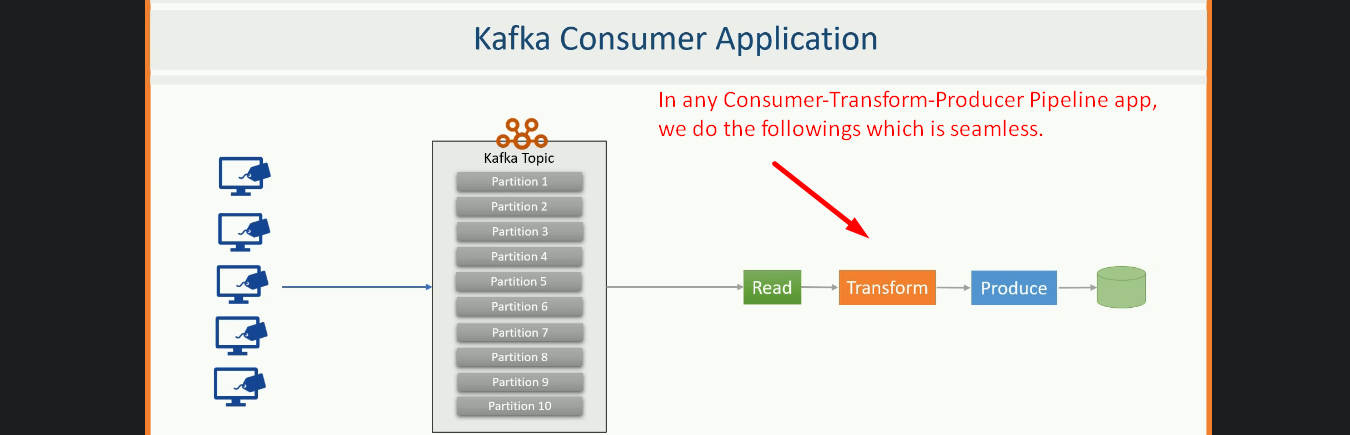
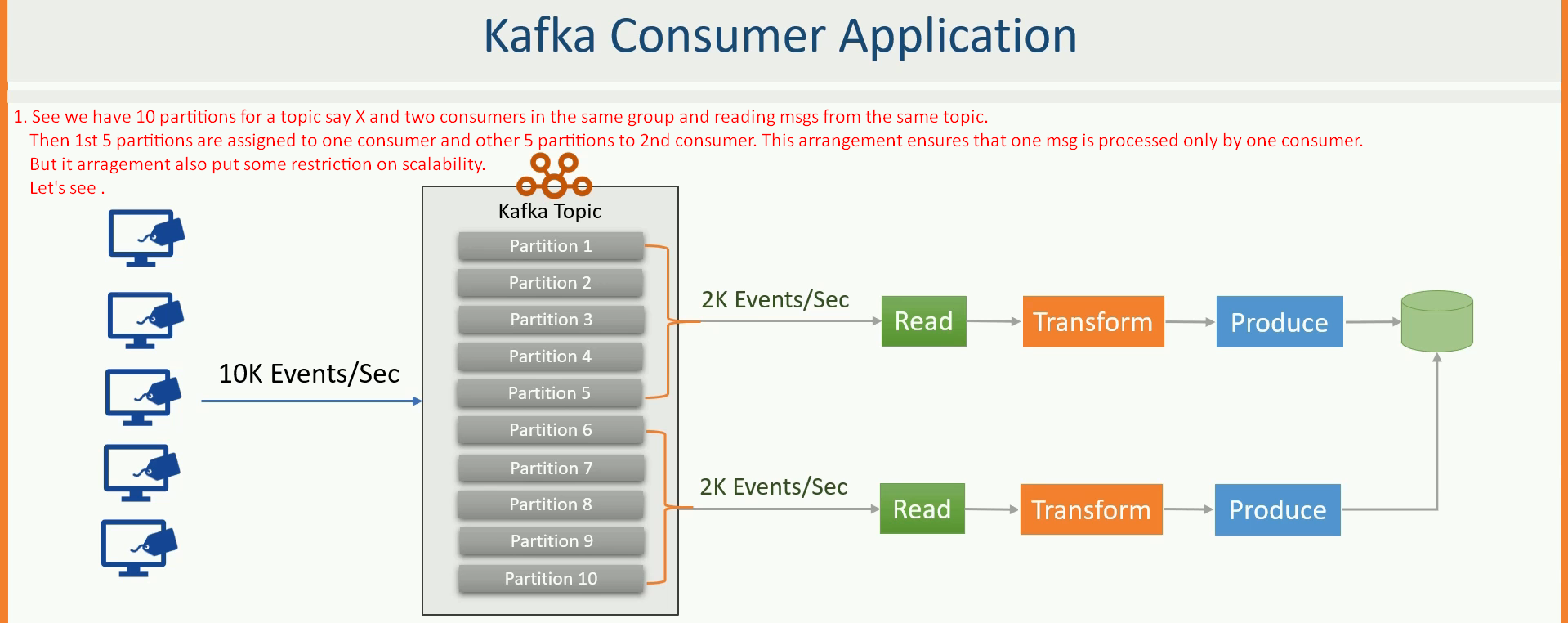
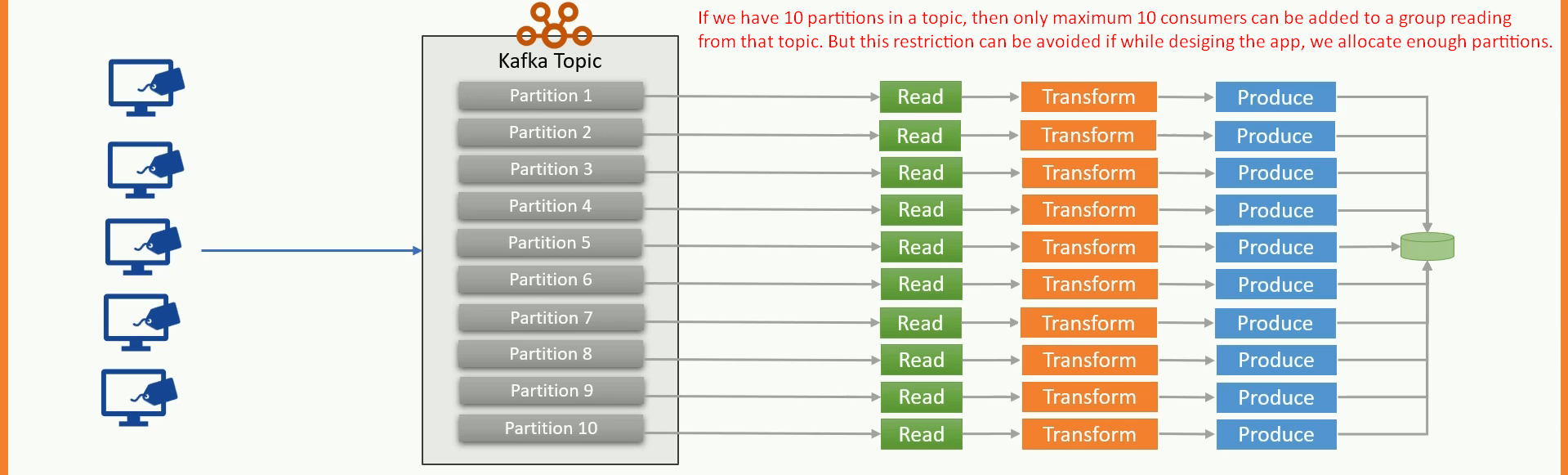
1. Text

