Eoghan Spillane R00175214

SOFT8026 - Data-driven Microservices
Assignment 1 Form

Instructions
Please complete the following form and include in the zip file you submit. Include screenshots / images in the appendices below the form.

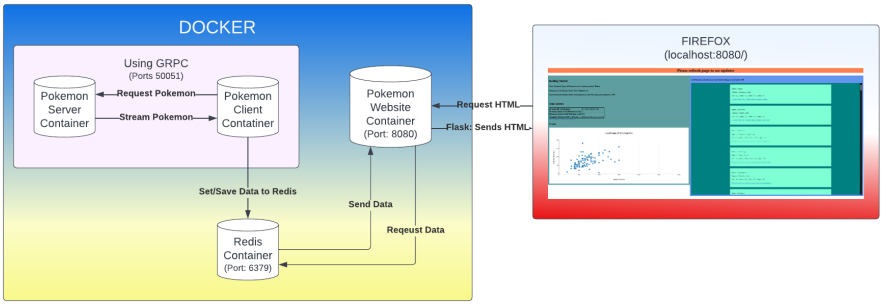
Which dataset (e.g. from Kaggle) did you use (provide a download link)?

Dataset: https://www.kaggle.com/thiagoazen/all-pokemon-with-stats

Briefly describe the data in the file and some of the columns you chose to analyse:

While this is only around 900 long I chose this as analyzing it was more interesting to be me. I worked out that it would still have roughly 9 minutes of data being streamed to do a rolling average.

I cleaned the data by removing any version of the pokemon with a mega evolution and then removed that column. I chose to analyze the types, health points and attack of the pokemon.

Discussion of Architecture	
Put a diagram of your architecture opposite (e.g. paste an image); indicate the microservices, integration endpoints, messages being sent, etc.	
Why did you opt for the microservices you did? You could address topics from the lectures, e.g. bounded contexts, independent pipelines, etc.	A **Server container**, which does some mild data clean up and randomizes the pokemon and then it waits for a **grpc** request. A **client container** that will request that data from the server and decode the grpc response. I've set this to only happen once instead of continuously for demonstration purposes. The client then analyses that data and saves the results to a **redis container** that can be asynchronously accessed. The **Website container** uses flask to build and render the website, when firefox checks port 8080, flask will then request the relevant data that's stored by redis, render/combine the html template with the analytics and send it back to firefox to display. I used redis as it proved to be an easier to implement system to sit between the website and the analytics client, it also functions asynchronously as the client is constantly sending it new data to store, but the website container only requests data on refresh it then gets the most up to date data on refresh. This then means that you could have multiple **scalable website containers** to deal with large amounts of web traffic etc.
Checklist of	**Delete as appropriate**

requirements completed	
Do you have a Dockerfile per microservice that you have created?	YES
Do you have a gRPC proto file with a service, rpcs and messages for each microservice?	YES
Do you have your Dockerfiles and other microservice files (python scripts, requirements.txt, etc) in separate folders?	YES
Do you have a functioning Docker Compose file?	YES
Are you streaming data, simulating real-time throughout?	YES
Do you have 4 pieces of analytics calculated, including a window-on-data (e.g. rolling 3-minute window)?	YES
Are you displaying live analytics on a web page?	YES (It auto refreshes every 15 seconds)
Any other comments, e.g. anything you want to highlight that could otherwise be missed by the marker?	I set up the website to also link to the pokedex website so that you can see what type of pokemon I'm referring to as I didn't recognise many of the names of newer pokemon and found I was googling them out of interest. E.g. The toughest pokemon or anyone that grabs your attention. I would have also liked to retrieve images from the website but the images don't use a predictable https link like the pokemon specific pages do.

Appendix – Screenshot(s) of your application running (e.g. Log output as it starts up, any log output as it runs, any web pages, etc.)

```
pokemon_client_1    | Requesting Pokemon
pokemon_client_1    | Sending Analytics
```

```
pokemon_website_1  | * Serving Flask app 'pokemon_website' (lazy loading)
pokemon_website_1  | * Environment: production
pokemon_website_1  |   WARNING: This is a development server. Do not use it in a production deployment.
pokemon_website_1  |   Use a production WSGI server instead.
pokemon_website_1  | * Debug mode: on
pokemon_website_1  | * Running on all addresses.
pokemon_website_1  |   WARNING: This is a development server. Do not use it in a production deployment.
pokemon_website_1  | * Running on http://172.23.0.5:5000/ (Press CTRL+C to quit)
pokemon_website_1  | * Restarting with stat
pokemon_website_1  | * Debugger is active!
pokemon_website_1  | * Debugger PIN: 470-909-174
```

```
pokemon_server_1    | Recieved -> RequestPokemon: 1
```

Please refresh page to see updates (Auto refreshs every 15 seconds)

Rolling Metric

Most Common Type of Pokemon over 3 minutes period: **Water**

Pokemon Type Rolling Status: **Not 3 minues yet**

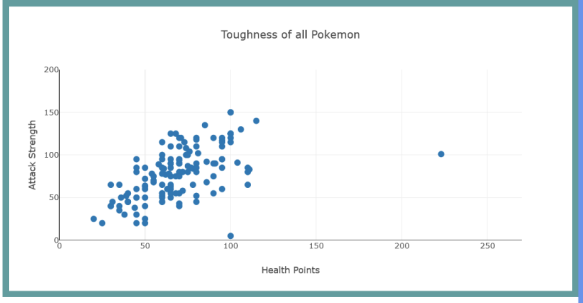
Current amount of types from each pokemon in list (Two types per pokemon): **219**

Other Metrics:

Average HP of Pokemon	68.92753623188406
Weakest attack of all Pokemon so far	5
Strongest attack of all Pokemon so far	150
Toughest Pokemon (HP + Attack)	Guzzlord with a score of 324

Open Pokedex Entry for Guzzlord

Graph



All Pokemon currently processed (Order Changes each build): **138**

Name: Mantyke
Types: Water, Flying
HP: 45, Atk: 20, Def: 50, Spd: 50
Click here to learn more about Mantyke

Name: Ambipom
Types: Normal, nan
HP: 75, Atk: 100, Def: 66, Spd: 115
Click here to learn more about Ambipom

Name: Tangela
Types: Grass, nan
HP: 65, Atk: 55, Def: 115, Spd: 60
Click here to learn more about Tangela

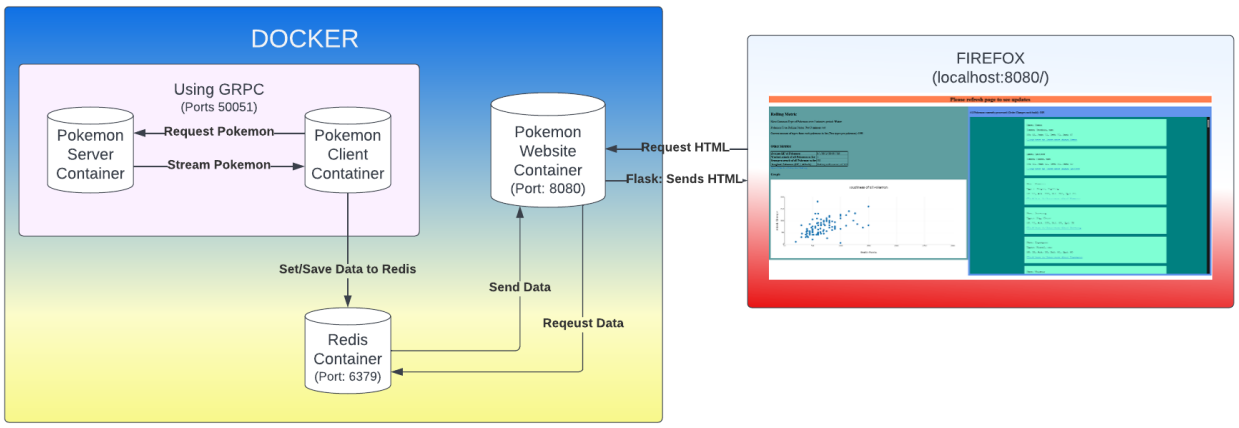
Name: Quagsire
Types: Water, Ground
HP: 95, Atk: 85, Def: 85, Spd: 35
Click here to learn more about Quagsire

Name: Monferno
Types: Fire, Fighting
HP: 64, Atk: 78, Def: 52, Spd: 81
Click here to learn more about Monferno

Name: Probopass
Types: Rock, Steel

```
pokemon_website_1  | Recieved Analytics
pokemon_website_1  | 172.23.0.1 - - [20/Mar/2022 21:55:05] "GET / HTTP/1.1" 200 -
```

