Bukalapak

Kafka as Messaging Bus for Product Information Enrichment

JVM Meetup - Allianz Building 2nd May 2018

@masykurm

Bukalapak Introduction

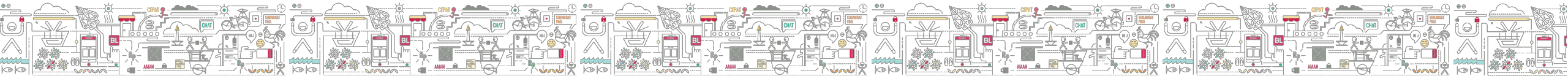
Short Overview

- One of the largest e-marketplace in Southeast Asia
- 3+ billion pageviews per month, millions of daily active users
- 15 million users , 1 Trillion IDR / month
- 1000+ total employees
- Tens of squads focusing on technology exploration and invention

Kafka Overview
Definition and Key Concepts (1/2)

Apache Kafka® is *a distributed streaming platform*

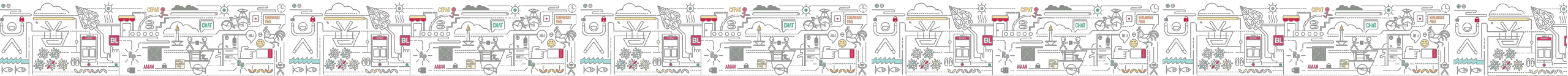
- Run as a cluster on one or more servers that can span multiple datacenters.
- Stores streams of *records* in categories called *topics*.
- Each record consists of a key, a value, and a timestamp



Kafka Overview
Definition and Key Concepts (2/2)

