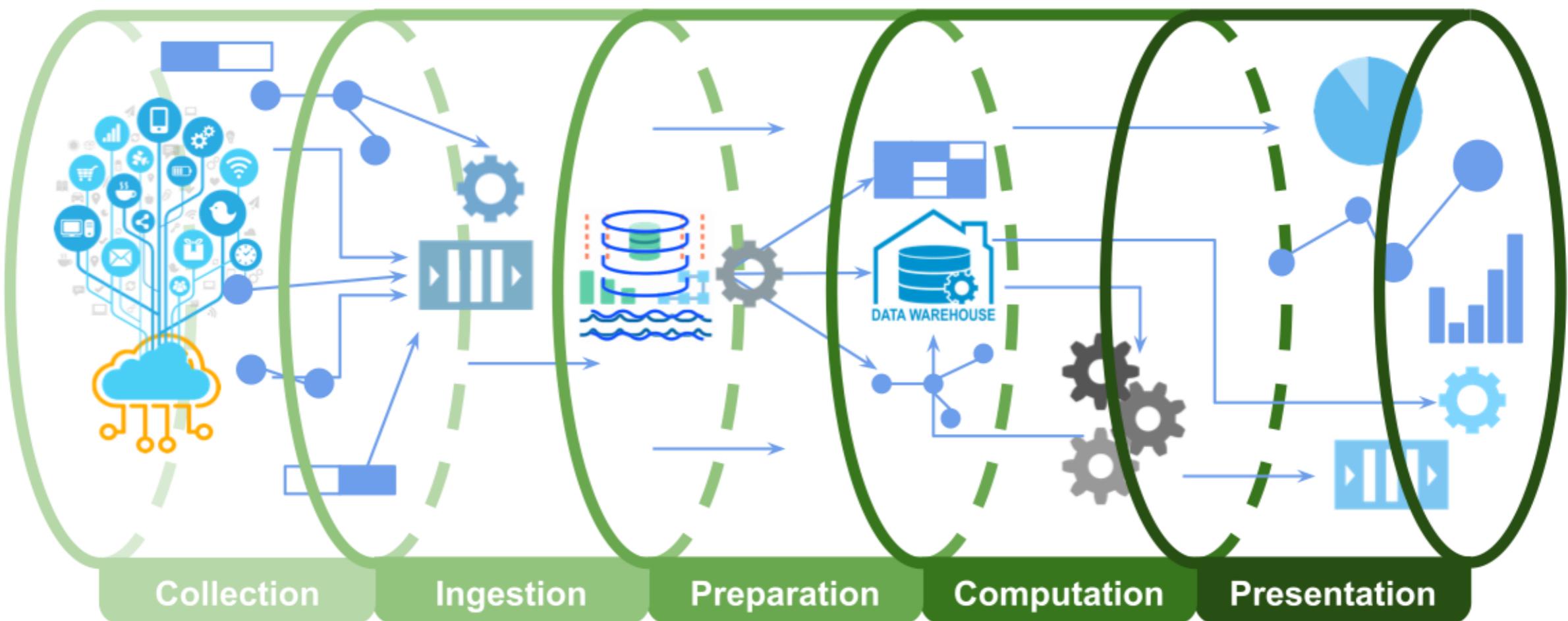


2110531 - Data Science and Data Engineering Tools

Data Ingestion

Asst.Prof. Natawut Nupairoj, Ph.D.
Department of Computer Engineering
Chulalongkorn University
natawut.n@chula.ac.th

Data Pipeline Value Chain



© Satish Chandra Gupta

@scgupta

The Tales of Smart Meters

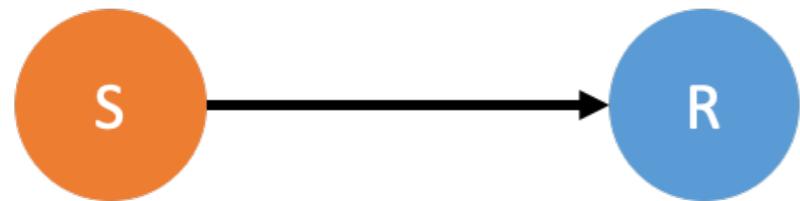
- Devices to measure power consumption
- Data is gathered every 15 minutes
- There are about 20m households in Thailand
- On the average = 22,222 events/sec
- Assume 1k data each event (power consumption, temperature, etc.)
- Total data = 80GB/hr. or 1.92 TB/day or > 700TB/year



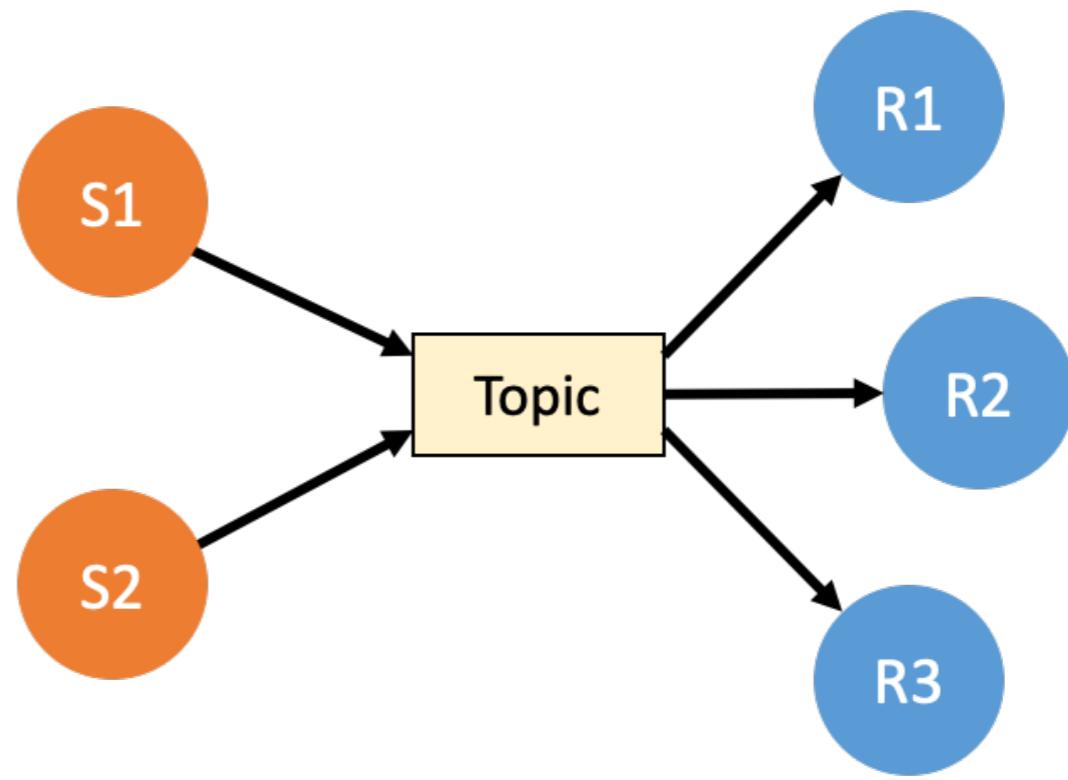
Data Collection and Data Ingestion Requirements

- Variety of data sources and destinations
- Complex data flow e.g. multiple data sources, multiple destinations, etc.
- Reliable data transfer
- In-flight data processing e.g. data cleansing, data standardization, data fusion
- Scalability and fault tolerance

Modes of Data Transfer



(a) Point-to-point



(b) Publish-subscribe

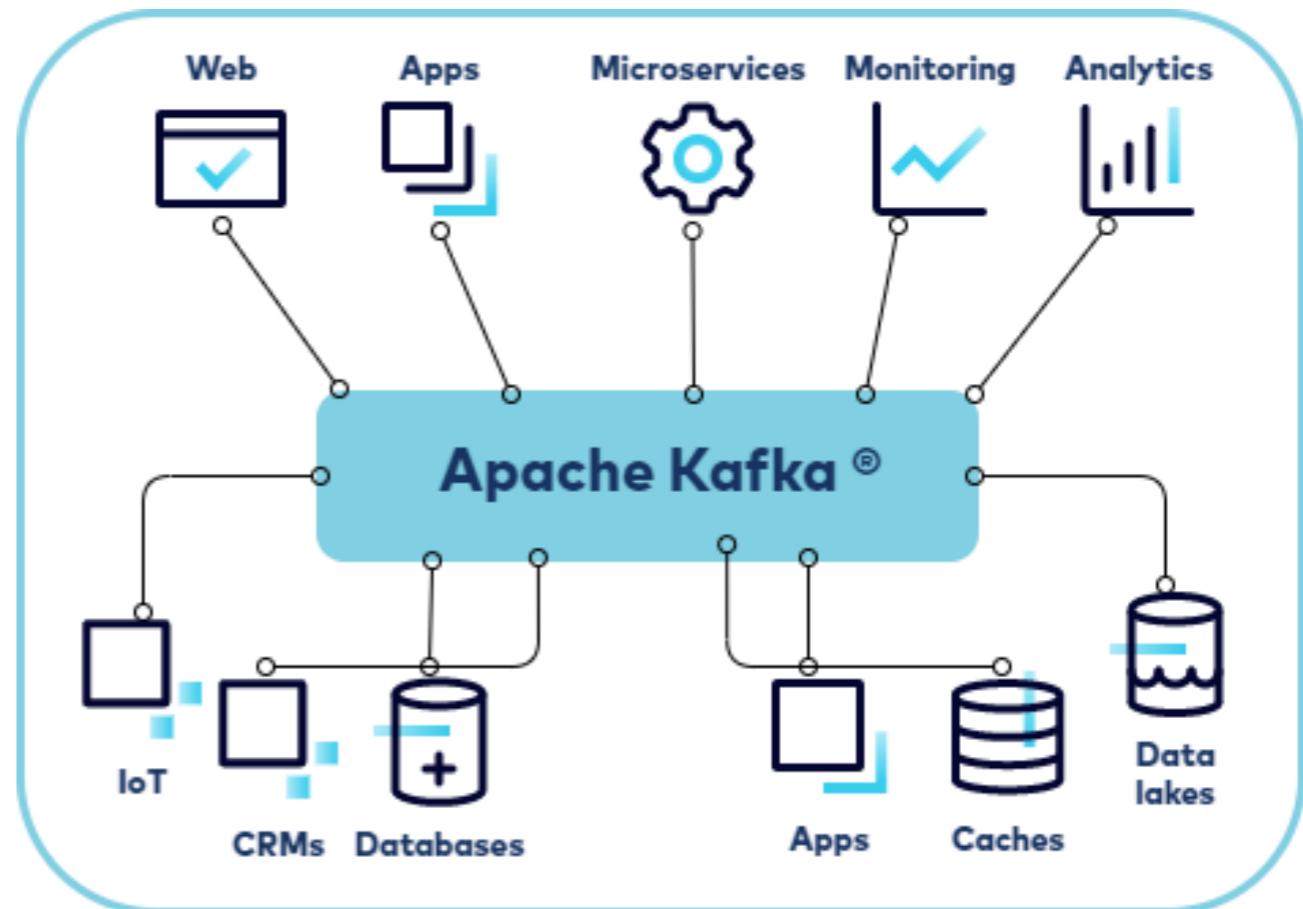
Introduction to Apache Kafka

Data Ingestion



Apache Kafka

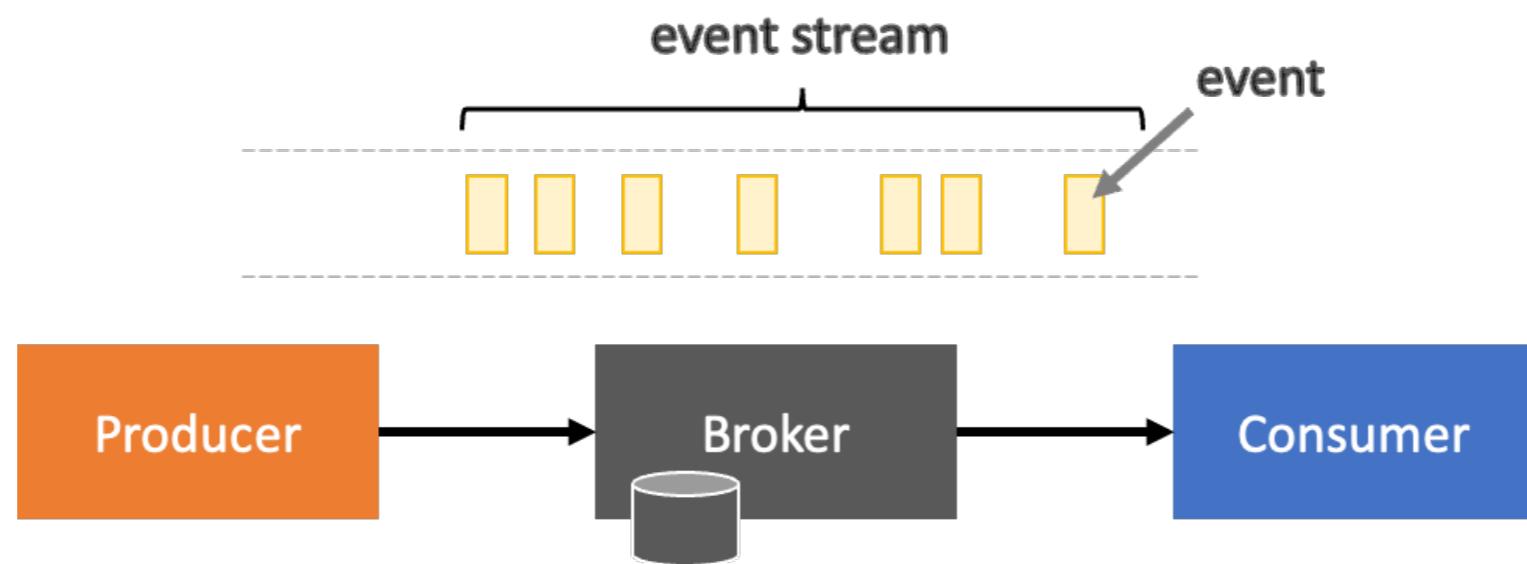
- Popular event stream processing from LinkedIn
- Real-time, reliable data delivery, highly scalable, and fault tolerance
- Being used in many large companies (LinkedIn, Grab, Line, Agoda, Netflix, Airbnb, Paypal, Tencent, Spotify)



