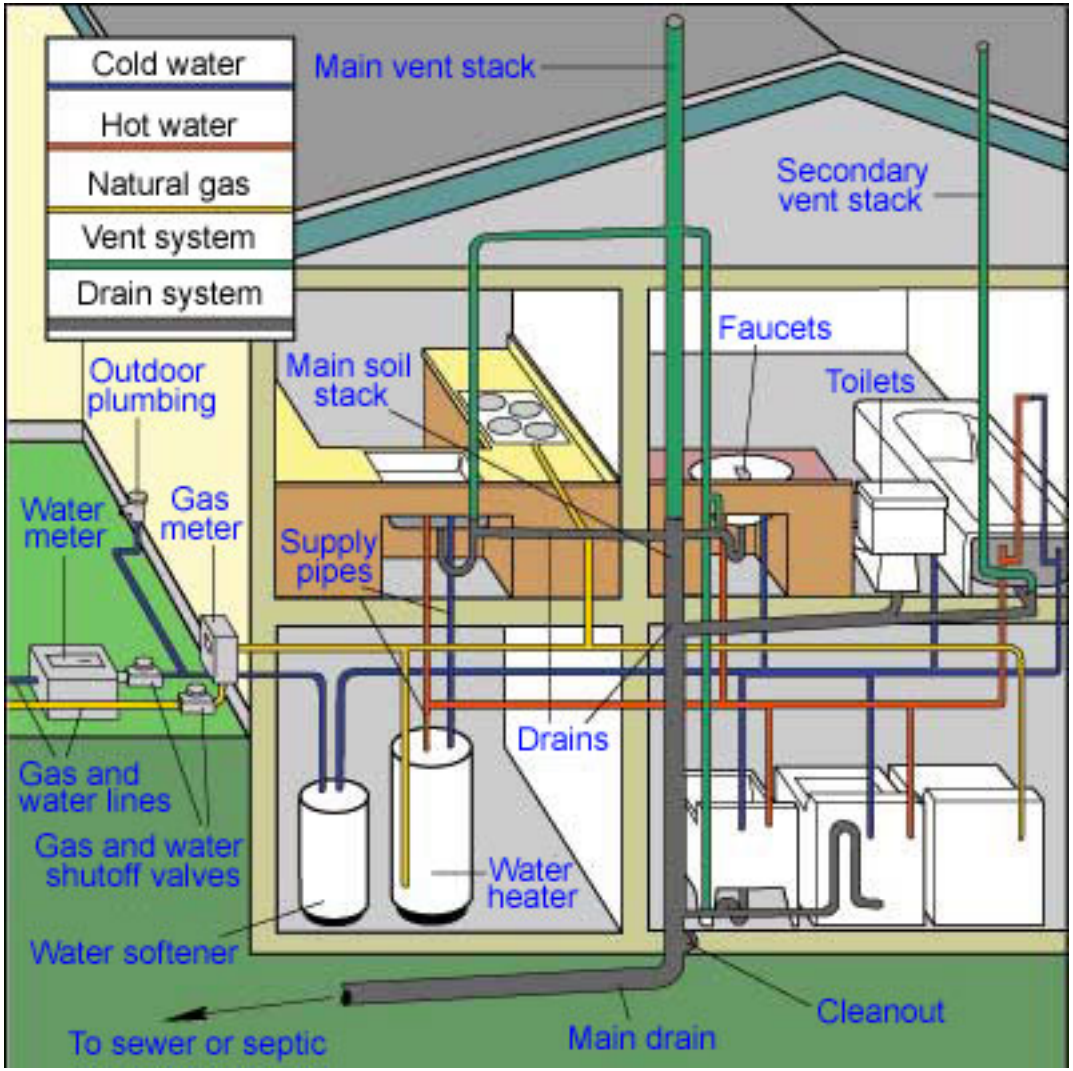
Kafka Overview

Hien Luu

Agenda

- Kafka Overview
- Kafka Architecture
- Kafka APIs

Kafka Overview



Kafka Overview

A unified platform for handling all the real-time data feeds a large company might have

Kafka Overview

A distributed messaging system for log processing that combines the benefits of traditional log aggregators and messaging systems

Kafka is a distributed, partitioned, replicated commit log service. It provides the functionality of a messaging system, but with a unique design

Kafka Overview

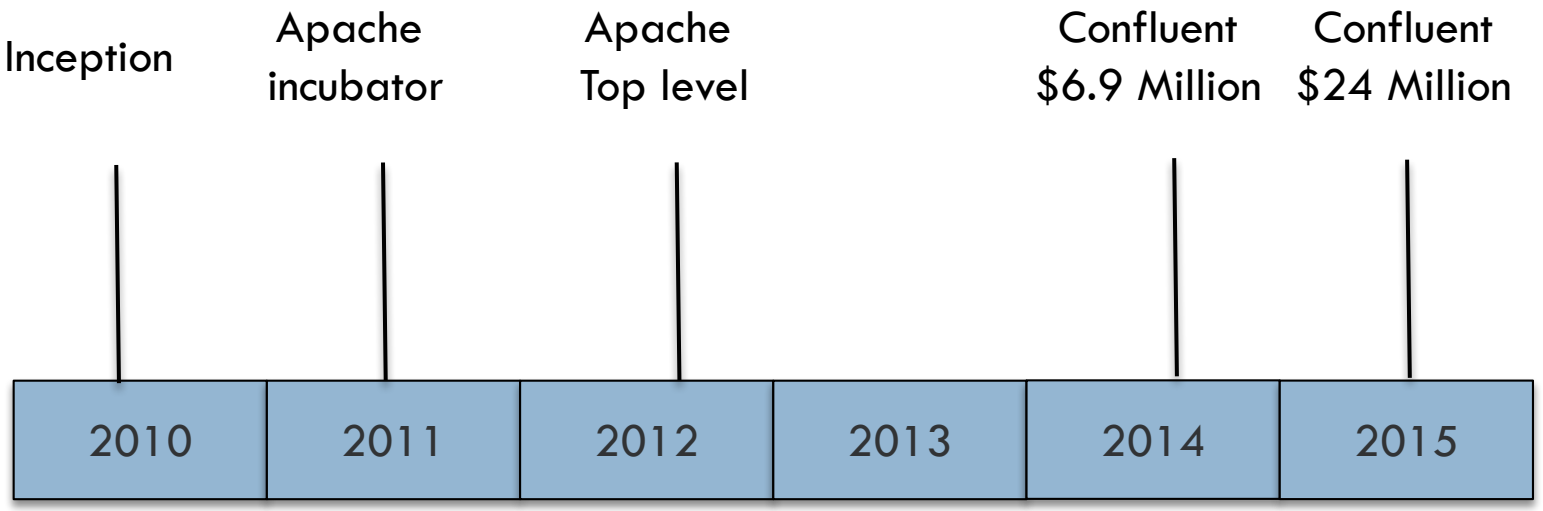
If data is the lifeblood of high technology, Apache Kafka is the circulatory system in use at LinkedIn

-Todd Palino

Kafka continues to be one of the key pillars in LinkedIn's data infrastructure

-Karthik Paramasivam

Kafka Overview



Kafka Overview

2,000,000,000,000 messages/day

(4 comma club)

1.5 PB/week

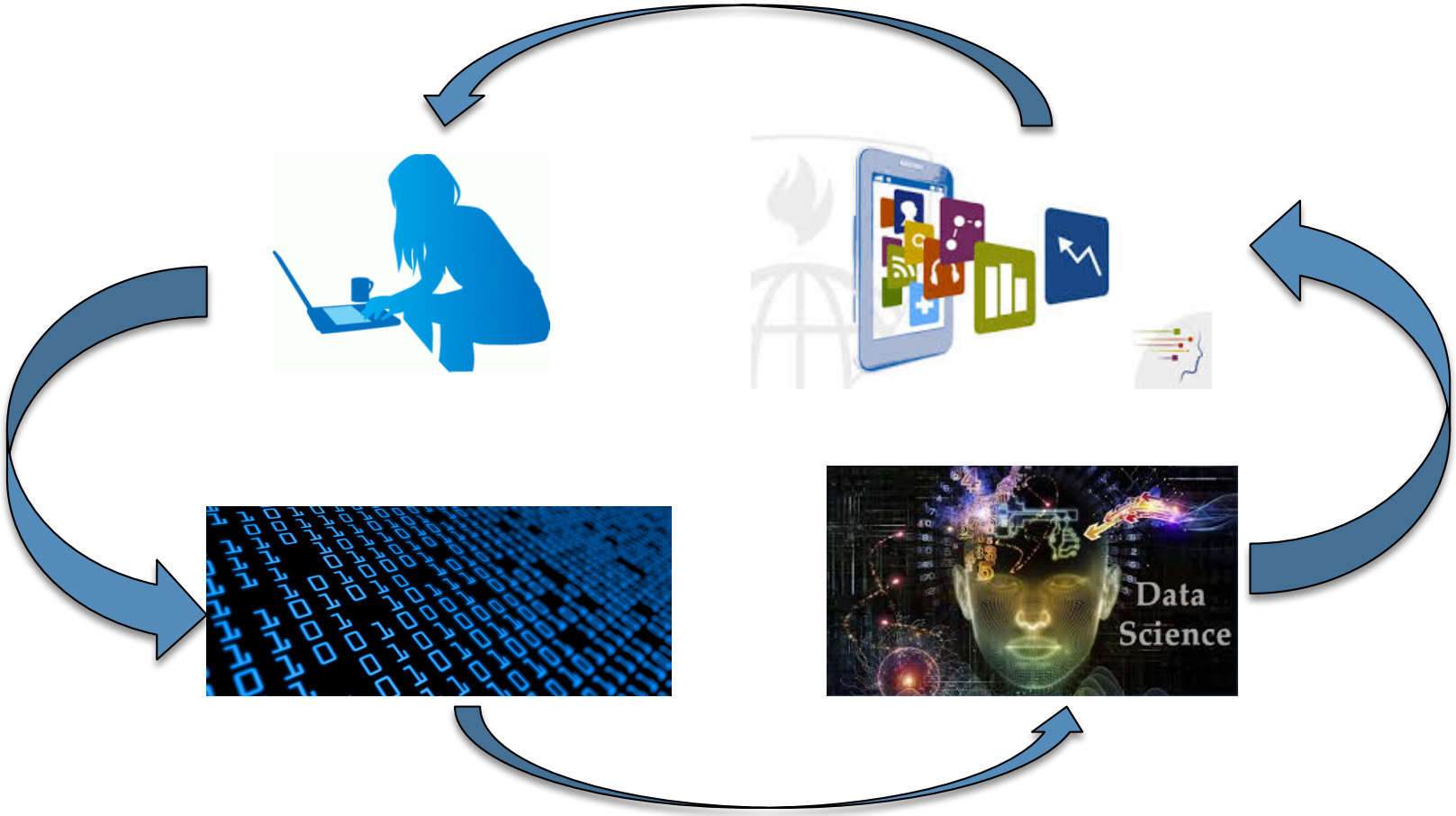
1800 servers

https://engineering.linkedin.com/apache-kafka/how-we_re-improving-and-advancing-kafka-linkedin

