# Message Compression in Apache Kafka using Spring Boot

Last Updated : 28 Apr, 2025

Generally, producers send text-based data, such as JSON data. It is essential to apply compression to the producer in this situation. Producer messages are transmitted uncompressed by default. There are two types of Kafka compression.

## 1. Producer-Level Kafka Compression

When compression is enabled on the producer side, no changes to the brokers or consumers are required. With the compression.type parameter, producers may select whether to compress messages or not. There are five options available for compression type,

- none
- gzip
- lz4
- snappy
- zstd

If enabled, compression is carried out by the producer client. Producers must batch messages together for higher throughput.

### Implementation

**BATCH_SIZE_CONFIG:** When sending multiple records to the same partition, the producer will attempt to combine records into fewer requests. Performance on the client and server is improved by this. Bytes are the default batch size set by this option. Any attempt to group

records bigger than this will fail. Multiple batches will be included in requests made to brokers, one for each partition with accessible data for sending. Small batch size will prevent batching and may lower throughput (a batch size of zero will disable batching entirely). As we always allocate a buffer of the provided batch size in anticipation of more records, a very large batch size may consume memory a little bit more wastefully.

**LINGER_MS_CONFIG:** The producer compiles all records that come in between broadcasts of the request into a single batch request. This usually only happens under load, when records arrive more quickly than they can be sent out. But sometimes, even with a light load, the client could choose to cut back on requests. This option does this by introducing a tiny amount of artificial delay; thus, the producer will wait up to the specified delay before sending a record in order to enable additional records to be sent so that the sends may be batch processed.

This setting establishes the upper limit of the batching delay: regardless of this setting, once we accumulate a batch.size worth of records for a partition, they will be sent immediately; however, if we have accumulated fewer bytes for this partition than this, we will "linger" for the specified amount of time while we wait for new records to arrive. Its default value is 0 (i.e., no delay). For instance, lowering linger.ms to 3



there is no load would be delayed by up to 3 ms.

**COMPRESSION _TYPE_CONFIG:** The same compression type will be used by the producer for all data. None is the default compression type (it means no compression). Valid compression type is none, gzip, snappy, zstd, and lz4. The efficiency of batching will also have an influence on the compression ratio, as compression uses whole batches of data (more batching means better compression).

