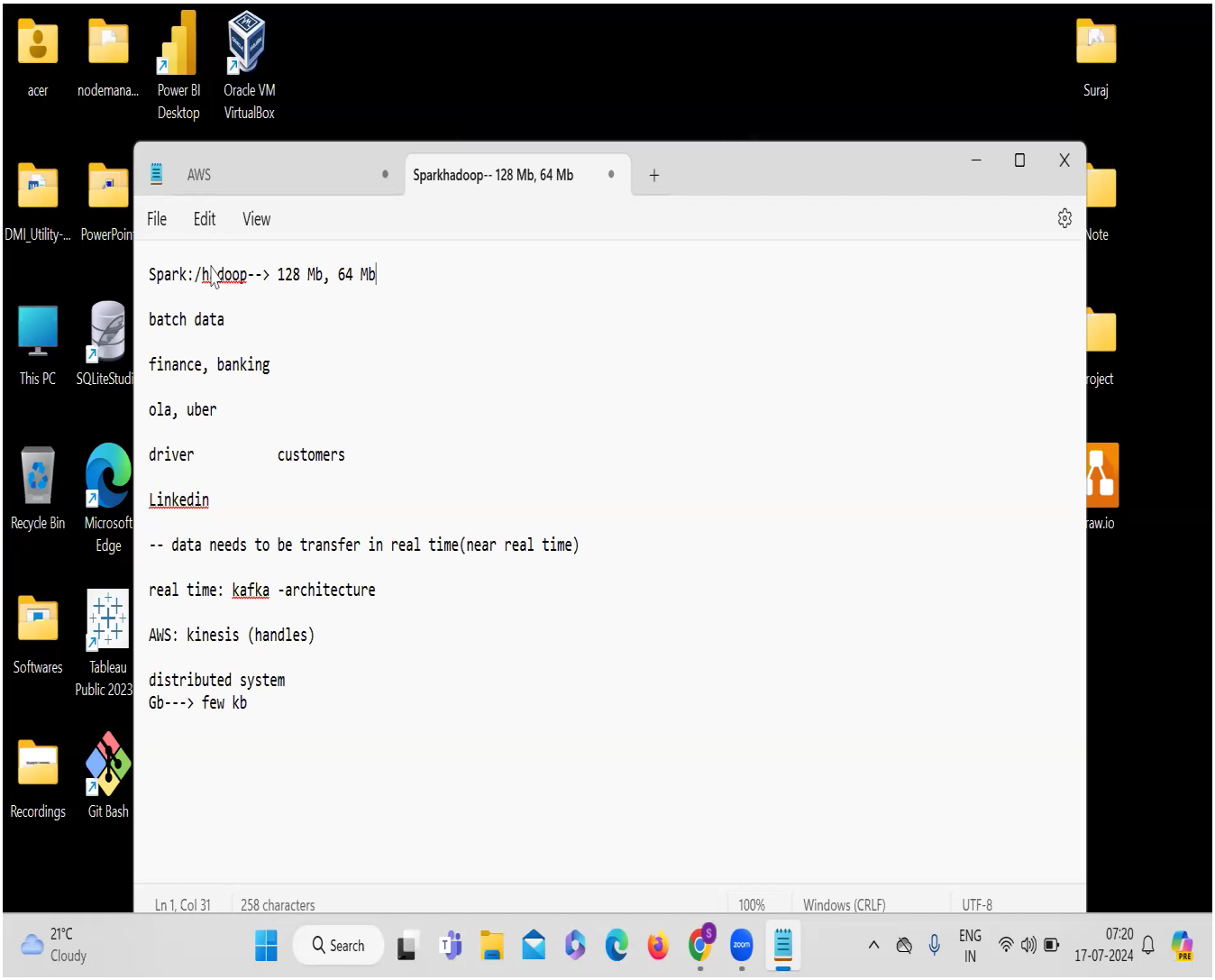
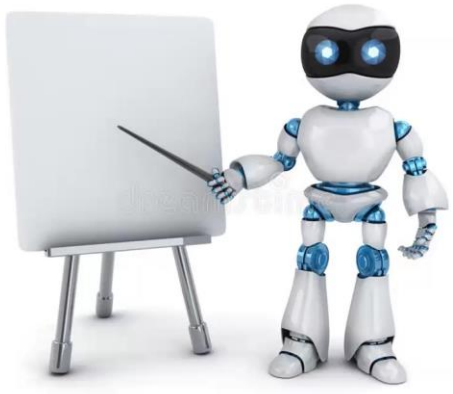
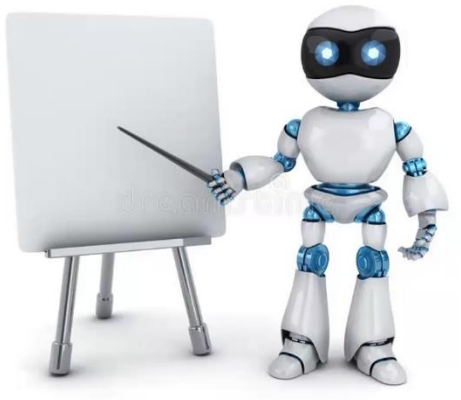




block/chunk size- few kb

Kafka for Big Data

Overview



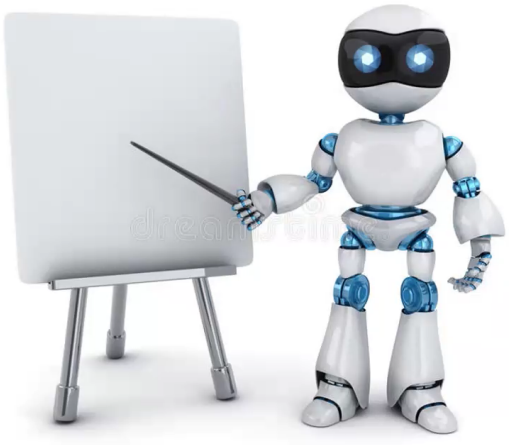
block/chunk size- few kb

Kafka for Big Data

Overview

Producer --> Consumer

Datamites



block/chunk size- few kb

Kafka for Big Data

Overview

publis

Producer --> Broker ---> Consumer
Cluster(message)

Topic 1
Topic 2
Topic 3

far beyond the ability of a single server. Additionally, partitions can be replicated, such that different servers will store a copy of the same partition in case one server fails.

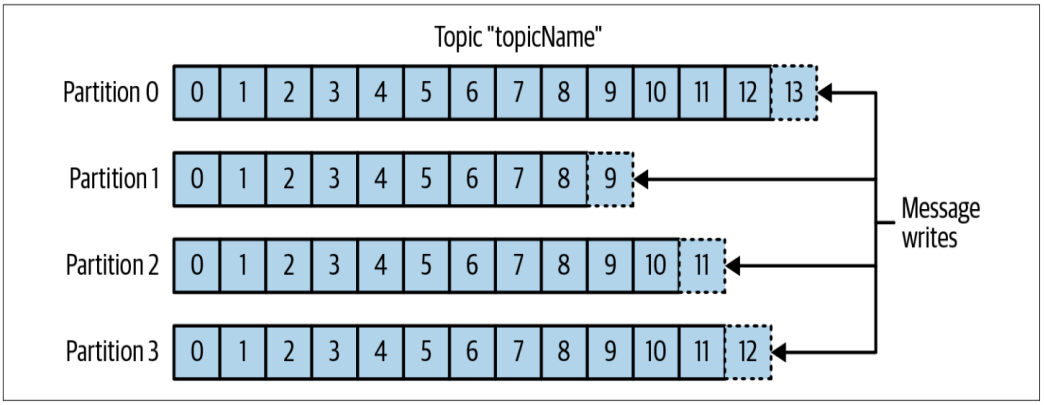


Figure 1-5. Representation of a topic with multiple partitions

The term *stream* is often used when discussing data within systems like Kafka. Most

