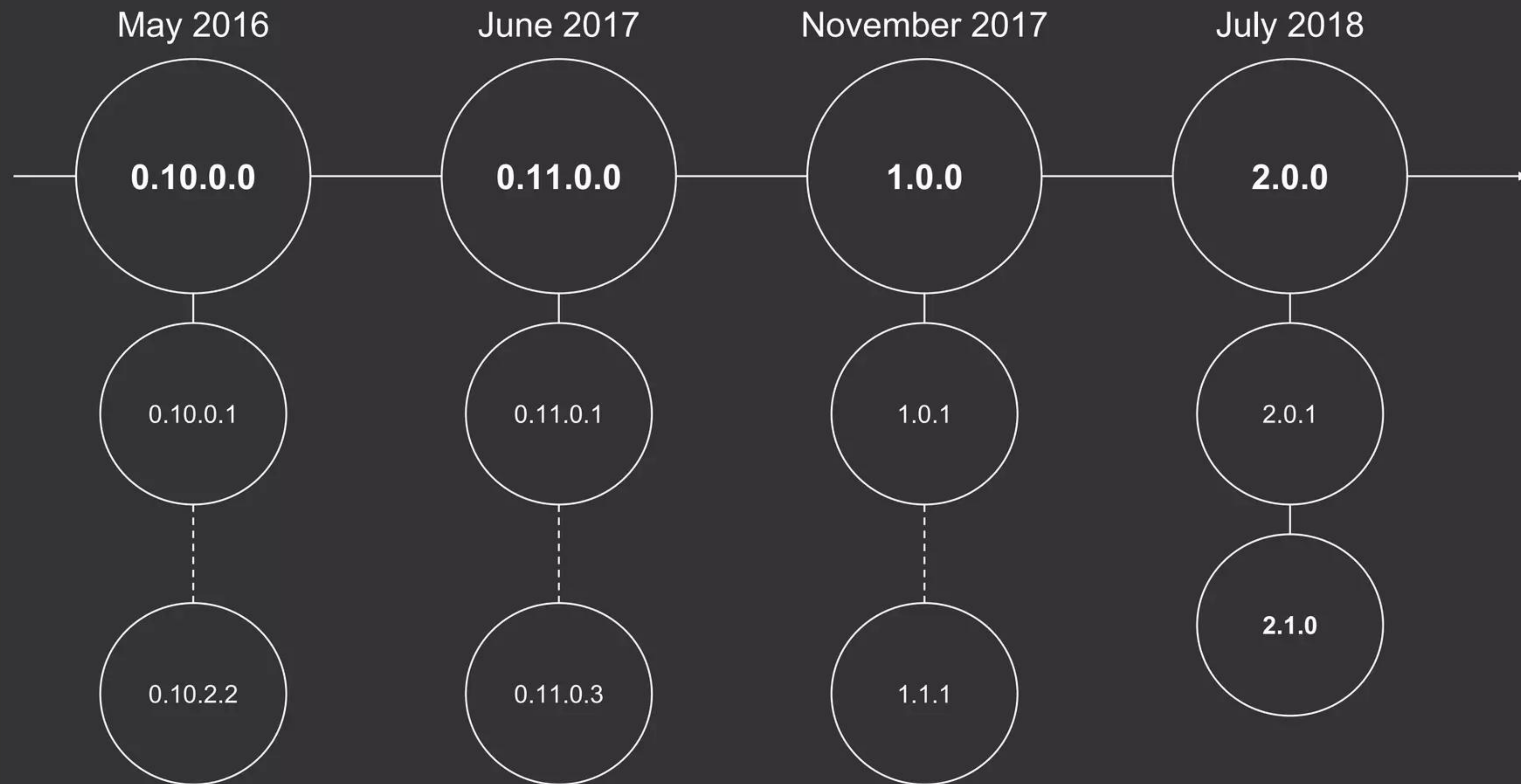


Apache Kafka: New Features That You Might Not Know About



Yaroslav Tkachenko
Software Architect at Activision

Apache Kafka Versions



0.11: New Message Format

Message Format v2

Record Batch:

```
...  
magic: 2  
...  
attributes:  
    ...  
    bit 4: isTransactional  
  
...  
producerId: int64  
producerEpoch: int16  
records: [Record]
```