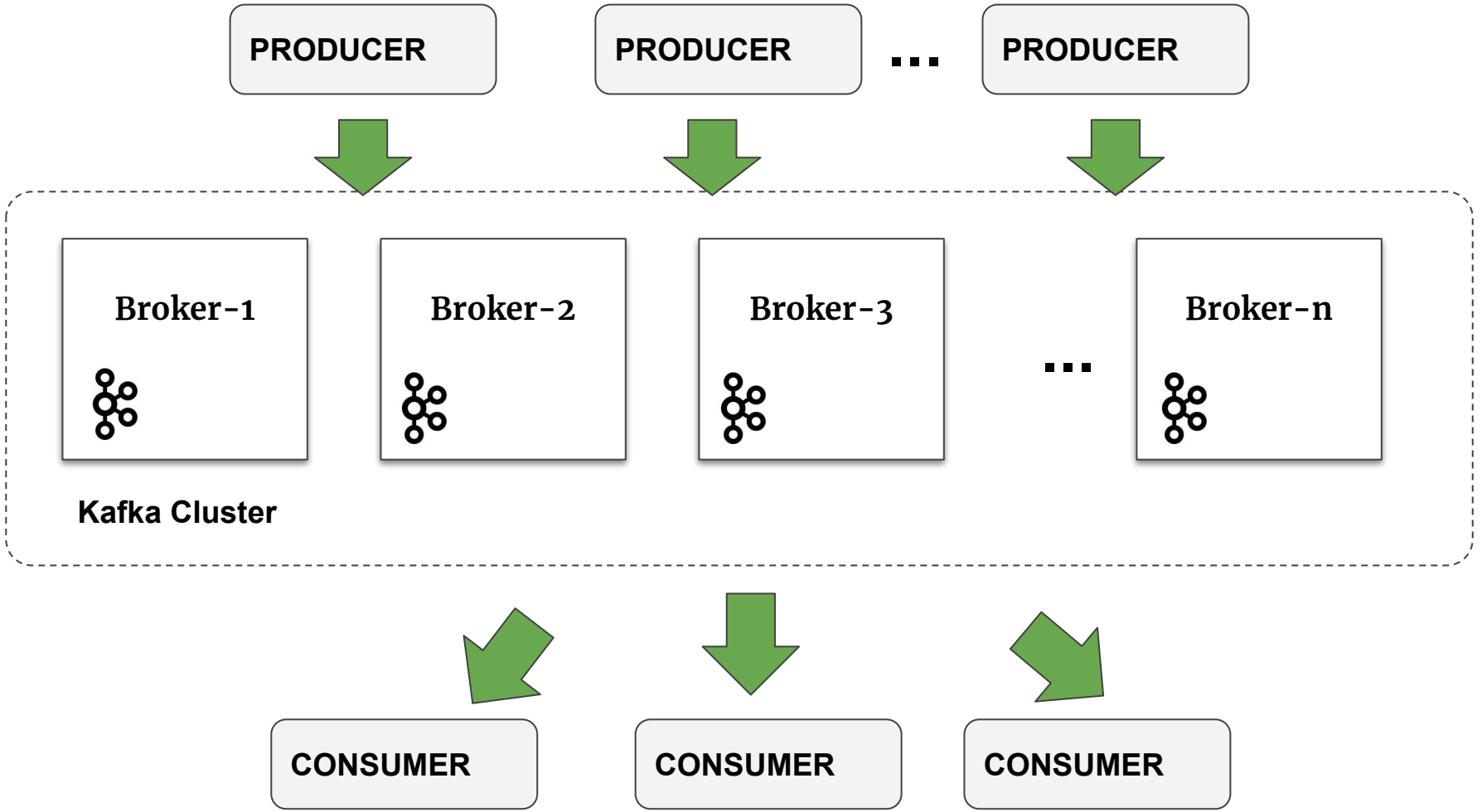
Key Usage of Apache Kafka®

- Kafka as Messaging System
  - Publish / Subscribe
  - Queuing System
- Kafka as Storage System
  - Acting as a storage system for the in-flight message
  - All data written in the disk and replicated for full fault tolerant
- Kafka for Stream Processing
  - Real time processing of continual streams of data from input topics, performs some processing on this input, and produces continual streams of data to output topics



Kafka Overview

Brokers, Topics, Replication (1/3)

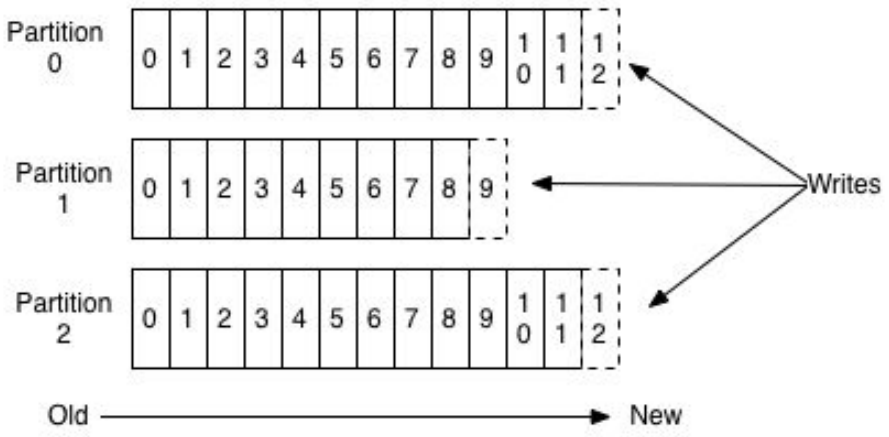


Kafka Overview
Brokers, Topics, Replication (2/3)

