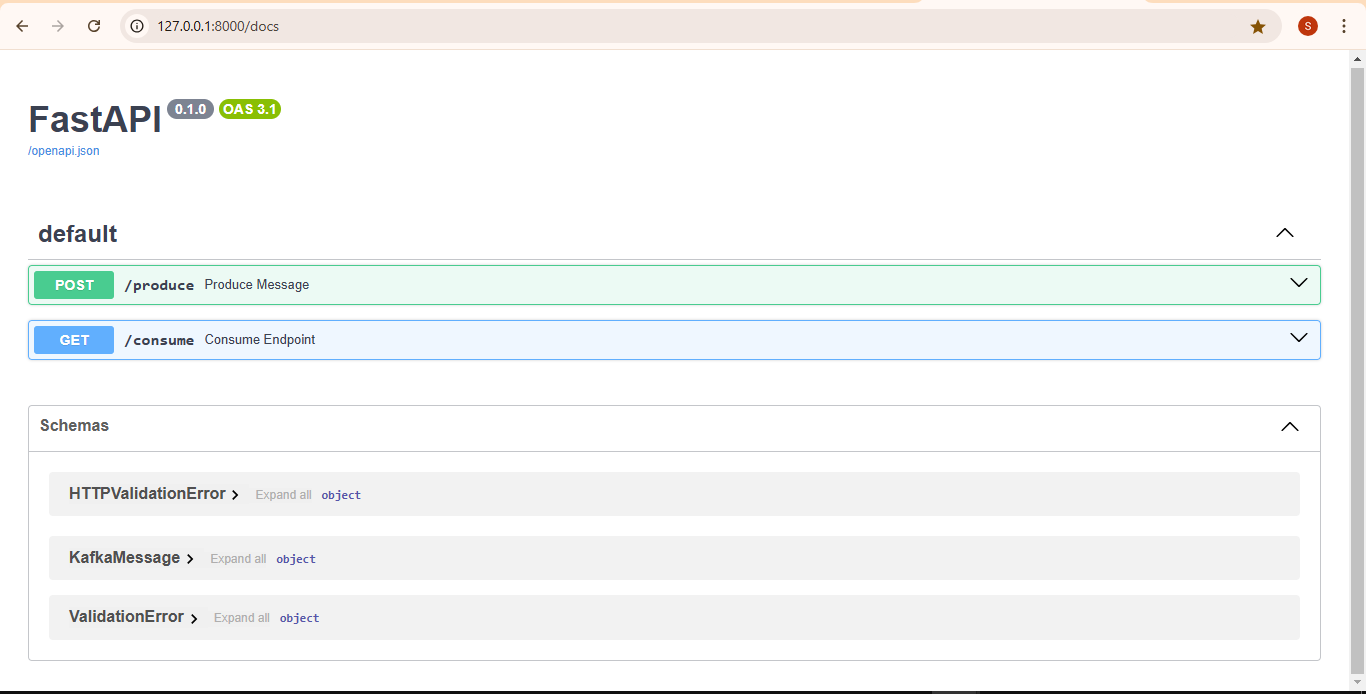
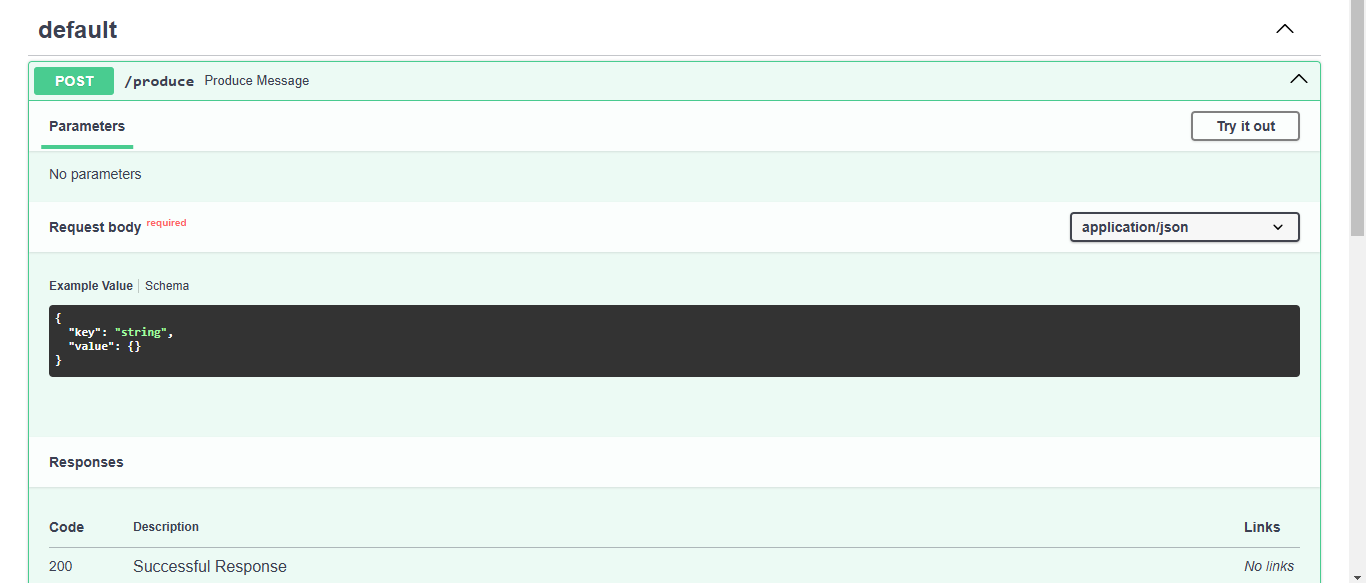
