By default, the producer does not care what partition a specific message is written to and **will balance messages over all partitions of a topic evenly**.

In some cases, the producer will direct messages to specific partitions. This is typically done using the message key and a partitioner that will generate a hash of the key and map it to a specific partition. This assures that all messages produced with a given key will get written to the same partition.

There are three primary methods of sending messages:

* Fire-and-forget - in which producer send a message to the server and don’t really care if it arrived succesfully or not. Most of the time, it will arrive successfully, since Kafka is highly available and the producer will retry sending messages automatically. However, some messages will get lost using this method.
* Synchronous Send - we send a message, the send() method returns a Future object and we use get() to wait on the future and see if the send() was successful or not.
* Asynchronous Send - we call the send() method with a callback function, which gets triggered when receive a response from the Kafka broker.

Also note that a single producer object can be used by multiple threads to send messages, or you can use multiple producers. You will probably want to start with one producer and one thread. If you need better throughput, you can add more threads that use the same producer

**SENDING A MESSAGE SYNCHRONOUSLY**

ProducerRecord<String, String> record =

new ProducerRecord<>("CustomerCountry", "Precision Products", "France");

producer.send(record).get(); // wait on the response

**Kafka Producer Exception and Errors:**

KafkaProducer has two types of errors.

1. Retriable errors Ex: Broker is not available. Try alternate broker
2. Non-Retriable errors Ex: Message size too large.

**SENDING MESSAGE ASYNCHRONOUSLY**

In most cases, producer really don’t need a reply - Kafka sends back the topic, partition and offset of the record after it was written and this information is usually not required by the sending app. In order to send messages asynchronously and still to handle error scenarios, the Producer supports adding a callback when sending a record.

|  |
| --- |
| private class DemoProducerCallback implements Callback { |
| @Override |
| public void onCompletion(RecordMetadata recordMetadata, Exception e) { |
| if (e != null) { |
| e.printStackTrace(); |
| } |
| } |
| } |
|  |
| ProducerRecord<String, String> record = |
| new ProducerRecord<>("CustomerCountry", "Biomedical Materials", "USA"); |
| producer.send(record, new DemoProducerCallback()); |

**How to connect to multiple brokers**

**kafkaProps.put("bootstrap.servers","broker1:9092,broker2:9092");**