

Pengembangan Aplikasi Terdistribusi

Apache Kafka
Panfapotan Napitupulu

Apache Kafka



Apache Kafka

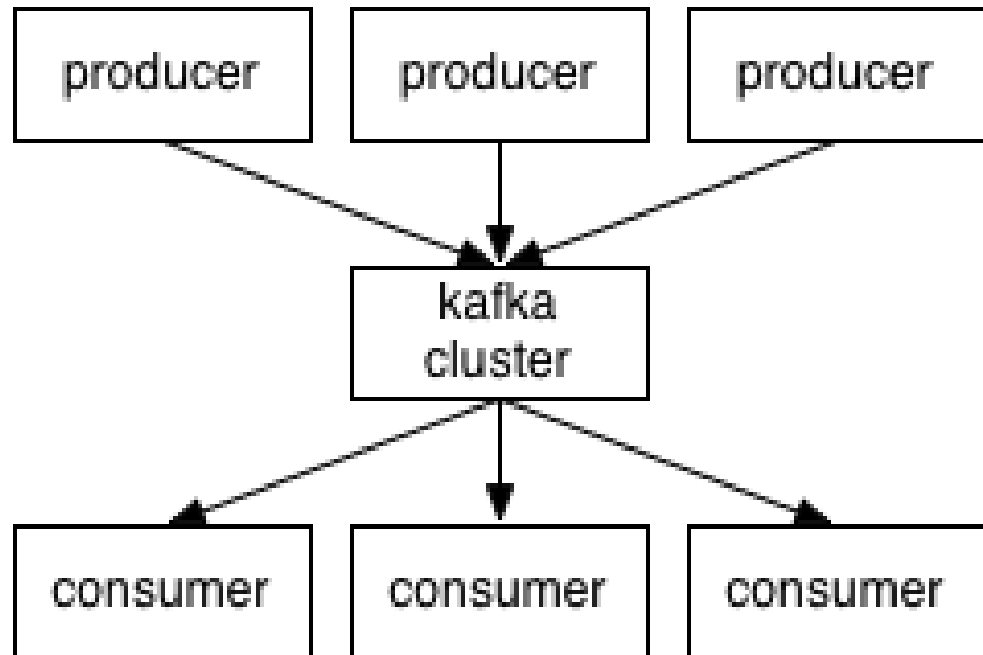
A high-throughput distributed messaging system.

- ▶ Dikembangkan di LinkedIn mulai 2011
- ▶ Diimplementasikan menggunakan Scala dan Java
- ▶ Goal
 - ▶ high throughput
 - ▶ support realtime processing
 - ▶ fault tolerant
 - ▶ fast
 - ▶ large data



