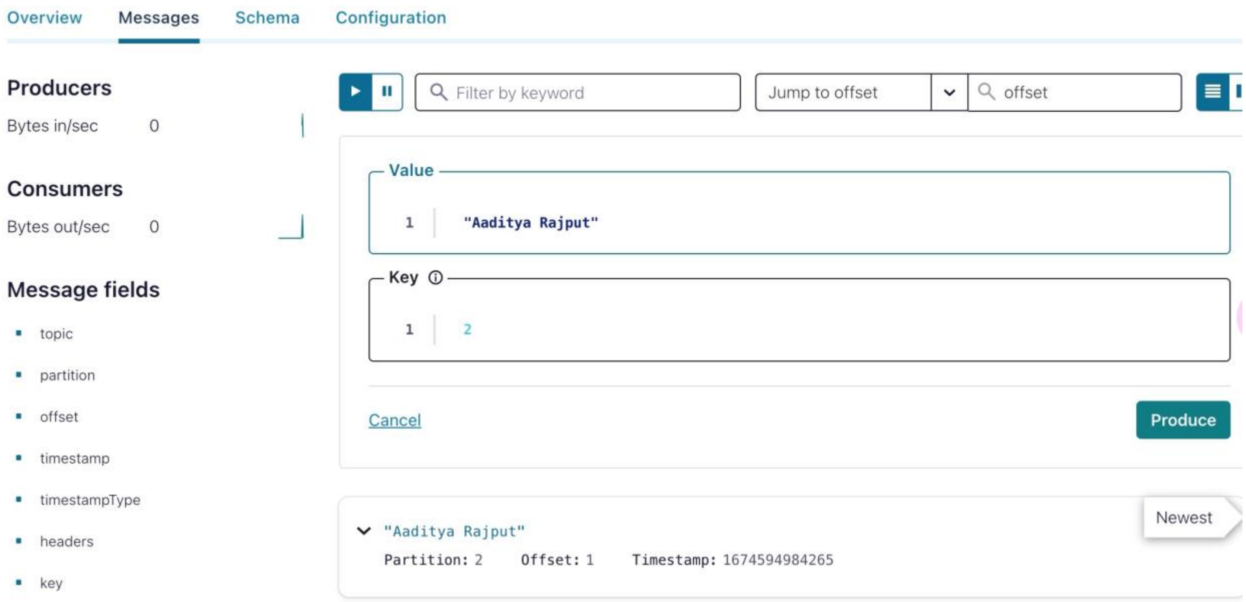
Project Milestone-- Data Ingestion Software-- Kafka Clusters

Aaditya Rajput- 100622434

Produced messages in a TestTopic in the Kafka cluster:



Value

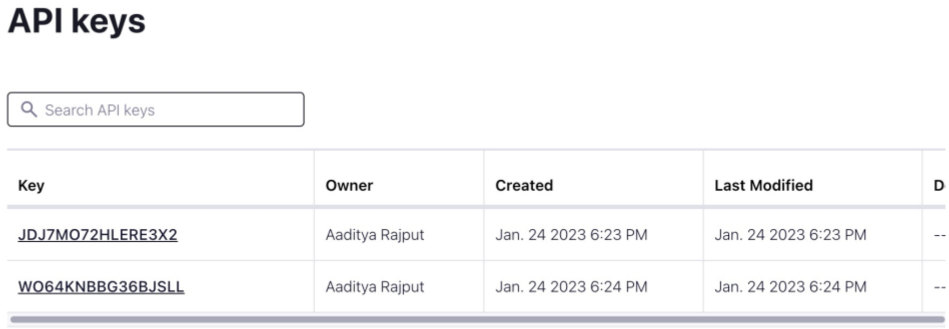
```
1  "Aaditya Rajput"
```

Key

```
1  2
```

```
"Aaditya Rajput"
Partition: 2    Offset: 1    Timestamp: 1674594984265
```

Created API Keys:

