Scalability using Consumer Groups [..Continued]

Consume messages with Group Id

bin/kafka-console-consumer.sh --topic kafka-workshop-partitions --bootstrap-server localhost:9092, localhost:9093 --group group1

Add another consumer to group

Start another consumer with same group id in another terminal

bin/kafka-console-consumer.sh --topic kafka-workshop-partitions --bootstrap-server localhost:9092, localhost:9093 --group group1

Observe that partitions are rebalanced among the consumers in group group1

bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092, localhost:9093 --group group1 --describe

Stop consuming messages from one consumer in the group

Stop one of the console consumer

Observe that partitions are rebalanced again to accommodate only one consumer in group group1

bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092, localhost:9093 --group group1 --describe

Stop consuming messages from all consumers in the group

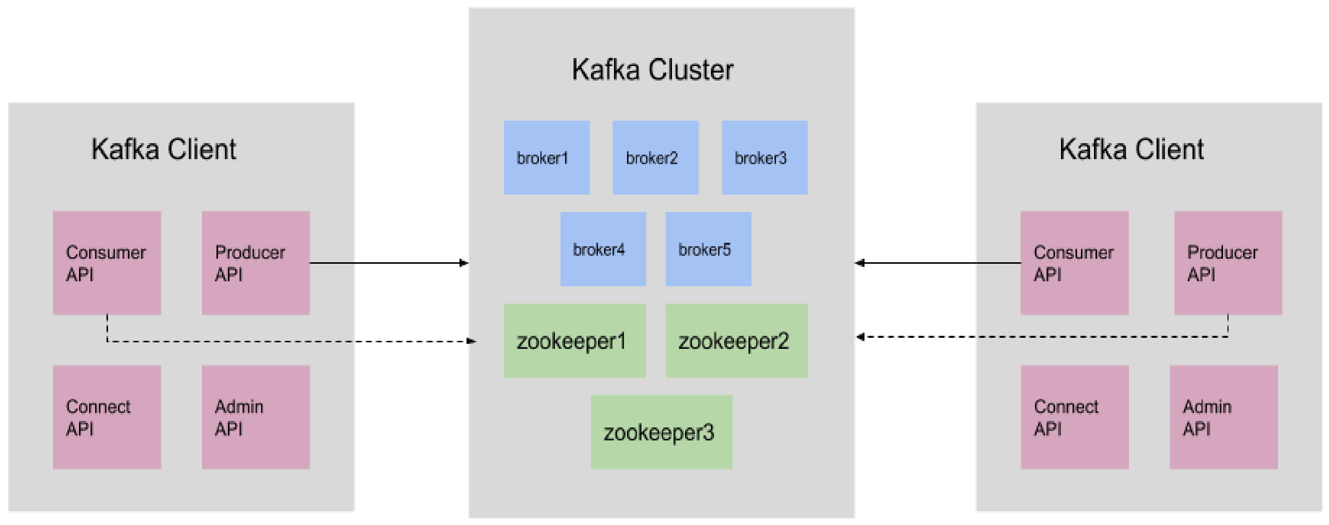
Continue producing messages

Consume messages with --from-beginning flag

Observe that all the **unprocessed** messages are consumed when --from-beginning flag is used.

bin/kafka-console-consumer.sh --topic kafka-workshop-partitions --bootstrap-server localhost:9092, localhost:9093 --group group1 --from-beginning

Apache Kafka Client



Producer API

Helps application to produce messages to a topic

ConsumerAPI

Helps application to consume messages from a topic

Connect API

Helps to continually pull from some source data system into Kafka or push from Kafka into some sink data system

