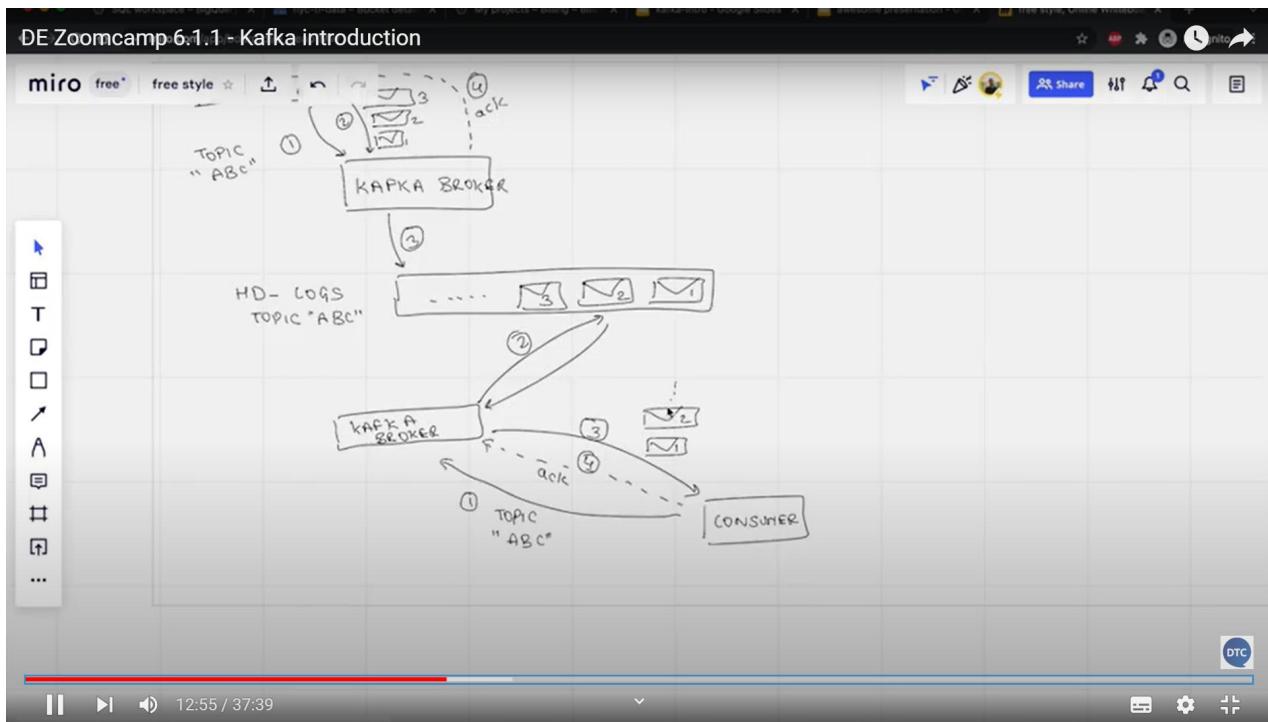
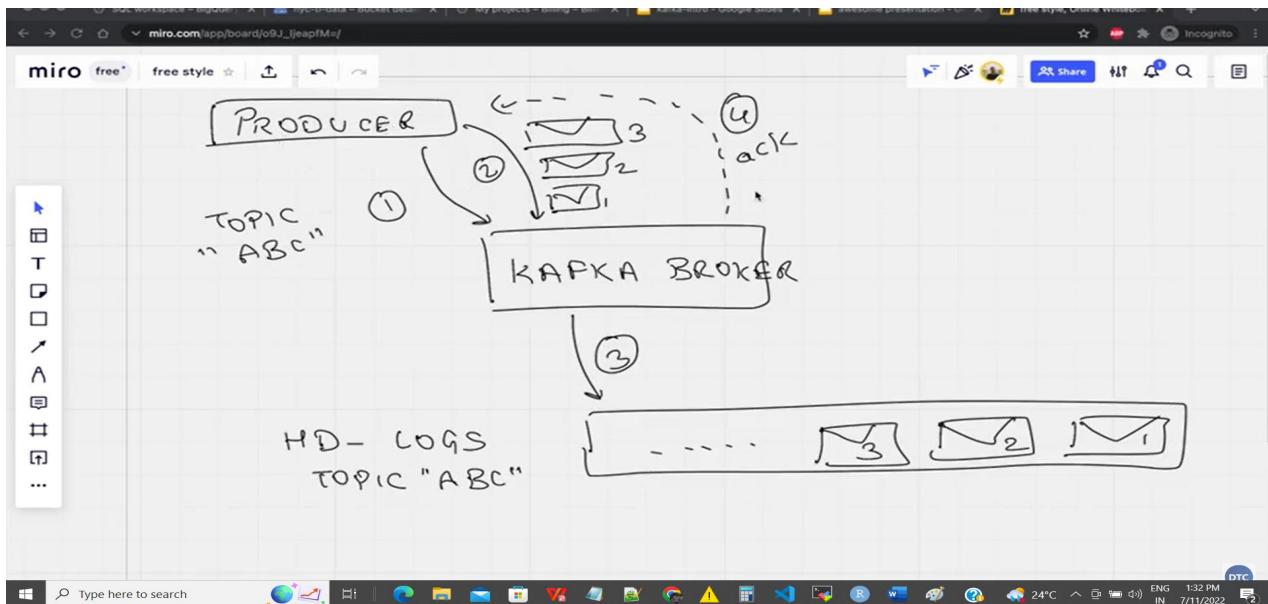
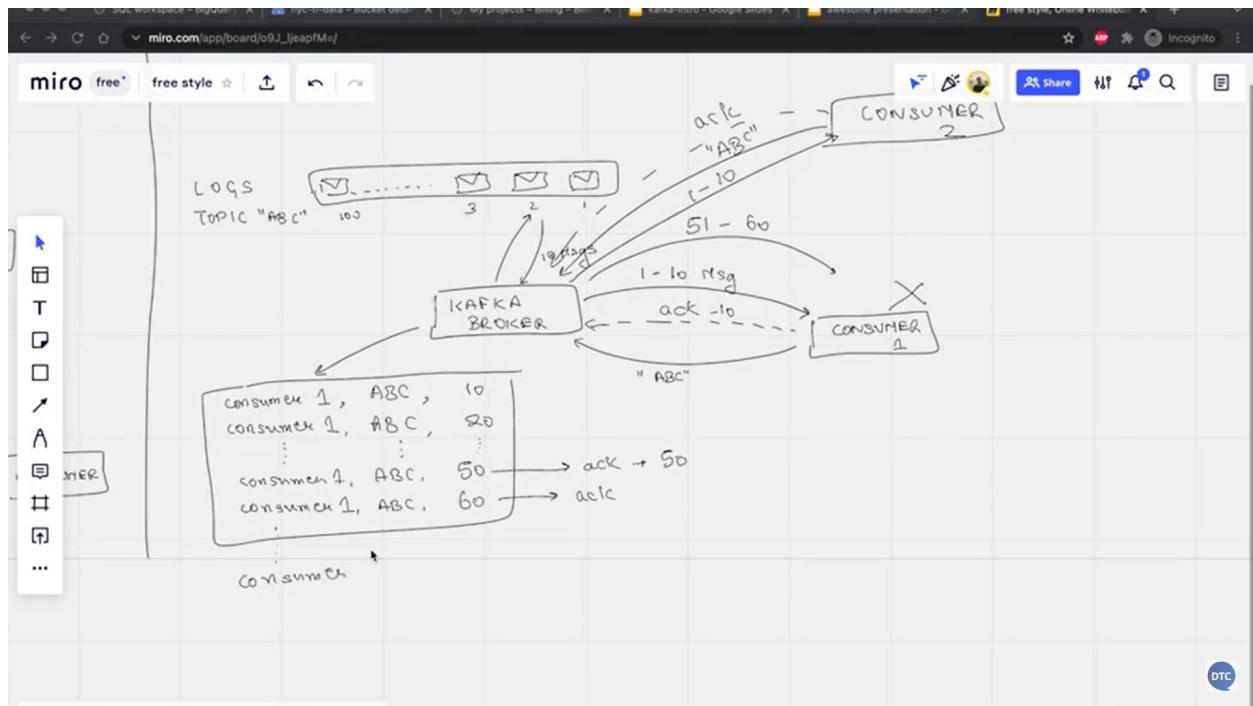
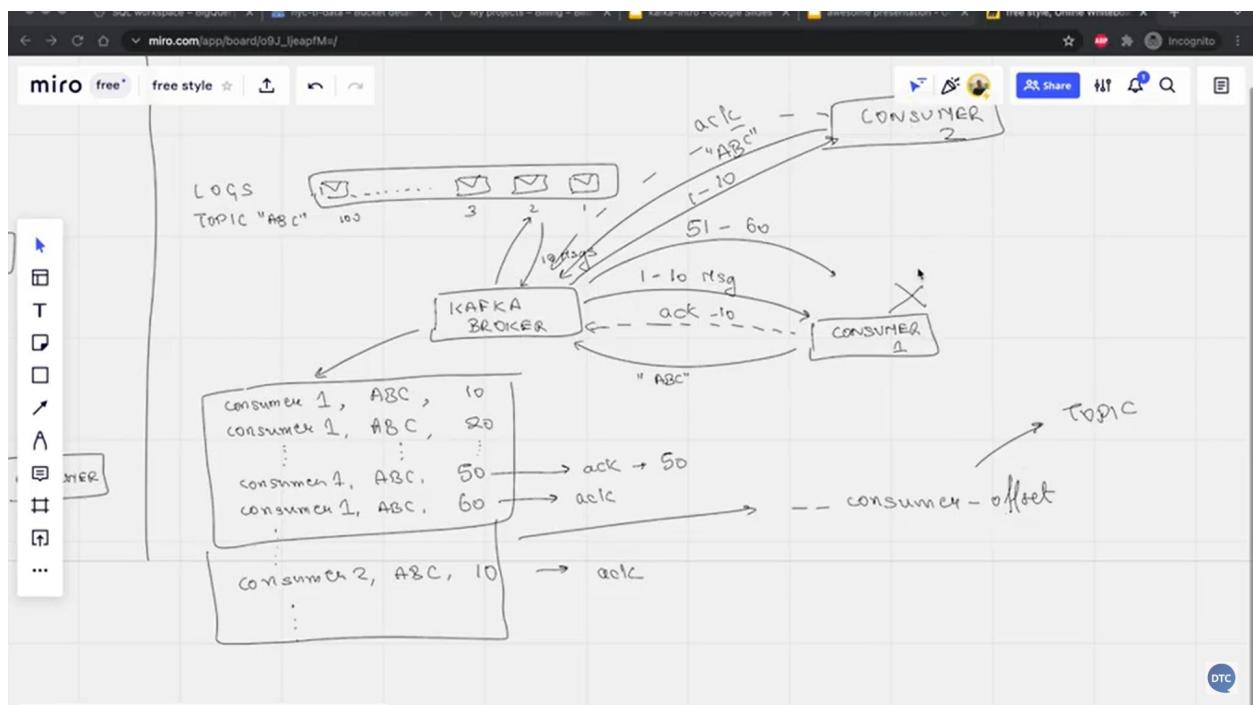