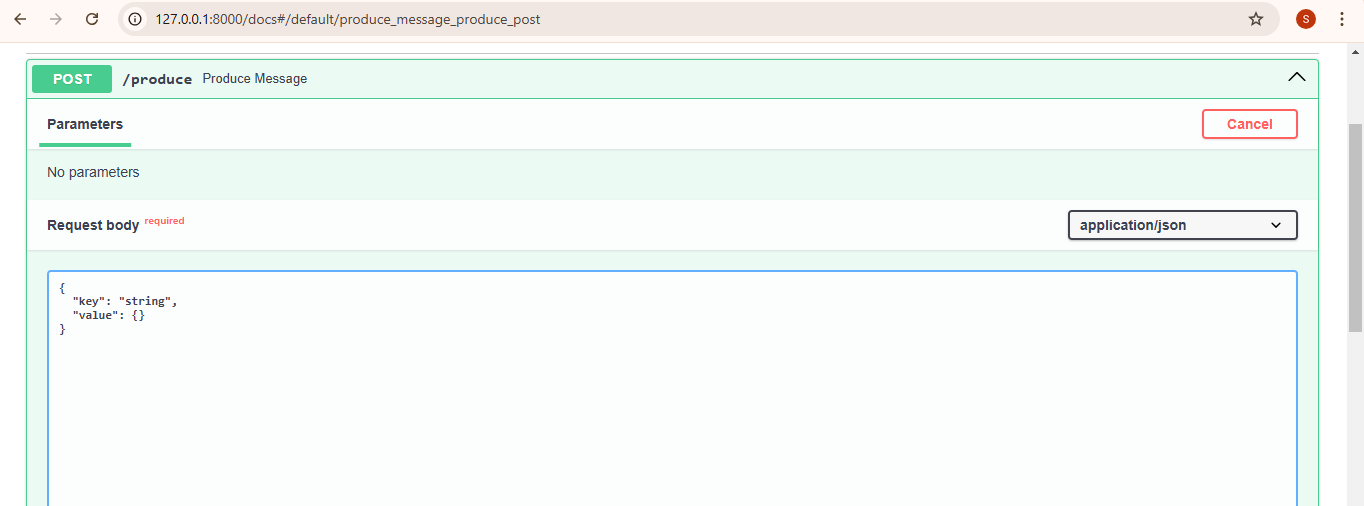
<http://127.0.0.1:8000/docs>