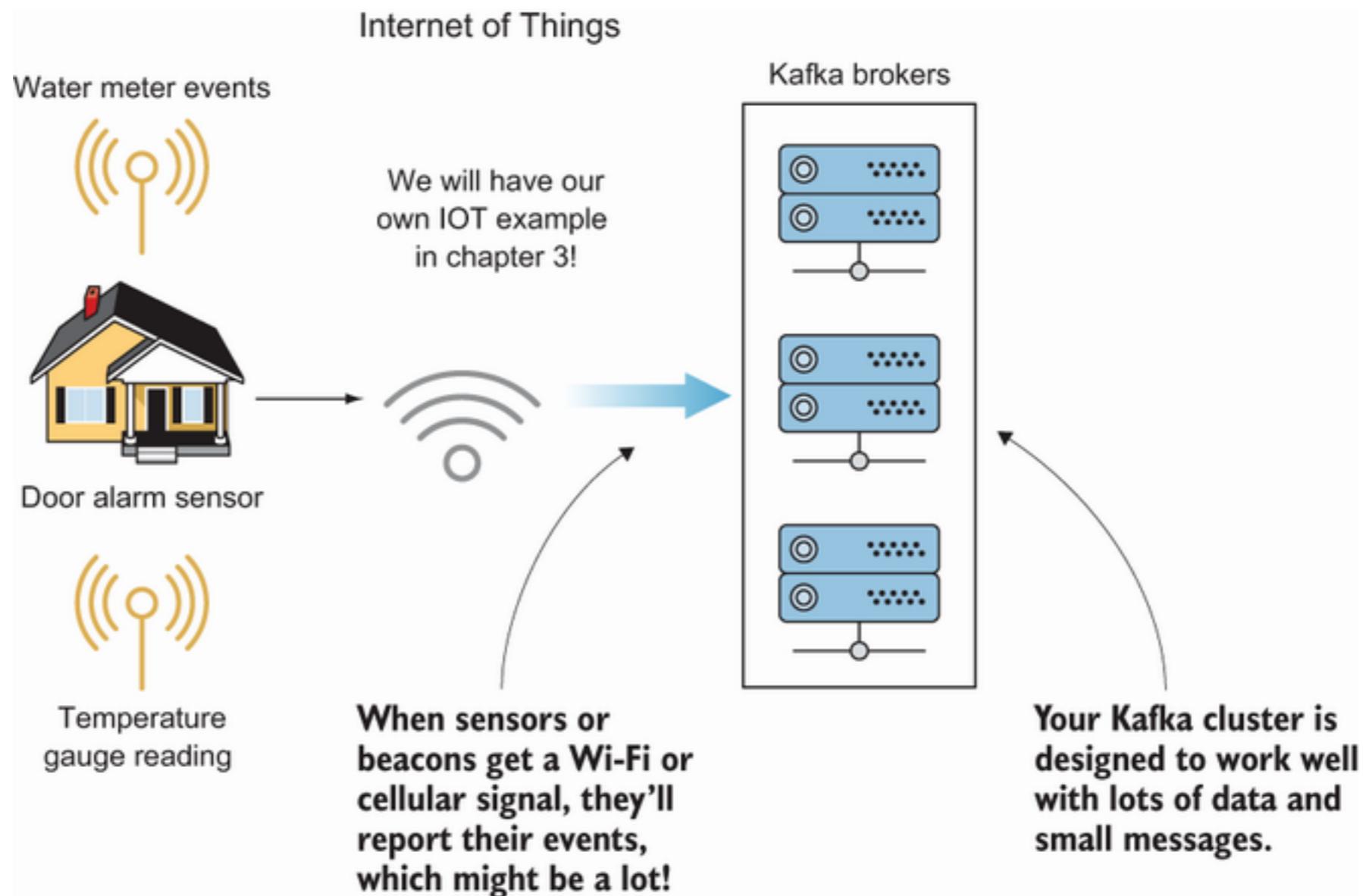
Source: <https://docs.confluent.io/kafka/introduction.html>

Apache Kafka's Main Capabilities

- Reading and writing records like a message queue
- Storing records with fault tolerance
- Processing streams as they occur

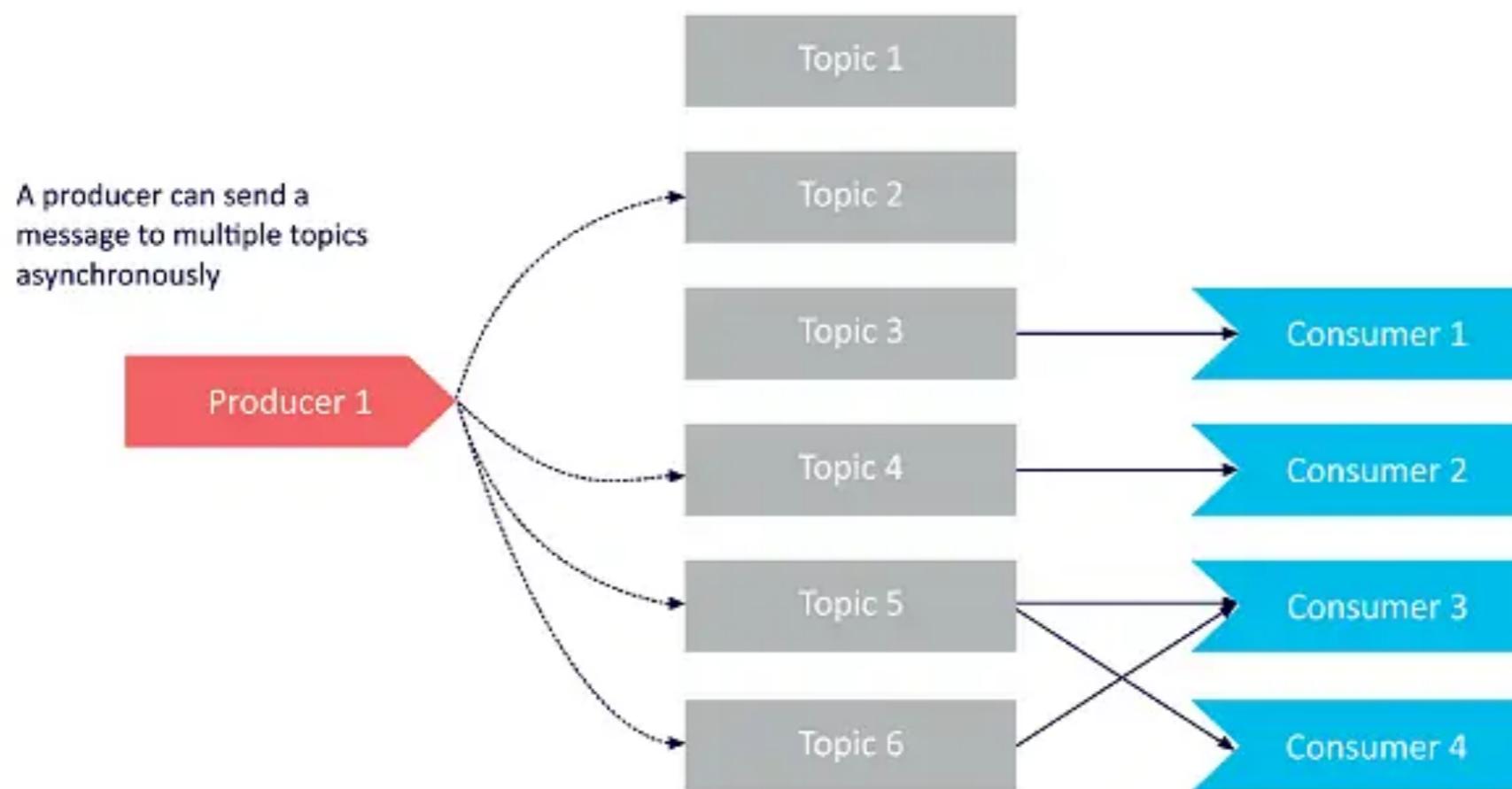


Kafka Example Use Case



Kafka Basic Concept

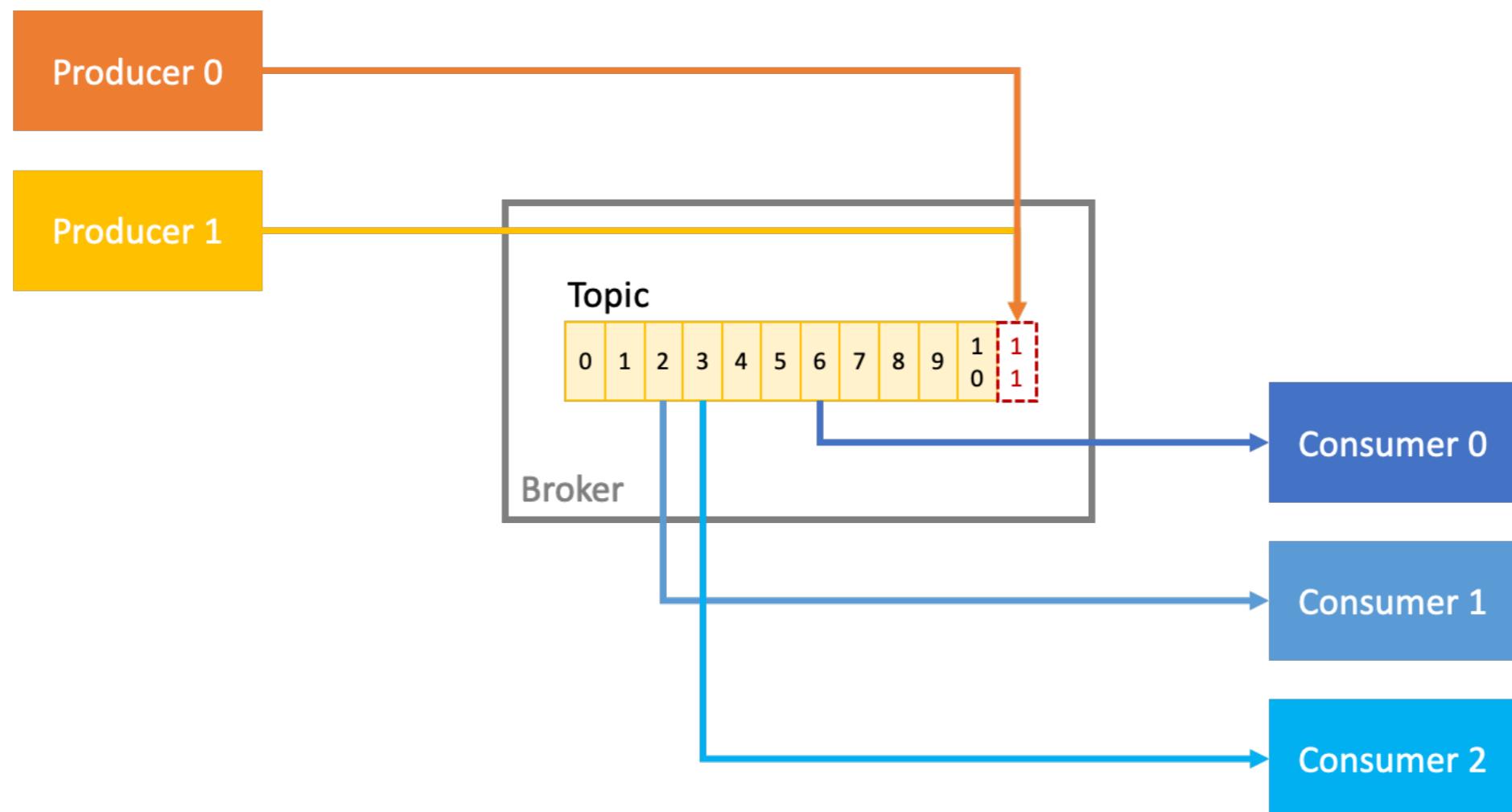
Source: <https://www.instaclustr.com/blog/apache-kafka-architecture/>



Kafka Topic

- Events or “records” are stored in topics
- Topic allows multi-producer and multi-consumer
- A consumer keeps track of the current record being consumed with “offset”
- A consumer auto commits offset periodically

Kafka Topic and Offsets



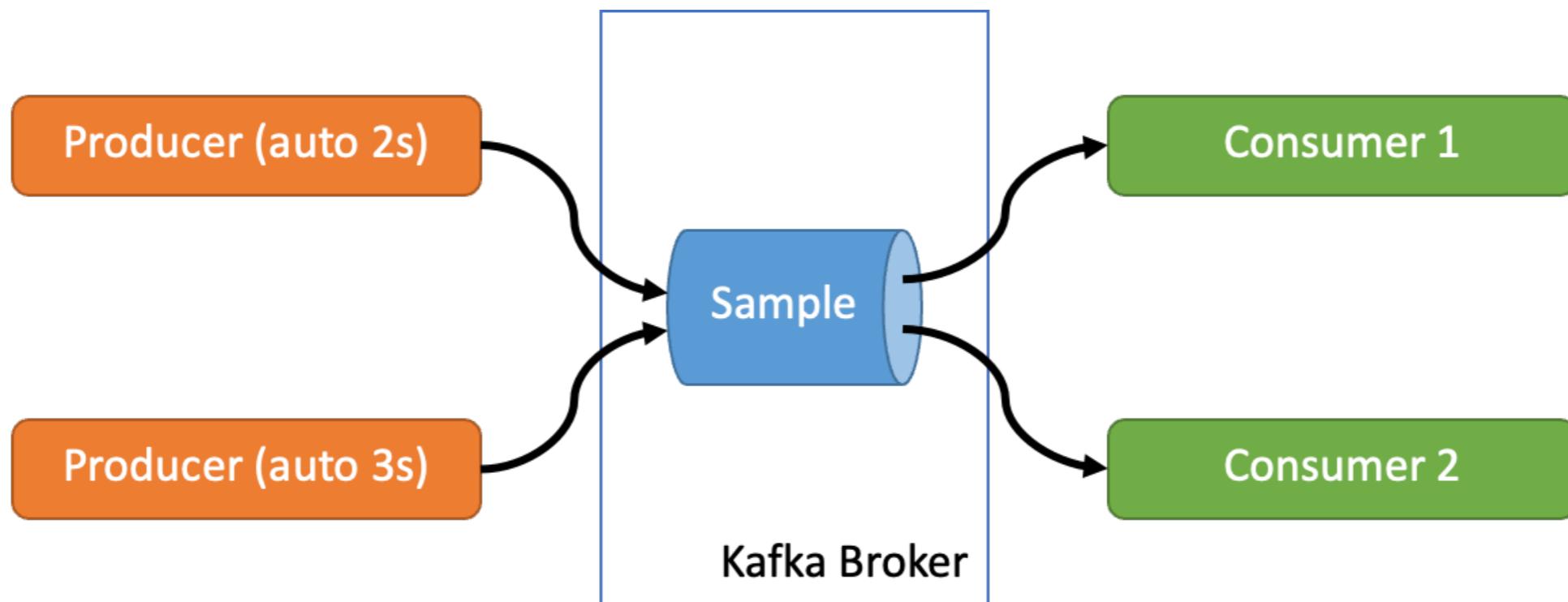
Playing with Kafka

- For client-side: you must install **kafka-python**

```
pip install kafka-python
```

- Server-side: use Kafka container from official repository (<https://github.com/apache/kafka>)
- Docker compose files are adapted from official repo and have been provided in the git
 - Single node (<https://github.com/apache/kafka/blob/trunk/docker/examples/docker-compose-files/single-node/plain-text/docker-compose.yml>)
 - Cluster (<https://github.com/apache/kafka/blob/trunk/docker/examples/docker-compose-files/cluster/combined/plain-text/docker-compose.yml>)
- You may study the following blog for more details regarding to Kafka containers (<https://medium.com/towards-data-engineering/unlock-the-power-of-apache-kafka-with-the-official-docker-image-5a65192e618b>)

Simple Example



ConsumerRecord:

- topic = topic of the message
- offset = offset of the message
- timestamp = epoch milliseconds
- value = message body

Kafka Producer

```
In [ ]: # import required libraries
from kafka import KafkaProducer
import time
```

```
In [ ]: # Connect to kafka broker running in your local host (docker). Change this to your kafka broker if needed
kafka_broker = 'venus:9092'
```

```
In [ ]: producer = KafkaProducer(bootstrap_servers=[kafka_broker])
```

```
In [ ]: for i in range(100):
    s = 'message #{} from producer-0'.format(i)
    producer.send('sample', s.encode('utf-8'))
    time.sleep(2)
```

Kafka Consumer

```
In [ ]: # import required libraries
from kafka import KafkaConsumer, TopicPartition
```

```
In [ ]: # Connect to kafka broker running in your local host (docker). Change this to your kafka broker if needed
kafka_broker = 'venus:9092'
```

```
In [ ]: consumer = KafkaConsumer(
    'sample',
    bootstrap_servers=[kafka_broker],
    enable_auto_commit=True,
    value_deserializer=lambda x: x.decode('utf-8'))
```

```
In [ ]: print('Running Consumer')
for message in consumer:
    print(message)
    print(message.value, message.offset, message.timestamp)
```

