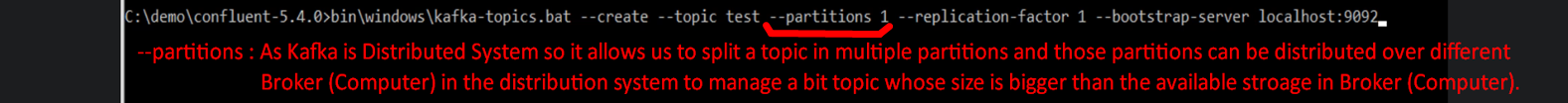
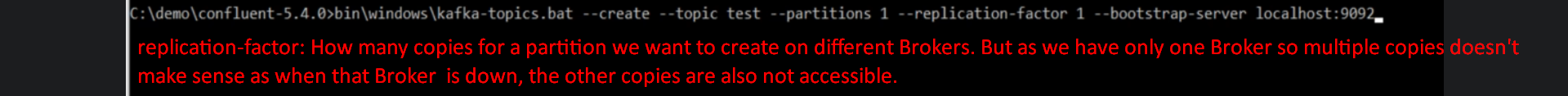
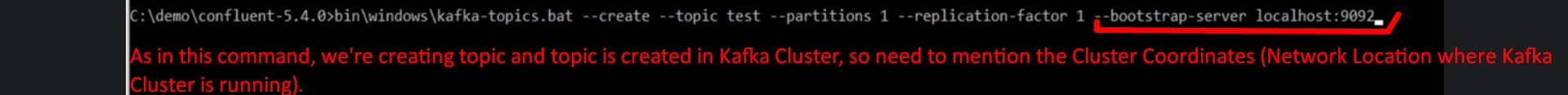
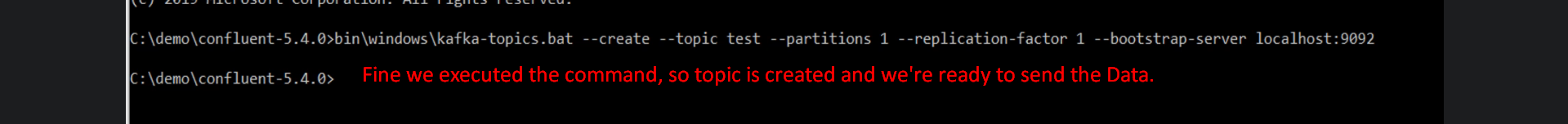
1. **Agenda**:
   1. We will be using **Console-Producer Tool** & a **Console-Consumer Tool** but not a **Producer or Consumer app**.
2. So, we want to do the followings in this demo.  
   Text

   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-topic Command Line.**
      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**.

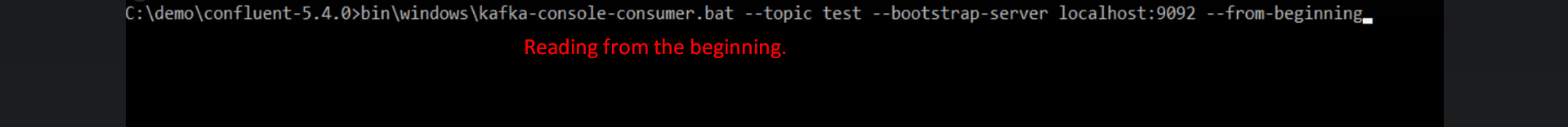
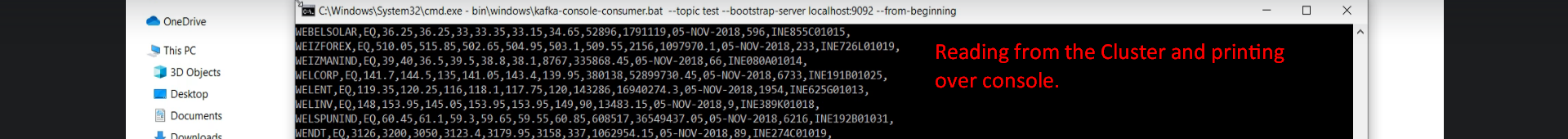
1. How to decide the number of **Partitions**:  
     
   So, there are **two evaluations**.
   * 1. **Storage Requirement**:
        1. As we have very small **Data Set** so we’re not going to consume a huge storage and that will fit fine into a single broker.   
           So, there is no storage concern.
     2. **Parallel Processing Requirement:**
        1. We will be reading this data using a single **Consumer**.  
           So, there is no need for **parallel processing**.

**Conclusion**: Single partition is sufficient.  
Now, the next thing mandatory thing in the command is to specify the **replication-factor**

1. **Replication-Factor**
   1. The **replication-factor** is the number of copies of each **partition**.
   2. You can create two or three copies of each **partition** by giving high **replication-factor**.   
      Why so many copies?  
      **Answer**: **Fault Tolerance Feature**.
   3. The copies are store at different **Brokers**.   
      If one **Broker** is down, we have another **Broker** with a copy of the **partition**.  
      **NOTE**: Since we have one **Broker**, so doesn’t make sense to create more than one **partition**.   
      So, the **replication-factor=1**
2. **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**.   
   
3. 



1. A screen shot of a computer

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