1. 
2. **Agenda**:
   1. A quick introduction to Kafka.
   2. What does it mean to developers to use **Apache Kafka as a streaming platform** before we get into the details of Kafka.
3. Before going into the details of Kafka, we will take a step back and highlight on the current state of Software Development.  
   Diagram

   Description automatically generated
   1. In the past, all the apps that were build was **monolithic apps** using monolithic architecture.  
      That means all the functionalities will reside in a single app.  
      **Example**: Retail App.  
      Some of the services are 🡺 Order, Inventory, Payment, Notification services.  
      All services reside in one single app and **share same DB**.  
      Such kind of architecture was proven **to fail under heavy load**.
   2. Things have changed. The current state of development uses more modern architecture which is MS architecture.  
      As you can see app is decomposed into microservices and each has its own DB.  
      But as a whole, in order to deliver the business functionality/value , multiple microservices have to **interact** with each other using some **communication protocol**.  
      Arrows indicate the communication among them but it looks **messy and spaghetti**.  
      **But expectations from today app that are built today have a new requirement which is providing the real time notifications and process the events as they occur**.   
      In order to support that we need to have a **middleware** in b/w and a full flown MS architecture looks like this.  
      Diagram

      Description automatically generated  
      The **middleware** that we’re using here is **Even Streaming Platform**. 🡺 Will explore it in a bit.  
      Basically, at its core, **each MS will have an APIs** which are going to **generate a lot of events**.  
      Other services **read those events** and **take necessary actions** as events occur.  
      **In nutshell, each MS has an API, event producer and even consumer.  
      All other microservice communicate with each other through Event Streaming Platform.**This fundamentally forms the basis for **Event Driven Micro Service.**
4. You might be having this question right now in your mind that what does an Event Streaming Platform fundamentally mean?

Graphical user interface, Teams

Description automatically generated

* 1. **Producers and Consumers subscribe to a stream of records**:
     1. Even Streaming Platform allows an app **to produce and consume a steam of records** like a **Messaging System**.  
        You can think of it as **Pub-Sub Model** (Publish-Subscribe Pattern).  
        Producer and consumers are **independent** as Producer doesn’t know which consumer is going to read the msg.
  2. **Store stream of Events**:
     1. Event Streaming Platform also stores the stream of events so that it can be replayed if necessary.
     2. Events are generally retained in multiple servers for providing **fault-tolerance and availability**.
  3. **Analyze and Process Events as they occur**:
     1. Event Streaming Platform also allows the app to process the records as they occur.

**NOTE**: Basically, these three principles form the fundamental for the Event Streaming Platform and Apache Kafka is build on these three principles.

1. **Many of you must have this question that Kafka is Enterprise Messaging System then**?  
   A picture containing table

   Description automatically generated

|  |  |
| --- | --- |
|  |  |
| 1. Once records are **read** the consumers, they will be **removed** from the **message broker**. | Events will be stored in a file where Kafka is installed for certain amount of time. |
|  | Events are **immutable**. |
| 1. Message Broker is **responsible to remove the messages** when read. | Consumer is responsible. Don’t worry we will see different techniques for doing this. |
| 1. A specific consumer can read the msg. | Any consumer having access to the Message Broker. |
| 1. **Doesn’t** follow the D**istribution System Principle** | It is distributed Streaming System. A distributed system is proven to handle variable load and has the intelligence to distribute the load. |

I believed the differences we discussed here justifies that Kafka is a streaming system not just a messaging system.

1. **Kafka Use-Cases**: