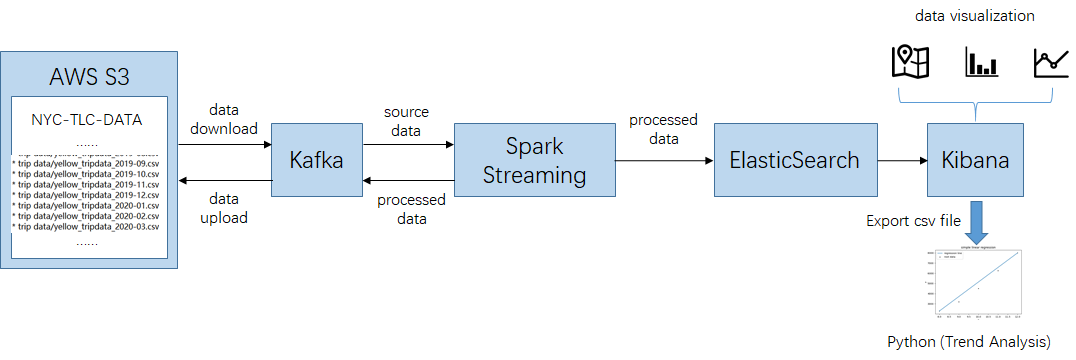


Figure 1

### Cluster framework

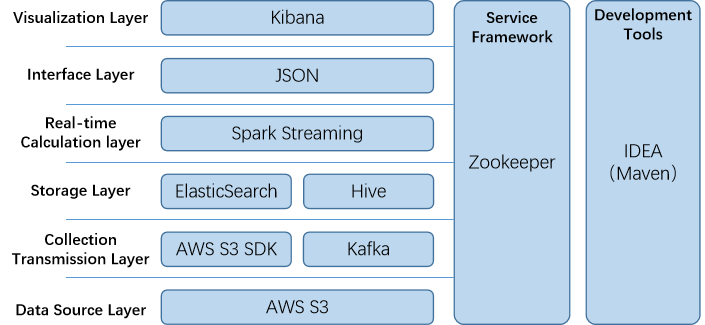


Figure 2

### Cluster deployment

The platform is deployed on 3 virtual machines, which have the same specifications.Deploy cluster need to notice the following rules: Don’t install the NameNode and SecondaryNameNode on the same server. ResourceManager also consumes lot of memory, so do not install it on the same machine as NameNode and SecondaryNameNode. Spark uses YARN mode, the YARN component of Hadoop performs resource and task scheduling. (Participate in the appendix A)

Table 1

|  |  |  |  |
| --- | --- | --- | --- |
| Host Name | hadoop102 | hadoop103 | hadoop104 |
| Cluster Framework | NameNode |  | SecondaryNameNode |
| DataNode | DataNode | DataNode |
|  | ResourceManager |  |
| NodeManager | NodeManager | NodeManager |
| Zookeeper | Zookeeper | Zookeeper |
| Kafka | Kafka | Kafka |
| Spark | Spark | Spark |
| ElasticSearch | ElasticSearch | ElasticSearch |
| Kibana |  |  |
| Hive |  |  |
| http-server\* |  |  |

\*The http-server is used to construct the regional hot spots map in kibana

## **Design Ideas**

Briefly introduce the key idea points in the process of platform design and implementation:

### Operate S3 object

Before downloading an S3 object, need to know the region name, the bucket name, and the key name of the object, but I didn’t find the key name on the page: https://registry.opendata.aws/nyc-tlc-trip-records-pds/, so I used the listObjects method returns an ObjectListing object that can list the key name of the objects[1].

I got the object’s contents by calling getObjectContent, this returns an S3ObjectInputStream that behaves as a standard Java InputStream object, this can perform streaming data download.

I didn’t upload the results to S3 due to permission issues, but I also searched for information and learned that streaming upload can be achieved using “multipart uploading”[2].

### Operate Kafka

Spark Streaming uses the Direct mode when reading Kafka data, by using the createDirectStream method. The Direct mode periodically query Kafka to obtain the latest offset of each topic+partition, thereby defining the range of offset for each batch. When a job that processes data is started, Kafka's simple consumer API will be used to obtain data in the specified offset range of Kafka.

### Data ETL in Spark Streaming

Each line of the source data got from S3 includes 18 attributes, and each line of the ETL processed result data includes 8 attributes (vendorID, puDT, doDT, diffDT, distance, avgSpeed, puLID, doLID). The ETL processing is divided into two stages: the first stage deletes the header row and attributes that have nothing to do with the final result (such as passenger\_count, fare\_amount, extra, etc.), and the second stage performs data structure conversion and calculates the data of the new attributes. This part is written in Scala, using map, filter, foreach and other RDD operators.

