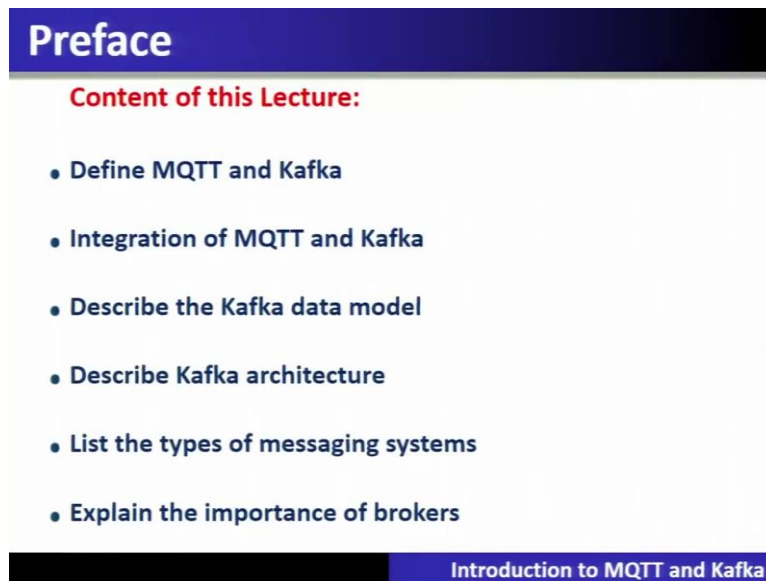


Foundation of Cloud IoT Edge ML
Professor Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna
Lecture - 16
Introduction to MQTT and Kafka in IoT Platform

I am Doctor Rajiv Misra from IIT Patna. The topic of today's lecture is Introduction to MQTT and Kafka in IoT Platform.

(Refer Slide Time: 0:25)



Preface

Content of this Lecture:

- Define MQTT and Kafka
- Integration of MQTT and Kafka
- Describe the Kafka data model
- Describe Kafka architecture
- List the types of messaging systems
- Explain the importance of brokers

Introduction to MQTT and Kafka

The content of this lecture is as follows. We will introduce MQTT protocol and Kafka connectors in this particular session, which is very useful for connectivity for Internet of things devices, to the cloud data center, and so on. Then, we will introduce about the integration of MQTT and Kafka, how both together can be inter integrated and what are the scenarios in which this integration is going to work. Then we will describe the Kafka data model. We will also introduce the Kafka architecture. Then we will also list the type of messaging system and explain the importance of brokers in Kafka.

(Refer Slide Time: 1:21)

Introduction: Internet of Things with MQTT

MQTT (Message Queuing Telemetry Transport): ✓ *AWS IoT, Azure IoT, Ad*

MQTT is a widely used ISO standard (ISO/IEC PRF 20922) client-server messaging protocol.

The protocol is lightweight and implements a publish/subscribe communication pattern.

MQTT is stable in unreliable environments of high latency and low network bandwidth which makes it a perfect match for Internet of Things scenarios like connected cars or smart homes.

MQTT has many implementations of client libraries and brokers like Mosquitto, HiveMQ, JoramMQ, etc and its primary purpose is to connect millions of devices — especially in the IoT context.

Introduction to MQTT and Kafka

So, let us begin with Internet of Things with MQTT. MQTT full form is Message Queuing Telemetry Transport, this is one of the most important messaging used for Internet of Things. So, MQTT is widely used is an ISO standard and it is working in the basis of client server messaging protocol. This particular protocol is lightweight and implements the publish subscribe communication mechanism.

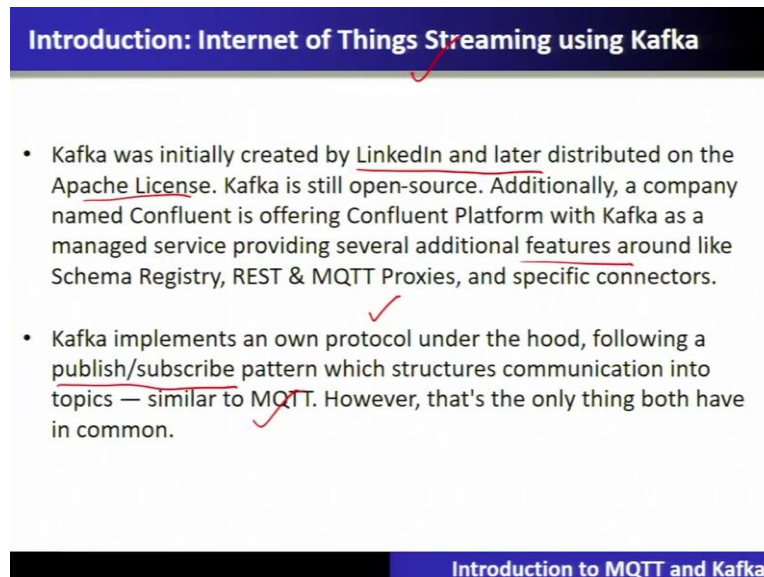
So, MQTT is a stable in unreliable environment of high latency and low bandwidth, which makes it a perfect match for Internet of Things scenarios like connected cars or smart homes. MQTT has many implementations of client libraries and brokers like Mosquitto, HiveMQ, JoramMQ, et cetera. And its primary purpose is to connect millions of devices especially in the Internet of Things context.

So, as the name suggests, as the details suggested here in this slide, so, we have to know summarize that MQTT is a messaging protocol and the full form is Message Queuing Telemetry Transport. Now, this particular protocol uses the concept of publish and subscribe communication mechanism and MQTT is stable even in unreliable environment and it supports high latency and low network bandwidth. And therefore, it is very useful in the situations like internet of things, which are the small devices or small devices are using very low bandwidth and often having high latency. So, these kind of protocol is very much useful for Internet of Things applications.

Now, there are various implementations because of its wide acceptability as an international ISO standard. This client server messaging protocol is widely used and is being provided in

various implementations like Mosquitto, HiveMQ and so on. And it is now a defacto standard to connect millions of devices especially in the IoT context, and we have also seen IoT platforms like AWS IoT and Azur IoT Hub. That IoT platform also we have seen the use of MQTT protocols.

(Refer Slide Time: 4:24)



Introduction: Internet of Things Streaming using Kafka

- Kafka was initially created by LinkedIn and later distributed on the Apache License. Kafka is still open-source. Additionally, a company named Confluent is offering Confluent Platform with Kafka as a managed service providing several additional features around like Schema Registry, REST & MQTT Proxies, and specific connectors.
- Kafka implements an own protocol under the hood, following a publish/subscribe pattern which structures communication into topics — similar to MQTT. However, that's the only thing both have in common.

Introduction to MQTT and Kafka

So, let us understand this MQTT protocol, which works in the following manner that is in the form of a client server messaging protocol implements the logic which is called the publish subscribe messaging communication protocol. So, MQTT you can understand here, it is a publish subscribe protocol. So, here you can find these IoT devices. And they publish when they will send the IoT data it is called this operation is called publish in MQTT protocol and this publish to the MQTT broker and this becomes a server and there this becomes a client. So, it works in a client server fashion.

So, when you say server, so obviously, that MQTT broker has a good amount of resources as far as computing and storage is concerned. Then as far as the other side is concerned consumer or the end use of these kind of messages required this particular protocol which is called subscribe. So, the end use that is the consumers will subscribe to these particular messages. Now, to subscribe this kind of publish subscribe mechanism. So, what MQTT broker has done is that the data which is sent by IoT devices is organized is indexed in the form of the topics.

So, they will publish the data according to this particular topic and that will be stored in MQTT broker which is nothing but a scalable server and I and whenever, as far as consumers

are concerned, they will subscribe to these particular topics and they will read through the MQTT broker.

So, Kafka when we talk about the Internet of Things is streaming using Kafka. So, Kafka was initially created by LinkedIn and later distribute the Apache License. So, it is an open source platform, which is working on. So, various companies like Confluent is offering this particular Kafka as a managed service, providing several additional features like registry, REST, and MQTT proxies, and a specific connector.

So, Kafka implements its own protocol under the hood, following again the same mechanism which is called publish subscribe pattern, which structures the communication into the topics very similar to MQTT, but the only this is the only thing which is common across MQTT and Kafka, but rest of the things let us see how it differs.