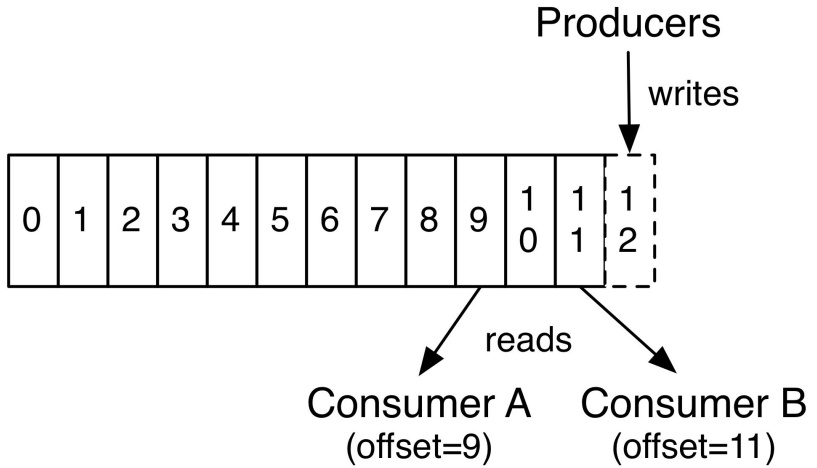
What is essentially inside Kafka Broker

Broker-1

1. Kafka Topics = Partitioned Logs

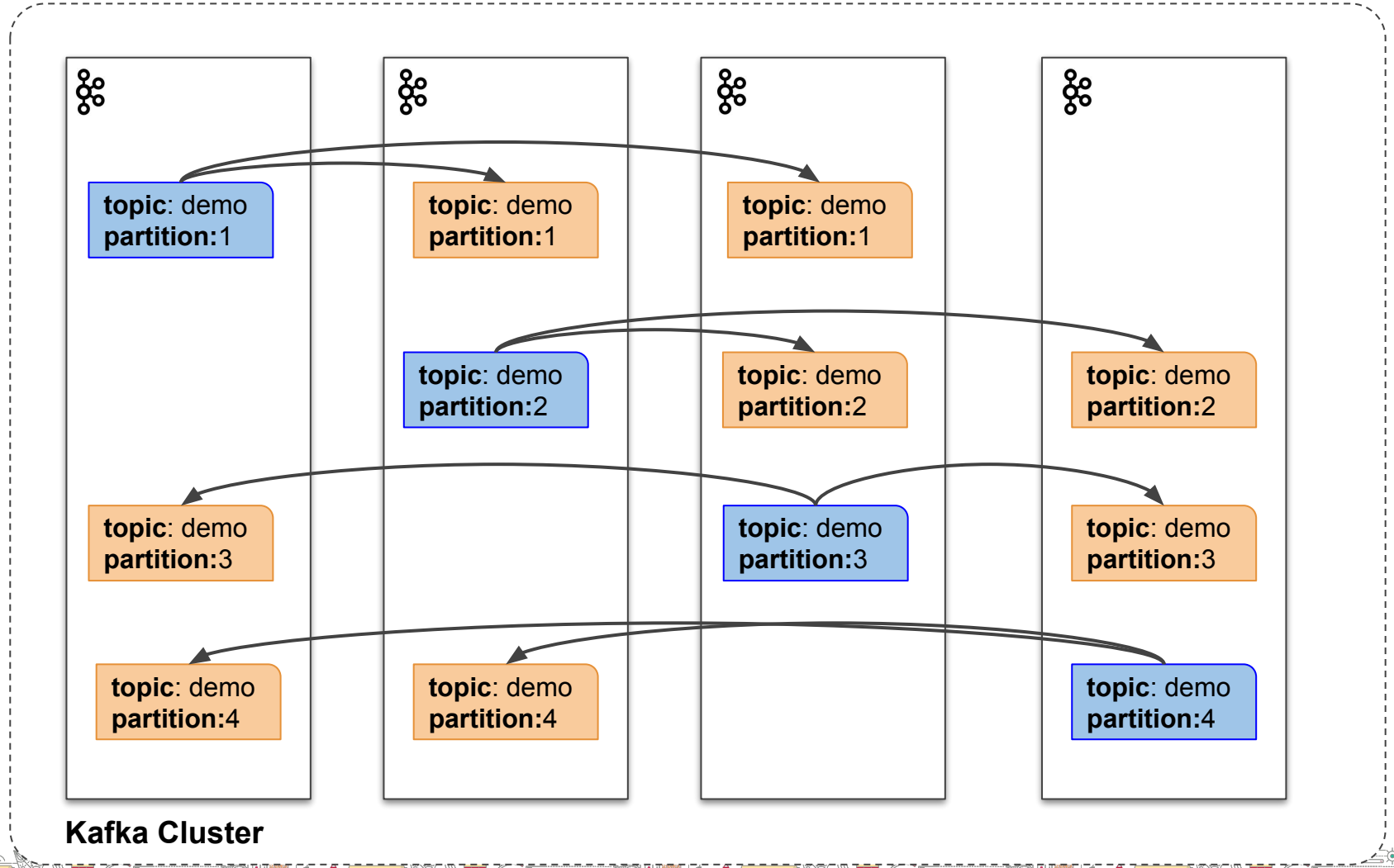


2. Read / Write Operation of Kafka Message



Kafka Overview

Brokers, Topics, Replication (3/3)



3. **Replication of Message over the cluster**

Kafka Overview

Understanding Kafka Record Batch Format (v11.0 and above)

Header (variable length)	Key Byte Array (variable length)	Value Byte Array (variable length)

```
baseOffset: int64
batchLength: int32
partitionLeaderEpoch: int32
magic: int8 (current magic value is 2)
crc: int32
attributes: int16
    bit 0~2:
        0: no compression
        1: gzip
        2: snappy
        3: lz4
    bit 3: timestampType
    bit 4: isTransactional (0 means not transactional)
    bit 5: isControlBatch (0 means not a control batch)
    bit 6~15: unused
lastOffsetDelta: int32
firstTimestamp: int64
maxTimestamp: int64
producerId: int64
producerEpoch: int16
baseSequence: int32
records: [Record]
```