### Batch write ElasticSearch

Spark Streaming has some ways to write processed data to ElasticSearch. I used the method of writing Json strings to ElasticSearch, and introduced the Fastjson library (using this library can convert Java objects to Json format, or convert Json strings to Java objects). Before writing data, I used Kibana to create an index template in ES, and use Scala to write a Case Class. At first I tried to use JestClient to write one line at one time, and later I changed to use Bulk object batch writing to improve writing efficiency.

### Visualize data with Kibana

In the data visualization stage, I used Vertical Bar, Line and Region Map to display the data processing results. The following briefly introduces the generation of Manhattan Regional Hot Spots Map[3]: The first step is to find the geojson file that contains the area division of Manhattan taxis. I found this file on Github[4]. The second step, I need to do here is to serve the geojson file from a web server, we used a little tool from npm, http-server, and ran command from the same directory that the geojson file lives in. The third step, I need to configure some custom settings in kibana.yml file to associate Kibana and geojson files. The last step is to use Kibana UI to generate a regional heat map.

### Linear regression for trend analysis

The platform counts the trading volume of taxi trips within a certain time (count by hour), and visualizes the data. By observing the visualized data for 3 consecutive days, it is found that the change of taxi trading volume in a period of time has a linear relationship with the time series, so the linear regression algorithm is used to analyze the trend of taxi loads. I use Python to process the csv file exported by Kibana, use the “Least Square” algorithm to fit a line, and compares the result with the result of the sklearn library. The algorithm of the least squares method to fit a straight line is as follows:

1. The known points set: (x1, y1)……(xn, yn)
2. Let the linear equation be: y = kx + b
3. Calculate the sum of squared errors from all points to the straight line:
4. According to the extremum theorem, the error equation takes the extremum when the first derivative is equal to 0, so it is differentiated with respect to k and b:
5. Calculate k, b value, let:

The formula in the step ④ is transformed into

## **Data Visualization**

Appendix B Manhattan Regional Hot Spots Map

Appendix C Trip Distance Statistics

Appendix D Trip Average Speed Statistics

Appendix E Pickup Time Statistics

Appendix F Dashboard

Appendix F Trend Analysis for Taxi Loads

## **Reference**

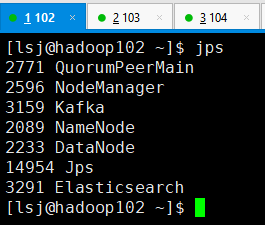
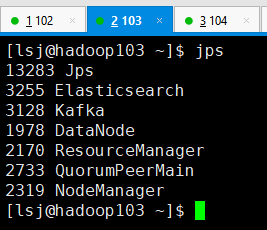
[1]<https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/examples-s3-objects.html>

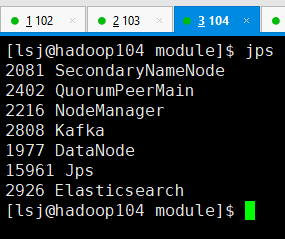
[2]<https://docs.aws.amazon.com/AmazonS3/latest/userguide/mpuoverview.html>

