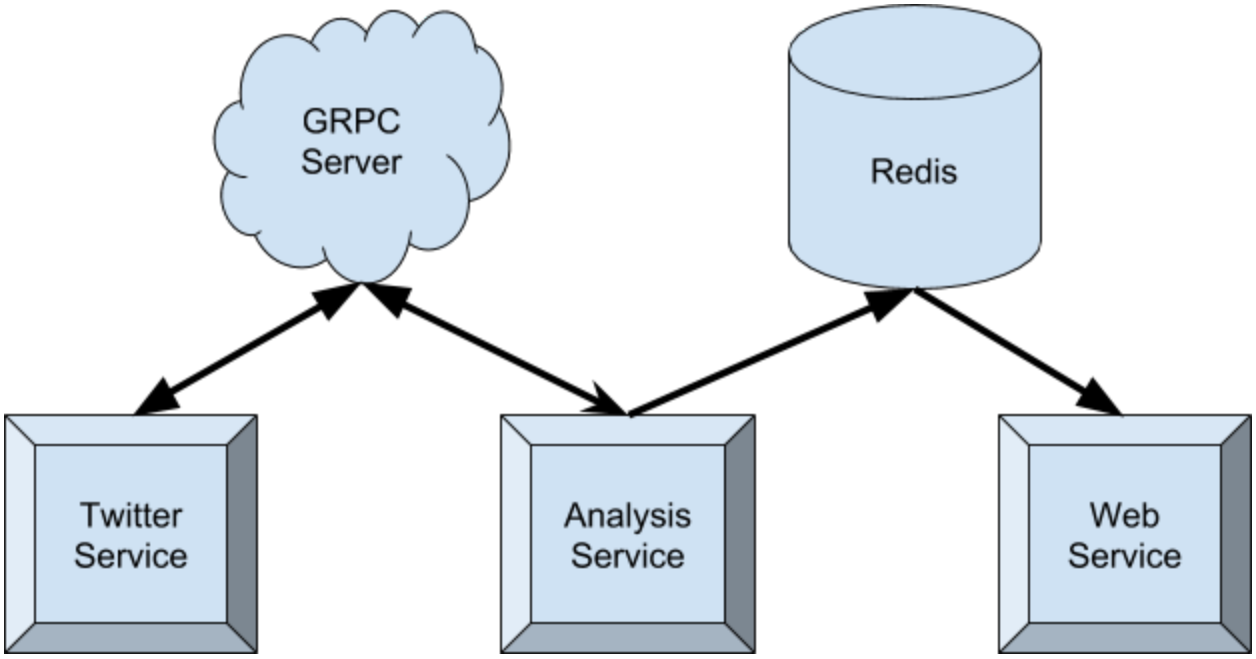
SOFT8026 Data Driven Microservices

Assignment 2

Name: Vilius Valiusis

Part 1

Microservice Architecture



Twitter Service:

The service acts both as a twitter client and and the GRPC server. The twitter client is a third party solution called go-twitter. For our use case, a continuous stream of tweets is received by the service. There these tweets are filtered via a keyword, such as '*dog*'. Then these tweets are take and submitted to the GRPC server via a stub to be consumed by any of the clients that are currently connected.

Analysis Service:

This service acts as an intermediary, where the analysis of tweets occurs. The service uses the GRPC client to connect to the server that is continuously transmitting tweets. Each received tweet goes through the 3rd party sentiment analysis package called sentiment by cdipaolo. The output of this process is a score of either 0 or 1, representing bad or good respectively.

Once the analysis is complete a timestamp is taken. This timestamp is used as a key and the score as its value in the key/value in the Redis in-memory data structure store.