More Kafka Consumer

```
In [ ]: # import required libraries
         from kafka import KafkaConsumer

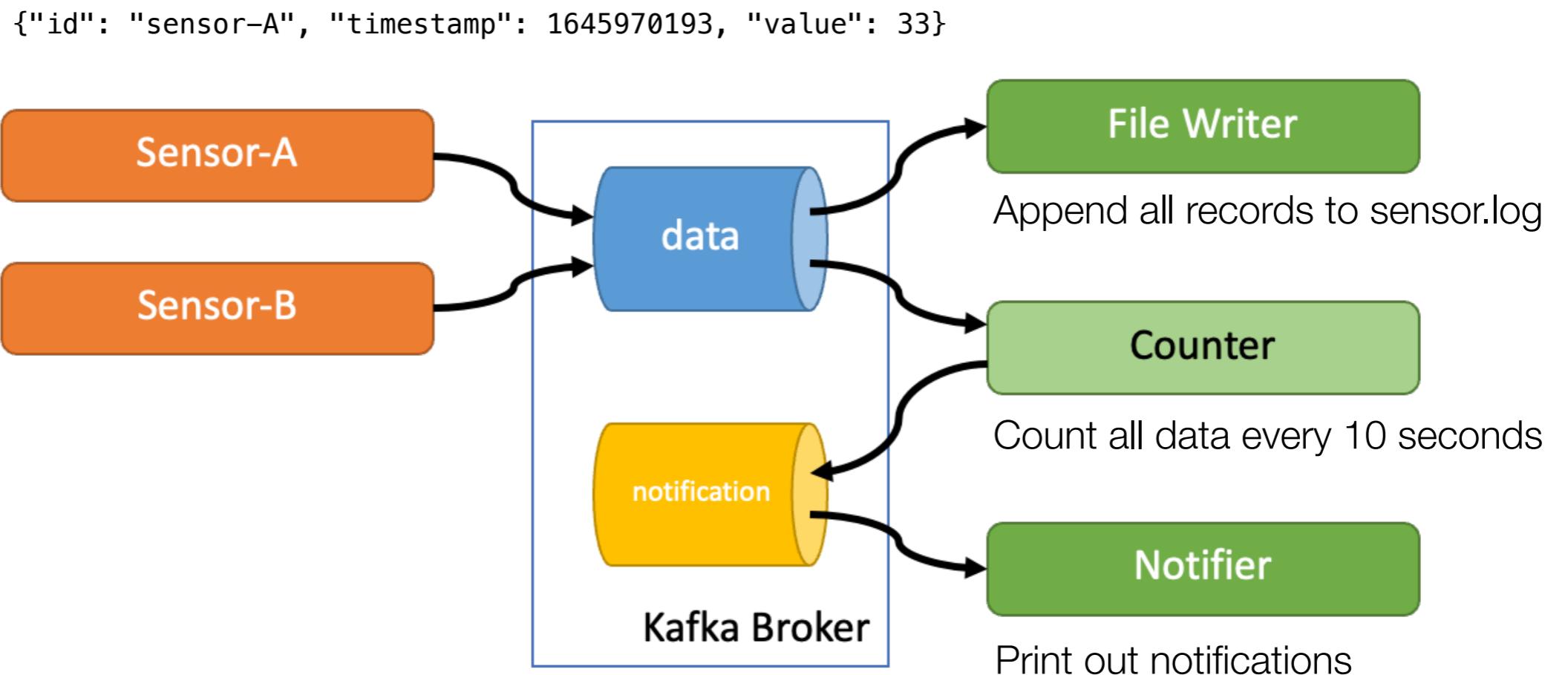
In [ ]: # Connect to kafka broker running in your local host (docker). Change this to your kafka broker if needed
         kafka_broker = 'venus:9092'

In [ ]: consumer = KafkaConsumer(
         bootstrap_servers=[kafka_broker],
         enable_auto_commit=True,
         value_deserializer=lambda x: x.decode('utf-8'))

In [ ]: consumer.subscribe('sample')

In [ ]: print('Running Consumer #2')
while(True):
    results = consumer.poll(timeout_ms=1000)
    if len(results) == 0:
        print('no message')
    else:
        for tp, messages in results.items():
            print('topic = {} -- total {} messages'.format(tp.topic, len(messages)))
            print('-----')
            for message in messages:
                print(message)
    print('#####')
```

More Complex Example



Sensor-A Producer

```
In [ ]: # import required libraries
from kafka import KafkaProducer
import time, random, json

In [ ]: sensor_id = 'sensor-A'

In [ ]: # Connect to kafka broker running in your local host (docker). Change this to your kafka broker if needed
kafka_broker = 'venus:9092'
topic = 'data'

In [ ]: producer = KafkaProducer(bootstrap_servers=[kafka_broker])

In [ ]: while(True):
    ts = round(time.time())
    value = round(random.uniform(27, 33))
    data = {
        'id': sensor_id,
        'timestamp': ts,
        'value': value
    }
    s = json.dumps(data)
    print(s)
    producer.send(topic, s.encode('utf-8'))
    time_to_sleep = round(random.uniform(1,3))
    time.sleep(time_to_sleep)
```

FileWriter Consumer

```
In [ ]: # import required libraries
from kafka import KafkaConsumer

In [ ]: # Connect to kafka broker running in your local host (docker). Change this to your kafka broker if needed
kafka_broker = 'venus:9092'

In [ ]: consumer = KafkaConsumer(
    'data',
    bootstrap_servers=[kafka_broker],
    enable_auto_commit=True,
    value_deserializer=lambda x: x.decode('utf-8'))

In [ ]: print('Running FileWriter')
counter = 0
for message in consumer:
    with open('sensor.log', 'a') as f:
        f.write(message.value+'\n')
    counter += 1
    if counter%10 == 0:
        print('{} records'.format(counter))
    if counter >= 100:
        break
```

```
In [ ]: import threading
import json
import time
from kafka import KafkaProducer, KafkaConsumer
```

```
In [ ]: class Counter:
    def __init__(self):
        self._lock = threading.Lock()
        self.reset()

    def reset(self):
        with self._lock:
            self.counter = 0

    def incr(self, value):
        with self._lock:
            self.counter += value

    def get(self):
        return self.counter
```

```
In [ ]: # Connect to kafka broker running in your local host (docker). Change this to your kafka broker if needed
kafka_broker = 'venus:9092'
data_topic = 'data'
notification_topic = 'notification'
```

```
In [ ]: producer = KafkaProducer(bootstrap_servers=[kafka_broker])
```

```
In [ ]: consumer = KafkaConsumer(
    bootstrap_servers=[kafka_broker],
    enable_auto_commit=True,
    value_deserializer=lambda x: x.decode('utf-8'))
consumer.subscribe(data_topic)
```

```
In [ ]: def monitor_thread(interval, counters, topic):
    print('[monitor] starting')
    while True:
        time.sleep(interval)
        for id in counters:
            count = counters[id].get()
            counters[id].reset()
            s = '{} - {} messages during last {}'.format(id, count, interval)
            producer.send(topic, s.encode('utf-8'))
    print('send notification', flush=True)
```

```
In [ ]: monitor_interval = 10
counters = {}
monitor = threading.Thread(target=monitor_thread, args=(monitor_interval, counters, notification_topic), daemon=True)
monitor.start()
```

```
In [ ]: for message in consumer:
    m = message.value
    data = json.loads(m)
    if 'id' in data:
        if data['id'] not in counters:
            counters[data['id']] = Counter()
            counters[data['id']].incr(1)
    else:
        print(data, flush=True)
```

AVRO

- Row-based schema-oriented data serialization framework
 - Use JSON to define schema
 - Serialize data in a compact binary format
 - Similar to Java Object Serialization
 - Support both primitive and complex data types (allows hierarchical data structure)
- Primary usage in Big Data
 - Apache Hadoop for persistent data (input/output/checkpoint)
 - A wire format for communication (e.g. Kafka, Spark Streaming)
- Have libraries for many programming languages

Create Home Competitions Datasets Code Discussions Courses More Your Work RECENTLY VIEWED Squid Game Netflix Tw... Netflix Movie Rating D... Netflix Movies and TV ... HPA 2020 16-Bit Traini... Netflix Original Films &...

 Dataset

Squid Game Netflix Twitter Data

This data set contains twitter dump for the hashtag #squidgame.

 Deep Contractor • updated 18 days ago (Version 11)

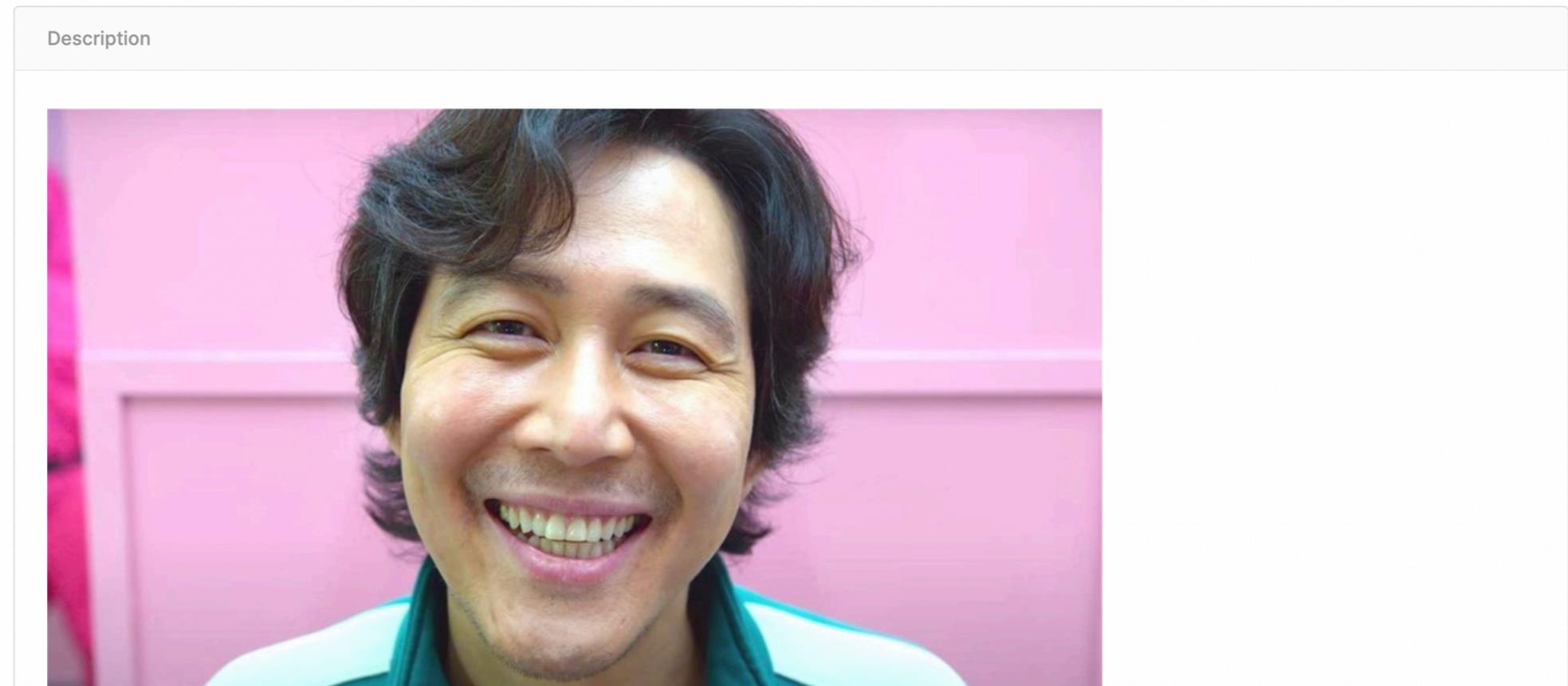
[Data](#) [Code \(2\)](#) [Discussion](#) [Activity](#) [Metadata](#)

[Download \(26 MB\)](#) [New Notebook](#) :

 Usability 10.0

 License CC0: Public Domain

 Tags arts and entertainment, online communities, text data, beginner, exploratory data analysis



- The dataset contains the recent tweets about the record-breaking Netflix show "Squid Game"
- The data is collected using tweepy Python package to access Twitter API.

 View Active Events**Data Explorer**

201 MB

/ tweets_v0.csv (201 MB)

1 / 57

< tweets_v8.csv (26.1 MB)

Detail Compact Column

A user_name	A user_locati...	A user_descr...	D user_creat...	# user_follo...	# user_friends	# user_favou...	✓ user_verified	D date	A text
the _ündér-ratèd niggáh		@ManUtd die hard❤️❤️↳↳ YOLO J'ai besoin de quelqu'un qui peut m'aimer au pire😊 Non, je ne...	2019-09-06 19:24:57+00:00	581	1035	8922	False	2021-10-06 12:05:38+00:00	When life hits and the same time poverty strikes you Gong Yoo : Lets play a game #SquidGame #Netfli...
Best uncle on planet earth			2013-05-08 19:35:26+00:00	741	730	8432	False	2021-10-06 12:05:22+00:00	That marble episode of #SquidGame ruined me. 😭😭
marcie		animal crossing. chicken nuggets. baby yoda. smol animals. tv shows. 🏳 pronouns: any	2009-02-21 10:31:30+00:00	562	1197	62732	False	2021-10-06 12:05:22+00:00	#Squidgame time
YoMo.Mdp	Any pronouns	Where the heck is the karma I'm going on my school grave brb #Technosupport	2021-02-14 13:21:22+00:00	3	277	1341	False	2021-10-06 12:05:04+00:00	//Blood on 1st slide I'm joining the squidgame thing, I'm already dead by sugar honeycomb ofc #Squi...
Laura Reactions	France	I talk and I make reactions videos about shows I love #theexpans #peakyblinders #thelastkingdom #la...	2018-12-19 20:38:28+00:00	330	152	2278	False	2021-10-06 12:05:00+00:00	The two first games, players were killed by the mask guys ; the bloody night and the third game, the...
Peyman KA	United Kingdom	Official @KardiaChain \$KAI Ambassador Marketing Advisor @kephigallery Graphics and Film Artist https...	2018-01-27 12:07:31+00:00	546	318	6265	False	2021-10-06 12:04:54+00:00	\$THG Going to explode to 4B Marketcap very soon. The world first MOBA This game is on another level!...
Aeriaaaa♡		Fujoshi 🐾/ Thai BL- obsessed/Always distracted by poetry 🌸/ CAP 🐾	2021-06-01 14:08:10+00:00	14	110	518	False	2021-10-06 12:04:45+00:00	@B_hundred_Hyun pls use that gun on me. 😞 #BAEKHYUN #EXO #weareoneEXO #SquidGame https://t.co/ksk...

AVRO Schema File

complex types (record, enum, array, map, union, and fixed)

```
1  {
2      "namespace": "example.avro.chula",
3      "type": "record",
4      "name": "squid_tweets",
5      "fields": [
6          {"name": "user_name", "type": "string"},
7          {"name": "user_location", "type": ["null", "string"]},
8          {"name": "user_description", "type": "string"},
9          {"name": "user_created", "type": {"type": "long", "logicalType": "local-timestamp-millis"}},
10         {"name": "user_followers", "type": "int"},
11         {"name": "user_friends", "type": "int"},
12         {"name": "user_favourites", "type": "int"},
13         {"name": "user_verified", "type": "boolean"},
14         {"name": "date", "type": {"type": "long", "logicalType": "local-timestamp-millis"}},
15         {"name": "text", "type": "string"},
16         {"name": "source", "type": "string"},
17         {"name": "is_retweet", "type": "boolean"}
18     ]
19 }
```

primitive types (null, boolean, int, long, float, double, bytes, and string)

