SOFE 4630U
Cloud Computing

Project Milestone
2022/03/15

Data Storage - KV + Relational

Sabesan Sivakumar (100701928),

Describe the following:

Sink and Source connectors.

- Sink connectors are what is used to export data out of Kafka
- Sink connectors are widely supported by multiple databases such as Oracle or DB2
- Source connectors do the opposite of sink and they import data into Kafka

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

- The use of connectors allows data to be easily imported and exported from Kafka to other systems
- Scalable as they run via steam and batch-oriented systems

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

Avro-Converter

- Direct mapping
- Compact Format
- Gets the Kafka Schemas and turns them into Avro Schemas, from that it gets the message keys and values and puts them into a binary form

Protobuf-Converter

- Seralizer (Send messages to Kafka of Protobuf type)
- Deserializer (Receive a message from Kafka of Protobuf type)

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

- Is a database in which data is only stored in a key-value format.
- Used for storing, retrieving, and managing the arrays in the database that is associated with the

What are KV databases' advantages and disadvantages?

Sabesan Sivakumar:

Advantages

- Straight Forward Commands (delete, export, import, etc..)
- Scalability: (horizontal scalability)

Disadvantages

- Theirs is no filtering of the value fields
- Individual values cannot be updated on as a whole they can be updated

How do Kafka connectors maintain availability?

Sabesan Sivakumar:

Kafka connectors maintain availability because the connector itself contains a worker process. These worker processes are responsible for executing connectors and tasks. With multiple worker processes, they can ensure availability in a system as if one fails another can replace it as they are highly coupled

List some popular KV databases.

Sabesan Sivakumar:

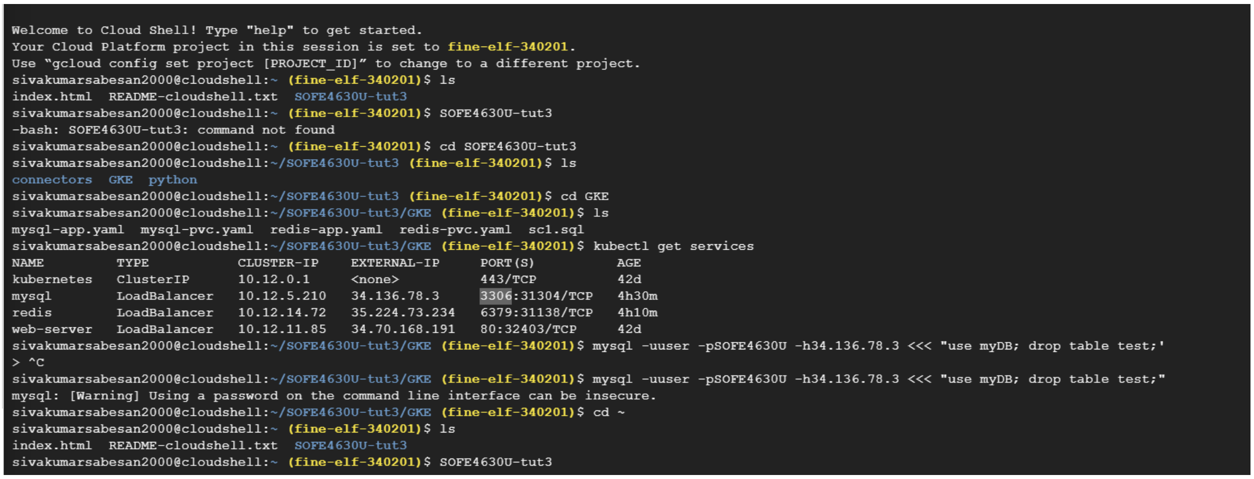
- Aerospike
- Redis
- Amazon DynamoDB

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

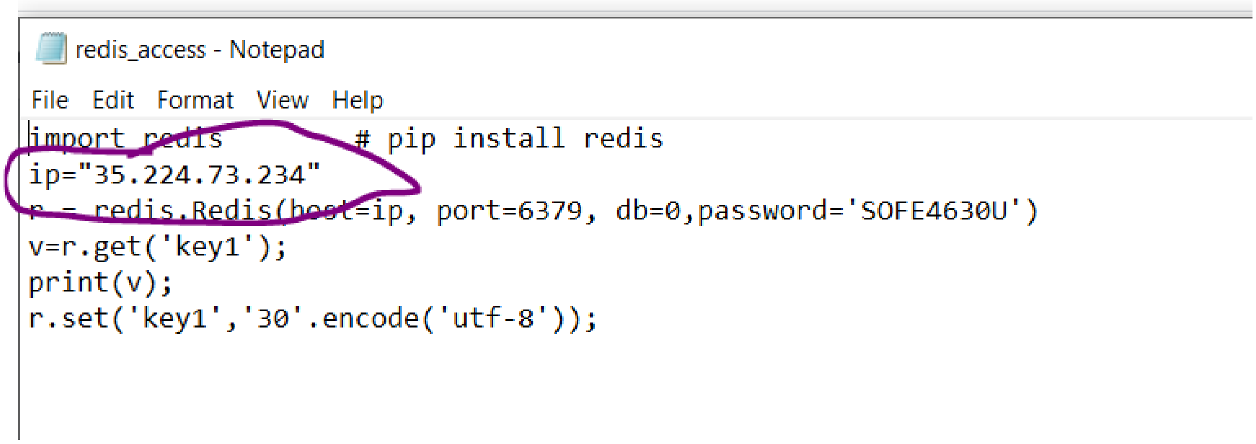
Sabesan Sivakumar:

Some possible application that can be implemented is self-driving cars. The sensors on the car would allow to extract data of the road and based on the data , the car will adjust its speed.

