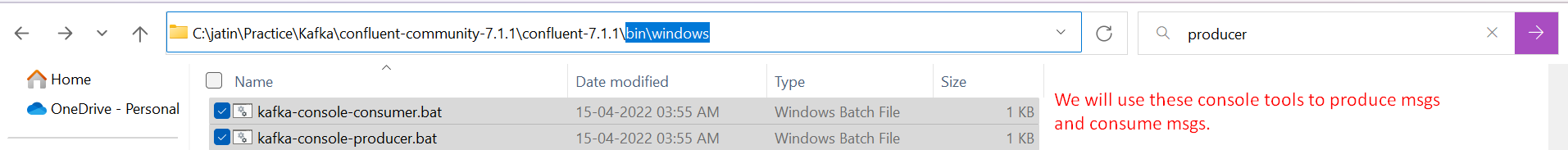
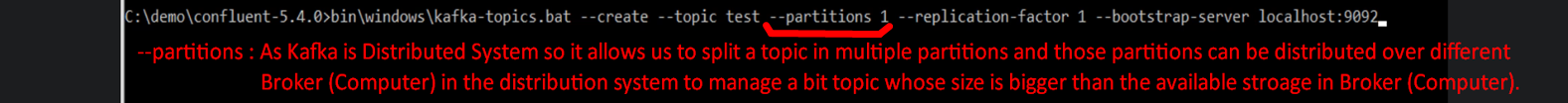


1. **Agenda**:
   1. We will be using **Console-Producer Tool** & a **Console-Consumer Tool** but not a **Producer or Consumer app**.
   2. 
2. So, we want to do the followings in this demo.  
   Text

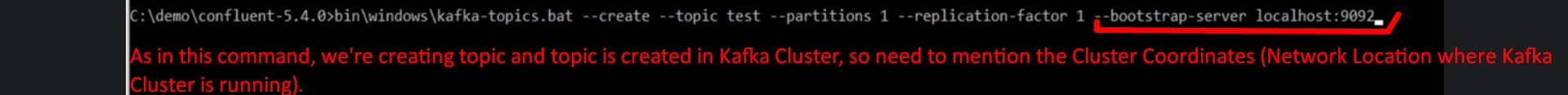
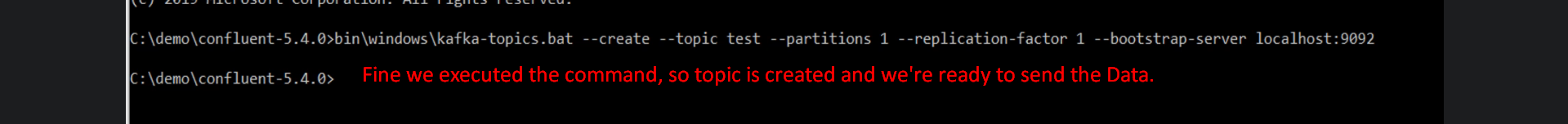
   Description automatically generated
   1. So, we have a data file (**CSV**) having some data which we want to send to **Kafka Cluster**.
   2. However, a **Producer** should send data always to a **Kafka Topic** (inside **Kafka Broker**).  
      We will do it in two steps.
      1. **Step 01-A**: Create a **Topic** first. We will be using **Kafka-topics Command Line Tool.**
      2. **Step 01-B:** We will be sending all the data from the file to the **Kafka Cluster**.  
         To read data from a file and send it to **Kafka Cluster**, we will be using **kafka-console-producer Tool which is a script file**.  
           
         Once, data starts coming to the Kafka Cluster, we will move to the next step.
      3. **Step 02-B**: We will be running a **Kafka-console-consumer Tool** to read(**consume**) all the data from the Kafka Topic & display it to the console.  
         
3. **How to create a Kafka Topic?**

  
While creating a topic in the Kafka Cluster, we also need to tell the number of **partitions**.

