**Kafka:-**

*V4 of kafka we are going to learn :*

*If u have* ***4*** *source system and 6 target system then each of them connect with each other we need* ***24*** *target systems.*

* *Each integration comes with difficulties around* 
  + *How the data is transported(TCP, Http, REST, JDBC, FTP)*
  + *Data format – how data is parsed(Binary, CSV, Json, Avro, protobuf).*
  + *How the data is shape and changed.*
* *Each source system will have increase load from the connections.*

Source System

Source System Source System

Source System

Source System

Systems

* Website events , Pricing data, Financials Transactions , user Interactions.
* **Target System are Databases, Notifications Engines etc.**

High performance

Netflix is using kafka to apply recommendations in real time while your are watching tv

**Uber** uses kafka to gather user, taxi and trip data in real time to compute and forecast demand, and compute surge pricing in real timing

***Kafka Topics :***

* *Topics a particular stream of data (Logs\_topic, purchases\_topic,) same we can have table in ”DB“.*
* *Like table in a databasee (Without all constraints)*
* *Support any kind of message format*
* *You Cannot query topic, user kafka producers to send the data and kafka Consumers to read the data*

***Partitions in Kafka :***

* *Topics are split into partitions (example 100 partitions)*
  + *Message with in each partition are in ordered*

*Partition id is going to increase if we write the data in partitions from 0 to n*

* *Kafka topics are immutable once data is written to a partitions, it cannot be changed.*

***Example :***

***T****ruck Gps data*

* ***Every truck send data of his location in 20 seconds to kafka and each msg will contain the truck Id nd the truck position***
* ***You can create a topic trucks\_gps that contain all the position of the trucks***
* ***We choose to create the topic of 10 partitions (arbitrary Number)***

*#Important things :*

1. *Once data is written to partition it can’t be change that is immutability*
2. *Data is kept only for limited time (default is one week – configurable)*
3. *Offset only have meaning for a specific partitions* 
   1. *Eg. Offset 3 in partition 0 doesn’t represent the same data as offset 3 in partition 1*
   2. *Offset are not re-used even if previous message have been deleted.*
4. *Order is guaranteed only with in the partition not across the partitions*
5. *You can have as many partition per topic as you want.*

*------------------------------------------------------------------------------*

***Producers:***

*Producers write the data to the topics (Which are made of partitions)*

*Producers => Sends Data => topic-a/ partition 0*

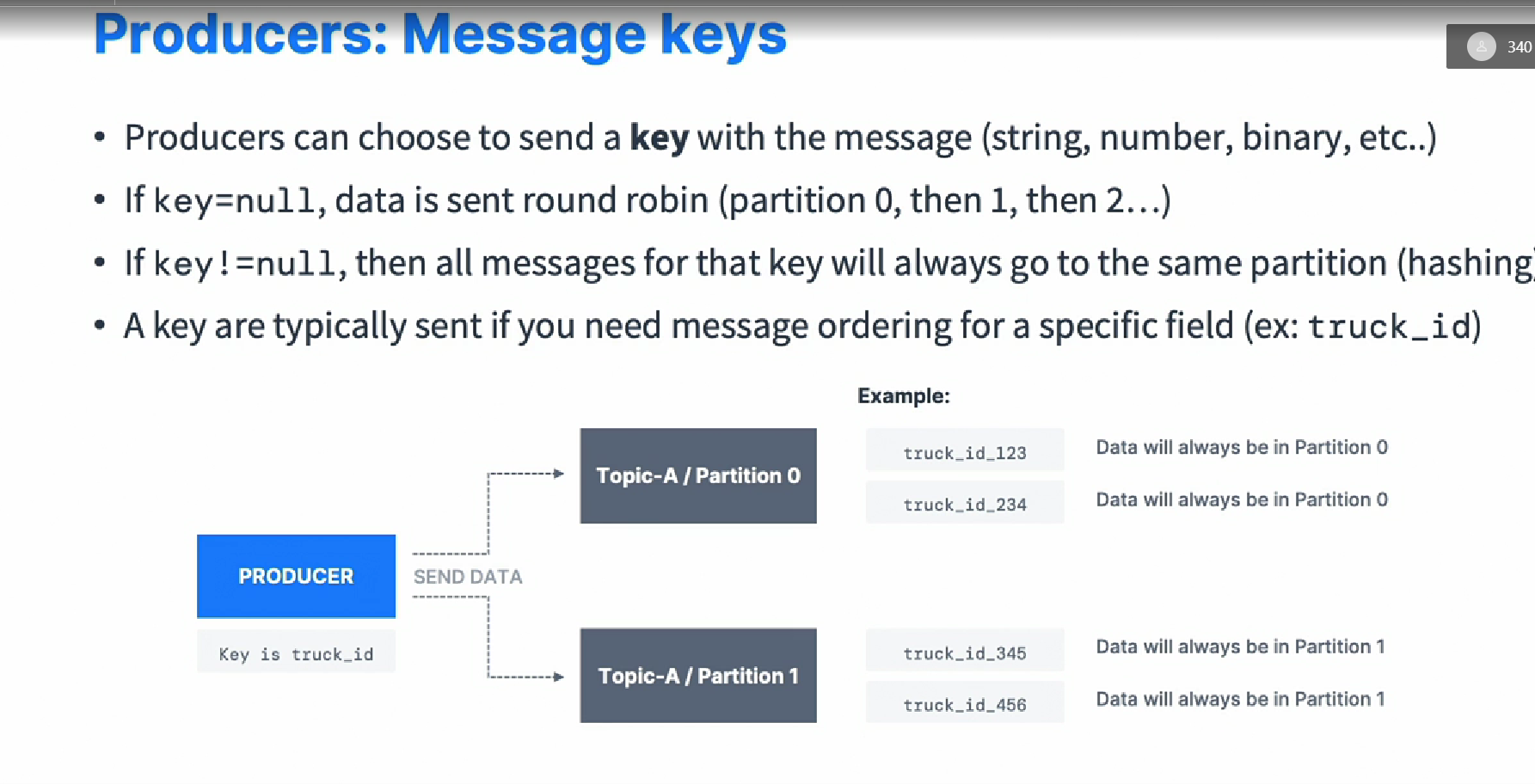
*Topic-b/ partition 1*

*Topic –c / partition 2*

**

*Producers know to which partition they have to write (and which Kafka broker has it)*

