WISSEN

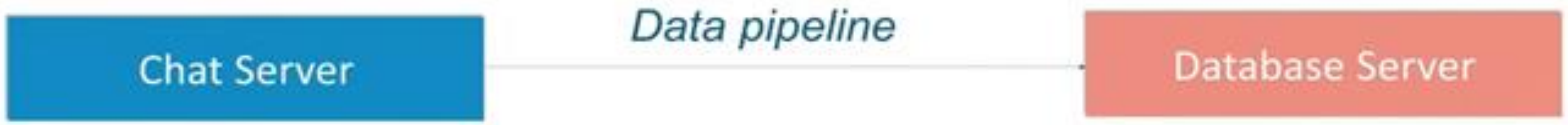
Great Place To Work. Certified SEP 2020–AUG 2021 INDIA

Exploring Partnership Opportunities

Nagendra.Kommana@wissen.com
(+91) (9989339903)

Data Pipelines

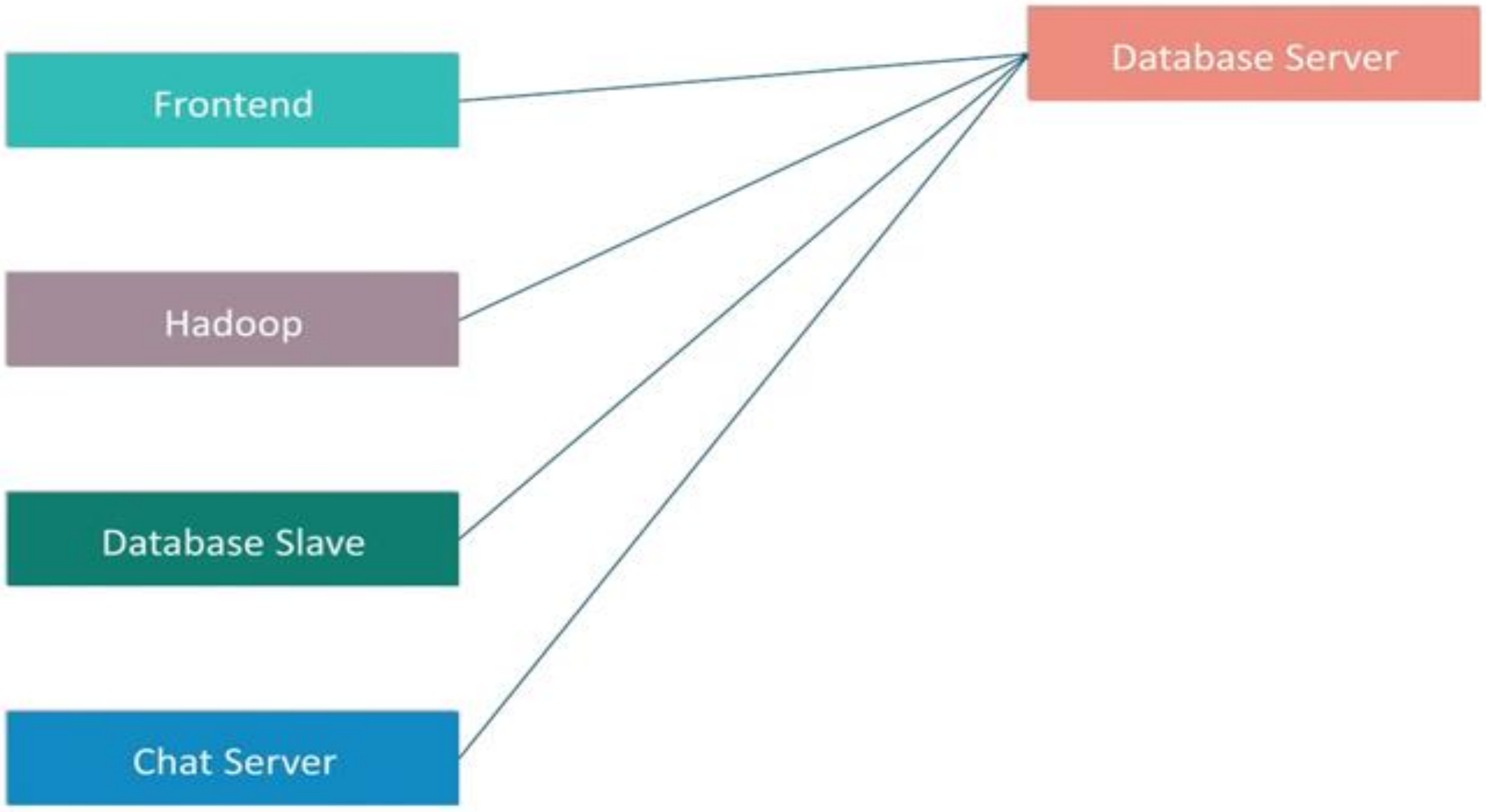
Communication is required between different systems in the real-time scenario, which is done by using data pipelines.



☛ For Example: Chat Server needs to communicate with Database Server for storing messages

Increase in number of Nodes

Similarly, there may be many applications wanting to access the Database Server



Frontend

Hadoop

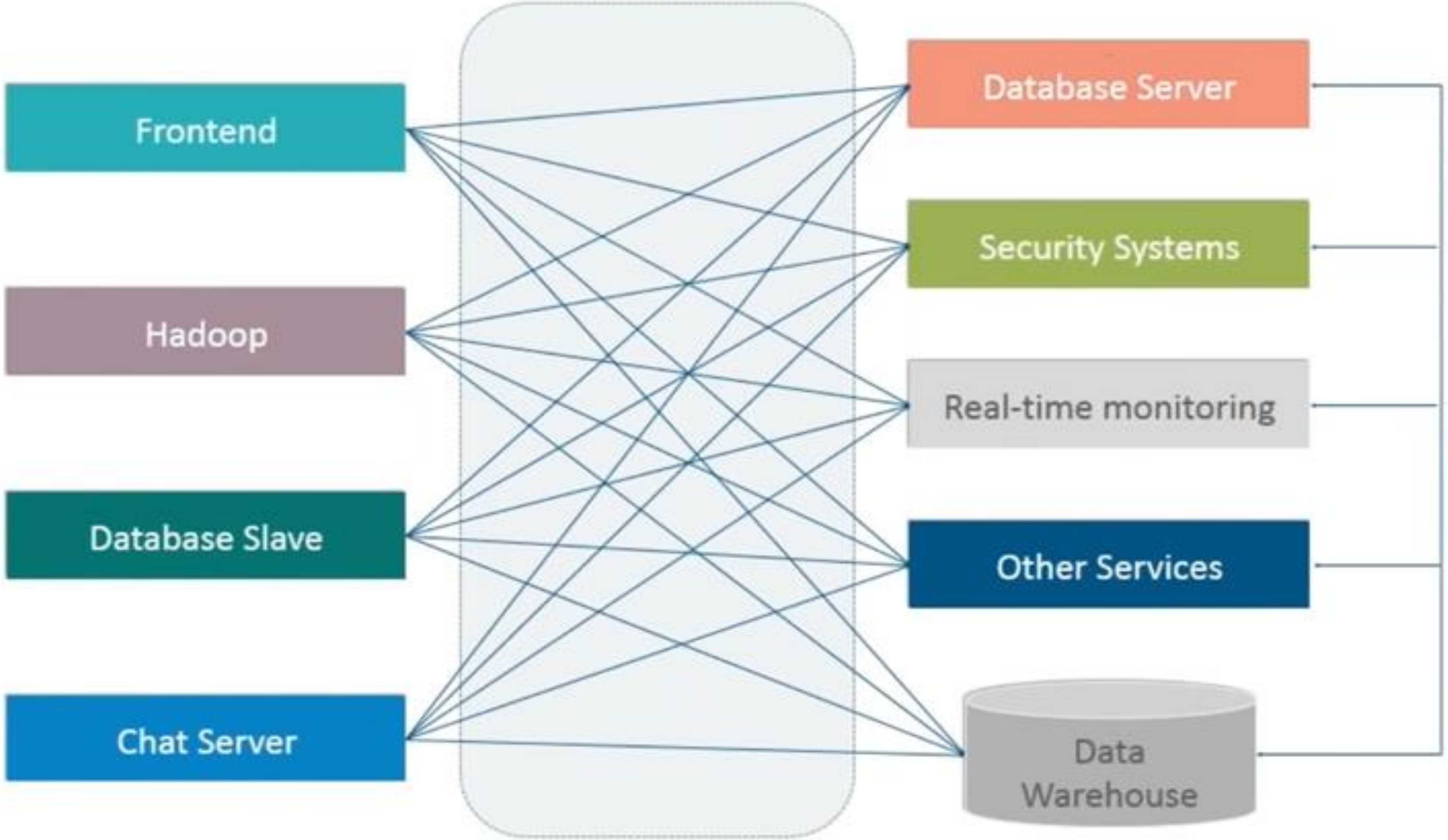
Database Slave

Chat Server

Database Server

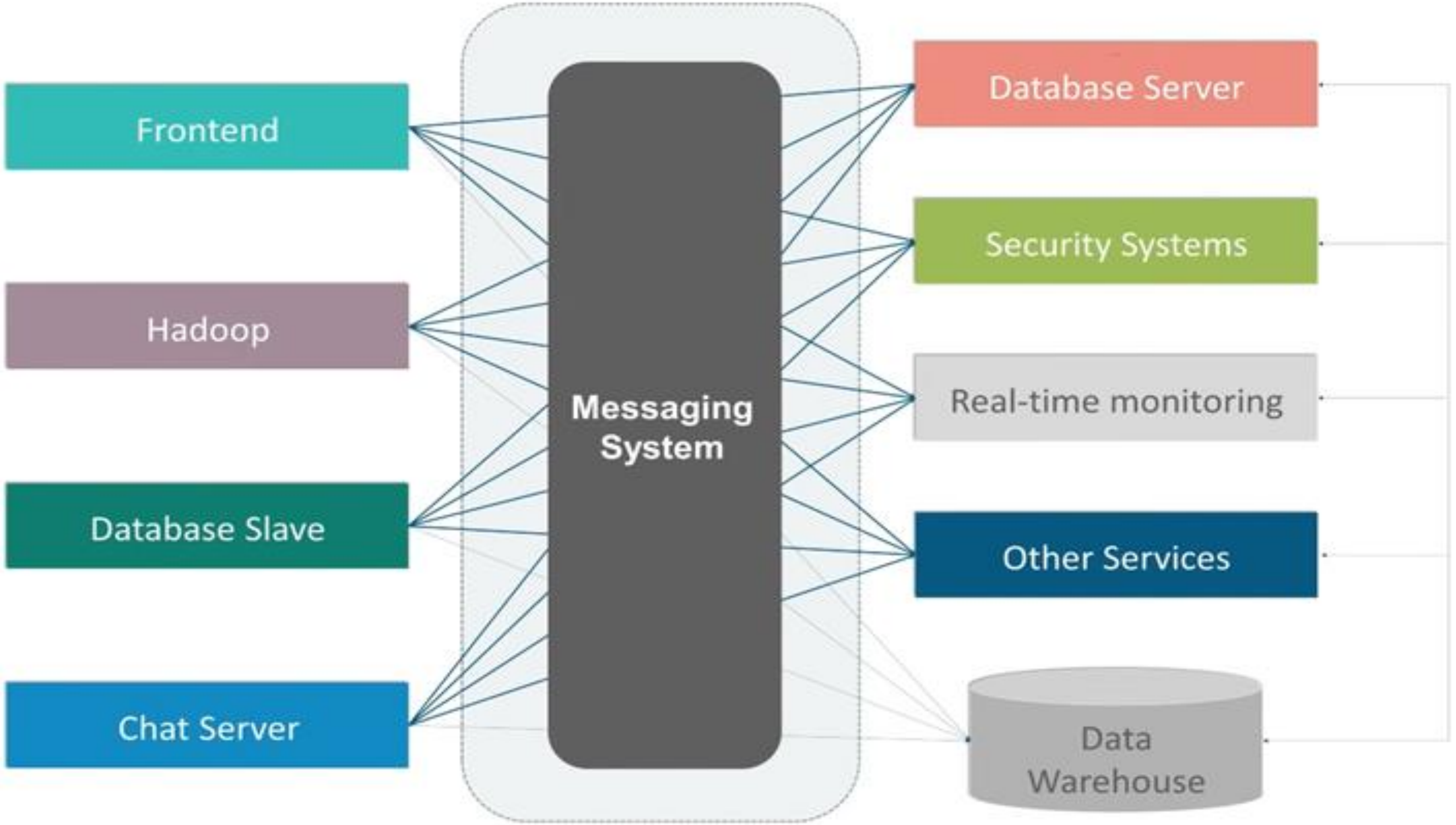
Complex Data Pipelines

Similarly, applications may also be communicating with Real-time monitoring and Other services in real-time scenario

