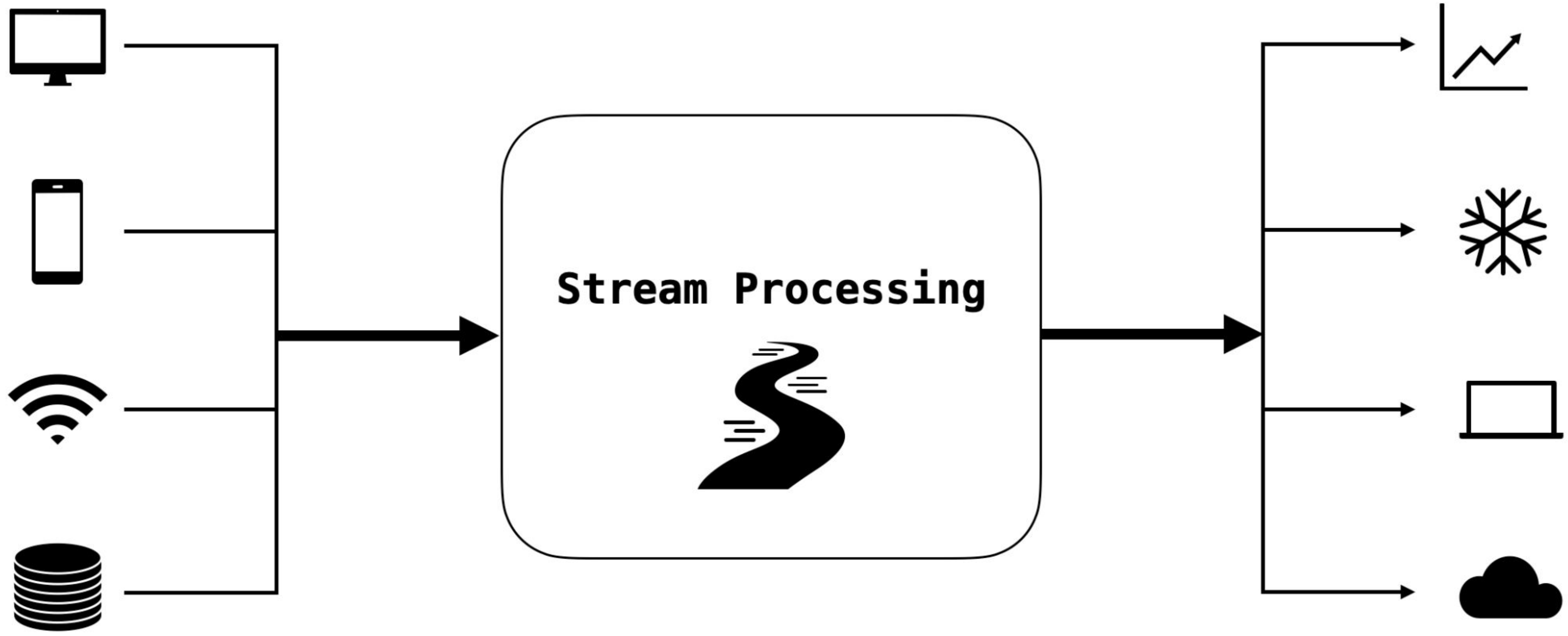
Real-time event streaming with Python

Dave Klein - Senior Developer Advocate


Stream Processing

Apache Kafka

Apache Pulsar

Red Panda

Amazon Kinesis

… and more

Apache Flink

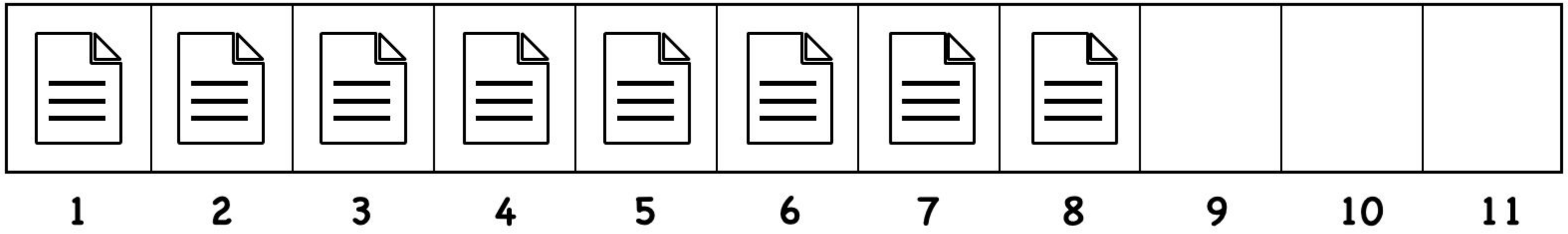
ksqlDB

Kafka Streams

… and more

Apache Kafka
A Primer

A Log (Topic)