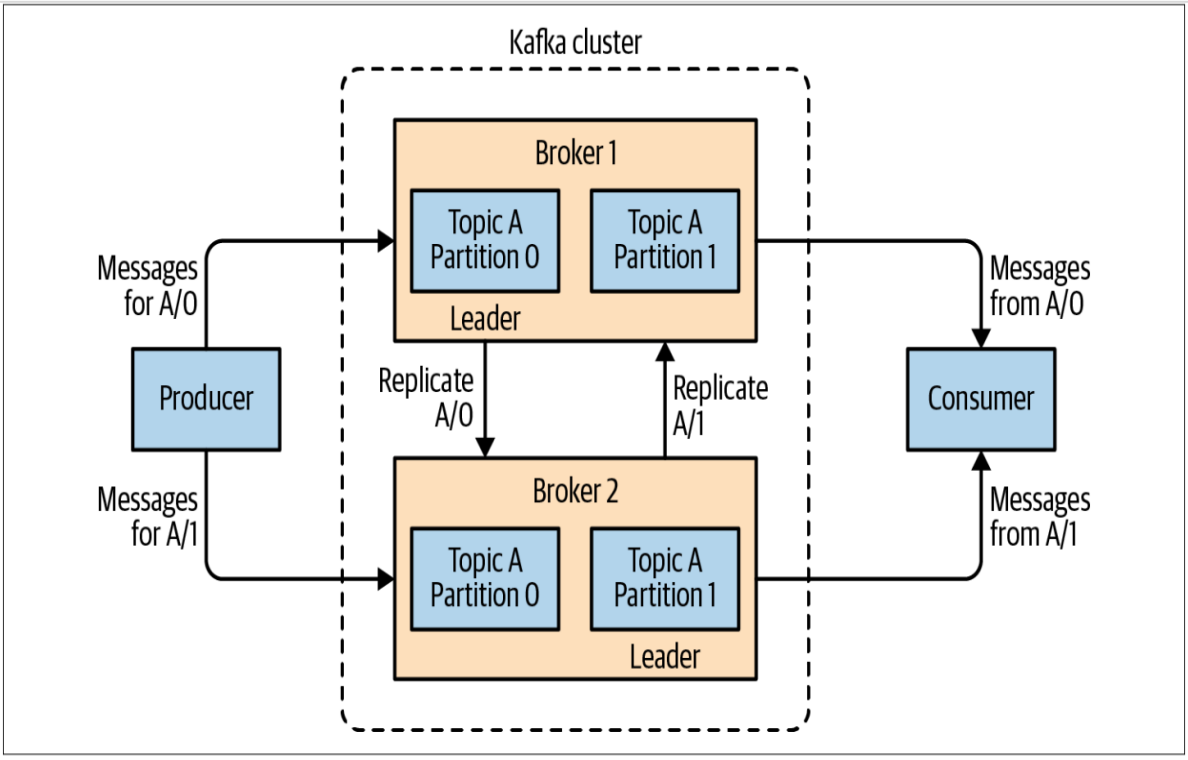


Figure 1-7. Replication of partitions in a cluster

defines a minimum amount of data available at any time. Individual topics can also be configured with their own retention settings so that messages are stored for only as long as they are useful. For example, a tracking topic might be retained for several days, whereas application metrics might be retained for only a few hours. Topics can also be configured as *log compacted*, which means that Kafka will retain only the last message produced with a specific key. This can be useful for changelog-type data, where only the last update is interesting.

Multiple Clusters

As Kafka deployments grow, it is often advantageous to have multiple clusters. There are several reasons why this can be useful:

- Segregation of types of data
- Isolation for security requirements
- Multiple datacenters (disaster recovery)

When working with multiple datacenters in particular, it is often required that messages be copied between them. In this way, online applications can have access to user activity at both sites. For example, if a user changes public information in their profile, that change will need to be visible regardless of the datacenter in which search

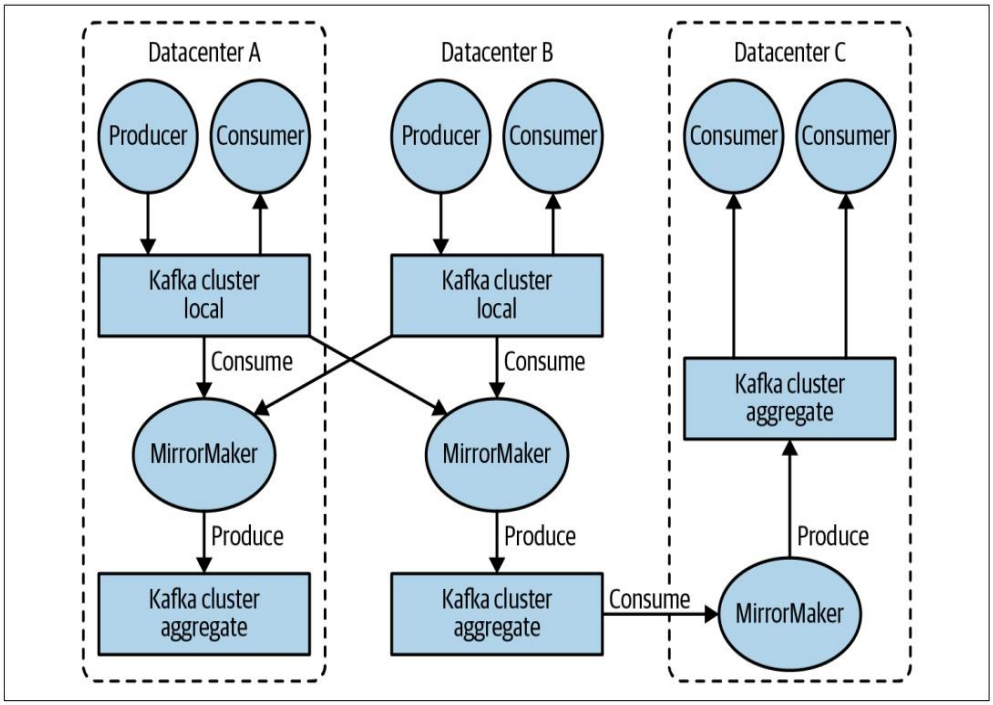


Figure 1-8. Multiple datacenters architecture

direct connections of any sort. Components can be added and removed as business cases are created and dissolved, and producers do not need to be concerned about who is using the data or the number of consuming applications.

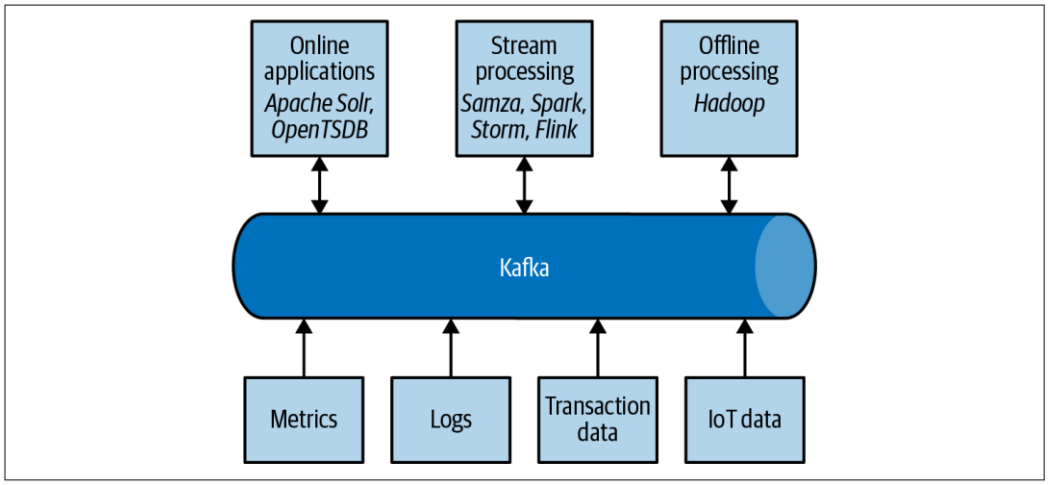


Figure 1-9. A big data ecosystem

Activity tracking

The original use case for Kafka, as it was designed at LinkedIn, is that of user activity tracking. A website's users interact with frontend applications, which generate messages regarding actions the user is taking. This can be passive information, such as

page views and click tracking, or it can be more complex actions, such as information that a user adds to their profile. The messages are published to one or more topics, which are then consumed by applications on the backend. These applications may be generating reports, feeding machine learning systems, updating search results, or performing other operations that are necessary to provide a rich user experience.

Messaging

Kafka is also used for messaging, where applications need to send notifications (such as emails) to users. Those applications can produce messages without needing to be concerned about formatting or how the messages will actually be sent. A single application can then read all the messages to be sent and handle them consistently, including:

- Formatting the messages (also known as *decorating*) using a common look and feel
- Collecting multiple messages into a single notification to be sent
- Applying a user's preferences for how they want to receive messages

Using a single application for this avoids the need to duplicate functionality in multiple applications, as well as allows operations like aggregation that would not otherwise be possible.

