SOFT8026 - Data-driven Microservices
Assignment 1 Form R00090111
Simone Rodigari

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

Brief discussion of your architecture – why the micro-services you chose, what messages are you passing between them?	I have chosen to implement a server to read the tweets csv and generate the gRPC stream which is then consumed by a client service which live-streams the tweets, processes some analytics and stores tweets/ analytics-results in a redis service. To complete the project I have developed a frontend service with python Flask which reads from redis and displays tweets (http:// localhost:8080). The server is streaming tweets with a 2 seconds delay between each tweet, so if you refresh the frontend you can see new tweets coming in. This architecture allows for separation of concerns where each service is in charge of a limited number of specific tasks. This architecture promotes low coupling and high cohesion where the only dependency is client-depend-on-server for the gRPC stream.
Briefly discuss how far you got with the following functional requirements:	

1. Reading and streaming of the tweet data	Server reads and create gRPC stream which is consumed by client
2. Data analysis; what analytics / metrics did you calculate?	1. Aggregate metric: total tweets been streamed since start of streaming 2. Rolling metric: sentiment within last 3 minutes (I have used TextBlob library to evaluate sentiment for each incoming twee 3. Single tweet: longest for the current stream
3. Web page with list or summary of analytics / metrics	The webpage displays the analytics mentioned in functional requirement 2
Checklist:	
I used gRPC	Y
I used streaming with gRPC	Y
I built docker images using Dockerfiles	Y
I orchestrated my application using Docker Compose and a YAML file	Y
Any other comments?	I have investigated gRPC-web and will definitely try implement that in my own time in the future. Unfortunately due to time-constraints I could not successfully complete that part.

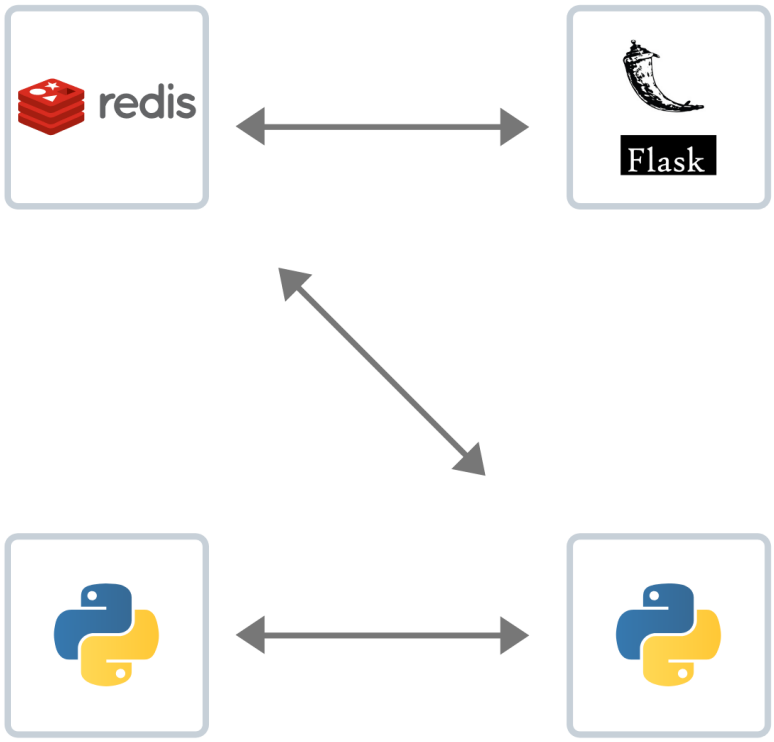
Appendix A – Your Architecture Sketch / Diagram

database

stores tweets and analysis data

web-server

reads from redis: http://localhost:8080



server

read tweets from .csv file and generate gRPC stream

client

consume gRPC stream, processes analytics and stores to redis

Appendix B – Output from a sample run

```
redis_1               | 1:C 17 Mar 2020 09:05:28.955 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
redis_1               | 1:C 17 Mar 2020 09:05:28.955 # Redis version=5.0.8, bits=64, commit=00000000, modified=0, pid=1, just started
redis_1               | 1:C 17 Mar 2020 09:05:28.956 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/
to/redis.conf
redis_1               | 1:M 17 Mar 2020 09:05:28.959 * Running mode=standalone, port=6379.
redis_1               | 1:M 17 Mar 2020 09:05:28.959 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower va
lue of 128.
redis_1               | 1:M 17 Mar 2020 09:05:28.959 # Server initialized
redis_1               | 1:M 17 Mar 2020 09:05:28.959 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usag
e issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain
 the setting after a reboot. Redis must be restarted after THP is disabled.
redis_1               | 1:M 17 Mar 2020 09:05:28.959 * Ready to accept connections
tweet_stream_client_1 | wait-for-it.sh: waiting 15 seconds for tweet_stream_server:9999
web_server_1          |  * Serving Flask app "web_server" (lazy loading)
web_server_1          |  * Environment: production
web_server_1          |    WARNING: This is a development server. Do not use it in a production deployment.
web_server_1          |    Use a production WSGI server instead.
web_server_1          |  * Debug mode: on
web_server_1          |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
web_server_1          |  * Restarting with stat
web_server_1          |  * Debugger is active!
web_server_1          |  * Debugger PIN: 719-800-417
tweet_stream_client_1 | wait-for-it.sh: tweet_stream_server:9999 is available after 1 seconds
tweet_stream_client_1 | real-time character count: 115
tweet_stream_client_1 | Client received: @switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer.  You shoulda got David Carr of Third Day to do it. ;D
tweet_stream_client_1 | 0.4
web_server_1          | 172.25.0.1 - - [17/Mar/2020 09:05:34] "GET / HTTP/1.1" 200 -
web_server_1          | 172.25.0.1 - - [17/Mar/2020 09:05:34] "GET /favicon.ico HTTP/1.1" 200 -
tweet_stream_client_1 | real-time character count: 226
tweet_stream_client_1 | Client received: is upset that he can't update his Facebook by texting it... and might cry as a result  School today also. Blah!
tweet_stream_client_1 | 0.0
tweet_stream_client_1 | real-time character count: 315
tweet_stream_client_1 | Client received: @Kenichan I dived many times for the ball. Managed to save 50%  The rest go out of bounds
tweet_stream_client_1 | 0.5
```

Appendix C – Screenshot from web page

Tweet stream

TOTAL COUNT: 49

LATEST TWEET: @tea oh! i'm so sorry i didn't think about that before retweeting.

SENTIMENT FOR LAST 3 MIN: positive (50 tweets in last 3 min) CURRENT TWEET: 0.0

LONGEST TWEET: Just checked my user timeline on my blackberry, it looks like the twanking is still happening Are ppl still having probs w/ BGs and UIDs?