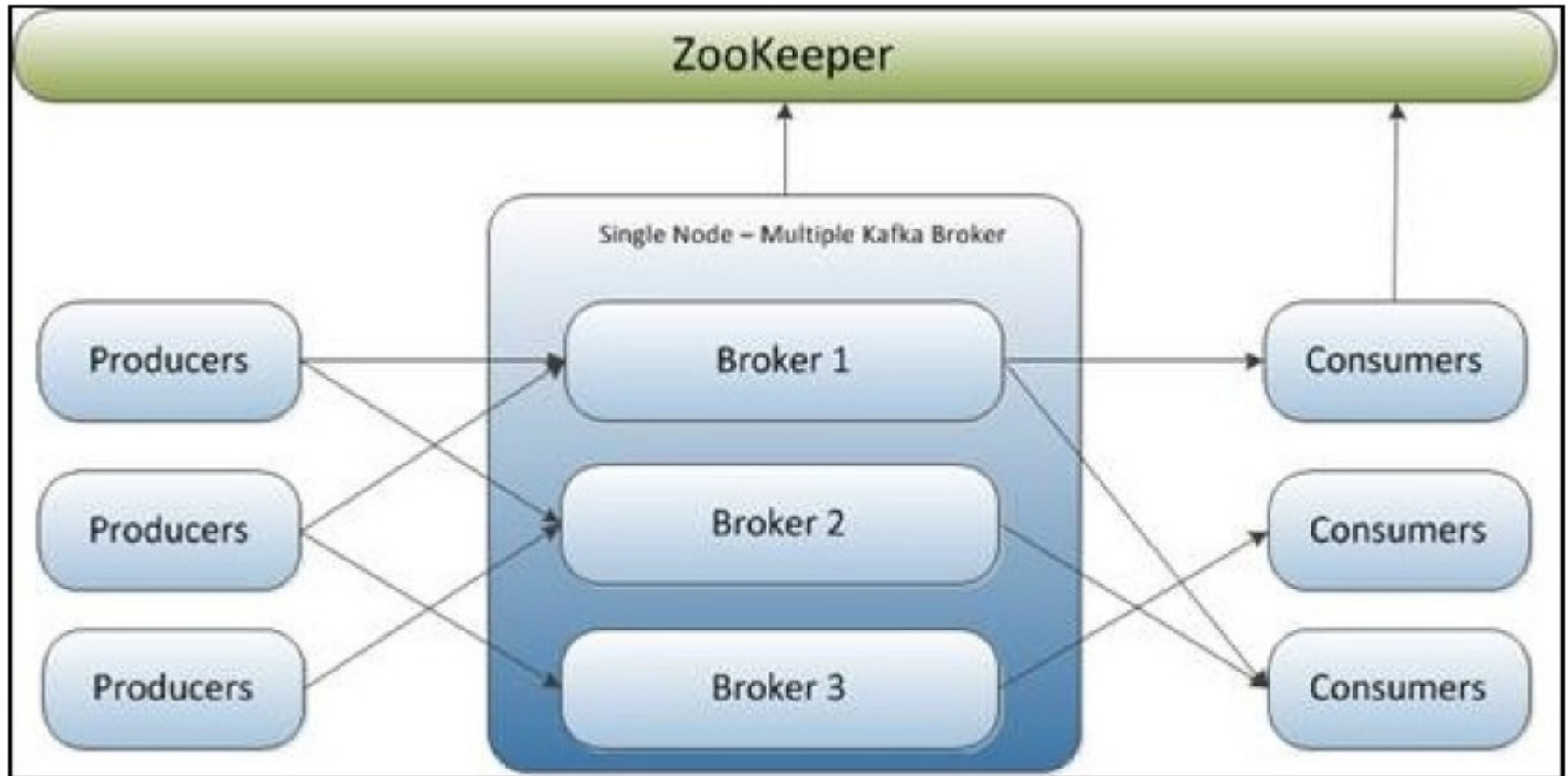
Konsep Dasar

- ▶ Producer
- ▶ Consumer
- ▶ Broker



Konsep Dasar

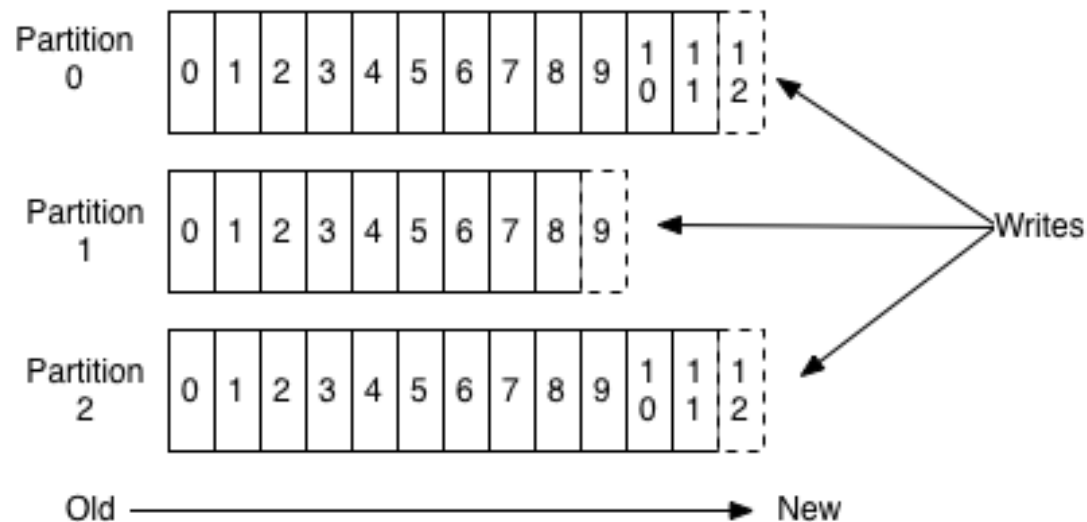
- ▶ Zookeeper:mengelola state dan koordinasi antar broker dan consumer