Metrics and logging

Kafka is also ideal for collecting application and system metrics and logs. This is a use

While almost all usage of Kafka can be thought of as stream processing, the term is typically used to refer to applications that provide similar functionality to map/reduce processing in Hadoop. Hadoop usually relies on aggregation of data over a long time frame, either hours or days. Stream processing operates on data in real time, as quickly as messages are produced. Stream frameworks allow users to write small applications to operate on Kafka messages, performing tasks such as counting metrics, partitioning messages for efficient processing by other applications, or transforming messages using data from multiple sources. Stream processing is covered in Chapter 14.

Kafka's Origin

Kafka was created to address the data pipeline problem at LinkedIn. It was designed to provide a high-performance messaging system that can handle many types of data and provide clean, structured data about user activity and system metrics in real time.

> Data really powers everything that we do.
>
> —Jeff Weiner, former CEO of LinkedIn

LinkedIn's Problem

Similar to the example described at the beginning of this chapter, LinkedIn had a system for collecting system and application metrics that used custom collectors and

Installing ZooKeeper

Apache Kafka uses Apache ZooKeeper to store metadata about the Kafka cluster, as well as consumer client details, as shown in Figure 2-1. ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. This book won't go into extensive detail about ZooKeeper but will limit explanations to only what is needed to operate Kafka. While it is possible to run a ZooKeeper server using scripts contained in the Kafka distribution, it is trivial to install a full version of ZooKeeper from the distribution.

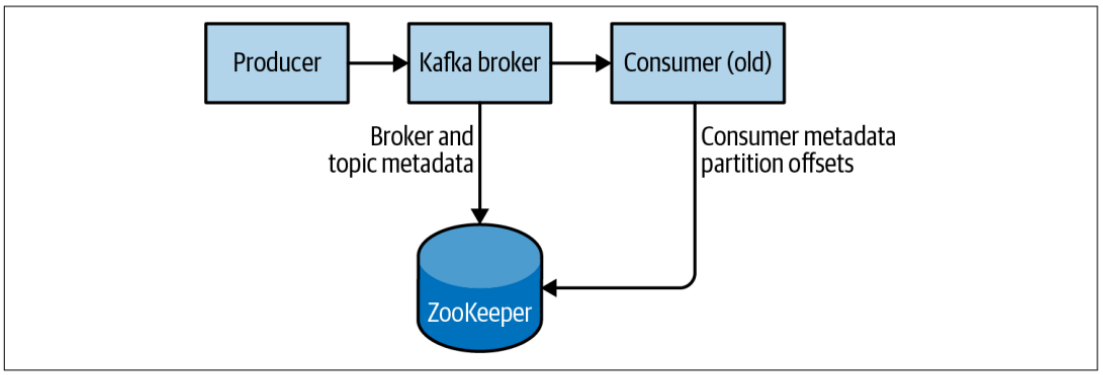
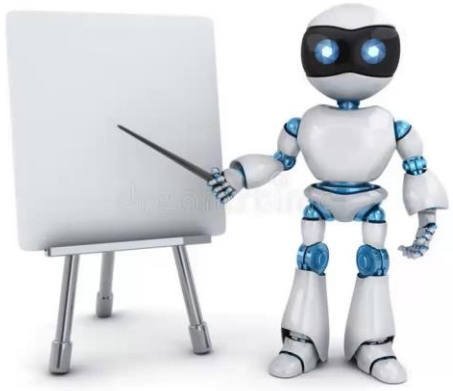


Figure 2-1. Kafka and ZooKeeper



block/chunk size- few kb

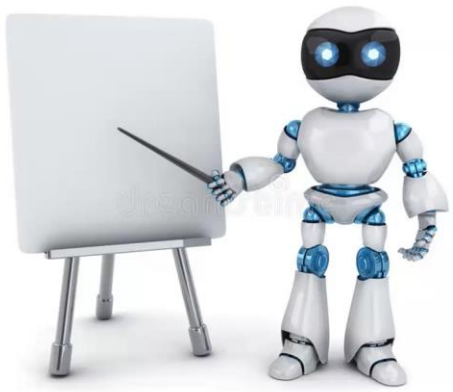
Kafka for Big Data

Overview

Producer --> Broker ---> Consumer
Cluster(message)

Topic 1
Topic 2
Topic 3

Datamites



block/chunk size- few kb

Kafka for Big Data

Overview

publish subscribe
Producer --> Broker ---> Consumer
Cluster(message)

Topic 1
Topic 2
Topic 3

Datamites

What is Streaming?

Streaming means continuous generation of data by multiple data sources.

The strategy used here is to send data in small chunks.

The size of chunk can vary upto few kilobytes.

Datamites

What is Kafka?

Apache Kafka is a distributed streaming platform that is used for building real-time data pipelines and streaming applications.

Kafka is a publish-subscribe messaging system, meaning that it allows multiple consumers to read from a topic (or "topic partition") at the same time. It also allows for storing streams of records in a fault-tolerant way.

Datamites

What is Kafka?

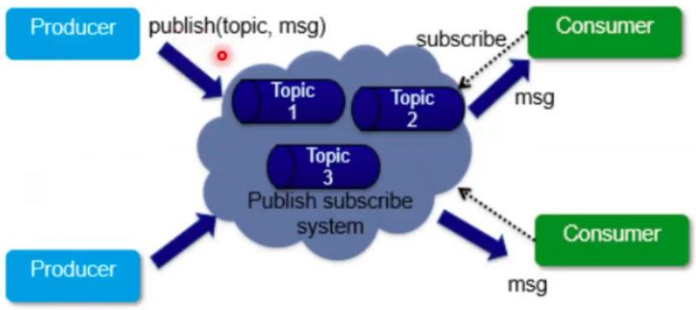
Apache Kafka is a distributed streaming platform that is used for building real-time data pipelines and streaming applications.

Kafka is a publish-subscribe messaging system, meaning that it allows multiple consumers to read from a topic (or "topic partition") at the same time. It also allows for storing streams of records in a fault-tolerant way.

```
            publish        subscribe
Producer --> Message  ---> Consumer
             Broker

             Topic 1,2,3,4,5...
```

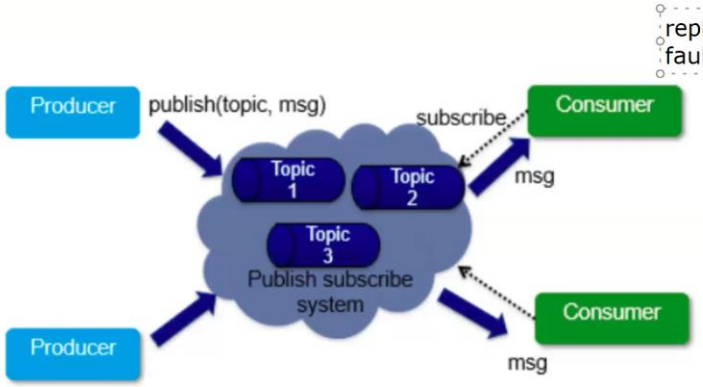
Datamites

What is Kafka?



Datamites

What is Kafka?



Datamites

What is Kafka?

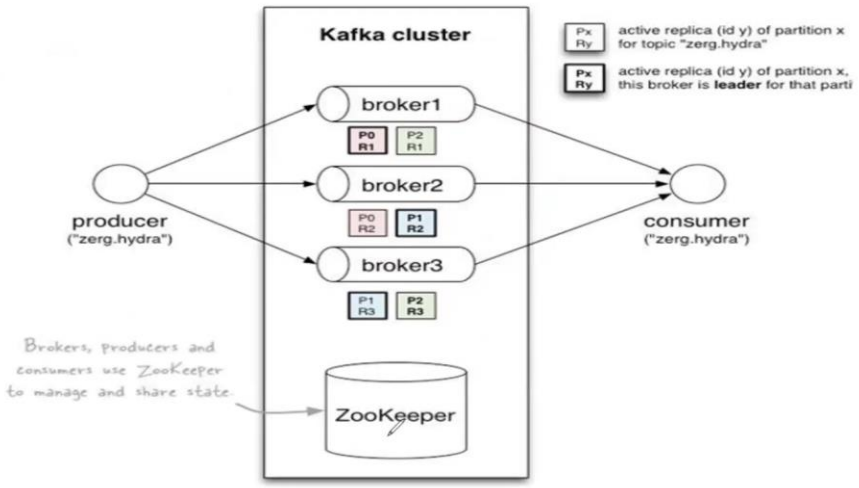
Producers: Applications that generate data and send it to topics in a Kafka cluster.

Topics: A category or feed name to which records are published.

Brokers: A cluster of servers that stores and distributes the published records
Consumers: Applications that read data from a topic.

