1. Text

   Description automatically generated  
   
2. We understood Serializer and partition, however there is a lot going on inside the Producer.
3. Let’s see what happens next.
4. Diagram

   Description automatically generated
5. 
6. Once msg is serialized and assigned a partition#, the msg goes to sit in the buffer waiting to be transmitted.
7. KafkaProducer object consists of partition-wise buffer space that holds the records not yet have been sent.
8. Producer also runs a background I/O thread. That is responsible for turning these records into a requests and transferring to the Kafka Cluster.  
   Diagram

   Description automatically generated  
   e
9. **Why do we have this buffering**?
10. The buffering of the msgs has been designed to offer two advantages.
    1. **Asynchronous:**
    2. **Network Optimization**.
11. Buffer makes the KafkaProducer.send(ProducerRecord) method asynchronous.  
    What does it mean?  
    The send() will add the msg to the buffer and returns without blocking.  
    Those msgs are then transmitted by the background I/O thread.  
    This arrangement is quick convincing as your send() method is not delayed for the network operation.  
    Buffering also allows the background I/O thread to combine multiple msgs from the same buffer (one buffer contains msgs only belonging to a specific partition of a specific topic) and transmit them together as a single packet to achieve better throughput.
12. But there is a critical consideration here.  
    If the records are posted faster than they can be transmitted to the server, then this buffer space will be exhausted and your next msg will be blocked for a few milliseconds until the buffer is freed by the I/O thread.  
    If your I/O thread takes too time, then your KafkaProducer.send() method throws a TimeoutException.  
    Diagram

    Description automatically generated  
    When you see such TimeoutException, you may want to increase the Producer Memory.  
    Default is 32MB.  
    Diagram

    Description automatically generated