Konsep Dasar

- ▶ Data disimpan pada **topics**
- ▶ **topics** dibagi ke dalam **partitions**, yang direplikasi
- ▶ sebuah message pada sebuah partition diberikan id unik yang disebut sebagai **offset**

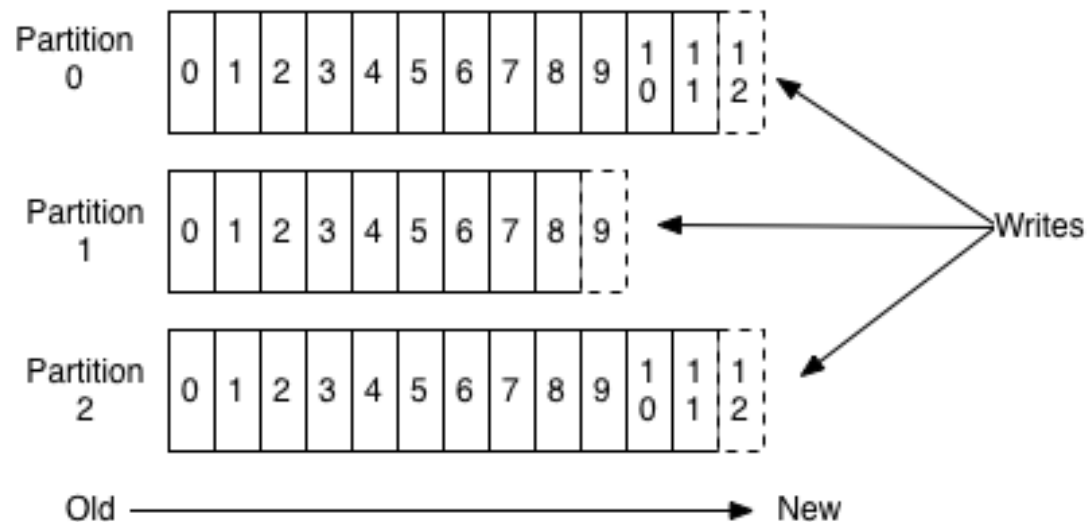
Anatomy of a Topic



Konsep Dasar

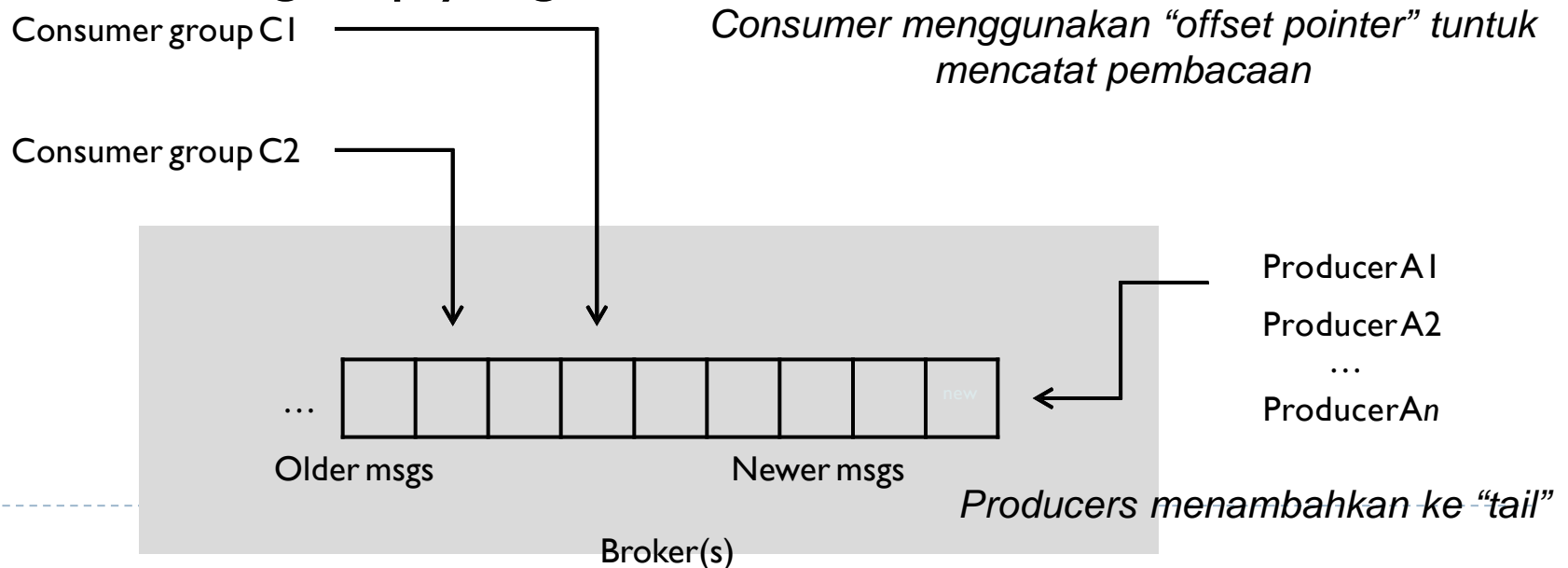
- ▶ Producer saat mengirim message ke sebuah topik, dapat menuliskan ke salah satu partisi yang ada.
- ▶ Message dalam sebuah partisi terjamin order nya berdasarkan offset id

