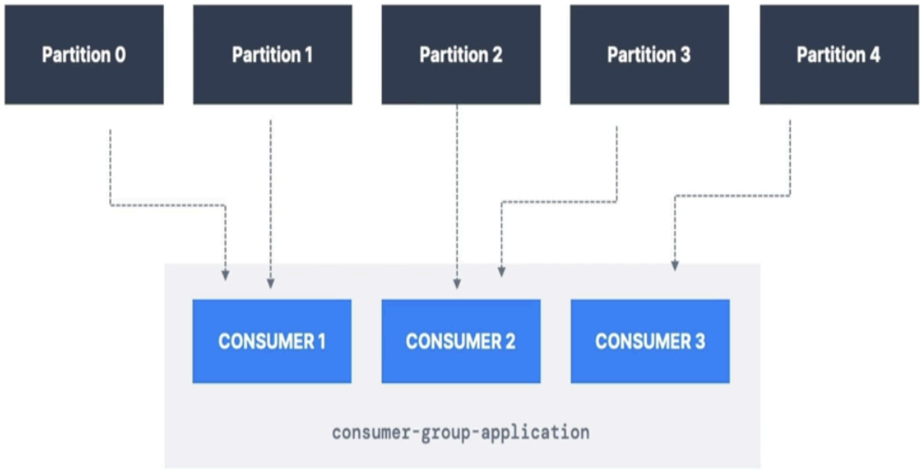
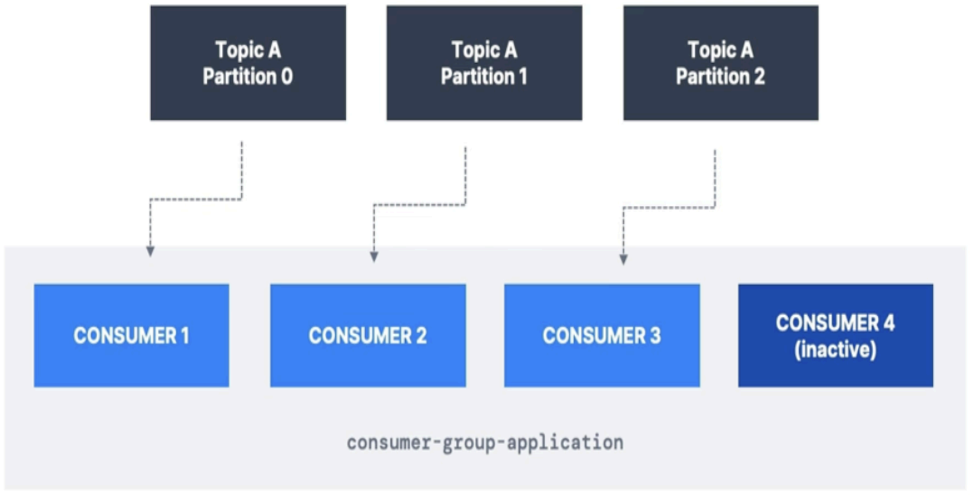
**Consumer Groups:**

- All the consumers in an application read data as a consumer group.
- Each consumer within a group reads from exclusive partitions.



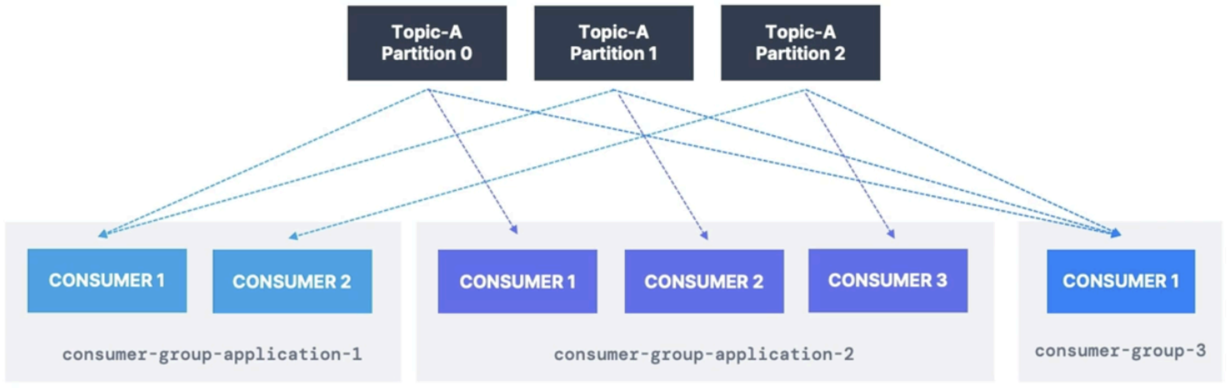
**What If too many consumers?**

- If you have more consumers than partitions, some consumers will be inactive.



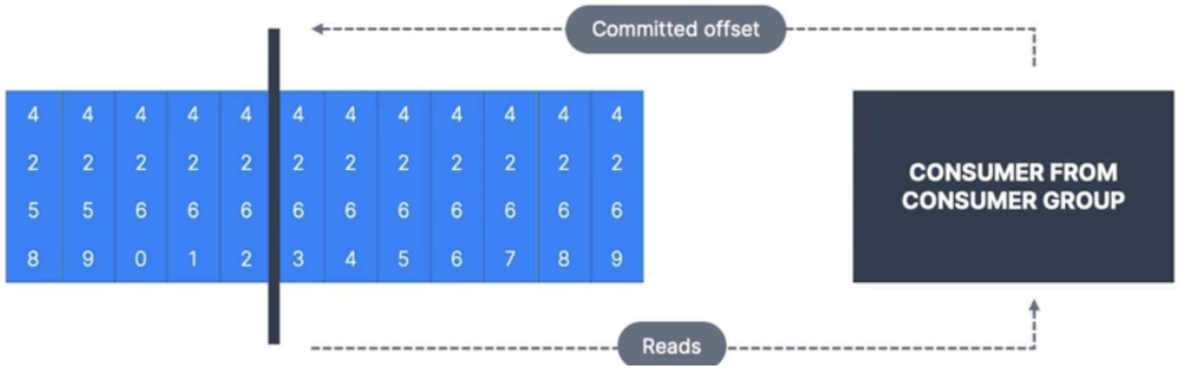
**Multiple Consumer Groups on one topic:**

- In Apache kafka, it is acceptable to have multiple consumer groups on the same topic.
- To create distinct consumer groups, use the consumer property **group.id**



**Consumer Offsets:**

- Kafka stores the offsets at which a consumer group has been reading from.
- The offsets committed are in a kafka topic named __consumer_offsets
- When a consumer in a group has processed data received from Kafka, it should be periodically committing the offsets (the Kafka will write to __consumer_offsets, not the group itself)
- If the consumer dies, it will be able to read back from where it left off, thanks to the committed consumer offsets!



**Delivery Semantics for consumers:**

- By default, java consumers will automatically commit offsets (at least once)
- There are 3 delivery semantics, if you choose to commit manually.
- At Least Once (Usually preferred):
    - Offsets are committed after the message is processed.
    - If the processing goes wrong, the message will be read again.
    - This can result in duplicate processing of messages. Make sure your processing is idempotent (i.e - processing again the messages won’t impact your systems)
- At Most Once:
    - Offsets are committed as soon as the message is received.
    - If the processing goes wrong, some messages will be lost (they won’t be read again).
- Exactly Once:
    - For Kafka => Kafka Workflows : use the Transactional API (easy with Kafka streams API).
    - For Kafka => External System Workflows : use and idempotent consumer.