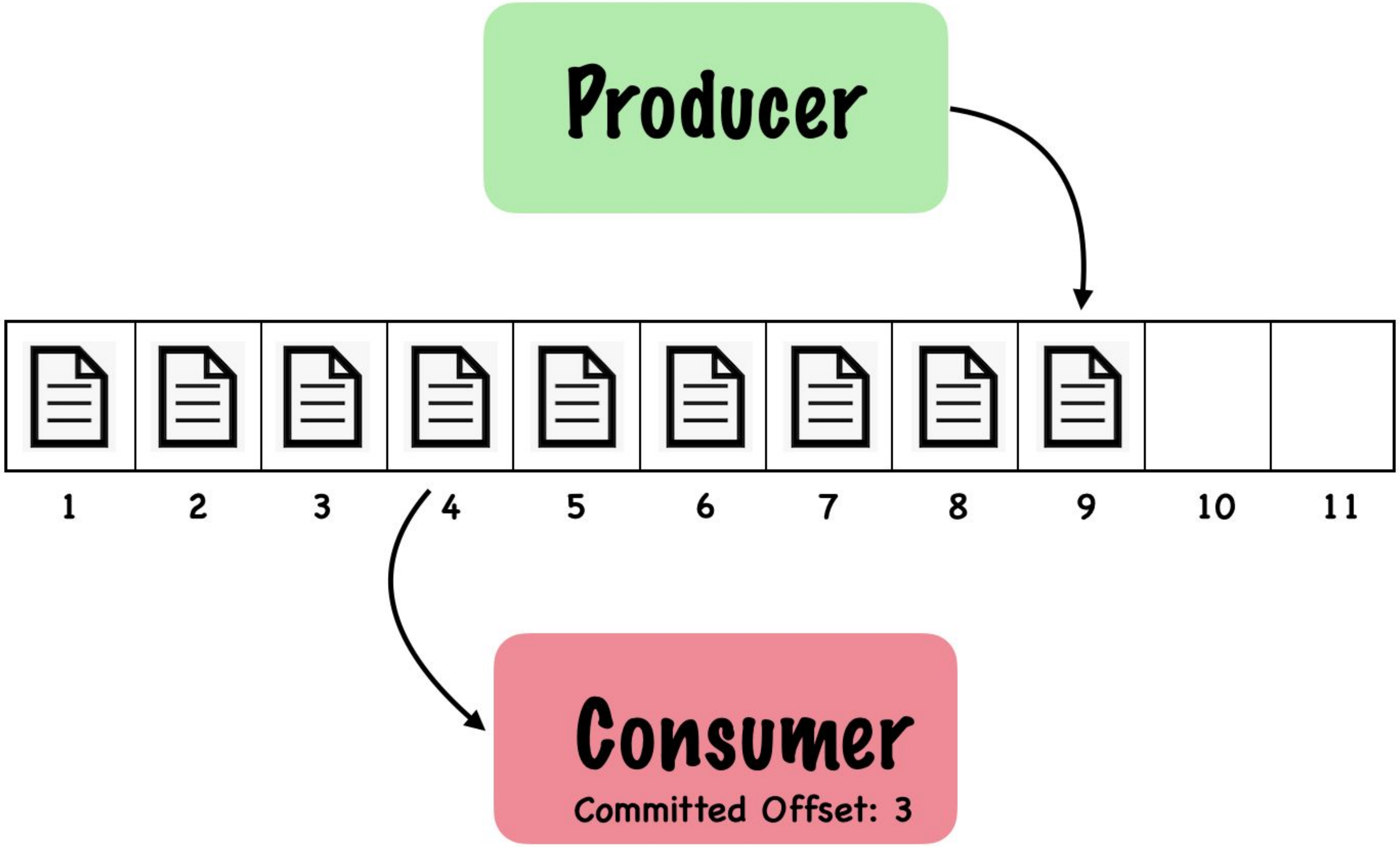
Immutable

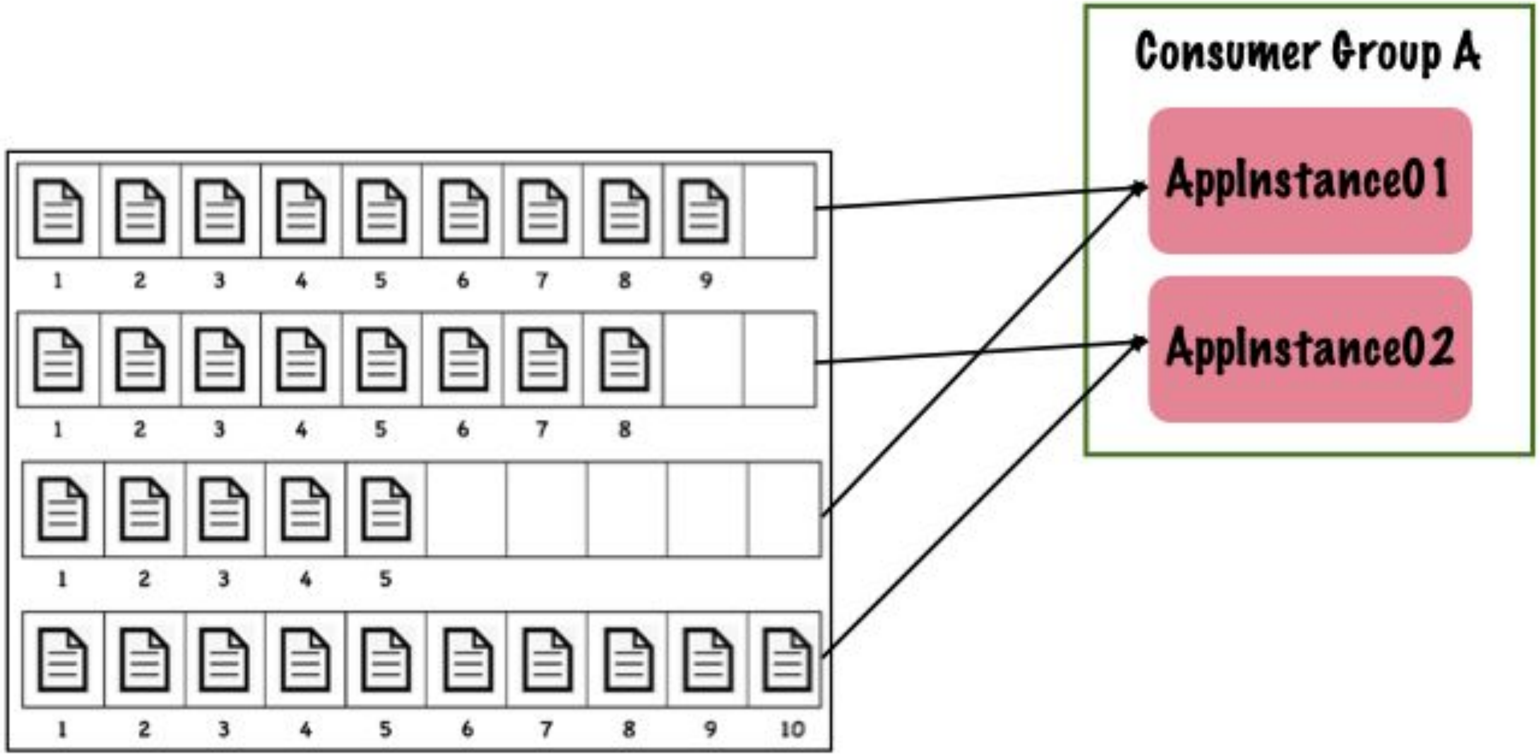
Append-only

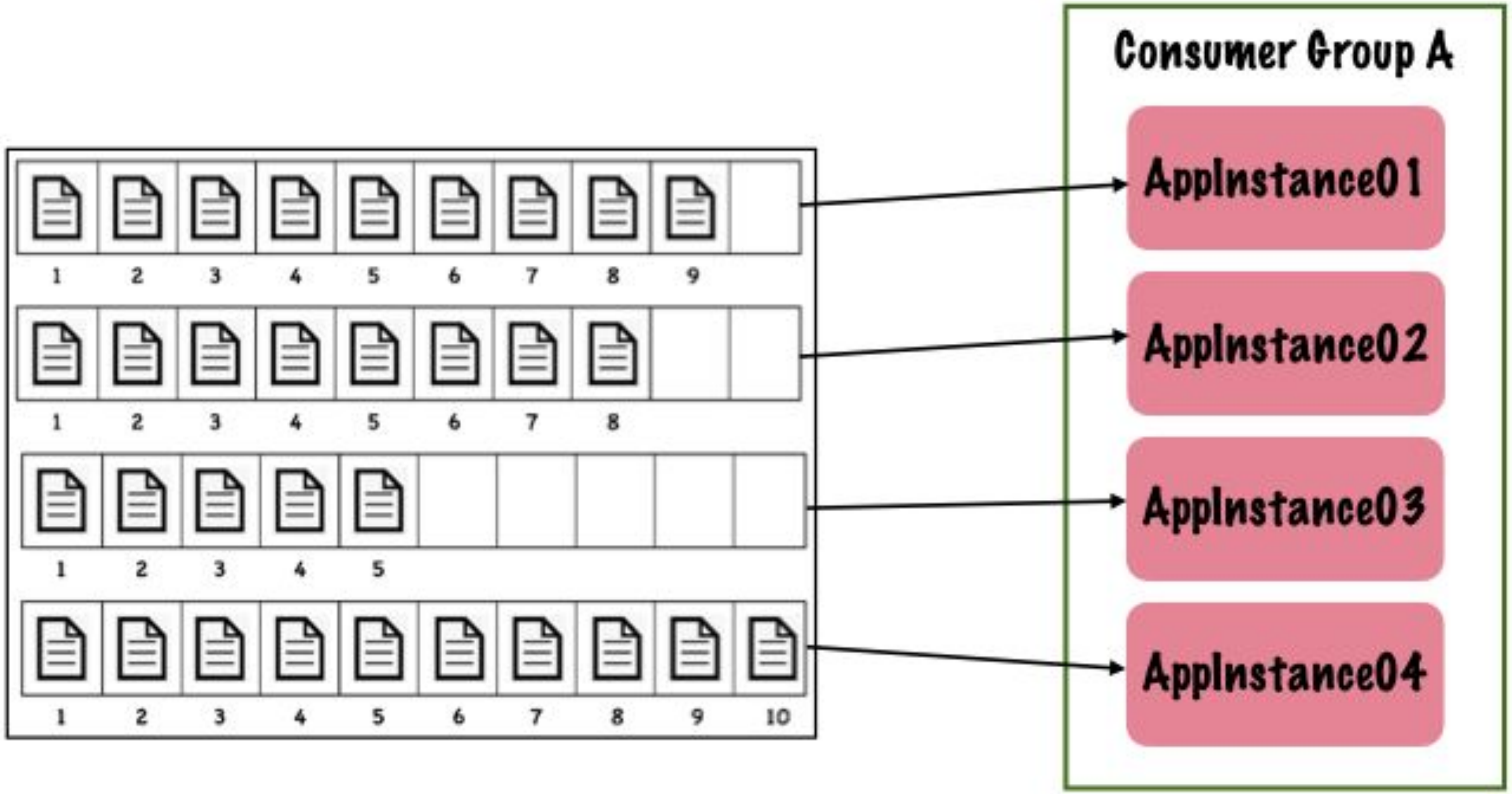
Ordered

Durable

```
"timestamp": "2023-04-27T18:30:01.223Z",
"key": "driver-0032",
"value": {
    "eventType": "delivery-update",
    "lat": 51.12321322,
    "lon": 14.132131212,
    "speed": 35,
    “deliveryId”: “pizza-my-heart-123213”
}
```


Producer
1
2
3
4
5
6
7
8
9
10
11
Consumer
Committed Offset: 3


Consumer Group A
AppInstance01
AppInstance02
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8
1 2 3 4 5
1 2 3 4 5 6 7 8 9 10


Consumer Group A
AppInstance01
AppInstance02
AppInstance03
AppInstance04
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8
1 2 3 4 5
1 2 3 4 5 6 7 8 9 10

Python Kafka Clients

- kafka-python

  https://github.com/dpkp/kafka-python

- confluent-kafka

  https://github.com/confluent-inc/confluent-kafka-python

- aiokafka

  https://github.com/aio-labs/aiokafka

Producing data to a Kafka topic

```
from confluent-kafka import Producer

producer = Producer({"bootstrap.servers": "localhost:29092"})

# some external system is feeding us delivery updates
del_update = recieve_delivery_update()

producer.produce("food-delivery", del_update, del_update["driver_id"])

producer.flush()
```

Consuming data from a Kafka topic

```
from confluent-kafka import Consumer
Import json

consumer = Consumer({"bootstrap.servers": "localhost:29092",
                     "group.id":"deliveries"})
consumer.subscribe(["food-delivery"])

while True:
    event = consumer.poll(1.0)
    if event is not None:
        val = event.value().decode("utf-8")
        del_update = json.loads(val)
        # do something with the event data
```

Stream Processing

Stream Processing - Stateless Operations

Filter

```
producer = Producer(...)
consumer = Consumer(...)

consumer.subscribe(["input_topic"])

while True:
    event = consumer.poll(1.0)
    if check_predicate(event):
        producer.produce("output_topic", event.value(), event.key())
```

Stream Processing - Stateless Operations

Map

```
producer = Producer(...)
consumer = Consumer(...)
consumer.subscribe(["input_topic"])

def transform_event(event):
    # some process we want done on each event

while True:
    event = consumer.poll(1.0)
    if event is not None:
        transformed_value = transform_event(event)
        producer.produce("output_topic", transformed_value, event.key())
```

Stream Processing - Stateful Operations

Count

```
. . .

counts = {}
while True:
    event = consumer.poll(1.0)
    if event.key() in counts:
        counts[event.key()] += 1
    else:
        counts[event.key()] = 1

    producer.produce("output", counts[event.key()], event.key())
```

Stream Processing - Stateful Operations

Sum

```
. . .

sums = {}

while True:
    event = consumer.poll(1.0)
    if event.key() in sums:
        sums[event.key()] += event.value()
    else:
        sums[event.key()] = event.value()

    producer.produce("output", sums[event.key], event.key())
```

Stream Processing - Stateful Operations

Join

```
# a_dict, b_dict, c_dict, and consumers declared above
while True:
    a_event = consumer_a.poll(1.0)
    a_dict[a_event.key()] = a_event
    if a_event.key() in b_dict and a_event.key() not in c_dict:
        c_event = join_func(a_event, b_dict[a_event.key()])
        c_dict[a_event.key()] = c_event
    b_event = consumer_b.poll(1.0)
    b_dict[b_event.key()] = b_event
    if b_event.key() in a_dict and b_event.key() not in c_dict:
        c_event = join_func(a_dict[b_event.key()], b_event)
        c_dict[b_event.key()] = c_event
    producer.produce(“output”, key=c_event.key(), value=c_dict[c_event.key()])
```

Demo

https://github.com/daveklein/top-tweeters

Top Tweeters for PyCon Italia 2023

User	Count
Saurav Jain (Open Source + Communities)	20
Marlene Mhangami	14
Ester 🇺🇦	14
🇺🇦🏳️‍🌈🐍fundor333@mastodon.social🐳👨‍💻	10
Cheuk Ting Ho at #PyCon IT	10
Danica Fine	9
Matteo Benci	9
Alessia Marcolini	7
Marcelo Trylesinski	7
Fiorella De Luca	7
Paolo Castagna	6
Valerio Maggio @leriomaggio@mastodon.social	6

Stream Processing Challenges

State

Scale

Python Streaming Clients

quixstreams

https://github.com/quixio/quix-streams

faust

https://github.com/faust-streaming/faust

do-it-yourself

Processing data with Quix Streams

```
import quixstreams as qx

client = qx.KafkaStreamingClient("localhost:29092")
consumer = client.get_topic_consumer("food-delivery", consumer_group="eta_calc")
producer = client.get_topic_producer(“food-delivery-with-eta”)

def received_handler(stream: qx.StreamConsumer, df: pd.DataFrame):
    df[“ETA”] = calc_eta(df[[“lat”, “lon”]], stream.stream_id)
    producer.timeseries.publish(df)

def on_stream_received_handler(stream_consumer: qx.StreamConsumer):
   stream_consumer.timeseries.on_data_received = received_handler

consumer.on_stream_received = on_stream_received_handler

qx.App.run()
```

Thank you

Quix

dave@quix.io | forum.quix.io