(Refer Slide Time: 7:36)

Introduction: Internet of Things Streaming using Kafka

- Kafka is designed to be deployable as a cluster of multiple nodes which makes it excellent for scaling. Additionally, it offers persistent storage of messages and integration to business on-premise or cloud data centers and applications.
- Its main use cases are distributed event streaming and storage/consumption of massive amounts of data as messages.
- It makes Kafka a perfect match for scenarios that require high-performance, scalable data pipelines, or data integration across multiple systems.

Introduction to MQTT and Kafka

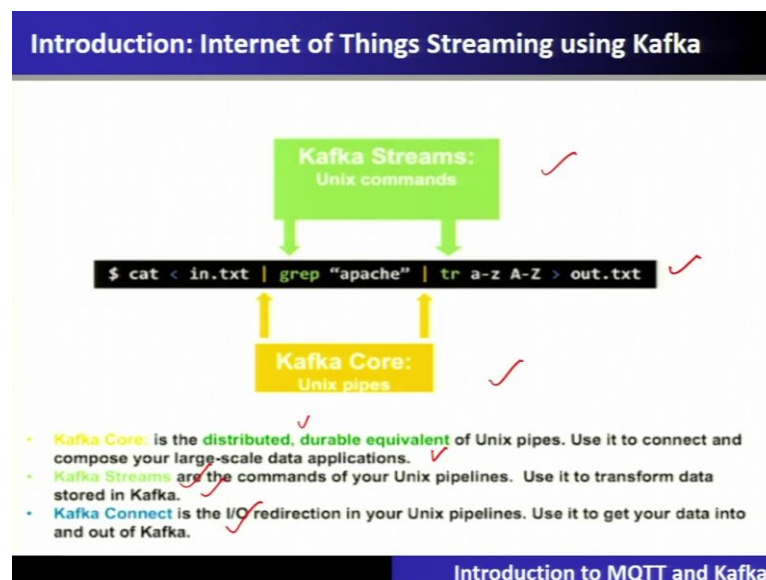
So, Internet of Things, uses Kafka for streaming data analysis, that is in the real time, streaming analytics, that is also called hot data analytics. So, Kafka is designed to be deployable as a cluster of multiple nodes. So, therefore Kafka is not a single server, but it is the cluster maybe a data center. So, which makes an excellent for scaling. So, it is not that only few Internet of Things devices, many devices in very large number also can use this particular service, why, because Kafka is designed to be scalable as deployable in a cluster of multiple nodes.

So, additionally, it offers a persistent storage of the messages and integration to the business on premises and the cloud data center. So, its main use cases are distributed event streaming,

and storage consumption of the massive amount of data as the messages. So, it makes Kafka a perfect match for the scenarios that require high performance scalable data pipeline or integration across multiple systems. So, you might have seen that several IoT devices on (())(9:02), they are now communicating to the edge device. So, Kafka is now also implementable across edge and cloud data center.

So, these kind of data ingestion, which is happening in this IoT environment, now require that data to be stored in databases and then perform the machine learning on it. Therefore, it makes this graph correct perfect match of this particular scenario, because this scenario requires high performance, scalable data pipelines or data integration across multiple systems, unlike MQTT.

(Refer Slide Time: 9:44)



So, let us understand about the Kafka streaming or a Kafka in more details. So, you can understand the Kafka streams and Kafka core, there are two important concepts. So, if you see this kind of Unix pipe, then what do you will find here is that Kafka core is a distributed, durable equivalent of Unix pipe and use it to connect and compose the large data applications. Whereas the Kafka streams are commands of your Unix pipelines and it transforms the data is stored in the Kafka. So, Kafka connect is an IO redirection in the Unix pipeline, and it get your data out and in of the Kafka system.

(Refer Slide Time: 10:36)

Introduction: Internet of Things Streaming using Kafka

- **Kafka is a high-performance, real-time messaging system. It is an open source tool and is a part of Apache projects.**
- The characteristics of Kafka are:
 1. It is a distributed and partitioned messaging system.
 2. It is highly fault-tolerant
 3. It is highly scalable.
 4. It can process and send millions of messages per second to several receivers.


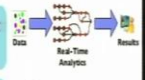



Introduction to MQTT and Kafka

So, let us go in more detail about the Kafka. So, Kafka is a high-performance real time messaging system. It is an open source tool, and it is a part of Apache a project. So, main characteristics of Kafka are as follows. So, it is a distributed and the partition messaging system, it is highly fault tolerant, it is highly scalable, and it can process and send millions of message per second to the several receiver.

(Refer Slide Time: 11:07)

Kafka Use Cases

- Kafka can be used for various purposes in an organization, such as:

Messaging service	Millions of messages can be sent and received in real-time, using Kafka.	
Real-time stream processing	Kafka can be used to process a continuous stream of information in real-time and pass it to stream processing systems such as Storm.	
Log aggregation	Kafka can be used to collect physical log files from multiple systems and store them in a central location such as HDFS.	
Commit log service	Kafka can be used as an external commit log for distributed systems.	
Event sourcing	A time ordered sequence of events can be maintained through Kafka.	

Introduction to MQTT and Kafka

So, Kafka can be used for various purposes in an organization, for example, as a messaging solution, where millions of messages can be sent and received in real time using Kafka, real time processing, which is also called a stream processing, so Kafka can be used to process a continuous stream of information in real time and pass it to the stream processing system for

its computation. Log aggregation, Kafka can be used to collect physical log files from multiple system and store them to a central location in a database, or in a distributed file system.

So, commit log service, Kafka can be used as an external commit log for distributed systems and event sourcing, that is the time ordered sequence of events can be maintained through Kafka. So, these are some of the details which are given for example, if it is a commit log, so, you can see that from various systems, it can read the log and do the in the distributed environment and store it similarly, log aggregation also requires this kind of details.

(Refer Slide Time: 12:16)

Apache Kafka: a Streaming Data Platform

> Most of **what a business does** can be thought as **event streams**. They are in a

- **Retail system**: orders, shipments, returns, ...
- **Financial system**: stock ticks, orders, ...
- **Web site**: page views, clicks, searches, ...
- **IoT**: sensor readings, ...

and so on.

Introduction to MQTT and Kafka

Now, let us get started. And looking up in more detail that Apache Kafka is a streaming data platform. So, most of the business can be thought of as an event stream, they are often seen in a retail system, where the order shipment returns they are working as the streams or an event streams. Similarly, in a financial system, where stock ticks, orders coming and so on. So, it is also to be seen as the stream system. Similarly, the IoT sensor reading is also having an event streams so, out of this sensor reading your heart to now capture the events in real time. So, they are all event streams, which is seen in various applications.

(Refer Slide Time: 13:12)

Why using both MQTT and Kafka?

If you need to build performant data pipelines, store massive amounts of messages, or integrate different business applications or data centers in real-time — use Kafka.

If you have lots of small applications or devices, running in unwired or unstable environments, exchanging messages in real-time on numerous different channels/topics — use MQTT.

There are two things that make it quite obvious to combine the two technologies:

- the communication structure in topics and
- the publish/subscribe message exchange pattern.

But in which scenarios would you use both Kafka and MQTT together? Lets see in further slides.

Introduction to MQTT and Kafka

Now, the question is why using both MQTT and Kafka, so if you need to build a prominent data pipeline, you store massive amount of messages, or integrate different business applications or data center in the real time, then you have to use the Kafka. So, if you have lots of small applications, or the devices running in an unwired or unstable environment, exchanging message in real time on a numerous different channels and the topics then you have to go for MQTT. So, it is depending upon the scale and the integration into different business application, so Kafka versus MQTT.

Now, can you see why these both are used in different kinds of application and scenarios? So, there are two kinds, there are two things that make it quite obvious to combine these two technologies. For example, in an IoT end to end application, we have to now see the use whether MQTT and Kafka can be used together in building up this kind of IoT system.

So, there are two things that make quite obvious to combine these two technologies. First thing is that communication is structured in topics and publish subscribe message exchange pattern, but in which would you use both Kafka and MQTT. Let us see in more details about that reasoning.

(Refer Slide Time: 14:49)

Use Case: Why using both MQTT and Kafka?

The most popular use case is probably the integration of MQTT devices with backend applications for monitoring, control, or analytics running in the companies' data centers or the cloud.