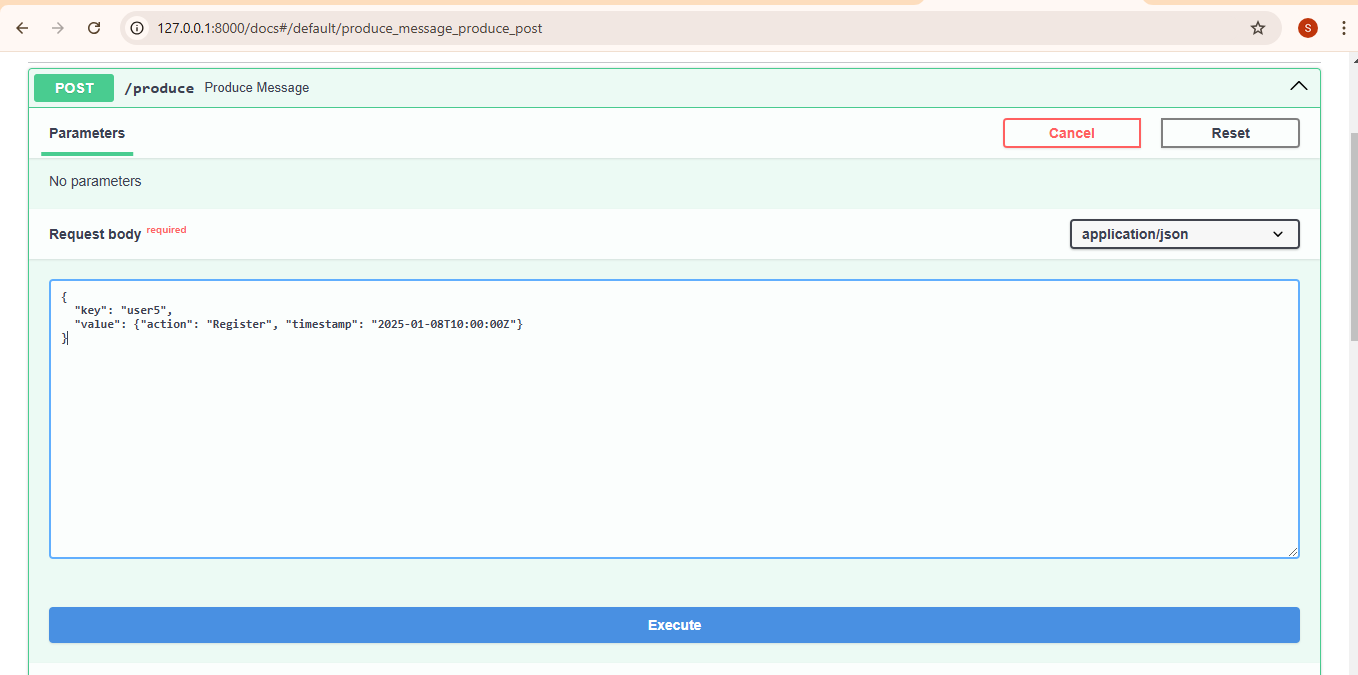
**Request** -

{

"key": "user5",

"value": {"action": "Register", "timestamp": "2025-01-08T10:00:00Z"}

}



##### Response body

**{**

**"status": "success",**

**"message": {**

**"key": "user5",**

**"value": {**

**"action": "Register",**

**"timestamp": "2025-01-08T10:00:00Z"**

**}**

**}**

**}**

**Log** –

D:\Python FAST API\Kafka\KeybasedPartitioning> **uvicorn main:app --reload**

INFO: Will watch for changes in these directories: ['D:\\Python FAST API\\Kafka\\KeybasedPartitioning']

INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)

INFO: Started reloader process [20376] using WatchFiles

INFO: Started server process [19248]

INFO: Waiting for application startup.