*In case Kafka broker failure producer will automatically recover it.*

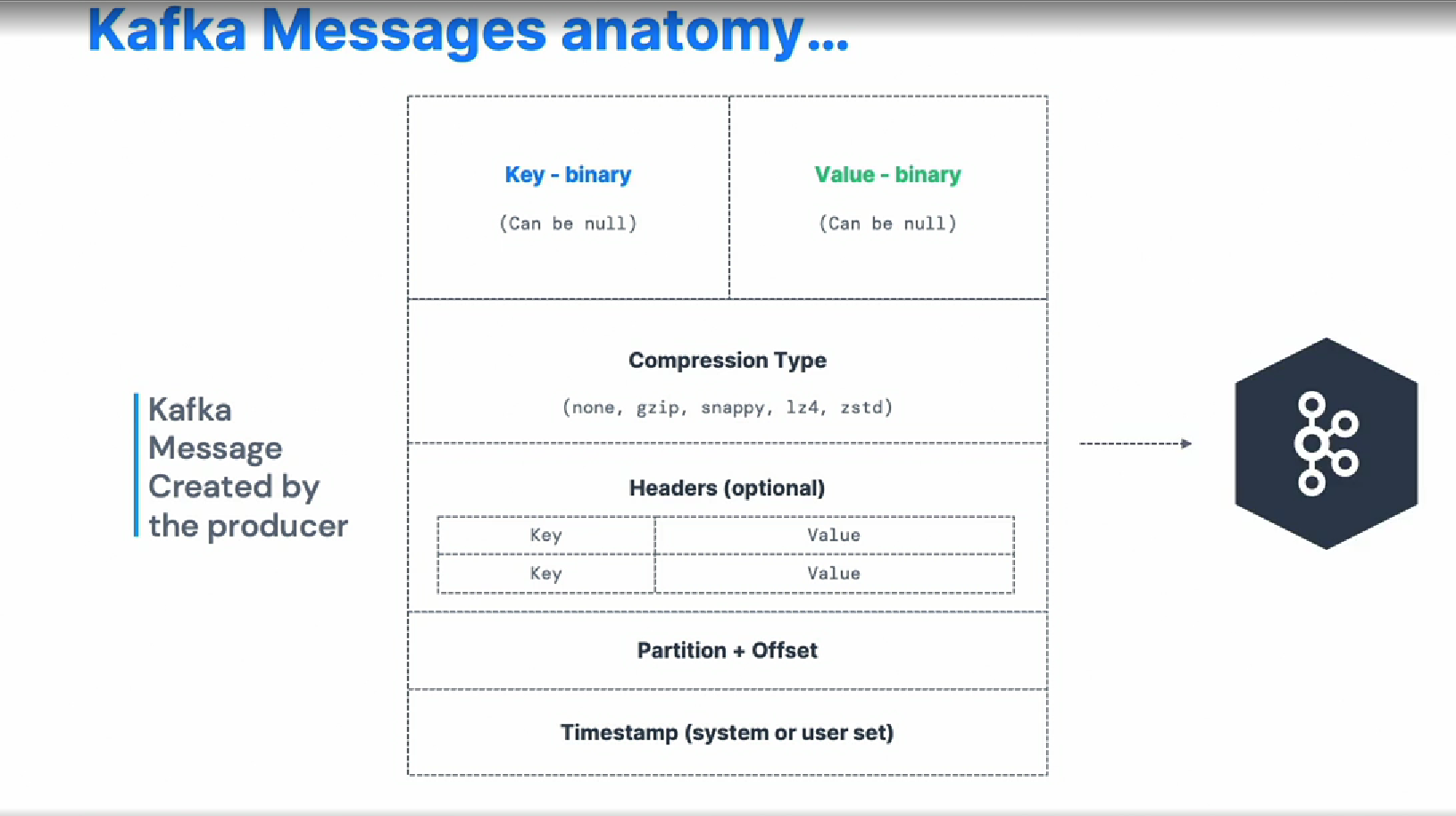
**

*If key != null the there is some value like truck\_id is the key*

*{  
“truck\_id”:”123”*

*}*

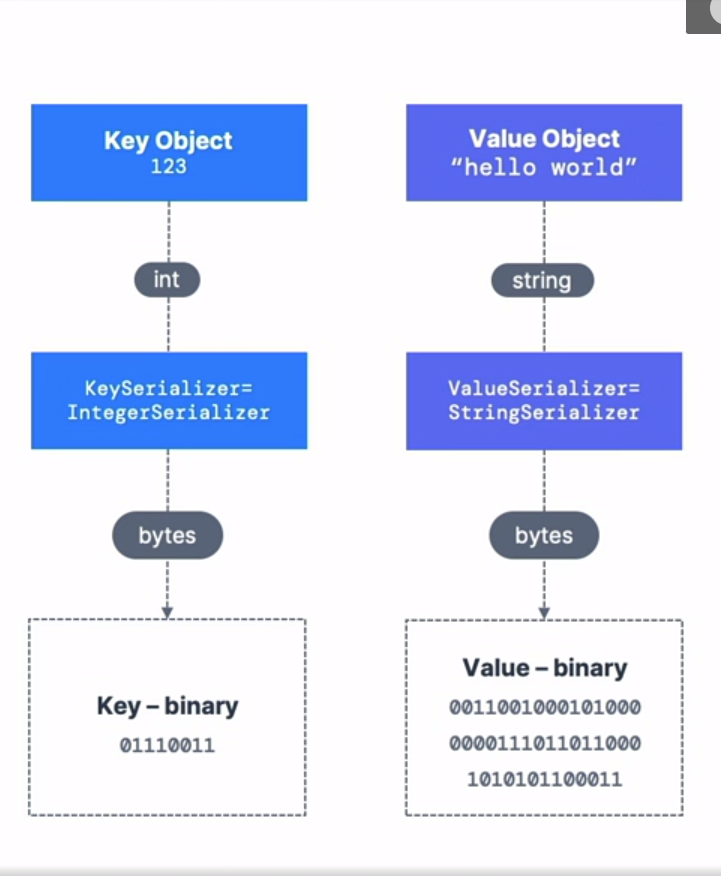
*We will call the hash() for the id value and check in which partition it will go!!*

**

*This the structure of kafka message that producer will create and send to kafka stream data !!.*

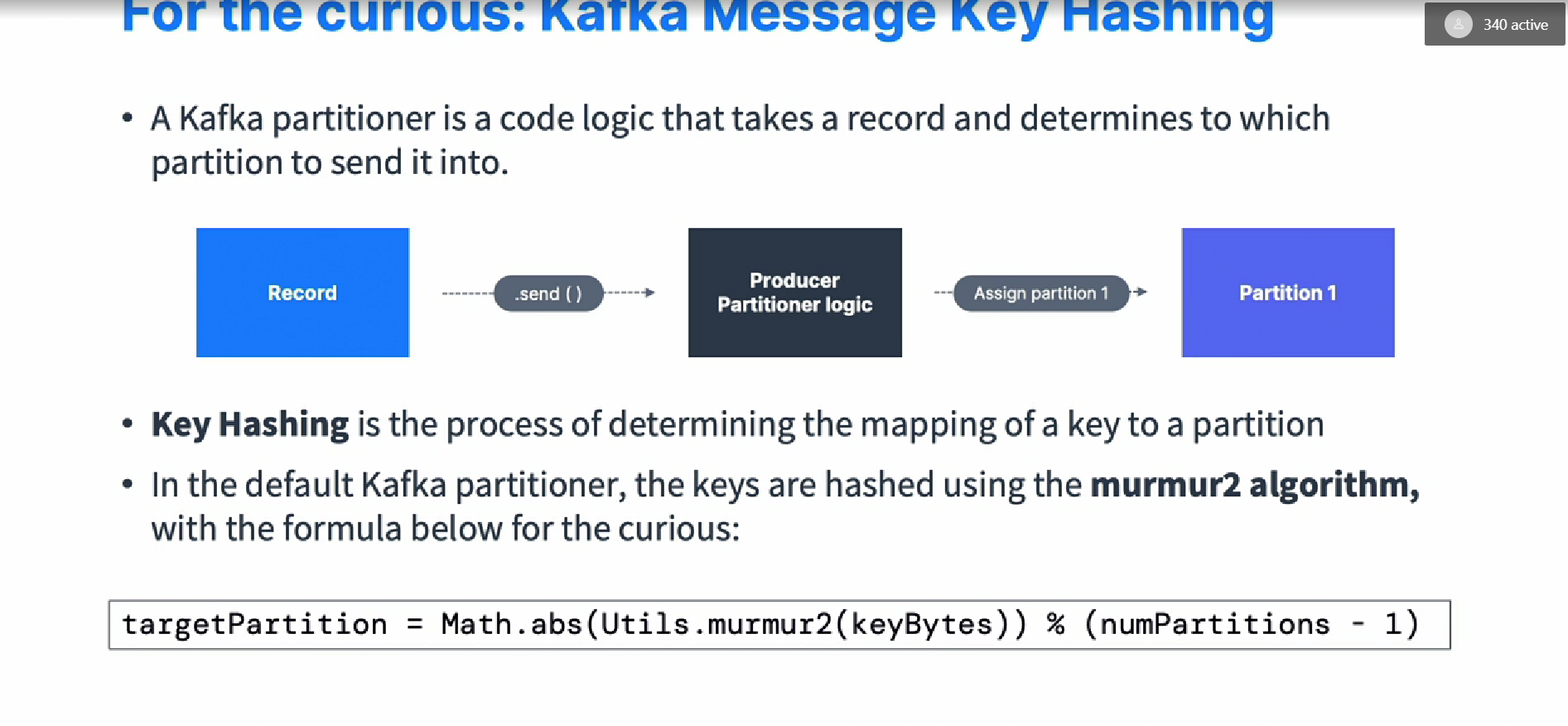
***Kafka Message Serializer:-***

* *Kafka only accept bytes as an input from producers and send bytes out as an output to consumers*
* *Message serialization means transforming data from one object to another*
* *They are used on the value and key*