Record:

```
...  
key: byte[]  
value: byte[]  
headers: [Header]
```

Header:

```
...  
headerKey: String  
value: byte[]
```

0.11: Headers

Message Headers

```
public interface Header {  
    String key();  
    byte[] value();  
}
```

```
List<Header> headers = Arrays.asList(  
    new RecordHeader("hkey1", "hvalue1".getBytes()),  
    new RecordHeader("hkey2", "hvalue2".getBytes())  
);  
  
new ProducerRecord<>("topic", 0, "key", "value", headers);
```

Message Headers

Pros

- No need to deserialize the whole message payload for routing / filtering use-cases

Cons

- Harder to save the headers together with the payload when archiving, persisting to data stores or integrating with 3rd party systems

Message Headers

 confluentinc / **confluent-kafka-python**

 Watch 135  Star 1,033  Fork 260

 Code  Issues 56  Pull requests 11  Projects 0  Wiki  Insights

message header support #299

 **Merged** edenhill merged 17 commits into confluentinc:master from johnistan:jdennison/message_header_support on Feb 16, 2018

 Conversation 50  Commits 17  Checks 0  Files changed 7 +250 -12



johnistan commented on Jan 16, 2018

Contributor

Initial stab at closing #287.

This implements the tuple idea mentioned in #287. Each header is a list of tuples [(header_key, header_value)...] This was the simplest to implement and I think will cover most use cases. For unique header_keys and user can just `dict(header_list)` for python 3.6 or `OrderedDict(header_tuple)` for before python 3.5.

This is still a WIP. Need to:

- throw an error on incorrect version if librdkafka
- better testing around malformed headers and incorrect types

Some questions:

- Should the client perform any validation on the shape of the headers or just trust the user.
- Right now the `producer.produce()` only accepts list. We can expand to support any iter that contains key, value pairs.

Reviewers

 edenhill 

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

 Subscribe

0.11: Transactions

Transactions

- Atomic writes to multiple Kafka topics and partitions
- Offset commits happen in the same transaction
- transactional.id + epoch for every producer
- Consumers must use “read_committed” isolation level for consuming only committed transactional data

