

Describe the following:

Sink and Source connectors.

Sink connectors export data to another system and source connectors import data from another system. Sink connectors often export the data to indexes, batch systems or databases. Source connectors often import data from systems such as databases or message brokers.

The applications/advantages of using Kafka Connectors with data storage.

Kafka connectors can make it simple to import data using a source connector into a kafka topic and exporting data from a kafka topic to an external system using sink connectors.

Advantages

- Scalability
 - Connectors runs with stream and batch systems
- Flexibility
 - Can be scaled using distributed systems or standalone systems

How do Kafka connectors maintain availability?

Kafka connectors maintain availability by using worker processes to execute tasks and when one fails another worker process can take over the failed task.

List the popular Kafka converters for values and the properties/advantages of each.

- **Avro**
 - Direct mapping to and from JSON
 - Compact
 - Fast
 - Extensible schema language
- **Protobuf**
 - Defines binary serialization format
 - Defines Json serialization format
 - Switch to JSON when readability is required
 - Very Fast
- **String**
 - Converts data object to a native object
- **ByteArray**
 - Converts data into a byte array

Search the internet to answer the following question:

What's a Key-Value (KV) database?

A KV database is a non relational database where data is stored in a key-value format where the key value is a unique key used to retrieve the data associated with that particular key.

What are KV databases' advantages and disadvantages?**Advantages**

- Highly partitionable
- Horizontal scaling
- Fast read and write operations
- Simplicity

Disadvantages

- Filtering values is difficult
- The only fast queries are the simple ones
- Not optimized for looking up values

List some popular KV databases.

- Amazon DynamoDB
- Oracle NoSQL
- InfinityDB

Follow the following videos to deploy and use Redis and MySQL databases using GKE.

Record a video showing the configuration of Kafka connectors, producers' python script, a proof of successfully stored data into data storage.

During the progress on this step of the lab , I discovered that my free trial had ended and it had been charging me 50\$ in March alone. I was required to make a new account and have to restart from the kafka cluster section of the previous labs. I was not able to finish but this is the progress I had on importing the sensor's csv file data into kafka.

CSV File

	A	B	C	D	E	F	G	H	I
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0	
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0	
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0.005	
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0.005	
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0.005	
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0.005	
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0	
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0	
	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0	
0	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0	
1	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0.005	
2	1.37E+15	2	0	0.738168	-1.46099	nan	17.8	0	
3	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0	
4	1.37E+15	2	0	0.738168	-1.46099	nan	17.8	0.005	
5	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0.005	
5	1.37E+15	2	0	0.738168	-1.46099	nan	17.8	0.005	
7	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0.005	
8	1.37E+15	3	0	0.738168	-1.46099	269	17.8	0.005	

Description of fields for database names


Field	Description	Units
1	UTIME of the GPS fix	μs
2	Fix mode, as reported by the GPS unit: 0 means that the mode update is not yet seen, 1 means that there is no GPS fix, 2 means that the fix is good for longitude and latitude, 3 means that the fix is also good for altitude.	–
3	The number of satellites used in the fix	–
4	Latitude	rad
5	Longitude	rad
6	Altitude	m
7	Track	m
8	Speed	m/s

Custom sql file including sample input

```
CREATE DATABASE IF NOT EXISTS myDB;
USE myDB;

DROP TABLE IF EXISTS test;

CREATE TABLE IF NOT EXISTS test (
  id serial NOT NULL PRIMARY KEY,
  utime varchar(100),
  fixmode int(200),
  satellites varchar(200),
  latitude varchar(100),
  longitude varchar(100),
  altitude varchar(100) ,
  track varchar(100),
  speed varchar(100),
  modified timestamp default CURRENT_TIMESTAMP NOT NULL,
  INDEX `modified_index` (`modified`)
);
USE myDB;
INSERT INTO test (utime, fixmode, satellites,latitude,longitude,altitude,track,speed) VALUES (1.36518E+15,3,0,0.738168312,-1.460986747,269,17.8,0);


```

Producer code

```
scl.sql x prod_mysql.py x schema.avsc x
1 from kafka import KafkaProducer
2 from csv import DictReader
3 import json
4 import time
5 import io
6 from avro.io import DatumWriter, BinaryEncoder
7 import avro.schema
8
9 schemaID = 100003
10
11 data = json.load(open('cred.json'))
12 bootstrap_servers = data['bootstrap_servers']
13 sasl_plain_username = data['Api Key']
14 sasl_plain_password = data['Api secret']
15
16 schema = avro.schema.parse(open("./schema.avsc").read())
17 writer = DatumWriter(schema)
18
19
20 def encode(value):
21     bytes_writer = io.BytesIO()
22     encoder = BinaryEncoder(bytes_writer)
23     writer.write(value, encoder)
24     return schemaID.to_bytes(5, 'big') + bytes_writer.getvalue()
25
26
27 with open('gps.csv', 'r') as read_obj:
28     csv_dict_reader = DictReader(read_obj)
29     for row in csv_dict_reader:
30         print(row)
31     value = {row, 'modified': int(1000 * time.time())};
32     producer = KafkaProducer(bootstrap_servers=bootstrap_servers, security_protocol='SASL_SSL', sasl_mechanism='PLAIN',
33                             sasl_plain_username=sasl_plain_username, sasl_plain_password=sasl_plain_password,
34                             value_serializer=lambda m: encode(m))
35     producer.send('ToMySQL', value)
36     producer.close()
37
```

I was only able to test with the sensor's data and this is my approach to the python producer code. It is similar to the version from the lab folder but opens the sensor's csv value and produces the row of values to the “ToMySQL” database. I could not attempt using the images as I did not want to be charged for 88 GB worth of image production.

This ideally would be sent to the “ToMySQL” database using a sink connector, allowing the data to move from the kafka topic to a database.

Which topics do you want to get data from?

topics

ToMySQL x

+ Create topic

How should we connect to your data?

Connector class ⓘ

MySQLSink

Name

MySQLSinkConnector_0

Input messages

Input Kafka record value format* ⓘ

AVRO

Input Kafka record key format ⓘ

STRING

Delete on null ⓘ

Then messages would be able to be seen in the messages topic tab in the confluent cloud or using a source connector and a python consumer script .

ToMySQL

Overview Messages Schema Configuration

Producers

Bytes in/sec --

Consumers

Bytes out/sec --

Message fields

Filter by keyword

Jump to offset

offset

+ Produce a new message to this topic

Message fields

List some possible applications that can be implemented by using the uploaded dataset.

The uploaded dataset would ideally be used as a training set for some type of robot that requires AI. The sensors and gps data would allow it to learn where to move around and images could allow a camera to recognize objects.