Introduction to Kafka



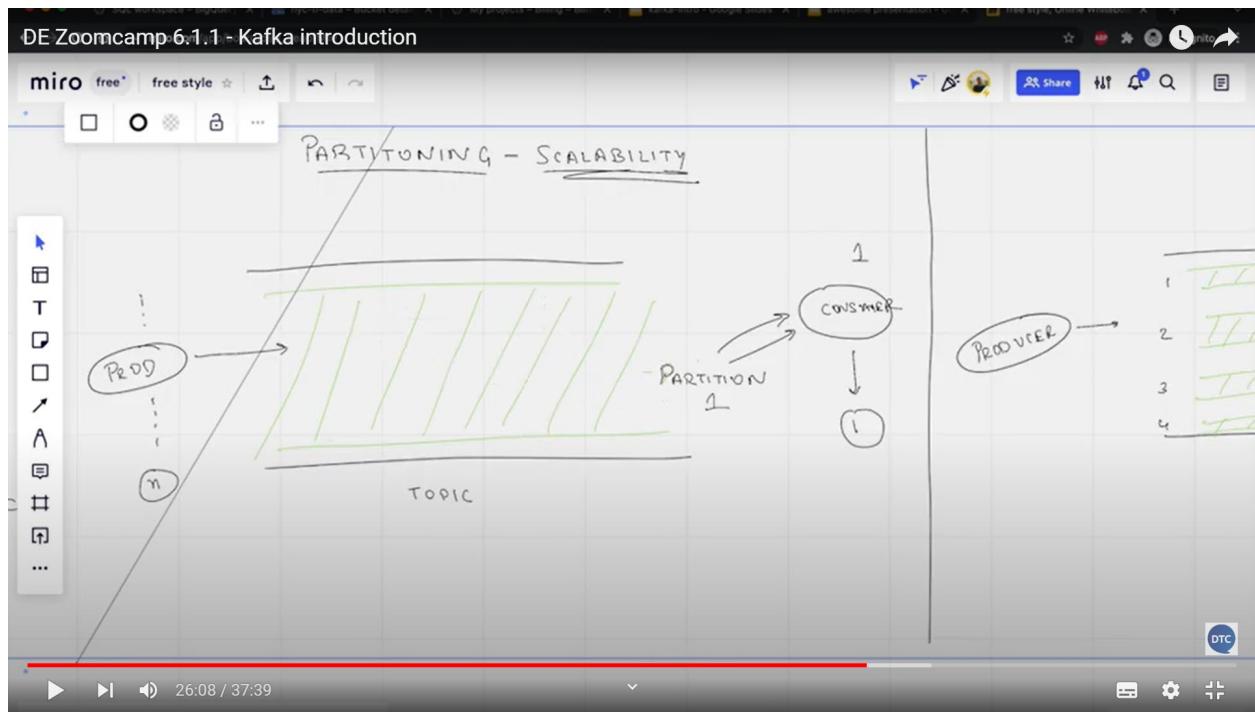


Consumer offset was maintained by kafka to know about num. of msg's read by each consumer. If a consumer dies and comes back, then it starts reading messages from where it left before dies.

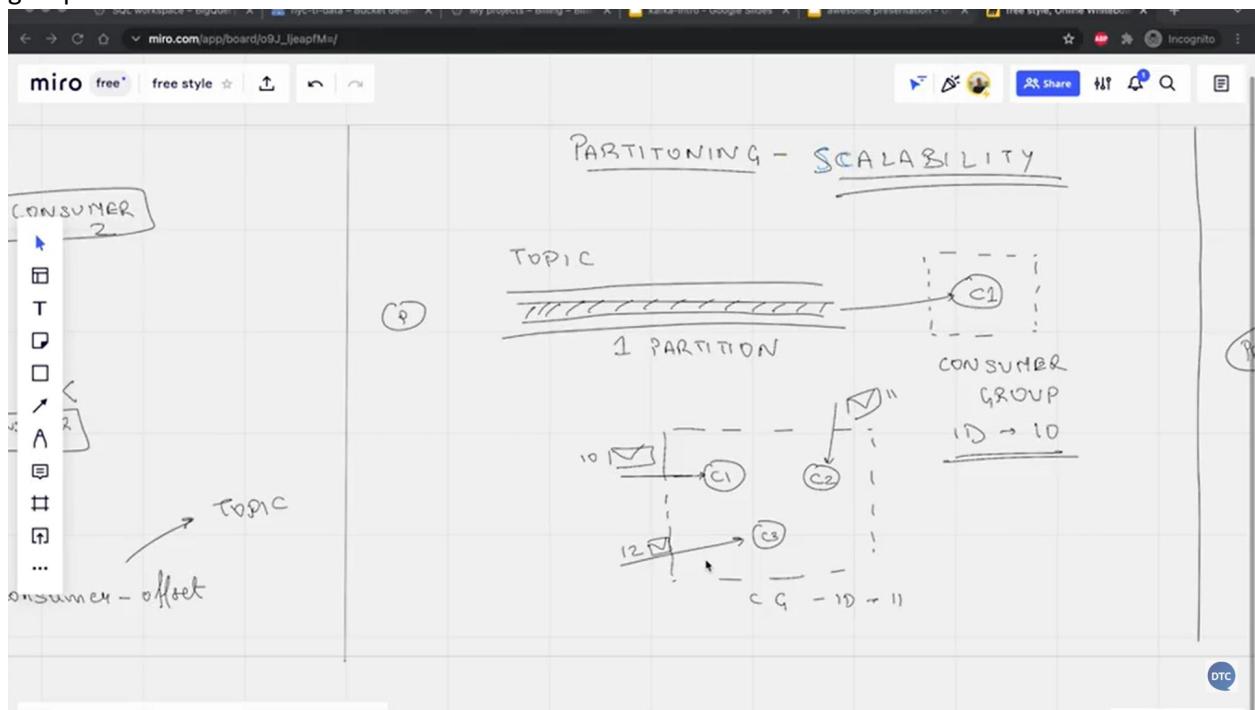


Kafka Partition: kafka partition happens over real time data to accomplish scalability.

1-producer-1partition-1consumer

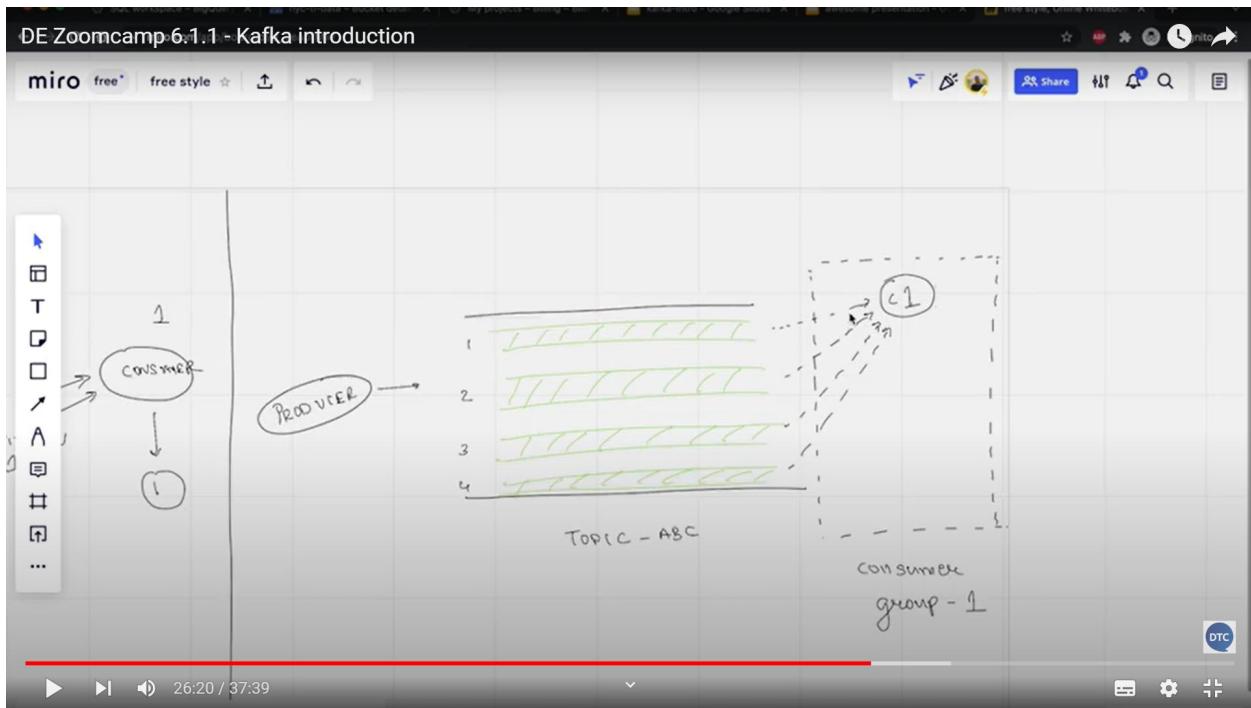


Consumer group: A consumer group can consist of multiple consumers, but only one consumer can receive a msg from kafka-broker for the subscribed topic. Once a consumer from a group received msg from broker the same can't be received by other consumer in the group.

