Logo, company name

Description automatically generated

**Faculty of Engineering & Applied Science**

**SOFE4630U – Cloud Computing**

**Project Milestone 1 – Data Ingestion Software – Kafka Clusters**

**Smart Meter using Kafka Video**

**Smart Meter using Google PUB/SUB**

**Due Date: 01/25/2023**

|  |  |  |
| --- | --- | --- |
| **First Name** | **Last Name** | **Student ID** |
| Abdul | Bhutta | 100785884 |
| Andrew | Mikaeel | 100525236 |
| Anson | Tu | 100655482 |
| Owen | Musselman | 100657709 |

**Confluent Kafka**

|  |
| --- |
| Graphical user interface, text, application, email  Description automatically generated  Above is producing our first kafka message. |
| Graphical user interface, text, application, email  Description automatically generated  Above is producing more messages. |
| Graphical user interface, text, application, email  Description automatically generated  Above shows that the messages have been consumed and the topic is empty. |
| Graphical user interface, text, application, email  Description automatically generated  Graphical user interface, text, application  Description automatically generated  Above shows changing the topic name (for the cred) and API values. |
| Graphical user interface, text  Description automatically generated  Above shows changing the topic name to testTopic2 for the producer. |
| Text  Description automatically generated    Creating the topic. |
| Text  Description automatically generated  Graphical user interface, application  Description automatically generated  Producing to the new topic and consuming the contents of the topic. |
| Text  Description automatically generated  Creating the smartMeter topic. |
| A screenshot of a computer  Description automatically generated with medium confidence  Above is the output of consumer.py for smartMeter. |
| Text  Description automatically generated  Above is the output of smartMeter.py. |
| Graphical user interface, application  Description automatically generated  Above is the confluent kafka output. |

**Discussion**

**What is EDA? What are its advantages and disadvantages?**

What is an event driven architecture? It is an architecture that makes use of events to allow for communication to take place between decoupled systems/services. For the EDA to work there is the need for three things, a producer (pushes events to the router), router (passes and filters events) and consumers.

Advantages of EDA is that it reduces the cost, as you do not pay for non-stop polling for events. The costs are reduced as there is less hardware utilization, and bandwidth usage, etc. It is easy to develop on as the router pushes to any consumers it needs to and filters events on it’s own. An additional advantage is that services will only know about a router. This allows for the services to be scaled proportionally with the incoming demand. This also allows the service to continue to run properly if an instance malfunctions.

Disadvantages of EDA is that there is a large learning curve to it. Also it is difficult to find the cause of a failure at times as the number of producers and consumers is ever changing. EDA can also have inconsistent behavior, where having an identical event can increasingly make it more troublesome to follow and monitor, which adds time in debugging the system when there is a fault in the system. Error handling can be cumbersome and with the EDA, you may need additional tools in order to efficiently observe errors in the system.

**In Kafka, what’s meant by cluster, broker, topic, replica, partition, zookeeper, controller, leader, consumer, producer, and consumer group?**

|  |  |
| --- | --- |
| **Component** | Description |
| Kafka Cluster | A kafka cluster is a collection of brokers, topics (and partitions). |
| Kafka Broker | Provides the consumer with the ability to get a message through the identification of what topic, partition, and along with the offset. |
| Kafka Topic | A topic is a log of events that are appended only and have an expiration on them (could be seconds or years). Topics are read using the offset. |
| Kafka Replication | Kafka replication is having various copies of the partition managed across various brokers. |
| Kafka Partition | Kafka partitioning breaks up a single topic into various smaller logs, where they can exist on a different node within the cluster. |
| Kafka ZooKeeper | The ZooKeeper is utilized by brokers so that they are able to discern the leader of a partition, it also is responsible for leader election. The ZooKeeper notifies kafka of any changes like broker failure, new or removed topics, etc. |
| Kafka Controller | It is a broker that functions as the controller, meaning it manages the states of each of the partitions, along with the replicas. It is important to know that there can only be a single controller in a cluster. |
| Kafka Leader | Responsible for replication of partitions, and topics. |
| Kafka Consumer | Act like applications where data is fetched from the kafka server where that data is published. Kafka consumers subscribe to the topic they are interested in. |
| Kafka Producer | It is essentially an application that generates the data that is fetched by the consumer. It is more simple than a consumer as group coordination is not necessary. |
| Kafka Consumer Group | A consumer group is really just a group of consumers. If the consumers have the same group id (consumer group) the data will be load balanced amongst all of the instances. |

**Design**

|  |
| --- |
| Text  Description automatically generated |
| Install google cloud pub sub |
| Text  Description automatically generated |
| Subscriber python script |
| Text  Description automatically generated |
| Publisher python script |
| A picture containing text, electronics, green  Description automatically generated |
| Run the publisher script |
| Table  Description automatically generated |
| Verify data on Google Cloud Topic |
| A picture containing text, electronics, green  Description automatically generated |
| Run the subscriber script |
| Graphical user interface, application, Teams  Description automatically generated |
| Verify all messages have been consumed |