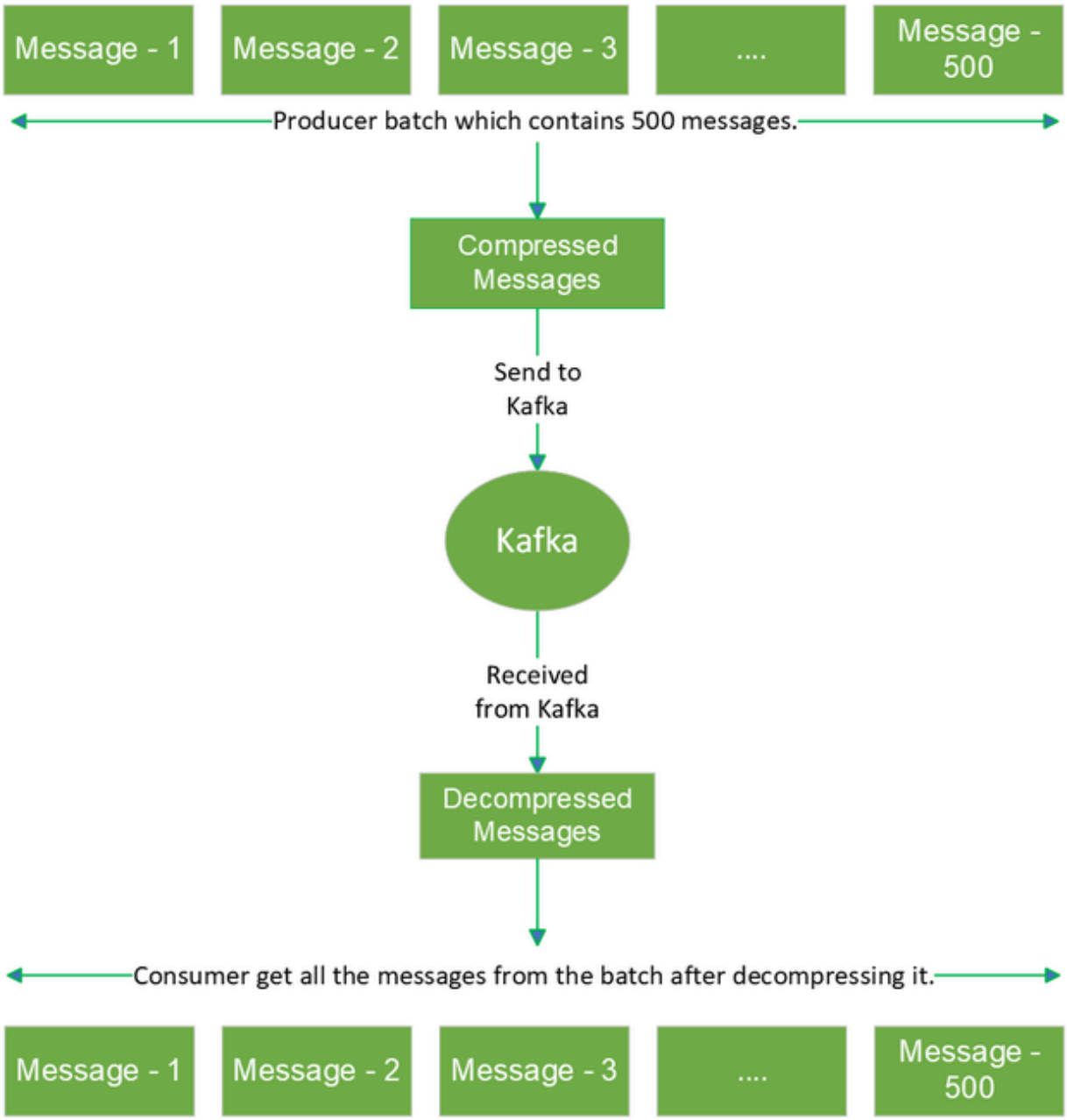
Add configuration to the producer in Java(spring boot),

```
import java.io.*;
```

```
@Configuration
class GFG {

    @Bean(name = "dummyData")
    public KafkaProducer<String, DummyData> dummyDataKafkaProducer() {
        Map<String, Object> config = new HashMap<>();
        config.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"localhost:9092");
        config.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class);
        config.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
JsonSerializer.class);
        config.put(ProducerConfig.DELIVERY_TIMEOUT_MS_CONFIG, 600000);

        // this configuration for compression
        config.put(ProducerConfig.BATCH_SIZE_CONFIG, 86016);
        config.put(ProducerConfig.LINGER_MS_CONFIG, 100);
        config.put(ProducerConfig.COMPRESSION_TYPE_CONFIG, "snappy");

        return new KafkaProducer<>(config, new StringSerializer(), new
JsonSerializer<>());
    }

}
```

## 2. Broker-Level Kafka Compression

The broker receives the compressed batch from the client and writes it straight to the topic's log file without re-compressing the data if the destination topic's compression configuration is set to compression.type=producer. By default, compression.type is set to the producer in the broker configuration. If the topic has its own compression option (for example, gzip) and the value matches the producer setting, the message batch is recorded to the log file in its current state; otherwise, the broker will decompress and re-compress the messages into its predetermined format.




*Kafka Message Compression*

## Conclusion

When we need to send massive amounts of data to Kafka, message compression comes in quite handy. Network calls will be drastically cut, and storage requirements for compressed messages will be greatly reduced.

selenium

Comment | More info

## Explore

### Java Basics

### OOP & Interfaces

### Collections

### Exception Handling

### Java Advanced

### Practice Java

GeeksforGeeks
Sanchhaya Education Private Limited

**Corporate & Communications Address:**

A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

**Registered Address:**

K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305