Imagine you want to send data from different IoT devices to a backend application for machine learning based pattern recognition or analytics. At the same time, the backend application should send back messages to control the IoT device based on the central insights (e.g. send control messages to avoid a device from overheating, ...).

Consequently, MQTT and Kafka are a perfect combination for end-to-end IoT integration from the edge to the business applications and data centers.

The IoT/edge devices can connect to the MQTT broker via MQTT protocol (with all the advantages it has in these environments).

The messages are then forwarded to Kafka to distribute them into the subscribing business applications and the other way around.

Introduction to MQTT and Kafka

Use Case: Why using both MQTT and Kafka?

The most popular use case is probably the integration of MQTT devices with backend applications for monitoring, control, or analytics running in the companies' data centers or the cloud.

Imagine you want to send data from different IoT devices to a backend application for machine learning based pattern recognition or analytics. At the same time, the backend application should send back messages to control the IoT device based on the central insights (e.g. send control messages to avoid a device from overheating, ...).

Consequently, MQTT and Kafka are a perfect combination for end-to-end IoT integration from the edge to the business applications and data centers.

The IoT/edge devices can connect to the MQTT broker via MQTT protocol (with all the advantages it has in these environments).

The messages are then forwarded to Kafka to distribute them into the subscribing business applications and the other way around.

Introduction to MQTT and Kafka

So, most popular use case is probably the integration of MQTT devices with the backend application for monitoring control analytics, running in the companies' data center or the cloud then both are needed. So, imagine you need to send the data from different IoT devices to the backend application for doing machine learning based pattern recognition or analytics. So, at the same time, the backend application should be sent back the message to control the IoT device based on the central insight that is sent the control message to avoid a device from overheating and so on.

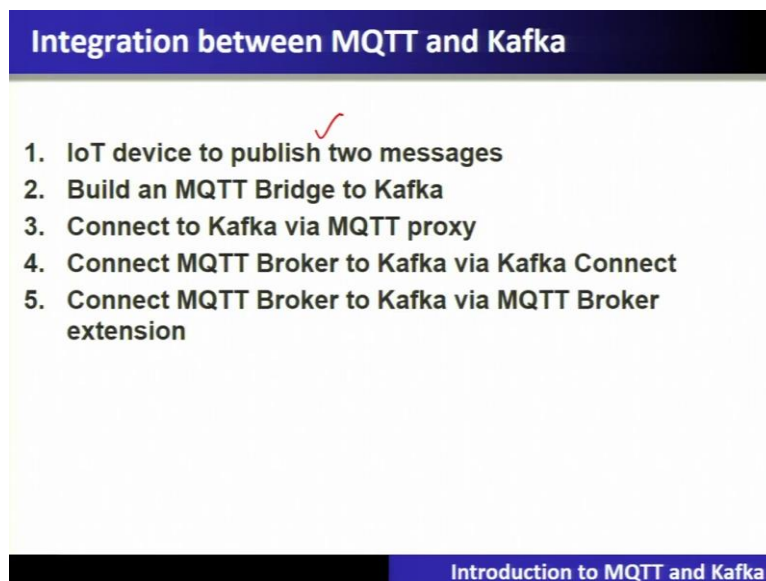
So, consequently MQTT and Kafka are the perfect combination for end to end IoT integration from edge to the business application and the data center. So, that means you can think of in this manner, a scenario, so, here there are things this is also called an IoT. Now, to

connect this IoT with the edge you often require machine to machine communication that is in the form of MQTT.

Now, these are various number of devices and the edge now, when it wants to connect to the cloud for doing the machine learning, then you require the data ingestion of a higher level if the application is quite big, and here you have to use Kafka. So, both MQTT and Kafka both are used in this particular situation.

And that situation is well stated here that, that Kafka and MQTT are the perfect combination for end to end IoT integration from the edge to the business applications and the data center. So, IoT edge devices can connect to the MQTT broker via the MQTT protocol and the messages are then forwarded to the Kafka to distribute them into the subscribing business applications.

(Refer Slide Time: 17:10)



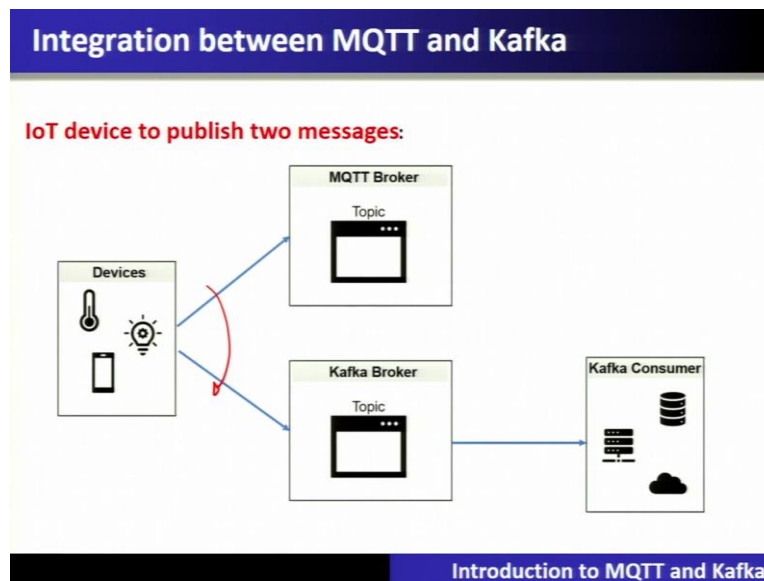
The slide is titled "Integration between MQTT and Kafka" in a dark blue header. It contains a numbered list of five steps. A red checkmark is drawn above the first step. The footer of the slide is a dark blue bar with the text "Introduction to MQTT and Kafka" in white.

1. IoT device to publish two messages ✓
2. Build an MQTT Bridge to Kafka
3. Connect to Kafka via MQTT proxy
4. Connect MQTT Broker to Kafka via Kafka Connect
5. Connect MQTT Broker to Kafka via MQTT Broker extension

Introduction to MQTT and Kafka

So, let us see the integration between MQTT and Kafka in different ways of integrating both MQTT and Kafka. So, let us see that IoT devices to publish messages we have to build an MQTT bridge to the Kafka.

(Refer Slide Time: 17:28)



So, let us say that IoT device used to publish the messages if you are keeping both MQTT broker and Kafka broker then this particular message two messages are needed to be copied both in MQTT broker and Kafka.

(Refer Slide Time: 17:44)

Integration between MQTT and Kafka

IoT device to publish two messages:

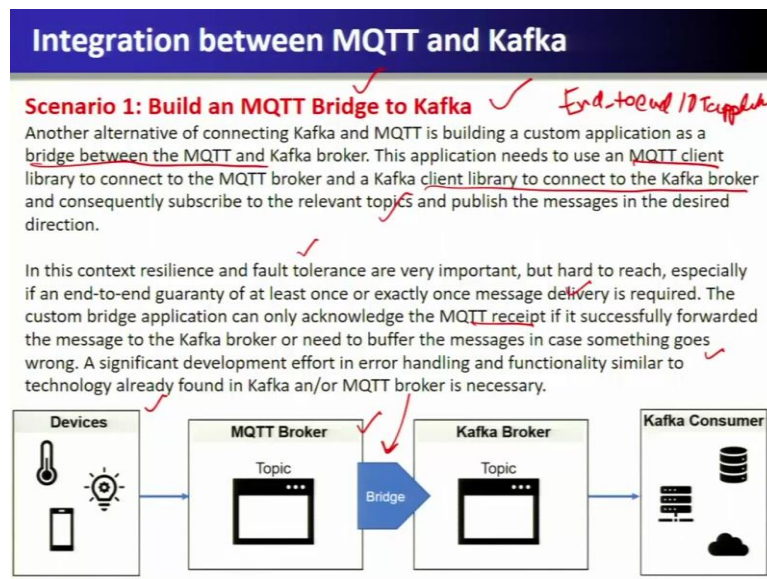
The IoT device can publish two messages — one to the topic of the MQTT broker and a second one to the topic of the ~~Kafka~~ broker. This has several drawbacks:

- The IoT device needs to check the delivery guarantees of both protocols and it must be ensured that the message is received by both or not at all. A lot of investment in error handling must be done.
- Additionally, most IoT devices are lightweight. Sending two messages with two different protocols is a huge overhead. Most IoT devices might have not even the possibility to connect to Kafka natively.
- Kafka is not designed to handle a massive amount of different topics with millions of different devices. A full-blown IoT scenario with this integration option could lead to issues on the Kafka broker side.

