Project Title: **Stock Open Price Prediction as a Service**

Team Members:
Anusha Basavaraja
Suresh Nayak

Project Goals:

The goal of this project is to build a Neural Networks model to predict the open price of a stock for the next day given the stock name and open price of current day. We used a NN model with the help of Keras package which in turn uses Tensor flow module to build this predictive model.

Components:

- ***Kubernetes cluster***: We deployed our service using kubernetes cluster which helped us in scaling our model as per the need.
- ***RabbitMQ***: We implemented message queues to trigger the worker nodes and also to log debug information pertaining our service.
- ***Redis***: We used the Redis Set database as a key-value store to store the results from our NN model (worker) after predicting the stock open prices. The output for each run consists of stock name, run date and predicted stock open price.
- ***REST API***: We built REST APIs as a front-end interface for our service which responds to REST Client requests.

Description:

The REST client makes a request which are basically REST API calls. We are using Thunder Client to make REST calls. Once the request is made, the REST server searches the Redis database if the result is available for the request made. If yes, then it fetches the result from Redis database else, it logs the request into RabbitMQ message queue which in turn triggers the worker to predict the stock open price for the next day given the stock name and current date open price. Once the prediction is made for given data, the model will load the results into Redis database which will be fetched by the REST server and returns it as a response to the client. We followed this mechanism because of Lab7 which made us get familiar with all these systems and their integration thereby boosting our confidence in using them in our project work.

Architecture Diagram:

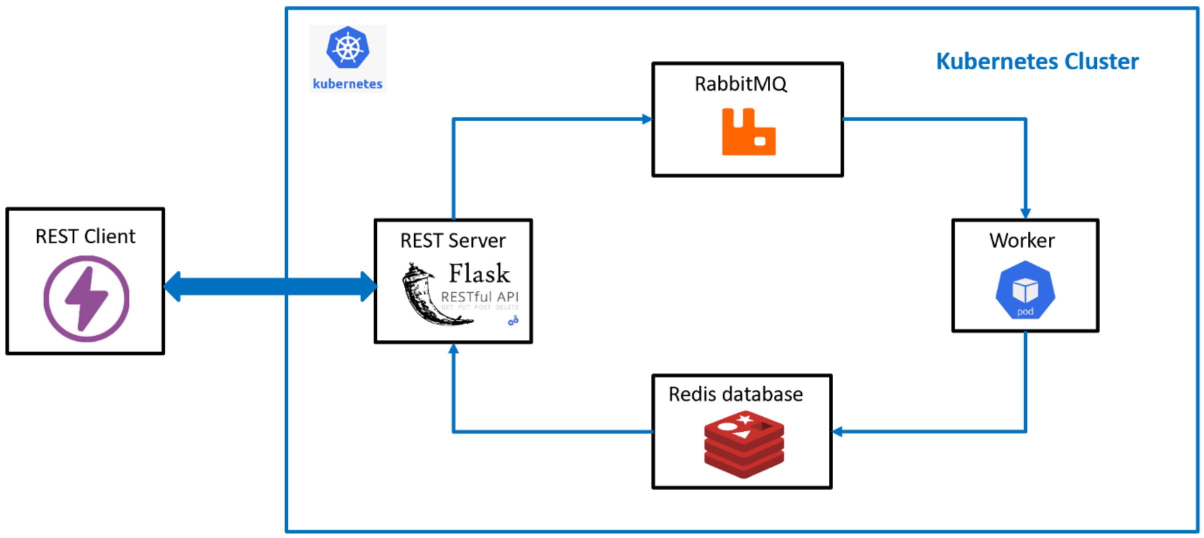

Fig: Architecture diagram of Stock Open price Prediction as a Service

Debugging:

While building the worker, we had some hard time understanding the end-to-end working of the model. So, we had to explore more on python packages and modules and their execution before deploying and building the Docker image for the Worker.

We faced some hard time in terms of matching the versions of different modules with python while building the Docker images and launching the Kubernetes cluster. In order to identify the actual error, we had to do lots of research with respect to the compatibility of modules with python version we were trying to use in the Worker while installing the Tensor flow and Keras packages by checking the logs of the Worker.

We also faced some issues when we tried to communicate with the Kubernetes cluster after applying the ingress. So, in order to understand the typical working of Kubernetes and a way to communicate with the cluster, we built and executed a simple REST server from Lab6 by modifying and deploying it in Kubernetes cluster. Using this we tried to understand the bits and pieces of working and establishing communications with the Kubernetes cluster. Once it was successful, then we used the same approach applicable to our end-to-end project.

The logs information that we used for debugging also helped us track our code in REST and Worker for the smooth execution after deploying our project in Kubernetes cluster.

System Working:

The first and foremost part in our project is the data input that the model receives from the REST client. This data includes stock name and open price of the stock during present date. Once these two values are sent using REST Client, the REST server uses this data to check whether the system has already executed the run for given data by searching the results in Redis database. If the results exist for the given data, then the REST server simple fetches the result from Redis database as a response for the Client. Otherwise, if the results are not present in the database for given data, then REST server logs the request into the RabbitMQ message queue to be processed by the worker.

Once the request is logged into the message queue, it triggers the Worker by picking the right training data for the given stock name. the value stock name from client acts as key for the Worker in choosing the correct training data for processing the logged request. Once the right training data is chosen, using the open price of the given stock of present date, the Worker will use NN model to predict the open price of next day. This predicted value will be then uploaded into Redis database along with the current date as result for given request. We are using Redis set to store the results.

During the course of each run, the information pertaining the progress of each stage will be logged into RabbitMQ message queues as debugging information which helps us to keep track of the execution. The result from the Redis database will be fetched by the REST server as response to the Client. This response consists of stock name, current date and predicted stock open price for the next day.

For this project, we have used three types of training data – i.e., NCS Stock data of Infosys, TCS and State Bank of India (SBIN). Based on the stock name in the request data, the corresponding training data will be pulled by the Worker to process the request. If we provide anything other than these three stock names in the request, the model will respond to the client to use right stock name in the request i.e., SBIN/INFY/TCS.

Limitations and Future work:

The future work for this project work includes extending this approach to a greater number of stocks. The current limitation of this project is manual triggering of the process by requesting the model via REST Client. In the future, this could be improvised by either scheduling the process at a particular time everyday or make it ad-hoc by incorporating more automation into this process by monitoring the NCS website to check if needed data is available for the run. Once that field is updated in the website, it could further trigger the process and update the results in the database which could be accessed by the customers either by designing or modifying a website or sending this information using subscription mechanism by replenishing the training data available for the model for each day.