KIP-512:Adding headers to RecordMetaData

- Status
- Motivation
- Public Interfaces
- Proposed Changes
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

Status

Current state: Under Discussion

Discussion thread: https://www.mail-archive.com/dev@kafka.apache.org/msg100621.html

Voting Thread: https://www.mail-archive.com/dev@kafka.apache.org/msg100812.html

JIRA: KAFKA-8830

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

As part of KIP-82, the headers are added to Kafka ProducerRecord and ConsumerRecord.As discussed in that KIP the headers provide lot of benefits especially metadata association with the record.

Currently the RecordMetadata contains details like topic, partition, offset, timestamp etc. But there is no information on for which record this RecordMetadata belongs to. Having headers information as part of RecordMetaData will bridge this gap.

It will make the context of the record available to Interceptors and open up possibility for building more features.

Example:

As the Kafka usage is growing, we see Kafka being used for Payment transactional, Settlement and reconciliation use cases. These kind of use cases are very sensitive to message loss.Current dataloss auditing systems like chaperone uses time window based aggregations, which is not sufficient when we need message level traceability. In order to build message level traceability, we need the capability to track message production and consumption.If we choose to track message production status based on acknowledgment the producer receives, we will need ability to relate acknowledgment to the record. Having this ability in Interceptors can help us to link the record to its acknowledgement.

Public Interfaces

We will change **RecordMetaData** and **FutureRecordMetaData** to have headers information.

Proposed Changes

1. Changes to **RecordMetaData**

RecordMetaData.java

```
public final class RecordMetadata {

    /**
     * Partition value for record without partition assigned
     */
    public static final int UNKNOWN_PARTITION = -1;

    private final long offset;
    ..........
    private final TopicPartition topicPartition;
    private final Header[] headers;

    private volatile Long checksum;

    public RecordMetadata(TopicPartition topicPartition, long baseOffset, long relativeOffset, long
timestamp,
                          Long checksum, int serializedKeySize, int serializedValueSize, Header[]
headers) {
        // ignore the relativeOffset if the base offset is -1,
        // since this indicates the offset is unknown
        this.offset = baseOffset == -1 ? baseOffset : baseOffset + relativeOffset;
        this.timestamp = timestamp;
        this.checksum = checksum;
        this.serializedKeySize = serializedKeySize;
        this.serializedValueSize = serializedValueSize;
        this.topicPartition = topicPartition;
        this.headers = headers;
    }
    ......

    /**
     * The headers of the record was sent to
     */

    public Header[] headers() {
        return this.headers;
    }


   ......
}
```

2. Changes to **FutureRecordMetaData**

FutureRecordMetaData.java

```
public final class FutureRecordMetadata implements Future<RecordMetadata> {

    private final ProduceRequestResult result;
    ..........
    private final Header[] headers;
    private volatile FutureRecordMetadata nextRecordMetadata = null;

    public FutureRecordMetadata(ProduceRequestResult result, long relativeOffset, long createTimestamp,
                                Long checksum, int serializedKeySize, int serializedValueSize, Time
time, Header[] headers) {
        this.result = result;
        this.relativeOffset = relativeOffset;
        this.createTimestamp = createTimestamp;
        this.checksum = checksum;
        this.serializedKeySize = serializedKeySize;
        this.serializedValueSize = serializedValueSize;
        this.time = time;
        this.headers = headers;
    }

.......

}
```

3. In ProduceBatch add headers to FutureRecordMetaData while appending record to the batch.

ProducerBatch.java

```
 public FutureRecordMetadata tryAppend(long timestamp, byte[] key, byte[] value, Header[] headers,
Callback callback, long now) {
         .......
        .........
            FutureRecordMetadata future = new FutureRecordMetadata(this.produceFuture, this.recordCount,
                                                                   timestamp, checksum,
                                                                   key == null ? -1 : key.length,
                                                                   value == null ? -1 : value.length,
                                                                   Time.SYSTEM, headers);
            // we have to keep every future returned to the users in case the batch needs to be
            // split to several new batches and resent.
            thunks.add(new Thunk(callback, future));
            this.recordCount++;
            return future;
        }
    }
```

4. Add headers to RecordMetaData for send error scenarios also.

Compatibility, Deprecation, and Migration Plan

- What impact (if any) will there be on existing users?

  There is no impact. Since we are passing same reference to the headers there is no memory impact.

- If we are changing behavior how will we phase out the older behavior?

  Older behavior does not change so no need to phase it out.

- If we need special migration tools, describe them here.

No special migration tools needed.

- When will we remove the existing behavior?

  Not applicable.

Rejected Alternatives

1. Producer callback provides a way to link record to its RecordMetaData.

In this case the context linking on client side and depends on client implementation where as interceptors having similar RecordMetaData but no access to headers onAcknowledgement(...).

Producer callback

```
producer.send(message, (RecordMetadata metadata, Exception exception) -> {
                if (metadata!=null && exception == null) {

                    String topic = metadata.topic();

                    Iterator<Header> it = message.headers().iterator();

                   .....
                }
            });
```