AVRO Example in Python

```
In [1]: import pandas as pd
import fastavro as favro
import json, os

In [2]: schema = json.load(open('tweets_v8.avsc'))
parsed_schema = favro.parse_schema(schema)

In [3]: parse_dates = ['user_created', 'date']
df = pd.read_csv('tweets_v8.csv', parse_dates=parse_dates, keep_default_na=False)
df.shape

Out[3]: (80019, 12)

In [4]: records = df.to_dict('records')

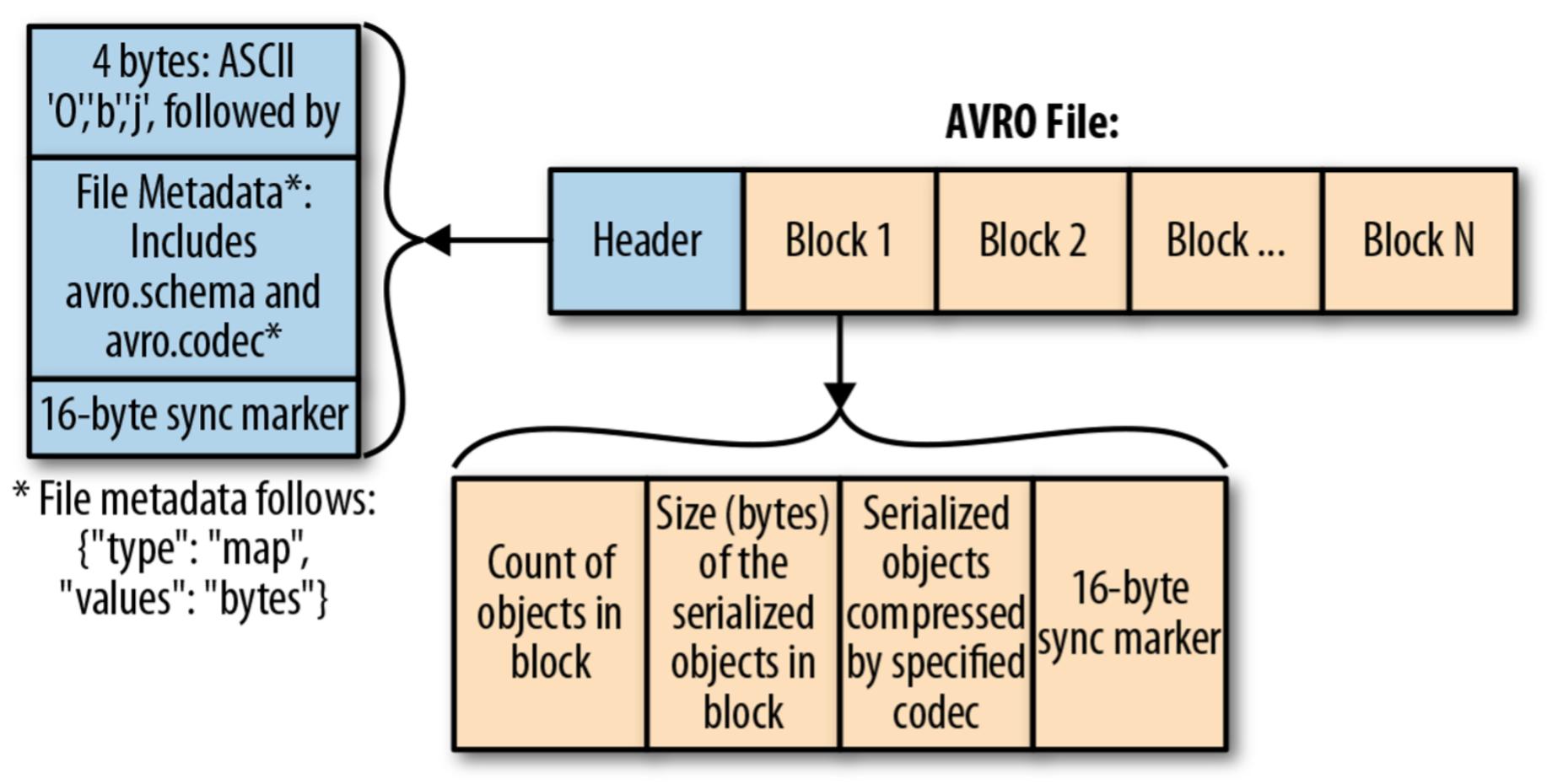
In [5]: with open('tweets_v8.avro', 'wb') as out:
    favro.writer(out, parsed_schema, records)

In [6]: with open('tweets_v8_deflate.avro', 'wb') as out:
    favro.writer(out, parsed_schema, records, codec='deflate')

In [7]: print('tweets_v8.csv = {:.5.2f}MB'.format(os.path.getsize('tweets_v8.csv')/(1024*1024)))
print('tweets_v8.avro = {:.5.2f}MB'.format(os.path.getsize('tweets_v8.avro')/(1024*1024)))
print('tweets_v8_deflate.avro = {:.5.2f}MB'.format(os.path.getsize('tweets_v8_deflate.avro')/(1024*1024)))

tweets_v8.csv = 24.89MB
tweets_v8.avro = 20.49MB
tweets_v8_deflate.avro = 11.51MB
```

AVRO File Structure



- Each row is “serialized”, multiple rows store in each block
- Data compressed per block
- Block's sync marker is for MapReduce file splitting

00000000	4F 62 6A 01 04 14 61 76 72 6F 2E 63 6F 64 65 63	Obj...avro.codec
00000010	08 6E 75 6C 6C 16 61 76 72 6F 2E 73 63 68 65 6D	.null.avro.schem
00000020	61 DC 0A 7B 22 74 79 70 65 22 3A 20 22 72 65 63	a■.{"type": "rec
00000030	6F 72 64 22 2C 20 22 6E 61 6D 65 22 3A 20 22 65	ord", "name": "e
00000040	78 61 6D 70 6C 65 2E 61 76 72 6F 2E 63 68 75 6C	xample.avro.chul
00000050	61 2E 73 71 75 69 64 5F 74 77 65 65 74 73 22 2C	a.squid_tweets",
00000060	20 22 66 69 65 6C 64 73 22 3A 20 5B 7B 22 6E 61	"fields": [{"na
00000070	6D 65 22 3A 20 22 75 73 65 72 5F 6E 61 6D 65 22	me": "user_name"
00000080	2C 20 22 74 79 70 65 22 3A 20 22 73 74 72 69 6E	, "type": "strin
00000090	67 22 7D 2C 20 7B 22 6E 61 6D 65 22 3A 20 22 75	g"}, {"name": "u
000000A0	73 65 72 5F 6C 6F 63 61 74 69 6F 6E 22 2C 20 22	ser_location", "
000000B0	74 79 70 65 22 3A 20 5B 22 6E 75 6C 6C 22 2C 20	type": ["null",
000000C0	22 73 74 72 69 6E 67 22 5D 7D 2C 20 7B 22 6E 61	"string"]}, {"na
000000D0	6D 65 22 3A 20 22 75 73 65 72 5F 64 65 73 63 72	me": "user_descr
000000E0	69 70 74 69 6F 6E 22 2C 20 22 74 79 70 65 22 3A	ption", "type":
000000F0	20 22 73 74 72 69 6E 67 22 7D 2C 20 7B 22 6E 61	"string"}, {"na
00000100	6D 65 22 3A 20 22 75 73 65 72 5F 63 72 65 61 74	me": "user_creat
00000110	65 64 22 2C 20 22 74 79 70 65 22 3A 20 7B 22 6C	ed", "type": {"l
00000120	6F 67 69 63 61 6C 54 79 70 65 22 3A 20 22 6C 6F	ogicalType": "lo

19 #SquidGame... https://t.co/N4UGv9hxx8",Twitter Web App,False
 20 Laura Reactions,France,I talk and I make reactions videos about shows I love #theexpansé #peakyblinders #thelastkingdom
 #lacasadepapel #atla #theboys #thewitcher,2018-12-19 20:38:28+00:00,330,152,2278,False,2021-10-06 12:05:00+00:00,"The two
 first games, players were killed by the mask guys ; the bloody night and the third game, they killed each o...
 https://t.co/Qf057XDJ7C",Twitter Web App,False
 21 Peyman KAI,United Kingdom,"Official @KardiaChain \$KAI Ambassador
 22 Marketing Advisor @kephigallery
 23 Graphics and Film Artist https://t.co/0sT0SEKrHt",2018-01-27 12:07:31+00:00,546,318,6265,False,2021-10-06
 12:04:54+00:00,"\$THG

00000690	74 65 72 20 57 65 62 20 41 70 70 00 1E 4C 61 75
000006A0	72 61 20 52 65 61 63 74 69 6F 6E 73 02 0C 46 72
000006B0	61 6E 63 65 94 02 49 20 74 61 6C 6B 20 61 6E 64
000006C0	20 49 20 6D 61 6B 65 20 72 65 61 63 74 69 6F 6E
000006D0	73 20 76 69 64 65 6F 73 20 61 62 6F 75 74 20 73
000006E0	68 6F 77 73 20 49 20 6C 6F 76 65 20 23 74 68 65
000006F0	65 78 70 61 6E 73 65 20 23 70 65 61 6B 79 62 6C
00000700	69 6E 64 65 72 73 20 23 74 68 65 6C 61 73 74 6B
00000710	69 6E 67 64 6F 6D 20 23 6C 61 63 61 73 61 64 65
00000720	70 61 70 65 6C 20 23 61 74 6C 61 20 23 74 68 65
00000730	62 6F 79 73 20 23 74 68 65 77 69 74 63 68 65 72
00000740	C0 A6 87 83 F9 59 94 05 B0 02 CC 23 00 C0 E7 F0
00000750	D7 8A 5F 9C 02 54 68 65 20 74 77 6F 20 66 69 72
00000760	73 74 20 67 61 6D 65 73 2C 20 70 6C 61 79 65 72
00000770	73 20 77 65 72 65 20 6B 69 6C 6C 65 64 20 62 79
00000780	20 74 68 65 20 6D 61 73 6B 20 67 75 79 73 20 3B
00000790	20 74 68 65 20 62 6C 6F 6F 64 79 20 6E 69 67 68
000007A0	74 20 61 6E 64 20 74 68 65 20 74 68 69 72 64 20
000007B0	67 61 6D 65 2C 20 74 68 65 79 20 6B 69 6C 6C 65
000007C0	64 20 65 61 63 68 20 6F E2 80 A6 20 68 74 74 70
000007D0	73 3A 2F 2F 74 2E 63 6F 2F 51 66 30 35 37 58 44
000007E0	4A 37 43 1E 54 77 69 74 74 65 72 20 57 65 62 20
000007F0	41 70 70 00 26 50 65 79 6D 61 6E 20 F0 9F 85 9A

ter Web App..| Lau
 ra Reactions..| Fr
 anceö.I talk and
 I make reaction
 s videos about s
 hows I love #the
 expansé #peakybl
 inders #thelastk
 ingdom #lacasad
 papel #atla #the
 boys #thewitcher
 Làçâ·Yö..||#..||_
 ||è_£|The two fir
 st games, player
 s were killed by
 the mask guys ;
 the bloody nigh
 t and the third
 game, they kille
 d each oΓÇª http
 s://t.co/Qf057XD
 J7C.Twitter Web
 App.&Peyman =fàÜ

```
{  
    "namespace": "sample.avro",  
    "type": "record",  
    "name": "Event",  
    "fields": [  
        {"name": "key", "type": "string"},  
        {"name": "value", "type": "string"},  
        {"name": "timestamp", "type": "long"}  
    ]  
}
```

sample.avsc

```
In [ ]: # import required libraries  
from kafka import KafkaProducer, KafkaConsumer  
import time
```

producer

```
In [ ]: import avro.schema  
import avro.io  
import io
```

```
In [ ]: schema_file = 'sample.avsc'  
schema = avro.schema.parse(open(schema_file).read())
```

```
In [ ]: def serialize(schema, obj):  
    bytes_writer = io.BytesIO()  
    encoder = avro.io.BinaryEncoder(bytes_writer)  
    writer = avro.io.DatumWriter(schema)  
    writer.write(obj, encoder)  
    return bytes_writer.getvalue()
```

```
In [ ]: # Connect to kafka broker running in your local host (docker). Change the  
kafka_broker = 'venus:9092'
```

```
In [ ]: producer = KafkaProducer(bootstrap_servers=[kafka_broker])
```

```
In [ ]: for i in range(100):  
    k = 'key{}'.format(i)  
    v = 'message #{}'.format(i)  
    o = {'key': k, 'value': v, 'timestamp': (int)(time.time()*1000)}  
    producer.send('avro', serialize(schema, o))  
    time.sleep(2)
```

consumer

```
In [ ]: # import required libraries
from kafka import KafkaConsumer
```

```
In [ ]: import avro.schema
import avro.io
import io
```

```
In [ ]: schema_file = 'sample.avsc'
schema = avro.schema.parse(open(schema_file).read())
```

```
In [ ]: def deserialize(schema, raw_bytes):
    bytes_reader = io.BytesIO(raw_bytes)
    decoder = avro.io.BinaryDecoder(bytes_reader)
    reader = avro.io.DatumReader(schema)
    return reader.read(decoder)
```

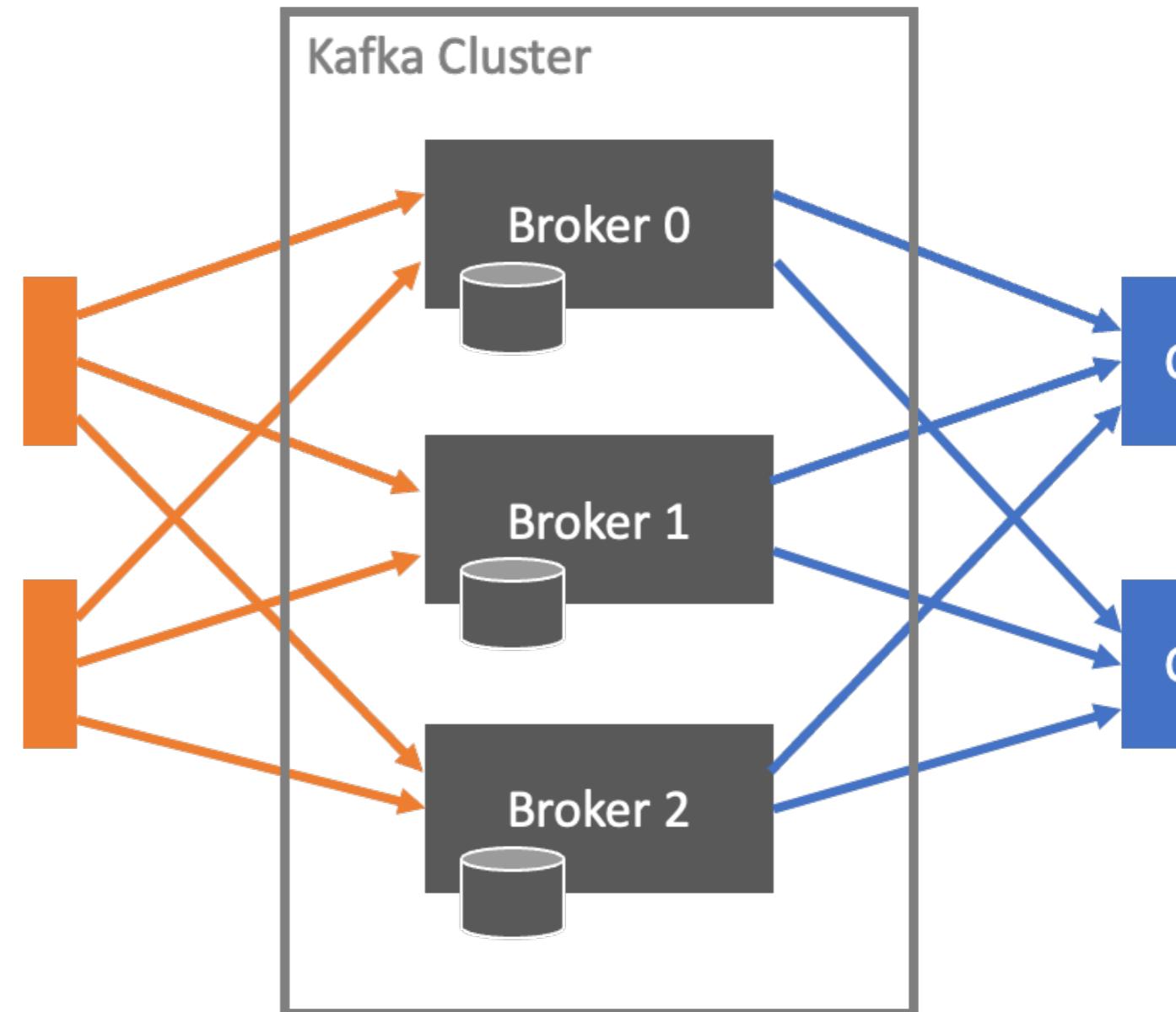
```
In [ ]: # Connect to kafka broker running in your local host (docker). Change this to your kafka broker if needed
kafka_broker = 'venus:9092'
```

```
In [ ]: consumer = KafkaConsumer(
    'avro',
    bootstrap_servers=[kafka_broker],
    enable_auto_commit=True,
    value_deserializer=lambda x: deserialize(schema, x))
```

```
In [ ]: print('Running Consumer with AVRO')
for message in consumer:
    print(message.value)
```