The bottom right corner of the slide has a blue bar with the text 'Introduction to MQTT and Kafka'.

So, IoT device to push two messages which is shown one to the topic of MQTT broker and second the topic of Kafka broker. So, it has several drawbacks, why, because these devices, IoT devices need to check the delivery guarantees, so there is no integration between MQTT and Kafka.

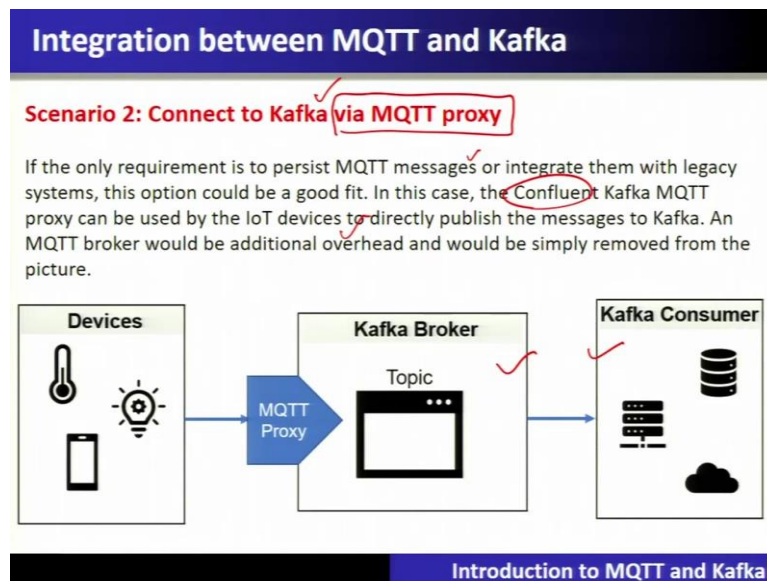
(Refer Slide Time: 18:04)



Now, we have to let us see that how to build MQTT bridge to the Kafka for end to end IoT application? For end to end IoT application the first scenario is to build a pipeline using MQTT bridge to a Kafka connector so, alternative of connecting Kafka is to build custom application is as the bridge between MQTT, and the Kafka broker. This application needs to use MQTT client library to connect to the MQTT broker and Kafka client library to connect to the Kafka broker and consequently, subscribe to the relevant topics and publish the message in the desired direction.

So, in this context, resilience and fault tolerance are very very important but hard to reach especially if the end to end guarantee of at least once or exactly once message delivery is required. So, the custom bridge application can only acknowledge the MQTT received if it is successfully forwarded the message to the Kafka broker or need to buffer the message in case something goes wrong a significant development is an error handling functionality similar to the technology already found in MQTT is required. So, this is a very challenging if let us say that you want to make a bridge with the MQTT client connecting to the Kafka broker.

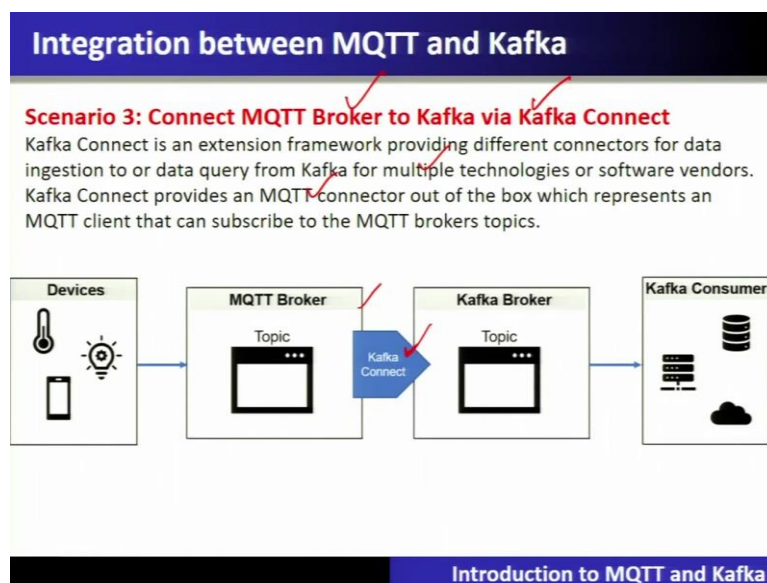
(Refer Slide Time: 19:39)



The second scenario is to connect to the Kafka via MQTT proxy. So, these we have failed a lot of companies are now providing this kind of MQTT proxy to connect to this Kafka in this particular pipeline. So, the only requirement is to persist MQTT messages or to integrate them with the legacy system this option could be a good fit.

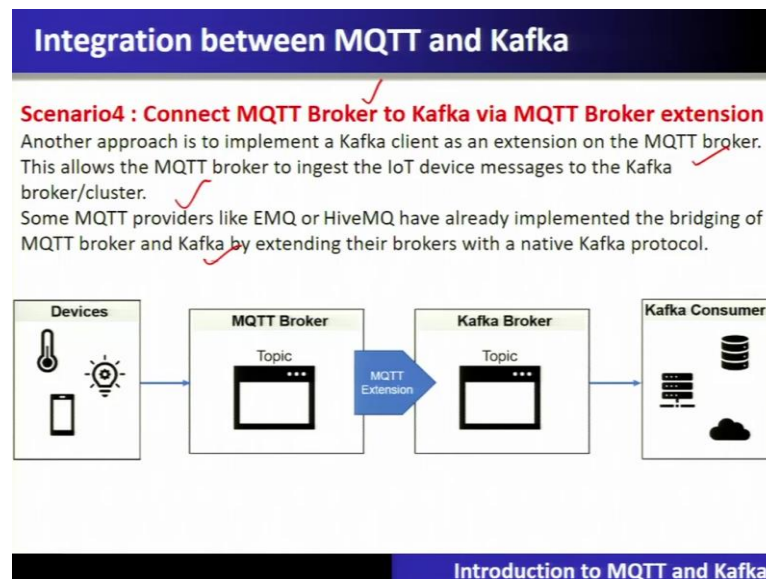
So, in this case, there are various proxy solutions are available from different companies like Confluent and many other companies, which uses the Kafka MQTT proxy can be used by the IoT devices to directly publish the messages to the Kafka broker here in this case, so MQTT broker would be an ideal overhead in this case.

(Refer Slide Time: 20:29)



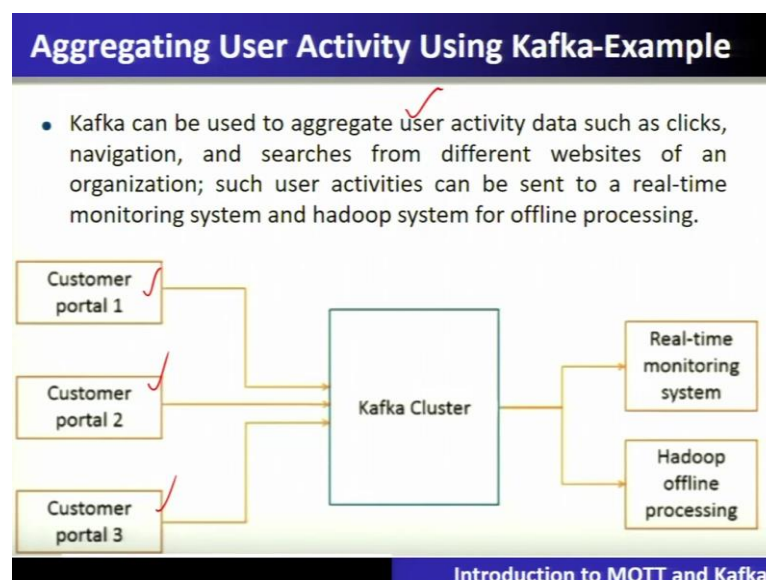
So, another scenario will be that to connect the MQTT broker to the Kafka via the Kafka Connect, so Kafka Connect is an extension framework providing different connectors for the data ingestion to or data query from the Kafka for multiple technologies or vendors so, Kafka Connect provides an MQTT connector out of the box, which represents MQTT declined that can subscribe to MQTT broker topics.