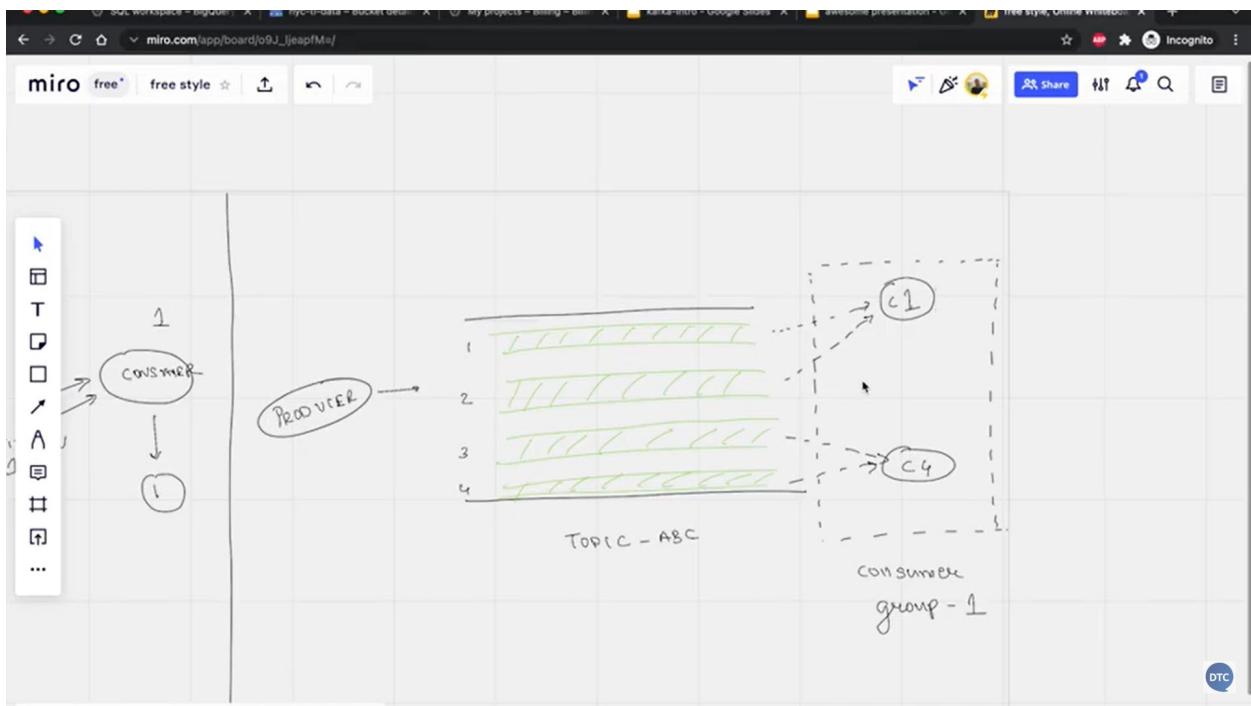


Producer pushing messages to a topic that consist of 4 partitions.

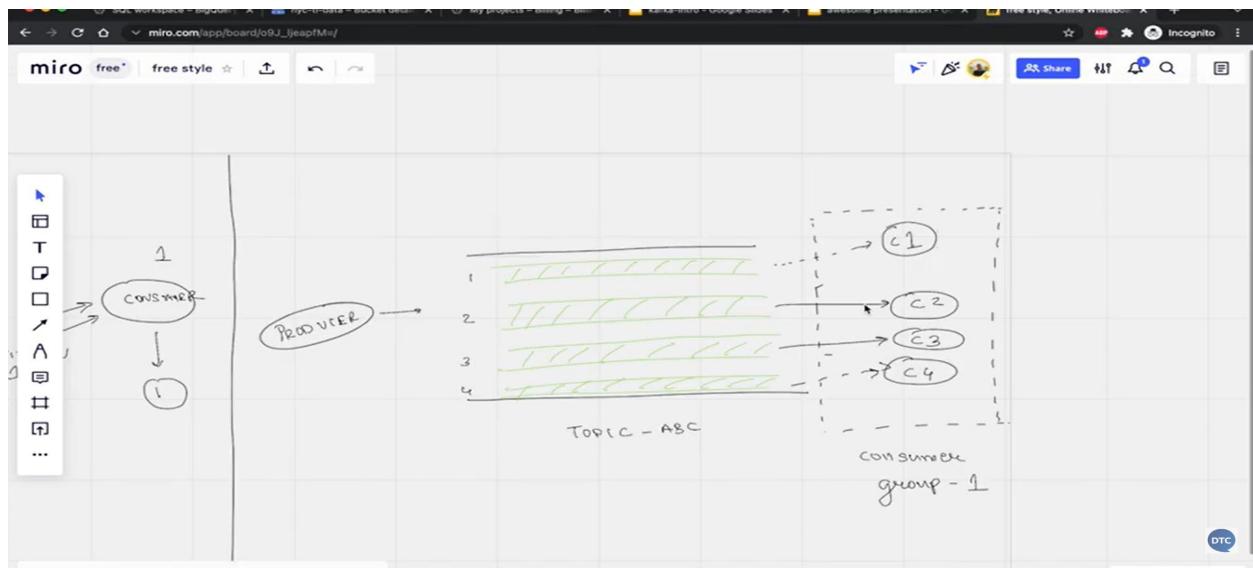
1-producer-4partition-1consumer group(1consumer)



1-producer-4partition-1consumer group(2consumers) 2 partition gets assigned to consumer1 and another 2 partition gets assigned to consumer2. Here reading the msg twice fast as earlier case.

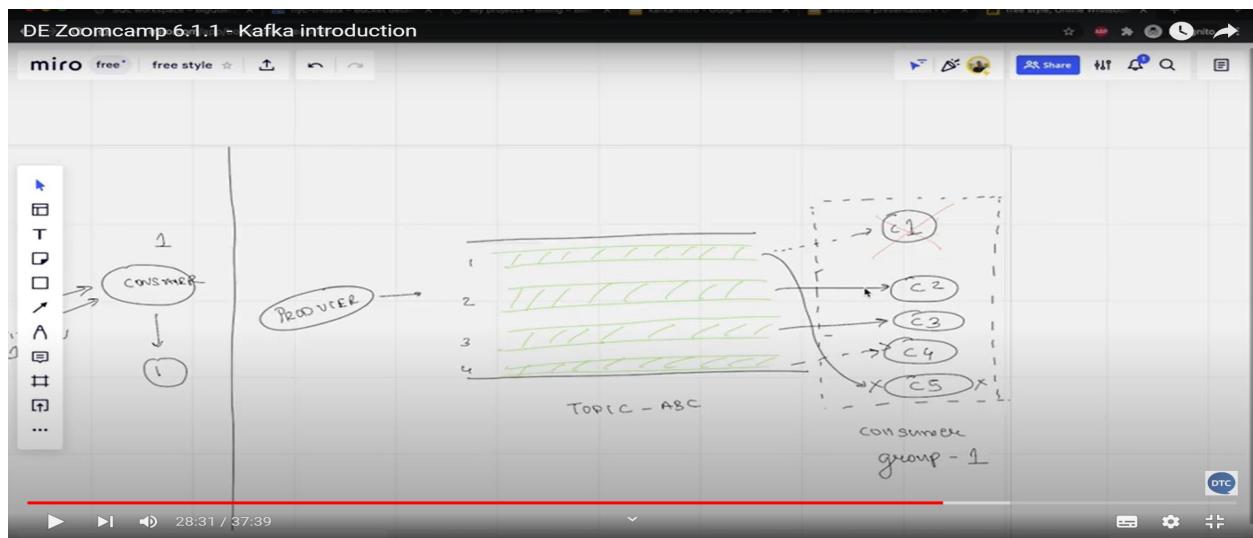


If 4 consumers were introduced in a consumer group, each 1 partition is assigned to each consumer in a group. Here the reading is 4 times speed as compare to 1-consumer case.



