Project Milestone 3 - Data Storage Implementation: KV + relational

Esam Uddin - 100711116

3/18/2022

Given Lab 3 Repository: https://github.com/goergedaoud/SOFE4630U-tut3

Group Project Repository:

https://github.com/esam191/Intelligent-Transportation-System

Objectives:

- Deploy Tabular and key-Value data storage to GKE.
- Get familiar with Key-Value data storage
- Get familiar with Kafka Connectors and their configuration.
- Configure and use Kafka source connector to Redis.
- Configure and use MySQL sink and source Kafka connectors.

Procedure:

1. **Watch the first three videos for Kafka connectors (focus on the concepts, not the details).**

2. **Describe the following:**

    - **Sink and Source connectors.**

        - The source connector basically imports data from any relational database like MySQL with another driver into a Kafka topic.
        - Source connector also collects and stores metrics from application servers in Kafka topics.
        - On the other hand, the sink connector exports data from Kafka topics to any relational database with the help of a driver/secondary indexes like Elasticsearch.
        - In other words, both connectors are essentially just used to copy data between Kafka and other systems where you want to push data to or pull data from.

- **The applications/advantages of using Kafka Connectors with data storage.**
    - Using Kafka connectors allows you to stream data both scalably and reliably between Kafka and other data storage systems.
    - Kafka connectors help with moving large data sets both into and out of Kafka.
    - Kafka connectors also allow for a data centric pipeline in which appropriate data abstractions are used.
    - Using Kafka connectors also provides flexibility as data streaming runs on a single node.

- **How do Kafka connectors maintain availability?**
    - Kafka connectors maintain availability by running the streaming and batch-oriented systems on a single node.
    - Kafka connectors provide distributed service, which basically means it is scaled to an organization wide service.
    - Kafka connectors provide lower time to production by using existing connectors (avoids downtime), which in turn provides availability.

- **List the popular Kafka converters for values and the properties/advantages of each.**
    - JsonConverter
        - Embeds schema in the JSON
        - Serializes message keys and values into JSON docs
    - AvroConverter
        - Specifies schema registry
        - Can be used for message payload
    - ProtobufConverter
        - performs protobuf serialization
    - StringConverter
        - Just a string
        - No schema to the data

3. **Search the internet to answer the following question:**

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

    - Provides storing, retrieving of key value data
    - Known as a dictionary or hash table
    - Records are stored and retrieved using a unique key

    Ex. table shows values associated with keys

Key	Value
K1	AA,FD,FG
K2	FS,GG,HH
K3	1234,SAS
K4	FF,12,2222

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

    - Advantages
        - Scalability: scalable both vertically and horizontally
        - Reliability: built-in redundancy
        - Speed: quick to respond, data has any type/multiple types
        - Simplicity: simple to use

    - Disadvantages
        - No query language: difficult to transport queries from one database to a KV database
        - No filtration of values: whole values are returned when a request is made
        - Too simple: not very refined

- **List some popular KV databases.**

    - Redis
    - Amazon DynamoDB
    - Oracle NoSQL DB
    - InfinityDB
    - Aerospike

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

   Video 1: https://www.youtube.com/watch?v=qVD1uVKMZYc

```
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl apply -f mysql-pvc.yaml
persistentvolumeclaim/mysql-volumeclaim created
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl get pvc
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-volumeclaim   Bound    pvc-99c2054b-145d-4368-bf17-494489544a17   10Gi       RWO            standard       10s
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl apply -f mysql-app.yaml
service/mysql created
deployment.apps/mysql created
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl get pods
NAME                     READY   STATUS              RESTARTS   AGE
mysql-7dcb5fd764-zjj6s   0/1     ContainerCreating   0          18s
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
mysql-7dcb5fd764-zjj6s   1/1     Running   0          34s
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mysql   1/1     1            1           54s
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl get services
NAME         TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)          AGE
kubernetes   ClusterIP      10.12.0.1     <none>           443/TCP          4m34s
mysql        LoadBalancer   10.12.14.37   104.197.66.164   3306:30768/TCP   74s
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$
```