CRC covers the data from the attributes to the end of the batch (i.e. all the bytes that follow the CRC). Clients must parse the magic byte before deciding how to interpret the bytes between the batch length and the magic byte

Compression type presented as 1 bit value. Kafka currently support gzip, snappy, and lz4 compression (default no compression) - decided by the Kafka Producer when sending message to Kafka

Record Header (variable length)	Record Body (variable length)

```
headerKeyLength: varint
headerKey: String
headerValueLength: varint
Value: byte[]
```

```
length: varint
attributes: int8
    bit 0~7: unused
timestampDelta: varint
offsetDelta: varint
keyLength: varint
key: byte[]
valueLen: varint
value: byte[]
```

Kafka in Bukalapak
High Level Overview

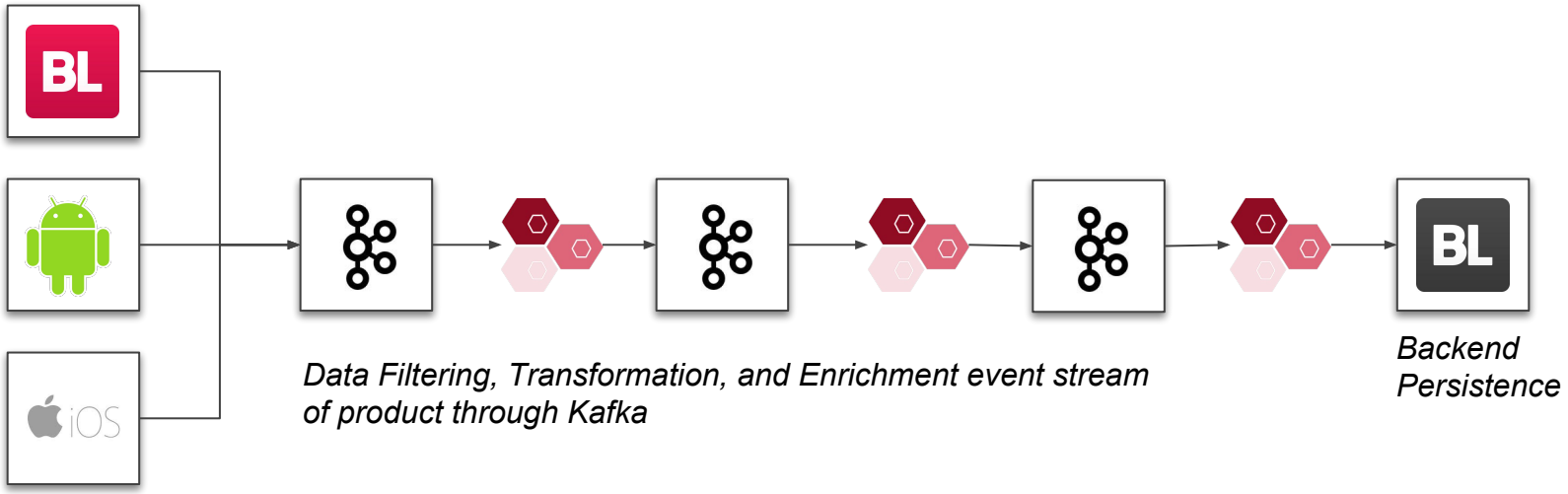
- **38+ Billions** message flowing in the clusters
- **125+ Millions** inbound message per seconds
- **1000+** Kafka Topics
- **120+** MBps bytes in throughput
- **700+** MBps bytes out throughput
- **500+** Kafka consumers