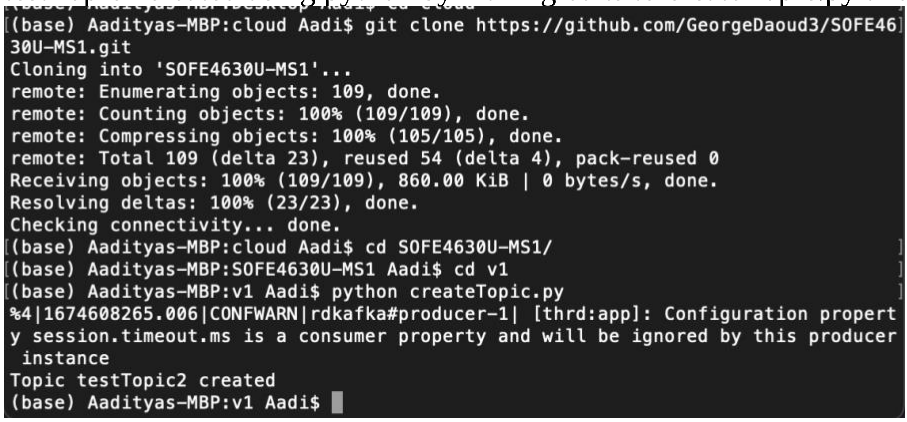


API keys

Key	Owner	Created	Last Modified	D
JDJ7MO72HLERE3X2	Aaditya Rajput	Jan. 24 2023 6:23 PM	Jan. 24 2023 6:23 PM	--
WO64KNBBG36BJSLL	Aaditya Rajput	Jan. 24 2023 6:24 PM	Jan. 24 2023 6:24 PM	--

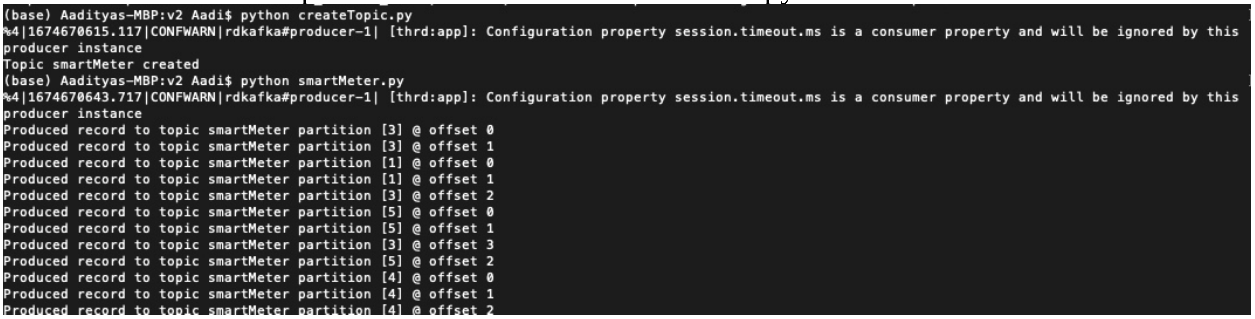
Using python to Pub/Sub

testTopic2 created using python by making edits to createTopic.py and cred.json



```
(base) Aadityas-MBP:cloud Aadi$ git clone https://github.com/GeorgeDaoud3/SOFE46
30U-MS1.git
Cloning into 'SOFE4630U-MS1'...
remote: Enumerating objects: 109, done.
remote: Counting objects: 100% (109/109), done.
remote: Compressing objects: 100% (105/105), done.
remote: Total 109 (delta 23), reused 54 (delta 4), pack-reused 0
Receiving objects: 100% (109/109), 860.00 KiB | 0 bytes/s, done.
Resolving deltas: 100% (23/23), done.
Checking connectivity... done.
(base) Aadityas-MBP:cloud Aadi$ cd SOFE4630U-MS1/
(base) Aadityas-MBP:SOFE4630U-MS1 Aadi$ cd v1
(base) Aadityas-MBP:v1 Aadi$ python createTopic.py
%4|1674608265.006|CONFWARN|rdkafka#producer-1| [thrd:app]: Configuration propert
y session.timeout.ms is a consumer property and will be ignored by this producer
 instance
Topic testTopic2 created
(base) Aadityas-MBP:v1 Aadi$
```