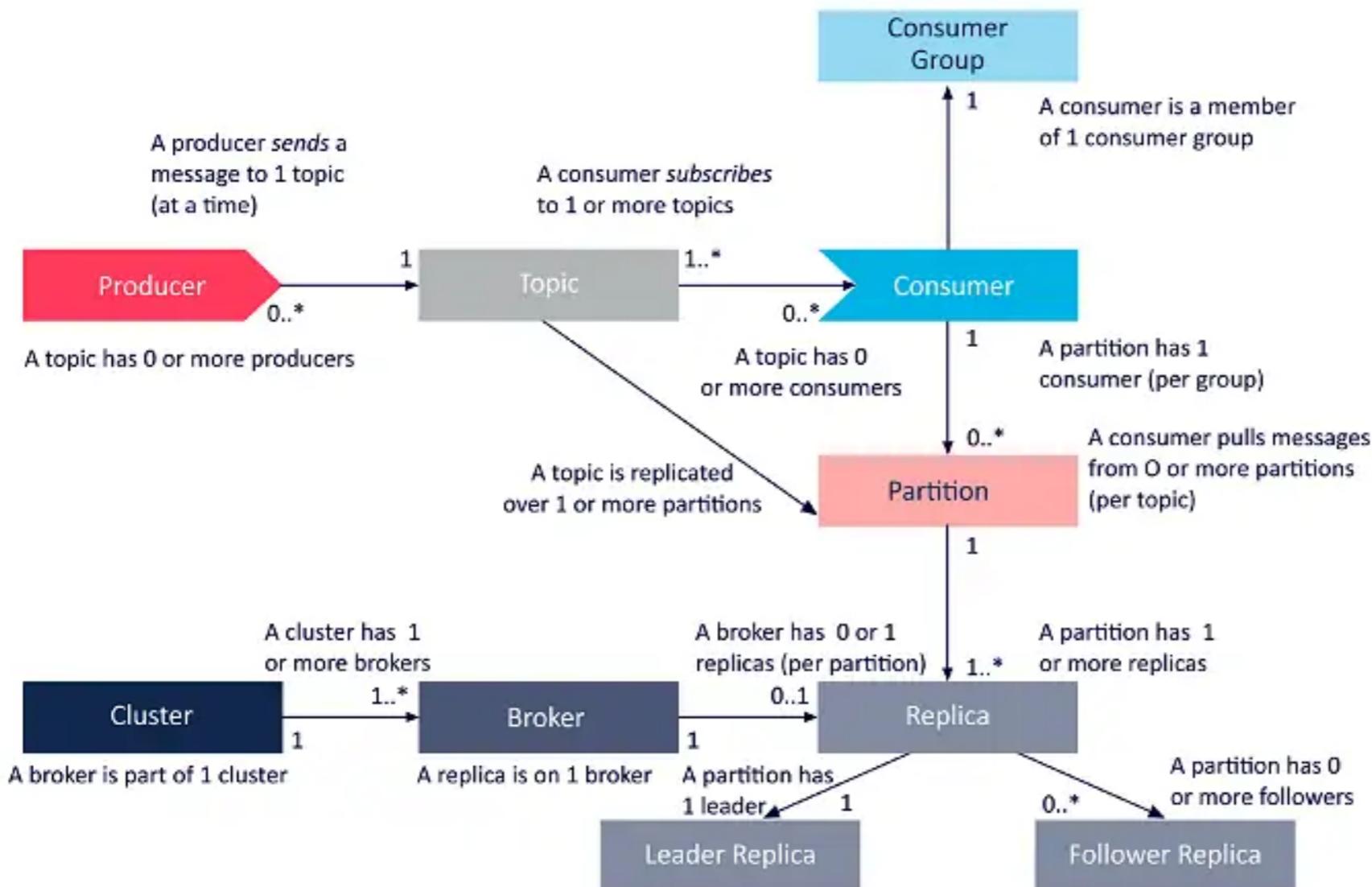
Kafka Architecture

Data Ingestion



ZooKeeper

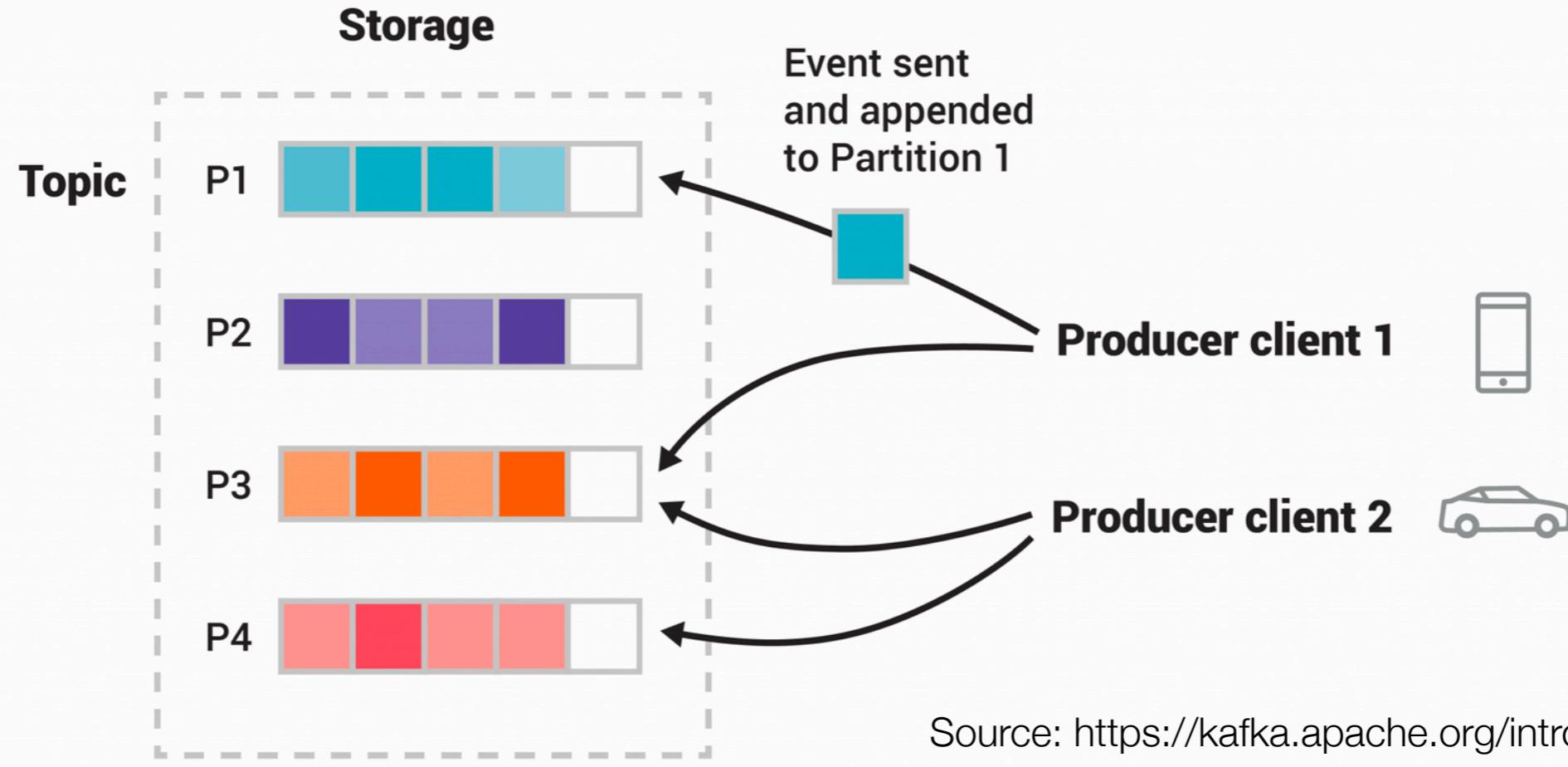
Apache Kafka Components



Kafka Partition

- A topic can be divided into multiple partitions
- Producer can send messages to a topic and messages will be distributed among partitions without duplication
- Orders of messages in the same partitions are guaranteed, but not across partitions
- For Kafka cluster with multiple brokers, partitions will be evenly distributed between those brokers
- Partition can be replicated to multiple brokers for failover with one broker becomes a leader of the partition and other brokers become followers

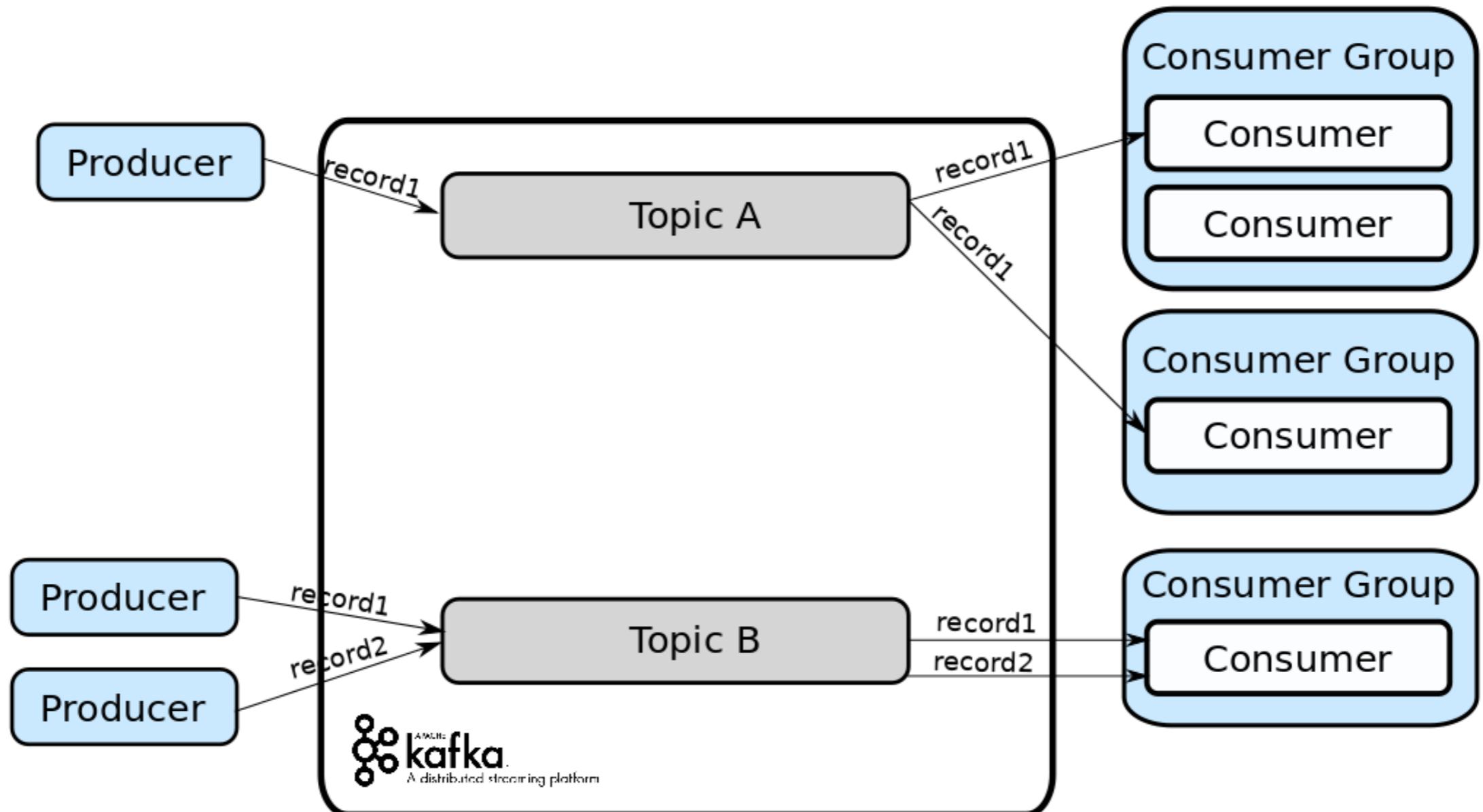
Anatomy of Kafka Topic



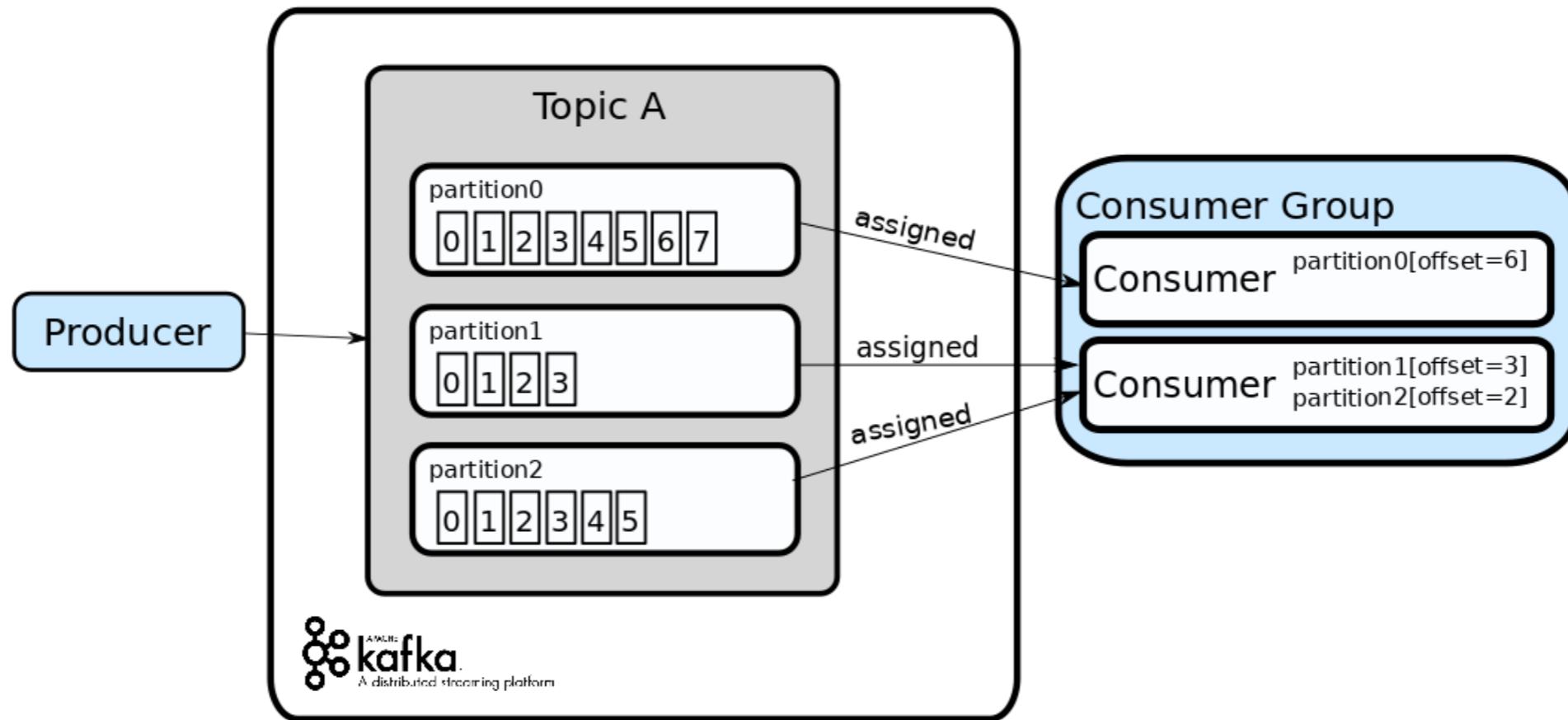
Kafka Consumer Group

- Consumers can form a consumer group
- Consumers in the same group will be assigned to read from different partitions of the topic; only one consumer will be assigned to each partition
- If there are more partitions than consumers in the group, some consumers may be assigned to handle more than one partition
- If there are more consumers in the group than partitions, some consumers will not be assigned any partition
- If an assigned consumer is out of contact from a broker, another consumer in the group will be assigned to handle the partition
- Kafka periodically reassigns consumers in the group to rebalance the partition handling of the consumer group