INFO:logger:Application is starting.

INFO: Application startup complete.

INFO: 127.0.0.1:56797 - "GET /docs HTTP/1.1" 200 OK

INFO: 127.0.0.1:56797 - "GET /openapi.json HTTP/1.1" 200 OK

INFO:Kafka.producer:Kafka Producer started.

INFO:Kafka.producer:Message produced to topic :'TestforKeybasedpartition' Partition : 'RecordMetadata(topic='TestforKeybasedpartition', partition=2, topic\_partition=TopicPartition(topic='TestforKeybasedpartition', **partition=2**), offset=0, timestamp=1736421322809, timestamp\_type=0, log\_start\_offset=0)': key=**'user5'** value={'action': 'Register', 'timestamp': '2025-01-08T10:00:00Z'}

**Request** –

{

"key": "user5",

"value": {"action": "login", "timestamp": "2025-01-08T10:00:00Z"}

}

##### Response body

**{**

**"status": "success",**

**"message": {**

**"key": "user5",**

**"value": {**

**"action": "login",**

**"timestamp": "2025-01-08T10:00:00Z"**

**}**

**}**

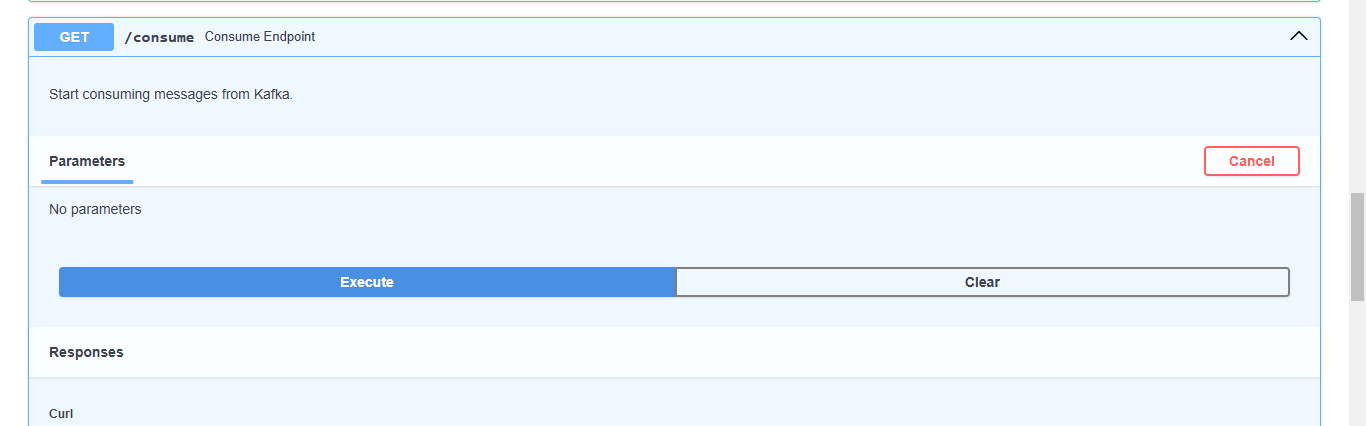
**}**

INFO:Kafka.producer:Message produced to topic :'TestforKeybasedpartition' Partition : 'RecordMetadata(topic='TestforKeybasedpartition', partition=2, topic\_partition=TopicPartition(topic='TestforKeybasedpartition'**, partition=2**), offset=1, timestamp=1736421530618, timestamp\_type=0, log\_start\_offset=0)': key=**'user5'** value={'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

