1. **Apache Kafka** : Distributed Streaming Platform.
   1. **Kafka Server**
   2. **Another Def**: Apache Kafka is highly Scalable, and distributed platform for creating and processing streams in real-time.
   3. **From Data Integration to Streaming Platform**.
2. Enterprise Data/Application Ecosystem.
3. Real-time Data Stream (Stream of Data), Real-time Stream Processing.
   1. **Continuous Stream of Data**: When data is flowing continuously within a given timeframe.
   2. **Real-time stream Processing**: Not accumulating the data but being processed as it is coming.
   3. **Creating Real-Time Stream**: Bringing data from source to Destination.
4. Publish-Subscribe Messaging System Architecture.
   1. **Enterprise Messaging System**.
   2. **Message Producer** : Publisher. Client App : Producer App to create and send data as a stream.
   3. **Message Consumer** : Subscriber. Client App : Consumer App to process the data stream in realtime.
   4. **Message Broker :** We install & configure Kafka Server as Message Broker**.**
5. Kafka Evolvement from 2011 to 2019
   1. **Server Software** : Broker.
   2. **Client API**: Java Based Library to create
      1. Producer App.
      2. Consumer App.
   3. **Kafka Connect**: To address the initial data integration problem.
   4. **Kafka Streams**: To create realtime stream processing App.
   5. **KSQL**: To become a realtime DB.
6. Producer App, Data = Message = Message Record, Record Structure = Schema = Array of Bytes.
7. Consumer, Read Permission,
8. Broker = Message Broker = Agent = Kafka Server = Kafka Broker,
9. Cluster = A set of computers where each computer is running an instance of Kafka Server.
10. **Topic** = An arbitrary unique name given to a data set or a data stream = A Database Table
11. **Partitions** = A small & independent portion of a topic.
12. **Partition Offset** = Offset Id = Arrival Order#, A unique Sequence ID of a Message in a partition, assigned by Broker, immutable, Zero
13. **Consumer Group**: To share workload.
    1. Retail Chain -> Stores -> Billing Counters -> Invoices -> Data Store.
14. Enterprise App = Custom Designed & Developed in-house + 3rd Party Apps
    1. Each has its own network. Each generating data. Some data from outside like youtube.
15. Financial Accounting Software needs data from Invoicing.  
    Inventory System from Warehouse + Shipment + invoicing  
    Analytics System from everywhere.
16. Source System, Producer App,
    1. Embedded Kafka Producer using Kafka Producer API if you have source code.
    2. Independent Kafka Producer: Kafka Connect.
17. Kafka Connect: Cluster -> A set of Kafka Connect Worker.
    1. **Kafka Connect Source Connector** B/W 🡪 Data Source(DB) and Kafka Cluster. Uses **Kafka Producer API** internally.
    2. **Kafka Connect Sink Connector** B/W 🡪 Kafka Cluster & Consumer. Uses **Kafka Consumer API** internally.
    3. Kafka Connect Framework: To write custom connectors.
       1. Source Connector : Implement SourceConnector.java, SourceTask.java
       2. Sink Connector: Implement SinkConnector.java, SinkTask.java  
          Uber Jar or Zip Achieve File.
    4. JDBC Source Connector, Snowflake Sink Connector.
    5. Lecture 3 was skipped
18. d

Section 2 Apache Kafka - Getting Started

1. Single/Multi Node Kafka Cluster.
2. Command Line Kafka Prouder/Consumer.
3. Consumer Group: Consumer Workload Sharing.
4. Three Kafka Flavors:
   1. **Opensource** - Apache Kafka (operational issues, open bugs, infrastructure issues, Kafka Admins, Infrastructural Experts),
   2. **Confluent Kafka**: Commercial Distribution - Community Distribution, Commercial Distribution.
   3. **Managed Services**: Confluent Cloud, Amazon MSK, Aiven.io
5. **Confluent Community Edition**
   1. **zookeeper-server-start.bat** zookeeper.properties   
      **kafka-server-start.bat** server.properties -> broker id
6. **Apache Opensource Edition**:
   1. **zookeeper-server-start.bat** zookeeper.properties
   2. **kafka-server-start.bat** server.properties
7. Console Producer/Consumer Tool.
8. **Kafka -topics.bat** --create --topic test --partitions 1 --replication-factor 1 --bootstrap-server localhost:9092
9. **kafka-console-producer.bat** --topic test --broker-list localhost-9092 < sample1.csv.
10. **Kafka-console-consumer.bat** --topic test --bootstrap-server localhost-9092 --from-beginning --group <group-name>
11. **broker.id (**auto-assign identifier)**, log.dirs=/tmp/kafka-logs (**dir where partitions data will be stored)