Scalability and Reliability with Consumer Group

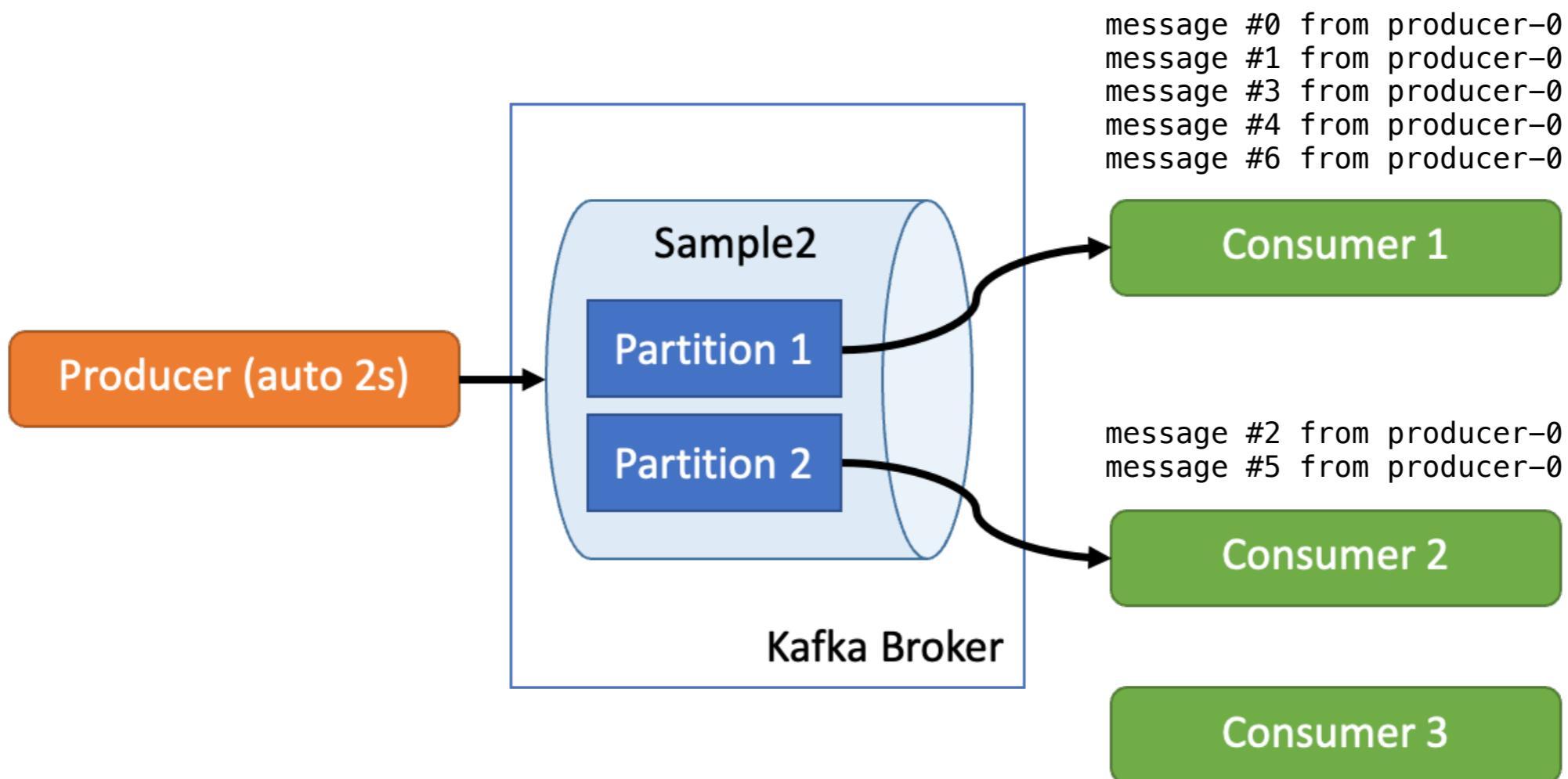


Partitions / Consumer / Consumer Group



- All consumers must send heartbeats every **heartbeat.interval.ms**
- A consumer who has been out of contact for **session.timeout.ms** will be kicked out of the group
- A group will be rebalance if there is no consumer processes records within **max.poll.interval.ms**

Consumer Group Example



Consumer Group Example

```
In [ ]: # import required libraries
from kafka import KafkaConsumer
from time import localtime, strftime
```

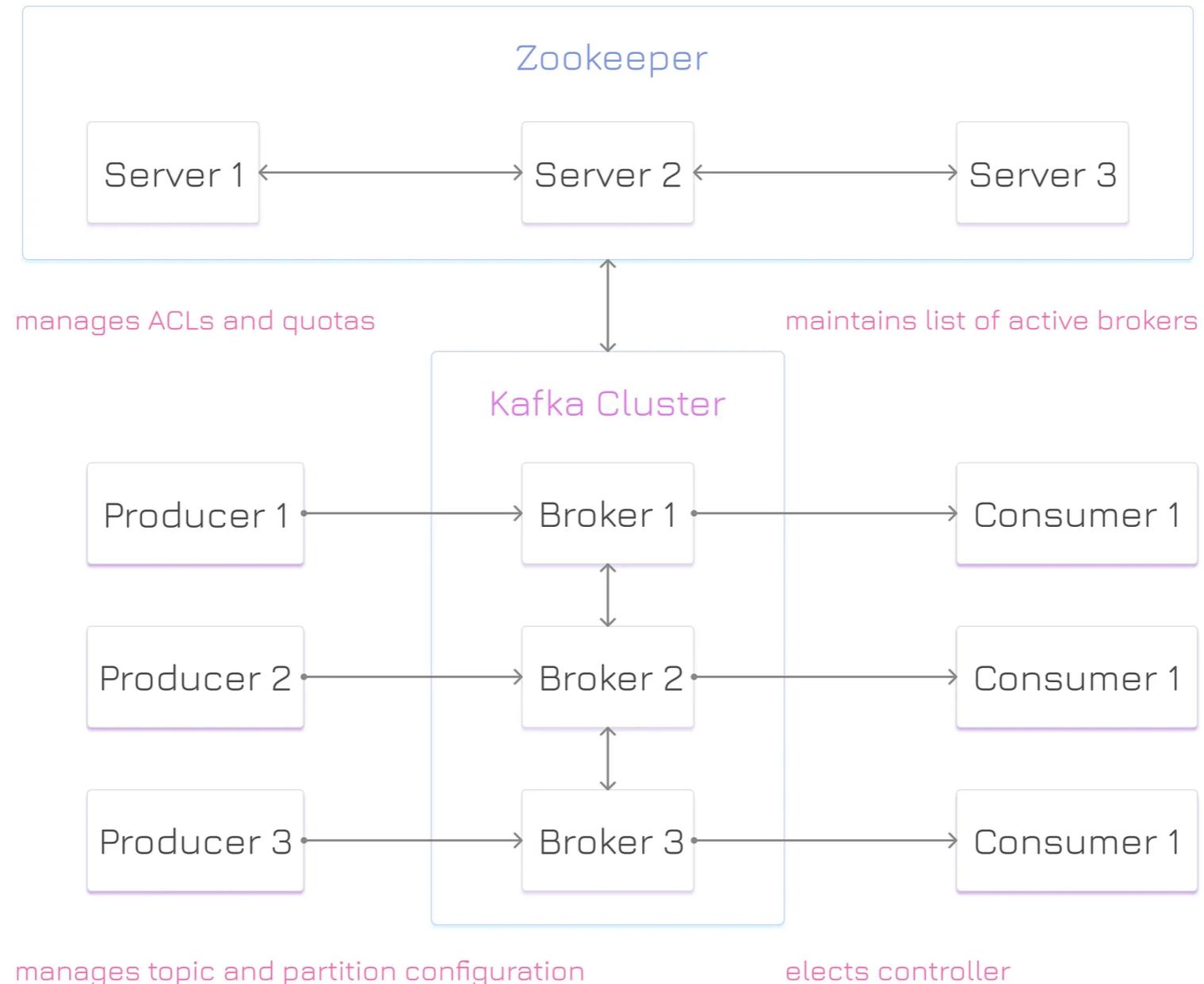
```
In [ ]: # Connect to kafka broker running in your local host (docker). Change this to your kafka broker if needed
kafka_broker = 'venus:9092'
```

```
In [ ]: consumer = KafkaConsumer(
    'sample2',
    bootstrap_servers=[kafka_broker],
    enable_auto_commit=True,
    group_id='my-group',
    session_timeout_ms=6000,
    max_poll_interval_ms=6000,
    value_deserializer=lambda x: x.decode('utf-8'))
```

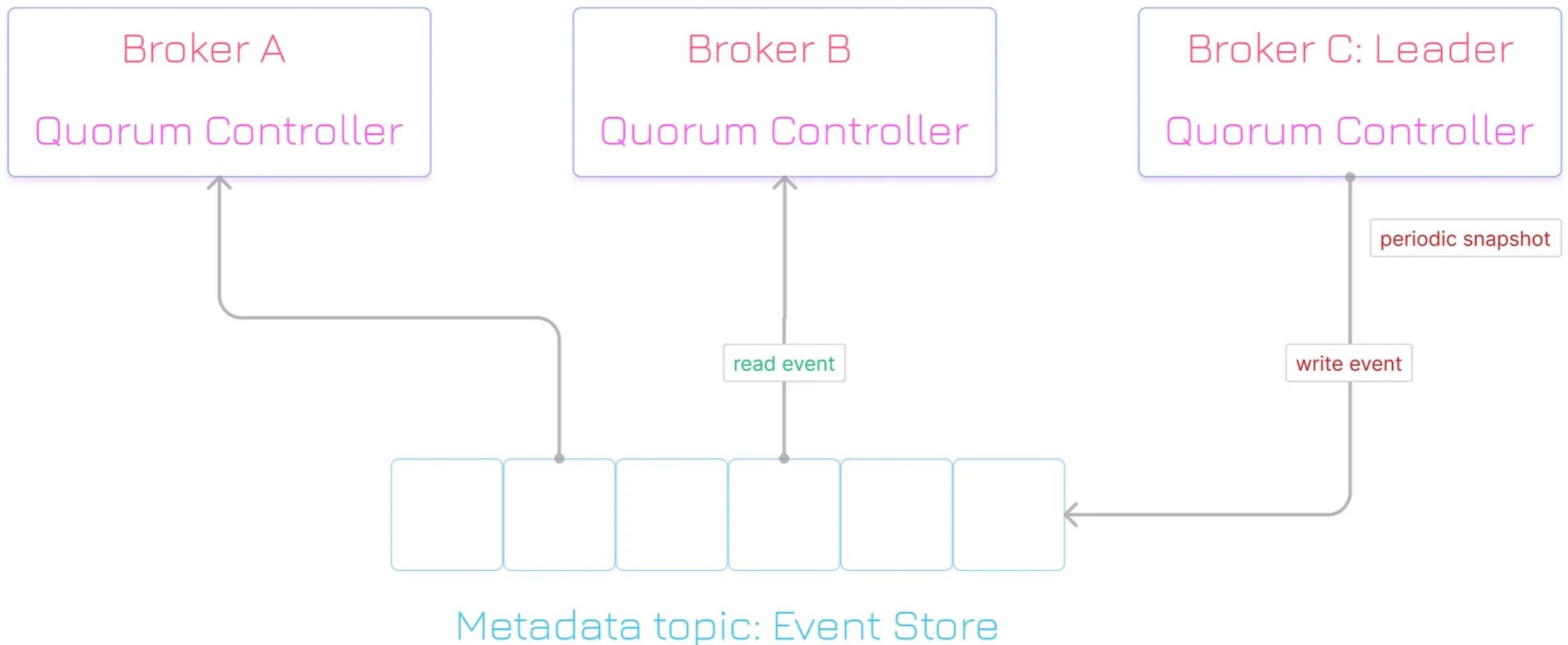
```
In [ ]: print('Running Consumer Group')
for message in consumer:
    ts = strftime("%H:%M:%S", localtime())
    print('[{}] {}'.format(ts, message.value, message.offset))
    print(consumer.assignment())
```

```
In [ ]: consumer.close()
```