INFO: 127.0.0.1:56869 - "POST /produce HTTP/1.1" 200 OK

**Consumer** –

**http://127.0.0.1:8000/consume**



##### Response body

Download

**{**

**"status": "success",**

**"message": "Consuming messages in the background."**

**}**

**Log**-

INFO: 127.0.0.1:56869 - "POST /produce HTTP/1.1" 200 OK

INFO: 127.0.0.1:56885 - "GET /consume HTTP/1.1" 200 OK

INFO:aiokafka.consumer.subscription\_state:Updating subscribed topics to: frozenset({'TestforKeybasedpartition'})

INFO:aiokafka.consumer.group\_coordinator:Discovered coordinator 0 for group KeybasedPartitioning-group

INFO:aiokafka.consumer.group\_coordinator:Revoking previously assigned partitions set() for group KeybasedPartitioning-group

INFO:aiokafka.consumer.group\_coordinator:(Re-)joining group KeybasedPartitioning-group

INFO:aiokafka.consumer.group\_coordinator:Joined group 'KeybasedPartitioning-group' (generation 1) with member\_id aiokafka-0.12.0-dc07e986-5bb7-4c94-8308-6c0ec88a4fe3

INFO:aiokafka.consumer.group\_coordinator:Elected group leader -- performing partition assignments using roundrobin

INFO:aiokafka.consumer.group\_coordinator:Successfully synced group KeybasedPartitioning-group with generation 1

**INFO:aiokafka.consumer.group\_coordinator:Setting newly assigned partitions {TopicPartition(topic='TestforKeybasedpartition', partition=3), TopicPartition(topic='TestforKeybasedpartition', partition=0), TopicPartition(topic='TestforKeybasedpartition', partition=1),** **TopicPartition(topic='TestforKeybasedpartition', partition=2)} for group KeybasedPartitioning-group**

INFO:Kafka.consumer:Kafka Consumer started.

INFO:Kafka.consumer:Consumed: {'action': 'Register', 'timestamp': '2025-01-08T10:00:00Z'} from partition 2

INFO:logger:Processed message: {'action': 'Register', 'timestamp': '2025-01-08T10:00:00Z'}

INFO:Kafka.consumer:Consumed: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'} from partition 2

INFO:logger:Processed message: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

**Produce again** –

INFO:Kafka.producer:Message produced to topic :'TestforKeybasedpartition' Partition : 'RecordMetadata(topic='TestforKeybasedpartition', **partition=2**, topic\_partition=TopicPartition(topic='TestforKeybasedpartition', partition=2), offset=2, timestamp=1736421906306, timestamp\_type=0, log\_start\_offset=0)': key=**'user1'** value={'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

INFO: 127.0.0.1:56928 - "POST /produce HTTP/1.1" 200 OK

INFO:Kafka.consumer:Consumed: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'} from partition 2

INFO:logger:Processed message: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

INFO:Kafka.producer:Message produced to topic :'TestforKeybasedpartition' Partition : 'RecordMetadata(topic='TestforKeybasedpartition', partition=1, topic\_partition=TopicPartition(topic='TestforKeybasedpartition', **partition=1**), offset=0, timestamp=1736422016985, timestamp\_type=0, log\_start\_offset=0)': key=**'user2'** value={'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

INFO: 127.0.0.1:56944 - "POST /produce HTTP/1.1" 200 OK

INFO:Kafka.consumer:Consumed: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'} from partition 1

INFO:logger:Processed message: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

INFO:Kafka.producer:Message produced to topic :'TestforKeybasedpartition' Partition : 'RecordMetadata(topic='TestforKeybasedpartition', partition=3, topic\_partition=TopicPartition(topic='TestforKeybasedpartition'**, partition=3**), offset=0, timestamp=1736422091519, timestamp\_type=0, log\_start\_offset=0)': key=**'user3'** value={'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

INFO: 127.0.0.1:57253 - "POST /produce HTTP/1.1" 200 OK

INFO:Kafka.consumer:Consumed: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'} from partition 3

INFO:logger:Processed message: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

