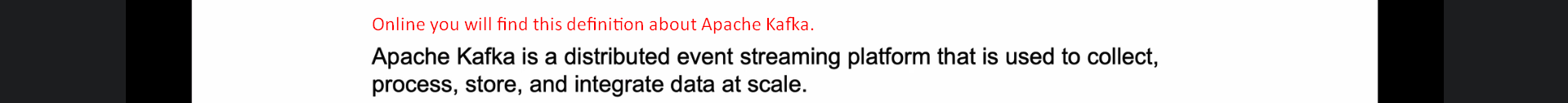
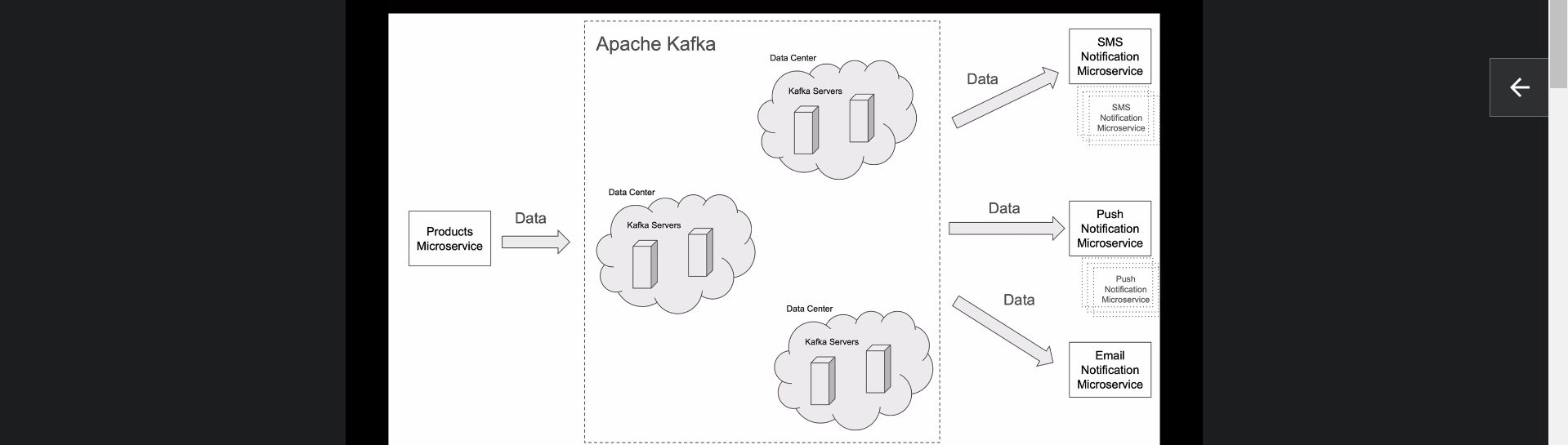
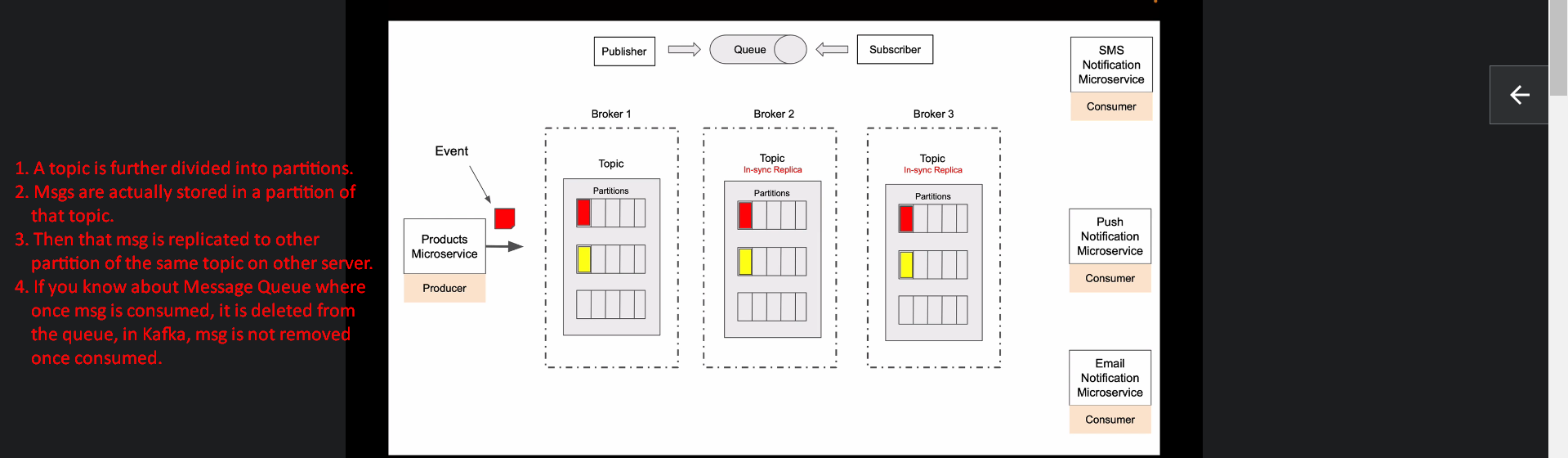
1. 
2.   
   When I think about it as Microservice Developer, I think of Apache Kafka as a Platform that can run a cluster of Multiple servers which can span across several Data Centres.  
   It enables us (Developers) to build Event-Driven Mss that can continuously exchange data with each other and we can scale these MSs as needed to support larger traffic.

Let’s look at more detailed example how Apache Kafka can be used with Microservices.

1. Apache kafka can help MSs to communicate with each other.  
   For example: On the left hand side of the Diagram, Products MS can to inform other MSs in the system about new product creation and it can do by publishing an event. It can publish the event “CREATED-NEW-PRODUCT” in the form of JSON payload.  
   As this Products MS can publish/produce an event in Kafka, so it is called **Producer**.  
   Kafka provides **Producer API**.  
   So, one of the core components in applications that uses Kafka is **Producer** and in Microservice Architecture, a **Producer** is a microservice that publishes an event.  
   On the right hand side of the diagram, the MSs are waiting for the events to be consumed. As they are waiting for msgs, in Kafka they are called **Consumers.**Consumer is second core component in applications that uses kafka. In Microservice Architecture, a consumer is a microservice that consumes an event.   
   Consumers can scale up and down as per load or due to some issue and recovery and their IP:PORT will keep on changing but Producer doesn’t know about it. Their location is completely transparent to the Producer.
2. **Broker**:   
   A producer sends msg to Kafka Broker which stores the msg on Hard-Disk.   
   We can start more than one instance for more reliability like in case one is down, other one can serve the request.  
   In Broker is another important architectural component in Kafka Based Application and it has many important responsibilities.
3. **Topic**:   
   Msg is store inside a topic which is another important component in app architecture that uses Kafka.  
   We can think of a topic as a very durable storage of events.  
   the msg inside a topic, it is stored on hard disk and replicated on other Kafka cluster.  
   So, in case data is lost on one server or server itself is down, other server can serve the data and consumers will not know about this downfall.  
   
4. **Scaling**:

As Msgs are stored in different partitions of the same topic, so multiple consumers can consume msgs parallelly from different partitions.   
it allows us to scale up the consumer to read msgs faster and thus fast processing.