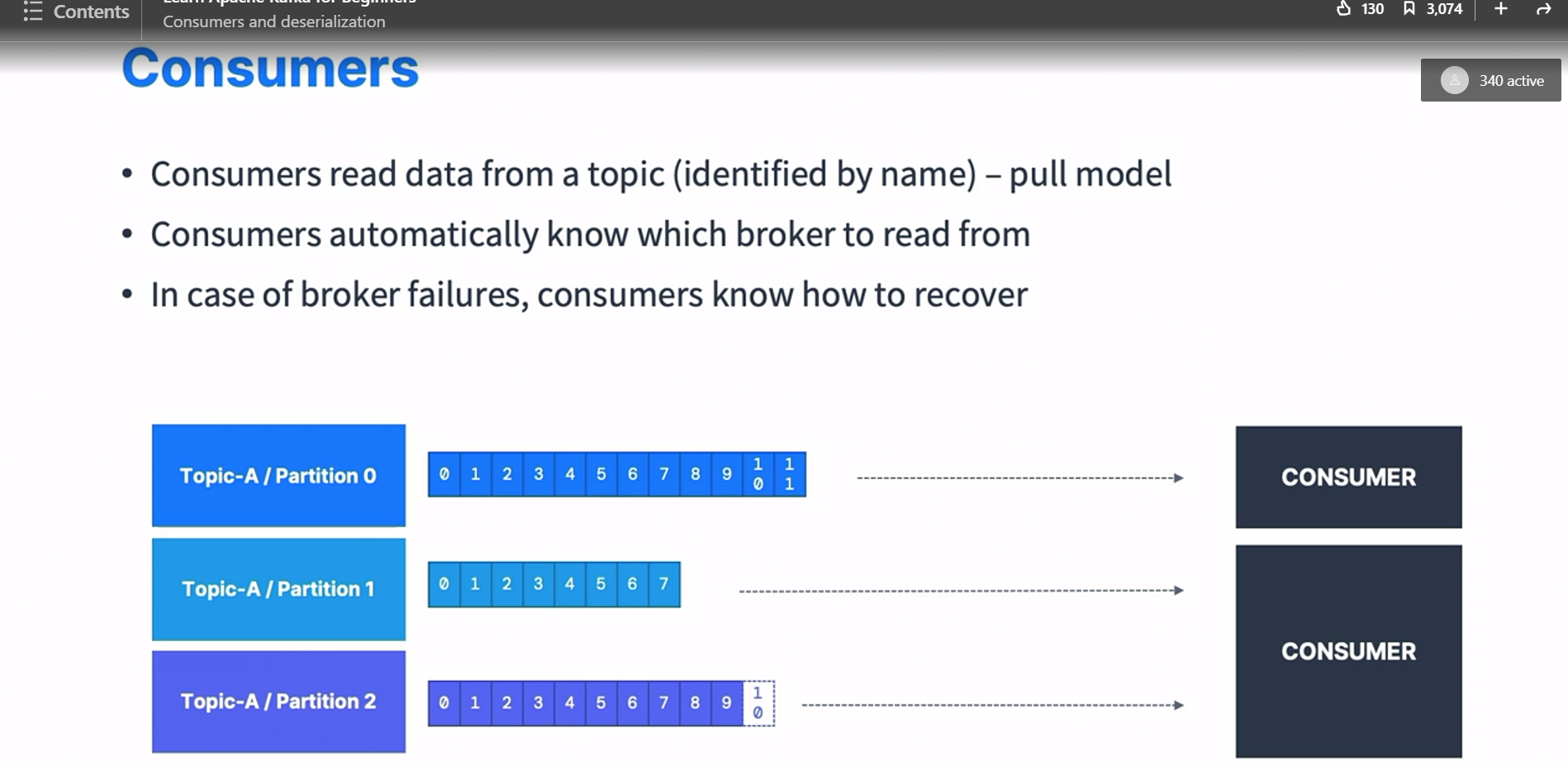
**

* *Kafka producer comes with default serialiazer*

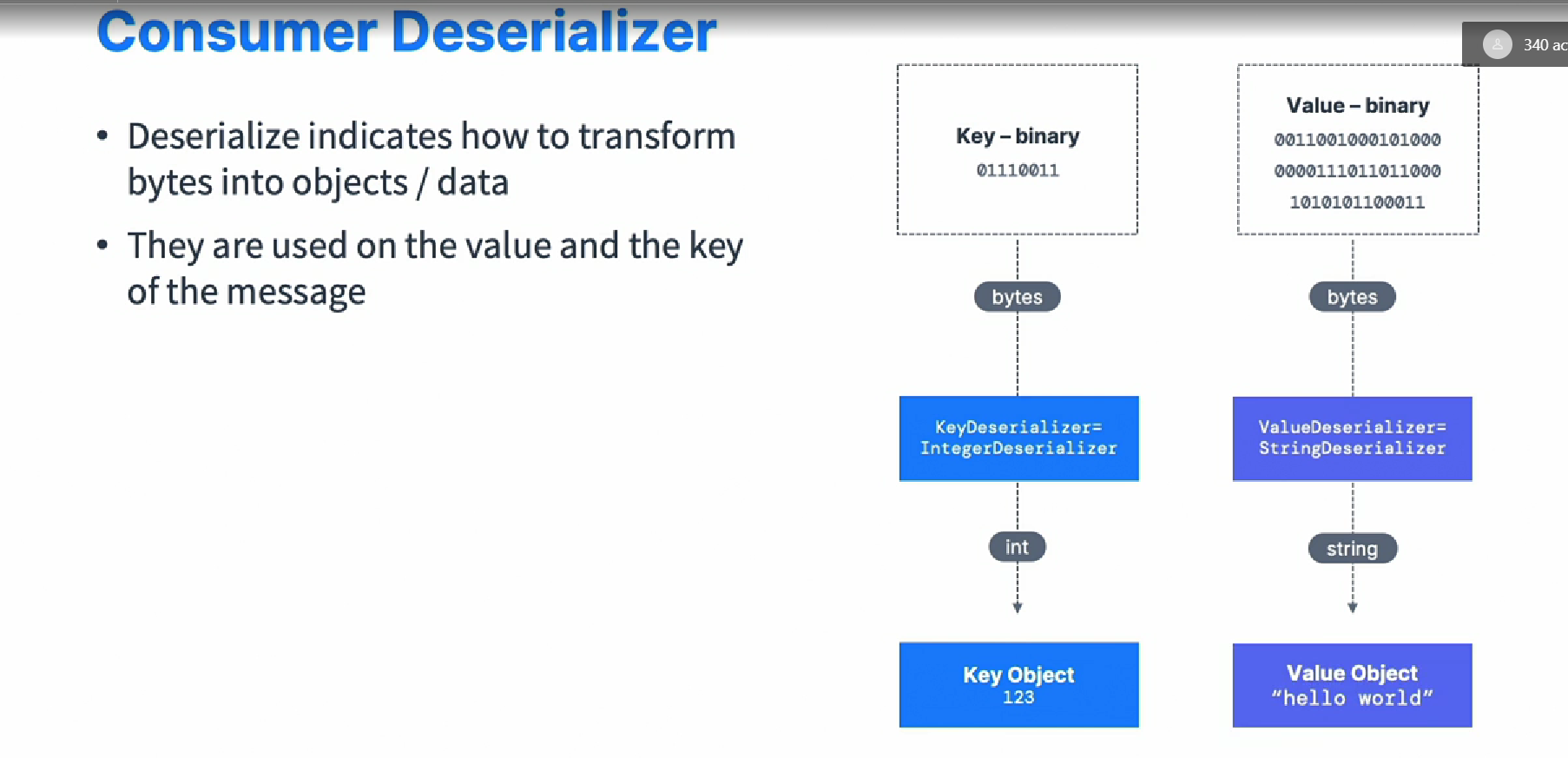
*#****Kafka Message Key Hashing:-***

**

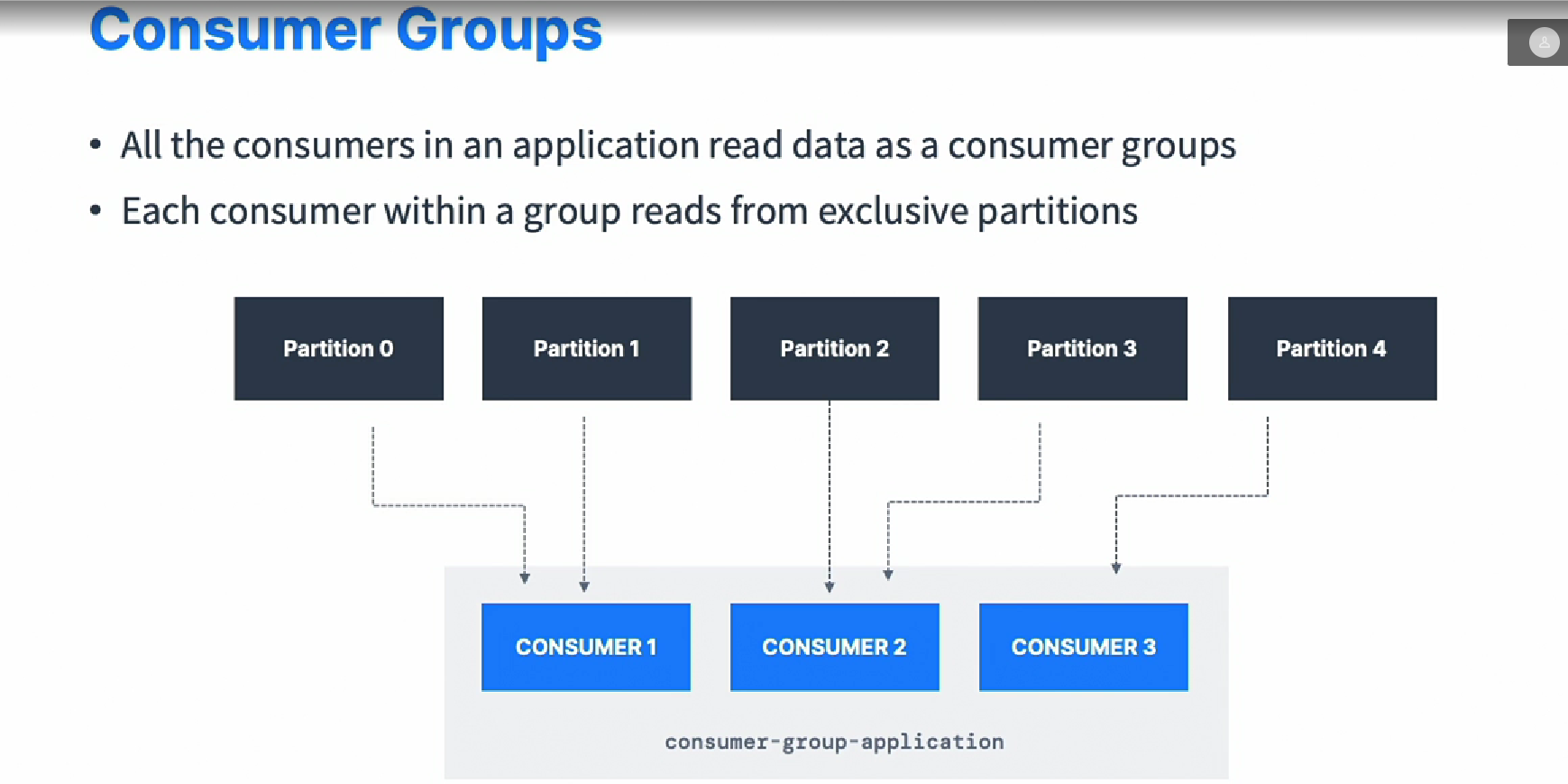
***Consumers:***

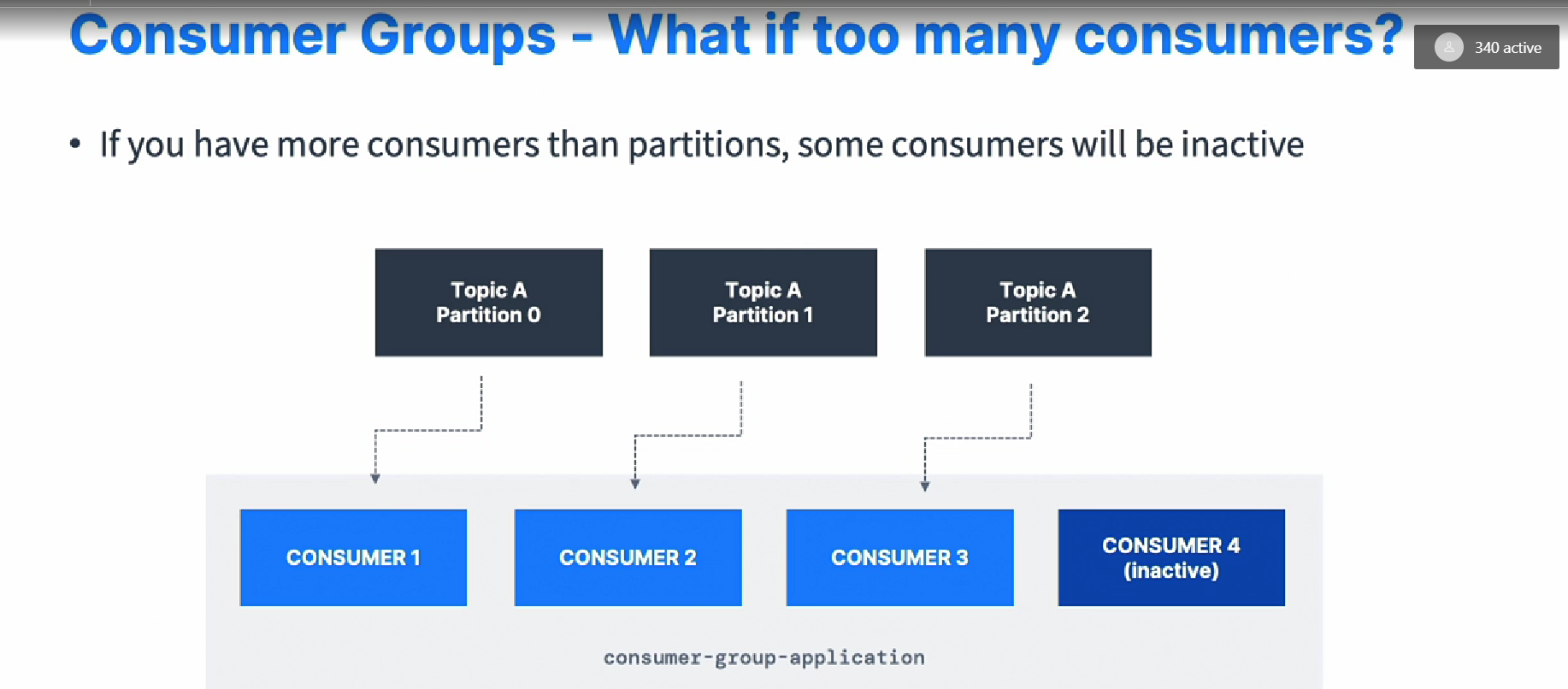