Eoghan Spillane R00175214



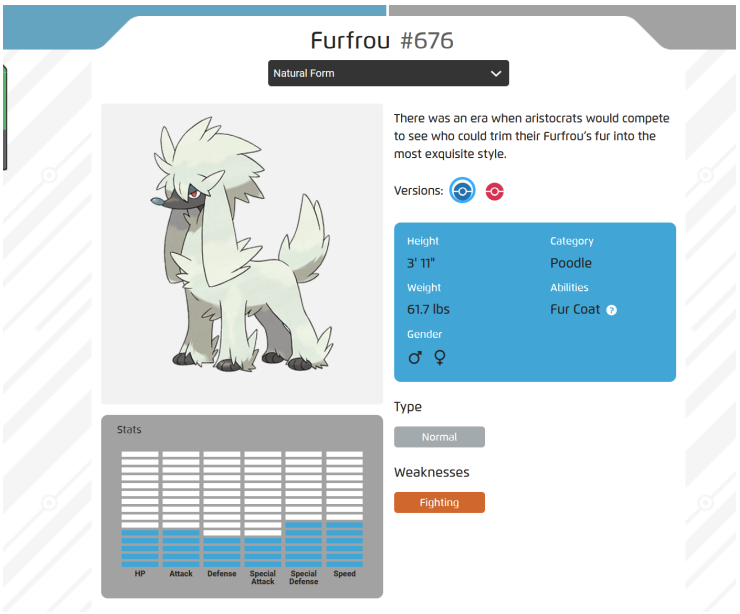
Name: Furfrou

Types: Normal, nan

HP: 75, Atk: 80, Def: 60, Spd: 102

Click here to learn more about Furfrou

Eoghan Spillane R00175214



Rolling Metric

Most Common Type of Pokemon over 3 minutes period: **Water**

Pokemon Type Rolling Status: **Rolling has started**

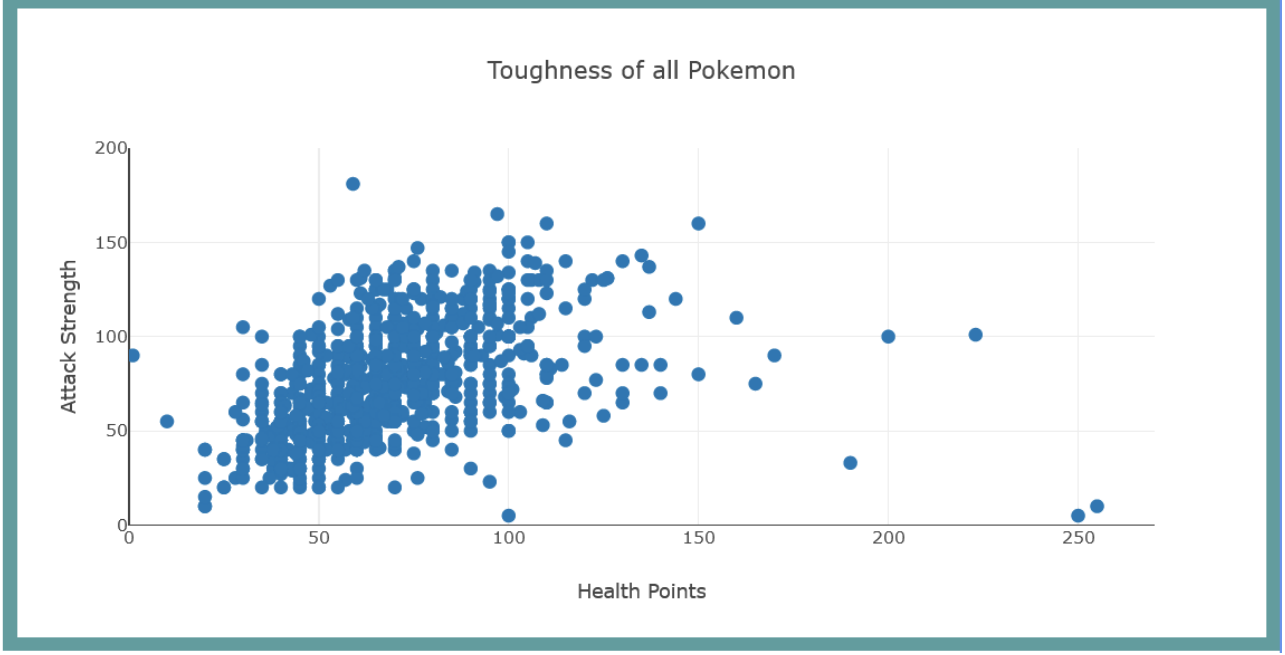
Current amount of types from each pokemon in list (Two types per pokemon): **538**

Other Metrics:

Average HP of Pokemon	68.87757437070938
Weakest attack of all Pokemon so far	5
Strongest attack of all Pokemon so far	181
Toughest Pokemon (HP + Attack)	Guzzlord with a score of 324

Open Pokedex Entry for Guzzlord

Graph



Eoghan Spillane R00175214

All Pokemon currently processed (Order Changes each build): **874**

Name: Sizzlipede

Types: Fire, Bug

HP: 50, Atk: 65, Def: 45, Spd: 45

Click here to learn more about Sizzlipede

Name: Dusknoir

Types: Ghost, nan

HP: 45, Atk: 100, Def: 135, Spd: 45

Click here to learn more about Dusknoir

Name: Skiddo

Types: Grass, nan

HP: 66, Atk: 65, Def: 48, Spd: 52

Click here to learn more about Skiddo

Name: Timburr

Types: Fighting, nan

HP: 75, Atk: 80, Def: 55, Spd: 35

Click here to learn more about Timburr

Name: Delphox

Types: Fire, Psychic

HP: 75, Atk: 69, Def: 72, Spd: 104

Click here to learn more about Delphox

Name: Jellicent

Types: Water, Ghost