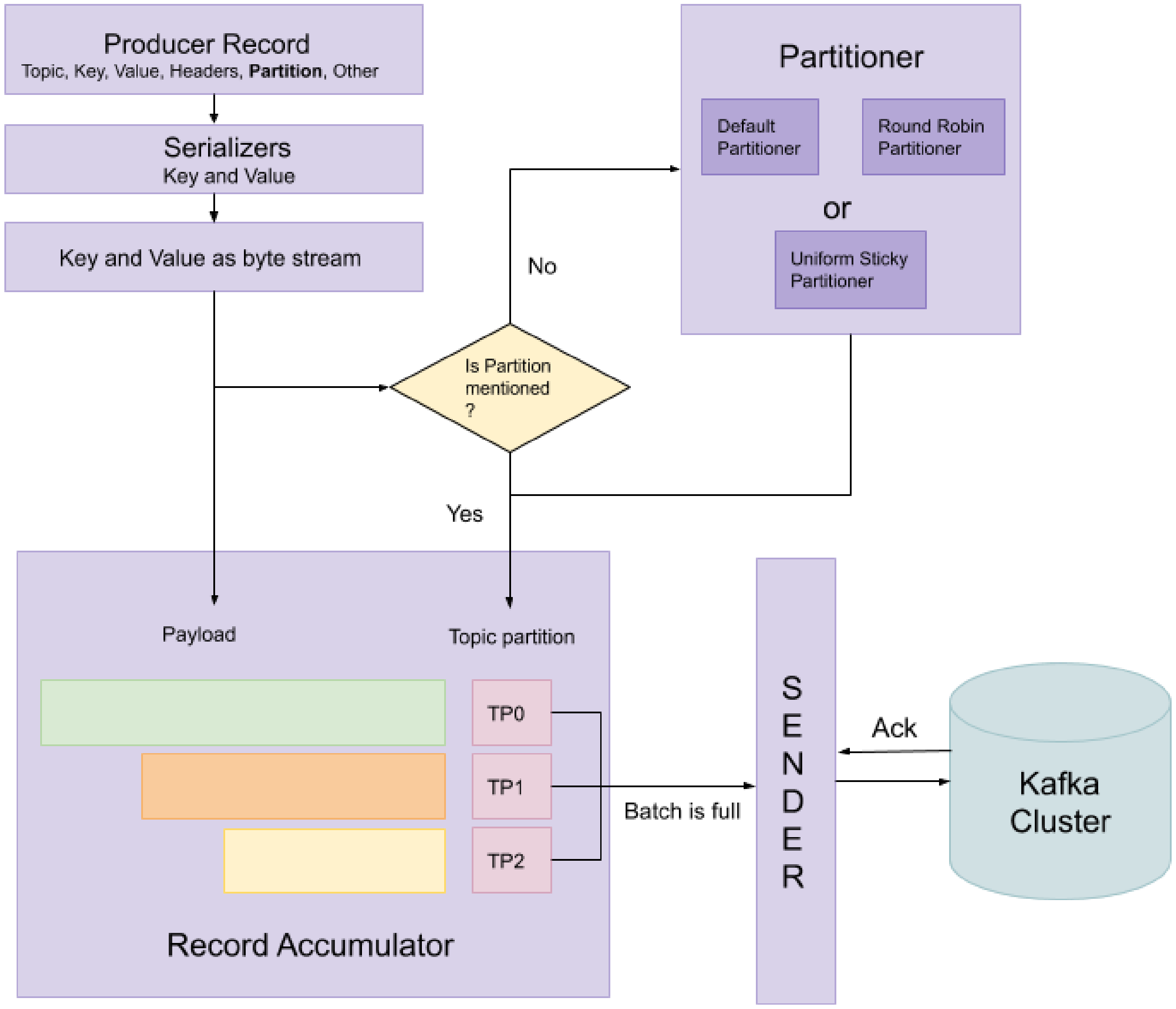
Admin API

Helps in managing and inspecting topics, brokers, acls, and other Kafka objects

Gradle dependency

compile group: 'org.apache.kafka', name: 'kafka-clients', version: '2.5.0'

Introduction to Producer API



Producer Record

Payload to be produced

Serializers

Converts producer Records's key and value into byte stream

Partitioner

Decides which TopicPartition batch this record goes into if partition is not mentioned in Producer record

Few Partitioner Strategies are

- UniformStickyPartitioner
- DefaultPartitioner
- RoundRobinPartitioner

Record Accumulator

Accumulates all the records till the batch is full

Sender

Sends the batch to Kafka cluster once the batch is full

Hands-On

Set up the below project

https://github.com/swathi-kurella/kafka-workshop-series.git

Prerequisites:

Language: Java

Build: Gradle

Create a sample topic

```
bin/kafka-topics.sh --create --topic kafka-workshop-eg --partitions 3 --bootstrap-server localhost:9092 --replication-factor 1
```

Run the sample Producer

Start SampleProducer class and provide desired messages to be produced in the input console

Start console consumer to validate

Observe that messages being produced to the topic

```
bin/kafka-console-consumer.sh --topic kafka-workshop-eg --bootstrap-server localhost:9092
```