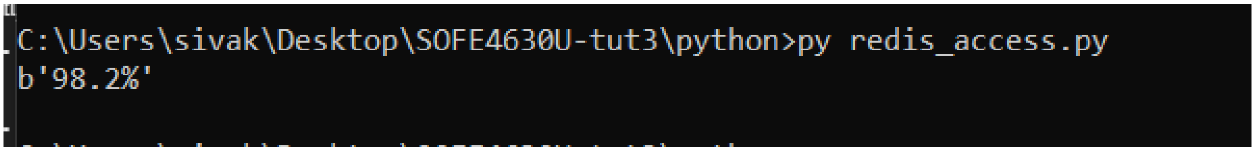
Screenshots:



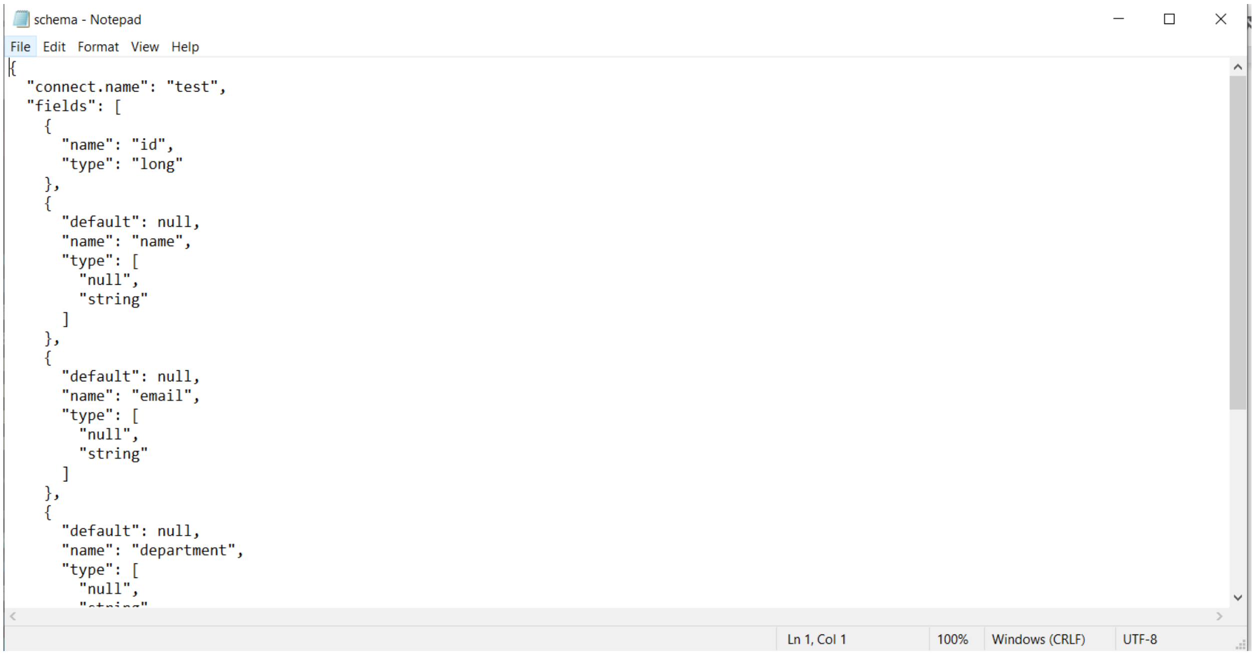
```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to fine-elf-340201.
Use “gcloud config set project [PROJECT_ID]” to change to a different project.
sivakumarsabesan2000@cloudshell:~ (fine-elf-340201)$ ls
index.html  README-cloudshell.txt  SOFE4630U-tut3
sivakumarsabesan2000@cloudshell:~ (fine-elf-340201)$ SOFE4630U-tut3
-bash: SOFE4630U-tut3: command not found
sivakumarsabesan2000@cloudshell:~ (fine-elf-340201)$ cd SOFE4630U-tut3
sivakumarsabesan2000@cloudshell:~/SOFE4630U-tut3 (fine-elf-340201)$ ls
connectors  GKE  python
sivakumarsabesan2000@cloudshell:~/SOFE4630U-tut3 (fine-elf-340201)$ cd GKE
sivakumarsabesan2000@cloudshell:~/SOFE4630U-tut3/GKE (fine-elf-340201)$ ls
mysql-app.yaml  mysql-pvc.yaml  redis-app.yaml  redis-pvc.yaml  sc1.sql
sivakumarsabesan2000@cloudshell:~/SOFE4630U-tut3/GKE (fine-elf-340201)$ kubectl get services
NAME         TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
kubernetes   ClusterIP      10.12.0.1     <none>          443/TCP          42d
mysql        LoadBalancer   10.12.5.210   34.136.78.3     3306:31304/TCP   4h30m
redis        LoadBalancer   10.12.14.72   35.224.73.234   6379:31138/TCP   4h10m
web-server   LoadBalancer   10.12.11.85   34.70.168.191   80:32403/TCP     42d
sivakumarsabesan2000@cloudshell:~/SOFE4630U-tut3/GKE (fine-elf-340201)$ mysql -uuser -pSOFE4630U -h34.136.78.3 <<< "use myDB; drop table test;'
> ^C
sivakumarsabesan2000@cloudshell:~/SOFE4630U-tut3/GKE (fine-elf-340201)$ mysql -uuser -pSOFE4630U -h34.136.78.3 <<< "use myDB; drop table test;"
mysql: [Warning] Using a password on the command line interface can be insecure.
sivakumarsabesan2000@cloudshell:~/SOFE4630U-tut3/GKE (fine-elf-340201)$ cd ~
sivakumarsabesan2000@cloudshell:~ (fine-elf-340201)$ ls
index.html  README-cloudshell.txt  SOFE4630U-tut3
sivakumarsabesan2000@cloudshell:~ (fine-elf-340201)$ SOFE4630U-tut3
```