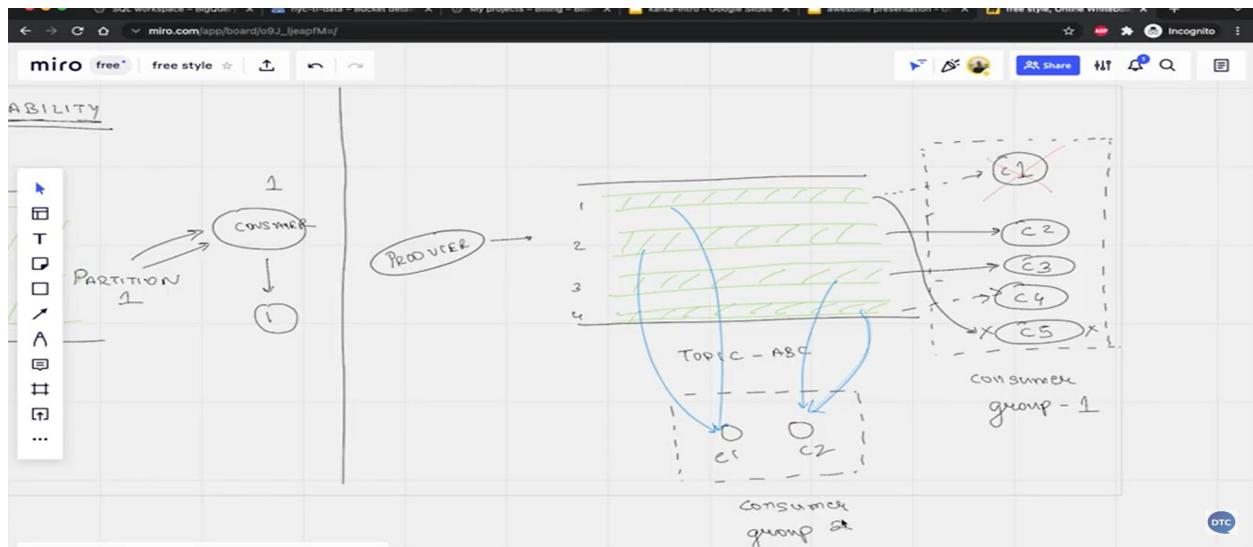
Number of consumers = Partition size+1

good practice to create a consumer greater than one partition size. Additional consumer keeps idle, if any one of a consumer in a group fails this consumer take over that job.



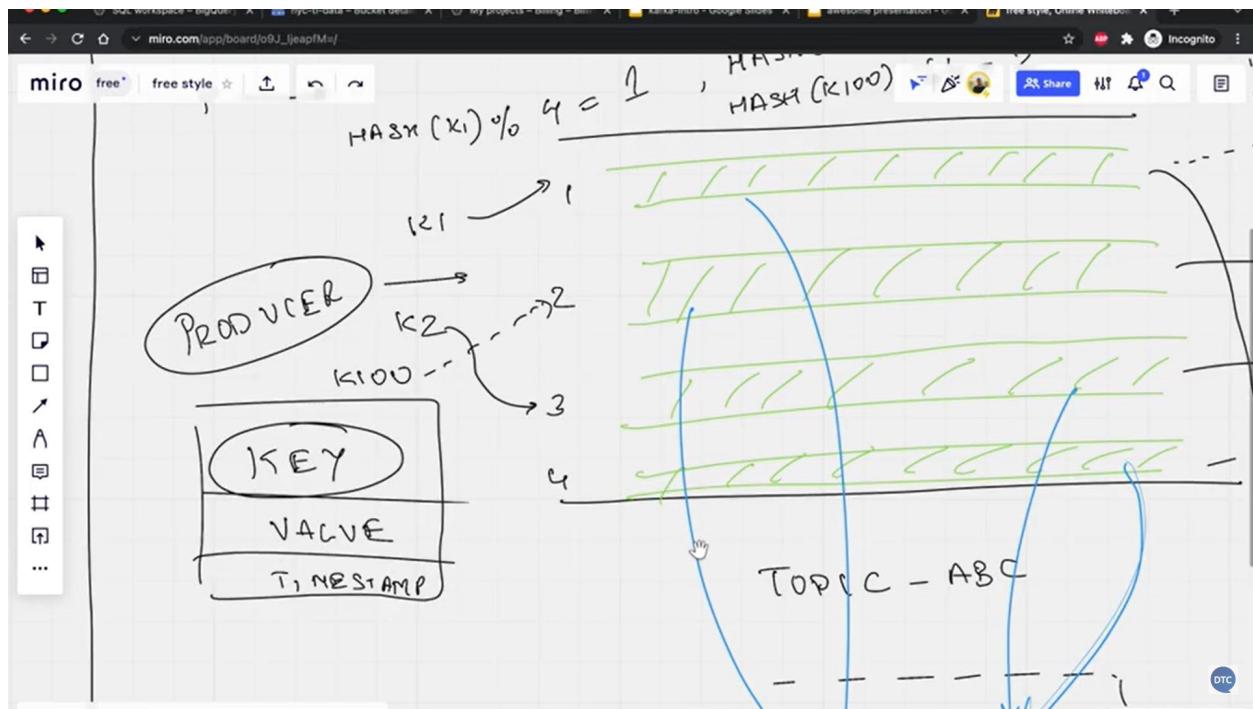
Consumer1- 4partition - 2consumer groups:

Here partitions were assigned to consumer group1 and consumer group2 separately. Number of messages read by CG1, CG2 can be different and not related to each other.



How does kafka group same key into same partition?

For each key value, $\text{hash}(\text{key}) \bmod 4 = 1$, then key goes to partition1.
Here, $\text{hash}(k1) \% 4 == 1$, then key goes to partition1.



Replication in Kafka

Fault tolerance

DTC

|| ▶ 🔍 35:18 / 37:39

▼

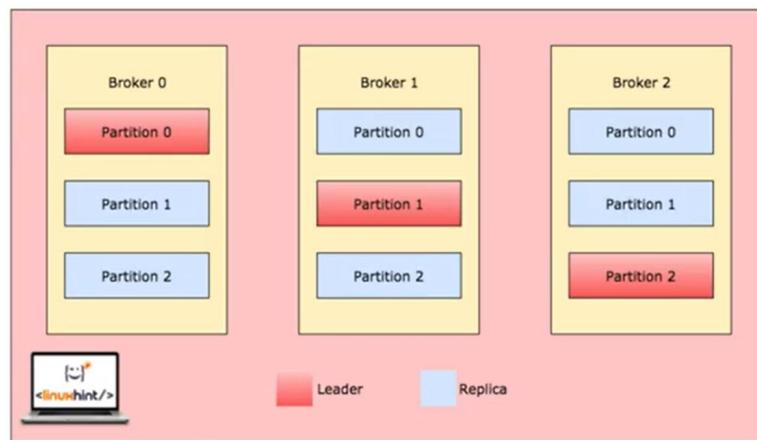
▶

◀

✖

✖

✖



©ankushkhanna

DTC

Docker configuration

DE Zoomcamp 6.1.2 - Kafka configuration and demo

Watch Later Share

Configurations Topic

- **retention.ms** => Amount of time logs will stay before they get deleted
- **cleanup.policy**=[**delete**|**compact**], either delete the messages from topic or compact them
- **partition**, scalability count
- **replication**, number of times a partition will be replicated

0:59 / 23:01

©ankushkhanna

HD YouTube

DE Zoomcamp 6.1.2 - Kafka configuration and demo

Watch Later Share

Configuration Consumer

- **offset** => What has already been read by the consumer
- **consumer.group.id** => Identifier for the consumer group
- **auto.offset.reset**=[**earliest**|**latest**], when consumer connects for first time to a topic (offset does not exists for this consumer.group.id), where to start reading from. From first (earliest) message or last (latest) message

MORE VIDEOS

3:17 / 23:01

©ankushkhanna

HD YouTube

DE Zoomcamp 6.1.2 - Kafka configuration and demo

Watch Later Share

Configuration Producer

- **acks: [0|1|all]**
 - 0 => Does not wait for leader or replica broker to write the message to disk
 - 1 => Waits for leader broker to write the message to disk
 - all => Waits for leader and all replica to write the message to disk

MORE VIDEOS

©ankushkhanna

4:32 / 23:01

DTC

DE Zoomcamp 6.1.2 - Kafka configuration and demo

Watch Later Share

```
version: '3'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:5.4.0
    hostname: zookeeper
    container_name: zookeeper
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

  broker:
    image: confluentinc/cp-server:5.4.0
    hostname: broker
    container_name: broker
    depends_on:
      - zookeeper
    ports:
      - "9092:9092"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: "zookeeper:2181"
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://broker:29092,PLAINTEXT_HOST://localhost:9092
      KAFKA_METRIC_REPORTERS: io.confluent.metrics.reporter.ConfluentMetricsReporter
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
```

MORE VIDEOS

7:46 / 23:01

DTC

The screenshot shows a video player interface with a dark theme. The title bar reads "DTC DE Zoomcamp 6.1.2 - Kafka configuration and demo". The main content area displays a Docker Compose configuration file (`docker-compose.yml`) for Kafka. The file defines several services: `zookeeper`, `broker`, `kafka-tools`, `schema-registry`, and `control-center`. The `zookeeper` service uses the image `confluentinc/cp-zookeeper:5.4.0` and binds port 2181. The `environment` section sets `ZOOKEEPER_CLIENT_PORT` to 2181 and `ZOOKEEPER_TICK_TIME` to 2000. The `broker` service is shown with a placeholder for 6 keys. The `kafka-tools`, `schema-registry`, and `control-center` services are also listed with their respective key counts. On the left, a sidebar shows the project structure with files like `week_6_stream_processing`, `producer.py`, and `consumer.py`. At the bottom, there's a "MORE VIDEOS" button, a progress bar at 8:30 / 23:01, and various video control icons.

```
version: '3'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:5.4.0
    hostname: zookeeper
    container_name: zookeeper
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
  broker: <6 keys>
  kafka-tools: <5 keys>
  schema-registry: <6 keys>
  control-center: <6 keys>
```

\$ docker-compose up // start docker container for kafka

\$ docker ps // to check docker has started w/t error.

UI for docker configuration and management as part of enterprises edition, this can be done in command line too.

<http://localhost:9021/cluster>

Create a new topic using “topics” column.

The latest available data for this cluster was 9 minutes ago.

For high availability in product use cases, start your Enterprise trial by adding more brokers to your cluster and increasing your topic replication factor.