INFO:Kafka.producer:Message produced to topic :'TestforKeybasedpartition' Partition : 'RecordMetadata(topic='TestforKeybasedpartition', **partition=2**, topic\_partition=TopicPartition(topic='TestforKeybasedpartition', partition=2), offset=3, timestamp=1736422163886, timestamp\_type=0, log\_start\_offset=0)': key=**'user5'** value={'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

INFO: 127.0.0.1:57282 - "POST /produce HTTP/1.1" 200 OK

INFO:Kafka.consumer:Consumed: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'} from partition 2

INFO:logger:Processed message: {'action': 'login', 'timestamp': '2025-01-08T10:00:00Z'}

Key-based partitioning in Kafka allows you to send messages with the same key to the same partition. This can be useful for ensuring data locality, ordering, or grouping similar messages.

To implement key-based partitioning in Kafka using **AIOKafka**, you need to set a key for each message when producing. Kafka uses the key to determine the partition by applying a hash function.

### ****Steps to Implement Key-Based Partitioning in AIOKafka****

1. **Install AIOKafka**: Make sure you have the library installed:

bash

Copy code

pip install aiokafka

1. **Setup Kafka Producer**: Use AIOKafkaProducer and include the key in the send method.
2. **Partitioning Logic**: Kafka uses the hash of the key and the number of partitions to decide the partition.

### ****Example Code****

import asyncio

from aiokafka import AIOKafkaProducer

# Kafka configuration

KAFKA\_BOOTSTRAP\_SERVERS = "localhost:9092"

TOPIC\_NAME = "example\_topic"

async def produce\_messages():

# Create a Kafka producer instance

producer = AIOKafkaProducer(

bootstrap\_servers=KAFKA\_BOOTSTRAP\_SERVERS

)

# Start the producer

await producer.start()

try:

for i in range(10):

key = f"key-{i % 3}".encode("utf-8") # Key determines partition

value = f"message-{i}".encode("utf-8")

# Send message with key

partition = await producer.send\_and\_wait(TOPIC\_NAME, value=value, key=key)

print(f"Sent: {value.decode()} to partition {partition}")

finally:

# Ensure the producer is closed

await producer.stop()

# Run the producer

asyncio.run(produce\_messages())

### ****Explanation****

1. **Key-Based Partitioning**:
   * The key argument in producer.send\_and\_wait is hashed by Kafka to determine the partition.
   * Messages with the same key are sent to the same partition, ensuring ordering within that key.
2. **Example Output**: Assuming the topic has 3 partitions:

Sent: message-0 to partition 1

Sent: message-1 to partition 2

Sent: message-2 to partition 0

Sent: message-3 to partition 1

Sent: message-4 to partition 2

Sent: message-5 to partition 0

**Partition Count**: If the number of partitions changes (e.g., through topic reconfiguration), the hash-to-partition mapping may change, leading to re-balancing.

### ****Consumer Handling****

To consume messages in the same partition, use a consumer group. Kafka ensures that only one consumer in the group reads from a specific partition.

from aiokafka import AIOKafkaConsumer

async def consume\_messages():

consumer = AIOKafkaConsumer(

TOPIC\_NAME,

bootstrap\_servers=KAFKA\_BOOTSTRAP\_SERVERS,

group\_id="example\_group"

)

await consumer.start()

try:

async for msg in consumer:

print(f"Consumed: {msg.value.decode()} from partition {msg.partition}")

finally:

await consumer.stop()

asyncio.run(consume\_messages())

### ****Advantages of Key-Based Partitioning****

1. **Message Ordering**: Messages with the same key are delivered in order within a partition.
2. **Data Grouping**: Group related messages (e.g., user activity logs, device telemetry) by key.
3. **Efficient Processing**: Consumers can process specific partitions independently.

**Implement key based partitioning using AIOKafka for sending json messages**

Here’s how you can implement **key-based partitioning** using **AIOKafka** to send JSON messages to Kafka. This approach ensures that messages with the same key are sent to the same partition.

