

1. Levantar VM con Kafka
2. Levantar un Docker que hará una ingesta continua de datos al Kafka.
3. Levantar Google SQL Postgress

## Data simulator

Abrir ssh de la máquina virtual donde ya tenemos instalado el Kafka

Para entrar como super usuario:

```
Sudo -i
```

Entemos a la carpeta de Kafka

```
cd kafka_2.12-3.3.1
```

configuramos con vim la ip publica de Kafka en:

```
vim config/server.properties
```

nos dirigimos a la línea a configurar y presionamos i para editar, cambiamos la IP, escape para salir de modo edición y :wq para salir de vim

con el siguiente comando comprobamos si hemos cambiado correctamente la propiedad:

```
cat config/server.properties | grep advertised
```

levantamos el zookeeper:

```
bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
```

levantamos el kafka:

```
bin/kafka-server-start.sh -daemon config/server.properties
```

corremos el simulador de datos desde un Docker. Sustituir ip publica oír instanci de vm de Google (ip publica vm)

```
docker run -it -e KAFKA_SERVERS=34.88.224.164:9092 agutlop/data-simulator:1.0
```

en otro terminal vamos a crear los topics una para el ejercicio final y otro para el proyecto (creo que los crea el propio Docker simulador de datos):

```
bin/kafka-topics.sh --bootstrap-server localhost:9092 --topic test --partitions 1 --replication-factor 1 --create
```

```
bin/kafka-topics.sh --bootstrap-server localhost:9092 --topic devices --partitions 1 --replication-factor 1 --create
```

Con el siguiente commando comprobamos los topics creados en el Kafka:

```
bin/kafka-topics.sh --bootstrap-server localhost:9092 --list
```

con los siguientes comandos nos conectamos al consumer para ver si están entrando los datos en el topic:

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic  
devices --property print.key=true
```

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic  
antenna_telemetry --property print.key=true
```

## jdbc provisioner

será el modulo encargado de crear las tablas con las información con la que se enriquecerá el streaming así como de crear las tablas donde se almacenará la información ya tratada.

En la variable ipServer deberemos poner la IP pública de la base de datos de Google SQL. Previamente habremos configurado las conexiones de la base de datos abiertas a todo internet con 0.0.0.0/0

Navegmos a la instancia de la base de datos y abrimos el clod Shell con el que tendremos acceso a la base de datos. Cuando creemos las tablas desde el código podremos verificar su creación desde este Shell.

Las tablas creadas son aquellas con el prefijo “proyecto”

```
postgres=> \dt
```

List of relations			
Schema	Name	Type	Owner
public	final_antenna_1h_agg	table	postgres
public	final_antenna_agg	table	postgres
public	final_antenna_errors_agg	table	postgres
public	final_antenna_percent_agg	table	postgres
public	final_metadata	table	postgres
public	proyecto_batchbytesporantena	table	postgres
public	proyecto_batchbytesporaplicacion	table	postgres
public	proyecto_batchbytesporusuario	table	postgres
public	proyecto_streamingbytesporantena	table	postgres
public	proyecto_streamingbytesporaplicacion	table	postgres
public	proyecto_streamingbytesporusuario	table	postgres
public	proyecto_user_metadata	table	postgres

(12 rows)