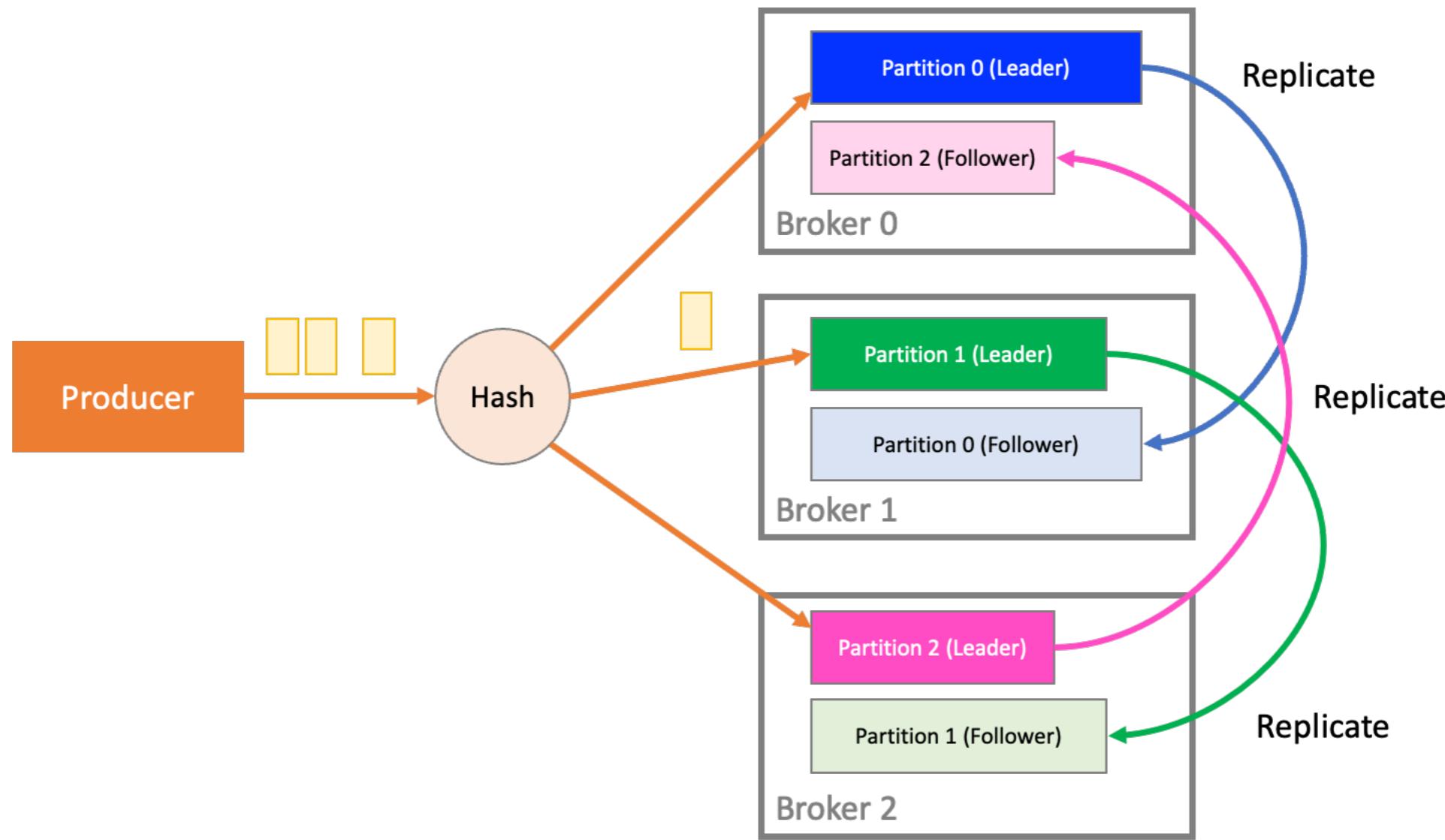
Kafka Zookeeper-based Cluster



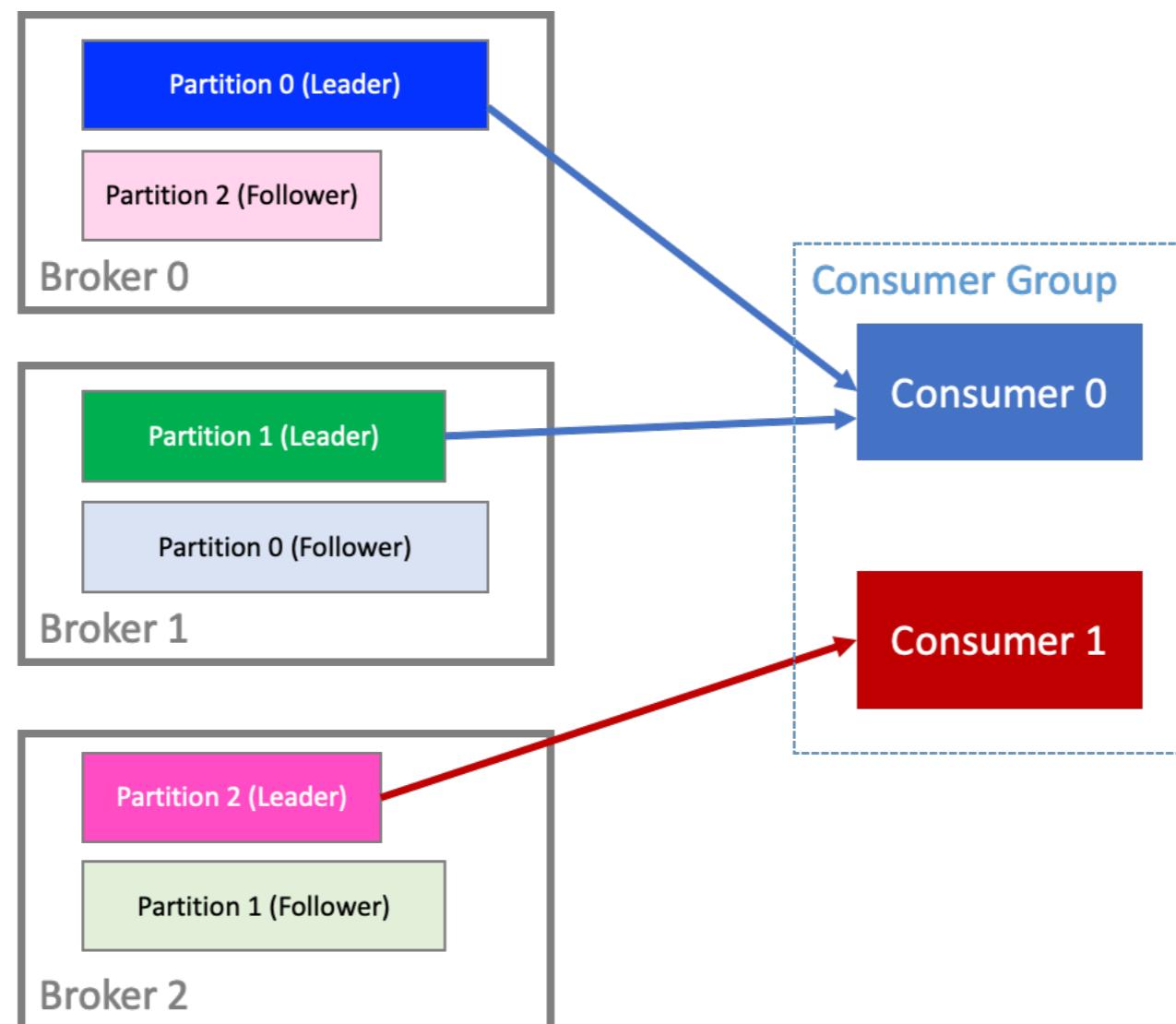
Kafka Kraft-based Cluster



Partitions with Replication in Broker Cluster: Producer



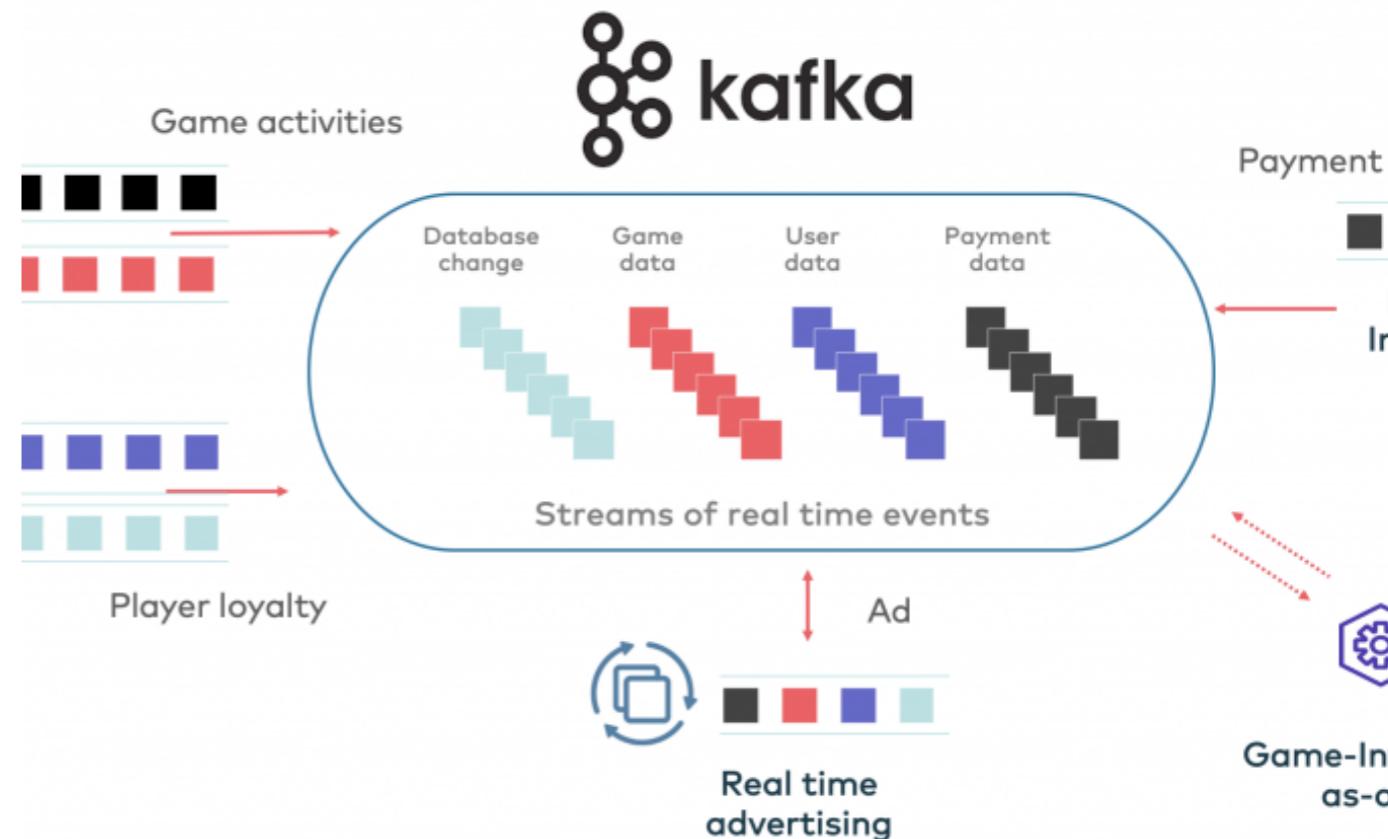
Partitions with Replication in Broker Cluster: Consumer