Kafka Overview

Kafka Overview

Data Patterns in Data-Driven Companies



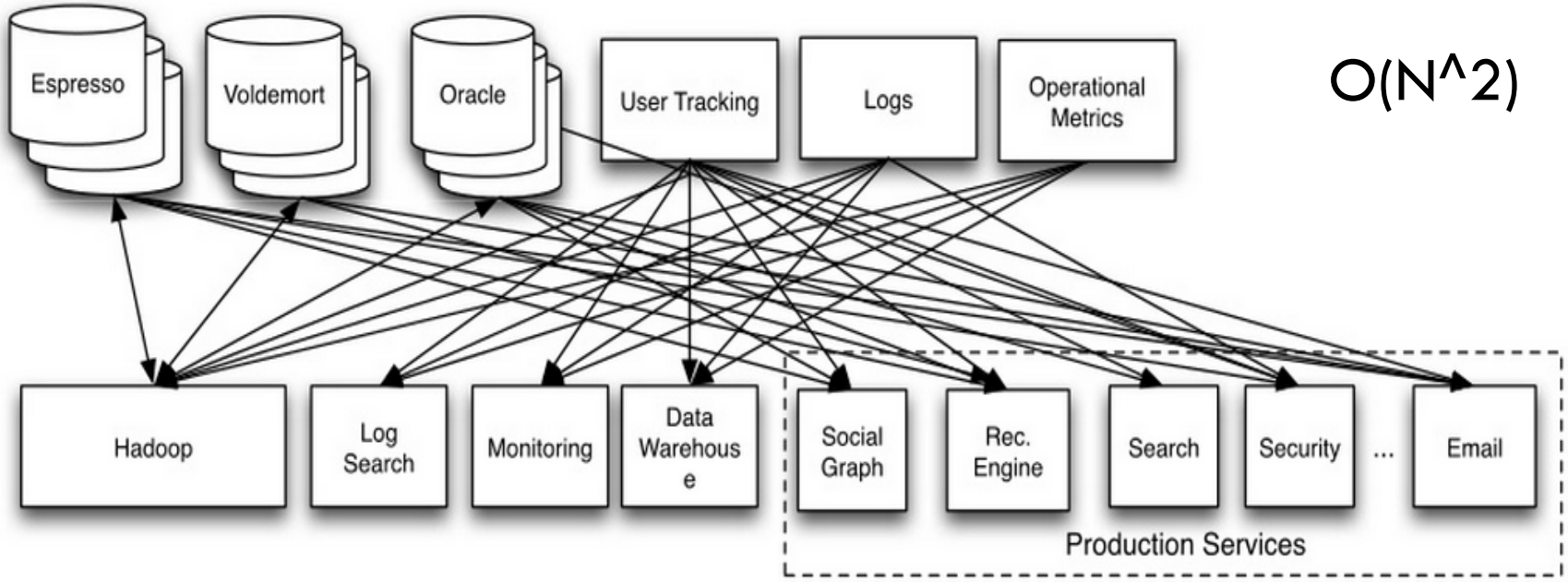
Kafka Overview

Data Sources

- Database data
  - User, products, orders
- Events
  - Clicks, impressions, pageviews, search queries
- Application metrics
  - CPU usage, request/second
- Application logs
  - Service call, errors

Kafka Overview

Early Days At LinkedIn

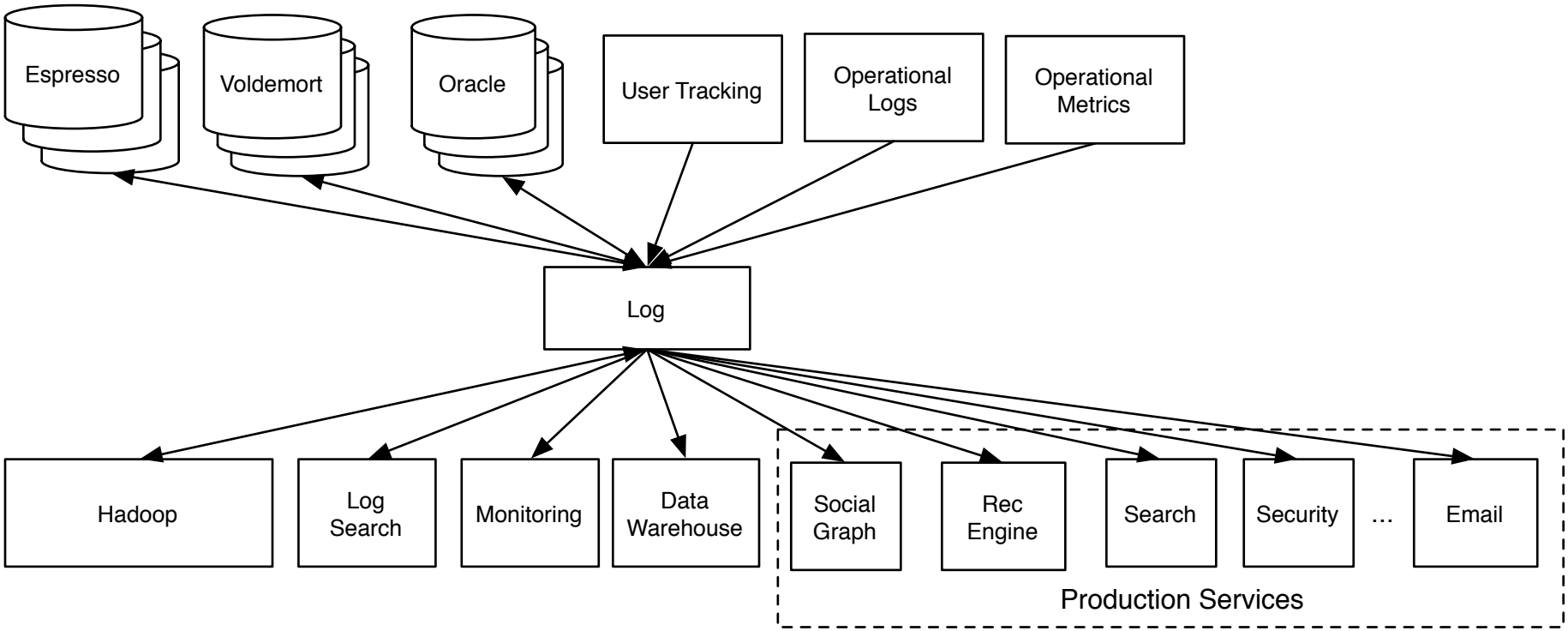


O(N^2)

