Recitation 2

Apache Kafka

What is Kafka?

- Distributed large-scale messaging system / event-streaming platform / event log
- Based on publish-subscribe pattern
- Provides the ability to store, process / reprocess streaming data in real-time
- Reliable, scalable, and high-throughput platform
- Facilitates event-driven architecture (event notifications, event sourcing, etc.)

Where is it used in the industry?

Companies:

- Twitter
- Netflix
- Uber
- LinkedIn
- PayPal
- ...

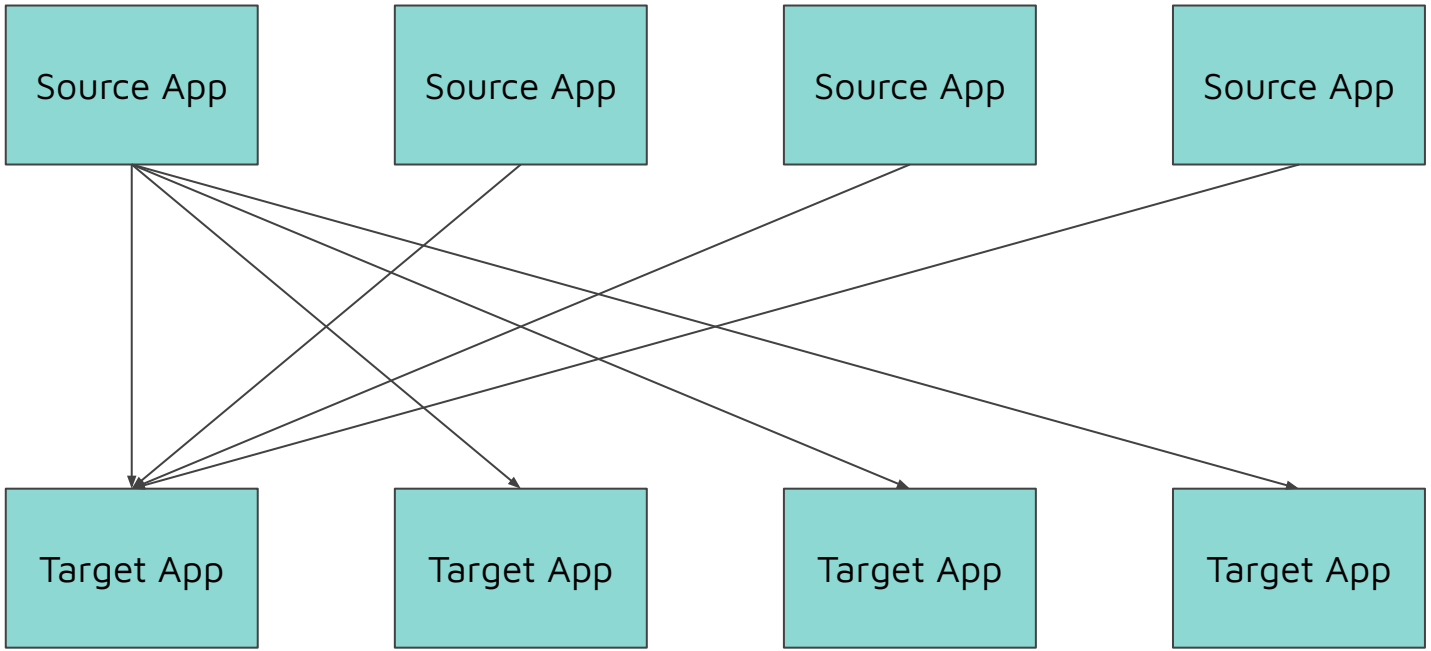
Use Cases:

- ETL (Extract Transform Load)
- CDC (Change Data Capture)
- Big data
- Metrics
- Log aggregation
- Data pipelines for ML
- Task workflows
- Activity tracking
- Stream processing
- ...