[3]<https://www.elastic.co/cn/blog/custom-region-maps-in-kibana-6-0>

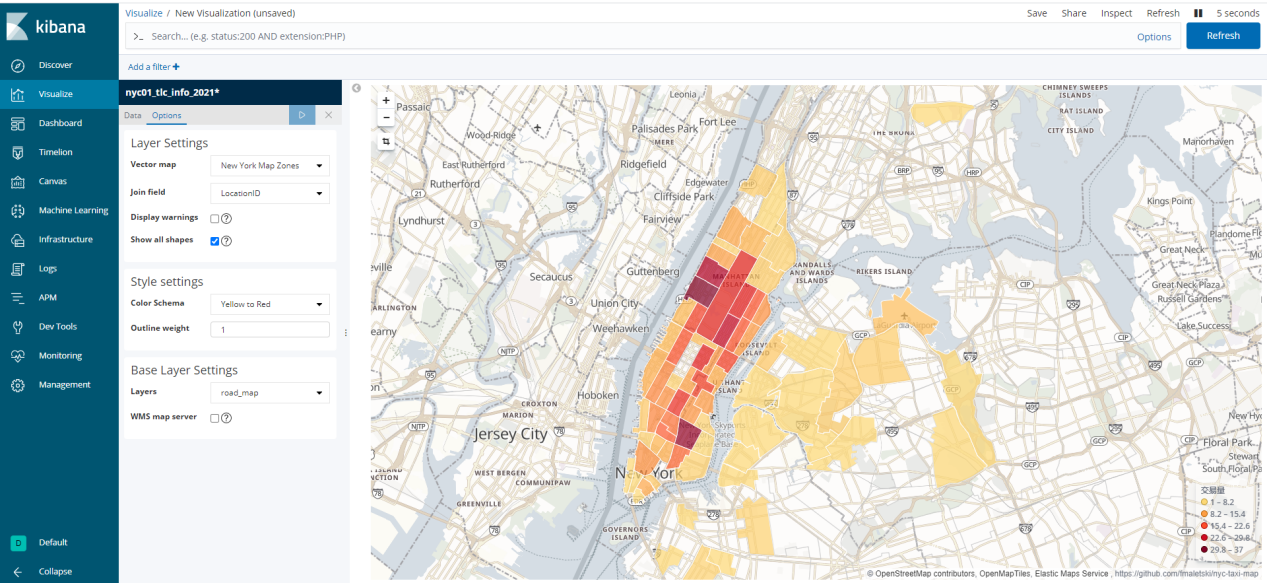
[4]<https://github.com/fmaletski/nyc-taxi-map>