******

***Data is read in order from low to high offset with in each partitions .***

***Consumer Deserializer :  
***

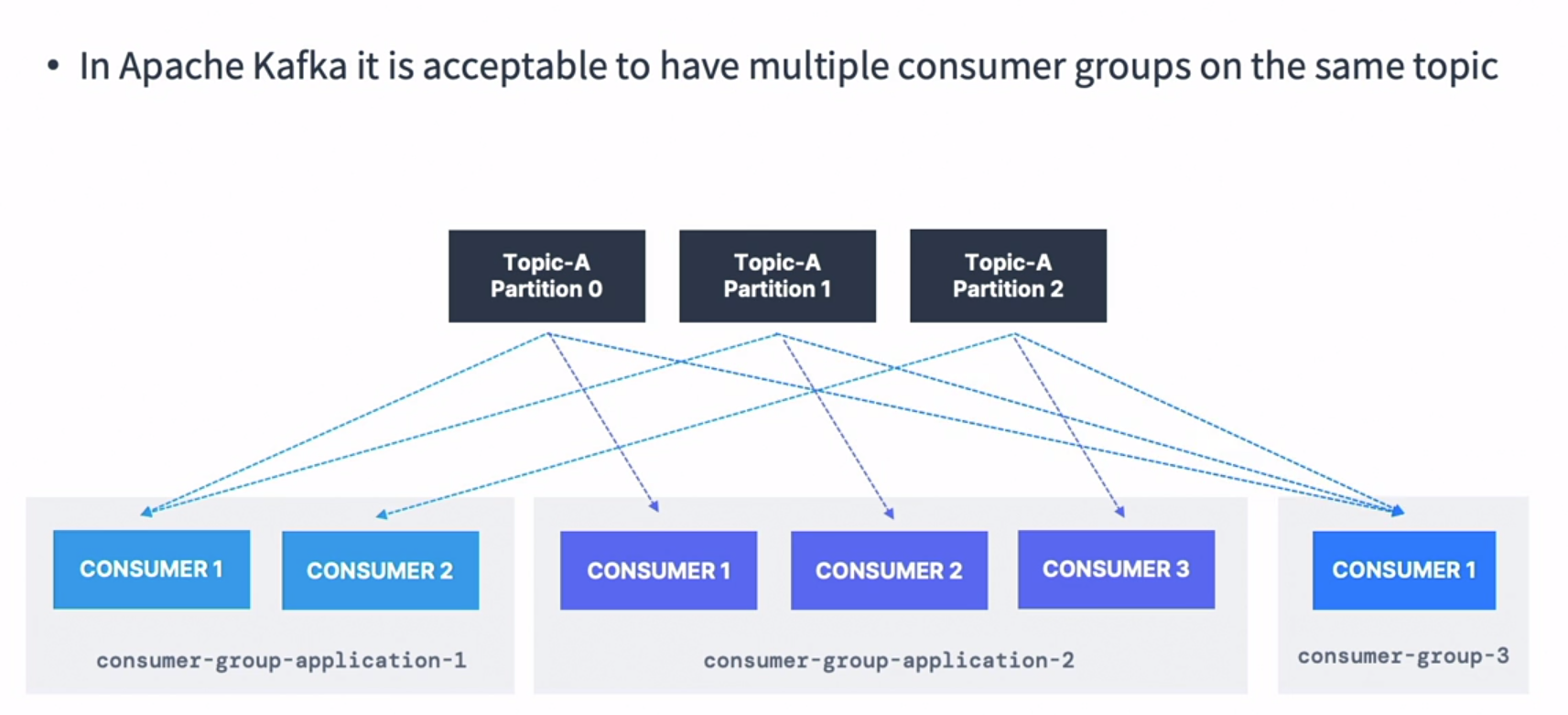
***The*** *Serial and deserial type must not be changed during the life cycle (create a new topic instead)*

