1.   
   
2. So, the problem of scalability of consumer is taken care of by the Consumer Group.  
   The fault-tolerance is also taken care of by the rebalancing within Consumer Group
3. However, we still have an open question.
4. Table

   Description automatically generated
5. 
6. We already learnt that a msg is identified within a partition by its offset.
7. Kafka also maintains two offset positions for each partition for consumer.
8. Text

   Description automatically generated with medium confidence
   1. **Current Offset**:
      1. Current Offset position of the consumer is the offset of the next record that will be given out to the consumer for the next poll().
      2. The initial value maybe null or unknown for a newly subscribed consumer.
      3. If we set the config 🡺 **auto-offset-reset=earliest**, then the Current Offset is set to the first record in the partition and will start giving you all the msgs from the beginning.  
         Text

         Description automatically generated with medium confidence

Otherwise the default value is latest 🡺 **auto-offset-reset=latest**which will send us only upcoming msgs after the consumer subscribe and ignore all earlier msgs.

* + 1. Every time we call poll() to fetch some msgs, these offset (Current, Committed) are automatically advanced.
    2. The **Current Offset** is persistent to the Consumer session.   
       If the consumer fails or restarts, the **Current Offset** is determined once again.  
       So, if you restart a consumer after a failure, you may start getting the records once again which were processed in the earlier session with the following setting.  
       **auto-offset-reset=earliest**To avoid such situation, Kafka also maintains a **committed offset position**.  
       Every time you **poll()**, the consumer will automatically commit the earlier **Current Offset** and sends some more records.  
       These new records are automatically committed by the next **poll()**This mechanism is known as auto-commit.  
       The **Committed Offset Position** is the last offset that has been stored securely at the Broker.  
       So, if a Consumer restarts after a failure or the partition is assigned to some other consumer in the same group, the **Committed Offset** is used to override the **Current Offset Position** for the Consumer.  
       Timeline

       Description automatically generated with low confidence

1. **Summary**:
   1. Committed Offset is securely stored with the Broker.  
      When a consumer restarts or the partition is assigned to some other consumer in the same group, **Committed Offset** is used to avoid the duplicate processing.  
      This all happens automatically but you can take the control in your hand and do it manually using **Commit APIs**.T
   2. The **Current Offset** is determined as latest or earliest based on the configuration (**auto-offset-reset=earliest or auto-offset-reset=latest**) only when there is no **Committed Offset** (When we start Kafka **Cluster Committed** Offset is unknown or null)