## **Appendix A Cluster Deployment**

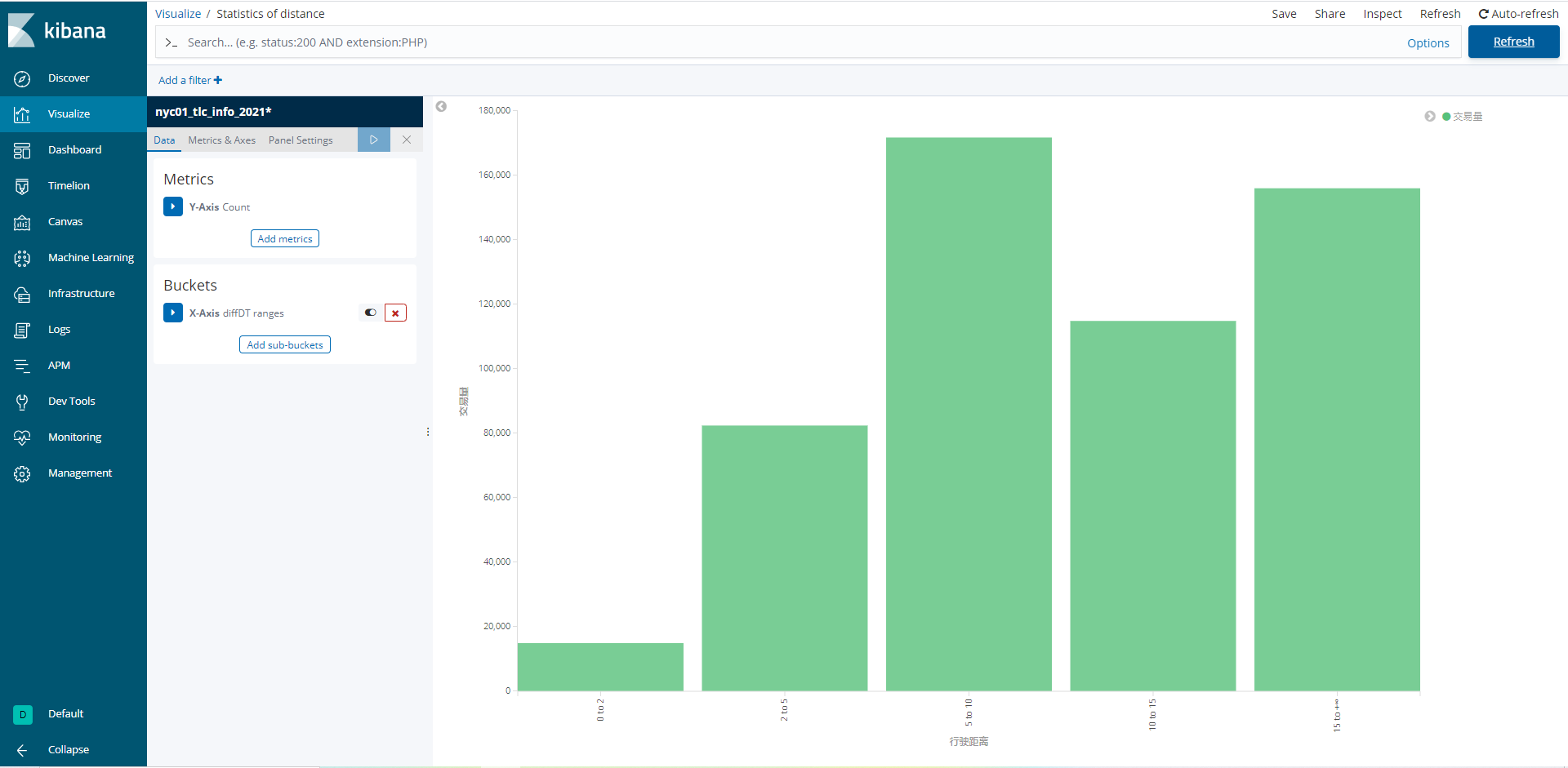
 



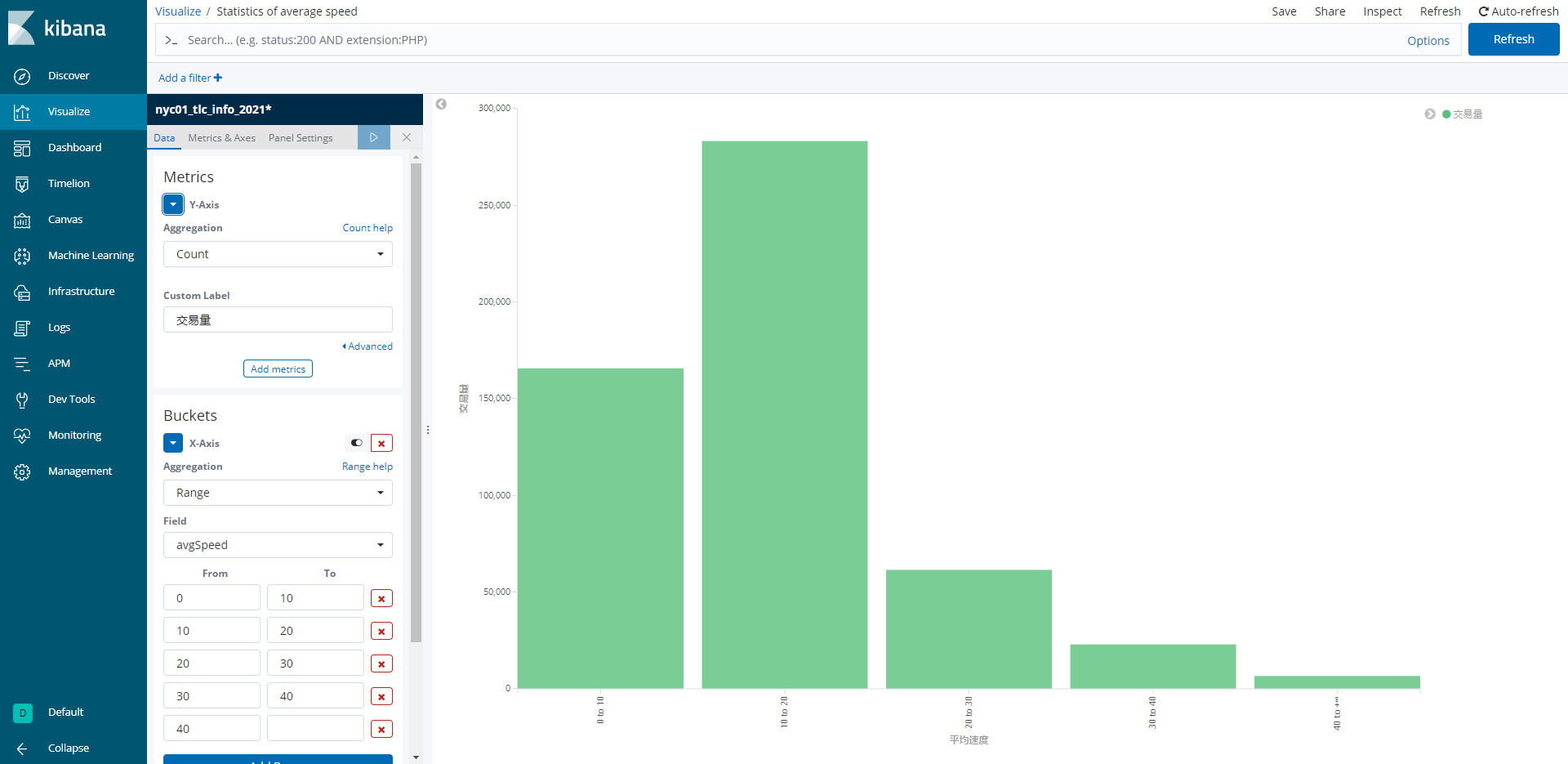
## **Appendix B Manhattan Regional Hot Spots Map**