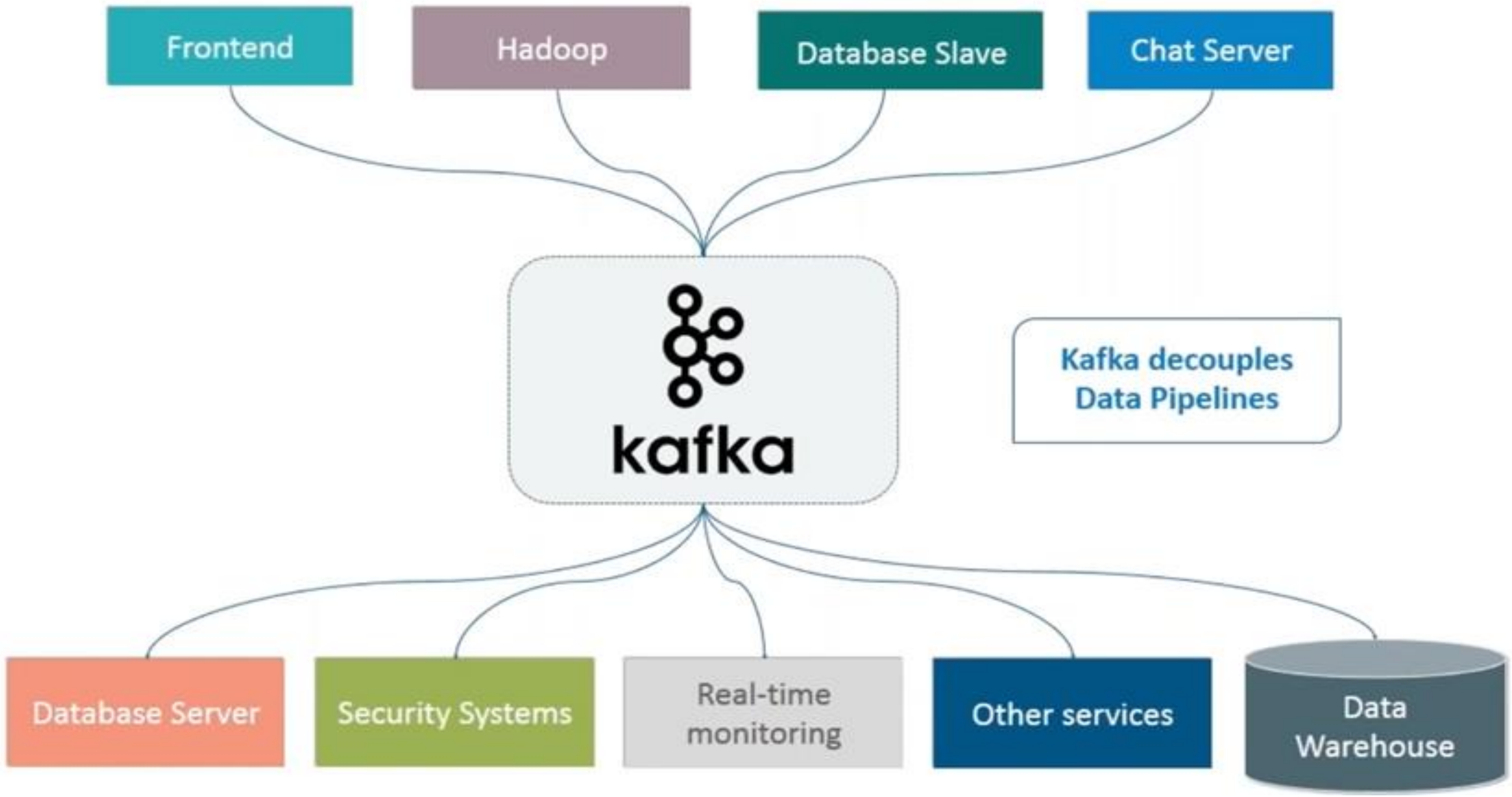


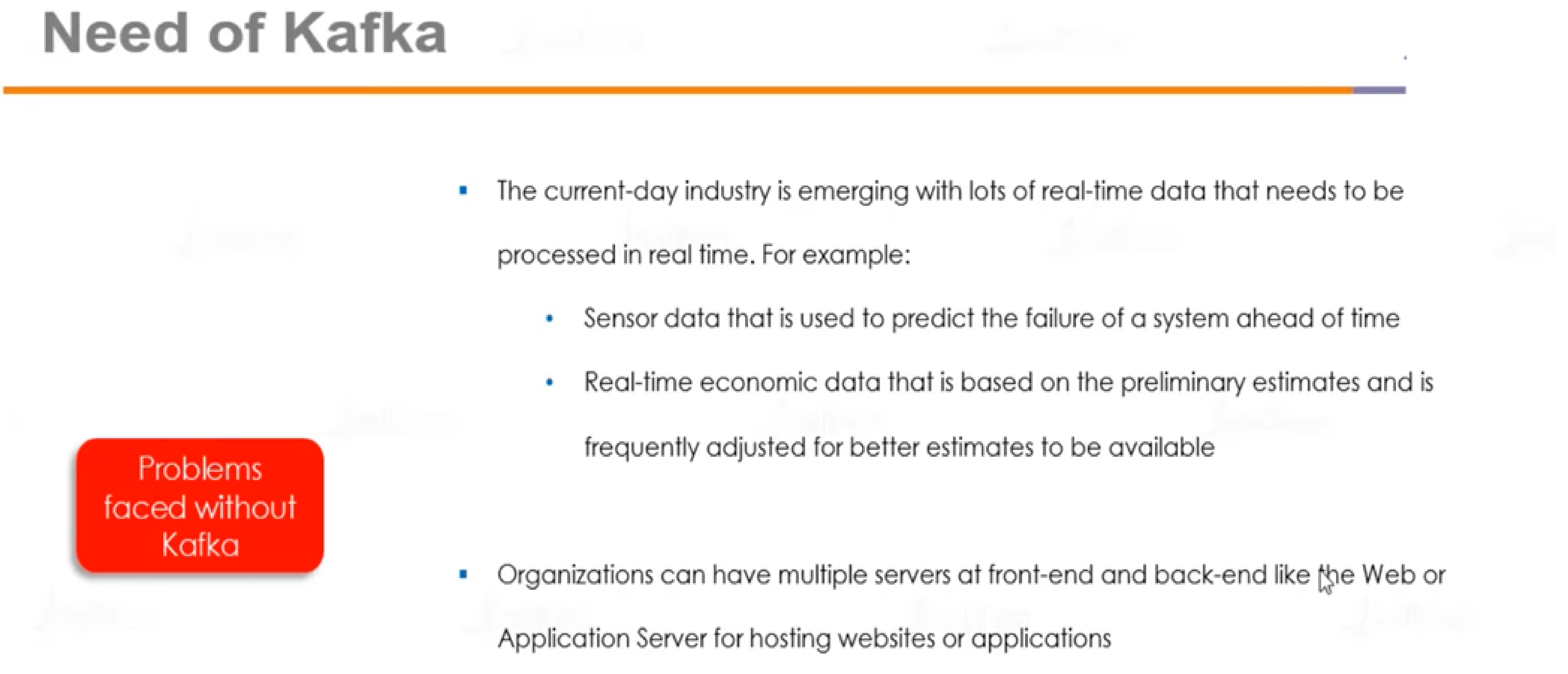
Solution to the Complex Data Pipelines

Messaging Systems helps managing the complexity of the pipelines



Kafka Decouples Data Pipelines





Need of Kafka

- The current-day industry is emerging with lots of real-time data that needs to be processed in real time. For example:
  - Sensor data that is used to predict the failure of a system ahead of time
  - Real-time economic data that is based on the preliminary estimates and is frequently adjusted for better estimates to be available
- Organizations can have multiple servers at front-end and back-end like the Web or Application Server for hosting websites or applications

Problems faced without Kafka

What is Kafka?

- **Apache Kafka** is a distributed *publish-subscribe* messaging system
- It was originally developed at LinkedIn and later on became a part of Apache Project
- Kafka is fast, scalable, durable, fault-tolerant and distributed by design