Example scenario

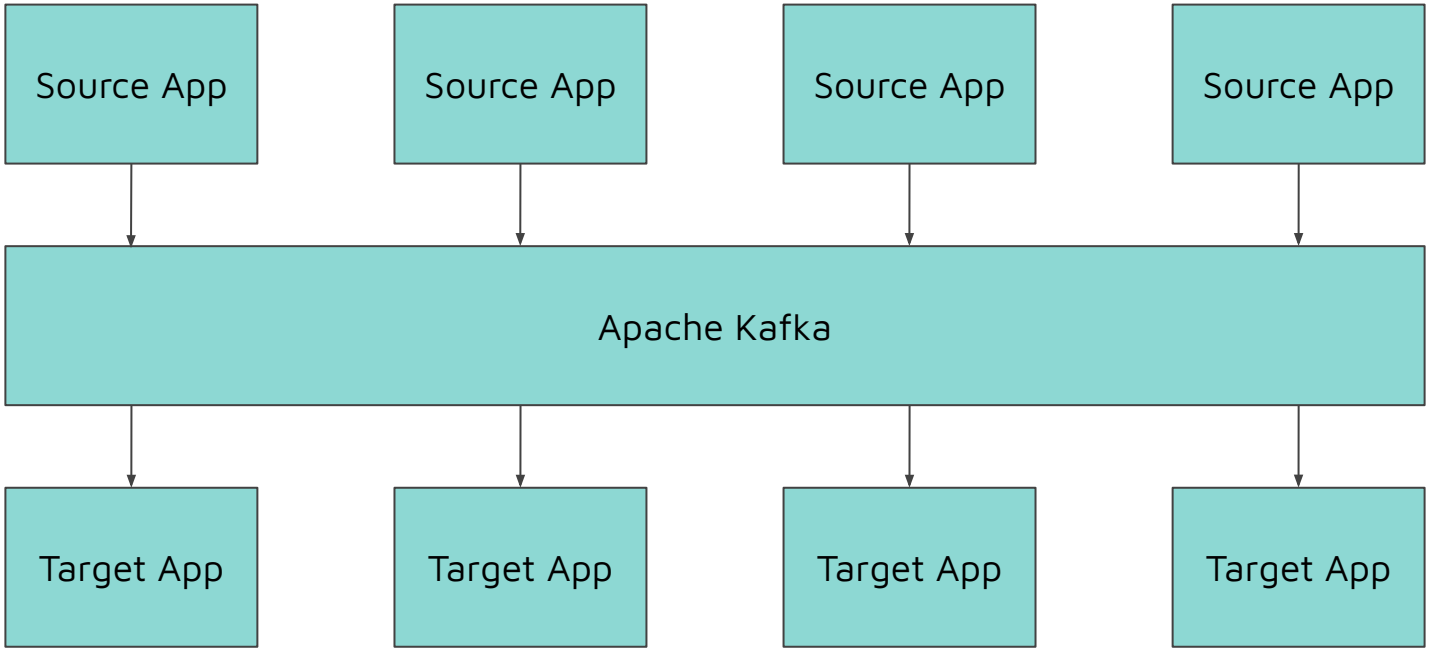
- There are hundreds of services that produce messages simultaneously
  - Sensor readings, Log messages, Messages that indicate the happening of some event, etc.
- There are hundreds of services that consume and process these messages
- There can be multiple services that consume data produced from a single service
- We should be able to quickly ramp up the services that produce/consume messages
- All messages should be processed without fail

Is this design scalable?



