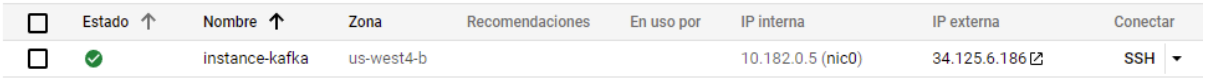
Crear instancia VM en Google Cloud Computer



Estado	Nombre	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
✓	instance-kafka	us-west4-b			10.182.0.5 (nic0)	34.125.6.186	SSH

importante, las conexiones a la máquina se harán mediante la IP externa, así que hay que crear una regla firewall para permitir las conexiones entrantes.

Instalar Kafka y Docker

Se procede a entrar a la estancia virtual mediante SSH, realizar los pasos (con sudo -i), Docker en este caso lo usaremos como productor de kafka para una simulación de envío, para ello se utilizará la ip externa mencionada antes.

Se va a crear el topic “devices”, que usaremos en el proyecto

Storage

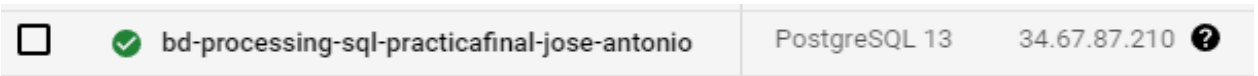
La idea del proyecto es crear un Bucker de Google Cloud Storage, pero en este caso se va a realizar en local, Storage va a ser una carpeta dentro de la raíz del sistema.

ID

El ID va a ser Intellij Idea como hemos estado viendo en clase, en este caso no se va a usar % provided dado que se va a ejecutar en local (la opción cloud sería levantar un cluster Google Dataproc, enviar el jar resultante del proyecto, y ejecutarlo como job)

SQL

Se va a utilizar en este caso Google Cloud SQL -> PostgreSQL, creamos la instancia admitiendo las conexiones públicas

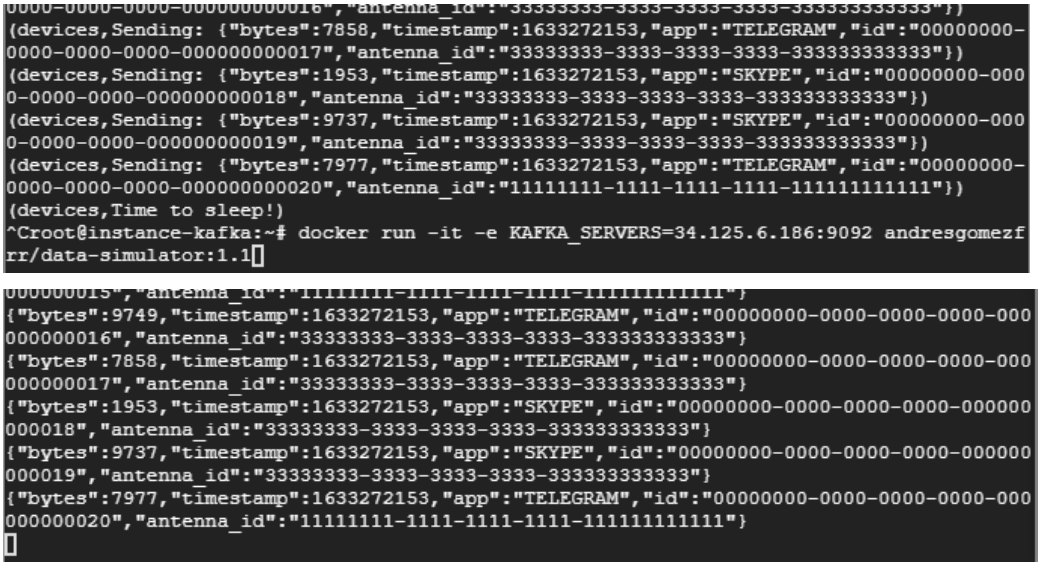


	bd-processing-sql-practicafinal-jose-antonio	PostgreSQL 13	34.67.87.210

también usaremos la ip publica para conectarnos desde el ID en la máquina local

Primera prueba, comunicación Docker - Kafka

Ejecutamos el docker provisto, y abrimos otro SSH como Consumer para comprobar que los datos llegan