Kafka @LinkedIn

- *1100+ commodity machines*
- *31,000+ topics*
- *350,000+ partitions*

- *675 billion messages/day*
- *150 TB/day in*
- *580 TB/day out*

Peak Load

- *10.5 million messages/sec*
- *18.5 GB/sec Inbound*
- *70.5 GB/sec Outbound*

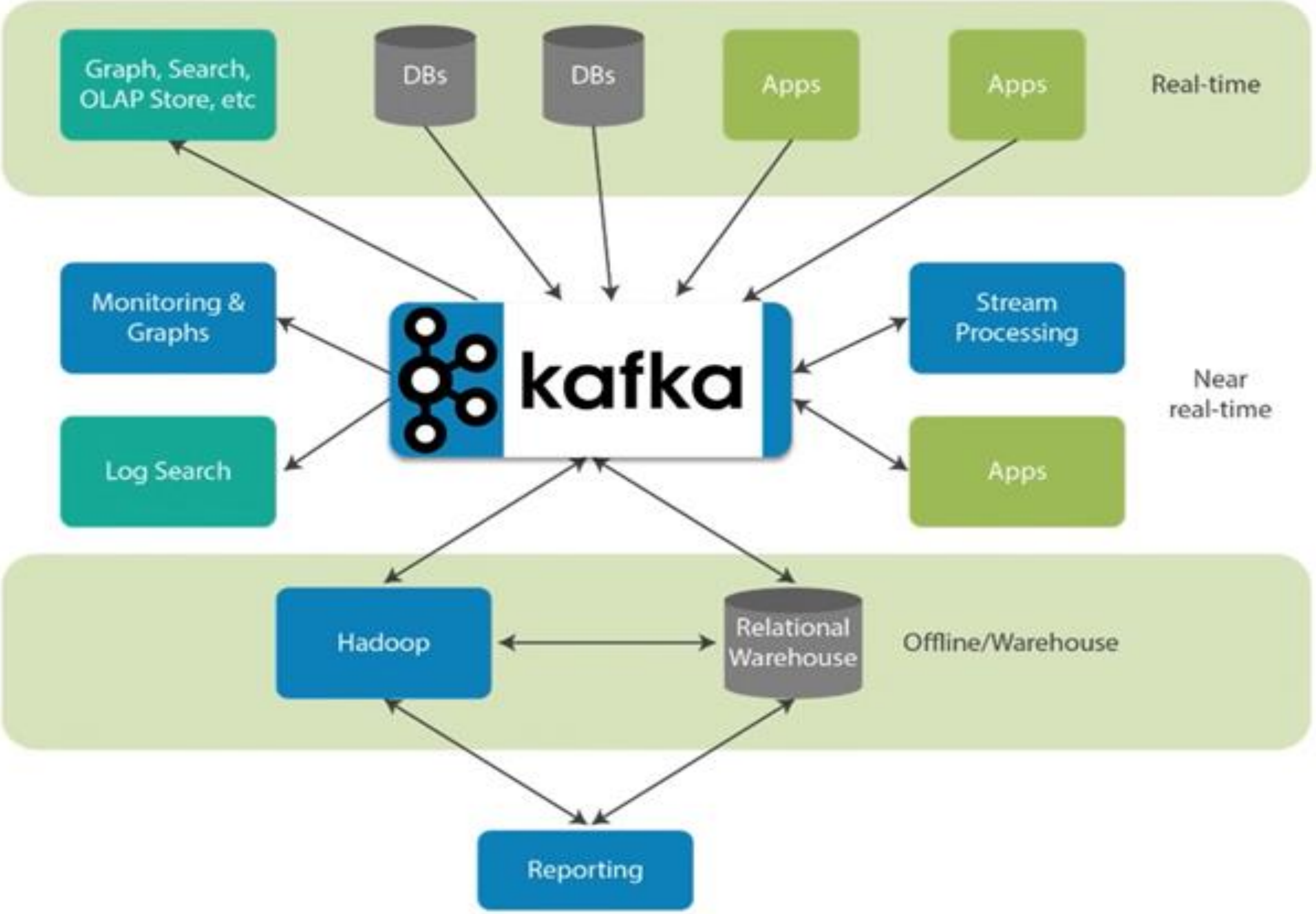


Fig: A modern stream-centric data architecture built around Kafka

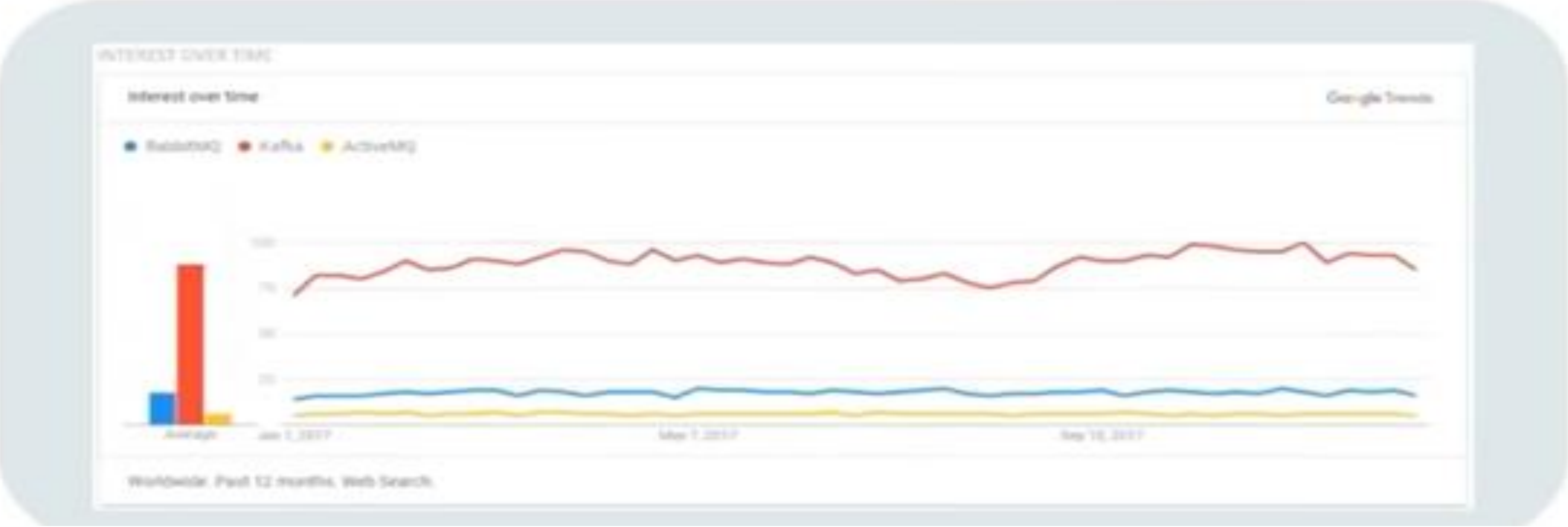
Kafka Growth Exploding

- More than **1/3** of all Fortune **500** companies use **Kafka**.
- These companies includes the top ten travel companies, **7** of top ten banks, **8** of top ten insurance companies, **9** of top ten telecom companies.
- **LinkedIn**, **Microsoft** and **Netflix** process billions of messages a day with Kafka (1,000,000,000,000).
- **Kafka** is used for **real-time streams** of data & used to collect big data for **real time analysis**.

86% of respondents reported that the number of their systems that use Kafka is increasing

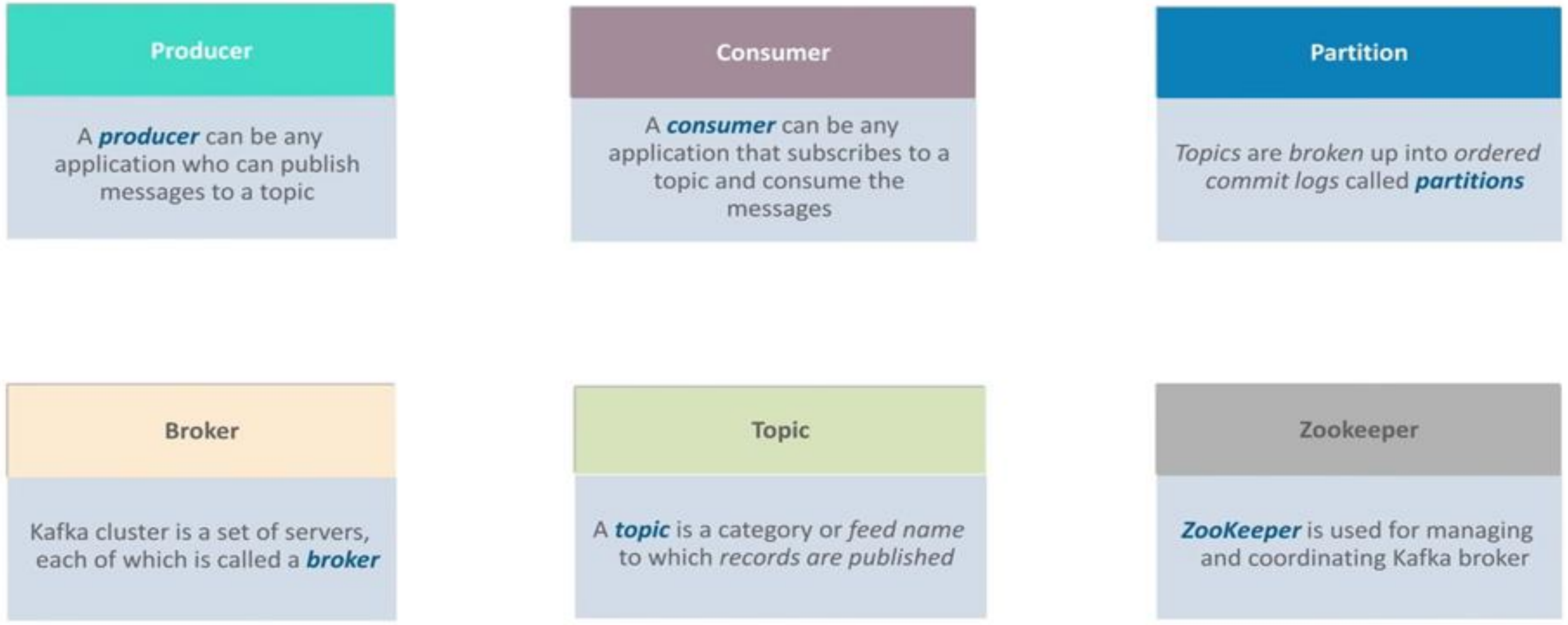
20% reported that the number is "growing a lot!"

52% of organizations have at least 6 systems running Kafka



Source: Google Trends

Kafka Terminologies



Producer

A ***producer*** can be any application who can publish messages to a topic

Consumer

A ***consumer*** can be any application that subscribes to a topic and consume the messages

Partition

Topics are *broken* up into *ordered commit logs* called ***partitions***

Broker

Kafka cluster is a set of servers, each of which is called a ***broker***

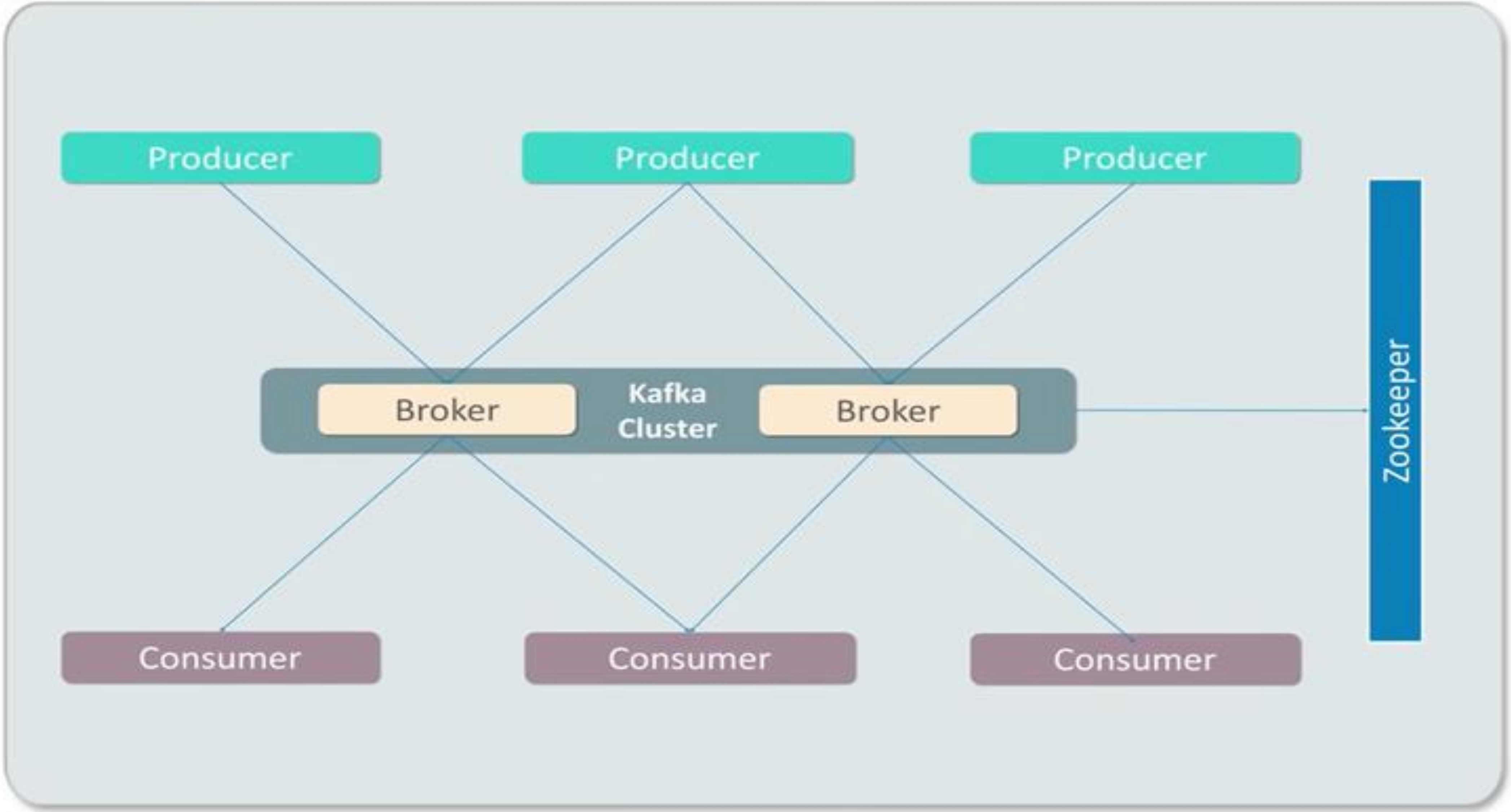
Topic

A ***topic*** is a category or *feed name* to which *records are published*

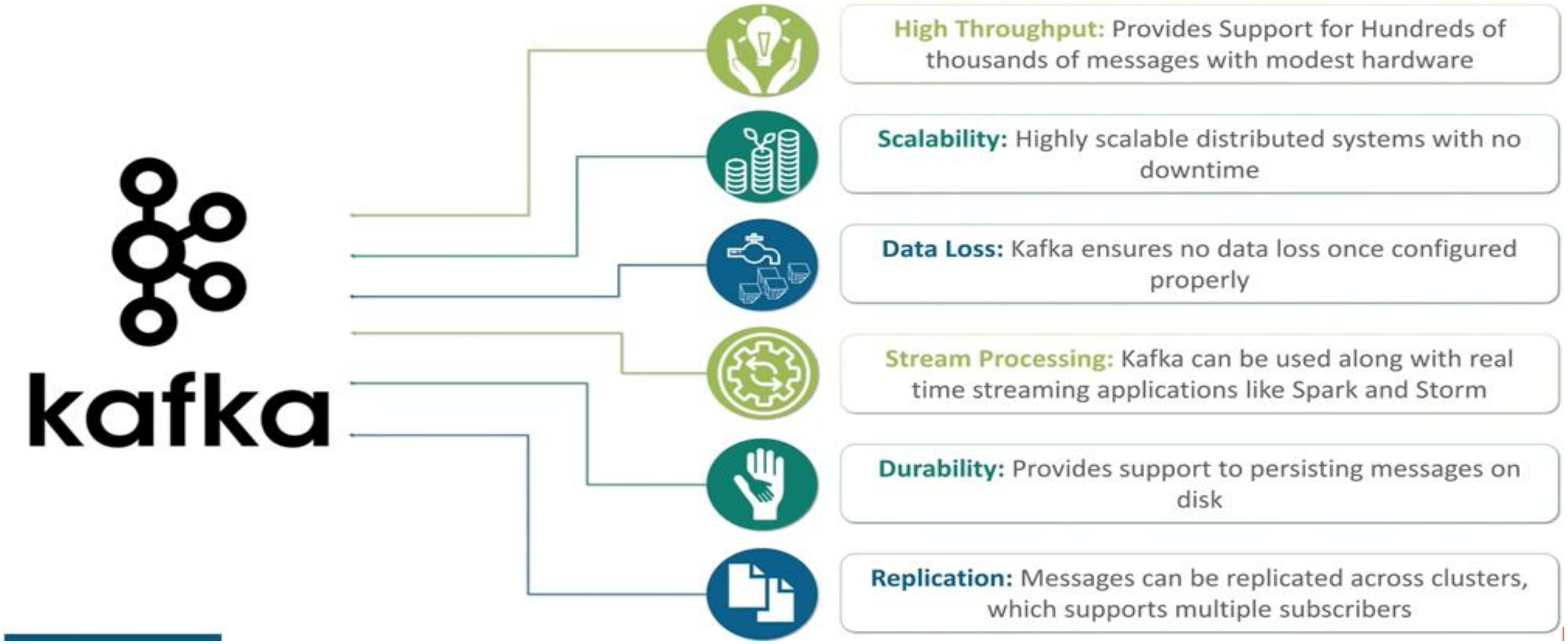
Zookeeper

ZooKeeper is used for managing and coordinating Kafka broker

Kafka Cluster



Kafka Features



High Throughput: Provides Support for Hundreds of thousands of messages with modest hardware