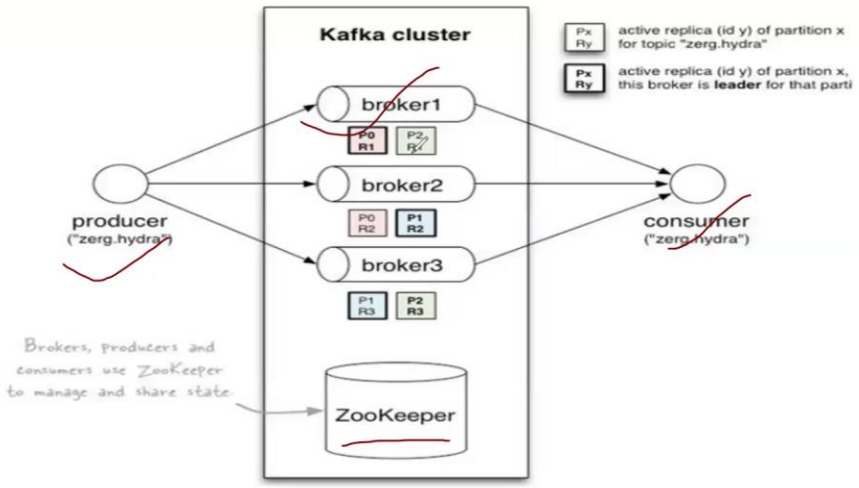
Datamites

Kafka Architecture



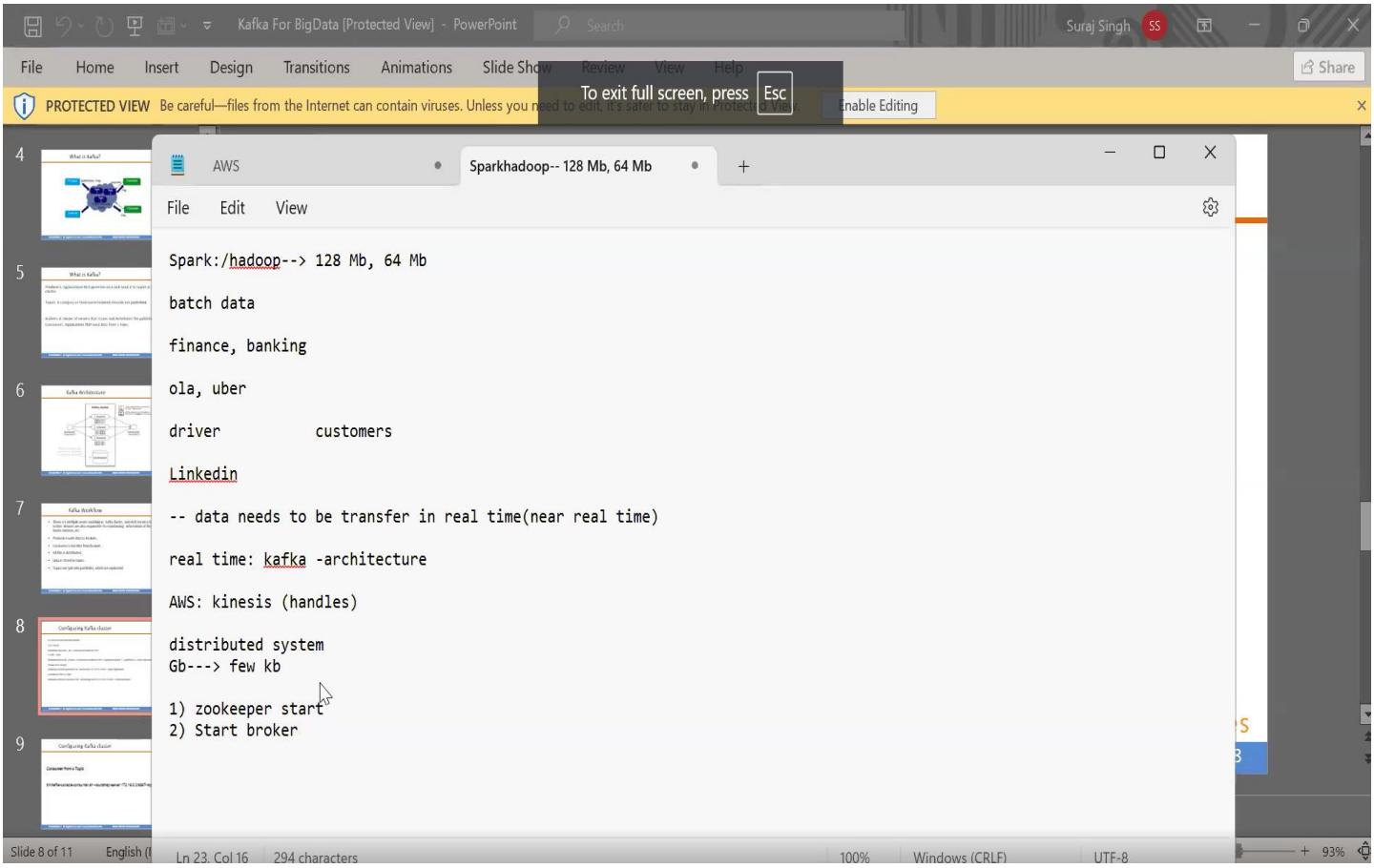
Datamites

Kafka Architecture



Datamites

© DataMites®. All Rights Reserved | www.datamites.com DATA SCIENCE FOUNDATION 6



https://kafka.apache.org/downloads

3.8.0

- Released July 29, 2024
- Release Notes
- Docker image: apache/kafka:3.8.0.
- Docker Native image: apache/kafka-native:3.8.0.
- Source download: kafka-3.8.0-src.tgz (asc, sha512)
- Binary downloads:
  - Scala 2.12 - kafka_2.12-3.8.0.tgz (asc, sha512)
  - Scala 2.13 - kafka_2.13-3.8.0.tgz (asc, sha512)

We build for multiple versions of Scala. This only matters if you are using Scala and you want a version built for the same Scala version you use. Otherwise any version should work (2.13 is recommended).

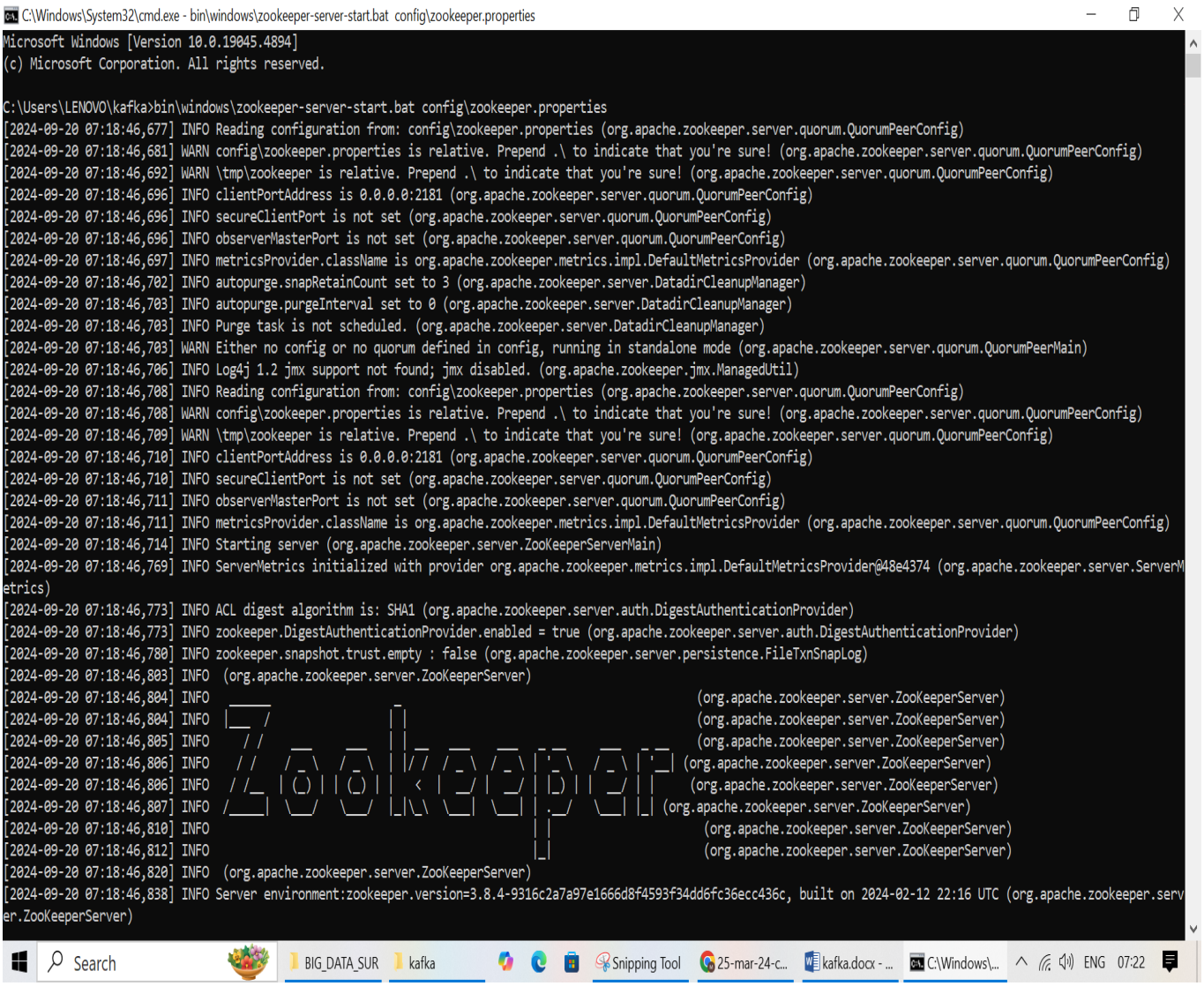
Kafka 3.8.0 includes a significant number of new features and fixes. For more information, please read our blog post and the detailed Release Notes.