https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying

Kafka Overview

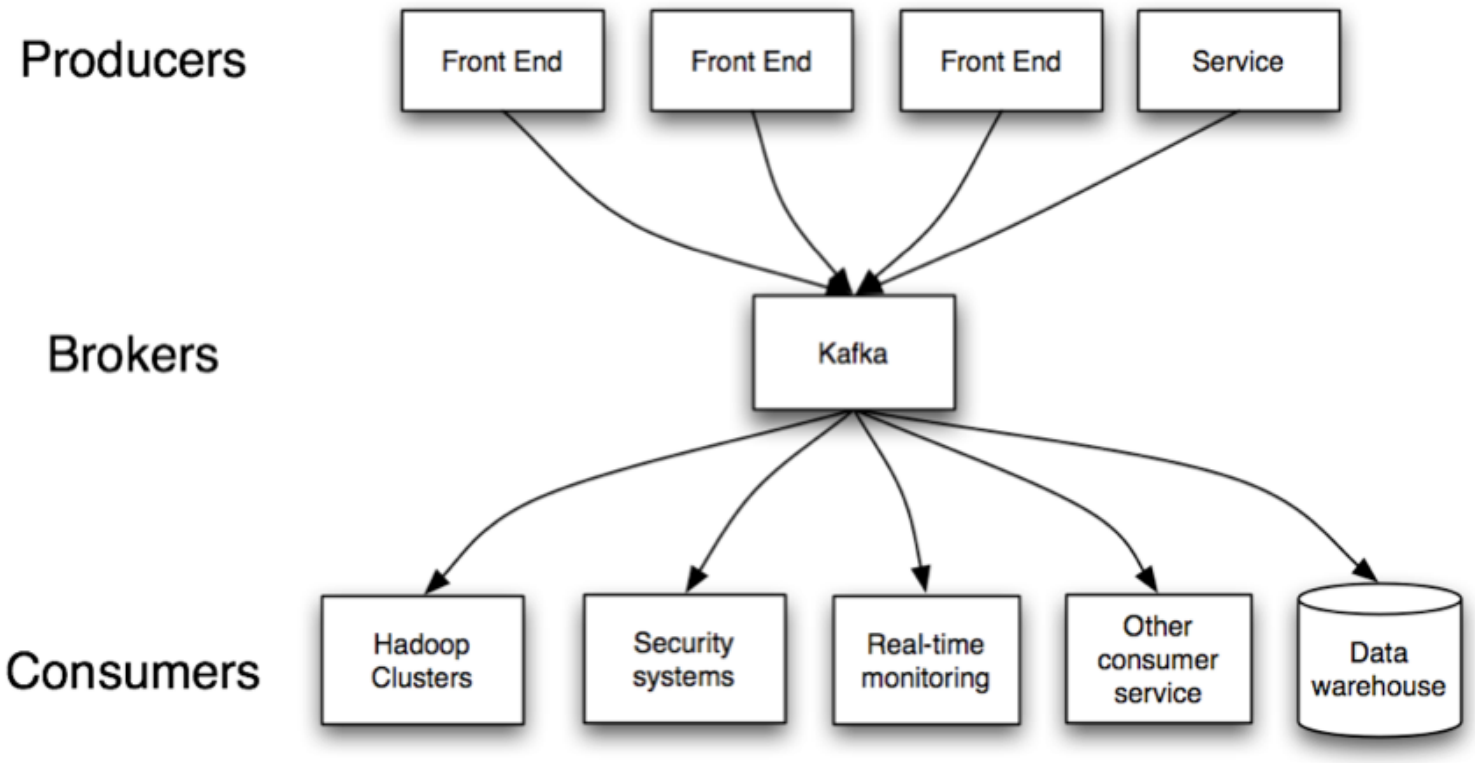
Later Days At LinkedIn



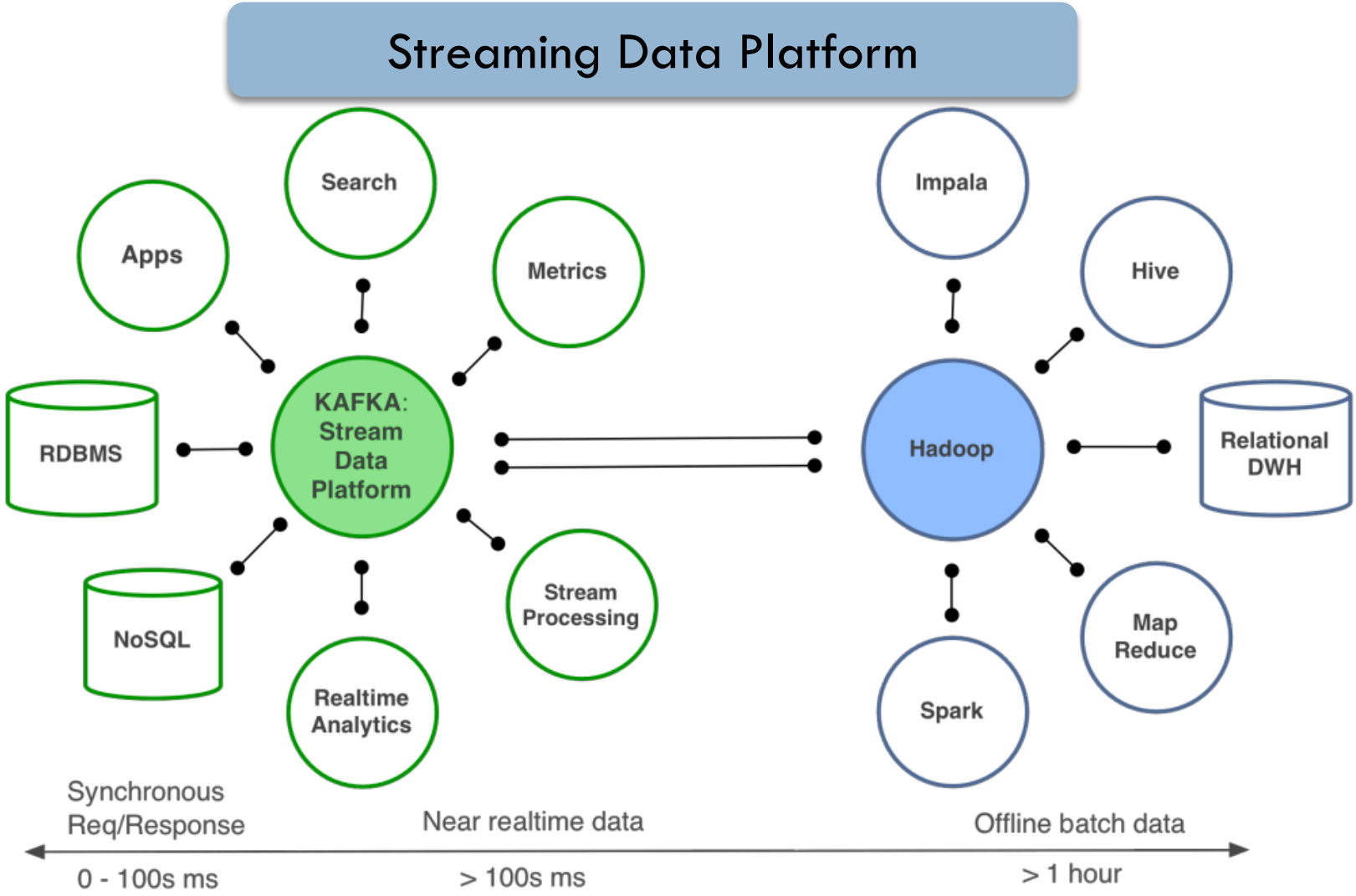
https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying

Kafka Overview

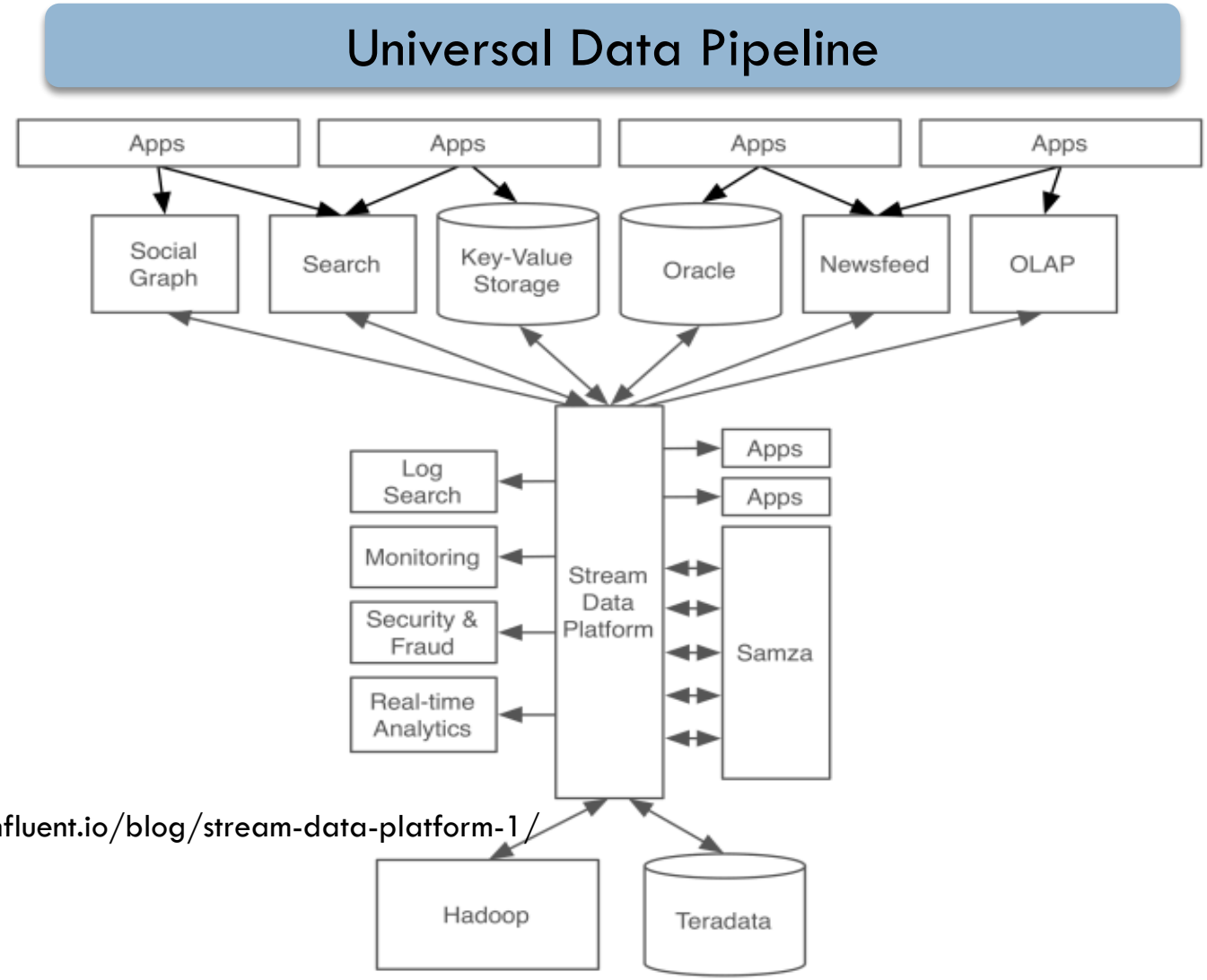
De-couples Data Pipelines



Kafka Overview



Kafka Overview



http://www.confluent.io/blog/stream-data-platform-1/

Kafka Overview

AWS 10/2015

Amazon Kinesis

Amazon Kinesis services make it easy to work with real-time streaming data in the AWS cloud.