Scalability: Highly scalable distributed systems with no downtime

Data Loss: Kafka ensures no data loss once configured properly

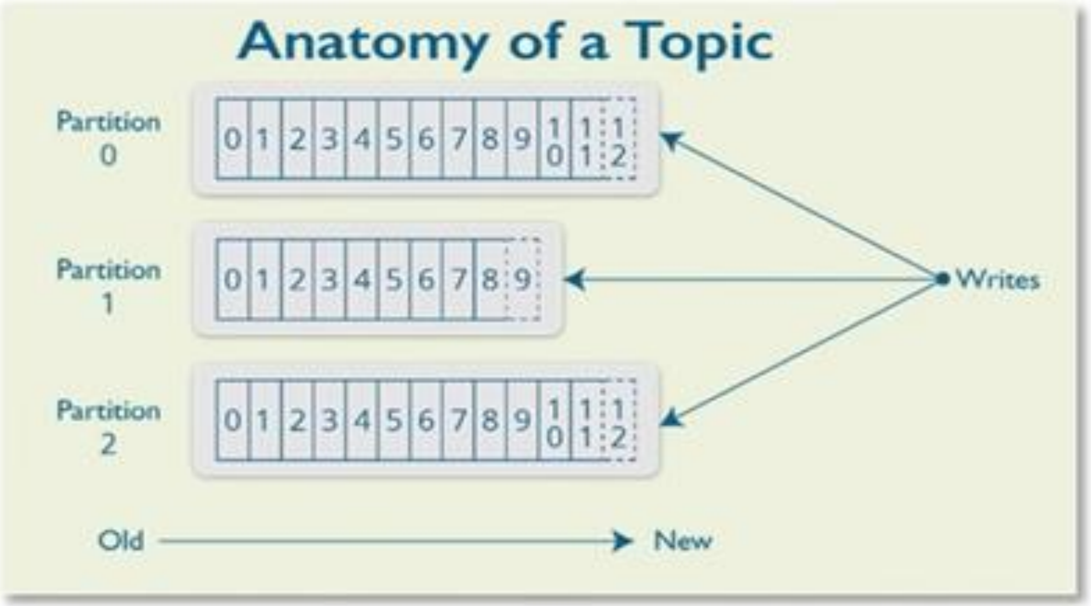
Stream Processing: Kafka can be used along with real time streaming applications like Spark and Storm

Durability: Provides support to persisting messages on disk

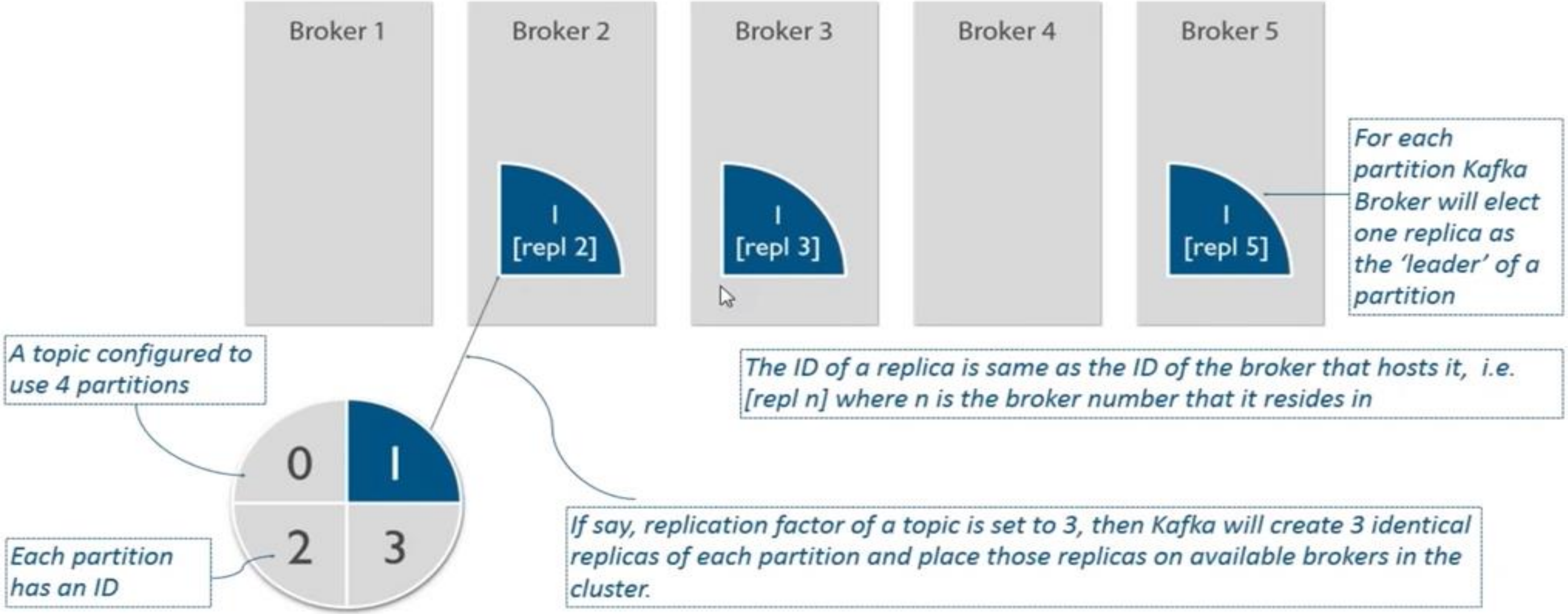
Replication: Messages can be replicated across clusters, which supports multiple subscribers

Kafka Components - Topics and Partitions

- A *topic* is a category or *feed name* to which *records are published*
- *Topics* are *broken* up into *ordered commit logs* called *partitions*
- Each *message* in a *partition* is assigned a *sequential id* called an *offset*
- Data in a topic is retained for a *configurable period of time*
- *Writes* to a partition *are* generally *sequential* thereby *reducing* the *number of hard disk seeks*
- *Reading messages* can either be from *beginning* & *also* can *rewind* or *skip* to any *point* in partition by *giving* an *offset value*



Kafka Components - Topics, Partitions & Replicas



For each partition Kafka Broker will elect one replica as the 'leader' of a partition

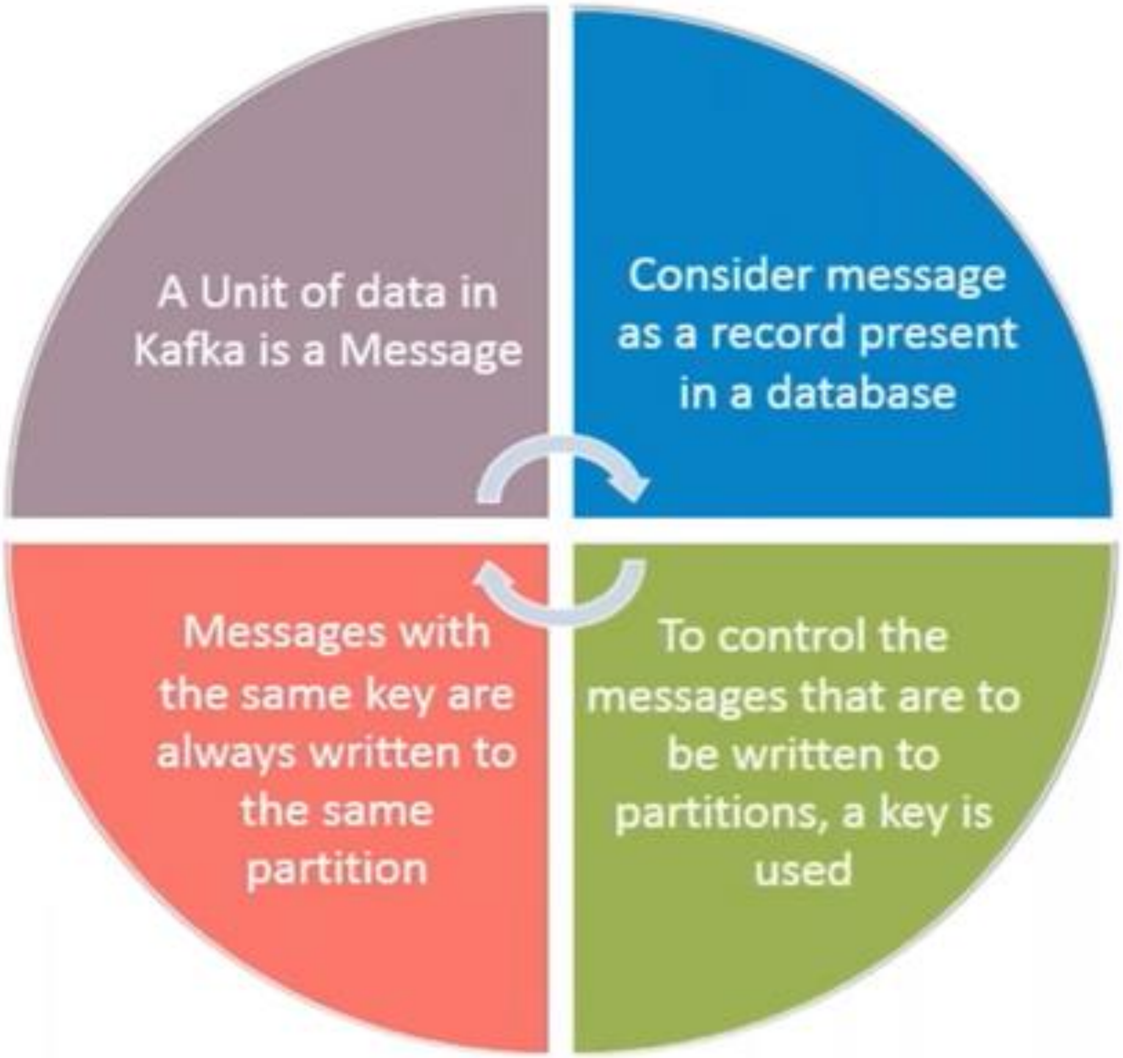
The ID of a replica is same as the ID of the broker that hosts it, i.e. [repl n] where n is the broker number that it resides in

A topic configured to use 4 partitions

Each partition has an ID

If say, replication factor of a topic is set to 3, then Kafka will create 3 identical replicas of each partition and place those replicas on available brokers in the cluster.