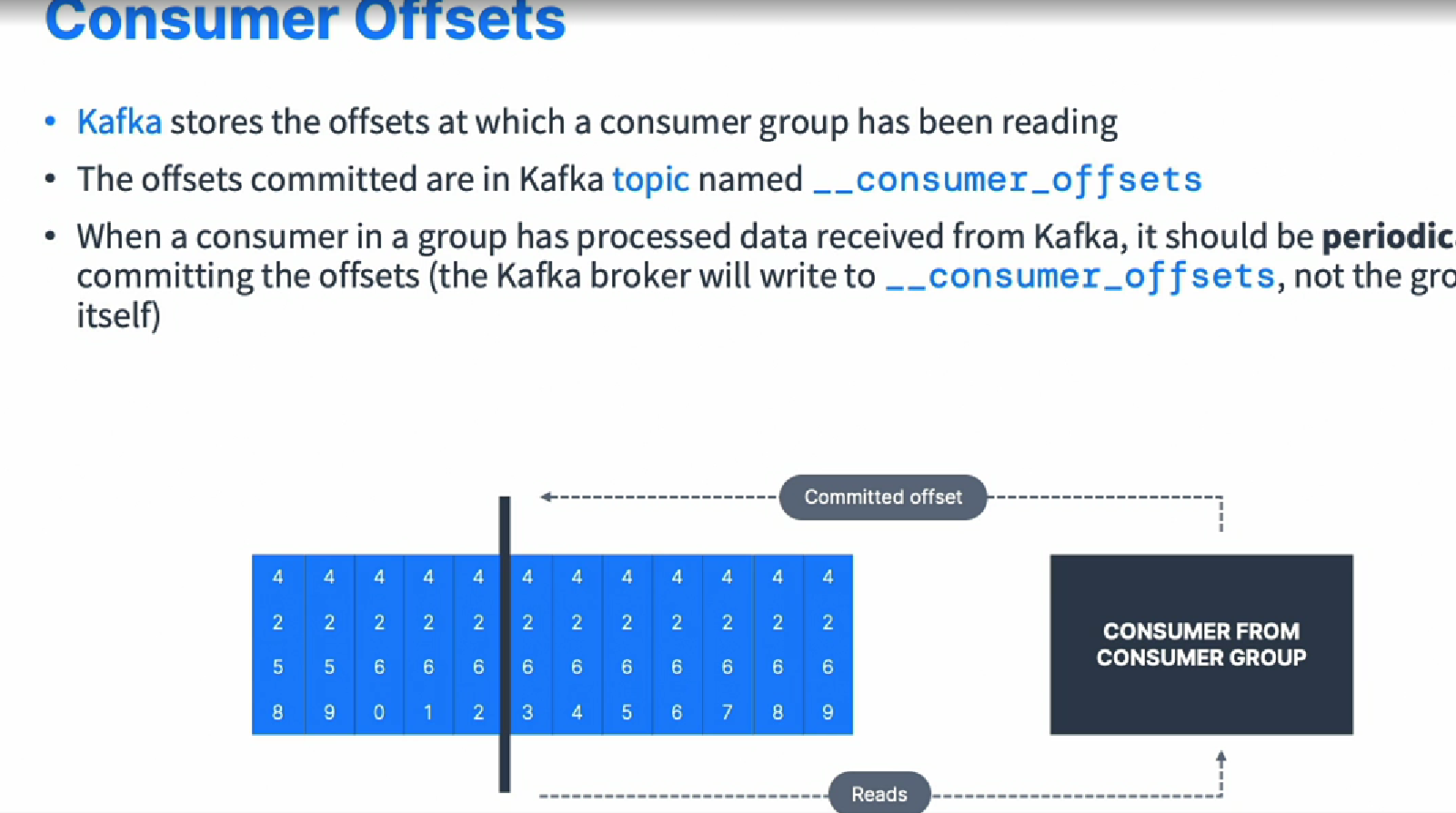
**

**

*# Multiple Consumers on one topic*

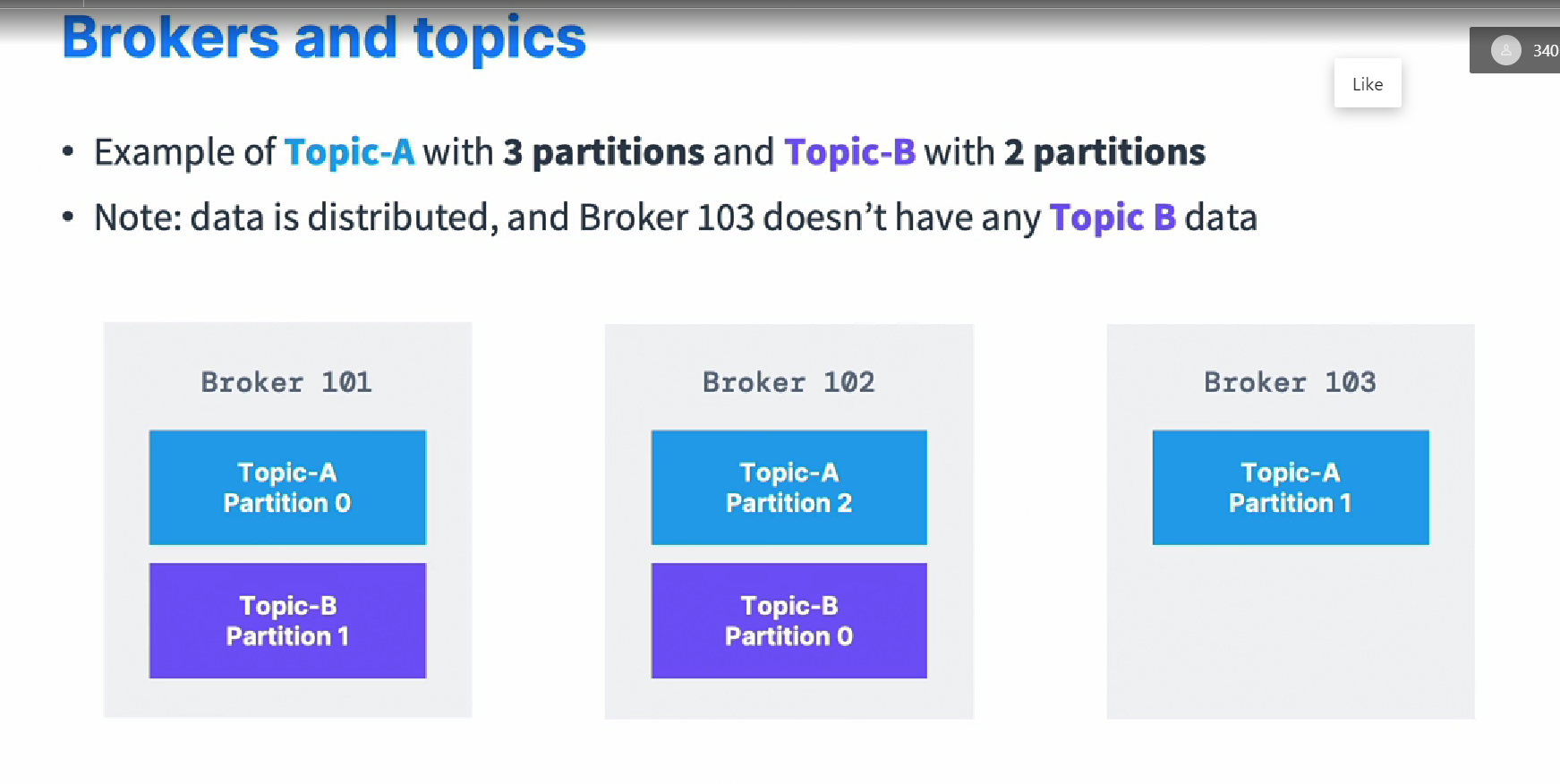
* *In apache Kafka it is acceptable to have multiple consumer groups on the same topic*
* *To create distinct consumer group use the consumer property* ***group.id***