Anatomy of a Topic



Konsep Dasar

- ▶ **Consumer Group:** pembacaan message oleh consumer dijamin unik per consumer group.
- ▶ **Publish-subscribe:** setiap consumer memiliki consumer group sendiri
- ▶ **load balancing queue:** semua consumer terhubung ke consumer group yang sama.



Menjalankan Kafka

- ▶ distribusi apache kafka menyediakan berbagai script untuk menjalankan zookeeper, server, topic management, console publisher dan console consumer
- ▶ ikuti tutorial pada website:
<http://kafka.apache.org/documentation.html#quickstart>



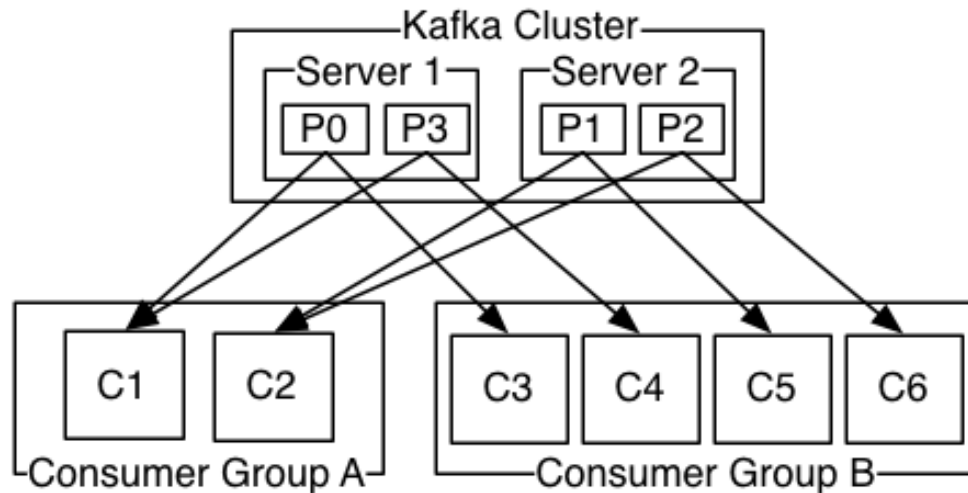
Contoh: membuat topic:

- ▶ `> bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test`
- ▶ auto create topic dapat diset dengan menggunakan property: `auto.create.topics.enable=true` pada file konfigurasi. (config/server.properties)