   Video 2: https://www.youtube.com/watch?v=GBvGjVLbYIs

```
mysql> exit
Bye
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ ls
mysql-app.yaml  mysql-pvc.yaml  redis-app.yaml  redis-pvc.yaml  sc1.sql
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ cat sc1.sql
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,
  name varchar(100),
  email varchar(200),
  department varchar(200),
  modified timestamp default CURRENT_TIMESTAMP NOT NULL,
  INDEX `modified_index` (`modified`)
);
USE myDB;
INSERT INTO test (name, email, department) VALUES ('alice', 'alice@abc.com', 'eng.');
INSERT INTO test (name, email, department) VALUES ('bob1', 'bob1@abc.com', 'sales');
INSERT INTO test (name, email, department) VALUES ('bob2', 'bob2@abc.com', 'sales');
INSERT INTO test (name, email, department) VALUES ('bob3', 'bob3@abc.com', 'sales');
INSERT INTO test (name, email, department) VALUES ('bob4', 'bob4@abc.com', 'sales');
```

```
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ cat scl.sql |kubectl exec -i mysql-7dcb5fd764-zjj6s -- mysql -uuser -pSOFE4630U
mysql: [Warning] Using a password on the command line interface can be insecure.
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ echo "select * from myDB.test;" | kubertl exec -i mysql-7dcb5fd764-zjj6s -- mysql -uuser -pSOFE
4630U
-bash: kubertl: command not found
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ echo "select * from myDB.test;" | kubectl exec -i mysql-7dcb5fd764-zjj6s -- mysql -uuser -pS
OFE4630U
mysql: [Warning] Using a password on the command line interface can be insecure.
id      name    email   department      modified
1       alice   alice@abc.com   eng.    2022-03-18 07:18:09
2       bob1    bob1@abc.com    sales   2022-03-18 07:18:09
3       bob2    bob2@abc.com    sales   2022-03-18 07:18:09
4       bob3    bob3@abc.com    sales   2022-03-18 07:18:09
5       bob4    bob4@abc.com    sales   2022-03-18 07:18:09
6       bob5    bob5@abc.com    sales   2022-03-18 07:18:09
7       bob6    bob6@abc.com    sales   2022-03-18 07:18:09
8       bob7    bob7@abc.com    sales   2022-03-18 07:18:09
9       bob8    bob8@abc.com    sales   2022-03-18 07:18:09
10      bob9    bob9@abc.com    sales   2022-03-18 07:18:09
```

```
mysql> use myDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from test;
+----+-------+---------------+------------+---------------------+
| id | name  | email         | department | modified            |
+----+-------+---------------+------------+---------------------+
|  1 | alice | alice@abc.com | eng.       | 2022-03-18 07:18:09 |
|  2 | bob1  | bob1@abc.com  | sales      | 2022-03-18 07:18:09 |
|  3 | bob2  | bob2@abc.com  | sales      | 2022-03-18 07:18:09 |
|  4 | bob3  | bob3@abc.com  | sales      | 2022-03-18 07:18:09 |
|  5 | bob4  | bob4@abc.com  | sales      | 2022-03-18 07:18:09 |
|  6 | bob5  | bob5@abc.com  | sales      | 2022-03-18 07:18:09 |
|  7 | bob6  | bob6@abc.com  | sales      | 2022-03-18 07:18:09 |
|  8 | bob7  | bob7@abc.com  | sales      | 2022-03-18 07:18:09 |
|  9 | bob8  | bob8@abc.com  | sales      | 2022-03-18 07:18:09 |
| 10 | bob9  | bob9@abc.com  | sales      | 2022-03-18 07:18:09 |
+----+-------+---------------+------------+---------------------+
10 rows in set (0.00 sec)
```

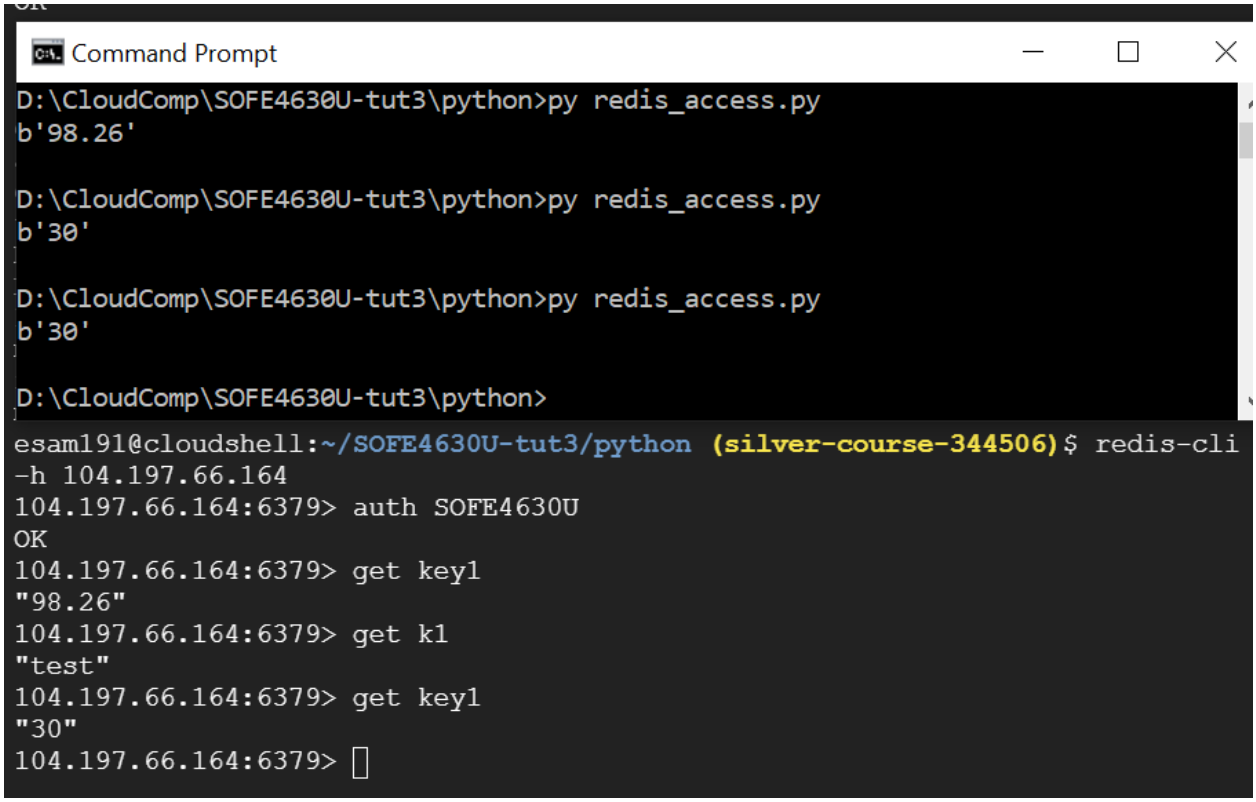
```
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ mysql -uuser
-pSOFE4630U -h34.134.216.154 <<< "select * from myDB.test"
mysql: [Warning] Using a password on the command line interface can be insecu
re.
id      name    email   department      modified
1       alice   alice@abc.com   eng.    2022-03-18 07:28:42
2       bob1    bob1@abc.com    sales   2022-03-18 07:28:42
3       bob2    bob2@abc.com    sales   2022-03-18 07:28:42
4       bob3    bob3@abc.com    sales   2022-03-18 07:28:42
5       bob4    bob4@abc.com    sales   2022-03-18 07:28:43
6       bob5    bob5@abc.com    sales   2022-03-18 07:28:43
7       bob6    bob6@abc.com    sales   2022-03-18 07:28:43
8       bob7    bob7@abc.com    sales   2022-03-18 07:28:43
9       bob8    bob8@abc.com    sales   2022-03-18 07:28:43
10      bob9    bob9@abc.com    sales   2022-03-18 07:28:43
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$
```