Kafka Components - Messages



A Unit of data in Kafka is a Message

Consider message as a record present in a database

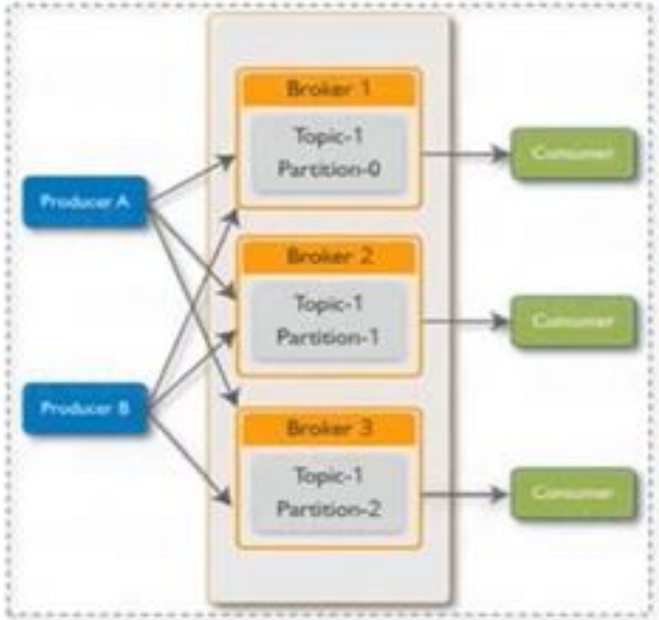
To control the messages that are to be written to partitions, a key is used

Messages with the same key are always written to the same partition

Kafka Components - Producer

1

Producer (publisher or writer) publishes a new message to a **specific topic**

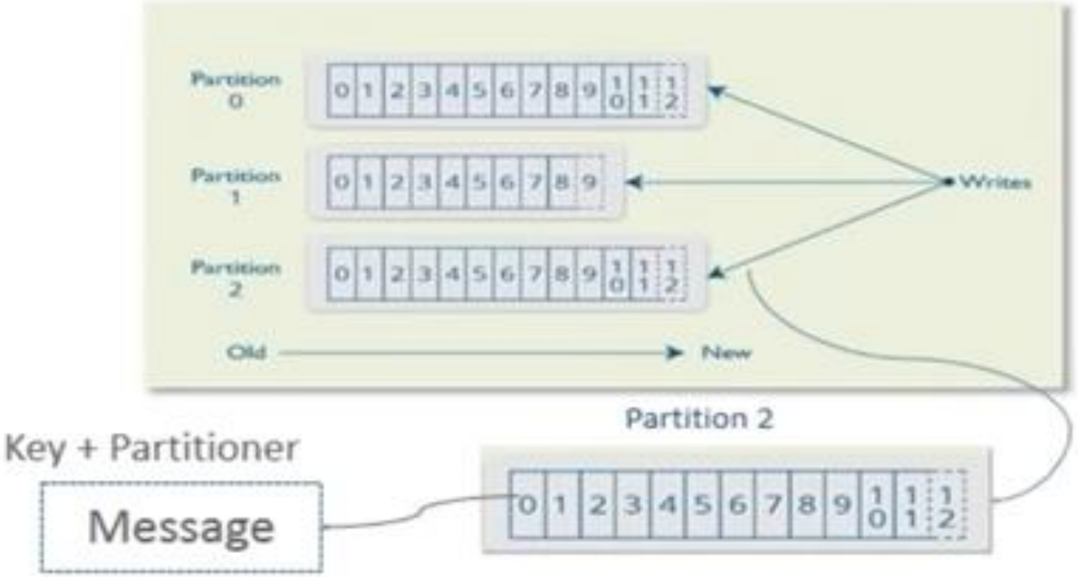


2

The producer does not care what partition a specific message is written to and will balance messages over every partition of a topic evenly

3

Directing messages to a partition is done using the **message key** and a **partitioner**, this will generate a hash of the key and map it to a partition

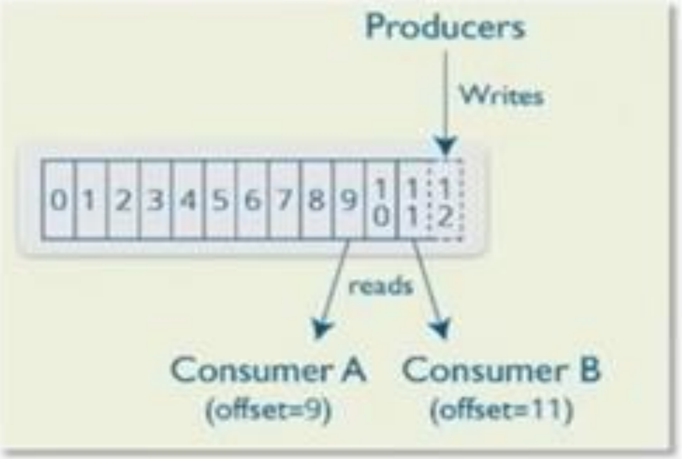


4

Every message a producer publishes in the form of a **key : value** pair

Kafka Components - Consumer

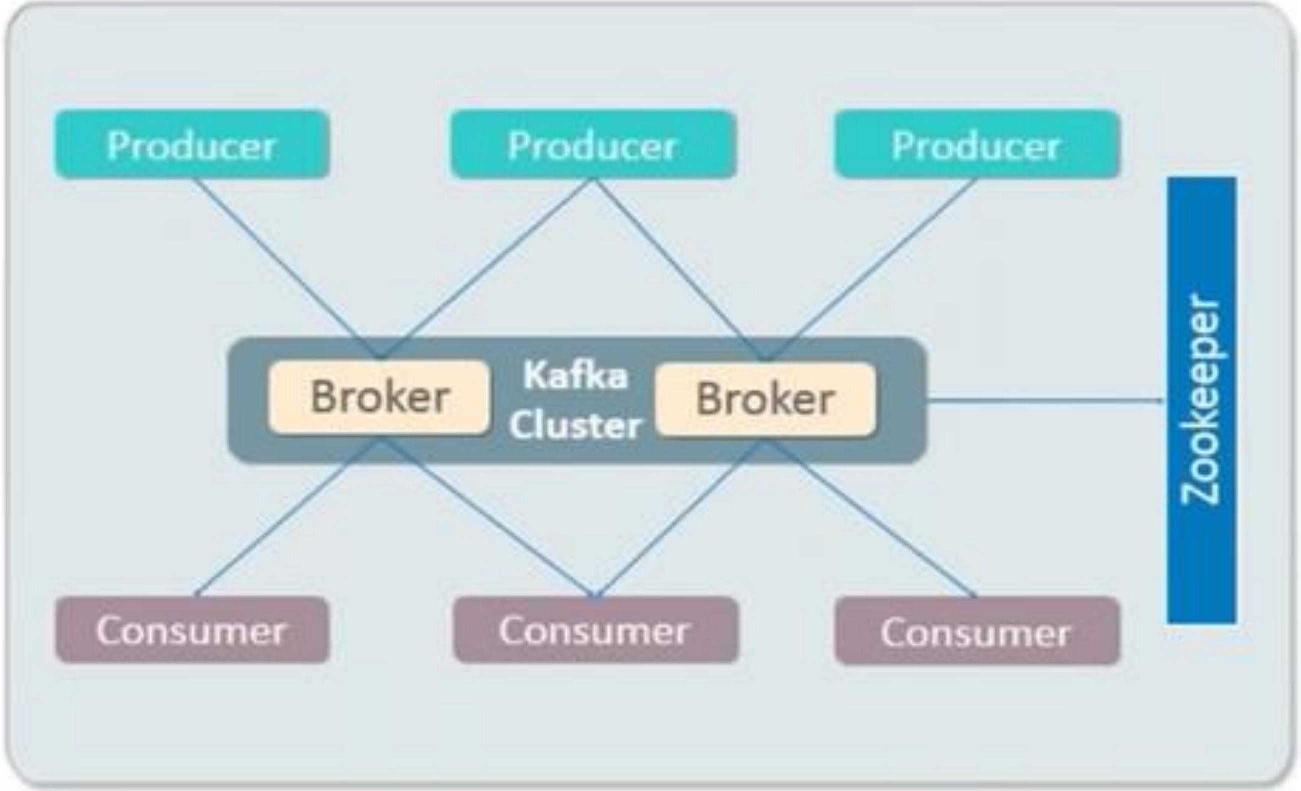
- *Consumers*(subscribers or readers) read messages
- The consumer subscribes to one or more topics and reads the messages sequentially
- The consumer keeps track of the messages it has consumed by keeping track on the offset of messages
- The *offset* is bit of metadata(an integer value that continually increases)that Kafka adds to each message as it is produced
- Each partition has a *unique offset* which is stored
- With the offset of the last consumed message, a consumer can *stop and restart without losing its current state*

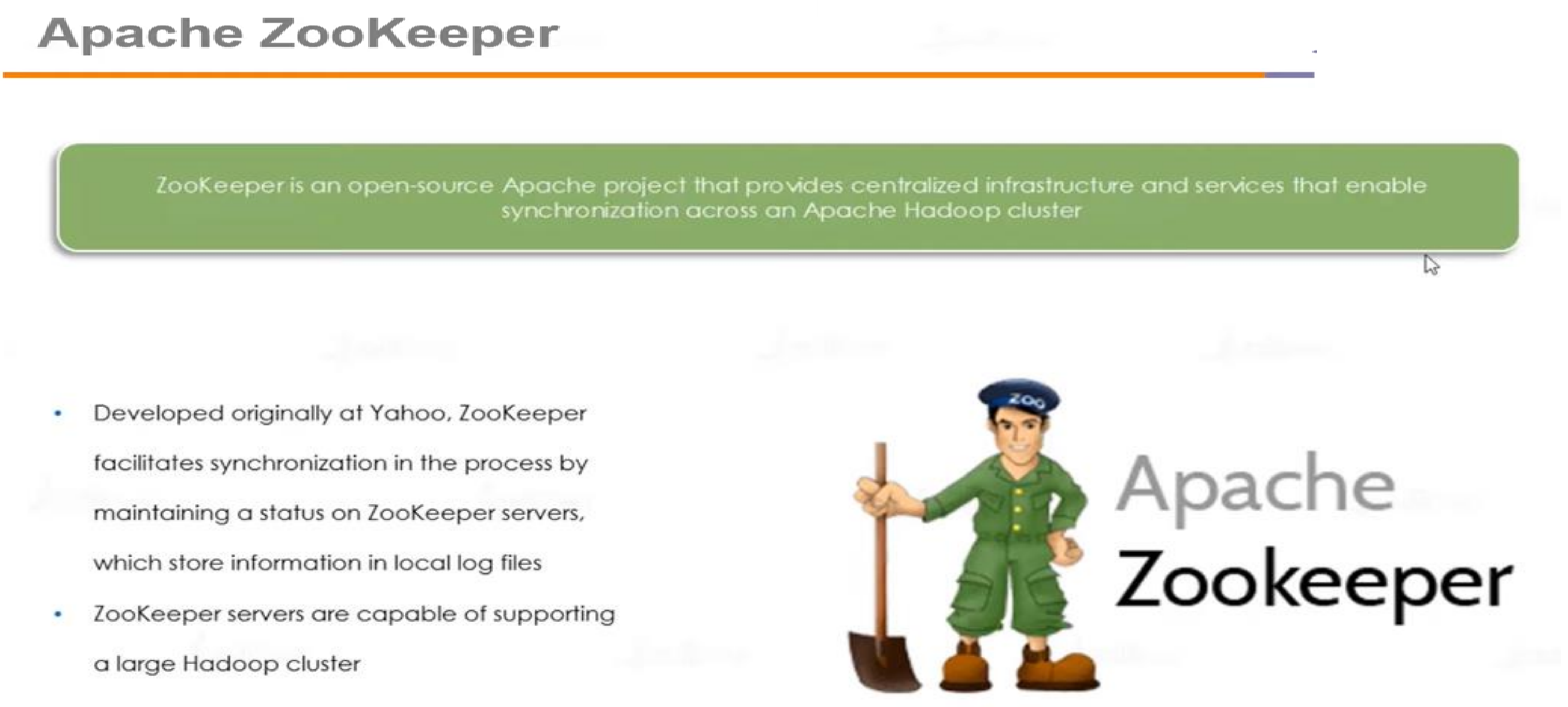


Kafka Components - ZooKeeper

ZooKeeper is used for managing and coordinating Kafka broker

- Zookeeper service is mainly used for co-ordinating between brokers in the Kafka cluster
- Kafka cluster is connected to ZooKeeper to get information about any failure nodes