**

**

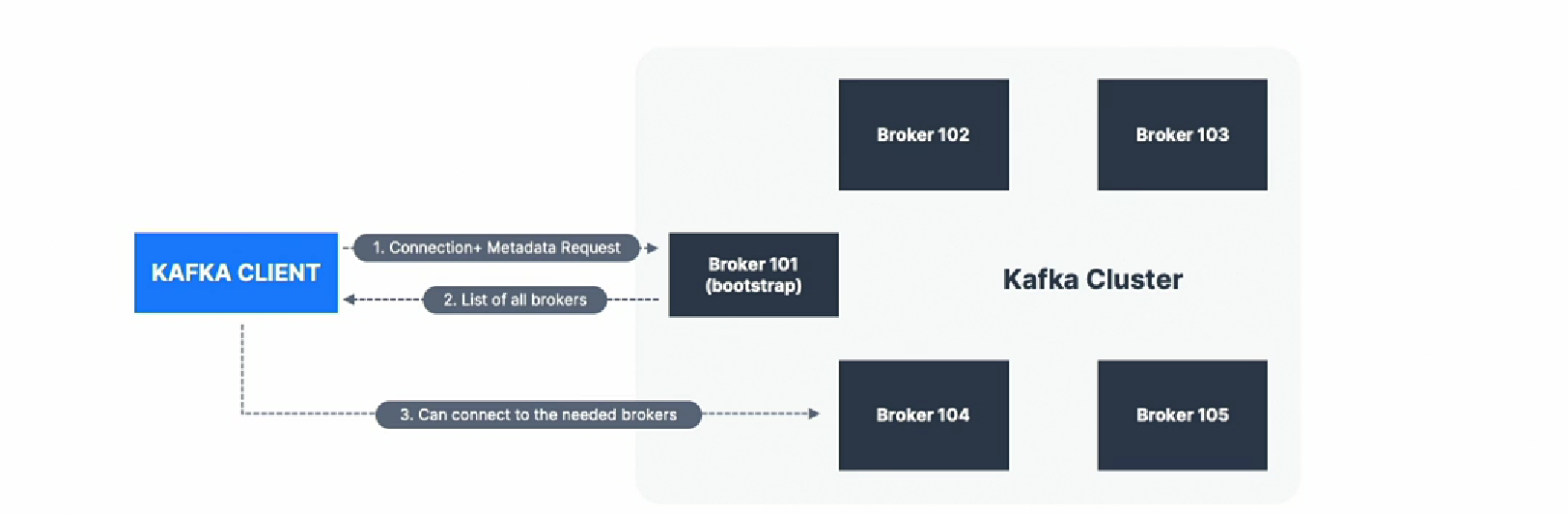
***#Kafka – Brokers***

* ***A*** *kafka cluster is composed of multiple brokers (Servers)*
* *Each Brokers is identified with it unique ID (integer).*
* *Each broker contains certain topics partitions.*
* *After connecting to any broker you will be connected to entire cluster (kafka clients have smart machines for that).*
* *A good number to start to have 3 brokers but some big clusters have 100 of brokers.*



***# Kafka Broker Discovery***

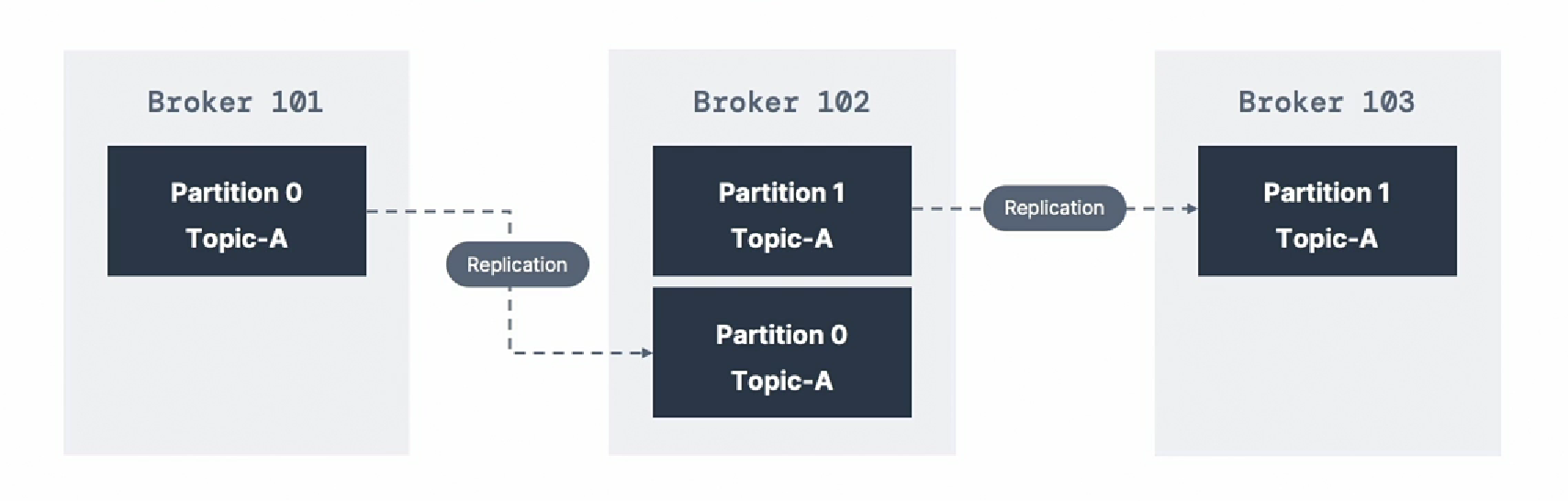
* *Every kafka broker is also called a “Bootstrap Server”.*
* *That means that you only need to connect to one broker and the Kafka client will know how to be connected to the entire cluster (smart clients).*
* *Every brokers knows about all brokers, topics and partitions (meta-data).*

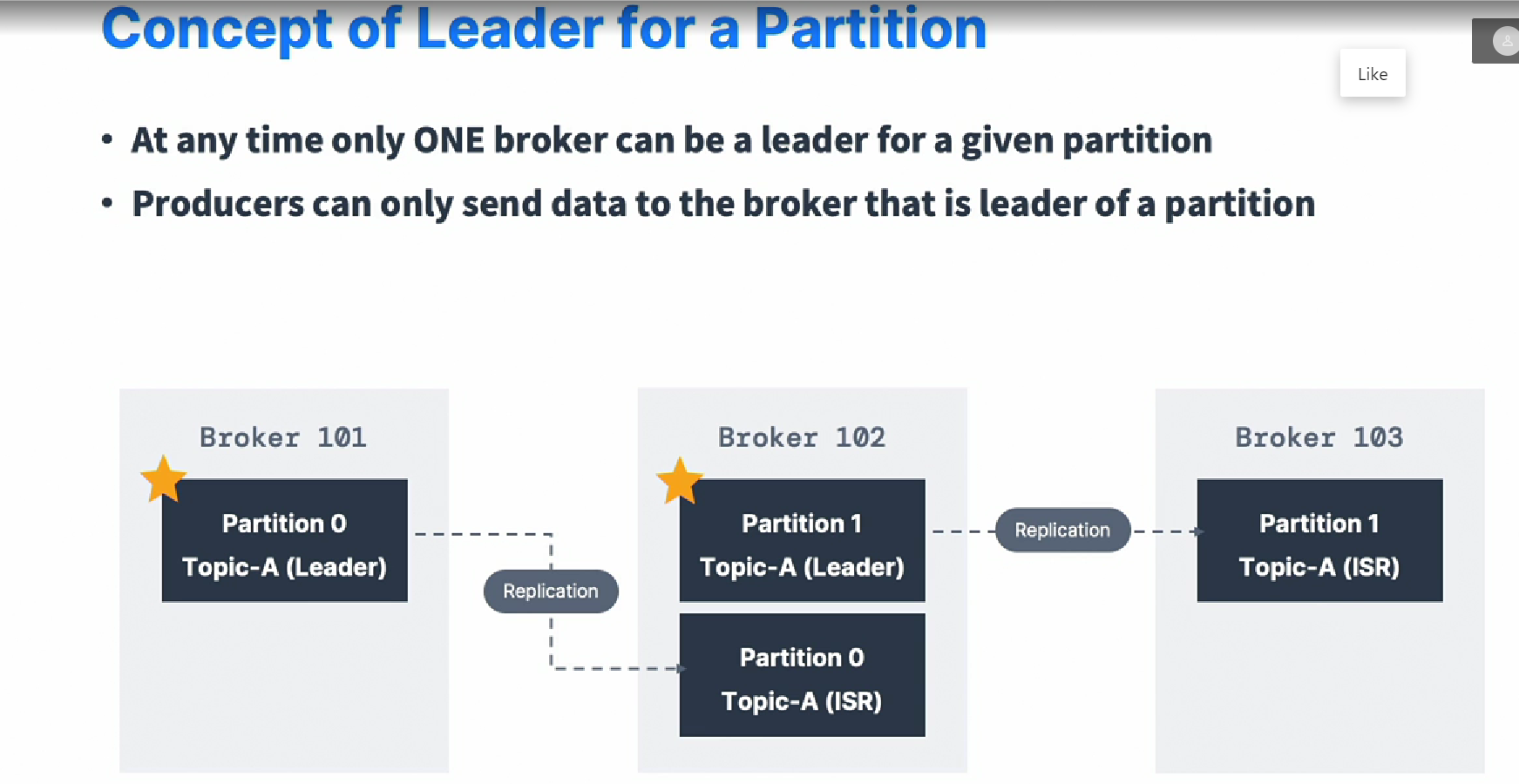
**

***#Topic – Replication – Factor***

***Topics should have a replication factor > 1 (Usually btw 2 and 3)***

* *The way if a broker is down, another broker can serve the data*
* *Example: Topic-A with 2 partitions and replications factor of 2.*