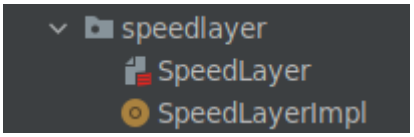


Speed Layer

Los argumentos de entrada van a ser los siguientes

kafkaServer	34.125.6.186:9092
topic	devices
jdbcUri	jdbc:postgresql://34.67.87.210:5432/postgres
jdbcMetadataTable	user_metadata
aggJdbcTable	bytes
jdbcUser	postgres
jdbcPassword	keepcoding1

storagePath	/tmp/spark-project



La idea según las funciones son

1. Leo desde kafka el topic devices
2. Parseo desde Json los datos extraído en kafka
3. Leo desde SQL la tabla de metadatos de usuario, extraigo dataframe
4. Uno las dos tablas mediante el identificador “id”
5. Calculo las métricas
    a. Agrupando por ventana y usuario (sumo los bytes)
    b. Agrupando por ventana y Antena (sumo los bytes)
    c. Agrupando por ventana y App (sumo los bytes)
    d. Formateo las 3 con el mismo “schema” (tabla bytes)
6. Escribo en JDBC las 3 tablas resultantes (en la misma tabla bytes)
7. Escribo en Storage local (punto 6 y 7 en paralelo)

Dado que tuve problemas para unir las 3 tablas por incompatibilidad entre “unión, join” y los Dataframes de origen Streaming, opté por la solución Seq[Futures] para escribir las 3 en paralelo en la misma tabla en la base de datos

```
val aggFutureUser = writeToJdbc(aggBybytesDFUser, jdbcUri, aggJdbcTable, jdbcUser, jdbcPassword)
val aggFutureApp = writeToJdbc(aggBybytesDFApp, jdbcUri, aggJdbcTable, jdbcUser, jdbcPassword)
val aggFutureAntenna = writeToJdbc(aggBybytesDFAntenna, jdbcUri, aggJdbcTable, jdbcUser, jdbcPassword)
Await.result(Future.sequence(Seq(aggFutureUser, aggFutureApp, aggFutureAntenna, storageFuture)), Duration.Inf)
```

```
tonyzetag@DESKTOP-P7T7R2D:~$ ls /tmp/spark-project
checkpoint  data
tonyzetag@DESKTOP-P7T7R2D:~$ ls /tmp/spark-project/data
 _spark_metadata  'year=2021'
tonyzetag@DESKTOP-P7T7R2D:~$ ls /tmp/spark-project/data/year\=2021/
'month=10'
tonyzetag@DESKTOP-P7T7R2D:~$ ls /tmp/spark-project/data/year\=2021/month\=10/
'day=3'
tonyzetag@DESKTOP-P7T7R2D:~$ ls /tmp/spark-project/data/year\=2021/month\=10/day\=3/
'hour=16'
tonyzetag@DESKTOP-P7T7R2D:~$ ls /tmp/spark-project/data/year\=2021/month\=10/day\=3/hour\=16/
part-00000-96990aff-049c-4942-a8c4-051b69f5a7b8.c000.snappy.parquet
```

Batch Layer

Después de guardar la tabla bytes desde el Speed Layer, como podemos ver en la imagen, estos archivos parquet son los que se utilizará en este apartado

1. Leo los archivos parquet desde el Storage
2. Leo los metadatos desde SQL
3. Uno mis 2 tablas, por el campo id
4. Calculo mis nuevas métricas, esta vez como ya tenemos la tabla bytes, usaremos la misma pero con una ventana con tamaño de 1 hora, está será nuestra tabla bytes_hourly
5. Cálculo la tabla user_quota_limit
6. escribo en la base de datos