New topic

Topic name*

Number of partitions*

Create with defaults **Customize settings** Cancel

TOPIC SUMMARY	
name	--
partitions	1
replication.factor	1
cluster	controlcenter cluster
min.insync.replicas	1
cleanup.policy	delete
retention.ms	604800000
retention.bytes	

```

data-engineering-zoomcamp week_6_stream_processing consumer.py
Project Project data-engineering-zoomcamp ~/Documents/persona
  > images
  > project
  > serving_dir
  > week_1_basics_n_setup
  > week_2_data_ingestion
  > week_3_data_warehouse
  > week_4_analytics_engineering
  > week_5_batch_processing
  > week_6_stream_processing
    > week6_venv
      .gitignore
      consumer.py
      docker-compose.yml
      producer.py
      requirements.txt
      .gitignore
      arch_diagram.md
      asking-questions.md
      dataset.md
      README.md
  > External Libraries
  > Scratches and Consoles

from kafka import KafkaConsumer
from json import loads
from time import sleep

consumer = KafkaConsumer(
    'demo_1',
    bootstrap_servers=['localhost:9092'],
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    group_id='consumer.group.id.demo.1',
    value_deserializer=lambda x: loads(x.decode('utf-8')))

while(True):
    print("inside while")
    for message in consumer:
        MESSAGE = message.value
        print(message)
    sleep(1)

```

Topic partitioned 2 and 2 consumers running in parallel:

The screenshot shows a terminal window with several tabs open. The active tab is titled "DE Zoomcamp 6.1.2 - Kafka configuration and demo". It displays the output of a Python script named "consumer.py" which is reading from a Kafka topic. The output consists of a series of JSON objects, each containing a key "number" with a value ranging from 313 to 368. The terminal also shows the user's last login information and some system status icons at the bottom.

```
Last login: Tue Feb 15 18:54:19 on ttys018
ak381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp k6_kafka 1+ python3 week_6_stream_processing/consumer.py
inside while
{'number': 313}
{'number': 314}
{'number': 315}
{'number': 316}
{'number': 317}
{'number': 318}
{'number': 320}
{'number': 321}
{'number': 323}
{'number': 324}
{'number': 327}
{'number': 328}
{'number': 329}
{'number': 333}
{'number': 335}
{'number': 337}
{'number': 340}
{'number': 344}
{'number': 345}
{'number': 347}
{'number': 350}
{'number': 352}
{'number': 354}
{'number': 355}
{'number': 357}
{'number': 361}
{'number': 362}
{'number': 363}
{'number': 364}
{'number': 366}
{'number': 367}
{'number': 368}
{'number': 370}
{'number': 371}
{'number': 372}
{'number': 373}
{'number': 374}
{'number': 375}
{'number': 376}
{'number': 380}
{'number': 382}
{'number': 384}
{'number': 386}
{'number': 388}
{'number': 389}
{'number': 390}
{'number': 392}
{'number': 394}
{'number': 395}
{'number': 396}
{'number': 397}
{'number': 398}
{'number': 399}
{'number': 401}
{'number': 403}
{'number': 405}
```

Consumers > number of partition that consumer is idle and not processing any data.

Topic demo_1 with 2 partitions and 3 consumers running in parallel. 2 consumers were consuming data and third consumer is idle.

The screenshot shows a terminal window with several tabs open. The active tab is titled "1.python3 week_6_stream_processing/consumer.py (Python)". It displays the output of a Python script named "consumer.py" which is reading from a Kafka topic. The output consists of a series of JSON objects, each containing a key "number" with a value ranging from 357 to 406. The terminal also shows the user's last login information and some system status icons at the bottom.

```
Last login: Wed Feb 16 09:36:11 on ttys002
ak381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp k6_kafka 1+ python3 week_6_stream_processing/consumer.py
inside while
{'number': 357}
{'number': 361}
{'number': 362}
{'number': 363}
{'number': 364}
{'number': 366}
{'number': 367}
{'number': 368}
{'number': 370}
{'number': 371}
{'number': 372}
{'number': 373}
{'number': 374}
{'number': 375}
{'number': 376}
{'number': 380}
{'number': 382}
{'number': 384}
{'number': 386}
{'number': 388}
{'number': 389}
{'number': 390}
{'number': 392}
{'number': 394}
{'number': 395}
{'number': 396}
{'number': 397}
{'number': 398}
{'number': 399}
{'number': 401}
{'number': 403}
{'number': 405}
```

Killed consumer2 and 3rd consumer got registered into kafka and start receiving data from kafka:

The screenshot shows a terminal window with four tabs. Tab 1 contains Python code for a Kafka consumer. Tab 2 shows a log entry for a user named 'ak381b' logging in at 09:36:11 on ttys002. Tab 3 shows a command to kill process 3748. Tab 4 shows the Python script running, with the output pane displaying a sequence of numbers starting from 371, indicating messages being processed by consumer1.

```
{'number': 371}, {'number': 372}, {'number': 373}, {'number': 374}, {'number': 375}, {'number': 376}, {'number': 380}, {'number': 382}, {'number': 384}, {'number': 386}, {'number': 388}, {'number': 389}, {'number': 390}, {'number': 392}, {'number': 394}, {'number': 395}, {'number': 396}, {'number': 397}, {'number': 398}, {'number': 399}, {'number': 401}, {'number': 403}, {'number': 405}, {'number': 408}, {'number': 411}, {'number': 413}, {'number': 417}, {'number': 425}, {'number': 427}, {'number': 429}, {'number': 430}, {'number': 431},
```

```
Last login: Wed Feb 16 09:36:11 on ttys002
ak381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
python3 week6_kafka +* python3 week6_stream_processing/consumer.py
inside while
{'number': 421}, {'number': 422}, {'number': 423}, {'number': 424}, {'number': 426}, {'number': 428}, {'number': 432}, {'number': 433}, |
```

```
ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
x ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
x ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
x ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
x ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
```

Killed consumer2 & consumer3, now consumer1 start receiving all the messages sequentially:

The screenshot shows a terminal window with four tabs. Tab 1 contains Python code for a Kafka consumer. Tab 2 shows a log entry for a user named 'ak381b' logging in at 09:36:11 on ttys002. Tab 3 shows a command to kill process 3748. Tab 4 shows the Python script running, with the output pane displaying a sequence of numbers starting from 408, indicating messages being processed sequentially by consumer1.

```
{'number': 408}, {'number': 411}, {'number': 413}, {'number': 417}, {'number': 425}, {'number': 427}, {'number': 429}, {'number': 430}, {'number': 431}, {'number': 434}, {'number': 436}, {'number': 441}, {'number': 443}, {'number': 444}, {'number': 447}, {'number': 449}, {'number': 452}, {'number': 453}, {'number': 442}, {'number': 445}, {'number': 446}, {'number': 448}, {'number': 450}, {'number': 451}, {'number': 454}, {'number': 455}, {'number': 456}, {'number': 457}, {'number': 458}, {'number': 459}, {'number': 460}, {'number': 461}, {'number': 462},
```

```
ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
python3 week6_kafka +* python3 week6_stream_processing/consumer.py
inside while
{'number': 421}, {'number': 422}, {'number': 423}, {'number': 424}, {'number': 426}, {'number': 428}, {'number': 432}, {'number': 433}, |
```

```
ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
x ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
x ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
x ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
x ok381b@ALT03748 ~-/Documents/personal/learning/data-engineering-zoomcamp
```

Experiment:

Kafka 1 producer and 1 consumer

```
thamizh$python consumer.py
In consumer
{'vehicle_number': 0}
{'vehicle_number': 1}
{'vehicle_number': 2}
{'vehicle_number': 3}
{'vehicle_number': 4}
{'vehicle_number': 5}
{'vehicle_number': 6}
{'vehicle_number': 7}
{'vehicle_number': 8}
{'vehicle_number': 9}
{'vehicle_number': 10}
{'vehicle_number': 11}

thamizh$python producer.py
producer: vechile_number:0
producer: vechile_number:1
producer: vechile_number:2
producer: vechile_number:3
producer: vechile_number:4
producer: vechile_number:5
producer: vechile_number:6
producer: vechile_number:7
producer: vechile_number:8
producer: vechile_number:9
producer: vechile_number:10
producer: vechile_number:11
```

Install Development Libraries
Download and build Snappy from <https://google.github.io/snappy/>
Ubuntu:
apt-get install libsnappy-dev

1 producer, 2 partitions, 3 Consumers configured. Consumer2 & 3 started processing kafka topics.
Consumer1 is idle.

```
thamizh$python consumer.py
In consumer
{'vehicle_number': 6}
{'vehicle_number': 10}
{'vehicle_number': 11}
{'vehicle_number': 12}
{'vehicle_number': 15}
{'vehicle_number': 18}

thamizh$python consumer.py
In consumer
{'vehicle_number': 0}
{'vehicle_number': 1}
{'vehicle_number': 2}
{'vehicle_number': 3}
{'vehicle_number': 4}
{'vehicle_number': 5}
{'vehicle_number': 7}
{'vehicle_number': 8}
{'vehicle_number': 9}
{'vehicle_number': 13}
{'vehicle_number': 14}
{'vehicle_number': 16}
{'vehicle_number': 17}
{'vehicle_number': 19}

thamizh$python producer.py
producer: vechile_number:0
producer: vechile_number:1
producer: vechile_number:2
producer: vechile_number:3
producer: vechile_number:4
producer: vechile_number:5
producer: vechile_number:6
producer: vechile_number:7
producer: vechile_number:8
producer: vechile_number:9
producer: vechile_number:10
producer: vechile_number:11
producer: vechile_number:12
producer: vechile_number:13
producer: vechile_number:14
producer: vechile_number:15
producer: vechile_number:16
producer: vechile_number:17
producer: vechile_number:18
producer: vechile_number:19
```