A fire hose of streaming data, deliver continuously into AWS

Kafka Architecture

Desired Properties

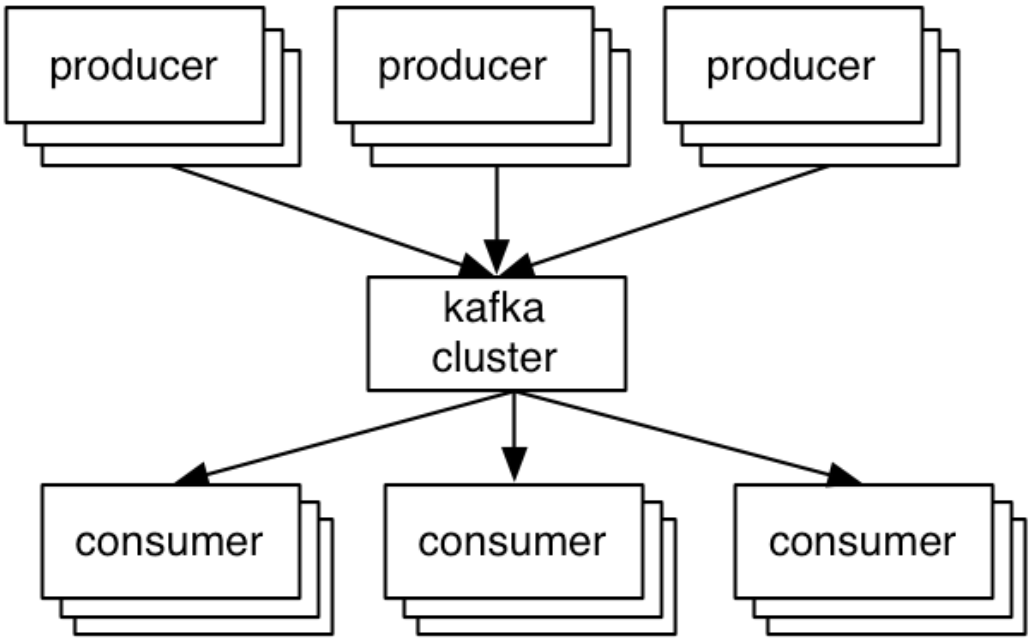
- Scalability
  - Partitioning
- Guarantees of a database
  - Order and persistent
- High throughput, high volume
- Low latency
- Fault tolerant

Kafka Architecture

- Open source messaging systems
  - ActiveMQ
  - RabbitMQ
- Challenges
  - Low throughput
    - Due to strong delivery guarantees
    - No batching support
  - Weak in distributed support
  - No good persistency model
    - Near immediate message consumption model
  - Push model