- **3** Kafka Brokers
- **3** Zookeepers

Kafka Use Case Sample in Bukalapak

Ingest and enriching product information at scale

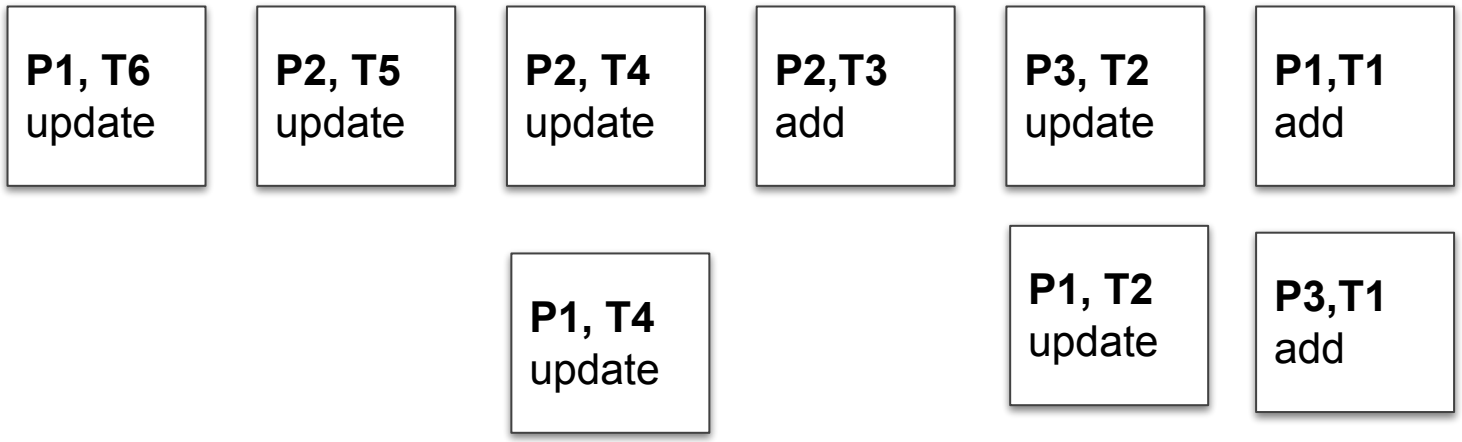
- Real Time User experience is important
- Enrichment process might involves multiple ingestion steps and may took unpredictable processing time
- Since we are handling product's event messages (creation, update, deletion), then we need to handle carefully on the **order delivery** and **processing** at it **scale**



Kafka Use Case Sample in Bukalapak

Designing streaming pipeline to ensure events delivery are processed with right order with most optimum cost

Let say we have following Event from Product X at timestamp Tn

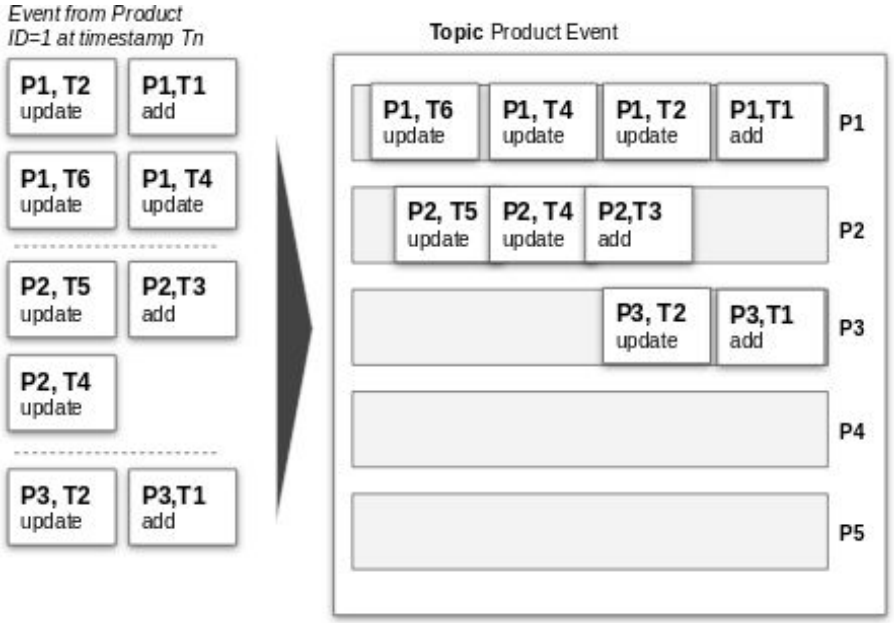


"How to make sure that Product X information that we received / processed is the latest one at large scale ?"

