KIP-391: Allow Producing with Offsets for Cluster Replication [Discarded]

co-authored-by: Mickael Maison <mickael.maison@gmail.com>

co-authored-by: Edoardo Comar <ecomar@uk.ibm.com>

Status

Current state: *"Discarded"*

Discussion thread: *here* *or* *here*

JIRA: *KAFKA-7666*

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

Motivation

Replicating topic data across multiple Kafka clusters is a very common scenario and there are many tools that provide this functionality. However simply copying topic records (key/value/headers/timestamp) may not be enough in many use cases.

Offsets are currently automatically assigned by brokers upon receiving messages. So when replicating data into another cluster, messages in the destination cluster are likely to have a different offsets than the originals in the source cluster. This makes replicating the __consumer_offsets topic ineffective and consumers can't rely on their group metadata stored in the source cluster when switching to another cluster.

We propose a mechanism to replicate records able to maintain the same offset in both clusters.
Such a mechanism could be easily used in Kafka Connect, although the Connect framework will need to be updated slightly. That enhancement will be the subject of a follow-up KIP.

Proposed Changes

We propose allowing producers to send each record data (key/value/headers/timestamp) with an offset. That offset can be accepted by the broker as the offset in the topic-partition log.

- Producers can send a record-with-offset.
  - The Record Accumulator yields batches that only contain records with consecutive offsets.
  - The Sender sends `ProduceRequests` with a `"use_offset"` flag.
- When the broker handles requests with this flag set:
  - it checks against a new required permission (`ReplicatorWrite`) in the ACL. This operation is checked against the `ClusterResource`.
  - it uses the provided offsets instead of generating new incremental ones.
- The broker still ensures offsets are monotonically increasing.
  - If a batch violates this requirement, an `InvalidProducerOffset` Error is returned.
- The broker ensures batches only contain records with sequential offsets.
  - If a batch violates this requirement, a `CorruptRecord` Error is returned.
  - Batches are required to only contain sequential offsets because the last offset is used to compute the last sequence for Idempotent Producers.
- The `ProduceResponse` contains per-topic-partition LogEndOffsets.

Public Interfaces

Network protocol

new Error Code

```
INVALID_PRODUCE_OFFSET (77)
```

new version for ProduceRequest (v8)

```
Produce Request (Version: 8) => transactional_id acks timeout [topic_data]
  transactional_id => NULLABLE_STRING
  acks => INT16
  timeout => INT32
  topic_data => topic [data]
    topic => STRING
    data => partition record_set
      partition => INT32
      record_set => RECORDS
  use_offset => BOOLEAN    <--- NEW
```

new version for ProduceResponse (v8)

```
Produce Response (Version: 8) => [responses] throttle_time_ms
  responses => topic [partition_responses]
    topic => STRING
    partition_responses => partition error_code base_offset log_append_time log_start_offset
      partition => INT32
      error_code => INT16
      base_offset => INT64
      log_append_time => INT64
      log_start_offset => INT64
      log_end_offset => INT64     <--- NEW
  throttle_time_ms => INT32
```

Changes to the Java API

New class **InvalidProduceOffsetException**

InvalidProduceOffsetException.java

```
package org.apache.kafka.common.errors;
/**
 * Thrown when the offset specified in a Produce request is smaller than the current Log End Offset
 * @see org.apache.kafka.clients.producer.ProducerRecordWithOffset
 */
public class InvalidProduceOffsetException extends InvalidOffsetException {
      public InvalidProduceOffsetException(String message);
      public InvalidProduceOffsetException(String message, long logEndOffset);
      public long getLogEndOffset();
}
```

This exception is mapped to the new protocol Error 77

Change to **ProducerRecord**

ProducerRecord.java

```
package org.apache.kafka.clients.producer;
public class ProducerRecord {
    // ...
        // new method
        public OptionalLong offset() {
           return OptionalLong.empty();
        }
}
```

New class **ProducerRecordWithOffset**

ProducerRecordWithOffset.java

```
package org.apache.kafka.clients.producer;
public class ProducerRecordWithOffset<K, V> extends ProducerRecord<K, V> {
        //constructor
        public ProducerRecordWithOffset(String topic, Integer partition, Long timestamp,
                                    K key, V value, Iterable<Header> headers,
                                    long offset);
        @Override
        public OptionalLong offset() {
                    return OptionalLong.of(this.offset);
        }
}
```

new Acl Operation

AclOperation.java

```
package org.apache.kafka.common.acl;
public enum AclOperation {
    /**
     * REPLICATOR_WRITE operation (Produce with offsets).
     */
    REPLICATOR_WRITE((byte) 13);
}
```

Command line tools and arguments

- The `AclCommand` tool has been updated to handle the new ACL operation

Compatibility, Deprecation, and Migration Plan

- This is new functionality that does not impact existing users.

Rejected Alternatives

- Allow batches with non consecutive offsets (gaps):
    - Sending batches with non consecutive offsets could provide more efficient batching on the client side,. However allowing such gaps would require changes to how the idempotent producers state is stored in the brokers, i.e. changes to the log format. The current consecutive offset assumption allows to easily compute the next expected sequence for idempotent producers.
- Add new producer method for sending regular records plus offsets
    - To keep the Producer API simple, we decided to extend the ProducerRecord
- Mixing records with and without offset in a single ProduceRequest
    - The replicator use case does not need to handle ProduceRequest in which some partitions have assigned offsets and some don't. So, in order to keep the implementation simple, the new flag is per-ProduceRequest and not per-topic-partition.

- Log level replication including empty/control batches
  - To fit in Kafka Connect, we preferred a replicator running at the regular client level instead of a lower level client based on raw protocol messages
  - By not replicating the control batches, consumers in the target cluster won't be able to distinguish between committed/uncommitted records.
    The replicator will have to run choosing one isolation level.