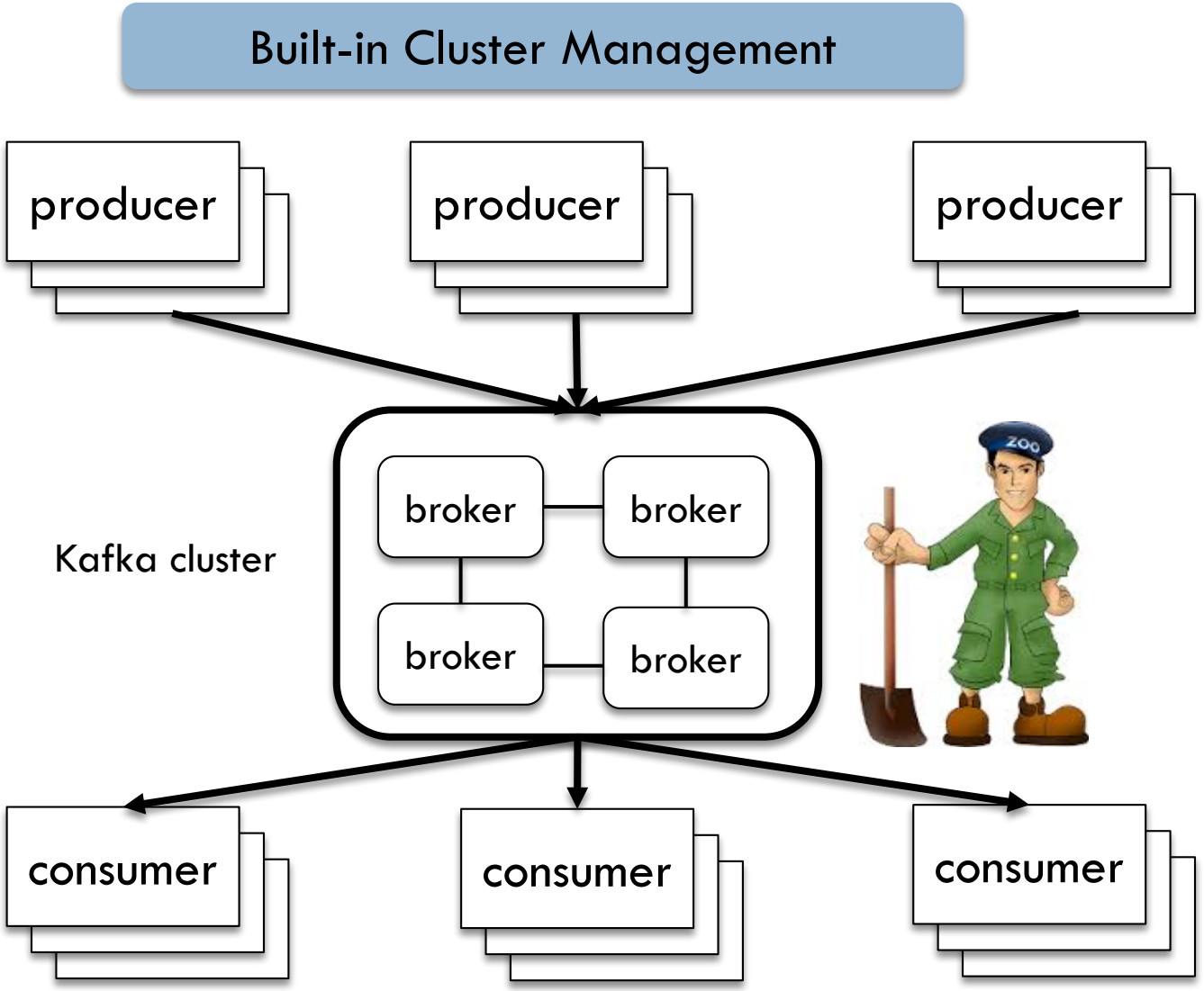
Kafka Overview

Distributed Publish-Subscribe Messaging System

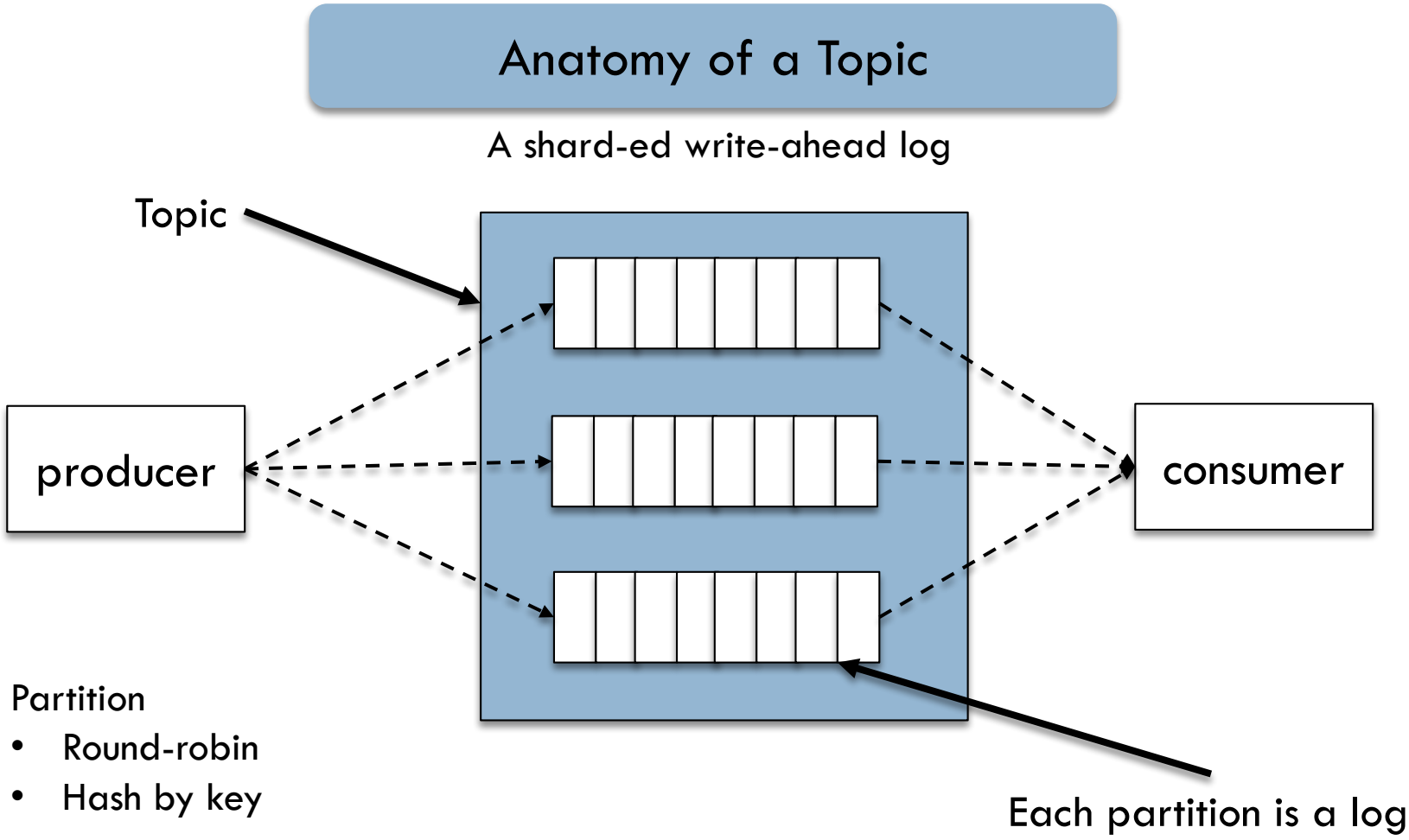


- Fast
- Durable
- Scalable

Kafka Architecture

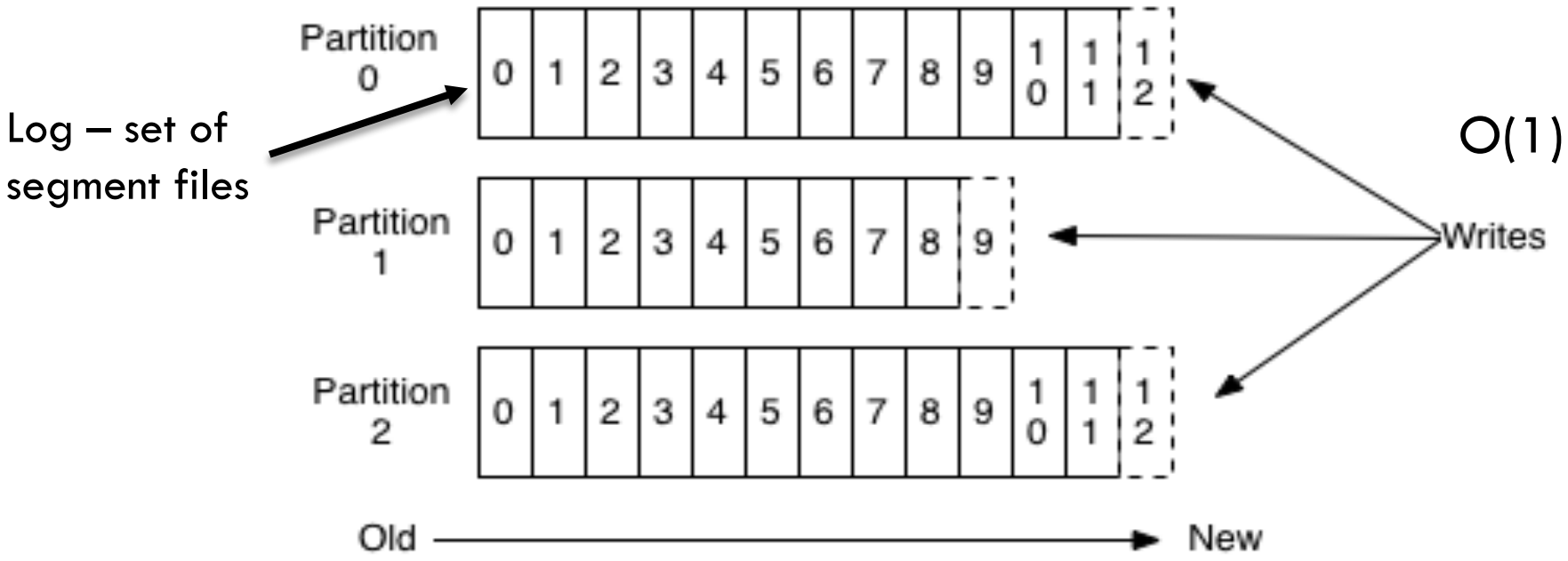


Kafka Architecture



Kafka Architecture

Partition: ordered + immutable sequence of messages



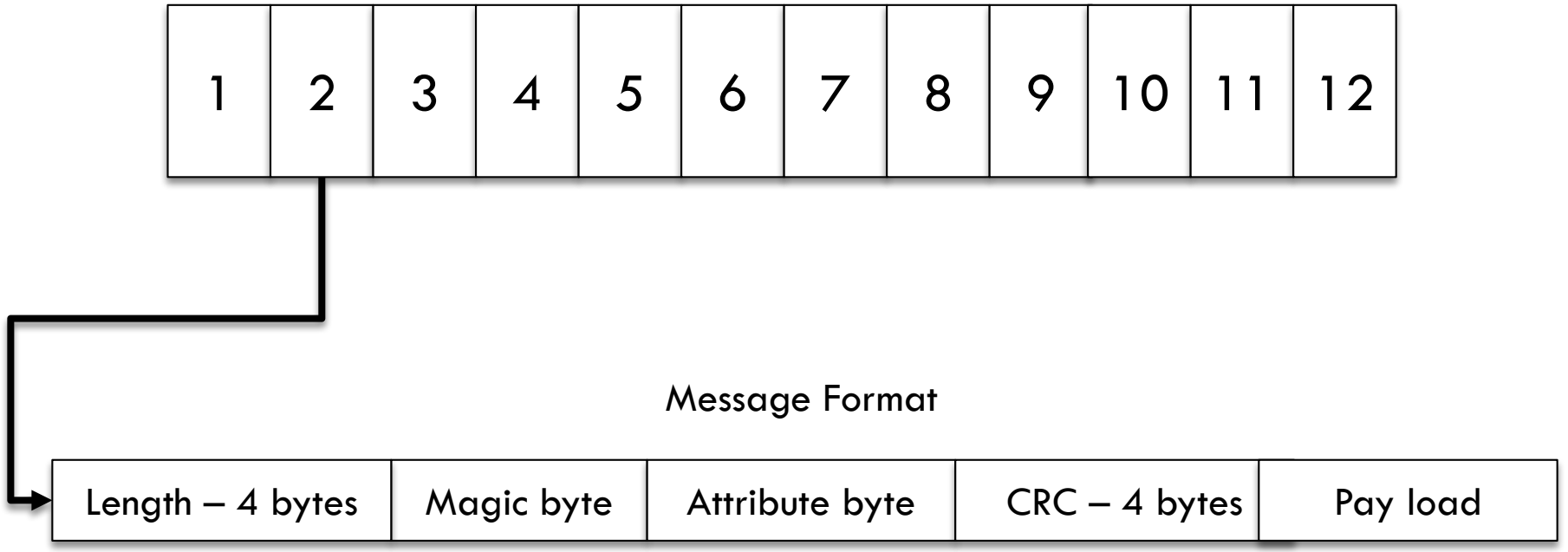
Partition is a unit of parallelism for throughput

Kafka performance is effectively constant with respect to data size.

