ID-2221 Project Report

Jón Rúnar Baldvinsson, Leifur Daníel Sigurdarson

October 2020

1 Data

The data we will use will be taken from the COVID19 API (Link). The api returns data where each entry represents infection information for each country for a given day, such as confirmed cases and deaths so far. Though the data is pretty solid, it still has some flaws. The cumulative confirmed cases sometimes goes down between dates, probably due to a country publishing corrections in the data, which sometimes made our data processing more difficult. Some data from the united states was also unavailable, so it is missing for some of our calculations.

2 Method

The data is fetched from the COVID19 API using a python HTTP Client. The data is fetched incrementally, one country at a time, and the response is streamed using Kafka . Using the KafkaService.py file we streamed the data to 3 topics, *CountryInfo*, *CountryCases* and *ProvinceCases*. The topics were then consumed in different Scala files.

CountryInfo We streamed some basic info about each country to the *CountryInfo topic*, such as population, covid death rate, total confirmed cases and total confirmed deaths.

CountryCases To the *CountryCases* topic we streamed the whole case history of that country, including confirmed cases, active cases, deaths and the date. All of this data was cumulative so we needed to process to get the values we wanted. To do that we used MapWithState. We saved the last 14 days in state and used that to calculate the number of confirmed cases and deaths in the last 14 days and new daily cases.

ProvinceCases To the *ProvinceCases* topic we streamed the whole case history of each province available, including latitude, longitude, confirmed cases, active cases. deaths and the date. All of this data was cumulative here as well

so we used MapWithState, same as in Country Cases. We were only interested in the day to day values so we only saved the last dates values. We then used a second MapWithState function to calculate the number of new cases of each week in the year, starting from week 4.

The data is then all saved to Cassandra, and from there the server queries the data. The Client then fetches the data from the server and publishes it. The dataflow can be seen in Figure 1

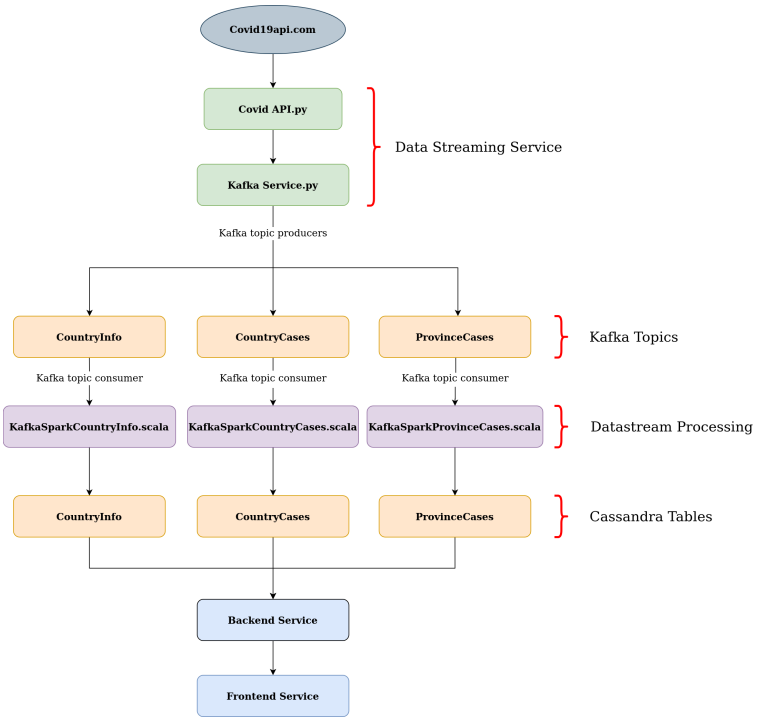


Figure 1: Dataflow

3 Results

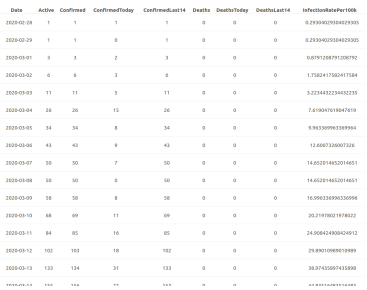
The results of our project can be seen on the website provided. On the frontpage you can see two heatmaps (Figure 4). The heatmaps are of the confirmed cases and deaths for each week of the year. You can switch between weeks to see the development of the pandemic. The data the heatmaps are showing is the data we processed in the *ProvinceCases* topic. On the frontpage there is also a table (Figure 2a). That table shows the data from the *CountryInfo* topic. If you click on a country you can see more information about that country. (Figure 2b, **??**). That is the data from the *CountryCases* topic.

4 How to run

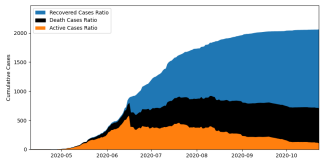
Kafka Topic Producers: To stream to the Kafka topics, you first have to run the Kafka zookeeper and the Kafka server. Now, to stream the data, run *KafkaService.py*.
Data processing and Cassandra: To consume to Kafka topics for processing, and ultimately saving to Cassandra, one has to start the Cassandra service. Now, you can run the .scala files for each respective topic processing. The data should now be saved in the Cassandra keyspace *covid*.
Backend: To run the backend server, run the Server.py located in the backend folder.
Frontend: To run the frontend you need to have node.js and npm installed (Link). Then you go into the id2221-frontend folder and install some packages using npm install. Then write npm start in the terminal. The frontend will start in the browser.

5 Images from produced website

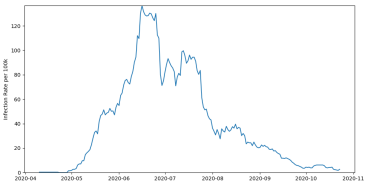
(a) Frontpage table



(b) Country table

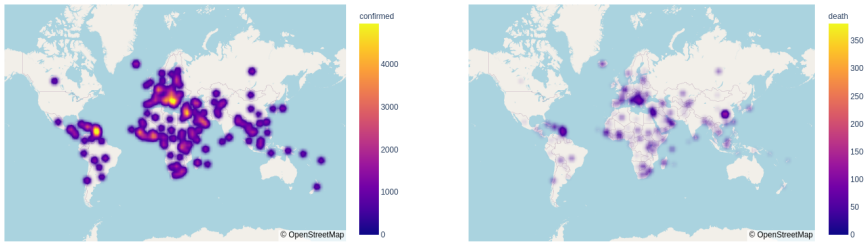


(a) Daily cases



(b) Infection Rate

Figure 3: Country Plots



(a) Confirmed cases (b) Deaths

Figure 4: Heatmaps from week 9