(Refer Slide Time: 21:00)



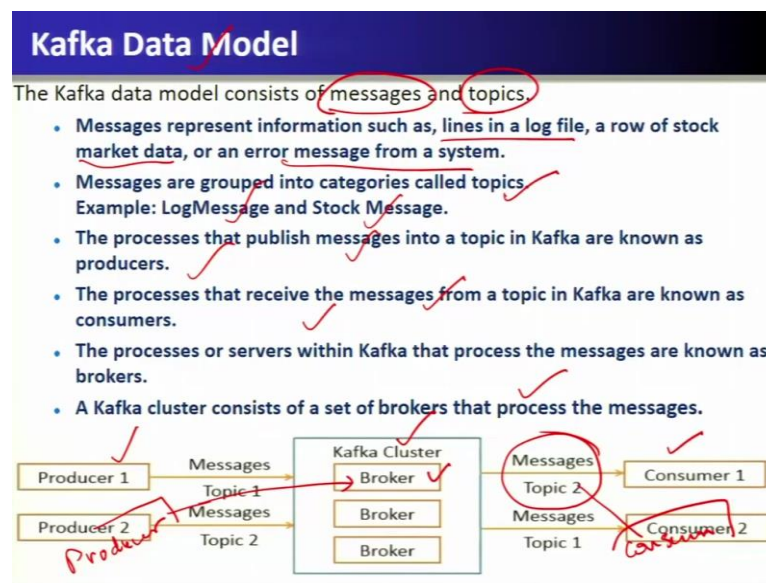
Fourth scenario is to connect MQTT broker to the Kafka via MQTT broker extension. So, this is another way of providing this kind of connectivity. Now, some MQTT providers like EMQ Hive have already implemented the bridging of MQTT broker and Kafka by extending their brokers with the native Kafka protocols.

(Refer Slide Time: 21:27)



Now let us see the applications the user activities using Kafka example so, Kafka can be used to aggregate the user activity, data such as clicks, navigation, searches from various websites of an organization such user activities can be sent to the real time monitoring system and a big data system for offline. So, here you can see the customer's portal are now linked to the Kafka cluster. So, these customer portals are sending different events or the messages streamed the messages from different websites. And this is sent to the Kafka for real time monitoring and offline computation.

(Refer Slide Time: 22:13)



So, let us see the Kafka data model. Kafka data model consists of the messages and the topics. So, messages represents the information such as lines in the log file, or row in the stock market data or an error message from the system. So, messages are grouped into the categories which are called the topics. For example, error message is one topic, the stock market will be another topic and so on. So, therefore, the processes that publish messages into the topic in the Kafka are known as the producers.

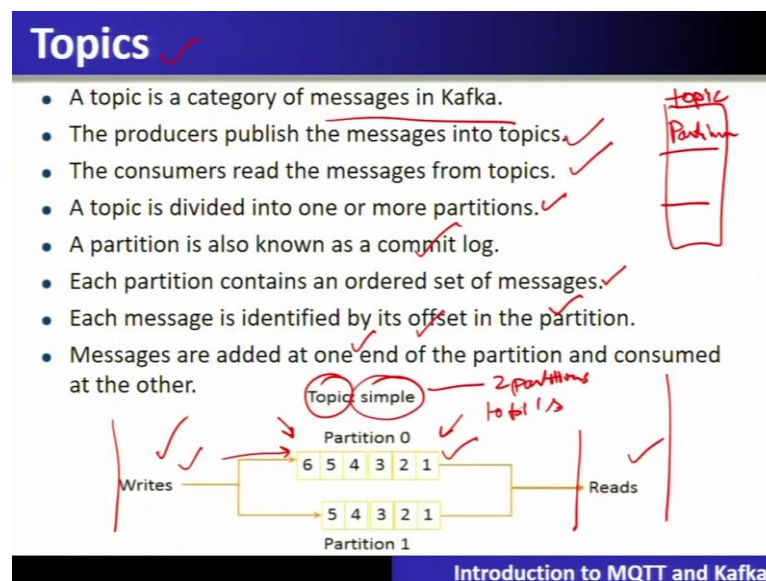
So, the processes that receive the message from the topic into Kafka are known as the consumers. So, the processes or the servers within the Kafka that process the message are known as the broker. So, Kafka cluster consists of a set of brokers that process the messages. So, this is the Kafka data model.

Again, let me summarize, so Kafka data model consists of the producer on one end, and consumer on the other end, now the producers of different type can be integrated using the topics, so, they will be producing the messages of different applications, for example, log

message or stock market message or if the messages are too much in numbers, then basically they are divided into the topic.

Similarly, the consumer has to subscribe for these specific topics, so when the producer produces the messages, it will be handled by the particular broker. That is called Kafka broker. So, this particular broker is the process or the software which runs on the cluster. So, that is called Kafka cluster. So, Kafka cluster becomes the infrastructure, and broker becomes the software which runs on it in a scalable manner.

(Refer Slide Time: 24:18)

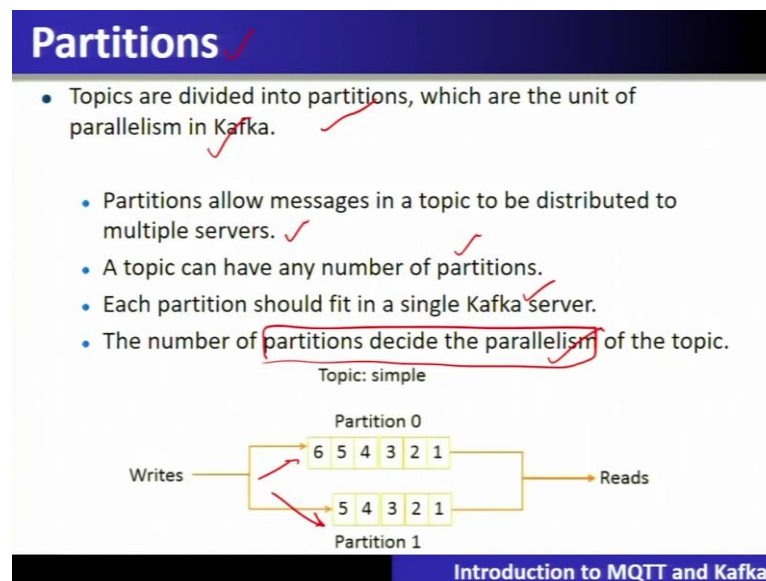


So, let us go into looking up how the messages is further indexed as a form of the topics. So, topic is a category of a message in Kafka. So, producer publish the message into the topics and the consumer read the message from the topics. And the topic is divided into one or more partition. For example, if the number of messages in a particular topic is too much, then it will be divided into the topic is divided into the partitions and the partition also known as the commit log, each partition contains an ordered set of messages so, each message is identified by its offset in the partition.

Now, the messages are added at one end of the partition and consumed at the other end. So, here in this particular example, you can see that the rights that is the producer will write the messages into the topic and the topics is divided into let us say to partition for the same topic which is called a simple is divided into two partition. So, this particular message will go from one end and this particular place where the message inserted called the offset in the partition.

So, messages are added at one end of the partition and consumed from the other end of the partition that is shown in this particular example of a topic simple which is having two partitions. So, this particular topic is having two partitions and the producer writes the messages in these particular partition from one end and the consumer reads from the partition of a particular topic from the other side or the other end. So, two ends, they have to synchronize automatically here in the Kafka.

(Refer Slide Time: 26:26)



Now, let us talk about the partition. So, topics are divided into the partition, which is a unit of parallelism in the Kafka. So, partitions allow the messages in a topic to be distributed across multiple servers. So, the topic can have any number of partitioned, each partition should fit into a single Kafka cluster or a server the number of partitions decides the parallelism of the topic.

So, here you can see that simultaneously the right can happen on both the partition, why, because these partition are separately handled on the distributed servers. So, they can go in parallel, had it been only one partition, so, one at a time the message will be there. So, therefore, the number of partition will decide the parallelism of that particular topic and therefore, scalability and efficiency is also ensured.