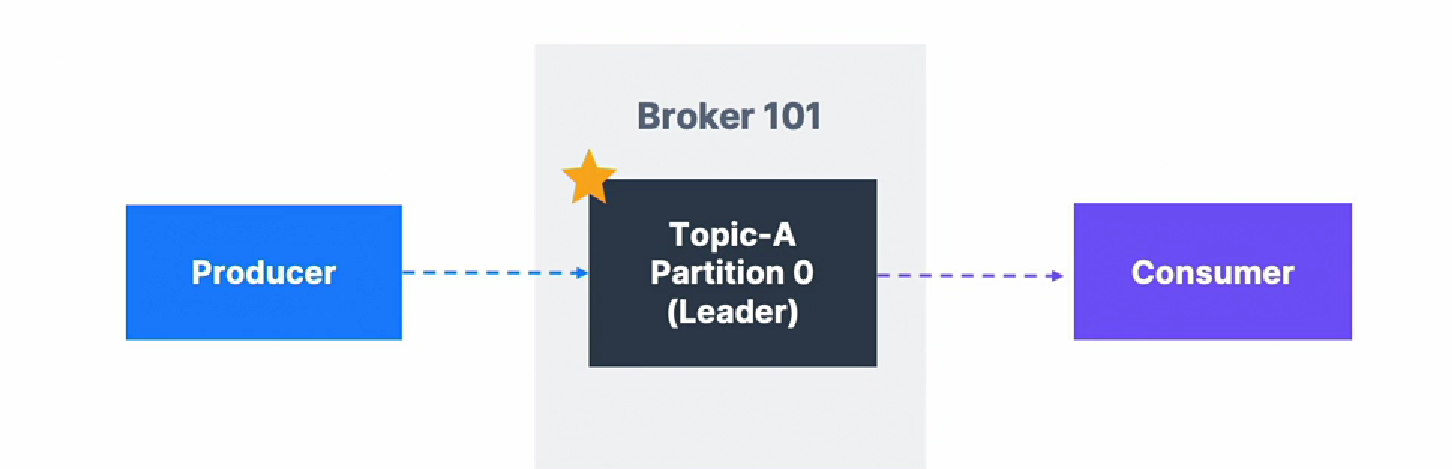
**

**

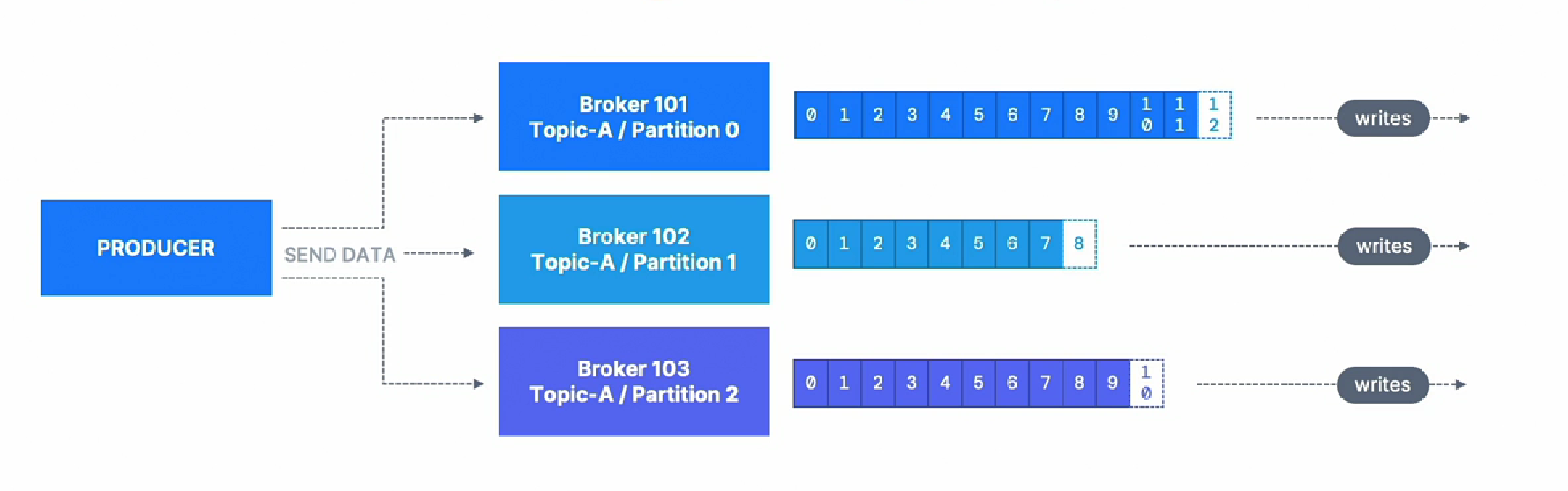
*ISR -> in sync Replica !!*

*#Default producer & consumer behaviour with leader*

* *Kafka producer can only write to the leader broker of a partitions*
* *Kafka Consumer by default will read from the leader broker for a partitions*

**

***#Producer – Acknowledgment (acks)***

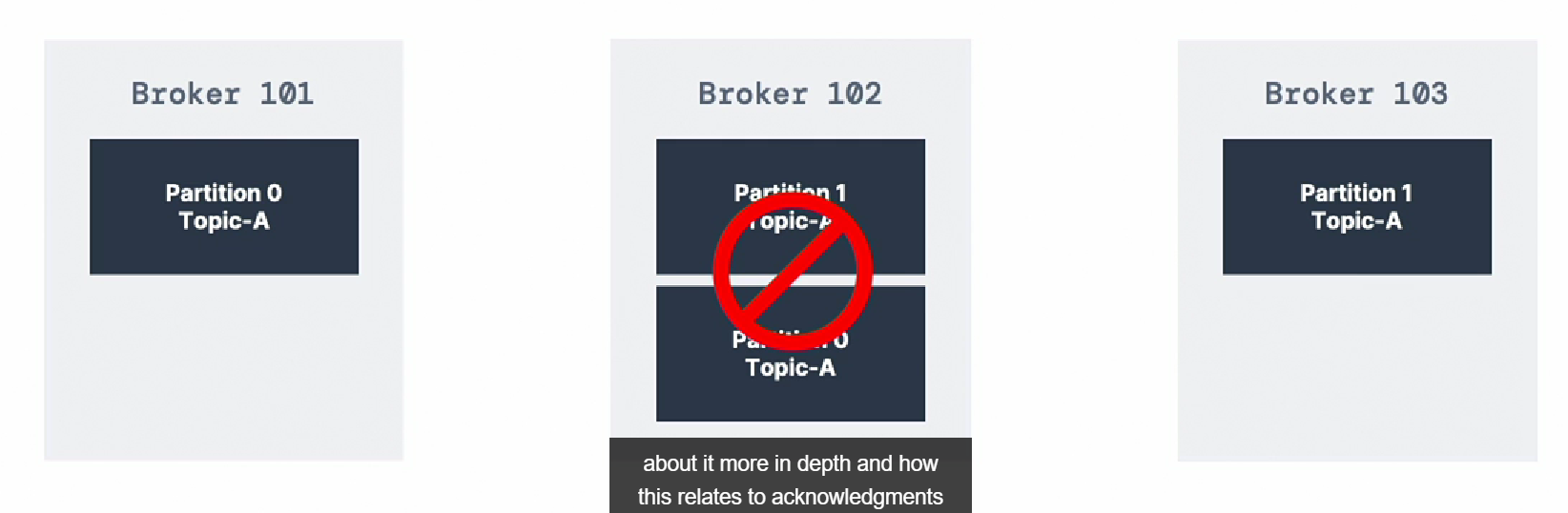
**

*Producer can choose to receive ack of data writes:*

* *Acks = 0 producer won’t wait for acks (possible data loss)*
* *Acks = 1 producer will wait for leader ack (limited data loss)*
* *Acks = all leader + replica acks (no data loss)*