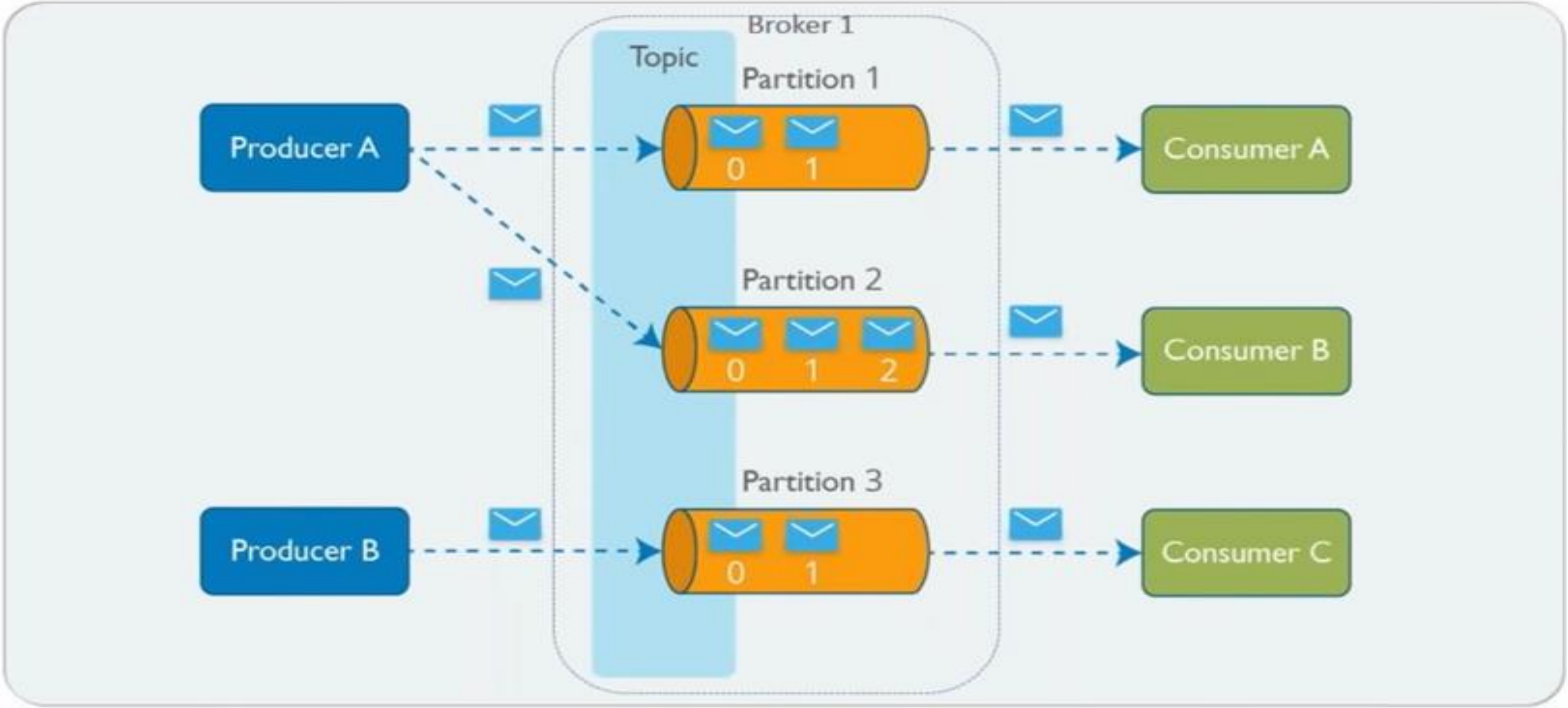


Apache ZooKeeper

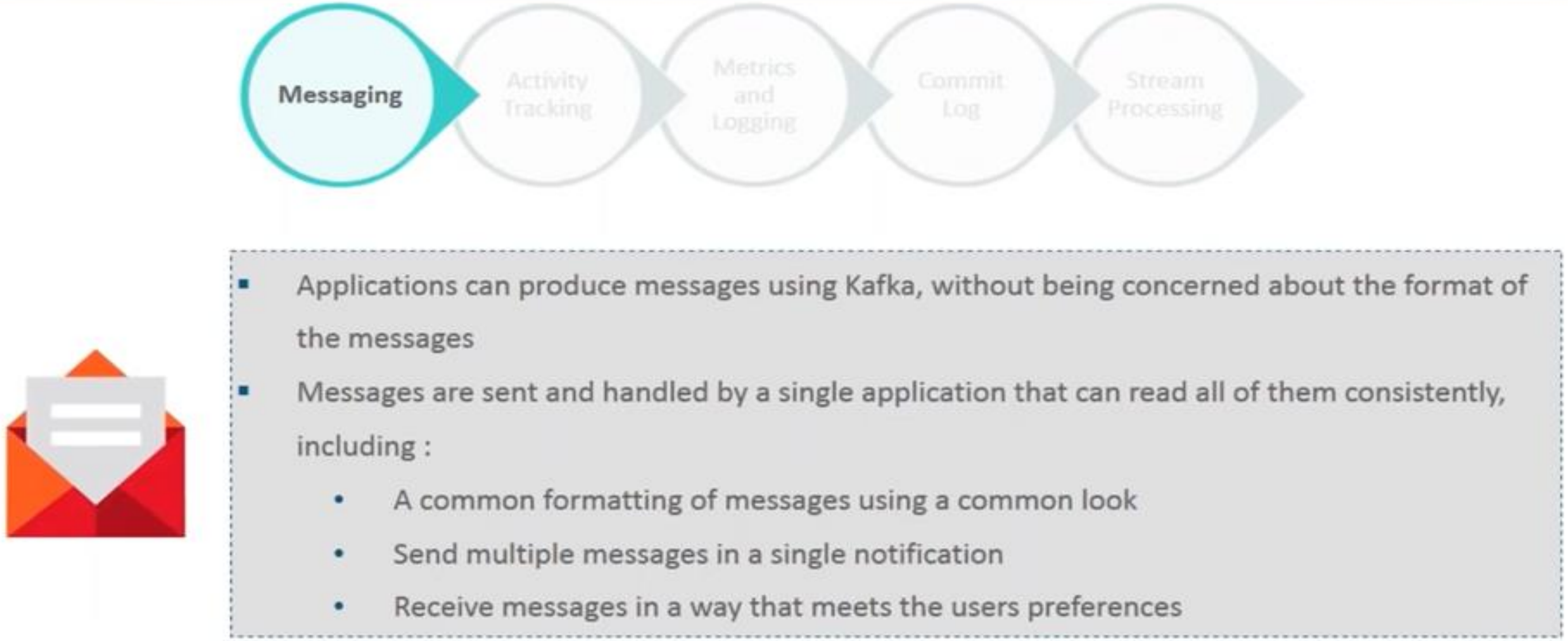
ZooKeeper is an open-source Apache project that provides centralized infrastructure and services that enable synchronization across an Apache Hadoop cluster

- Developed originally at Yahoo, ZooKeeper facilitates synchronization in the process by maintaining a status on ZooKeeper servers, which store information in local log files
- ZooKeeper servers are capable of supporting a large Hadoop cluster

Kafka Architecture



Kafka - Use Cases



- Applications can produce messages using Kafka, without being concerned about the format of the messages
- Messages are sent and handled by a single application that can read all of them consistently, including :
  - A common formatting of messages using a common look
  - Send multiple messages in a single notification
  - Receive messages in a way that meets the users preferences

Kafka - Use Cases

Messaging

Activity Tracking

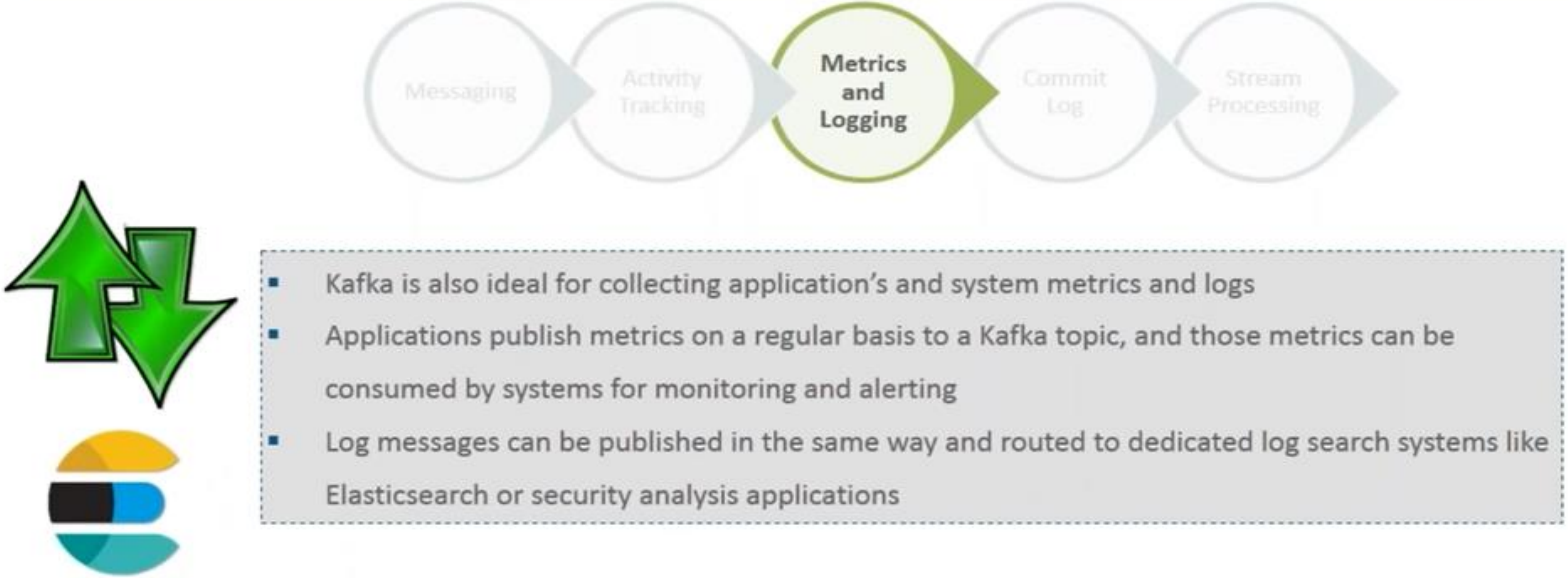
Metrics and Logging

Commit Log

Stream Processing

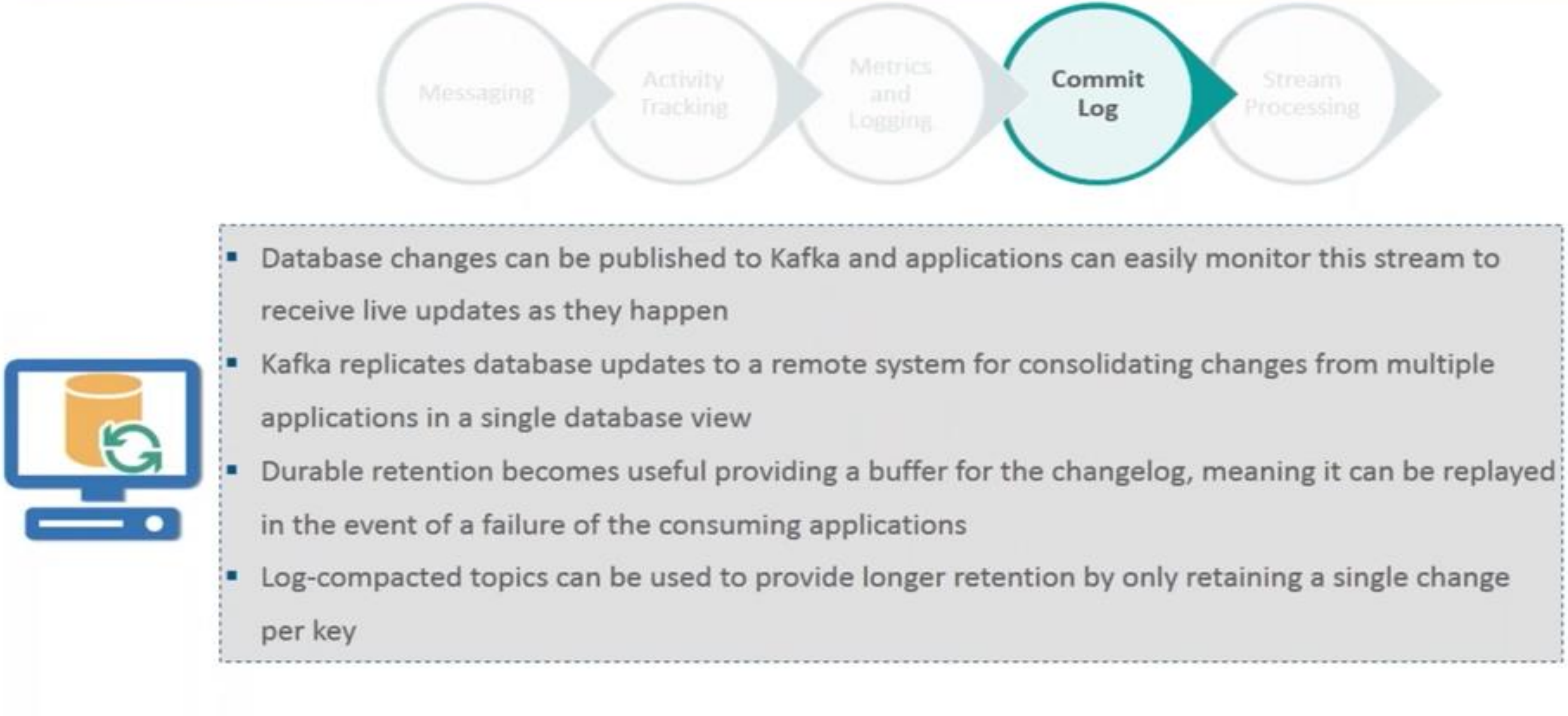
- Originally Kafka was designed at LinkedIn, to track user activity
- When a user interacts with frontend applications, which generates messages regarding actions the user is taking
- Kafka keeps track of simple information like click tracking to complex information like data in a user’s profile