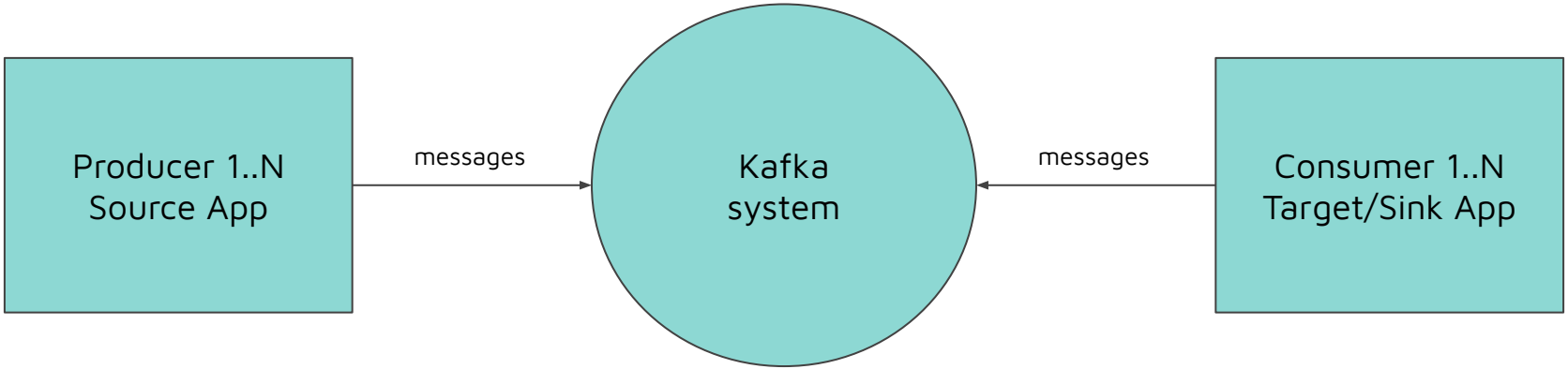
Source: https://medium.com/rock-your-data/getting-started-with-apache-kafka-efc616bd6dd5

Use an intermediary. But what's the trade-off with this design?



Source: https://medium.com/rock-your-data/getting-started-with-apache-kafka-efc616bd6dd5

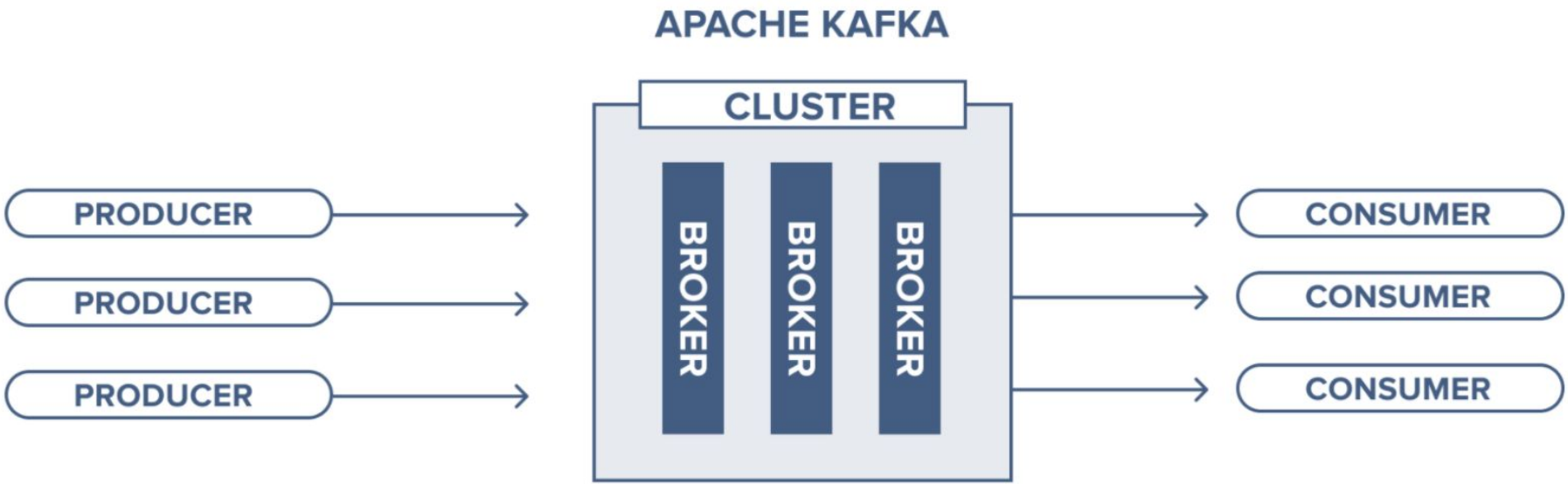
High-level Context diagram - Kafka



Parts of a Kafka system

- Producer : An application or process that produce messages or “events”
- Consumer : An application or process that consume messages or “events”
- Broker : A server running Kafka that handle requests from clients
- Zookeeper : Keeps track of the state of the Kafka cluster

Kafka cluster. Why do we need multiple brokers? Why not just one?