Created smartMeter Topic in v2 folder and ran smartMeter.py:



```
(base) Aadityas-MBP:v2 Aadi$ python createTopic.py
%4|1674670615.117|CONFWARN|rdkafka#producer-1| [thrd:app]: Configuration property session.timeout.ms is a consumer property and will be ignored by this
producer instance
Topic smartMeter created
(base) Aadityas-MBP:v2 Aadi$ python smartMeter.py
%4|1674670643.717|CONFWARN|rdkafka#producer-1| [thrd:app]: Configuration property session.timeout.ms is a consumer property and will be ignored by this
producer instance
Produced record to topic smartMeter partition [3] @ offset 0
Produced record to topic smartMeter partition [3] @ offset 1
Produced record to topic smartMeter partition [1] @ offset 0
Produced record to topic smartMeter partition [1] @ offset 1
Produced record to topic smartMeter partition [3] @ offset 2
Produced record to topic smartMeter partition [5] @ offset 0
Produced record to topic smartMeter partition [5] @ offset 1
Produced record to topic smartMeter partition [3] @ offset 3
Produced record to topic smartMeter partition [5] @ offset 2
Produced record to topic smartMeter partition [4] @ offset 0
Produced record to topic smartMeter partition [4] @ offset 1
Produced record to topic smartMeter partition [4] @ offset 2
```

Discussion:
What is EDA? What are the advantages and disadvantages?

EDA is an acronym for the term event driven architecture. In event driven architecture, events are triggered and used to send messages from one service to another. It is a type of architecture that aids in the communication between micro services. An event can be for example; a state change, an update in the system, an alert, etc. This architecture is on decoupled services publishing and subscribing to messages and events in order to communicate with each other. When an event occurs, the information about that event is sent out to all services that require it. The advantages of EDA include, scalability, granularity, security and loose coupling. The nature of event driven architecture requires the components of the system to be more specific in function. This makes them loosely coupled and therefor one micro service will not rely on another to do its job. This can allow for the micro service to be individually updated and secured without causing grief to the rest of the system. The disadvantages of EDA include greater complexity in developing it and it is harder to debug and configure.

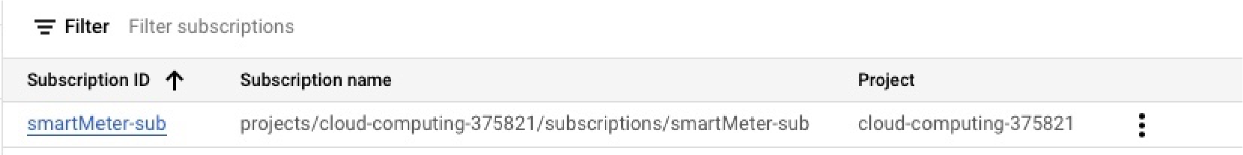
In Kafka, what’s meant by cluster, broker, topic, replica, partition, zookeeper, controller, leader, consumer, producer, and consumer group?

Cluster: A cluster is a group of Kafka nodes, which are multiple systems working together and are composed of the following Kafka nodes; broker, topics, zookeepers, producers and consumers.
Broker: A broker is a node that handles requests from clients. It is in control of topics and message storage and load balancing.
Topic: A topic is a node that is a group of messages under the same topic name. Consumers read from topics and producers write to topics.
Replica: A replica is a replication of a node, or in other words, the copies of the same data across of multiple servers.
Partition: A partition is a partition of topic logs which are distributed over brokers.
Zookeeper: A zookeeper is a node that is control of the data management of the Kafka System. It is usefully for maintaining the names and configuration of the other nodes.
Leader: A leader is the broker node that is in charge of the other brokers and determines the replication of topic partitions.
Controller: A controller is in charge of managing the states of the partitions and replicas and reassigning partitions.
Consumer: A consumer is a node that reads the topics/ messages.

Producer: A producer is a node that writes to the topics/ messages.
Consumer Group: A consumer group is a group consumer that work together to consume some data from each topic. Each consumer in the group consumes a different part of the topic partition.