Extract>> bin >> windows

Run this command in CLI

bin\windows\zookeeper-server-start.bat config\zookeeper.properties

zookeeper is running on client port address 2181



Now we need to start the kafka

bin\windows\kafka-server-start.bat config\server.properties

bin\windows\kafka-topics.bat --create --topic demo-topic

bin\windows\kafka-topics.bat --create --topic demo-topic --bootstrap-server:9092 --partitions 3

bin\windows\kafka-topics.bat --create --topic demo-topic --partitions 3 --replication-factor 1 --bootstrap-server localhost:9092

bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 –list

create two CLI , in the producer cli paste this

bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic demo-trail

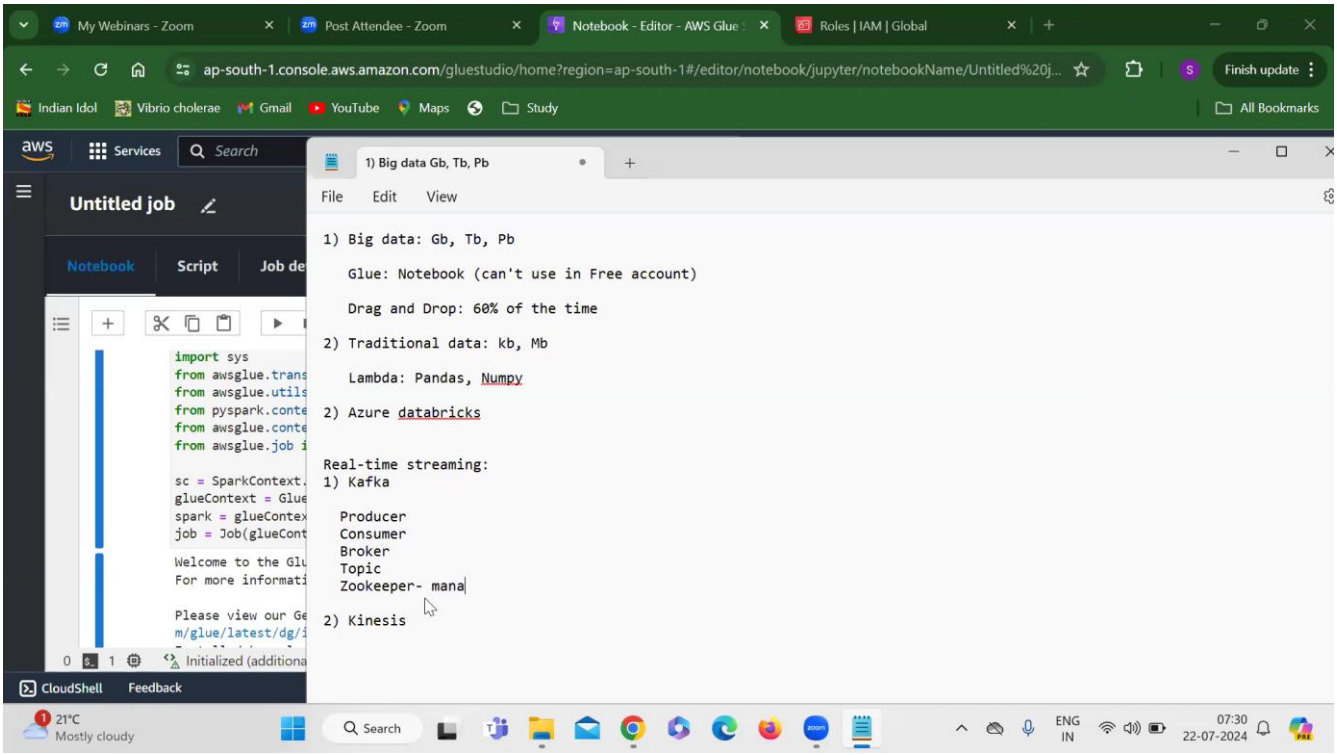
in consumer cli

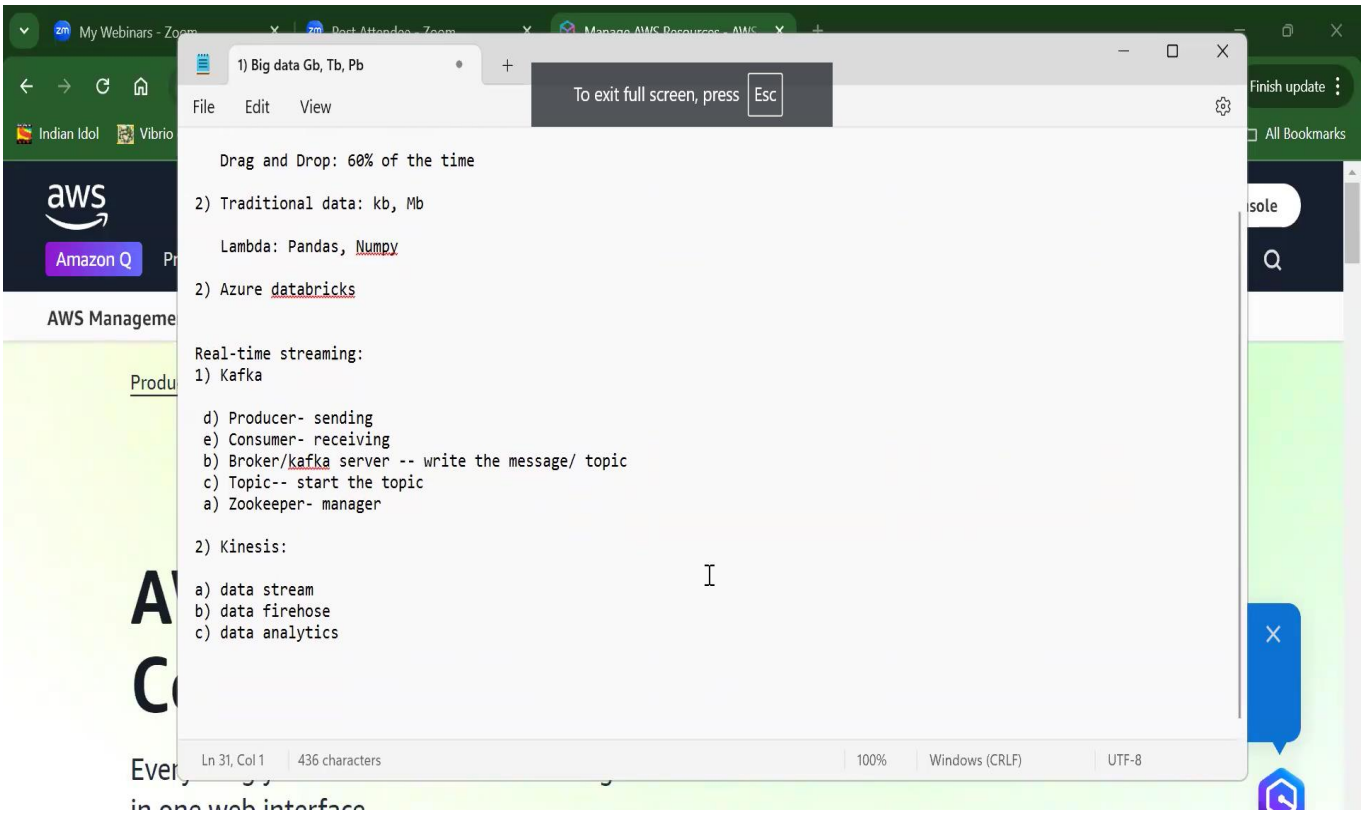
bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic demo-trail --from-beginning