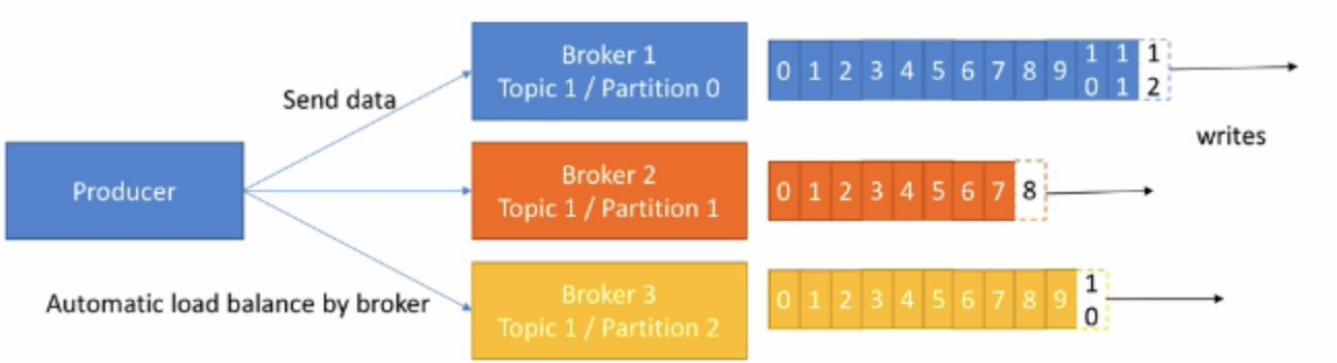
Source: https://www.cloudkarafka.com/blog/2016-11-30-part1-kafka-for-beginners-what-is-apache-kafka.html

Kafka topic

- Message - A single unit of data that can be sent or received
  - It is just a byte array
- A category (name) to which events can be produced to, and consumed from
- Producers produce to topics
- Consumers subscribe to topics
- A topic can have one or more partitions
- Each partition has its own commit log
- Each record in a partition is assigned an ID (or offset)

Source: https://www.youtube.com/watch?v=JalUUBKdcA0

Topic partitions



Kafka broker

- Responsible for:
  - Receiving messages from producers
  - Assigning offsets
  - Committing messages to the disc
  - Responds to consumer's fetch requests with messages
- One broker will act as the cluster controller
  - Responsible for assigning partitions to brokers
  - Monitoring for broker failures
- A partition is always owned by a single broker in a cluster (leader of the partition)
- A partition may be assigned to multiple brokers (partition will be replicated; provides redundancy of messages in the partition)

Source: https://www.youtube.com/watch?v=JalUUBKdcA0

Consumers

- Can subscribe to one or more topics
- Consumers keeps track of its position in the data stream
- By storing the offsets in zookeeper or in Kafka itself, the consumer can stop/restart without losing its position in the data stream
- Consumers within a consumer group work together to consume a topic
    - Group makes sure that each partition is consumed only by one member of the group
    - Consumers can scale horizontally to consume topics with a large number of messages
    - To consume the same messages multiple times, we need different consumer groups

Source: https://www.youtube.com/watch?v=JalUUBKdcA0

Design considerations

- Is it acceptable to lose messages?
- Should all messages be processed at least once?
- Should it be possible to process old messages again?
  - Should messages be available even after they've been processed?
- Is the order in which messages are processed important?
- Is the system under consideration a real-time system (low latency)?
- Do you have a need to buffer messages?
- Do the messages contain state, or are they just notifications?
- Consider scalability, modifiability, availability, etc.