Consumer3 was killed

```

thamizh@DESKTOP-KMM8Q4: ~
In consumer
{'vehicle_number': 57}
{'vehicle_number': 62}
{'vehicle_number': 66}
{'vehicle_number': 67}
{'vehicle_number': 69}
{'vehicle_number': 70}
{'vehicle_number': 76}
{'vehicle_number': 78}
{'vehicle_number': 79}
{'vehicle_number': 83}

thamizh@DESKTOP-KMM8Q4: ~
{'vehicle_number': 36}
{'vehicle_number': 39}
{'vehicle_number': 42}
{'vehicle_number': 45}
{'vehicle_number': 46}
{'vehicle_number': 47}
{'vehicle_number': 48}
{'vehicle_number': 53}
{'vehicle_number': 44}
{'vehicle_number': 49}
{'vehicle_number': 50}
{'vehicle_number': 51}
{'vehicle_number': 52}
{'vehicle_number': 54}
{'vehicle_number': 55}
{'vehicle_number': 56}
{'vehicle_number': 58}
{'vehicle_number': 59}
{'vehicle_number': 60}
{'vehicle_number': 61}
{'vehicle_number': 63}
{'vehicle_number': 64}
{'vehicle_number': 65}
{'vehicle_number': 68}
{'vehicle_number': 71}
{'vehicle_number': 72}
{'vehicle_number': 73}
{'vehicle_number': 74}
{'vehicle_number': 75}
{'vehicle_number': 77}
{'vehicle_number': 80}
{'vehicle_number': 81}
{'vehicle_number': 82}
{'vehicle_number': 84}
{'vehicle_number': 85}
{'vehicle_number': 86}
{'vehicle_number': 87}

thamizh@DESKTOP-KMM8Q4: ~
File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/consumer/group.py", line 1201, in next_v2
    return next(self._iterator)
      File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/consumer/group.py", line 1116, in _message_generator
        record_map = self.poll(timeout_ms=timeout_ms,
        records = self._poll_once(remaining,
        max_records, update_offsets=update_offsets)
      File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/consumer/group.py", line 655, in poll
        self._client.poll(timeout_ms=timeout_ms)
      File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/client_async.py", line 602, in poll
        self._poll(timeout / 1000)
      File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/client_async.py", line 634, in _poll
        ready = self._selector.select(timeout)
      File "/home/thamizh/anaconda3/lib/python3.9/selectors.py", line 469, in select
        fd_event_list = self._selector.poll(
        timeout, max_ev)
KeyboardInterrupt

thamizh$
```

1 producer, 2 partitions, 3 Consumers configured. Initially consumer 2 & 3 started receiving kafka messages, after consumer 3 got killed, consumer1 automatically started processing kafka topics:

```

thamizh@DESKTOP-KMM8Q4: ~
In consumer
{'vehicle_number': 57}
{'vehicle_number': 62}
{'vehicle_number': 66}
{'vehicle_number': 67}
{'vehicle_number': 69}
{'vehicle_number': 70}
{'vehicle_number': 76}
{'vehicle_number': 78}
{'vehicle_number': 79}
{'vehicle_number': 83}
{'vehicle_number': 88}
{'vehicle_number': 89}
{'vehicle_number': 90}
{'vehicle_number': 94}
{'vehicle_number': 98}
{'vehicle_number': 99}
{'vehicle_number': 100}
{'vehicle_number': 102}
{'vehicle_number': 108}
{'vehicle_number': 109}
{'vehicle_number': 110}
{'vehicle_number': 111}
{'vehicle_number': 113}
{'vehicle_number': 114}
{'vehicle_number': 117}
{'vehicle_number': 118}
{'vehicle_number': 121}
{'vehicle_number': 122}
{'vehicle_number': 126}
{'vehicle_number': 129}
{'vehicle_number': 130}

thamizh@DESKTOP-KMM8Q4: ~
{'vehicle_number': 75}
{'vehicle_number': 77}
{'vehicle_number': 80}
{'vehicle_number': 82}
{'vehicle_number': 84}
{'vehicle_number': 85}
{'vehicle_number': 86}
{'vehicle_number': 87}
{'vehicle_number': 91}
{'vehicle_number': 92}
{'vehicle_number': 93}
{'vehicle_number': 95}
{'vehicle_number': 96}
{'vehicle_number': 97}
{'vehicle_number': 101}
{'vehicle_number': 103}
{'vehicle_number': 104}
{'vehicle_number': 105}
{'vehicle_number': 106}
{'vehicle_number': 107}
{'vehicle_number': 108}
{'vehicle_number': 109}
{'vehicle_number': 111}
{'vehicle_number': 115}
{'vehicle_number': 116}
{'vehicle_number': 119}
{'vehicle_number': 120}
{'vehicle_number': 123}
{'vehicle_number': 124}
{'vehicle_number': 125}
{'vehicle_number': 127}
{'vehicle_number': 128}
{'vehicle_number': 131}
{'vehicle_number': 132}
{'vehicle_number': 133}
{'vehicle_number': 135}
{'vehicle_number': 138}
{'vehicle_number': 140}

thamizh@DESKTOP-KMM8Q4: ~
File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/consumer/group.py", line 1116, in _message_generator
    record_map = self.poll(timeout_ms=timeout_ms,
    records = self._poll_once(remaining,
    max_records, update_offsets=update_offsets)
      File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/consumer/group.py", line 655, in poll
        self._client.poll(timeout_ms=timeout_ms)
      File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/client_async.py", line 602, in poll
        self._poll(timeout / 1000)
      File "/home/thamizh/anaconda3/lib/python3.9/site-packages/kafka/client_async.py", line 634, in _poll
        ready = self._selector.select(timeout)
      File "/home/thamizh/anaconda3/lib/python3.9/selectors.py", line 469, in select
        fd_event_list = self._selector.poll(
        timeout, max_ev)
KeyboardInterrupt

thamizh$python consumer.py
In consumer
{'vehicle_number': 134}
{'vehicle_number': 136}
{'vehicle_number': 137}
{'vehicle_number': 139}

thamizh$
```

2 kafka broker partition, 3 consumers configured, only one consumer running and receiving messages

```
thamizh$ {"vehicle_number": 178}
{"vehicle_number": 181}
{"vehicle_number": 183}
{"vehicle_number": 184}
{"vehicle_number": 185}
{"vehicle_number": 186}
{"vehicle_number": 189}
{"vehicle_number": 191}
{"vehicle_number": 193}
{"vehicle_number": 196}
{"vehicle_number": 198}
{"vehicle_number": 199}
{"vehicle_number": 190}
{"vehicle_number": 192}
{"vehicle_number": 194}
{"vehicle_number": 195}
{"vehicle_number": 197}
{"vehicle_number": 200}
{"vehicle_number": 201}
{"vehicle_number": 202}
{"vehicle_number": 203}
{"vehicle_number": 204}
{"vehicle_number": 205}
{"vehicle_number": 206}
{"vehicle_number": 207}
{"vehicle_number": 208}
{"vehicle_number": 209}
{"vehicle_number": 210}
{"vehicle_number": 211}
{"vehicle_number": 212}
{"vehicle_number": 213}
{"vehicle_number": 214}
{"vehicle_number": 215}
{"vehicle_number": 216}
{"vehicle_number": 217}
{"vehicle_number": 218}
{"vehicle_number": 219}
thamizh$ ^producer: vechile_number:188
producer: vechile_number:189
producer: vechile_number:190
producer: vechile_number:191
producer: vechile_number:192
producer: vechile_number:193
producer: vechile_number:194
producer: vechile_number:195
producer: vechile_number:196
producer: vechile_number:197
producer: vechile_number:198
producer: vechile_number:199
producer: vechile_number:200
producer: vechile_number:201
producer: vechile_number:202
producer: vechile_number:203
producer: vechile_number:204
producer: vechile_number:205
producer: vechile_number:206
producer: vechile_number:207
producer: vechile_number:208
producer: vechile_number:209
producer: vechile_number:210
producer: vechile_number:211
producer: vechile_number:212
producer: vechile_number:213
producer: vechile_number:214
producer: vechile_number:215
producer: vechile_number:216
producer: vechile_number:217
producer: vechile_number:218
producer: vechile_number:219
producer: vechile_number:220
producer: vechile_number:221
producer: vechile_number:222
producer: vechile_number:223
producer: vechile_number:224
```

Avro Schema Registry:

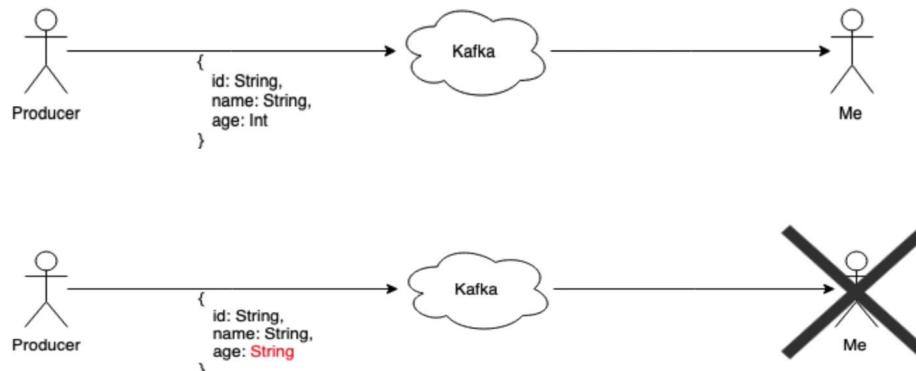
PHPAppln -> Kafka-> Java consumer

What is Avro

- **Avro** is a data serialization system
- **Schema stored separate from Record** (i.e. need schema to read record) (unlike *ProtoBuffers* or *JSON*)
- Records stored using binary encoding or JSON
- Avro advantages:
 - Smaller filesize (vs JSON)
 - Schema evolution
 - Avro clients provide automatic validation against schema



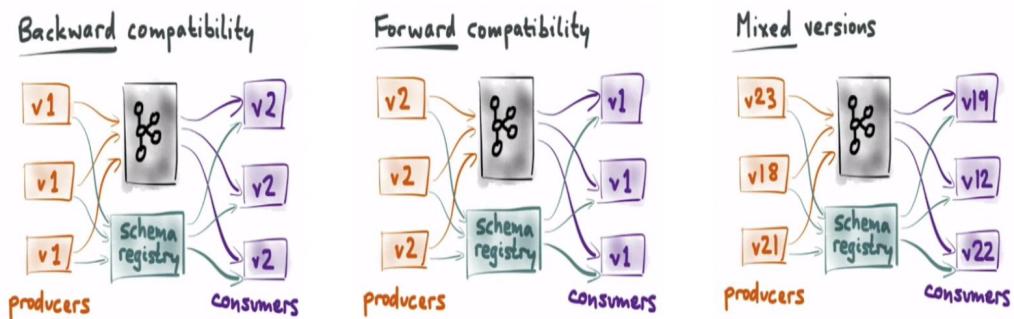
Why Schema Compatibility?