offsetexplorer.com/index.html

```
C:\Users\LENOVO\kafka>bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --list
__consumer_offsets
demo-topic
demo-topic11
demo-trail
kareem

C:\Users\LENOVO\kafka>
```

```
C:\Users\LENOVO\kafka>bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --describe --topic kareem
[2024-09-20 18:30:36,301] WARN [AdminClient clientId=adminclient-1] The DescribeTopicPartitions API is not supported, using Metad
ka.clients.admin.KafkaAdminClient)
Topic: kareem   TopicId: Lub1ihy_TlC2epPQtkr9cw PartitionCount: 3       ReplicationFactor: 1       Configs:
        Topic: kareem   Partition: 0    Leader: 0       Replicas: 0     Isr: 0  Elr: N/A    LastKnownElr: N/A
        Topic: kareem   Partition: 1    Leader: 0       Replicas: 0     Isr: 0  Elr: N/A    LastKnownElr: N/A
        Topic: kareem   Partition: 2    Leader: 0       Replicas: 0     Isr: 0  Elr: N/A    LastKnownElr: N/A

C:\Users\LENOVO\kafka>
```


Notebook - Editor - AWS Glue
Roles | IAM | Global
Finish update
Indian Idol
Vibrio cholerae
Gmail
YouTube
Maps
Study
All Bookmarks
Services
Search
Untitled job
Notebook
Script
1) Big data Gb, Tb, Pb
File
Edit
View
1) Big data: Gb, Tb, Pb
Glue: Notebook (can't use in Free account)
Drag and Drop: 60% of the time
2) Traditional data: kb, Mb
Lambda: Pandas, Numpy
2) Azure databricks
Real-time streaming:
1) Kafka
Producer
Consumer
Broker
Topic
Zookeeper- mana
2) Kinesis
CloudShell
Feedback
21°C
Mostly cloudy
Search
ENG
IN
07:30
22-07-2024


1) Big data Gb, Tb, Pb
To exit full screen, press Esc
File
Edit
View
Drag and Drop: 60% of the time
2) Traditional data: kb, Mb
Lambda: Pandas, Numpy
2) Azure databricks
Real-time streaming:
1) Kafka
d) Producer- sending
e) Consumer- receiving
b) Broker/kafka server -- write the message/ topic
c) Topic-- start the topic
a) Zookeeper- manager
2) Kinesis:
a) data stream
b) data firehose
c) data analytics
Ln 31, Col 1
436 characters
100%
Windows (CRLF)
UTF-8
Amazon Q

1) Big data Gb, Tb, Pb

File Edit View

```
   Glue: Notebook (can't use in Free account)

   Drag and Drop: 60% of the time

2) Traditional data: kb, Mb

   Lambda: Pandas, Numpy

2) Azure databricks


Real-time streaming:
1) Kafka

 d) Producer- sending
 e) Consumer- receiving
 b) Broker/kafka server -- write the message/ topic
 c) Topic-- start the topic
 a) Zookeeper- manager

2) Kinesis:

a) data stream
b) data firehose
c) data analytics



   EC2(logs) ---> data stream --- data firehose ---> s3

                        data stream
                  ------------------------
                  ----------------------------
```

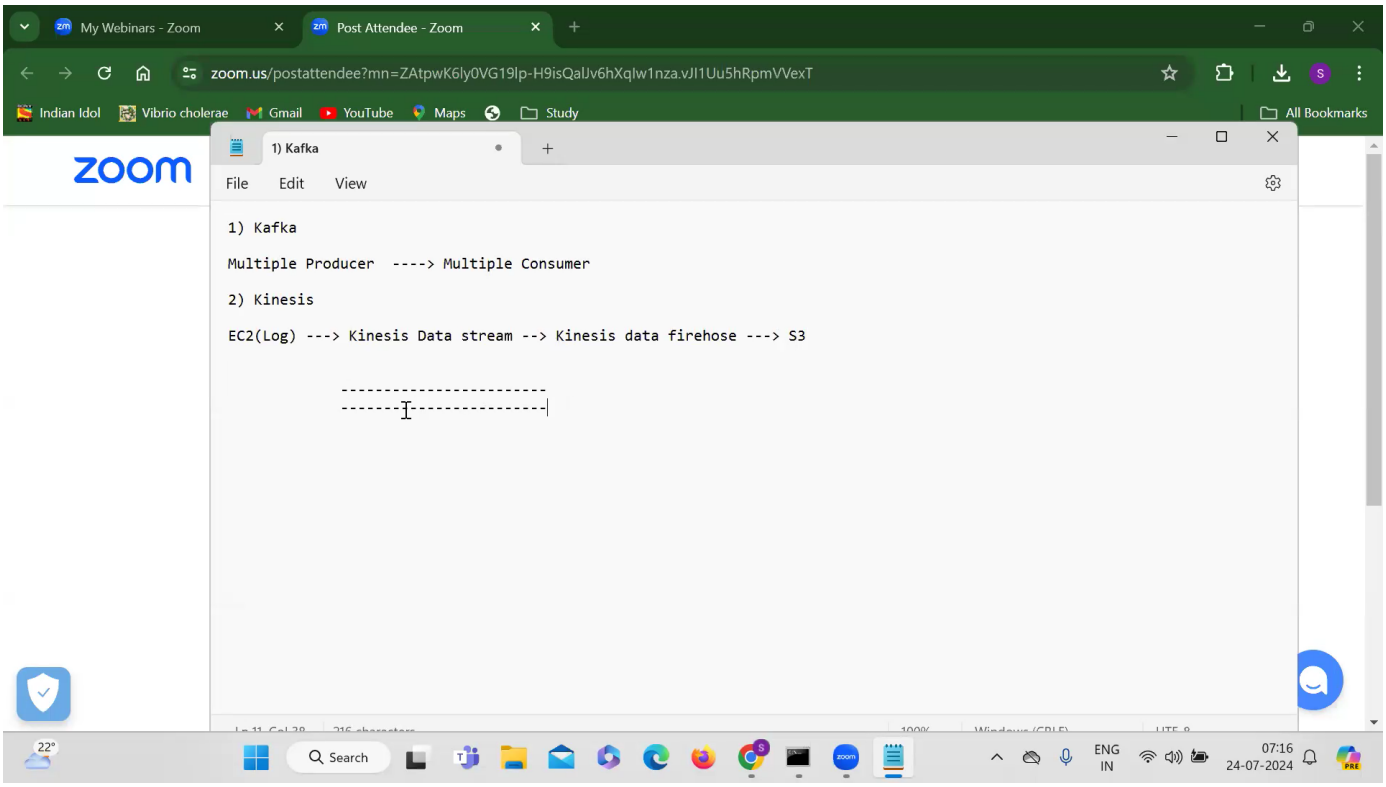
Ln 33, Col 41 | 634 characters | 100% | Windows (CRLF) | UTF-8