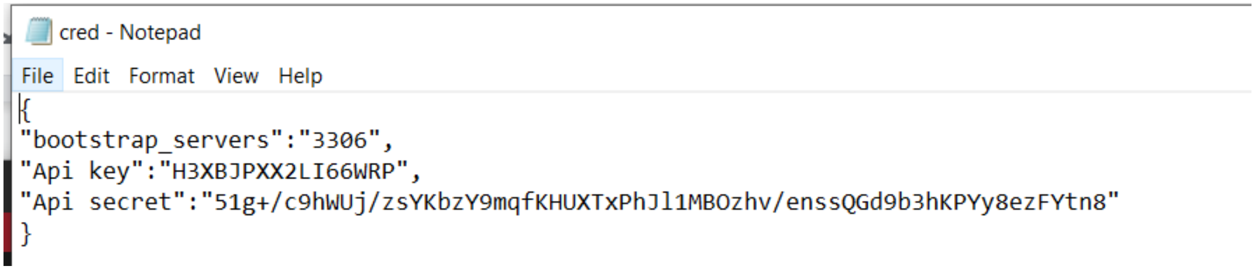
```
import redis        # pip install redis
ip="35.224.73.234"
r = redis.Redis(host=ip, port=6379, db=0,password='SOFE4630U')
v=r.get('key1');
print(v);
r.set('key1','30'.encode('utf-8'));
```



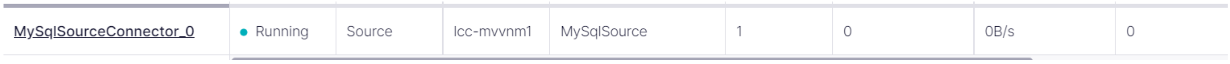
```
C:\Users\sivak\Desktop\SOFE4630U-tut3\python>py redis_access.py
b'98.2%'
```



```
{
  "connect.name": "test",
  "fields": [
    {
      "name": "id",
      "type": "long"
    },
    {
      "default": null,
      "name": "name",
      "type": [
        "null",
        "string"
      ]
    },
    {
      "default": null,
      "name": "email",
      "type": [
        "null",
        "string"
      ]
    },
    {
      "default": null,
      "name": "department",
      "type": [
        "null",
        "string"
```

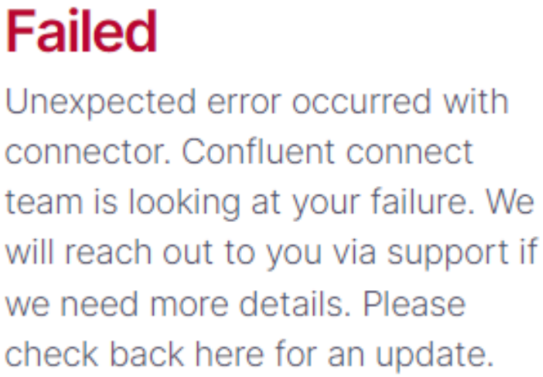


```
{
"bootstrap_servers":"3306",
"Api key":"H3XBJPXX2LI66WRP",
"Api secret":"51g+/c9hWUj/zsYKbzY9mqfKHUXTxPhJl1MBOzhv/enssQGd9b3hKPYy8ezFYtn8"
}
```



MySqlSourceConnector_0	● Running	Source	lcc-mvvnm1	MySqlSource	1	0	0B/s	0

When I run the mysql -uuser -pSOFE4630U -h34.136.78.3 < sc1.sql to see if a message appears I kept on getting this error.



Failed

Unexpected error occurred with connector. Confluent connect team is looking at your failure. We will reach out to you via support if we need more details. Please check back here for an update.