Kafka Use Case Sample in Bukalapak

Selecting attributes that can distinct the event as the serialized key

- Use Product ID as the **serialized key**.
- If specified, Kafka Clients library by default will calculate the partition number based on mod operation of the 32 bit hash of **the serialized key** with total number of destination topic partitions
- We can custom it btw or directly assign the partition

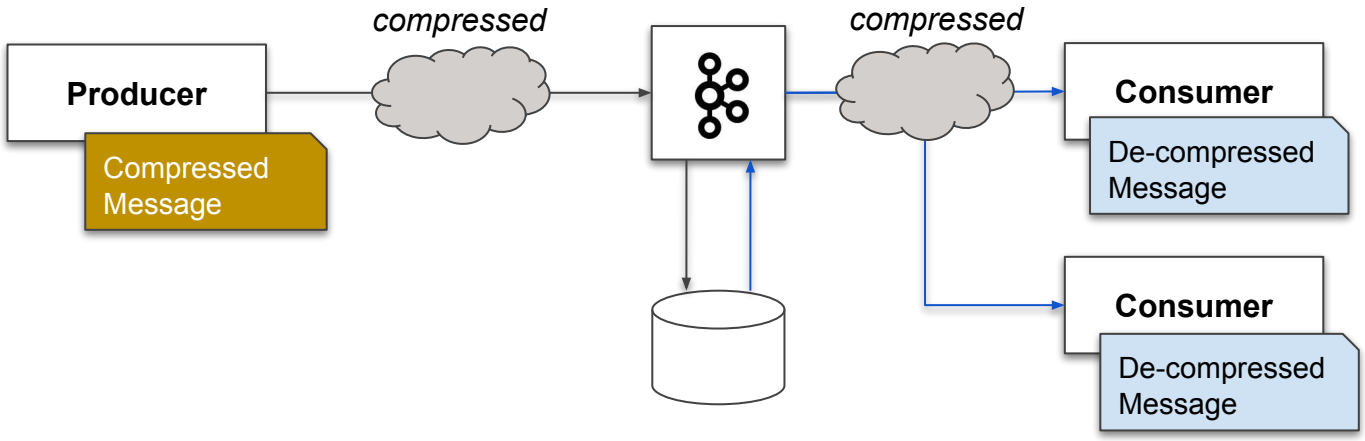



```
/**
 * Compute the partition for the given record.
 *
 * @param topic The topic name
 * @param key The key to partition on (or null if no key)
 * @param keyBytes serialized key to partition on (or null if no key)
 * @param value The value to partition on or null
 * @param valueBytes serialized value to partition on or null
 * @param cluster The current cluster metadata
 */
public int partition(String topic, Object key, byte[] keyBytes, Object value, byte[] valueByte
    List<PartitionInfo> partitions = cluster.partitionsForTopic(topic);
    int numPartitions = partitions.size();
    if (keyBytes == null) {
        int nextValue = nextValue(topic);
        List<PartitionInfo> availablePartitions = cluster.availablePartitionsForTopic(topic);
        if (availablePartitions.size() > 0) {
            int part = Utils.toPositive(nextValue) % availablePartitions.size();
            return availablePartitions.get(part).partition();
        } else {
            // no partitions are available, give a non-available partition
            return Utils.toPositive(nextValue) % numPartitions;
        }
    } else {
        // hash the keyBytes to choose a partition
        return Utils.toPositive(Utils.murmur2(keyBytes)) % numPartitions;
    }
}
```

Kafka Use Case Sample in Bukalapak

Enable message compression

- Kafka supports GZIP, Snappy and LZ4 compression protocols.
- The beauty of compression in Kafka is that it lets you trade off CPU vs disk and network usage
- Utilize kafka built-in message compression to optimize overhead on network bandwidth, and storage size