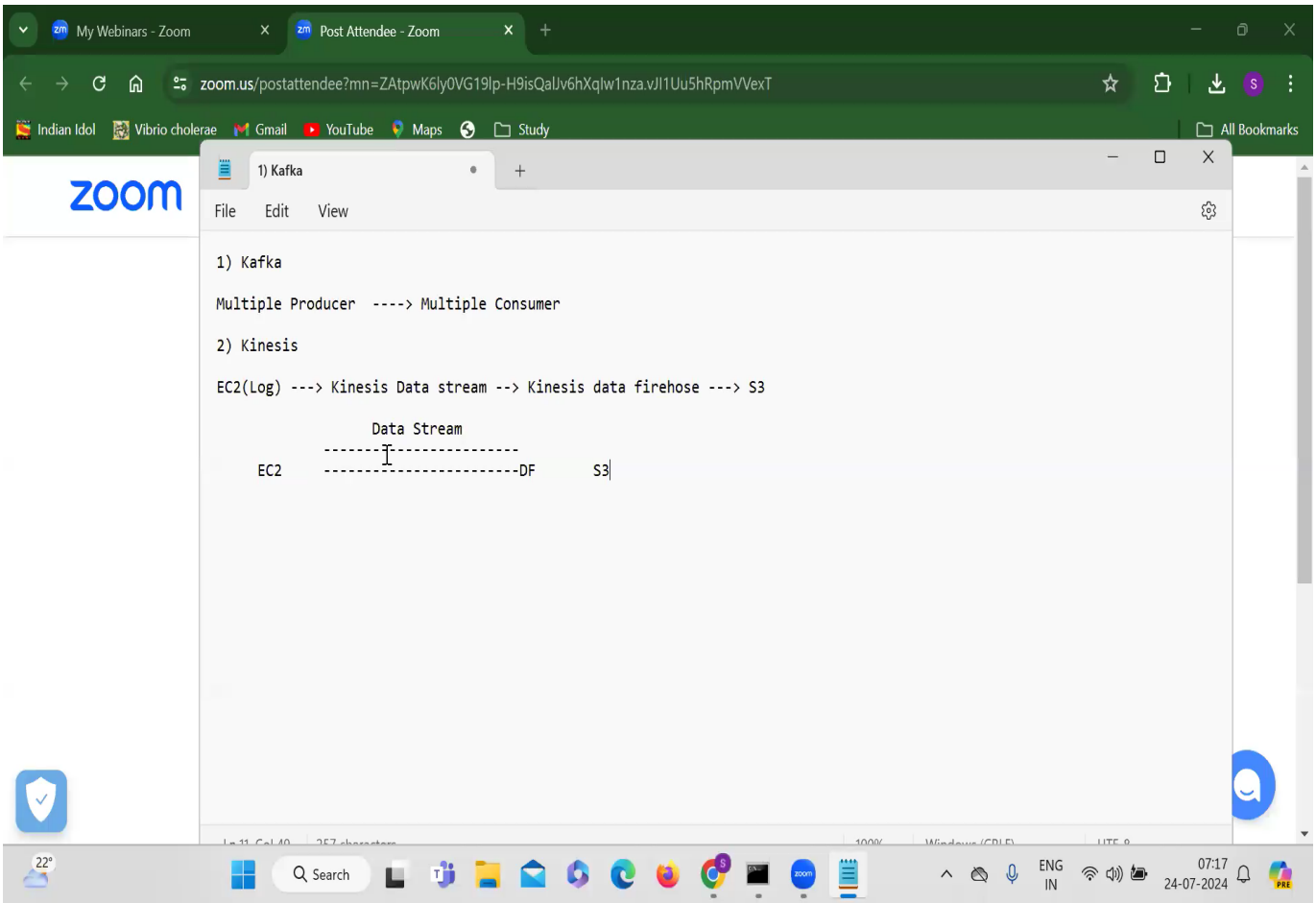
1) Big data Gb, Tb, Pb

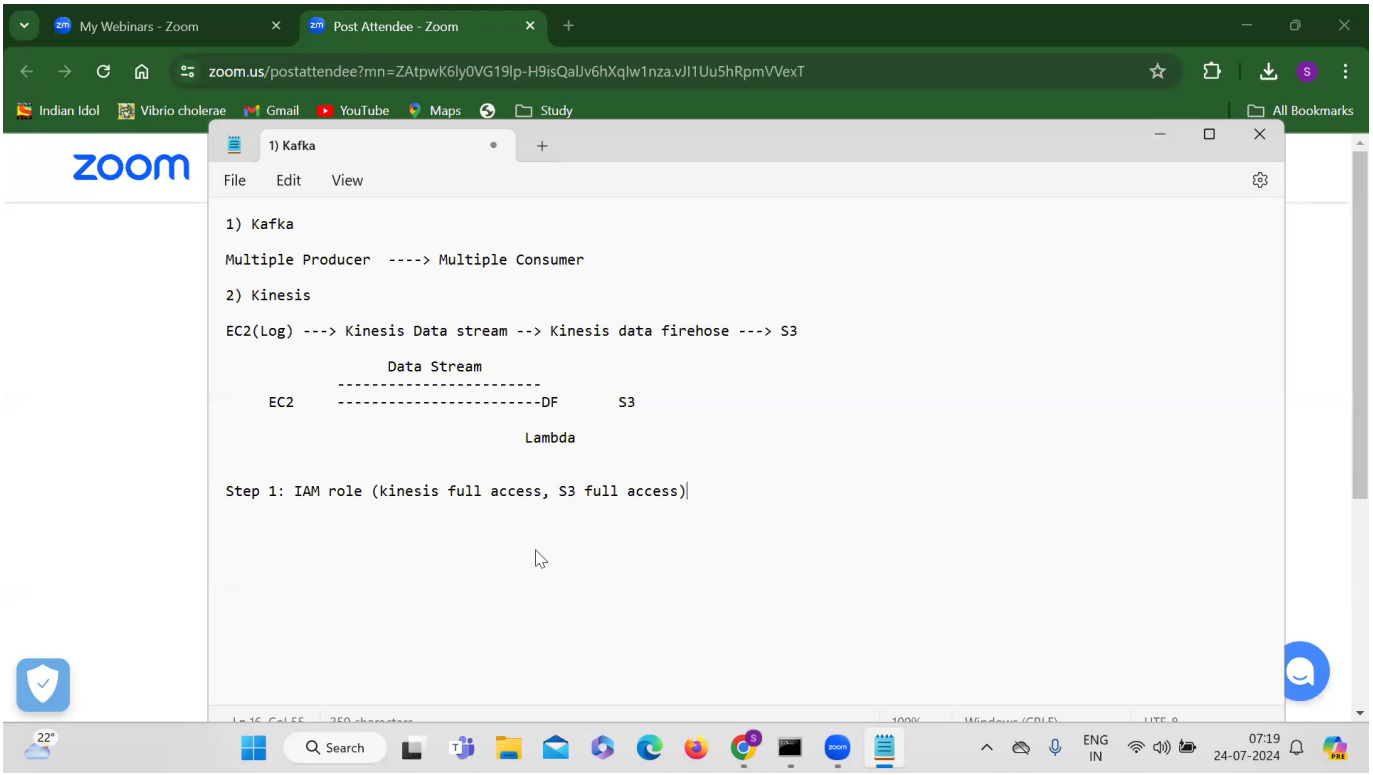
File Edit View

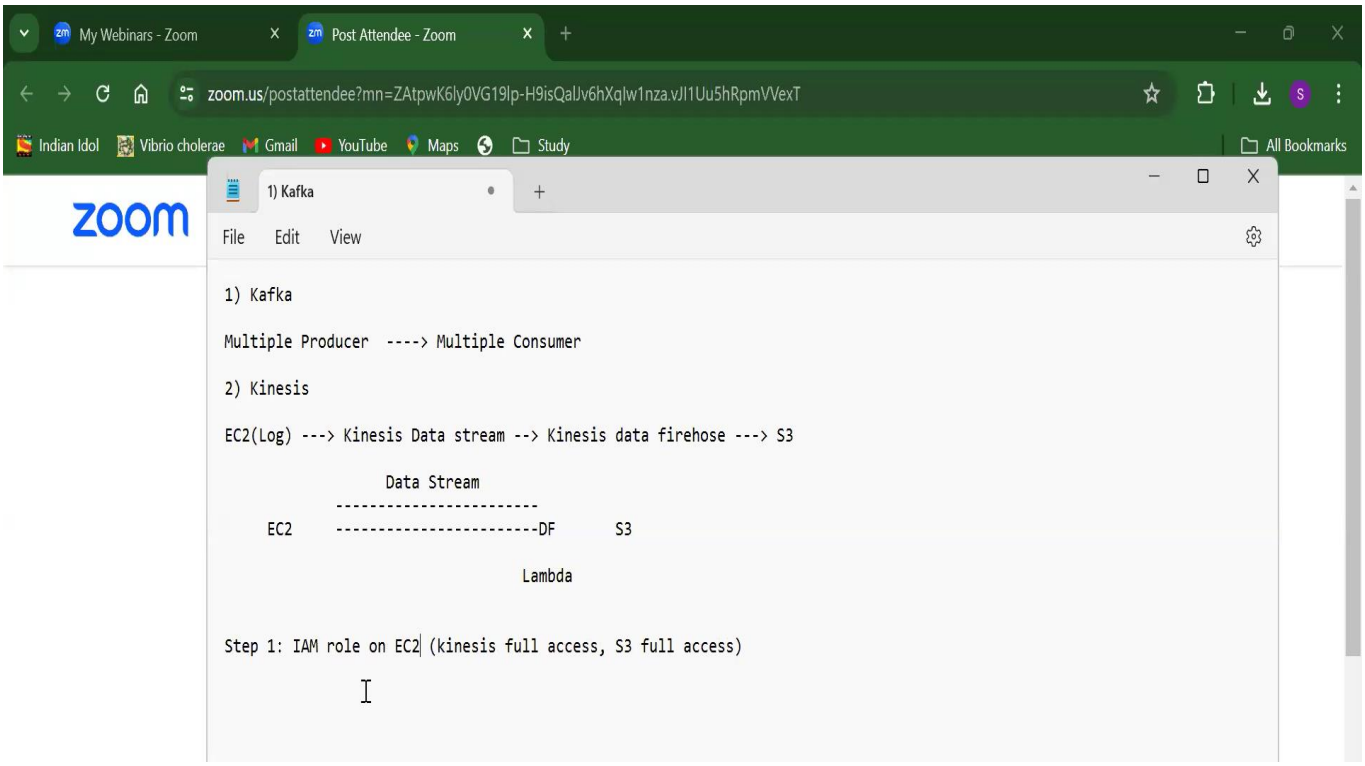
```
   Glue: Notebook (can't use in Free account)

   Drag and Drop: 60% of the time

2) Traditional data: kb, Mb

   Lambda: Pandas, Numpy

2) Azure databricks


Real-time streaming:
1) Kafka

 d) Producer- sending
 e) Consumer- receiving
 b) Broker/kafka server -- write the message/ topic
 c) Topic-- start the topic
 a) Zookeeper- manager

2) Kinesis:

a) data stream
b) data firehose
c) data analytics



   EC2(logs) ---> data stream --- data firehose ---> s3

                        data stream
              EC2    ------------------------     S3
                  DF ----------------------------DF
```