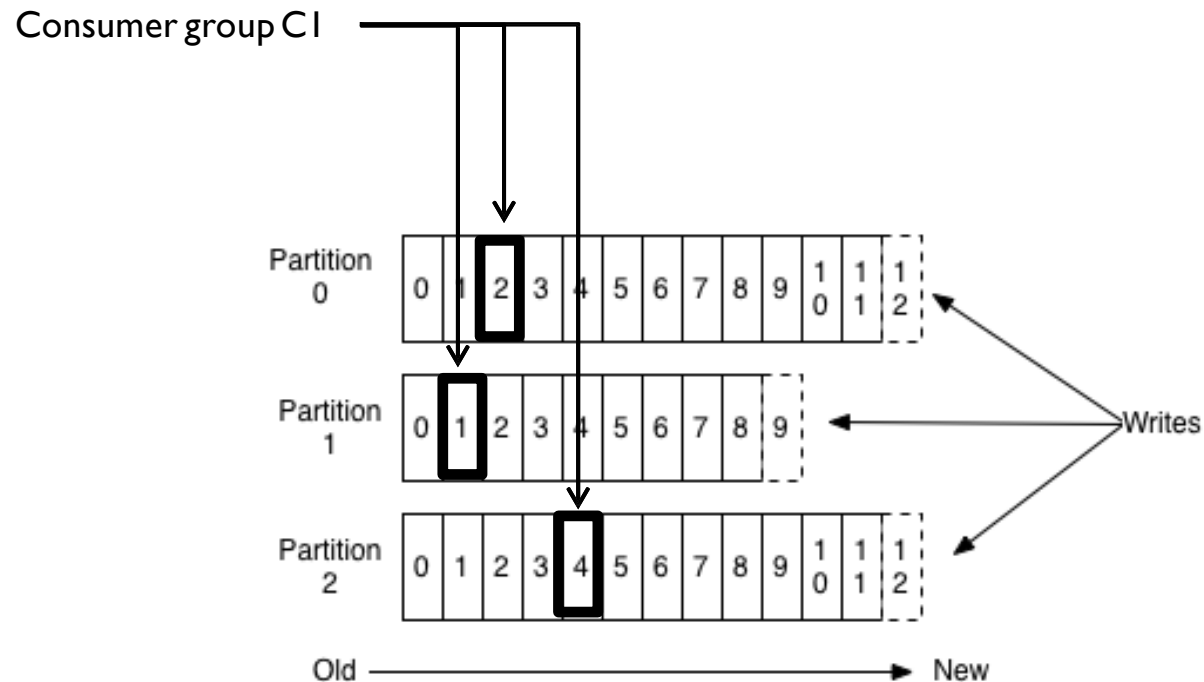
Partitions

- ▶ jumlah partisi dapat diatur/konfigurasi.
- ▶ jumlah partisi menentukan tingkat paralelisasi per consumer group



Partition offset

- ▶ message pada setiap partisi diberikan id unik yang disebut offset



Replica

- ▶ setiap partisi dapat memiliki replika, sesuai konfigurasi
- ▶ dalam sebuah partisi yang terdiri atas beberapa replica, 1 buah partisi akan menjadi leader, dan lainnya menjadi follower/backup
- ▶ writes/read hanya dilakukan pada leader
- ▶ jika leader fail, salah satu follower akan menjadi leader



Contoh kode producer

- ▶ <https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+Producer+Example>

```
1 Properties props = new Properties();
2 props.put("metadata.broker.list", "...");
3 ProducerConfig config = new ProducerConfig(props);
4
5 Producer p = new Producer(ProducerConfig config);
6 KeyedMessage<K, V> msg = ...; // cf. later slides
7 p.send(KeyedMessage<K, V> message);
```



Producer

```
1  class kafka.javaapi.producer.Producer<K,V>
2  {
3      public Producer(ProducerConfig config);
4
5      /**
6       * Sends the data to a single topic, partitioned by key, using either the
7       * synchronous or the asynchronous producer.
8       */
9      public void send(KeyedMessage<K,V> message);
10
11     /**
12      * Use this API to send data to multiple topics.
13      */
14     public void send(List<KeyedMessage<K,V>> messages);
15
16     /**
17      * Close API to close the producer pool connections to all Kafka brokers.
18      */
19     public void close();
20 }
```



Producer

▶ setting async producer

```
1 Properties props = new Properties();  
2 props.put("producer.type", "async");  
3 ProducerConfig config = new ProducerConfig(props);
```

▶ konfigurasi producer

client.id	identifies producer app,e.g. in system logs
producer.type	async or sync
request.required.acks	acking semantics, cf. next slides
serializer.class	configure encoder,cf.slides on Avro usage
metadata.broker.list	cf.slides on bootstrapping list of brokers



Consumer

► konfigurasi penting:

<code>group.id</code>	assigns an individual consumer to a “group”
<code>zookeeper.connect</code>	to discover brokers/topics/etc., and to store consumer state (e.g. when using the high-level consumer API)
<code>fetch.message.max.bytes</code>	number of message bytes to (attempt to) fetch for each partition; must be \geq broker's <code>message.max.bytes</code>