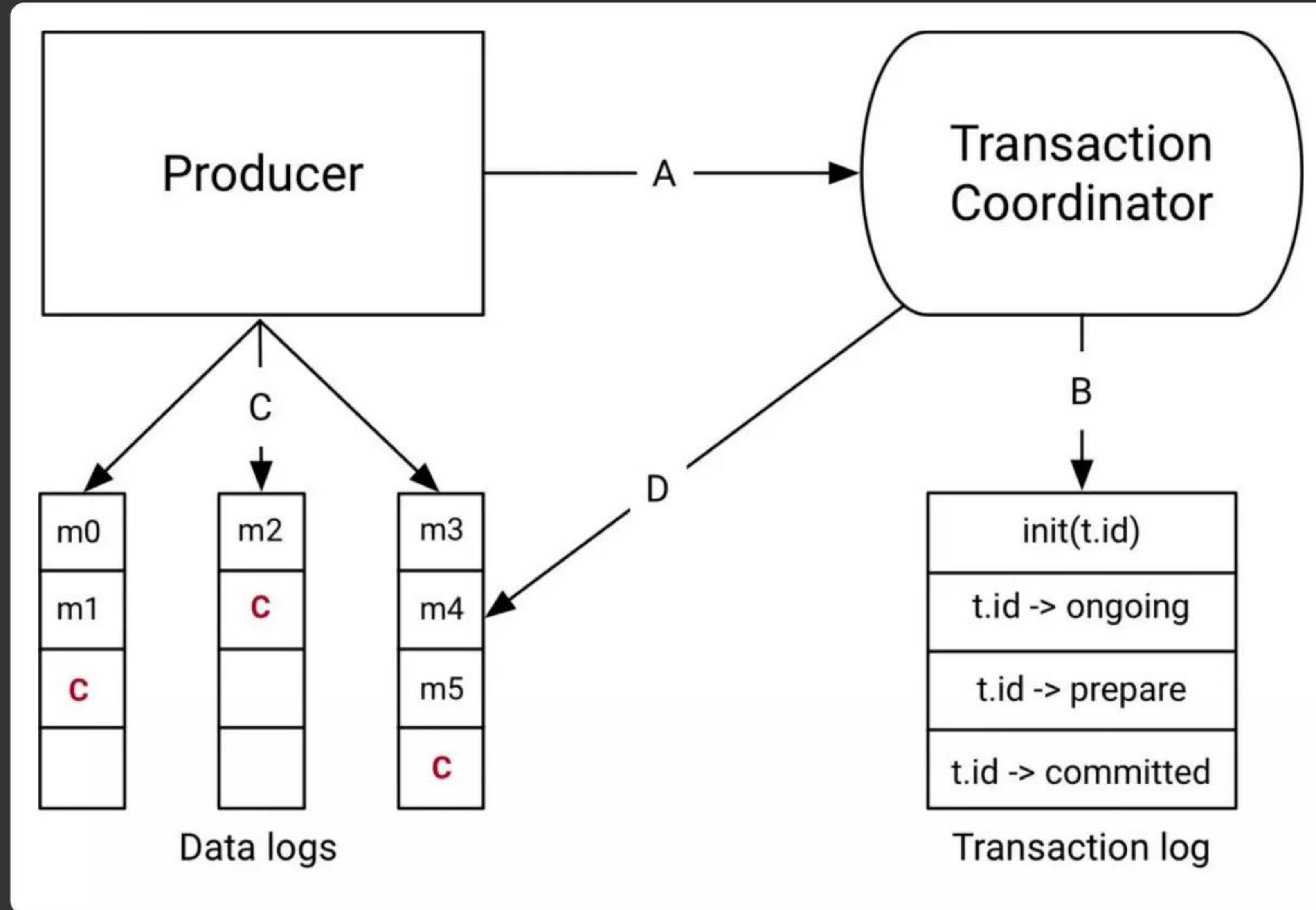
Transactions

```
KafkaProducer producer = ...  
producer.initTransactions();
```

```
KafkaConsumer consumer = ...  
consumer.subscribe("inputTopic");
```

```
ConsumerRecords records = consumer.poll(Long.MAX_VALUE);  
try {  
    producer.beginTransaction();  
    for (ConsumerRecord record : records) {  
        producer.send(processAndProduceRecord("outputTopic", record));  
    }  
    producer.sendOffsetsToTransaction(currentOffsets(consumer), groupId);  
    producer.commitTransaction();  
} catch (Exception e) {  
    producer.abortTransaction();  
}
```

Transactions



Transactions

In practice, for a producer producing 1KB records at maximum throughput, committing messages every 100ms results in only a 3% degradation in throughput.

<https://www.confluent.io/blog/transactions-apache-kafka/>

0.11: Exactly-Once Delivery

Exactly-Once: Why
is it so Hard?

Delivery Guarantees

At most once

- May or may not be received
- No duplicates
- Probably missing data

At least once

- Delivery guaranteed
- Possible duplicates
- No missing data

Exactly once

- Delivery guaranteed
- No duplicates
- No missing data

Transactions

Transactions API

Atomic writes and reads

Idempotence

Idempotent producer writes

Idempotence

- Unique producer ID is assigned to each producer
- Monotonically increasing sequence number is generated for every topic/partition write
- Broker persists and validates sequence numbers:
 - lower number → duplicate, **reject**
 - higher number → out-of-sequence error, **reject**
 - exactly one greater than the last → **allow**

Enabling Exactly-Once in Kafka Streams?

Just set “processing.guarantee” to “exactly_once”. That’s it!

Don’t need to think about checkpointing and related challenges (like in some other frameworks...)

1.1: Controller Improvements

Controller Improvements

- One Controller per cluster
- Responsible for state management of partitions and replicas
- Communicates with Zookeeper

Before 1.1.0

Updating partition leaders one by one, sequentially during the controlled shutdown

Zookeeper Synchronous API is used during the controlled shutdown and controller failover

Controlled shutdown time: 6.5 minutes

After 1.1.0

Updating partition leaders in batches during the controlled shutdown

Zookeeper Asynchronous API is used during the controlled shutdown and controller failover

Controlled shutdown time: 3 seconds

2.0: Kafka Streams Improvements

Kafka Streams Improvements

- Message header support in the Processor API
- TopicNameExtractor for dynamic routing
- kafka-streams-testutil helper for unit-testing
- Scala wrapper for the Streams DSL

Thanks !

@sap1ens