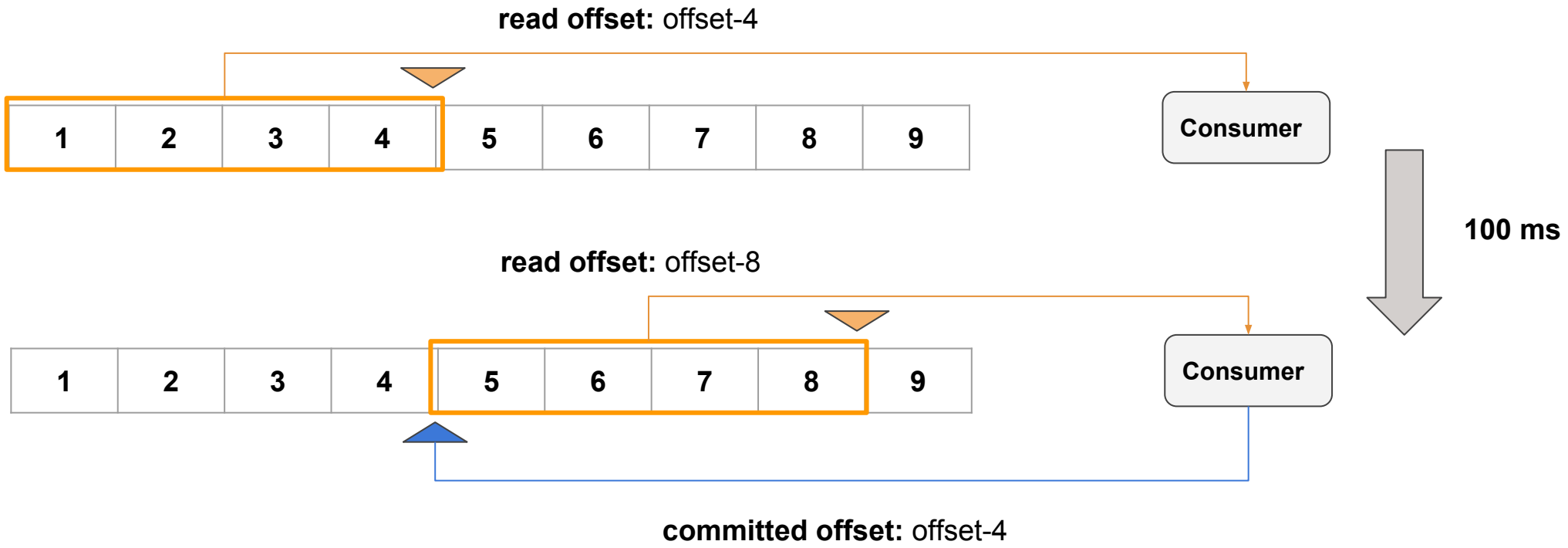


Compressor name	Ratio	Compression	Decompression
zstd	3.794	409 MB/s	844 MB/s
lz4	2.475	594 MB/s	2428 MB/s
snappy	2.313	446 MB/s	1344 MB/s

Kafka Use Case Sample in Bukalapak

Message commit manually handled on the Consumer Side

- Spring Kafka / Kafka Client by default using autocommit whenever the consumer read the message from Kafka Brokers in specified configurable time interval (through Consumer Config enable.auto.commit=true and auto.commit.interval.ms=<*some_interval_in_milliseconds*>



Kafka Use Case Sample in Bukalapak

Message commit manually handled on the Consumer Side

What is the risks by having autocommit enabled :

- If consumer suddenly terminated and processing has not finished sub milliseconds after auto commit performed. When the consumer get backs, it will read message in the next offset - Loss message
- If consumer processing finished and next sub milliseconds suddenly terminated , while autocommit has not yet performed. When the consumer get backs, it will reread and re-process the message again

Kafka Use Case Sample in Bukalapak

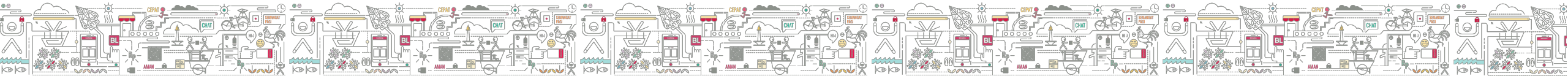
Message commit manually handled on the Consumer Side

- Nature of our use case, we cannot missed single message and need to process exactly once - so we decided to:
  - Disable auto commit
  - Full commit control on the consumer once the processing is finished
  - Handle consumer rebalancing when it happens
    i. Kafka consumer is single threaded
    ii. Partition will be revoked after Kafka Consumer completed current process - we have time to commit after processing
    iii. But if you still not sure - commit the offset when the partitions revoked

Summary

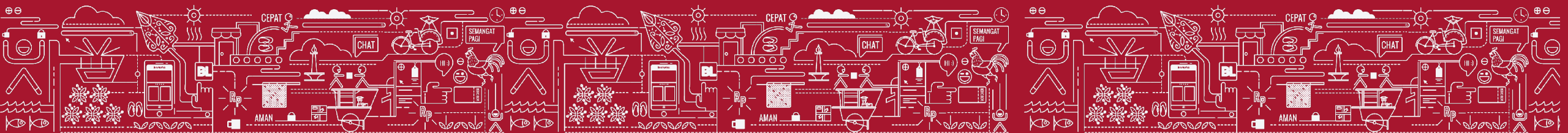
- Kafka is low latency and high throughput platform that can be used for any purposes - next gen messaging platform
- Designing producer and consumer behaviour (configuration) truly depending on the use case - no silver bullet
- However need to keep considering architectural concerns that might arise such as:
  - Capacity and throughput
  - Ordered / non-ordered processing sequences
  - Delivery semantics
  - High availability

and … we are hiring ;)