©ankushkhanna



Avro schema evolution

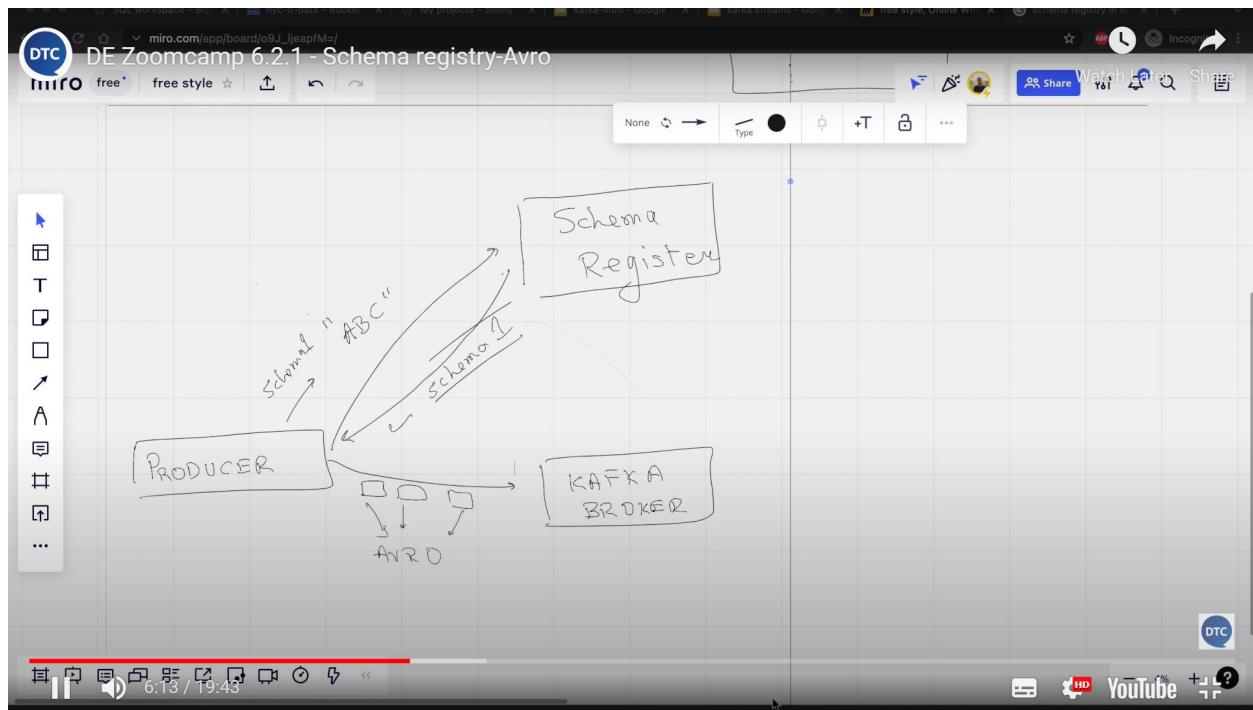


©ankushkhanna

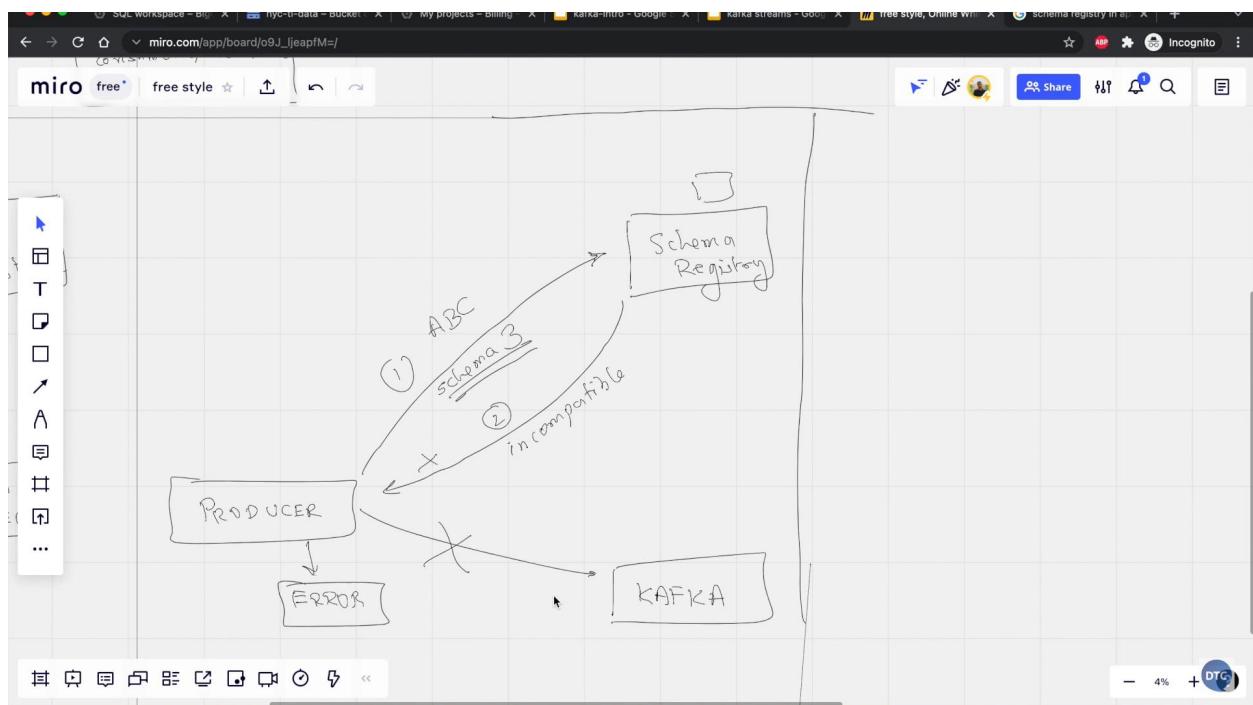


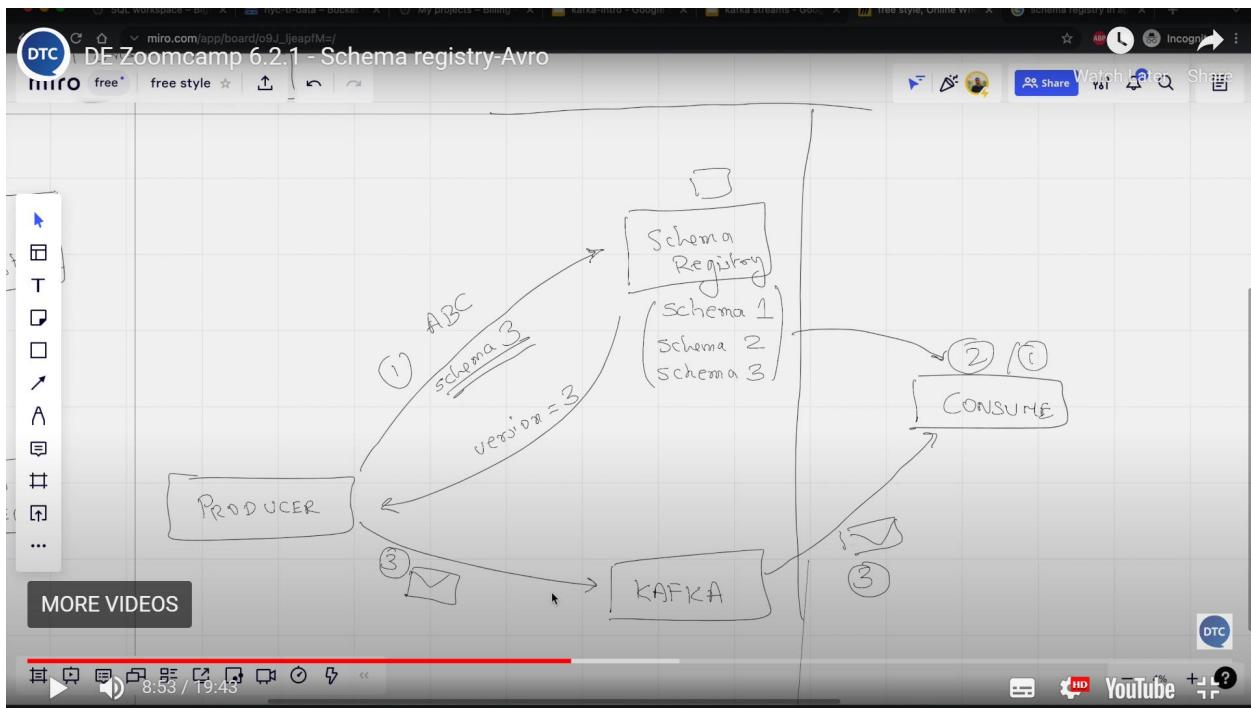
Avro serialization: schema is compatible

<https://inakianduaga.github.io/kafka-image-processor/#/1>

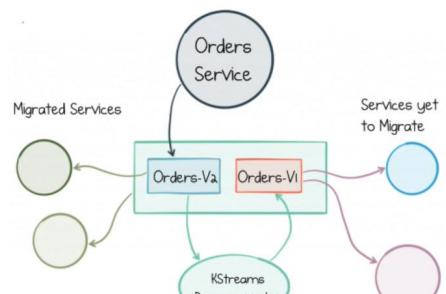


Avro serialization: schema is not compatible





What if Schema cannot be compatible?