## **Appendix C Trip Distance Statistics**



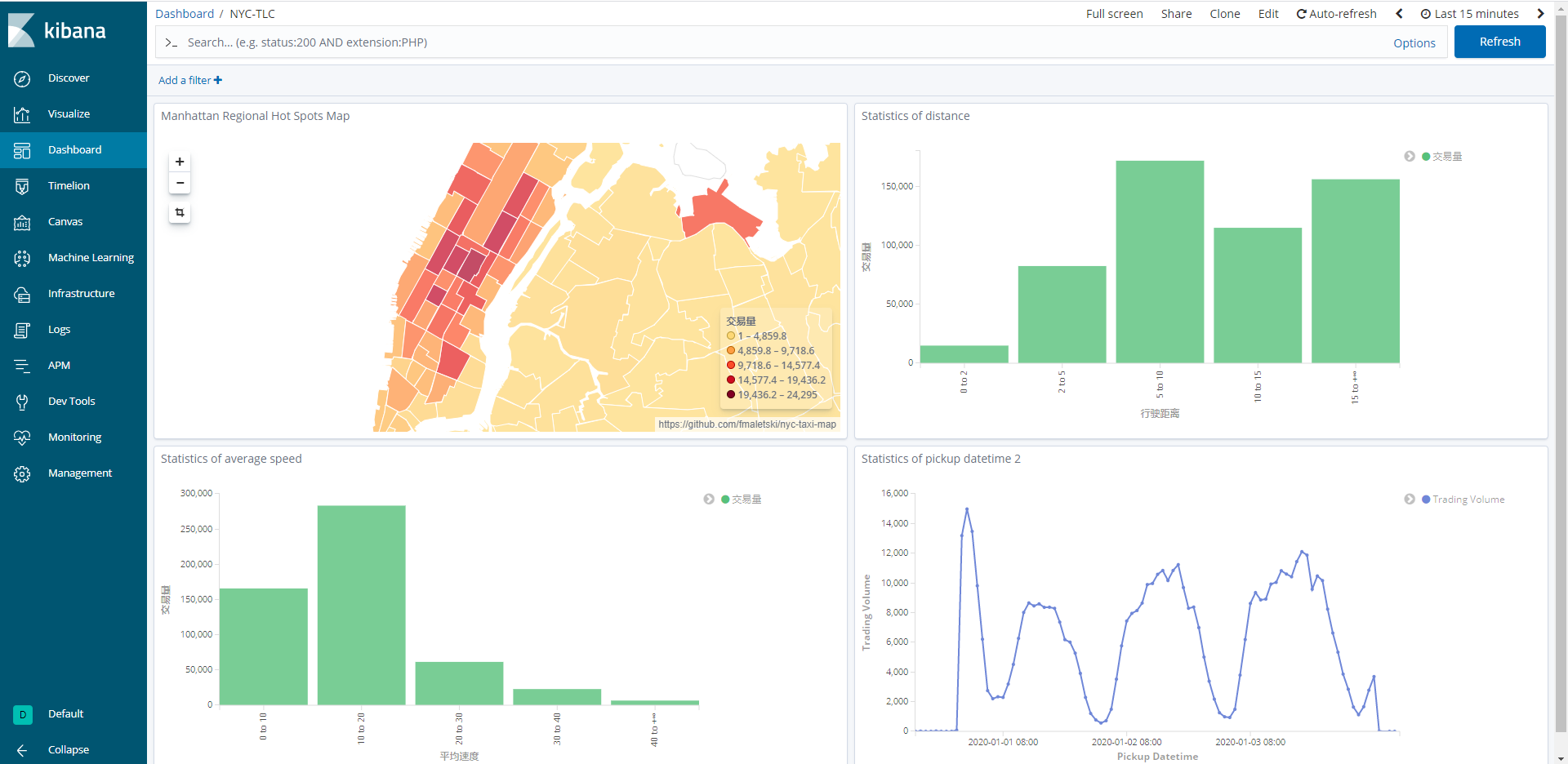
## **Appendix D Trip Average Speed Statistics**



## **Appendix E Pickup Time Statistics**



## **Appendix F Dashboard**



## **Appendix G Trend Analysis for Taxi Loads**