(Refer Slide Time: 27:23)

Partition Distribution

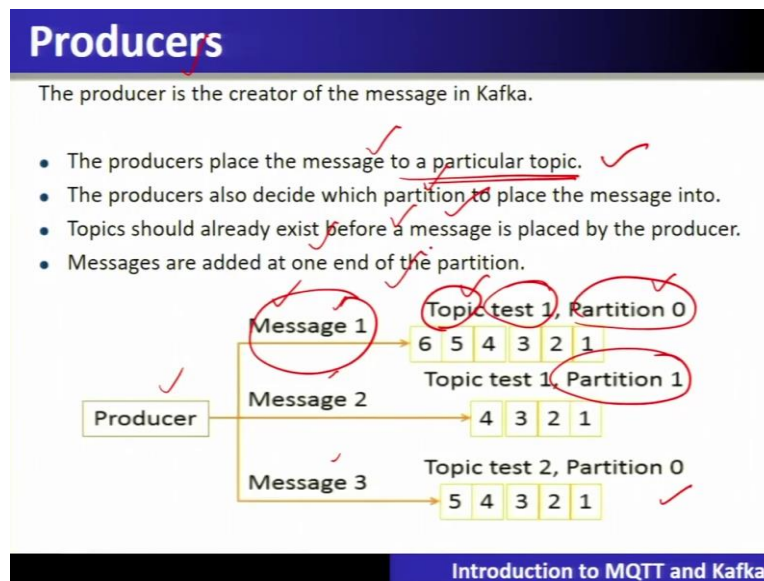
- Partitions can be distributed across the Kafka cluster.
- Each Kafka server may handle one or more partitions.
- A partition can be replicated across several servers for fault-tolerance.
- One server is marked as a leader for the partition and the others are marked as followers.
- The leader controls the read and write for the partition, whereas, the followers replicate the data.
- If a leader fails, one of the followers automatically become the leader.
- Zookeeper is used for the leader selection.

Introduction to MQTT and Kafka

Now, partition distribution. So, partition can be distributed across the Kafka cluster. So, Kafka cluster may handle one or more partitions. So, a partition can be replicated along several servers, that is to ensure the fault tolerance. So, replication of this particular partition ensures the fault tolerance, we have also seen the similar design for in no SQL database where the each key value data item was stored using the replication factor of three. So, here also the partition is replicated across the several servers to ensure the fault tolerance.

So, one server is marked in this case for the partition as the leader and all others are the followers. So, the leader controls the Read and Write for the partition whereas, the followers replicate the data. Now, if the leader fails, the one of the follower will automatically elected as the leader and for any leader election, the service which is called zookeeper is used here in this case.

(Refer Slide Time: 28:42)

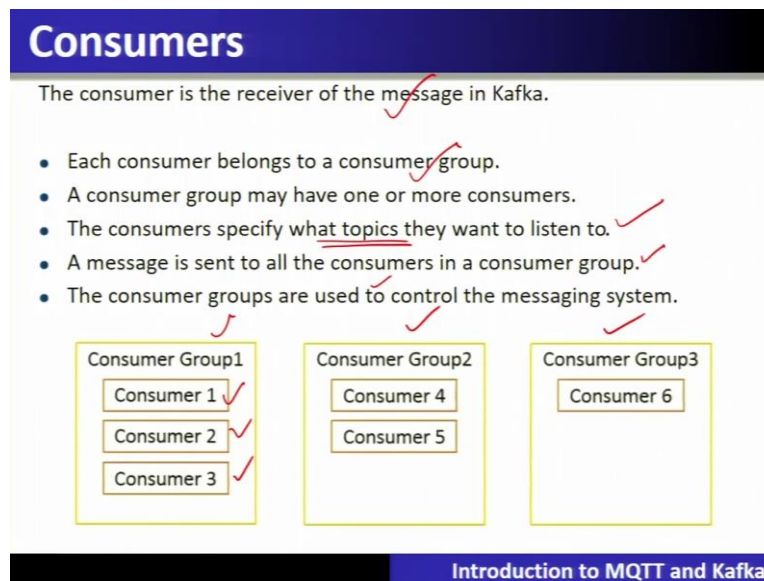


Let us talk about the producers. So, producer is the creator of the message in Kafka. So, producers place the message to a particular topic and producers also decide which partition to place the message into and the topics should already exist before the message is placed into the producer. So, messages are edit at one end of the partition. So, in this example, you can see that as far as the messages, there are three different messages and every message is having one partition is shown.

For example, for message 1, there is a topic which is called test 1 and similarly, for message 1 the topic test 1 is having partition 0, partition 1, so, there are two partition for a message 1 is shown in this particular example, similarly, for message 2 and for message 3 are different. So, therefore, what is shown over here is that the producer has to know which message and which topic and for that topic which partition it has to now right. So, again I am repeating this that producer is the creator of the message.

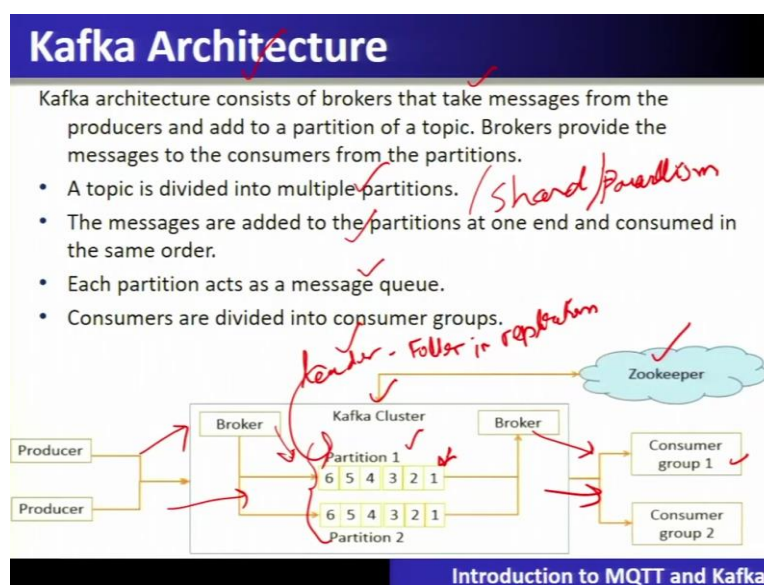
Now, the producer plays the message on it particular topic. So, message on a particular topic is to be decided first. So, producer also has to decide which partition to place the message into. So, topic should already exist before the message is placed. So, messages are added at one end of the partition.

(Refer Slide Time: 30:18)



Let us talk about the consumers. So, consumer is the receiver of the message in the Kafka. So, each consumer belongs to the consumer group. So, that is what is particular thing shown in this example. So, consumer group maybe having one or more consumers like which is shown here in this particular example. So, consumers specify what topic they want to listen to. So, a message is sent to all the consumers in the consumer group and the consumer group are used to control the messaging system. So, therefore according to the topic, these consumers are grouped, so, that they will be receiving the messages based on these particular topics on which they are listening for.

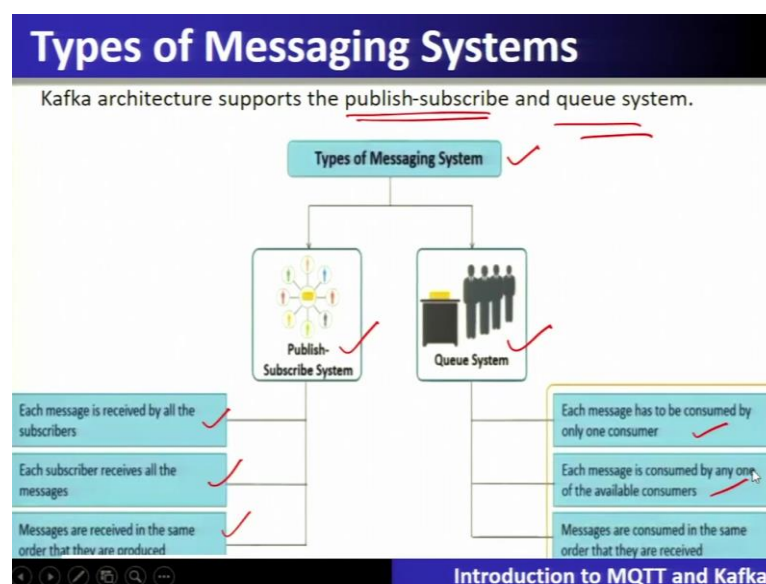
(Refer Slide Time: 31:05)



So, let us go ahead with the Kafka architecture. So, Kafka architecture consists of the brokers, that takes the message from the producer and add it to the partition of a topic. So, brokers provide the message to the consumer from that particular partition. So, a topic is divided into the multiple partition that is also called sharding. And it will introduce the parallelism here in this case.