Source: Summer 2020 - SE4AI Lecture - Data @ Scale

Kafka setup

- Open a new terminal
- Run “ssh -L 9092:localhost:9092 tunnel@128.2.204.215 -NT”
  - This command forwards the port 9092 of the course server to your localhost’s port 9092
  - Use the password given in Canvas
- Alternatively, use the SSH key found here and run “ssh -L 9092:localhost:9092 tunnel@128.2.204.215 -NT -i id_rsa
  - You need to have “400” permission for the file (chmod 400)
- Keep this running in the background
- Install kafkacat (CLI tool for Kafka)
  - “brew install kafkacat” OR “sudo apt-get install kafkacat”
- Test your connection
  - “kafkacat -b localhost -L”

Expected output

```
VaithyathansMBP:~ vaithya$ kafkacat -b localhost -L
Metadata for all topics (from broker 1: localhost:9092/1):
 1 brokers:
  broker 1 at localhost:9092 (controller)
 16 topics:
  topic "lol" with 1 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
  topic "movielog2" with 2 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
    partition 1, leader 1, replicas: 1, isrs: 1
  topic "movielog" with 1 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
  topic "movielog-2" with 1 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
  topic "movielog1" with 2 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
    partition 1, leader 1, replicas: 1, isrs: 1
  topic "__consumer_offsets" with 50 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
    partition 1, leader 1, replicas: 1, isrs: 1
```

Create a project to practice

- Create a directory and navigate to it, say “recitation-2”
- Install pip
  - (sudo) pip install virtualenv
- Set up your virtualenv
  - (python -m) virtualenv -p python3 venv
- Activate the virtualenv
  - source venv/bin/activate
- Install kafka library for python
  - (sudo) pip install kafka-python

Writing to Kafka (code)

producer.py

```
1  from time import sleep
2  from json import dumps
3  from kafka import KafkaProducer
4
5  # Create a producer to write data to kafka
6  producer = KafkaProducer(bootstrap_servers=['localhost:9092'],
7                           value_serializer=lambda x: dumps(x).encode('utf-8'))
8
9  # Write data via the producer
10 for e in range(10):
11     data = {'number' : e}
12     producer.send(topic='numtest', value=data)
13     sleep(1)
```

Expected output

```
VaithyathansMBP:~ vaithya$ kafkacat -b localhost -t recitation2_vaithyan
% Auto-selecting Consumer mode (use -P or -C to override)
{"number": 0}
{"number": 1}
{"number": 2}
{"number": 3}
{"number": 4}
{"number": 5}
{"number": 6}
{"number": 7}
{"number": 8}
{"number": 9}
% Reached end of topic recitation2_vaithyan [0] at offset 10
```

Reading from Kafka (code)

```
from kafka import KafkaConsumer
from json import loads

# Create a consumer to read data from kafka
consumer = KafkaConsumer(
    'numtest-<andrewid>',
    bootstrap_servers=['localhost:9092'],
    # Read from the start of the topic; Default is latest
    auto_offset_reset='earliest'
)

# Prints all messages, again and again!
for message in consumer:
    # Default message.value type is bytes!
    print(loads(message.value))
```

Expected output

```
(venv) VaithyathansMBP:~ vaithya$ python3 consumer.py
 {'number': 0}
 {'number': 1}
 {'number': 2}
 {'number': 3}
 {'number': 4}
 {'number': 5}
 {'number': 6}
 {'number': 7}
 {'number': 8}
 {'number': 9}
```

Make reads fault-tolerant

```
17  consumer = KafkaConsumer(
18      'numtest',
19      bootstrap_servers=['localhost:9092'],
20      auto_offset_reset='earliest',
21      # Consumer group id
22      group_id='numtest-group-<andrew_id>',
23      # Commit that an offset has been read
24      enable_auto_commit=True,
25      # How often to tell Kafka, an offset has been read
26      auto_commit_interval_ms=1000
27  )
28
29  # Prints messages once, then only new ones!
30  for message in consumer:
31      print(loads(message.value))
```

Expected output

```
(venv) VaithyathansMBP:~ vaithya$ python3 consumer.py
{'number': 0}
{'number': 1}
{'number': 2}
{'number': 3}
{'number': 4}
{'number': 5}
{'number': 6}
{'number': 7}
{'number': 8}
{'number': 9}
```

```
(venv) VaithyathansMBP:~ vaithya$ python3 consumer.py
```

Try this!

Run producer.py in one tab, and consumer.py in another

What is the output?

Resources

- https://kafka-python.readthedocs.io/en/master/
- https://github.com/edenhill/kafkacat
- https://www.youtube.com/watch?v=JalUUBKdcA0
- https://towardsdatascience.com/kafka-python-explained-in-10-lines-of-code-800e3e07dad1

Thank you!