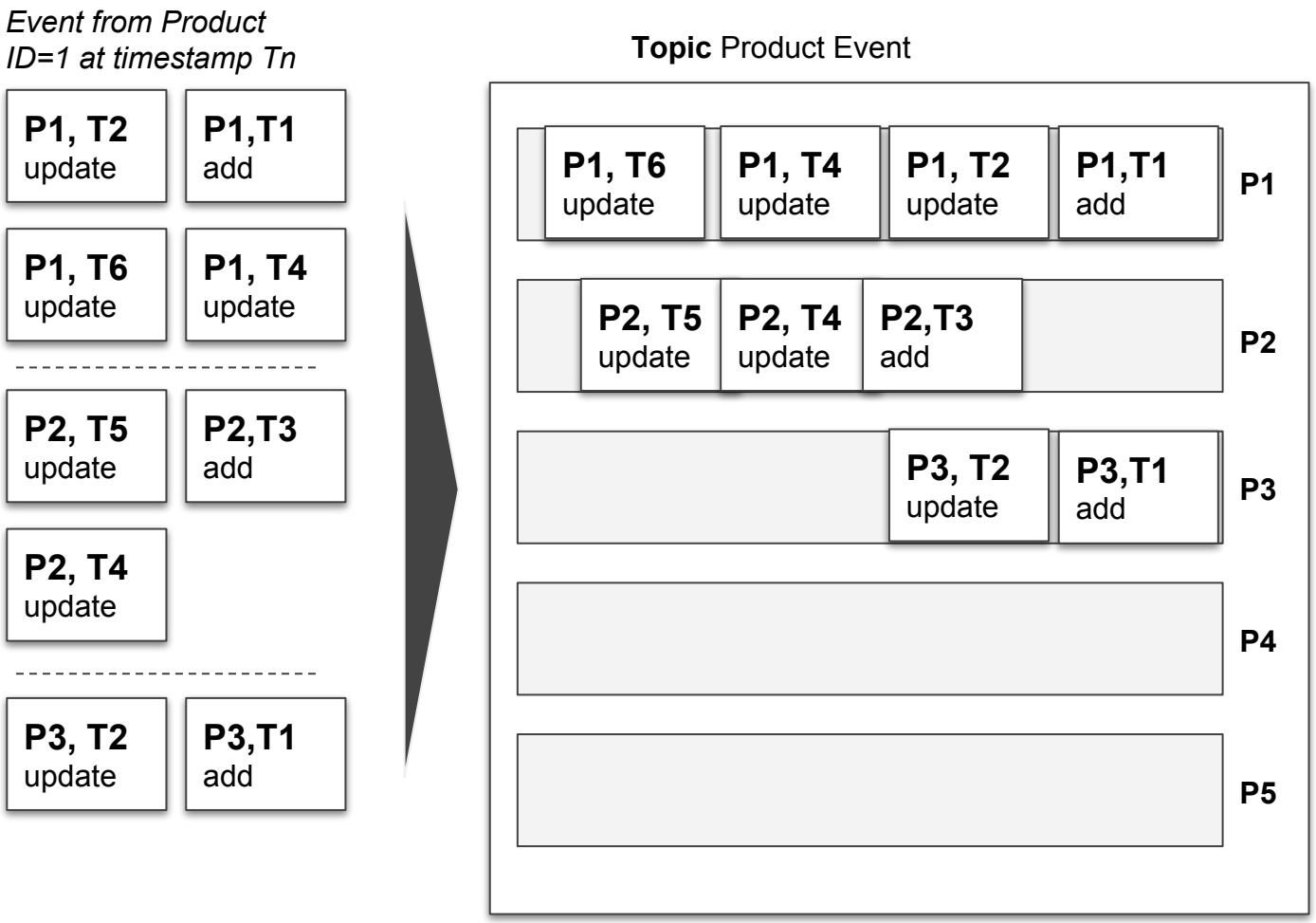
Terima Kasih

BukaLapak



Kafka Use Case Sample in Bukalapak

Designing streaming pipeline to ensure events delivery are in order



When sending event Message:

- Utilize kafka built-in message compression to minimize overhead on network bandwidth, and storage size
- Kafka supports GZIP, Snappy and LZ4 compression protocols.

Compressor name	Ratio	Compression	Decompression
zstd	3.794	409 MB/s	844 MB/s
lz4	2.475	594 MB/s	2428 MB/s
snappy	2.313	446 MB/s	1344 MB/s