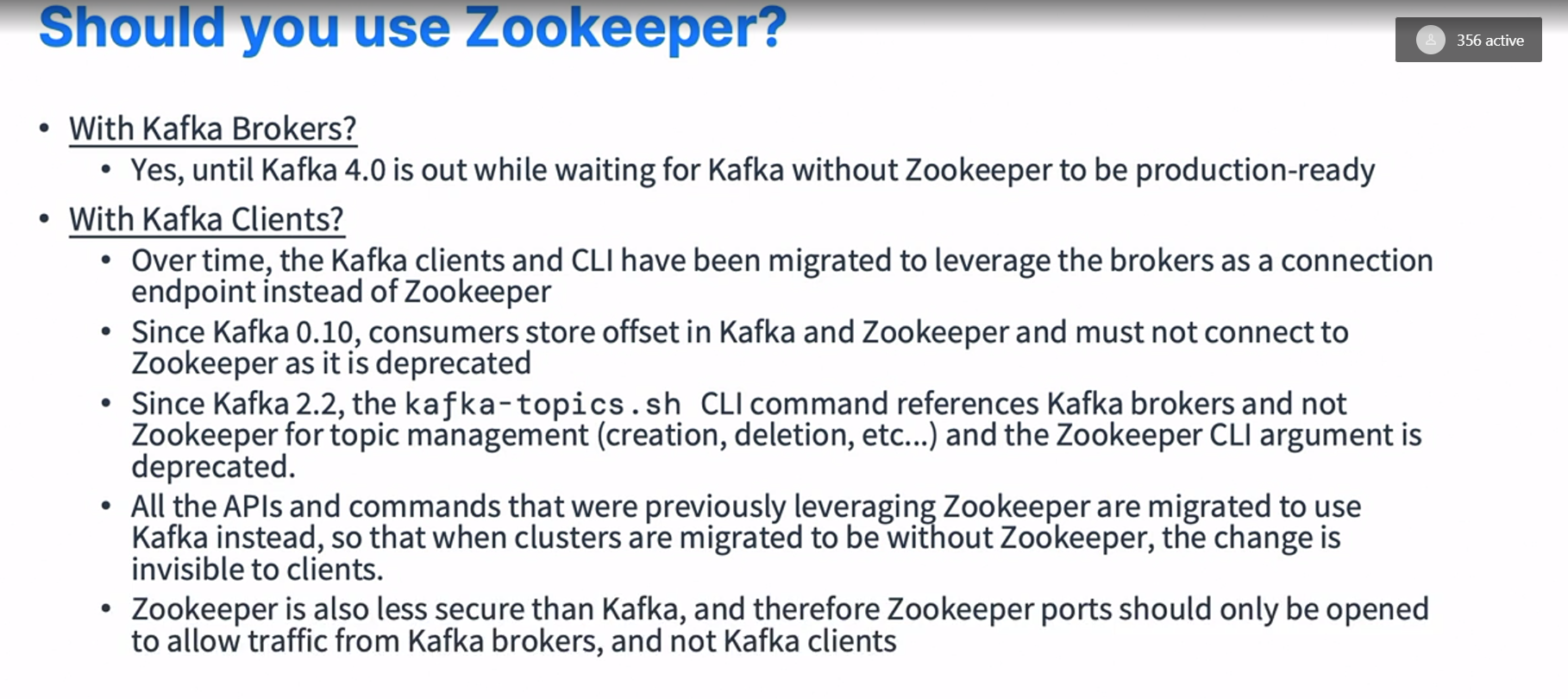
***# Kafka Topic Durability:***

* *For a topic replication factor of 3, topics data durability can withstand 2 broker loss.*
* *As a rule, for a replication factor of N, You can permanently recover your N-1 brokers and still recover your data.*

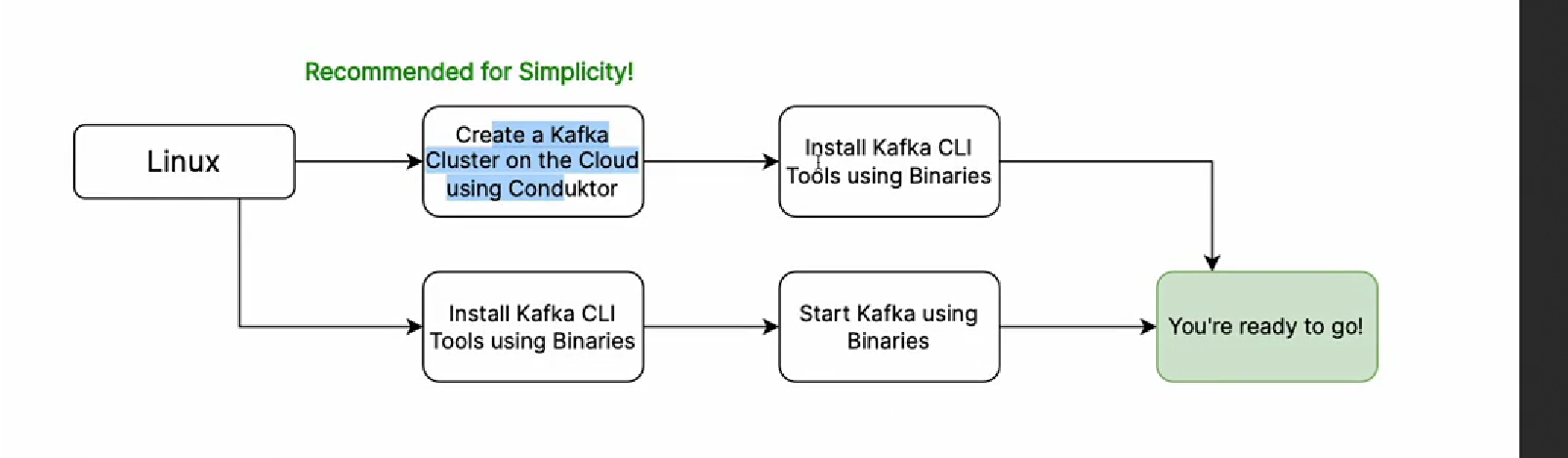
******

***#Zookeeper in Kafka:***

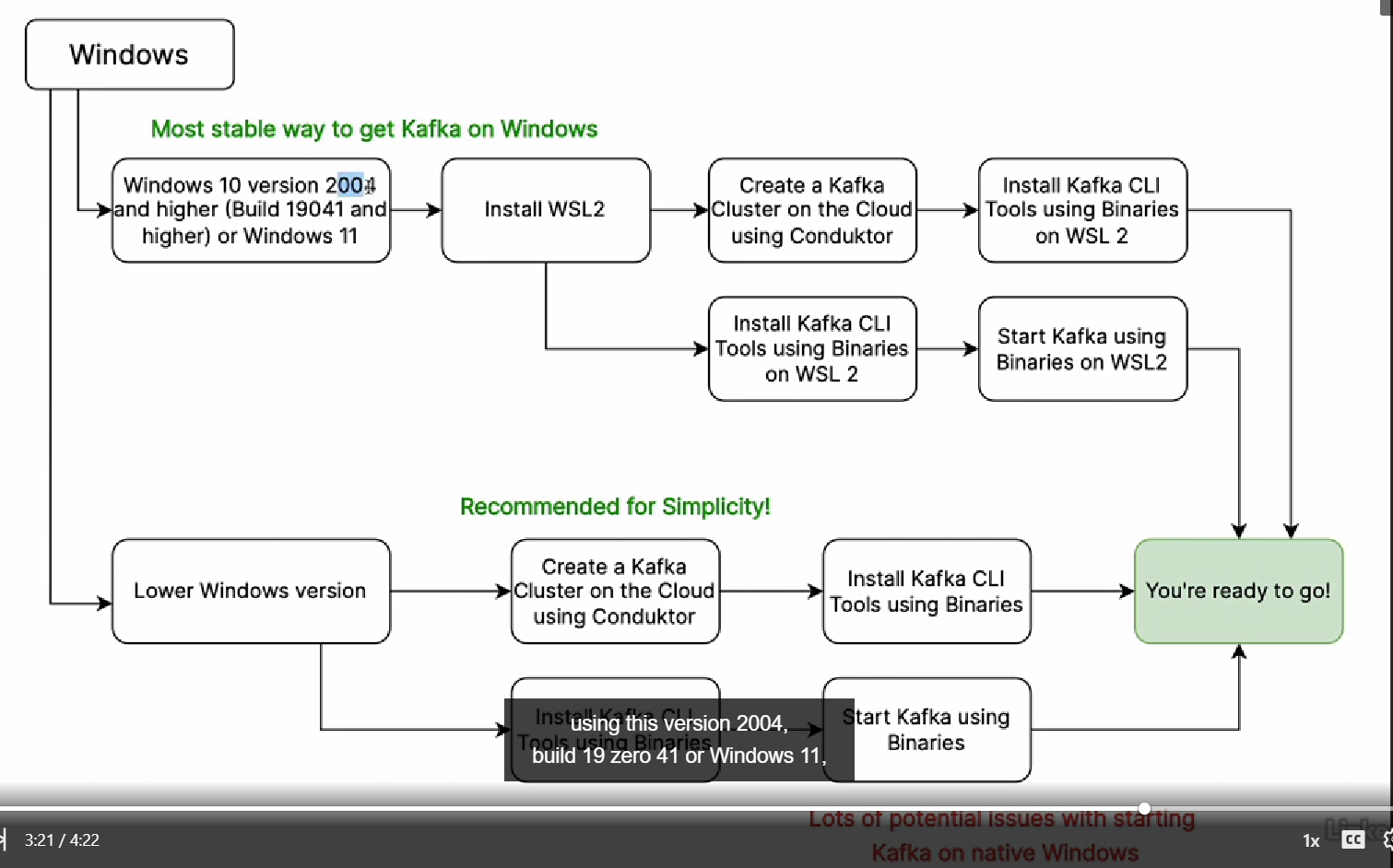
* *Zookeeper manages brokers (keeps a list of them)*
* *Zookeeper helps in performing leader election for partitions*
* *Zookeeper sends notifications to kafka in case of changes of (eg. New topic, broker dies, broker comes up, delete topics)*
* *Kafka 2.x can’t work without zookeeper*
* *Kafka 3.x can work with out zookeeper*
* *Kafka 4.x will not have zookeeper*
* *Zookeeper doesn’t store consumer offset with( kafka > v 0.10)*

**

*#Installation on Linux*

**

*#installation on windows*

**