Video 3: https://www.youtube.com/watch?v=9R3eEAlpOtk&feature=youtu.be

```
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl get services
NAME         TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)          AGE
kubernetes   ClusterIP      10.12.0.1     <none>           443/TCP          58m
redis        LoadBalancer   10.12.9.51    104.197.66.164   6379:30022/TCP   54s
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ kubectl exec-it redis-56644c686c-vsb5g -- redis-cli
127.0.0.1:6379> auth SOFE4630U
OK
127.0.0.1:6379> set k1 test
OK
127.0.0.1:6379> get k1
"test"
127.0.0.1:6379> set key1 98.26
OK
127.0.0.1:6379> get key1
"98.26"
127.0.0.1:6379> keys *
1) "key1"
2) "k1"
127.0.0.1:6379> keys k?
1) "k1"
127.0.0.1:6379> keys k?y*
1) "key1"
127.0.0.1:6379> set Course "Cloud Computing"
OK
127.0.0.1:6379> get Course
"Cloud Computing"
127.0.0.1:6379> exit
```

```
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ redis-cli - h
104.197.66.164
Unrecognized option or bad number of args for: '-'
esam191@cloudshell:~/SOFE4630U-tut3/GKE (silver-course-344506)$ redis-cli -h 104.197.66.164
104.197.66.164:6379> auth SOFE4630U
OK
104.197.66.164:6379> keys *
1) "key1"
2) "Course"
3) "k1"
104.197.66.164:6379> select 0
OK
104.197.66.164:6379> keys *
1) "key1"
2) "Course"
3) "k1"
104.197.66.164:6379> select 1
OK
104.197.66.164:6379[1]> keys *
(empty array)
104.197.66.164:6379[1]> select 0
OK
104.197.66.164:6379> keys *
1) "key1"
2) "Course"
3) "k1"
104.197.66.164:6379>
```

```
import redis          # pip install redis
ip="104.197.66.164"
r = redis.Redis(host=ip, port=6379, db=0,password='SOFE4630U')
v=r.get('key1');
print(v);
r.set('key1','30'.encode('utf-8'));
~
~
```


Command Prompt
D:\CloudComp\SOFE4630U-tut3\python>py redis_access.py
b'98.26'
D:\CloudComp\SOFE4630U-tut3\python>py redis_access.py
b'30'
D:\CloudComp\SOFE4630U-tut3\python>py redis_access.py
b'30'
D:\CloudComp\SOFE4630U-tut3\python>
esam191@cloudshell:~/SOFE4630U-tut3/python (silver-course-344506)$ redis-cli -h 104.197.66.164
104.197.66.164:6379> auth SOFE4630U
OK
104.197.66.164:6379> get key1
"98.26"
104.197.66.164:6379> get k1
"test"
104.197.66.164:6379> get key1
"30"
104.197.66.164:6379>