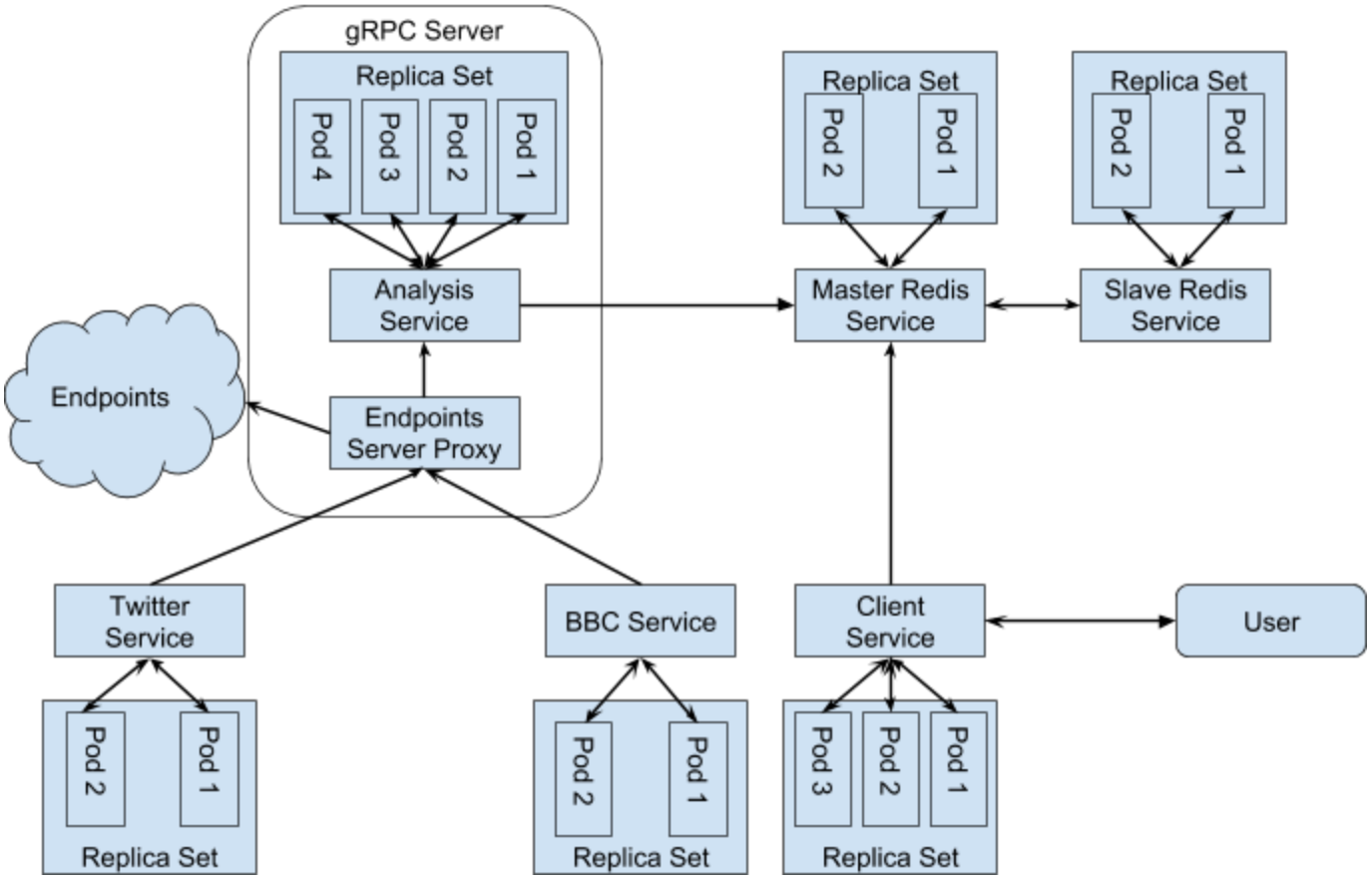
Web Service:

The web service mainly does two things: it process all data submitted in the last 60 seconds to the Redis database and it acts a the http server for anyone to connect.
The goal of processing is to calculate the average score of tweets in the last 60 seconds. This is accomplished via the use of timestamps in the format of seconds past from January 1970. However, this acts as a limiting factor as the maximum number of tweets that can be process is capped to 60, on the contrary this minimizes the number of calculation and requests required to 60. This is intentional, if for example, milliseconds were used as timestamps it would require 60,000 request to the redis database.

This results is take and printed to by the http server to the route /sentiment/. The web service can be accessed via **localhost:8080/sentiment/**.

Part 2

Microservice Architecture



Code Changes

A new bbc service was added. The service uses newapi.org API to read in all the news headlines and send them to the analysis service.

The analysis service now accepts two sources of connections from bbc and twitter services. Due to the affect of having two types of services the way data is stored in redis has changed. Now each entry in redis contains a key: ‘timestamp value’ and value: messages[]. Each message in the array has two values. First value is a score: 0 or 1, second values is DataSource: bbc or twitter. An example message **key:**1526762720**, value:**[(1,twitter),(0,twitter),(1,bbc)] etc.. Each submission to the redis service happens every 1 seconds. A submission can have **n** number of values in the values array.

Redis now store all values for 24 hours before they expire.

The client services now can count the average score for the bbc and twitter services.

Microservice Architecture Changes

Twitter and BBC services both contain clusters of two pods. Because these services only do a simple timed task it is enough to keep the pod count low.

The analysis services is one of the two critical steps within the system, therefore, it requires a higher number of pods to ensure the stability of the system.

The one that receives the most load is the redis service. The service must be able to handle as many connection as the client receives. Each of these connections requiring a read every 1 seconds, exponentially increasing the load on the service.

To solve this the redis service was split into two a master and slave, as suggested here. This increases the number of reads it can handle and also works as a backup in case one of the services completely fail.

The client service is the only one exposed via the Load Balancer. It receives all public connections making in esential to be alway online. Therefore, three pods were used help alleviate this problem.

The architecture is live on google cloud http://35.204.156.175:8080/sentiment/