Now, messages are added to the partition at one end consumed, this is the consumer consumption from the other end. So, each partition acts as a message queue here and the consumers are divided into the consumer groups. Now, as far as in the Kafka cluster, as we have seen, these partitions are also having replicated and they have to follow a leader follower model in the replication. Now, if that particular leader is down for a particular partition, then it has to now use the zookeeper for leader election.

(Refer Slide Time: 32:26)

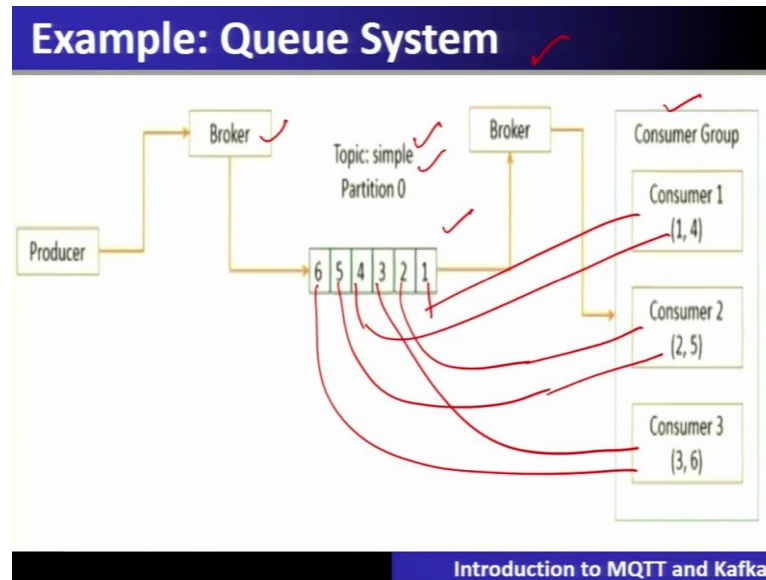


So, let us see the types of messaging systems so, Kafka architecture supports the publish subscribe and queuing system both. So, the types of messaging system here it is the publish subscribe system that we have already explained, it is also a queueing system because of the discipline in which the messages are consumed. So, as far as the publish subscribe system, each message is received by all the subscribers and each subscriber receives all the messages and the messages are received in the same order in which they are produced.

So, therefore, publish subscribe system is implemented again with the help of publisher and that is being coordinated with the help of a broker. So, it is also a queueing system that is

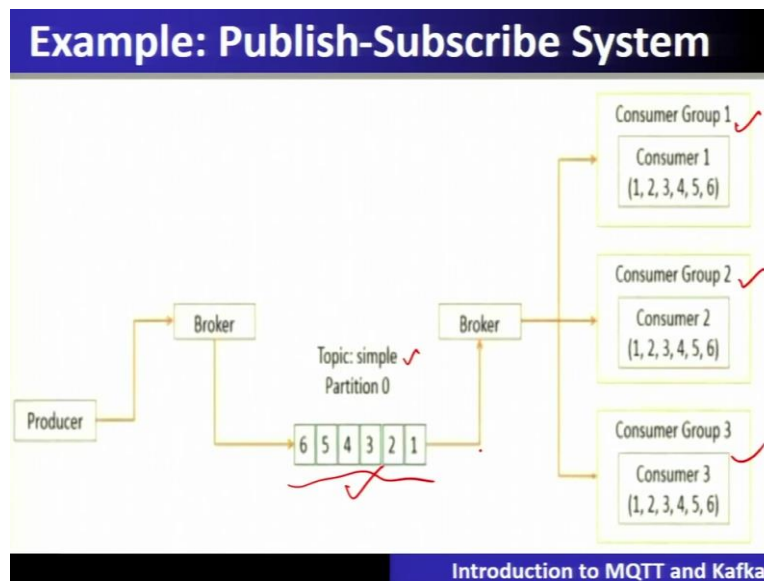
each message has to be consumed by only one consumer each message is consumed by any one of the available consumers and so on.

(Refer Slide Time: 33:20)



So, let us take this queue example in the Kafka. So, the producer produces producer writes a particular message for a particular topic into a particular partition with the help of a broker. Similarly, consumer group in a consumer consumes these particular messages, which they have subscribed for a particular topic out of that particular partition. So, in this example, you can see that consumer 1 will now consume 1 and 4 from a particular partition. Consumer 2 it will consume 2 and 5 from this particular partition, whereas, consumer 3 will consume 3 and 6 from this particular partition. So, therefore, you can see in this particular manner that the messages are consumed only once by any of these consumer into the consumer group.

(Refer Slide Time: 34:23)



Now, therefore, publish-subscribe system let us take an another example. So, there is this example so, that the partition 0 has generated out of this topic simple whereas, the broker has to coordinate in another mechanism where consumer group every consumer group is having only one consumer so, 1, 2, 3.

So, therefore, all the consumer group will receive the entire partition messages that is what is shown. So, therefore, if a consumer group has more than one consumers, so they will be dividing these messages so, any one of these consumer in the consumer group is good enough to receive these messages and messages will not be given again.

(Refer Slide Time: 35:08)

Brokers

Brokers are the Kafka processes that process the messages in Kafka.

- Each machine in the cluster can run one broker.
- They coordinate among each other using Zookeeper.
- One broker acts as a leader for a partition and handles the delivery and persistence, where as, the others act as followers.

The bottom right corner of the slide features the text 'Introduction to MQTT and Kafka'.

So, let us see about the brokers. So, brokers are the Kafka processes that process the messages in the Kafka. So, each machine in the cluster can run one broker, so they coordinate among each other using Zookeeper. So, one broker acts as the leader for the partition and handles the delivery and persistence where others act as the followers.

(Refer Slide Time: 35:30)

Kafka Guarantees

- Kafka guarantees the following:
 1. Messages sent by a producer to a topic and a partition are appended in the same order ✓
 2. A consumer instance gets the messages in the same order as they are produced. ✓
 3. A topic with replication factor N, tolerates upto N-1 server failures. ✓

Introduction to MQTT and Kafka

So, Kafka guarantees the following the message sent by the producer to a topic and the partitions are appended in the same order. So, consumer instance gets the message in the same order they are produced. So, the topic is the replication factor N, tolerates up to N minus 1 server failures.

(Refer Slide Time: 35:49)

Replication in Kafka

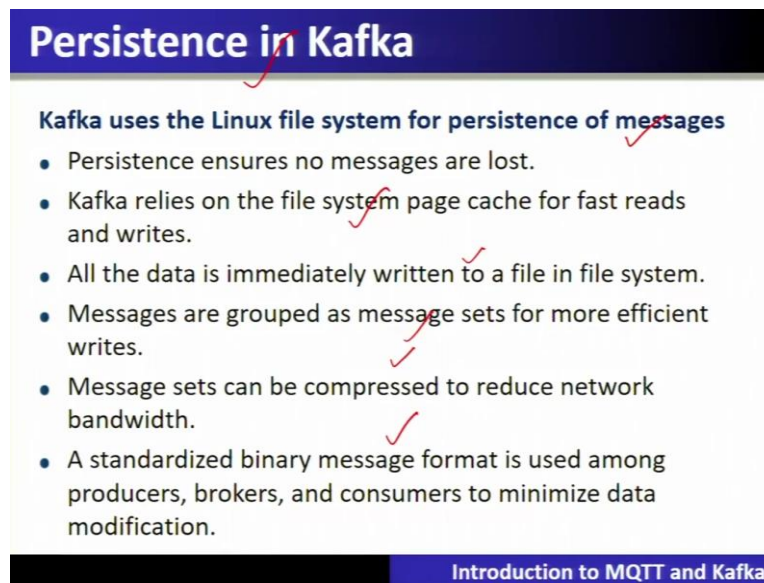
Kafka uses the primary-backup method of replication.