http://kafka.apache.org/documentation.html

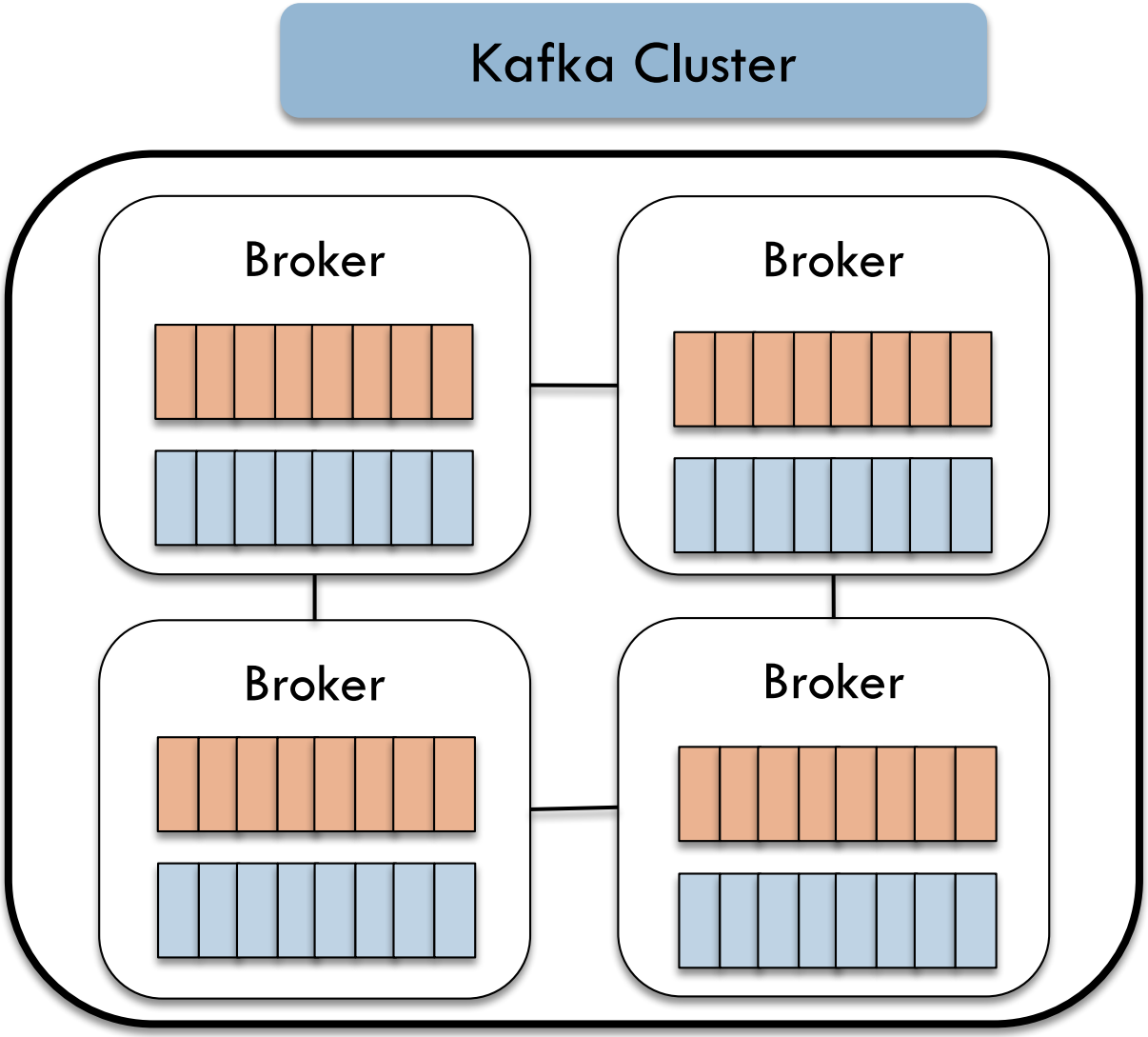
Kafka Architecture

Message Format



Each message is addressed by its logical offset in the log

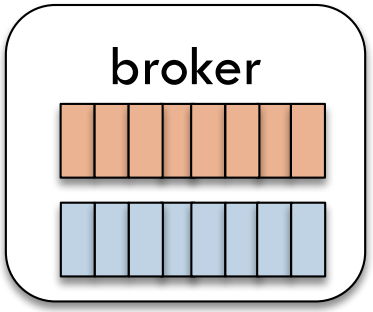
Kafka Architecture



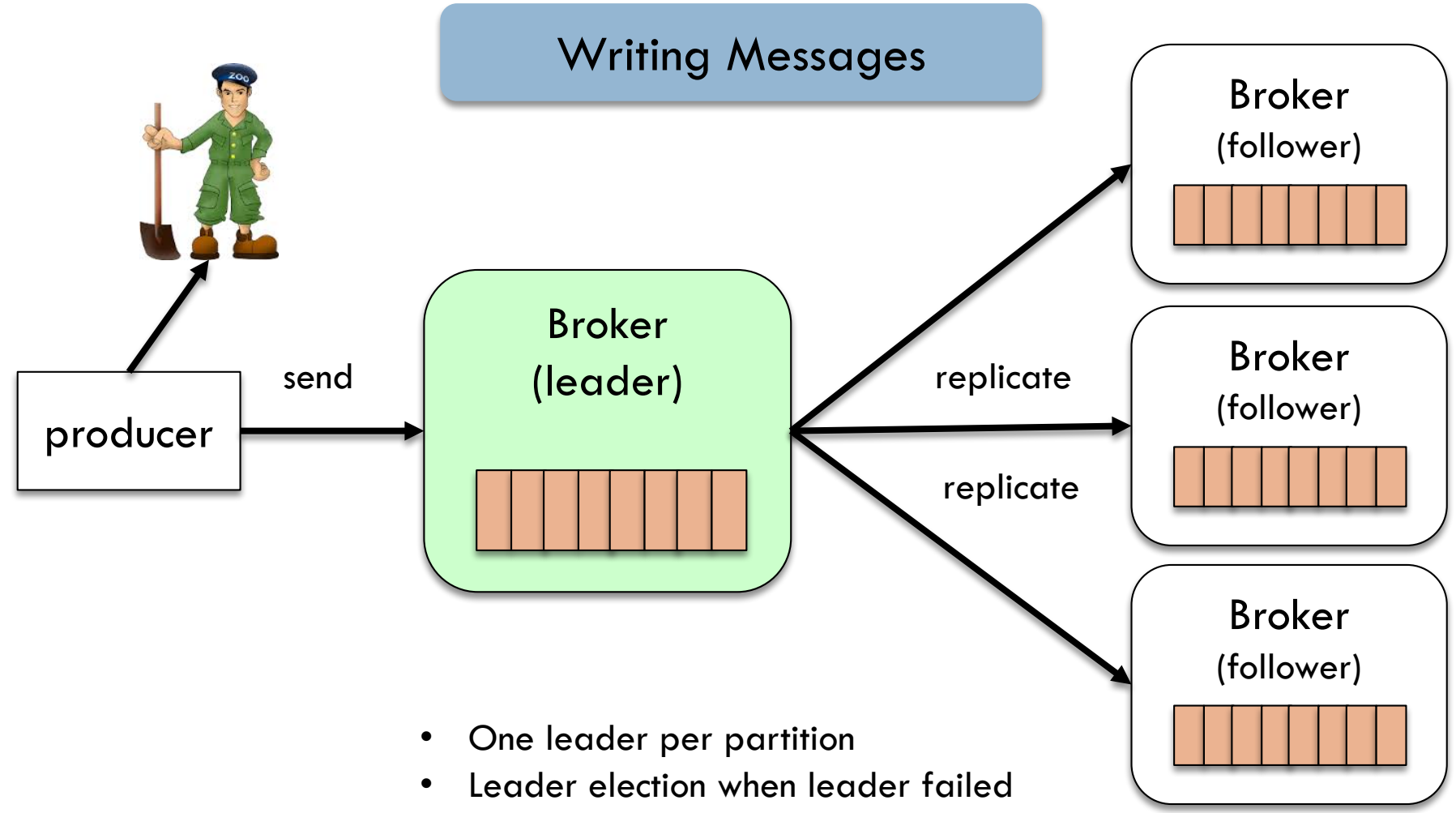
- Two topics
- Four replicas/topic

Kafka Architecture

- Partition
  - Maintain by a single broker
  - Replicate for fault-tolerance
  - SLA
    - Time based
    - Size
- Broker
  - Stateless
  - Responsible for 1 or more partitions
  - Handle replication
  - Leader for some partitions
  - Follower for some partitions

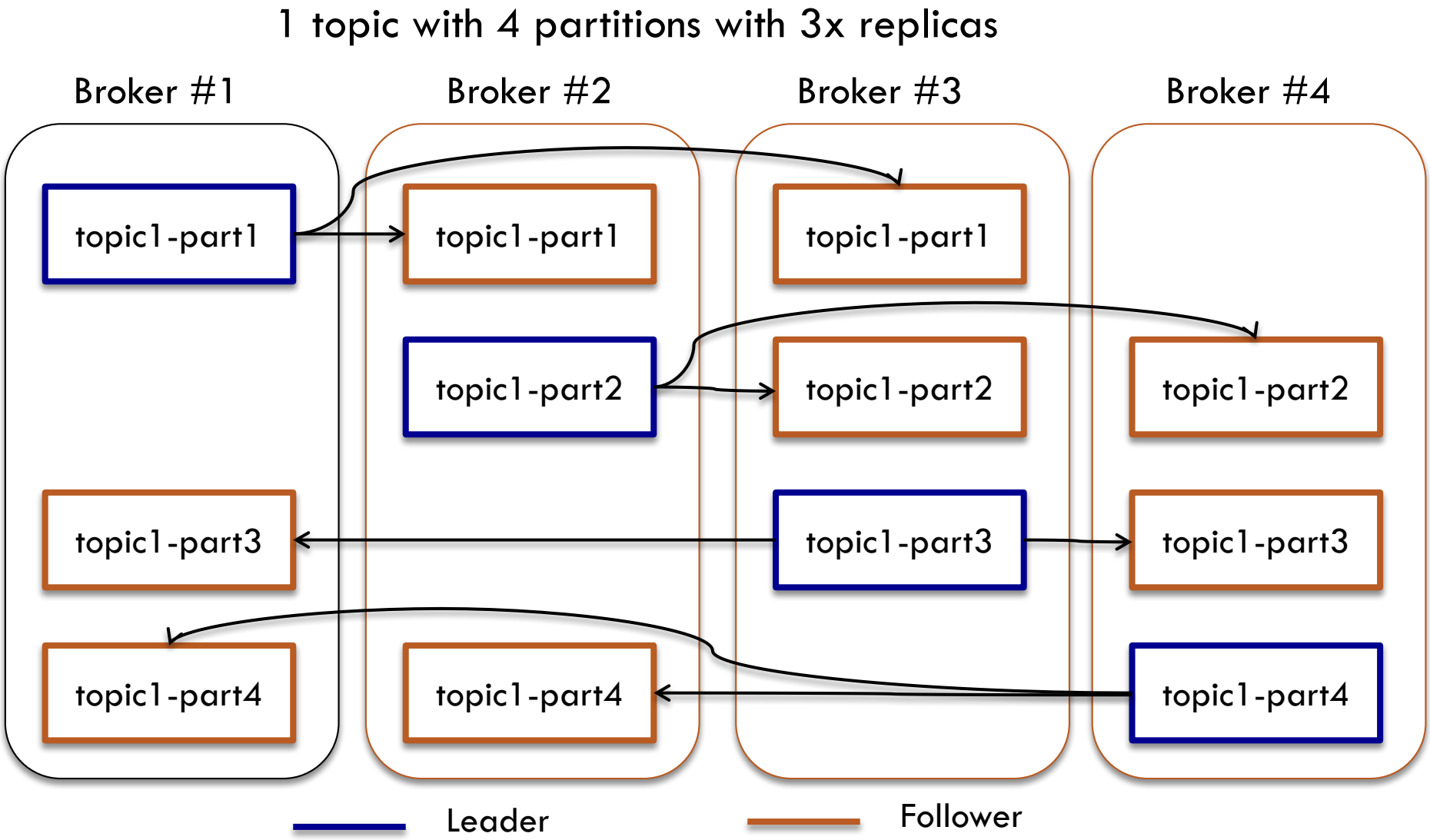


Kafka Architecture



- One leader per partition
- Leader election when leader failed

Kafka Architecture



Kafka Architecture

- Partition leader responsibility
  - Handle all reads and writes to a partition
  - A message is committed only after successfully replicated
  - Track "in-sync" replica list
  - Detect slow replicas and remove them from list
- Partition follower responsibility
  - Pull messages from the leader
  - Apply messages to its log

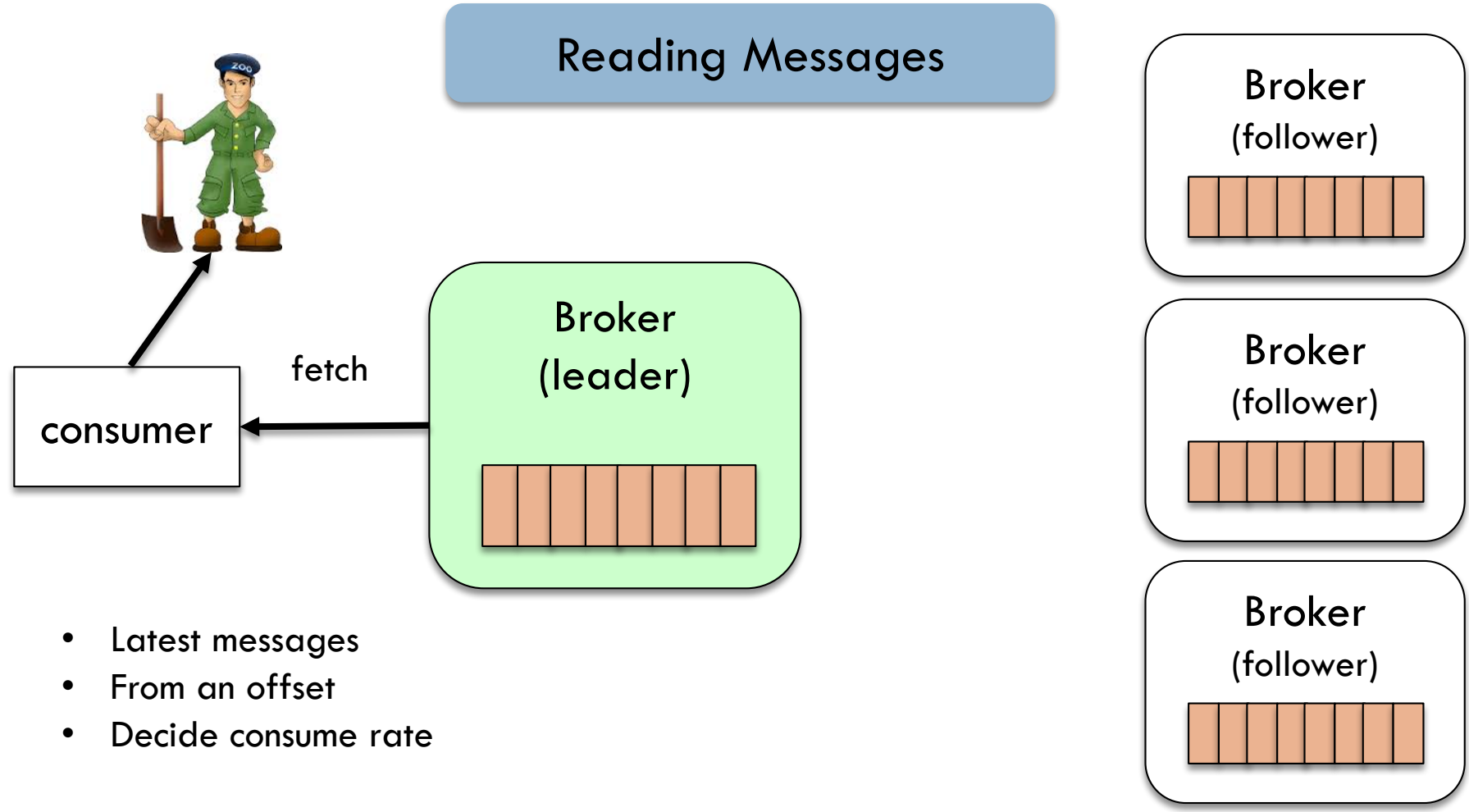
Only committed messages are ever given out to the consumer

Kafka Architecture

A paradigm shift – simplicity through delegation

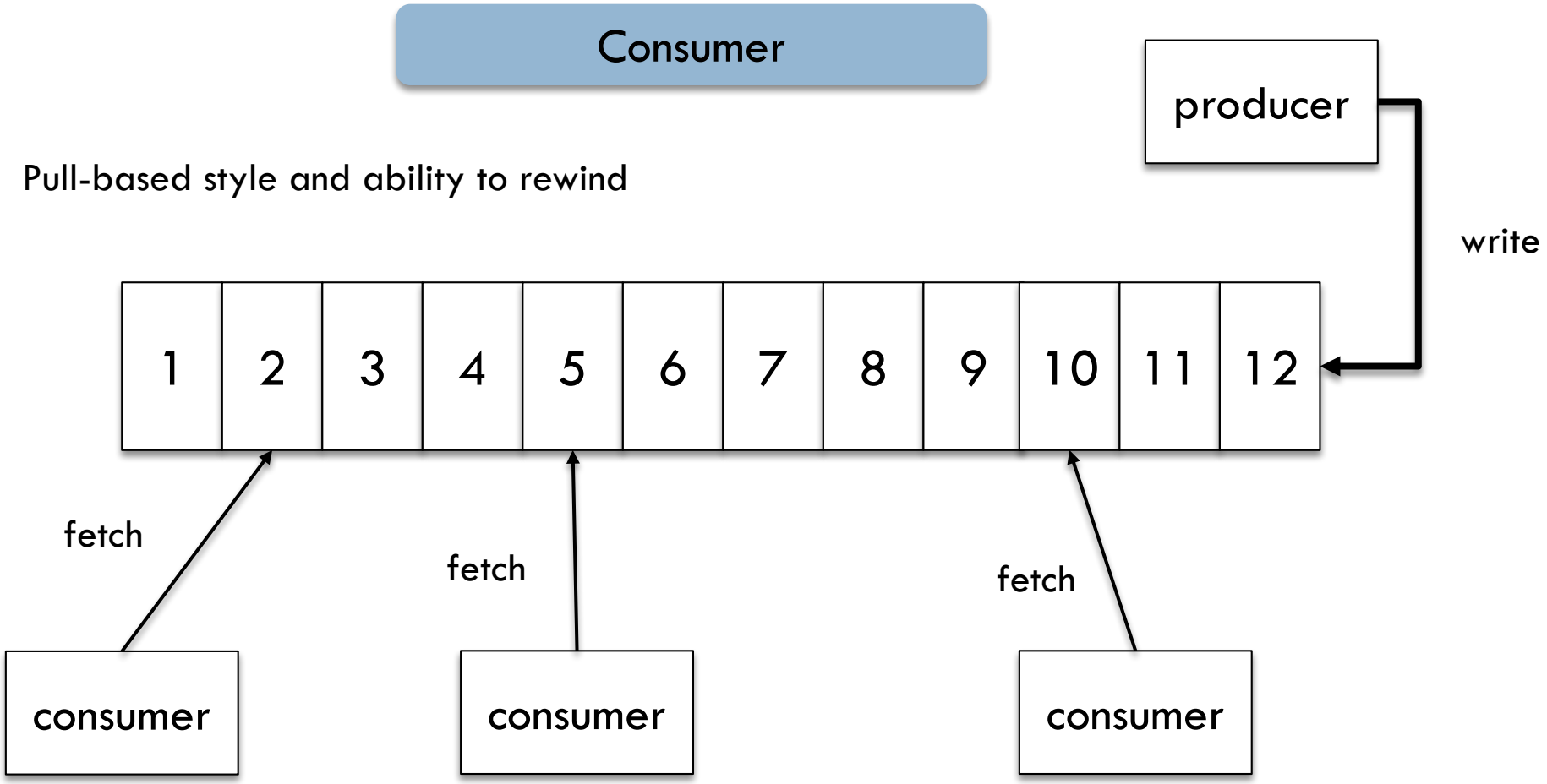
"Kafka doesn't keep track of which consumers consuming which messages"