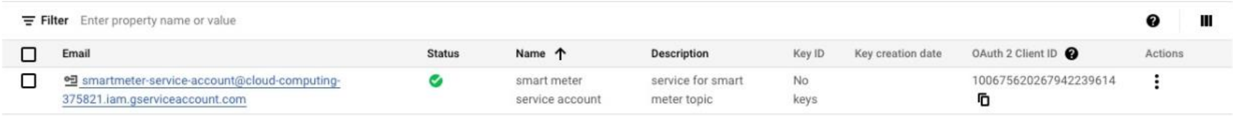
Design:
Google has an alternative ingestion tool called Google Pub/sub. Configure it and create a topic using Cloud interface. write a python code for a Google Pub/sub consumer and producer. The producer and the consumer should act as the smart meter shown before.

Creating topic in GCP:



Subscription ID ↑	Subscription name	Project
smartMeter-sub	projects/cloud-computing-375821/subscriptions/smartMeter-sub	cloud-computing-375821

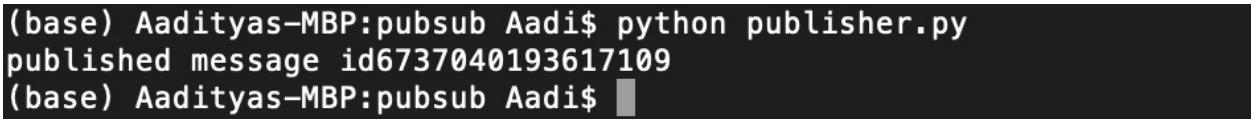
The screenshot also shows a subscription that was automatically created as well.



□	Email	Status	Name ↑	Description	Key ID	Key creation date	OAuth 2 Client ID	Actions
□	smartmeter-service-account@cloud-computing-375821.iam.gserviceaccount.com	✓	smart meter service account	service for smart meter topic	No keys		100675620267942239614	⋮

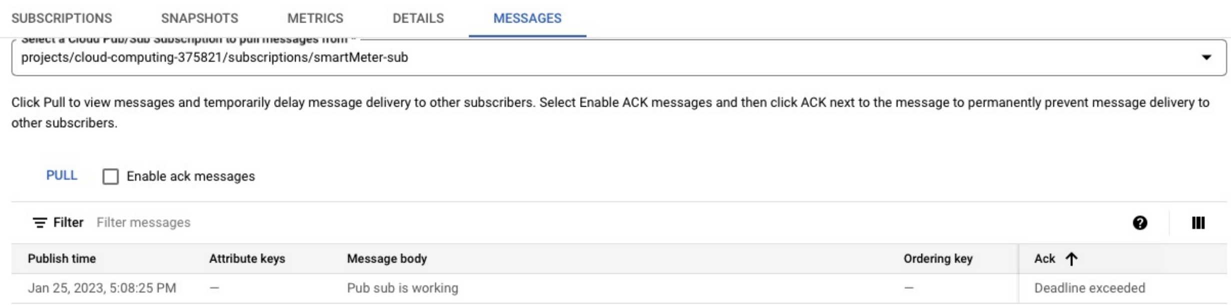
Pub/Sub service account with admin roles.

Created publisher.py which sends a message then ran it:



```
(base) Aadityas-MBP:pubsub Aadi$ python publisher.py
published message id6737040193617109
(base) Aadityas-MBP:pubsub Aadi$ 
```

When I pull messages on the subscription on gcp, it shows the message in my publisher.py file(Pub sub is working):