1. How to decide the number of **Partitions**:  
   
   1. 
   2. **options**:
      1. **--create**: To create a topic.
      2. **--partitions**: To define partitions number.
         1. **Question**: But the question is how many partitions?
         2. **Answer**: There are two evaluations when deciding about partitions number.
            1. **Storage Requirement:**As we have very small **Data Set** so we’re not going to consume a huge storage and that will fit fine into a **single Broker**.   
               So, there is no storage concern.
            2. **Parallel Processing Requirement**  
               We will be reading from this topic using a single **Consumer**.  
               So, there is no need for **parallel processing**.

So, the number of partitions = 1.

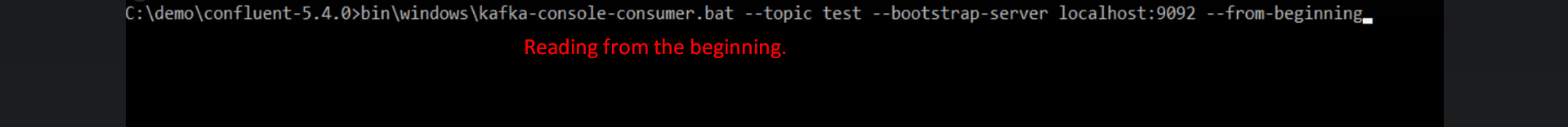
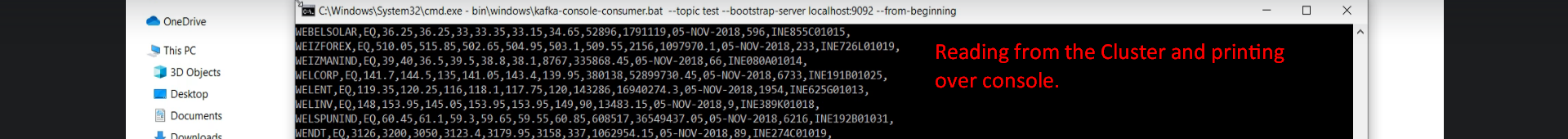
* + 1. **--replication-factor**:
       1. The number of copies of each partition in a topic.
       2. Provides **Fault-Tolerance** Feature.
       3. The copies are stored at different Brokers.   
          It is useless to store more than one copy on the same Broker Server.  
          But in our case, we’re setting replication-factor = 1 as we don’t have more than one Broker.
    2. **--bootstrap-server**:
       1. As we’re going to create the Topic in the Kafka Cluster.   
          So, we need to tell the Cluster Co-ordinates.
       2. This command takes the Cluster Co-ordinates as Bootstrap Server.
       3. This takes ip:port where ip is the Kafka Broker.

1. **~~Cluster Coordinates~~**~~: It is~~ **~~Kafka Cluster Network Location~~**~~.   
   As we want to create a Topic in the~~ **~~Kafka Cluster (Kafka Broker)~~**~~, so we need to specify the~~ **~~Kafka Cluster Coordinate~~** ~~(~~**~~Network Location~~**~~) in the command while creating the topic.   
   So, the command takes~~ **~~--bootstrap-server~~** ~~as~~ **~~Kafka Cluster Coordinates~~**~~.~~
2. 

Now we’re ready to send the CSV File Data to the Kafka Cluster.



1. We will use Kafka-console-producer tool to send file data to Topic Test in Kafka Cluster.   
     
   A screen shot of a computer

   Description automatically generated with low confidence  
     
   
2.    
   So, what did we do?
3. We simulated the following:  
   Icon

   Description automatically generated  
   **Kafka Cluster**:
   1. We have a Kafka Cluster in the Center.
   2. It was a Single-Node Kafka Cluster, but it could have been a 50-node-cluster.

**Kafka Producer**

1. We had some data files on a remote (Well, we did it on the same computer. But it could have been on remote device that is too far from the cluster and connected over **TCP/IP network**).
2. We sent those files to the Kafka broker.

**Kafka Consumer**:

1. We had a consumer sitting on the same machine but like **Kafka Producer**, it could have been on a different machine thus reading data over **TCP/IP Network**.