ID2221 Project - Group 333

Ibrahim Abdelkareem - Daniel Bruke - Erik Vindblad

Flight Route Aggregator

This project simulates live flight traffic data and uses spark to aggregate this data to show how many flights flew over different countries' airspace.

The project uses a lot of the topics discussed in this course:

- **NoSQL Database:** MongoDb
- **Message Broker:** Kafka
- **Data Processing:** Spark Structured Streaming
- **Containers:** Docker
- **Container Orchestration:** Docker Compose

Architecture

We've flight traffic data stored in hosted mongodb instance which is collected and sent over **Kafka** by **flight-route-publisher** which is a **.NET** application that can be found in `/apps/flight-route-publisher`.

Our **Scala** app **flight-route-aggregator-v2** which can be found in `/apps/flight-route-aggregator-v2` will read the Kafka topic as a **structured stream** via **Spark** and aggregates the data and uses a 3rd party library to determine the country's airspace for a flight given the latitude and longitude. Then it writes the aggregated data using **MongoDB** connector to mongodb.

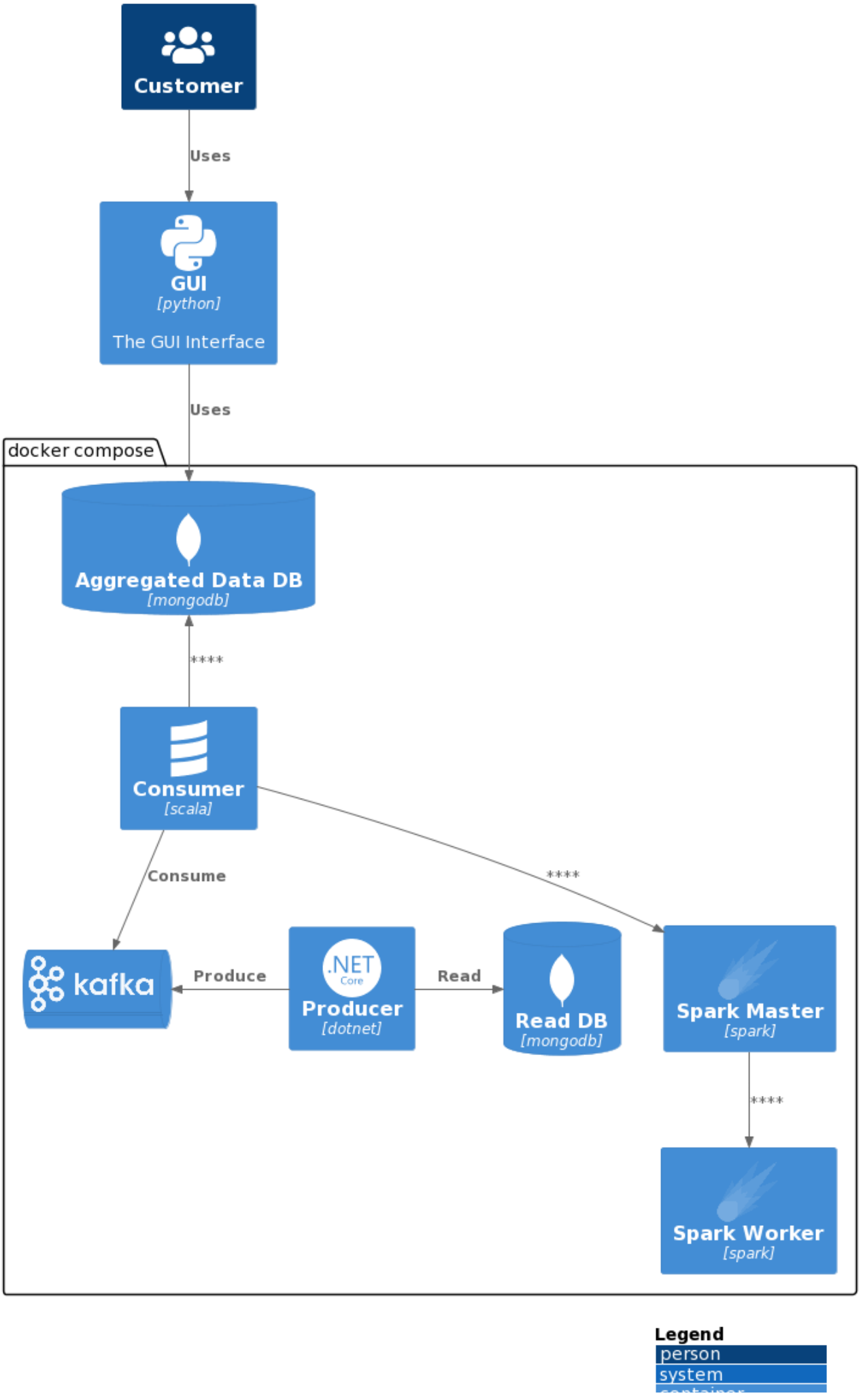
The Python GUI program connects to the stored mongodb collection of mapped country data by incoming stream. The python GUI updates every 10 seconds to account for planes that enter new airspaces. This is shown via proportional circles which show both the country and number of planes currently in that airspace

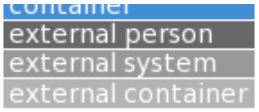
All the apps are **Dockerized** except the FE as it has a GUI interface which might not work well in a containerized envrionment, and the setup for the backend is made via **Docker Compose** (a lightweight orchestration framework, not as comprehensive as kubernetes but it does the job of spawning multiple containers, restart them on failure, and scaling them if needed. The implementation we did for docker-compose was simple to get all the containers running so the user of the project doesn't have to install and configure many dependencies such as spark or kafka).

The following Diagram should simplify the architecture.

NOTES:

- We switched from `python` implementation that can be found in `apps/flight-route-aggregator` to Scala implementation (hence v2) due to lake of support of datasets in `PySpark`.
- The 3rd party library we used to determine the country given latitude and longitude works offline and has simple implementation. We didn't want to use a hosted API so we don't hit the rate limit of usage.


Customer
Uses
GUI
[python]
The GUI Interface
Uses
docker compose
Aggregated Data DB
[mongodb]
****
Consumer
[scala]
Consume
****
kafka
Produce
.NET
Core
Producer
[dotnet]
Read
Read DB
[mongodb]
Spark Master
[spark]
****
Spark Worker
[spark]
Legend
person
system



Use

Back-End

- Install Docker on your local machine.
- Run `docker-compose up -d`
- To run the front-end application you should

```
cd ./frontend_gui
python3 plotting.py
```