Kafka - Use Cases



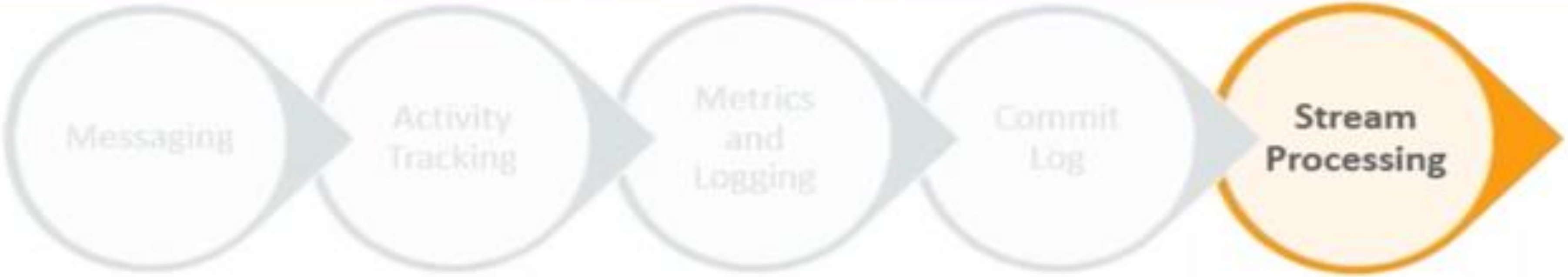
- Kafka is also ideal for collecting application's and system metrics and logs
- Applications publish metrics on a regular basis to a Kafka topic, and those metrics can be consumed by systems for monitoring and alerting
- Log messages can be published in the same way and routed to dedicated log search systems like Elasticsearch or security analysis applications

Kafka - Use Cases



- Database changes can be published to Kafka and applications can easily monitor this stream to receive live updates as they happen
- Kafka replicates database updates to a remote system for consolidating changes from multiple applications in a single database view
- Durable retention becomes useful providing a buffer for the changelog, meaning it can be replayed in the event of a failure of the consuming applications
- Log-compacted topics can be used to provide longer retention by only retaining a single change per key

Kafka - Use Cases

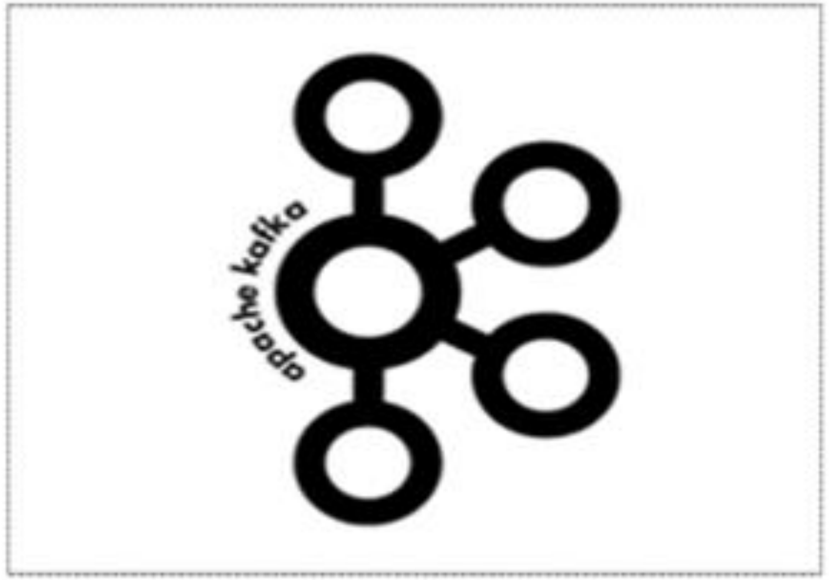


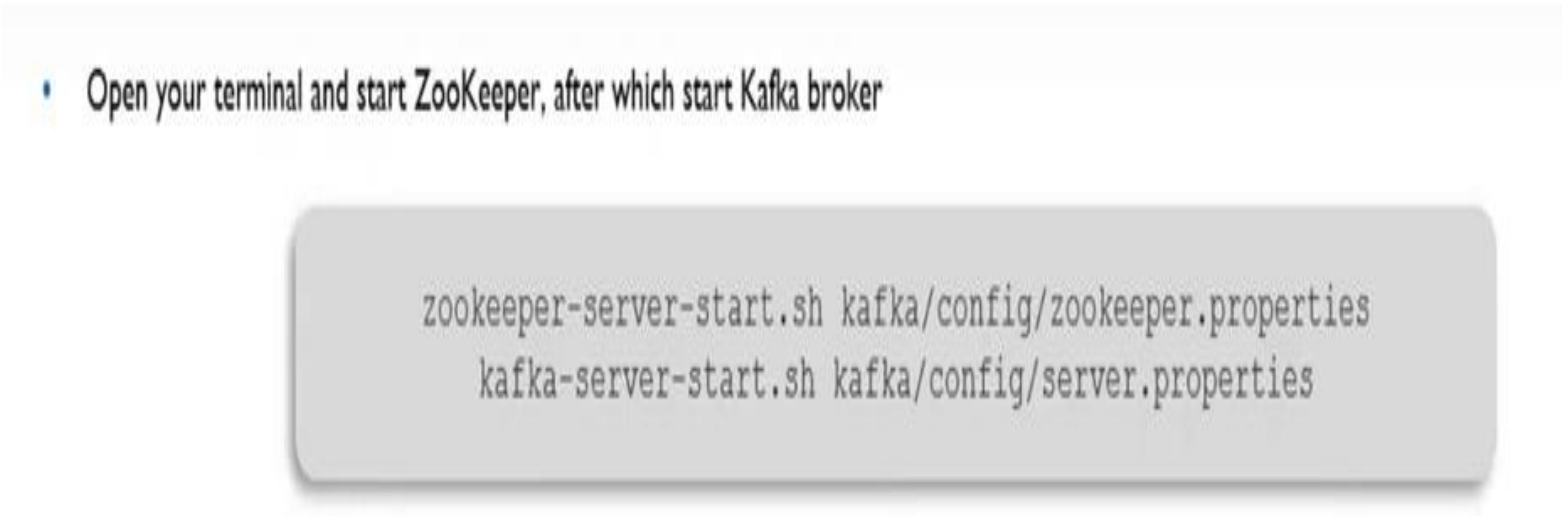
- Stream processing term is typically used to refer applications that provide similar functionality to map/reduce processing in Hadoop
- Stream processing operates on data in real-time, as quickly as messages are produced :
  - Write small applications to operate on Kafka messages,
  - Performing tasks such as counting metrics
  - Partitioning messages for efficient processing by other applications