projects/cloud-computing-375821/subscriptions/smartMeter-sub

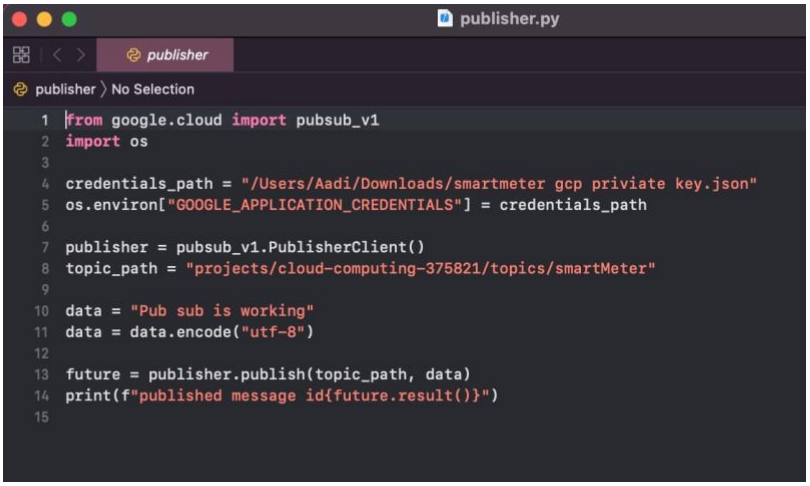
Click Pull to view messages and temporarily delay message delivery to other subscribers. Select Enable ACK messages and then click ACK next to the message to permanently prevent message delivery to other subscribers.

□ Enable ack messages

Publish time	Attribute keys	Message body	Ordering key	Ack ↑
Jan 25, 2023, 5:08:25 PM	–	Pub sub is working	–	Deadline exceeded

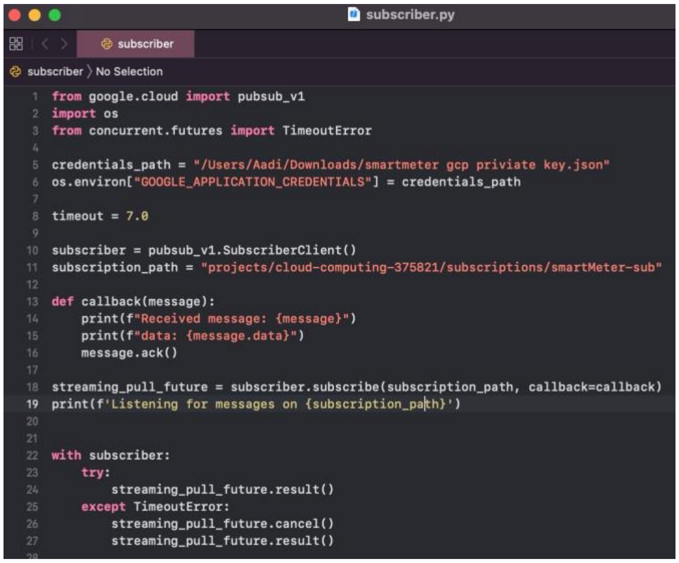
Publisher and Subscriber terminals communicating:

Publisher Code:

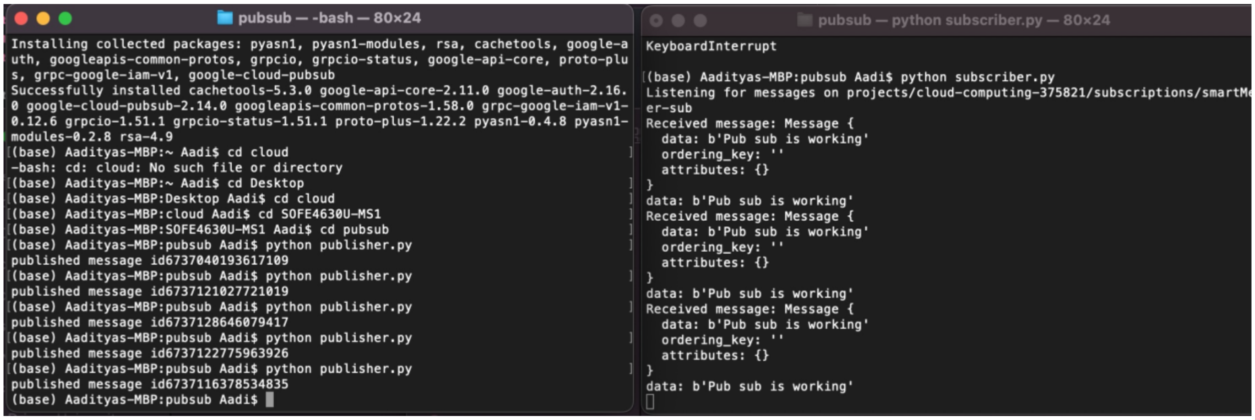


```python
from google.cloud import pubsub_v1
import os

credentials_path = "/Users/Aadi/Downloads/smartmeter gcp priviate key.json"
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = credentials_path

publisher = pubsub_v1.PublisherClient()
topic_path = "projects/cloud-computing-375821/topics/smartMeter"

data = "Pub sub is working"
data = data.encode("utf-8")

future = publisher.publish(topic_path, data)
print(f"published message id{future.result()}")
```