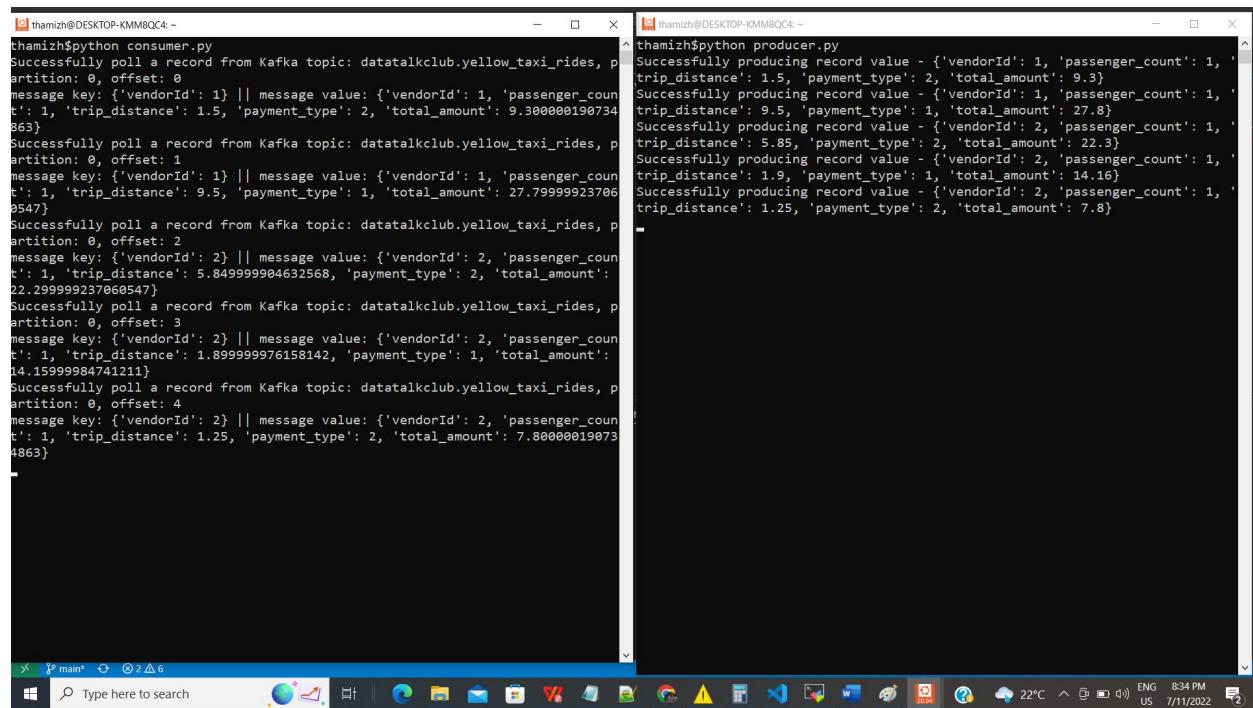


The same data coexists in two topics, with different schemas, so there is a window during which services can upgrade.

<https://www.confluent.io/blog/apache-kafka-for-service-architectures/>

@ankushkhanna

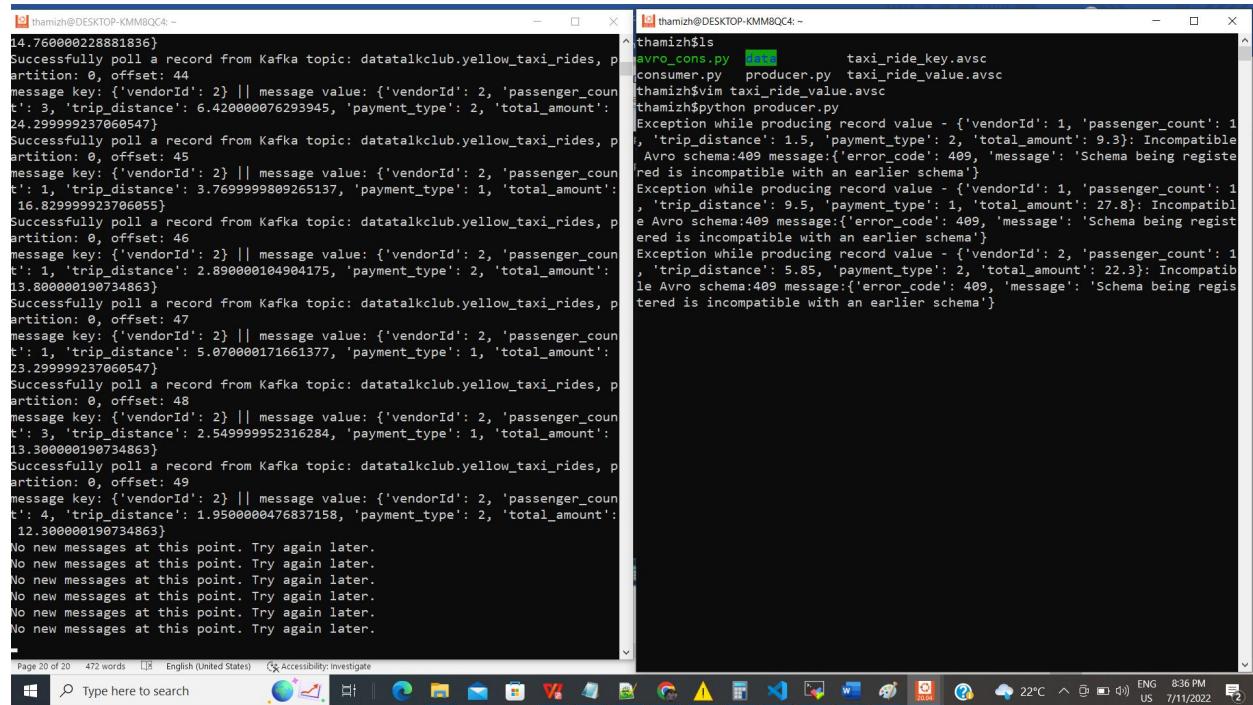
Kafka Producer consumer with schema validation:



```
thamizh$ python consumer.py
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 0
message key: {'vendorId': 1} || message value: {'vendorId': 1, 'passenger_count': 1, 'trip_distance': 1.5, 'payment_type': 2, 'total_amount': 9.3}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 1
message key: {'vendorId': 1} || message value: {'vendorId': 1, 'passenger_count': 1, 'trip_distance': 9.5, 'payment_type': 1, 'total_amount': 27.8}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 2
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 5.85, 'payment_type': 2, 'total_amount': 22.3}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 3
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 1.899999976158142, 'payment_type': 1, 'total_amount': 14.15999984741211}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 4
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 1.25, 'payment_type': 2, 'total_amount': 7.8}
4863}
```

```
thamizh$ python producer.py
Successfully producing record value - {'vendorId': 1, 'passenger_count': 1, 'trip_distance': 1.5, 'payment_type': 2, 'total_amount': 9.3}
Successfully producing record value - {'vendorId': 1, 'passenger_count': 1, 'trip_distance': 9.5, 'payment_type': 1, 'total_amount': 27.8}
Successfully producing record value - {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 5.85, 'payment_type': 2, 'total_amount': 22.3}
Successfully producing record value - {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 1.9, 'payment_type': 1, 'total_amount': 14.16}
Successfully producing record value - {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 1.25, 'payment_type': 2, 'total_amount': 7.8}
```

Changed taxi payment as string from int



```
thamizh$ python consumer.py
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 44
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 3, 'trip_distance': 6.420000076293945, 'payment_type': 2, 'total_amount': 24.760000228881836}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 45
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 3.7699999809265137, 'payment_type': 1, 'total_amount': 16.829999923706055}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 46
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 2.890000104984175, 'payment_type': 2, 'total_amount': 13.800000198734863}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 47
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 5.070000171661377, 'payment_type': 1, 'total_amount': 23.299999237060547}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 48
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 3, 'trip_distance': 2.549999952316284, 'payment_type': 1, 'total_amount': 13.300000198734863}
Successfully poll a record from Kafka topic: datatalkclub.yellow_taxi_rides, partition: 0, offset: 49
message key: {'vendorId': 2} || message value: {'vendorId': 2, 'passenger_count': 4, 'trip_distance': 1.9500000476837158, 'payment_type': 2, 'total_amount': 12.300000190734863}
No new messages at this point. Try again later.
No new messages at this point. Try again later.
No new messages at this point. Try again later.
No new messages at this point. Try again later.
No new messages at this point. Try again later.
No new messages at this point. Try again later.
```

```
thamizh$ ls
avro_cons.py  taxi_ride_key.avsc
consumer.py  producer.py  taxi_ride_value.avsc
thamizh$ im taxi_ride_value.avsc
thamizh$ python producer.py
Exception while producing record value - {'vendorId': 1, 'passenger_count': 1, 'trip_distance': 1.5, 'payment_type': 2, 'total_amount': 9.3}: Incompatible Avro schema:409 message:{'error_code': 409, 'message': 'Schema being registered is incompatible with an earlier schema'}
Exception while producing record value - {'vendorId': 1, 'passenger_count': 1, 'trip_distance': 9.5, 'payment_type': 1, 'total_amount': 27.8}: Incompatible Avro schema:409 message:{'error_code': 409, 'message': 'Schema being registered is incompatible with an earlier schema'}
Exception while producing record value - {'vendorId': 2, 'passenger_count': 1, 'trip_distance': 5.85, 'payment_type': 2, 'total_amount': 22.3}: Incompatible Avro schema:409 message:{'error_code': 409, 'message': 'Schema being registered is incompatible with an earlier schema'}
```