Datos insertados:

```
postgres=> select * from proyecto_user_metadata;
```

id	name	email	quota
00000000-0000-0000-0000-000000000001	andres	andres@gmail.com	200000
00000000-0000-0000-0000-000000000002	paco	paco@gmail.com	300000
00000000-0000-0000-0000-000000000003	juan	juan@gmail.com	100000
00000000-0000-0000-0000-000000000004	fedede	fedede@gmail.com	5000
00000000-0000-0000-0000-000000000005	gorka	gorka@gmail.com	200000
00000000-0000-0000-0000-000000000006	luis	luis@gmail.com	200000
00000000-0000-0000-0000-000000000007	eric	eric@gmail.com	300000
00000000-0000-0000-0000-000000000008	carlos	carlos@gmail.com	100000
00000000-0000-0000-0000-000000000009	david	david@gmail.com	300000
00000000-0000-0000-0000-000000000010	juanchu	juanchu@gmail.com	300000
00000000-0000-0000-0000-000000000011	charo	charo@gmail.com	300000
00000000-0000-0000-0000-000000000012	delicidas	delicidas@gmail.com	1000000
00000000-0000-0000-0000-000000000013	milagros	milagros@gmail.com	200000
00000000-0000-0000-0000-000000000014	antonio	antonio@gmail.com	1000000
00000000-0000-0000-0000-000000000015	sergio	sergio@gmail.com	1000000
00000000-0000-0000-0000-000000000016	maria	maria@gmail.com	1000000
00000000-0000-0000-0000-000000000017	cristina	cristina@gmail.com	300000
00000000-0000-0000-0000-000000000018	lucia	lucia@gmail.com	300000
00000000-0000-0000-0000-000000000019	carlota	carlota@gmail.com	200000
00000000-0000-0000-0000-000000000020	emilio	emilio@gmail.com	200000

(20 rows)

## Streaming Job

Primero leemos el topic devices de Kafka y lo guardamos en un dataframe. Los datos nos llegan en formato json:

```
Run: AntennaStreamingJob
```

```
2022-10-30 09:20:22 [INFO ] [ressions.codegen.CodeGenerator:54] Code generated in 13.0527 ms
```

key	value	topic	partition	offset	timestamp	timestampType
[null]	[7B 22 62 79 74 6...]	devices	0	637	2022-10-30 09:20:22	0
[null]	[7B 22 62 79 74 6...]	devices	0	638	2022-10-30 09:20:22	0
[null]	[7B 22 62 79 74 6...]	devices	0	639	2022-10-30 09:20:22	0
[null]	[7B 22 62 79 74 6...]	devices	0	640	2022-10-30 09:20:22	0
[null]	[7B 22 62 79 74 6...]	devices	0	641	2022-10-30 09:20:22	0
[null]	[7B 22 62 79 74 6...]	devices	0	642	2022-10-30 09:20:22	0

Parseamos el value del json a structuredType:

AntennaStreamingJob x

```

-----
+-----+-----+-----+-----+-----+
| timestamp| id| antenna_id|bytes| app|
+-----+-----+-----+-----+-----+
|2022-10-30 11:42:00|00000000-0000-000...|22222222-2222-222...| 3978| SKYPE|
|2022-10-30 11:42:00|00000000-0000-000...|22222222-2222-222...| 2082| FACETIME|
|2022-10-30 11:42:00|00000000-0000-000...|22222222-2222-222...| 4116| SKYPE|
|2022-10-30 11:42:00|00000000-0000-000...|44444444-4444-444...| 8589| SKYPE|
|2022-10-30 11:42:00|00000000-0000-000...|44444444-4444-444...| 7528| FACETIME|

```

Leemos la metadata desde la base de datos y enriquecemos el stream con estos datos mediante un join:

Batch: 1

```

-----
+-----+-----+-----+-----+-----+-----+-----+
| timestamp| id| antenna_id|bytes| app| name| email| quota|
+-----+-----+-----+-----+-----+-----+-----+
|2022-10-30 12:54:38|00000000-0000-000...|22222222-2222-222...| 1267|FACEBOOK|delicidas|delicidas@gmail.com|1000000|
|2022-10-30 12:54:38|00000000-0000-000...|44444444-4444-444...| 8839|FACEBOOK| luis| luis@gmail.com| 200000|
|2022-10-30 12:54:38|00000000-0000-000...|44444444-4444-444...| 1668|FACETIME| eric| eric@gmail.com| 300000|

```

- Hacemos las agragaciones
- Escribimos de vuelta los datos a postgres y Parquet en directorio local