- One machine (one replica) is called a leader and is chosen as the primary; the remaining machines (replicas) are chosen as the followers and act as backups. ✓
- The leader propagates the writes to the followers. ✓
- The leader waits until the writes are completed on all the replicas. ✓
- If a replica is down, it is skipped for the write until it comes back. ✓
- If the leader fails, one of the followers will be chosen as the new leader; this mechanism can tolerate n-1 failures if the replication factor is 'n' ✓

Introduction to MQTT and Kafka

So, replication in Kafka, let us understand this concept. So, Kafka uses the primary backup method for replication. That is one machine one replica is called a leader and a student as the primary and the remaining machines or the replicas chosen as the followers and acts as the backup. So, leader propagates the right to the followers and the reader waits until the rights are completed on all the replicas. So, if the replica is down, it is a skip for the right until it comes back. If the leader fails, one of the follower will be choosing as the new leader and this mechanism can tolerate up to $n - 1$ failures, if the replication factor is n .

(Refer Slide Time: 36:29)



Persistence in Kafka

Kafka uses the Linux file system for persistence of messages

- Persistence ensures no messages are lost.
- Kafka relies on the file system page cache for fast reads and writes.
- All the data is immediately written to a file in file system.
- Messages are grouped as message sets for more efficient writes.
- Message sets can be compressed to reduce network bandwidth.
- A standardized binary message format is used among producers, brokers, and consumers to minimize data modification.

Introduction to MQTT and Kafka

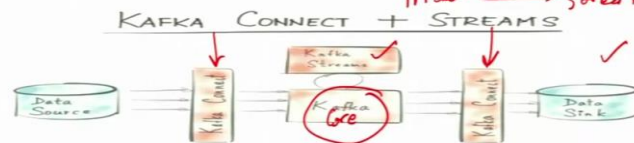
Let us see about the persistence how that is done in Kafka. So, Kafka uses Linux file system for persistence of the messages, so persistent ensure no messages are lost. So, Kafka relies on a file system page cache for fast reads and writes. So, all the data is immediately written to a file in a file system and the message or group as the message set for more efficient writes. Message sets can be compressed to reduce the network bandwidth, a standardized binary message format is used among the producers, brokers, consumers to minimize the data modification.

(Refer Slide Time: 37:07)

Apache Kafka: a Streaming Data Platform

> **Apache Kafka** is an open source streaming data platform (a new category of software!) with **3 major components**:

1. **Kafka Core**: A **central hub** to **transport** and **store** event streams in real-time. *IoT data Analytics*
2. **Kafka Connect**: A **framework** to **import** event streams from other source data systems into Kafka and **export** event streams from **Kafka** to destination data systems.
3. **Kafka Streams**: A **Java library** to **process** event streams live as they occur. *data streams*



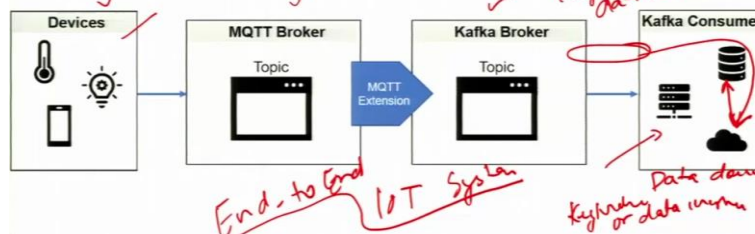
Introduction to MQTT and Kafka

Integration between MQTT and Kafka

Scenario4 : Connect MQTT Broker to Kafka via MQTT Broker extension

Another approach is to implement a Kafka client as an extension on the MQTT broker. This allows the MQTT broker to ingest the IoT device messages to the Kafka broker/cluster.

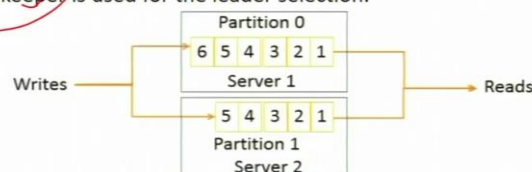
Some MQTT providers like EMQ or HiveMQ have already implemented the bridging of MQTT broker and Kafka by extending their brokers with a native Kafka protocol.



Introduction to MQTT and Kafka

Partition Distribution

- Partitions can be distributed across the Kafka cluster.
- Each Kafka server may **handle** one or more partitions.
- A partition can be **replicated** across several servers for fault-tolerance.
- One server is marked as a **leader** for the partition and the others are marked as followers.
- The leader controls the read and write for the partition, whereas, the followers replicate the data.
- If a **leader** fails, one of the followers automatically become the leader.
- **Zookeeper** is used for the leader selection.



Introduction to MQTT and Kafka

Now, let us see the streaming data platform using Kafka. Now, as we have seen that the Kafka is open source, the streaming data platform and this is a category of the software. So, as far as the IoT is concerned, this particular data analysis is called the hot so, hot data analytics is the streaming data analytics which is to be performed.

Now, let us see that this hot data analytics or streaming data analytics requires the support of Apache Kafka like open source. So, from that aspect, that IoT system or IoT platforms often support hot data analytics. So, therefore Kafka is a mechanism which has to be connected with an IoT system. So, let us see about Apache Kafka, how that can become a streaming data platform. So, you know that Kafka has the following major components. There is a Kafka core. This is a Kafka core, so the central hub to transport and store the events, streams in real time, this does that. So, there is a Kafka Connect.

So, you can see that there are two Kafka Connects are there, one is at the producer, the other is at the sink. So, the framework to import the event streams from the other source data system into the Kafka and export the event stream from Kafka to the destination data system is facilitated by this Kafka Connect. So, Kafka streams is a library to process the event stream live as they occur, you will be knowing that the streams when you talk about are the data streams. So, if it is an IoT data, it will be sending the bids or let us say the data in these particular streams. So, this is called a stream of data.

Now, this stream of the data is arriving and there will be a sliding window. So, most of these stream data analytics algorithm using sometimes the sliding window. So, inside the window, you will be able to monitor or do the analytics inside this particular window whether it is everything and so on. So, therefore, a lot of libraries are available for doing the stream processing here in this particular scenario.

Let us see more details about this. Now, you can see that MQTT broker is used to connect the IoT devices based on the publish subscribe system and this is called the MQTT broker. So, MQTT broker will bring the data into the edge computing system and this particular edge computing system will connect to the cloud and if let us say the cloud requires a data ingestion, for end to end or storage into the cloud system, then you require the Kafka broker.

So, once you say that, it is an IoT end to end IoT system so, end to end IoT system means that on one end, if it is the device is the other end, you have to now store that data into the data centers in the cloud, so for that you require the data ingestion mechanisms like Kafka will

perform the data ingestion and will store will enable it to store it into the database whether it is the key value store or data warehouse.

Now, this Kafka is also once it is stored and this particular data can be used for machine learning. So, if you want to build a machine learning pipeline that is on one end, it is a device the other end is the machine learning platform, which requires the data preparation and after data preparation data has to be used for machine learning implementation. So, this particular combination, or integration of MQTT and Kafka, we have shown here in this particular lecture, that both are there.

And as far as the Kafka is concerned, Kafka has the capacity or capabilities to store the data or to handle this publish subscribe system in the form of Kafka cluster and Kafka cluster is a scalable, so, you have also seen that the partitions are replicated to ensure that the data is fault tolerant, manner, all these messaging publish subscribe system is working.

Further on, we have also shown you that to have this particular replication, one of them one of the partitions in the replicated system is defined or elected as a leader. So, it has to handle all the receiving of the messages and also giving the messages to the consumers. So, therefore, for this leader election, it uses the zookeeper.

(Refer Slide Time: 43:23)

Conclusion

- Kafka is a high-performance, real-time messaging system. *data ingestion Kafka*
- Kafka can be used as an external commit log for distributed systems. *IoT - hot data Analytics*
- Kafka data model consists of messages and topics. *data ingestion*
- Kafka architecture consists of brokers that take messages from the producers and add to a partition of a topics.
- Kafka architecture supports two types of messaging system called publish-subscribe and queue system.
- Brokers are the Kafka processes that process the messages in Kafka.

Introduction to MQTT and Kafka

So, let us conclude this particular lecture, what we have shown is that Kafka is a very high performance, real time messaging system. And for IoT, we have also seen that it is used for hot data analytics and for that, it requires the support of data ingestion like Kafka. We have also seen that Kafka can be used as an external commit log for distributed system.

So, Kafka model often consists of messages and topics. So, Kafka architecture consists of the broker that take the message from the producer and add to the partition of a topic. So, Kafka architecture also supports two types of messaging system, they are called publish subscribe, and queuing system. So, the brokers are the Kafka processes that process the messages in the Kafka. So, with this, let us conclude this particular session. Thank you very much. Thank you.