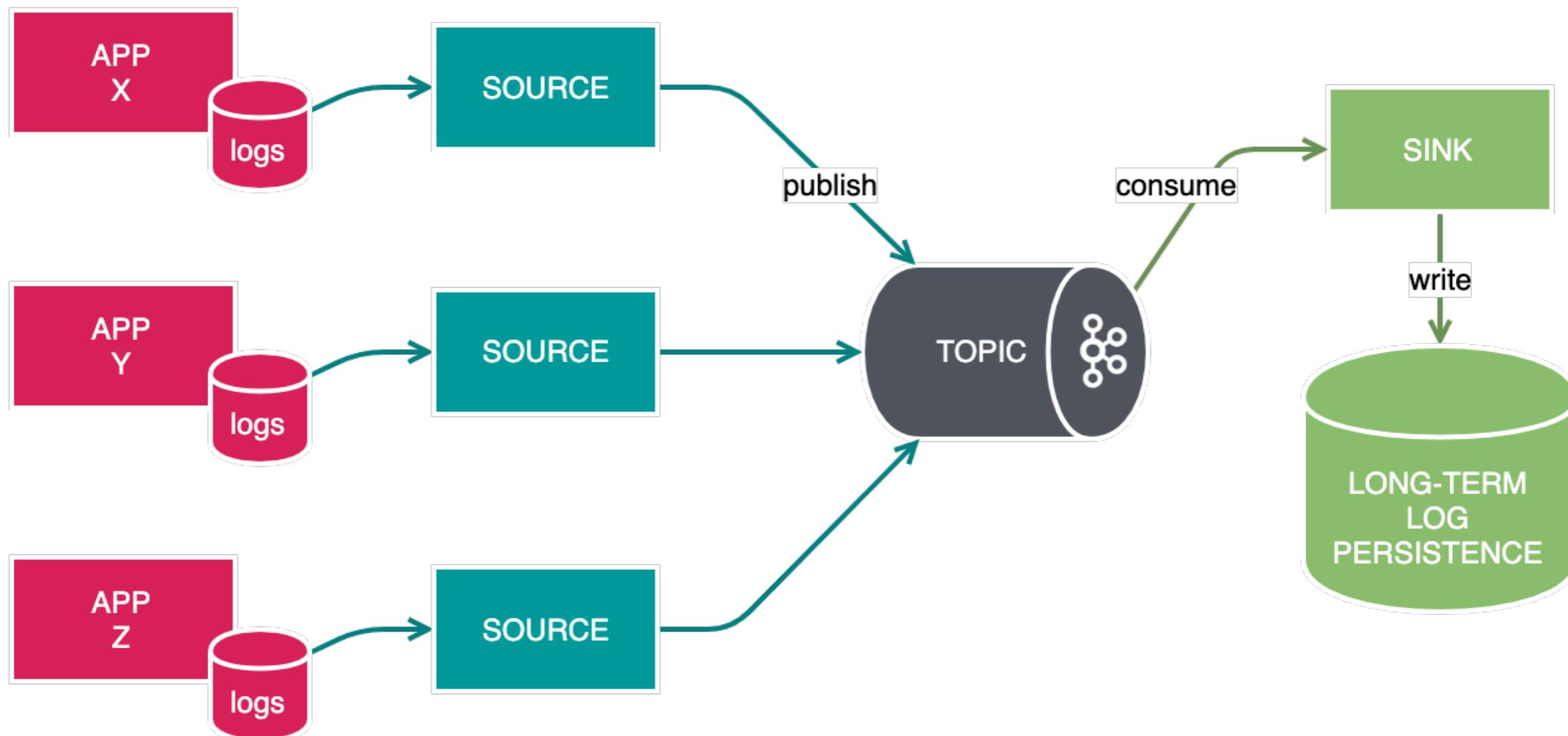
Kafka Use Cases

Data Ingestion

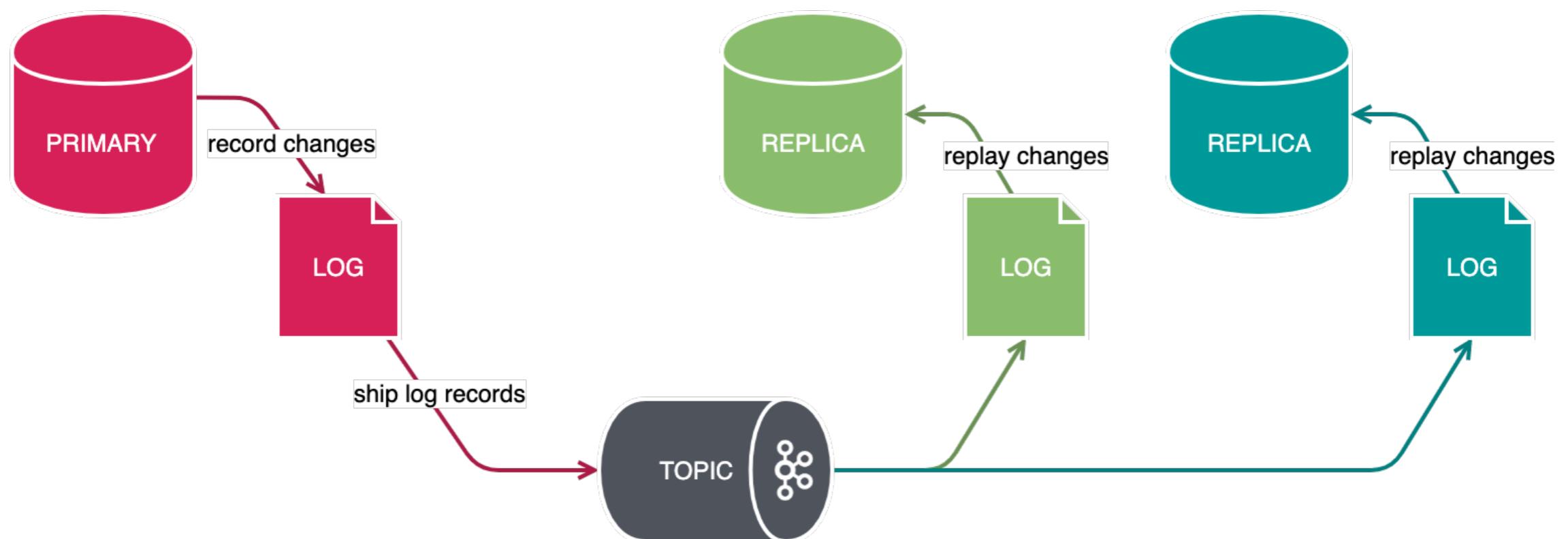
Game network



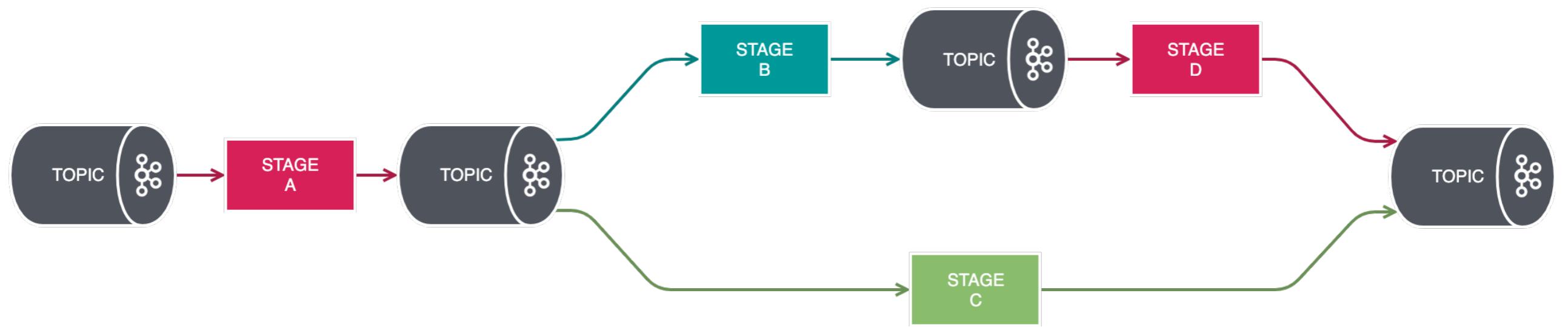
Kafka Use Cases: Log Aggregation



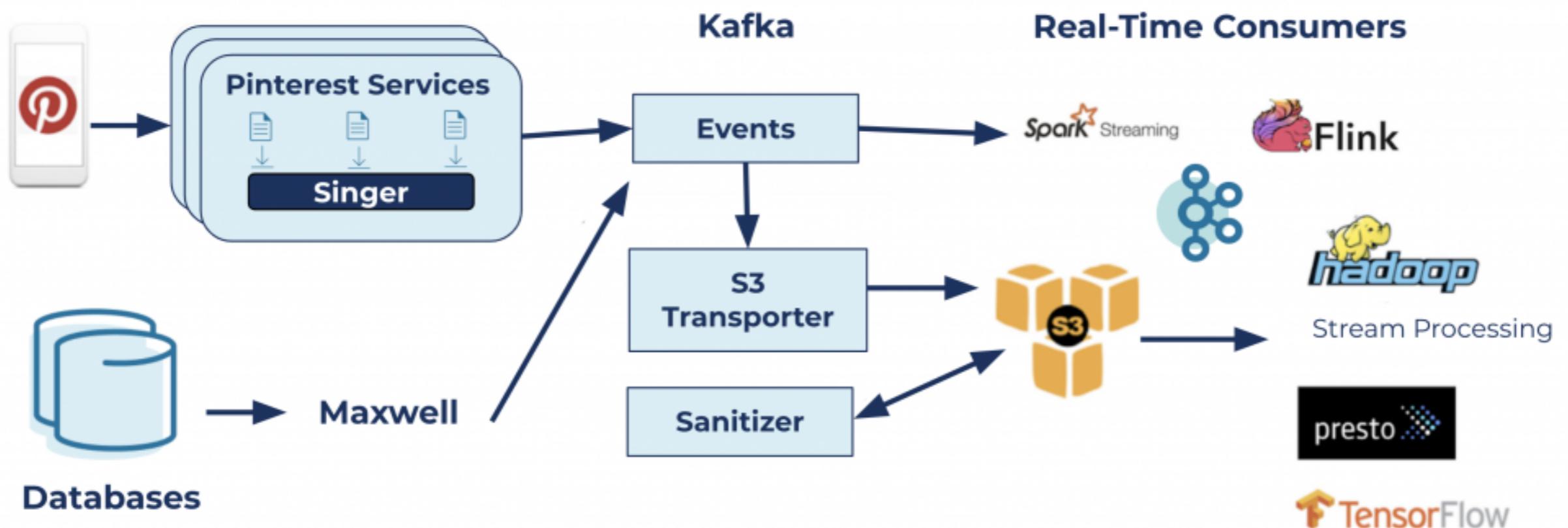
Kafka Use Cases: Log Shipping



Kafka Use Cases: Event Driven Architecture Pipeline

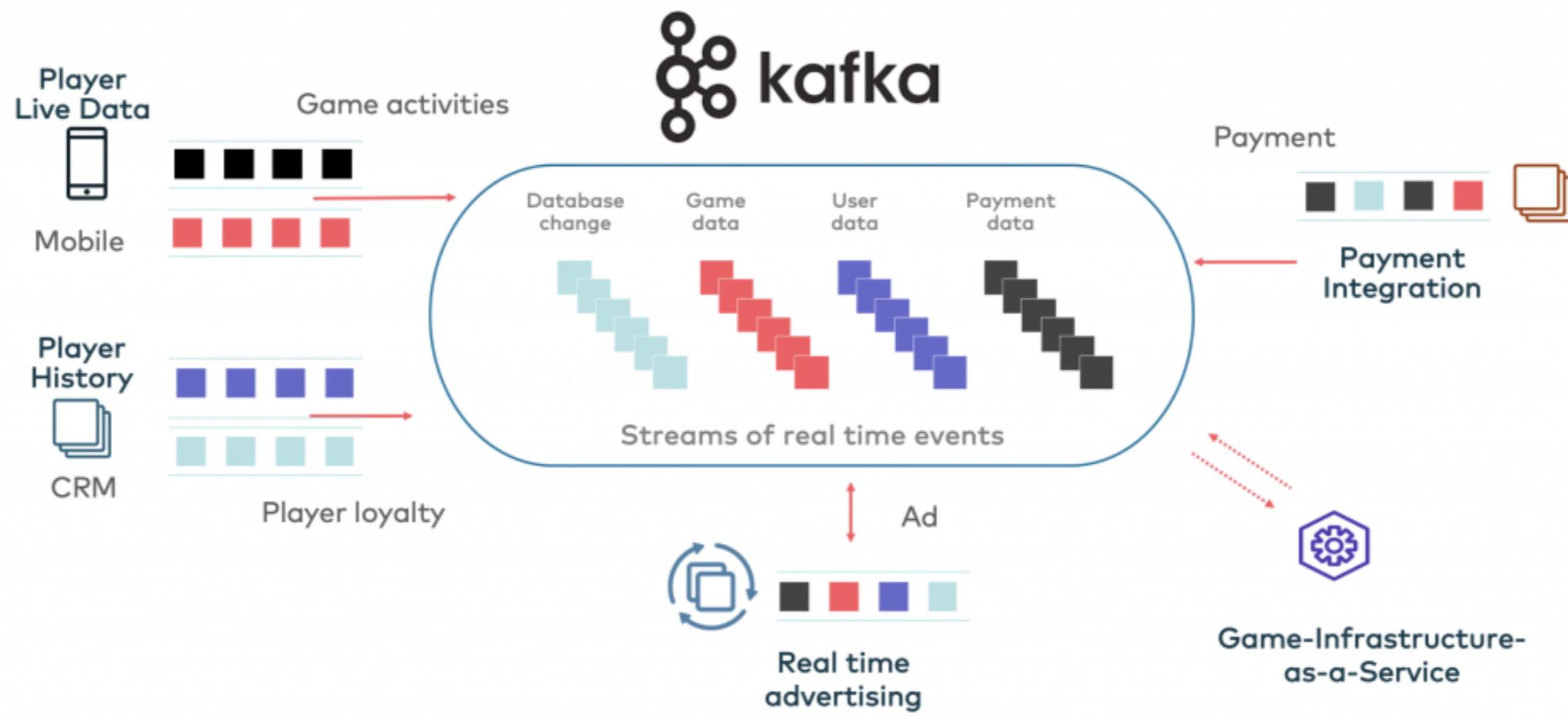


Kafka @ Pinterest



Source: <https://www.confluent.io/blog/running-kafka-at-scale-at-pinterest/>

Kafka and Gaming Industry



Apache Kafka in the Gaming Industry – @KaiWaehner - www.kai-waehner.de

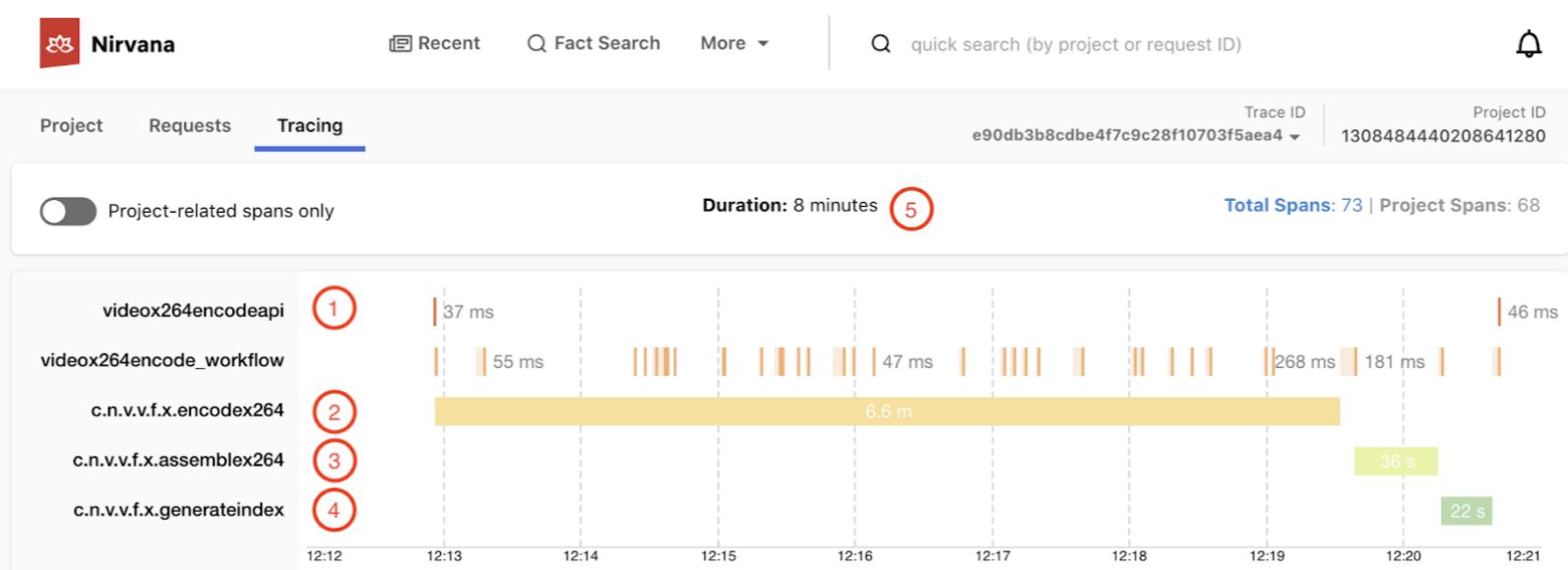
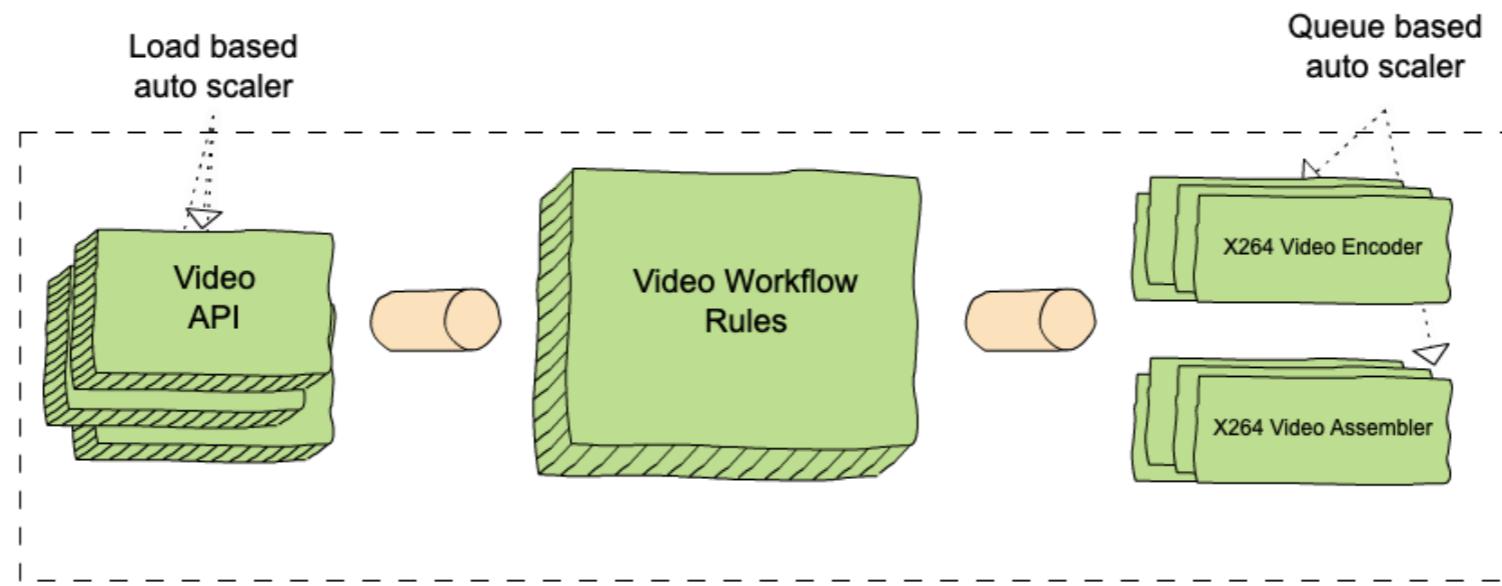
- In-game advertising
- Micro-transactions and in-game purchases
- Game-Infrastructure-as-a-Service: matchmaking, advertising, leader boards,...
- Partner network: Cross-sell game data, game SDK, game analytics,

Netflix's Timestone

- A high-throughput, low-latency priority queueing system based on kafka and redis
- Main function is to provide asynchronous communication layer to orchestrate workflows between subsystems of the following internal systems
 - Cosmos - media encoding platform
 - Conductor - general-purpose workflow orchestration engine
 - BDP Scheduler - scheduler for large-scale data pipelines
- Each message contains information regarding to project, function to be executed, deadline, priority, and metadata (for filtering)

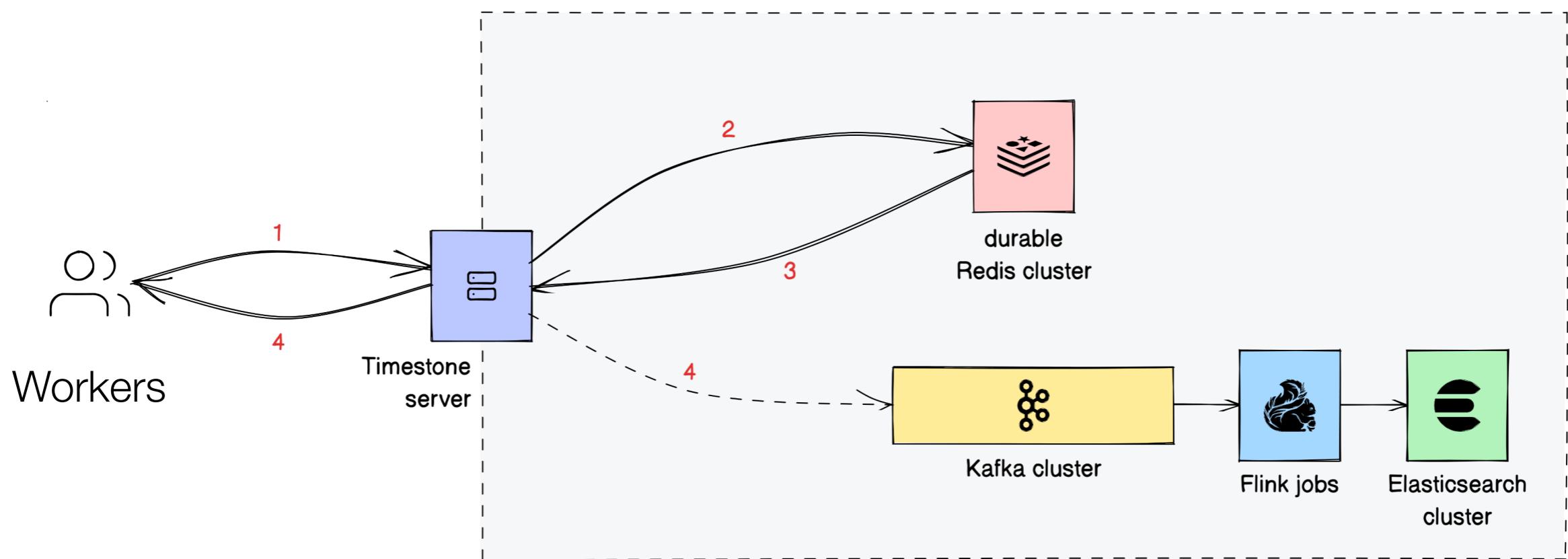
Netflix Cosmos Service

Source: “The Netflix Cosmos Platform”, <https://netflixtechblog.com/the-netflix-cosmos-platform-35c14d9351ad>



1. There is one API call to encode, which includes the video source and a recipe
2. The video is split into 31 chunks, and the 31 encoding functions run in parallel
3. The assemble function is invoked once
4. The index function is invoked once
5. The workflow is complete after 8 minutes

Timestone System Architecture



References

- J. Kreps, N. Neha, and R. Jun, "Kafka: A distributed messaging system for log processing", Proceedings of the NetDB. Vol. 11. 2011.
- E. Koutanov, "Apache Kafka in a Nutshell", <https://medium.com/swlh/apache-kafka-in-a-nutshell-5782b01d9ffb>
- M. Valcam, "Kafka: All you need to know", <https://medium.com/hacking-talent/kafka-all-you-need-to-know-8c7251b49ad0>
- P. Brebner, "The Power of Apache Kafka Partitions: How to Get the Most out of Your Kafka Cluster", <https://www.instaclustr.com/the-power-of-kafka-partitions-how-to-get-the-most-out-of-your-kafka-cluster/>
- M. Carter, "Apache Kafka Architecture: A Complete Guide", <https://www.instaclustr.com/blog/apache-kafka-architecture/>

References

- K. Christidis, "Timestone: Netflix's High-Throughput, Low-Latency Priority Queueing System with Built-in Support for Non-Parallelizable Workloads", <https://netflixtechblog.com/timestone-netflixs-high-throughput-low-latency-priority-queueing-system-with-built-in-support-1abf249ba95f>
- K. Wahner, "Apache Kafka in the Gaming Industry: Use Cases + Architectures", <https://dzone.com/articles/apache-kafka-in-the-gaming-industry-use-cases-arch>
- R. Glushach, "The Evolution of Kafka Architecture: From ZooKeeper to KRaft", <https://romanglushach.medium.com/the-evolution-of-kafka-architecture-from-zookeeper-to-kraft-f42d511ba242>