**Steps**

1. **Serialize the Message**: Use the json module to serialize your data into JSON format before sending it to Kafka.
2. **Specify the Key**: The key is used for partitioning. Kafka uses the hash of the key and the number of partitions to decide the partition.
3. **Send JSON Messages**: Use the send\_and\_wait method to send messages to a Kafka topic, passing the key and the serialized JSON data.

**Example Implementation**

import asyncio

import json

from aiokafka import AIOKafkaProducer

# Kafka configuration

KAFKA\_BOOTSTRAP\_SERVERS = "localhost:9092"

TOPIC\_NAME = "json\_topic"

async def produce\_json\_messages():

# Create a Kafka producer instance

producer = AIOKafkaProducer(

bootstrap\_servers=KAFKA\_BOOTSTRAP\_SERVERS

)

# Start the producer

await producer.start()

try:

# Example messages to send

messages = [

{"id": 1, "name": "Alice", "action": "login"},

{"id": 2, "name": "Bob", "action": "logout"},

{"id": 1, "name": "Alice", "action": "purchase"},

{"id": 3, "name": "Charlie", "action": "signup"},

{"id": 2, "name": "Bob", "action": "login"},

]

for msg in messages:

key = str(msg["id"]).encode("utf-8") # Use 'id' as the key

value = json.dumps(msg).encode("utf-8") # Serialize message to JSON

# Send the message to Kafka

partition = await producer.send\_and\_wait(TOPIC\_NAME, value=value, key=key)

print(f"Sent: {msg} to partition {partition}")

finally:

# Ensure the producer is closed

await producer.stop()

# Run the producer

asyncio.run(produce\_json\_messages())

**Explanation**

1. **Key-Based Partitioning**:
   * The key (msg["id"]) is hashed to determine the partition.
   * Messages with the same id are always sent to the same partition.
2. **JSON Serialization**:
   * The json.dumps() function serializes Python dictionaries to JSON strings.
   * Kafka requires byte data, so the serialized JSON is encoded to bytes using .encode("utf-8").
3. **Example Messages**:
   * Example messages contain an id, name, and action.
4. **Output Example**: Assuming the topic has 3 partitions, the output might look like:

Sent: {'id': 1, 'name': 'Alice', 'action': 'login'} to partition 1

Sent: {'id': 2, 'name': 'Bob', 'action': 'logout'} to partition 0

Sent: {'id': 1, 'name': 'Alice', 'action': 'purchase'} to partition 1

Sent: {'id': 3, 'name': 'Charlie', 'action': 'signup'} to partition 2

Sent: {'id': 2, 'name': 'Bob', 'action': 'login'} to partition 0

### ****Consumer Example****

Here’s how you can consume and deserialize the JSON messages:

from aiokafka import AIOKafkaConsumer

async def consume\_json\_messages():

consumer = AIOKafkaConsumer(

TOPIC\_NAME,

bootstrap\_servers=KAFKA\_BOOTSTRAP\_SERVERS,

group\_id="json\_group"

)

await consumer.start()

try:

async for msg in consumer:

value = json.loads(msg.value.decode("utf-8")) # Deserialize JSON

print(f"Consumed: {value} from partition {msg.partition}")

finally:

await consumer.stop()

asyncio.run(consume\_json\_messages())

### ****Considerations****

1. **Error Handling**: Add robust error handling to retry failed messages or log errors.
2. **Partition Consistency**: Adding or removing partitions to the topic can change the hash-to-partition mapping.
3. **Message Size**: Kafka has a default maximum message size of 1 MB. Ensure your JSON messages don't exceed this limit.
4. **Batching**: Consider using batching for improved performance by sending multiple messages in a single request.

This implementation ensures that messages with the same key (id) are sent to the same partition, and the JSON data is properly serialized and consumed. Let me know if you need further assistance!