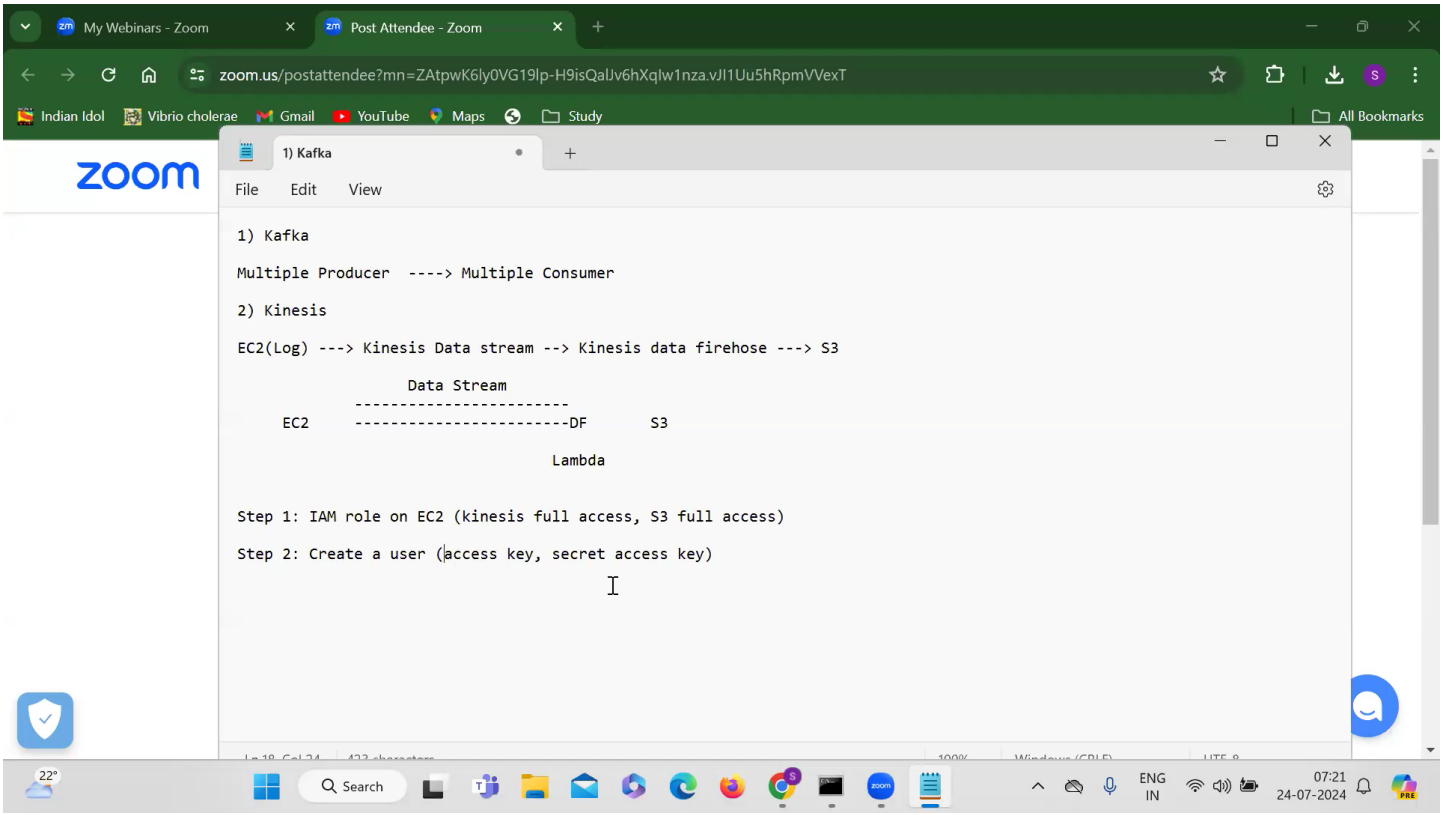
Ln 34, Col 54 | 645 characters | 100% | Windows (CRLF) | UTF-8


My Webinars - Zoom
Post Attendee - Zoom
zoom.us/postattendee?mn=ZAtpwK6ly0VG19lp-H9isQaIJv6hXqIw1nza.vJI1Uu5hRpmVVexT
Indian Idol
Vibrio cholerae
Gmail
YouTube
Maps
Study
All Bookmarks
zoom
1) Kafka
File
Edit
View
1) Kafka
Multiple Producer ----> Multiple Consumer
2) Kinesis
EC2(Log) ---> Kinesis Data stream --> Kinesis data firehose ---> S3
Search
ENG
IN
07:16
24-07-2024


My Webinars - Zoom
Post Attendee - Zoom
zoom.us/postattendee?mn=ZAtpwK6ly0VG19lp-H9isQaIJv6hXqIw1nza.vJI1Uu5hRpmVVexT
Indian Idol
Vibrio cholerae
Gmail
YouTube
Maps
Study
All Bookmarks
zoom
1) Kafka
File
Edit
View
1) Kafka
Multiple Producer ----> Multiple Consumer
2) Kinesis
EC2(Log) ---> Kinesis Data stream --> Kinesis data firehose ---> S3
Data Stream
EC2
DF
S3
Search
ENG
IN
07:17
24-07-2024


My Webinars - Zoom
Post Attendee - Zoom
zoom.us/postattendee?mn=ZAtpwK6ly0VG19lp-H9isQaIJv6hXqIw1nza.vJI1Uu5hRpmVVexT
Indian Idol
Vibrio cholerae
Gmail
YouTube
Maps
Study
All Bookmarks
zoom
1) Kafka
File
Edit
View
1) Kafka
Multiple Producer ----> Multiple Consumer
2) Kinesis
EC2(Log) ---> Kinesis Data stream --> Kinesis data firehose ---> S3
Data Stream
EC2
DF
S3
Lambda
Step 1: IAM role (kinesis full access, S3 full access)
Search
ENG
IN
07:19
24-07-2024


My Webinars - Zoom
Post Attendee - Zoom
zoom.us/postattendee?mn=ZAtpwK6ly0VG19lp-H9isQaIJv6hXqIw1nza.vJI1Uu5hRpmVVexT
Indian Idol
Vibrio cholerae
Gmail
YouTube
Maps
Study
All Bookmarks
zoom
1) Kafka
File
Edit
View
1) Kafka
Multiple Producer ----> Multiple Consumer
2) Kinesis
EC2(Log) ---> Kinesis Data stream --> Kinesis data firehose ---> S3
Data Stream
EC2
DF
S3
Lambda
Step 1: IAM role on EC2 (kinesis full access, S3 full access)


My Webinars - Zoom
Post Attendee - Zoom
zoom.us/postattendee?mn=ZAtpwK6ly0VG19lp-H9isQaIJv6hXqIw1nza.vJI1Uu5hRpmVVexT
Indian Idol
Vibrio cholerae
Gmail
YouTube
Maps
Study
All Bookmarks
zoom
1) Kafka
File
Edit
View
1) Kafka
Multiple Producer ----> Multiple Consumer
2) Kinesis
EC2(Log) ---> Kinesis Data stream --> Kinesis data firehose ---> S3
Data Stream
EC2
DF
S3
Lambda
Step 1: IAM role on EC2 (kinesis full access, S3 full access)
Step 2: Create a user (access key, secret access key)
22°
Search
ENG
IN
07:21
24-07-2024