   Description automatically generated  
   
2. In the earlier lecture, we created a realtime data validation pipeline.  
   Graphical user interface, application

   Description automatically generated  
   In the above case, we’re sending invoices to respective topic based on they are valid or invalid.   
   But we can store those invoices in any kind of store like DB.
3. In all such Consume-Transform-Produce Pipelines, your app will create a Kafka Consumer Object, subscribe to the appropriate topic, & start receiving the msgs, transforming them and writing the results.  
   This process should be seamless.  
   
4. However, your app will begin falling behind, if the rate which the Producer writes the msgs to a topic exceeds the rate at which you can transform and write them to some external storage.  
   Chart

   Description automatically generated with medium confidence  
   If you’re limited to a single consumer reading and processing the data, you may fall farther and farther behind and would not remain a realtime app.

In this scenario, we need to scale the consumption.  
But the question is how?  
How do we scale a Consumer Process?

1. 
2. We can scale the consumer process by dividing the work among multiple consumers.  
   Just like numerous producers can write to the same topic, we need to allow various consumers to read from the same topic.  
   However, we also need a mechanism to split the data among the consumers so that they work with own set of data and don’t interfere with other consumers.  
   This is where topic partitions are handy when we have multiple consumers working in a group and reading from the same topic, we can easily split the data among the consumers by assigning them one or more partitions.
3.   
     
   
4. Now the next question is
   1. how to create multiple consumers.
   2. How to add multiple consumers to same group.
   3. Do we need to do something to assign partitions to each consumer in the group.
5. Kafka offers automatic group management & rebalancing of the workload in a consumer group.  
   All we need to do is to set GROUP ID configuration.  
   Kafka automatically forms a consumer group and it will automatically add a consumer to the same group if the consumer has same GROUP ID.  
   Kafka will also take care of assigning the partitions to the consumer in the same group.  
   Membership in the consumer group is maintained dynamically.  
   So, if a consumer fails, its partitions will be assigned to some other consumer group.  
   Similarly, if a new consumer joins the group, the partitions will be moved from the existing consumer to the new one to maintain the workload balance.  
   Conceptually, you can think of a consumer group as being a single logical consumer that happens to be made of multiple processes sharing the workload.  
   Kafka automatically manages all of this and the who process is transparent to a user.  
   You as a Developer are responsible for creating a consumer with same GROUP ID and then starting it on the same machine or different machine.  
   