Kafka Architecture



- Latest messages
- From an offset
- Decide consume rate

Kafka Architecture

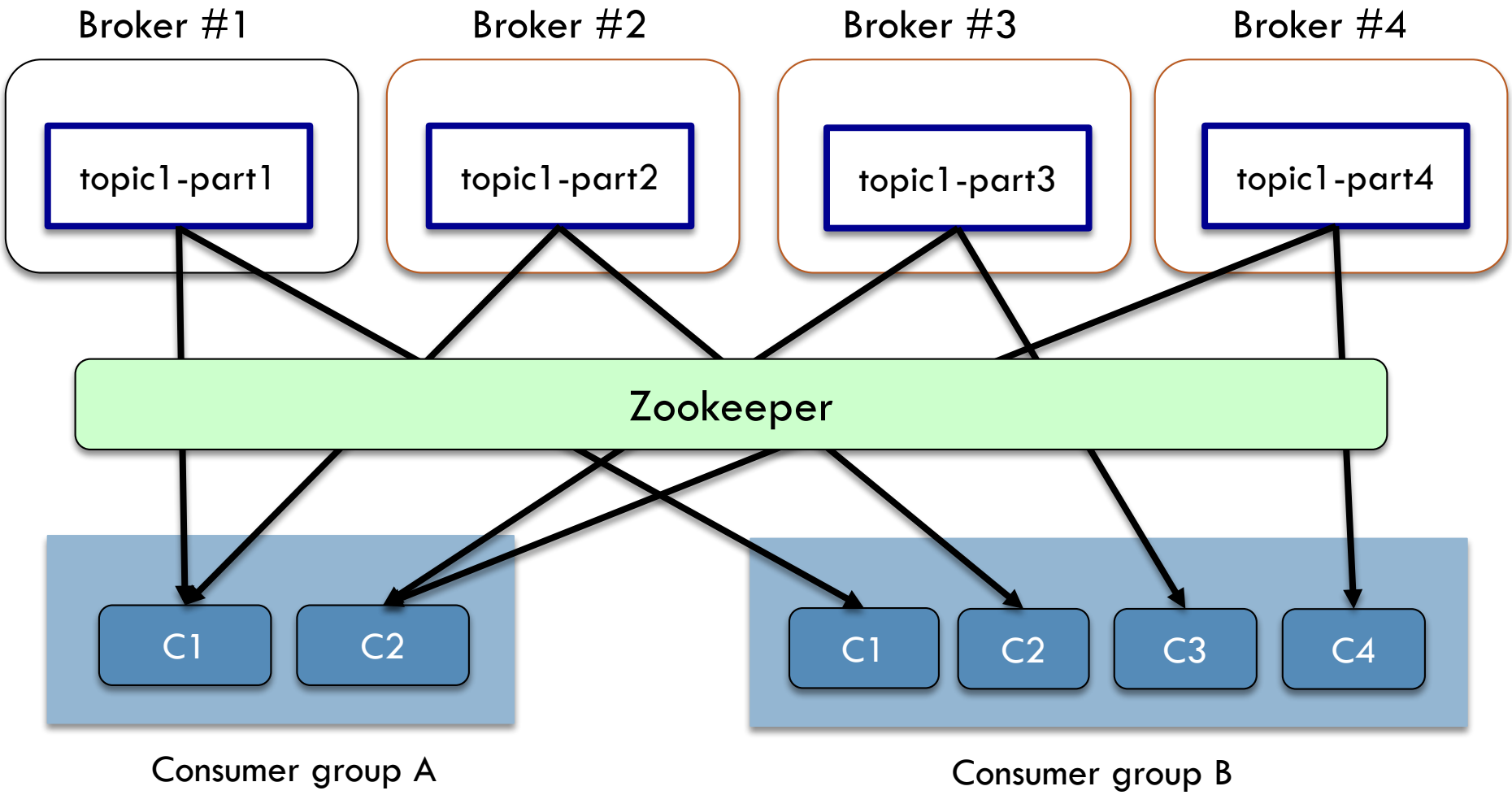


Leverage Unix sendfile API to deliver bytes from brokers to consumer

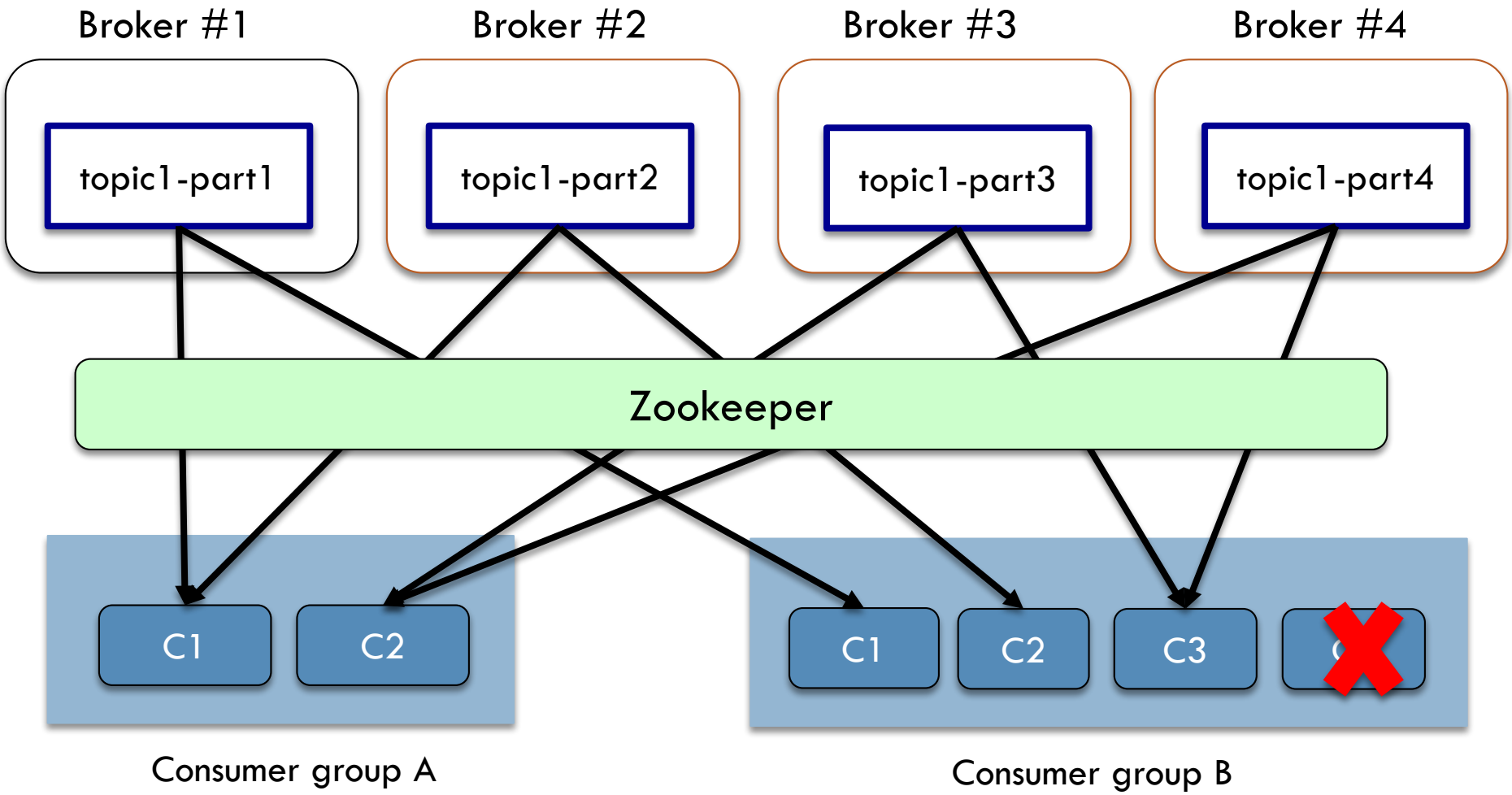
Kafka Architecture

- Consumer
  - Responsible for maintaining message offset
    - Combination of offset + partition + topic
    - Leverage Zookeeper for this
  - Consume messages from one partition
  - Flexible for rewinding
- Consumer group
  - A unique name/id within a Kafka cluster
  - A group of consumers consuming messages for a topic
  - Each member consumes messages from one partition

Kafka Architecture



Kafka Architecture



Kafka Architecture

- Message delivery guarantees – consumer in group
  - A message is delivered to only on consumer
  - Messages deliver in the order they were stored
  - Order is not guarantee across partitions
- To achieve total ordering
  - Consider single partition topic
  - Handle in your own application logic

Kafka Architecture

Message Delivery Semantics

Guarantee	Description
At most one	Messages maybe lost, but never redelivered
At least one	Messages are never lost but may be redelivered
Exactly one	Messages are never lost and deliver only once

Producer

Durability	Latency	Description
No Ack	No delay	Some data loss
Leader Ack	1 network rountrip	A few data loss
Full Ack	2 network rountrips	No data loss

Consumer - At least one guarantee delivery

Kafka Architecture

- Performance
  - 2 millions writes/second – 3 producers with 3x async replication
  - End-to-end latency
    - 2ms median, 14 ms (99.9th percentile)

https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines

Kafka APIs

Producer

```
public interface Producer<K, V> extends Closeable {
   public Future<RecordMetadata> send(ProducerRecord<K, V> record);
    public Future<RecordMetadata> send(ProducerRecord<K, V> record,
                                       Callback callback);

    public void flush();
    public List<PartitionInfo> partitionsFor(String topic);

    public Map<MetricName, ? extends Metric> metrics();

    public void close();
    public void close(long timeout, TimeUnit unit);
}
```

Kafka APIs

Producer

```
public class SimpleProducer {
  public static void main(String[] args) {
    String topic = (String) args[0];
    String count = (String) args[1];
    String msg = "Message Publishing Time - " + runtime;

    Properties props = new Properties();
    props.put("metadata.broker.list","<host>:<port>");
    props.put("serializer.class", "kafka.serializer.StringEncoder");
    props.put("request.required.acks", "1");
    ProducerConfig config = new ProducerConfig(props);

    Producer producer = new Producer<String, String>(config);
    KeyedMessage<String, String> data =
          new KeyedMessage<String, String>(topic, msg);

    producer.send(data);
  }
}
```

Kafka APIs

Consumer

```
public interface Consumer<K, V> extends Closeable {
   public void subscribe(Collection<String> topics);
   public ConsumerRecords<K, V> poll(long timeout);

   public void close();
   public void close(long timeout, TimeUnit unit);
}
```

Kafka APIs

Consumer

```
public class SimpleConsumer {
  public static void main(String[] args) {
    String topic = (String) args[0];

    Properties props = new Properties();
    props.put("zookeeper.connect","<host>:<port>");
    props.put("group.id", "mygroup");
    ConsumerConfig config = new ConsumerConfig(props);

    KafkaConsumer consumer = new KafkaConsumer(config);

    consumer.subscribe(Collections.singletonList(topic));
```

Kafka APIs

Consumer

```
    // continue from previous slide
    try {
      while (true) {
        ConsumerRecords<String,String> records = consumer.poll(100);
        for (ConsumerRecord<String,String> record : records) {
         String debugMsg = String.format("topic: %s, partition: %s,
           offset: %d, message: %s",
           record.topic(), record.partition(), record.offset(),
           record.value());
         System.out.println("message: " + debugMsg);
        }
      }
    } finally {
      consumer.close();
    }
  }
}
```

Kafka Summary

- Key takeaways
  - Distributed, partitioned and replicated
  - Simple storage
  - Message offset
  - Efficient data transfer
  - Stateless broker
  - Pull-based consumption model
  - Consumer group
  - Guarantee at-least-once delivery

Resources

- http://kafka.apache.org/documentation.htm
- http://www.confluent.io/blog
- https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines
- https://engineering.linkedin.com/kafka/running-kafka-scale
- https://engineering.linkedin.com/apache-kafka/how-we_re-improving-and-advancing-kafka-linkedin