Subscriber Code:



```python
from google.cloud import pubsub_v1
import os
from concurrent.futures import TimeoutError

credentials_path = "/Users/Aadi/Downloads/smartmeter gcp priviate key.json"
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = credentials_path

timeout = 7.0

subscriber = pubsub_v1.SubscriberClient()
subscription_path = "projects/cloud-computing-375821/subscriptions/smartMeter-sub"

def callback(message):
    print(f"Received message: {message}")
    print(f"data: {message.data}")
    message.ack()

streaming_pull_future = subscriber.subscribe(subscription_path, callback=callback)
print(f'Listening for messages on {subscription_path}')


with subscriber:
    try:
        streaming_pull_future.result()
    except TimeoutError:
        streaming_pull_future.cancel()
        streaming_pull_future.result()
```



```
Installing collected packages: pyasn1, pyasn1-modules, rsa, cachetools, google-a
uth, googleapis-common-protos, grpcio, grpcio-status, google-api-core, proto-plu
s, grpc-google-iam-v1, google-cloud-pubsub
Successfully installed cachetools-5.3.0 google-api-core-2.11.0 google-auth-2.16.
0 google-cloud-pubsub-2.14.0 googleapis-common-protos-1.58.0 grpc-google-iam-v1-
0.12.6 grpcio-1.51.1 grpcio-status-1.51.1 proto-plus-1.22.2 pyasn1-0.4.8 pyasn1-
modules-0.2.8 rsa-4.9
(base) Aadityas-MBP:~ Aadi$ cd cloud
-bash: cd: cloud: No such file or directory
(base) Aadityas-MBP:~ Aadi$ cd Desktop
(base) Aadityas-MBP:Desktop Aadi$ cd cloud
(base) Aadityas-MBP:cloud Aadi$ cd SOFE4630U-MS1
(base) Aadityas-MBP:SOFE4630U-MS1 Aadi$ cd pubsub
(base) Aadityas-MBP:pubsub Aadi$ python publisher.py
published message id6737040193617109
(base) Aadityas-MBP:pubsub Aadi$ python publisher.py
published message id6737121027721019
(base) Aadityas-MBP:pubsub Aadi$ python publisher.py
published message id6737128646079417
(base) Aadityas-MBP:pubsub Aadi$ python publisher.py
published message id6737122775963926
(base) Aadityas-MBP:pubsub Aadi$ python publisher.py
published message id6737116378534835
(base) Aadityas-MBP:pubsub Aadi$ █
```

```
KeyboardInterrupt

(base) Aadityas-MBP:pubsub Aadi$ python subscriber.py
Listening for messages on projects/cloud-computing-375821/subscriptions/smartMe
er-sub
Received message: Message {
  data: b'Pub sub is working'
  ordering_key: ''
  attributes: {}
}
data: b'Pub sub is working'
Received message: Message {
  data: b'Pub sub is working'
  ordering_key: ''
  attributes: {}
}
data: b'Pub sub is working'
Received message: Message {
  data: b'Pub sub is working'
  ordering_key: ''
  attributes: {}
}
data: b'Pub sub is working'
▯
```

As you can see the terminal window on the left is the publisher which is writing messages and the subscriber window on the right is subscribing to them.

Video Links:

SmartMeter using Confluent:

https://youtu.be/VcCwp8XnRek

Pub Sub:

https://youtu.be/FxNY3PULDCw