Getting Started with Kafka

- **Prerequisites :**

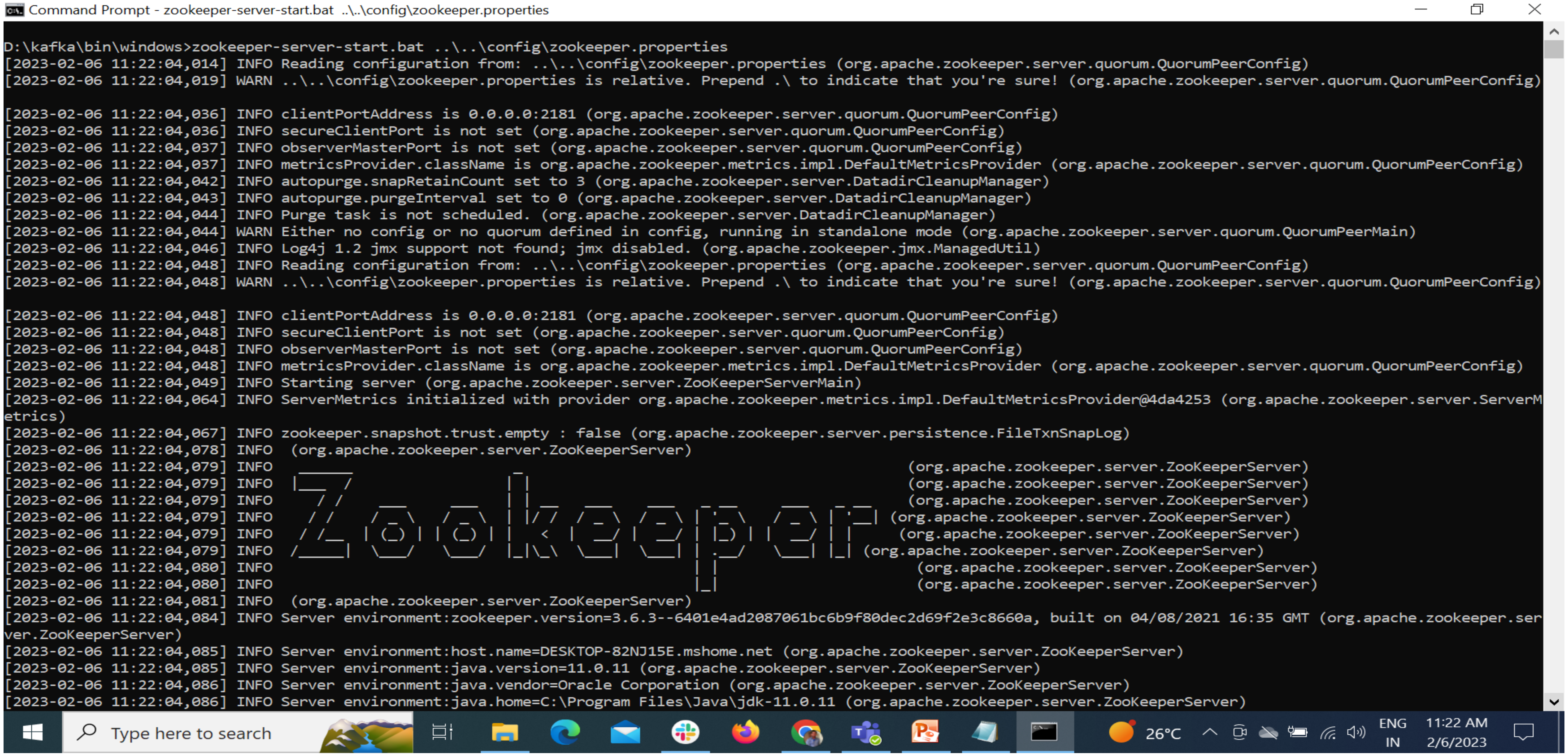
- **Components :**





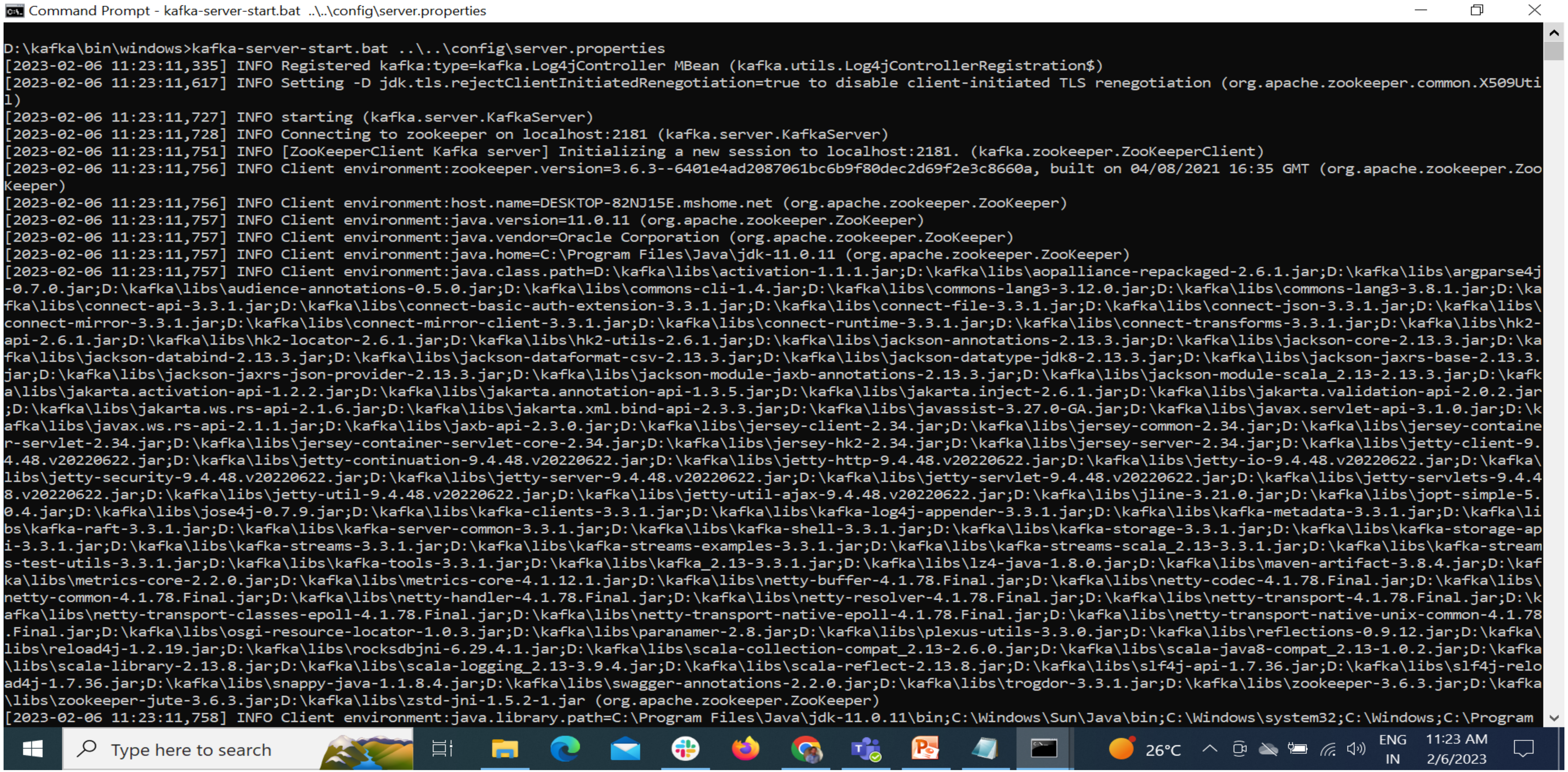
- Open your terminal and start ZooKeeper, after which start Kafka broker

```
zookeeper-server-start.sh kafka/config/zookeeper.properties
kafka-server-start.sh kafka/config/server.properties
```



Command Prompt - zookeeper-server-start.bat ..\..\config\zookeeper.properties

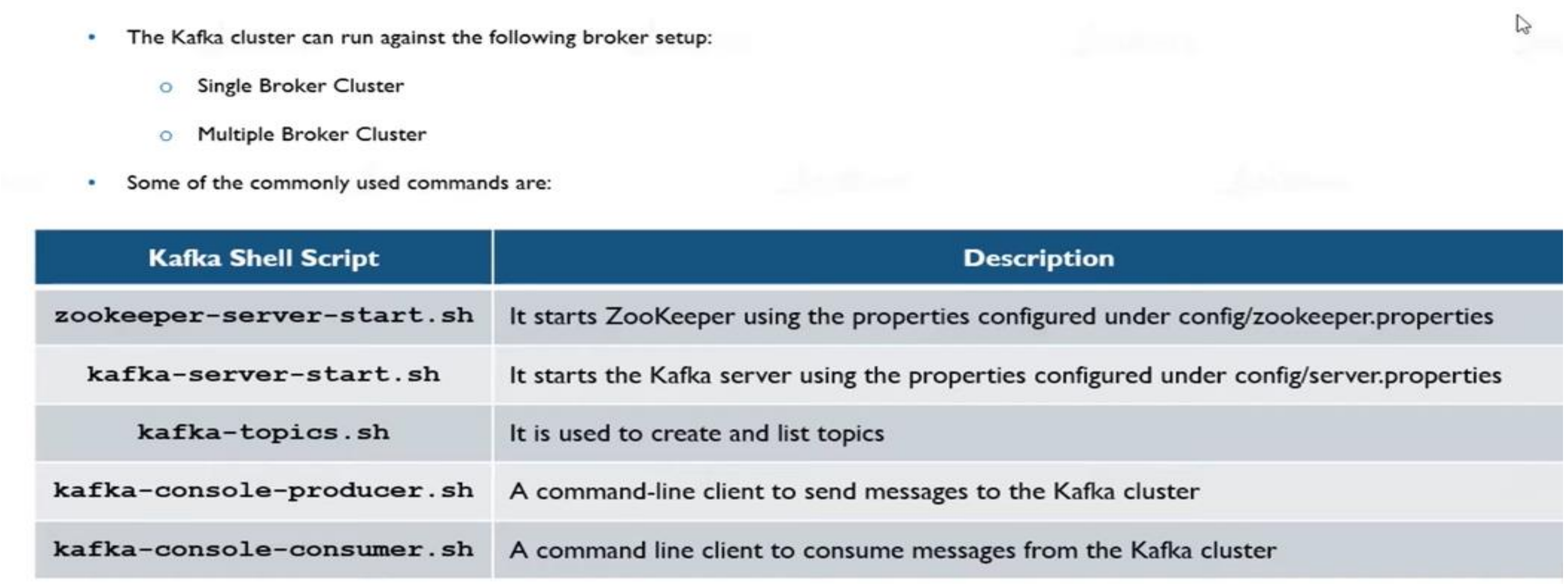
```
D:\kafka\bin\windows>zookeeper-server-start.bat ..\..\config\zookeeper.properties
[2023-02-06 11:22:04,014] INFO Reading configuration from: ..\..\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,019] WARN ..\..\config\zookeeper.properties is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2023-02-06 11:22:04,036] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,036] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,037] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,037] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,042] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2023-02-06 11:22:04,043] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2023-02-06 11:22:04,044] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2023-02-06 11:22:04,044] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2023-02-06 11:22:04,046] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2023-02-06 11:22:04,048] INFO Reading configuration from: ..\..\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,048] WARN ..\..\config\zookeeper.properties is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2023-02-06 11:22:04,048] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,048] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,048] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,048] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-02-06 11:22:04,049] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2023-02-06 11:22:04,064] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@4da4253 (org.apache.zookeeper.server.ServerM
etrics)
[2023-02-06 11:22:04,067] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2023-02-06 11:22:04,078] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,079] INFO   ______                  _                                           (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,079] INFO  |___  /                 | |                                          (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,079] INFO     / /    ___     ___   | | __   ___    ___   _ __     ___   _ __    (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,079] INFO    / /    / _ \   / _ \  | |/ /  / _ \  / _ \ | '_ \   / _ \ | '__| (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,079] INFO   / /__  | (_) | | (_) | |   <  |  __/ |  __/ | |_) | |  __/ | |     (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,079] INFO  /_____|  \___/   \___/  |_|\_\  \___|  \___| | .__/   \___| |_| (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,080] INFO                                               | |                      (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,080] INFO                                               |_|                      (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,081] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,084] INFO Server environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache.zookeeper.ser
ver.ZooKeeperServer)
[2023-02-06 11:22:04,085] INFO Server environment:host.name=DESKTOP-82NJ15E.mshome.net (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,085] INFO Server environment:java.version=11.0.11 (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,086] INFO Server environment:java.vendor=Oracle Corporation (org.apache.zookeeper.server.ZooKeeperServer)
[2023-02-06 11:22:04,086] INFO Server environment:java.home=C:\Program Files\Java\jdk-11.0.11 (org.apache.zookeeper.server.ZooKeeperServer)
```



Command Prompt - kafka-server-start.bat ..\..\config\server.properties

```
D:\kafka\bin\windows>kafka-server-start.bat ..\..\config\server.properties
[2023-02-06 11:23:11,335] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2023-02-06 11:23:11,617] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Uti
l)
[2023-02-06 11:23:11,727] INFO starting (kafka.server.KafkaServer)
[2023-02-06 11:23:11,728] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2023-02-06 11:23:11,751] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2023-02-06 11:23:11,756] INFO Client environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache.zookeeper.Zoo
Keeper)
[2023-02-06 11:23:11,756] INFO Client environment:host.name=DESKTOP-82NJ15E.mshome.net (org.apache.zookeeper.ZooKeeper)
[2023-02-06 11:23:11,757] INFO Client environment:java.version=11.0.11 (org.apache.zookeeper.ZooKeeper)
[2023-02-06 11:23:11,757] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2023-02-06 11:23:11,757] INFO Client environment:java.home=C:\Program Files\Java\jdk-11.0.11 (org.apache.zookeeper.ZooKeeper)
[2023-02-06 11:23:11,757] INFO Client environment:java.class.path=D:\kafka\libs\activation-1.1.1.jar;D:\kafka\libs\aopalliance-repackaged-2.6.1.jar;D:\kafka\libs\argparse4j
-0.7.0.jar;D:\kafka\libs\audience-annotations-0.5.0.jar;D:\kafka\libs\commons-cli-1.4.jar;D:\kafka\libs\commons-lang3-3.12.0.jar;D:\kafka\libs\commons-lang3-3.8.1.jar;D:\ka
fka\libs\connect-api-3.3.1.jar;D:\kafka\libs\connect-basic-auth-extension-3.3.1.jar;D:\kafka\libs\connect-file-3.3.1.jar;D:\kafka\libs\connect-json-3.3.1.jar;D:\kafka\libs\
connect-mirror-3.3.1.jar;D:\kafka\libs\connect-mirror-client-3.3.1.jar;D:\kafka\libs\connect-runtime-3.3.1.jar;D:\kafka\libs\connect-transforms-3.3.1.jar;D:\kafka\libs\hk2-
api-2.6.1.jar;D:\kafka\libs\hk2-locator-2.6.1.jar;D:\kafka\libs\hk2-utils-2.6.1.jar;D:\kafka\libs\jackson-annotations-2.13.3.jar;D:\kafka\libs\jackson-core-2.13.3.jar;D:\ka
fka\libs\jackson-databind-2.13.3.jar;D:\kafka\libs\jackson-dataformat-csv-2.13.3.jar;D:\kafka\libs\jackson-datatype-jdk8-2.13.3.jar;D:\kafka\libs\jackson-jaxrs-base-2.13.3.
jar;D:\kafka\libs\jackson-jaxrs-json-provider-2.13.3.jar;D:\kafka\libs\jackson-module-jaxb-annotations-2.13.3.jar;D:\kafka\libs\jackson-module-scala_2.13-2.13.3.jar;D:\kafk
a\libs\jakarta.activation-api-1.2.2.jar;D:\kafka\libs\jakarta.annotation-api-1.3.5.jar;D:\kafka\libs\jakarta.inject-2.6.1.jar;D:\kafka\libs\jakarta.validation-api-2.0.2.jar
;D:\kafka\libs\jakarta.ws.rs-api-2.1.6.jar;D:\kafka\libs\jakarta.xml.bind-api-2.3.3.jar;D:\kafka\libs\javassist-3.27.0-GA.jar;D:\kafka\libs\javax.servlet-api-3.1.0.jar;D:\k
afka\libs\javax.ws.rs-api-2.1.1.jar;D:\kafka\libs\jaxb-api-2.3.0.jar;D:\kafka\libs\jersey-client-2.34.jar;D:\kafka\libs\jersey-common-2.34.jar;D:\kafka\libs\jersey-containe
r-servlet-2.34.jar;D:\kafka\libs\jersey-container-servlet-core-2.34.jar;D:\kafka\libs\jersey-hk2-2.34.jar;D:\kafka\libs\jersey-server-2.34.jar;D:\kafka\libs\jetty-client-9.
4.48.v20220622.jar;D:\kafka\libs\jetty-continuation-9.4.48.v20220622.jar;D:\kafka\libs\jetty-http-9.4.48.v20220622.jar;D:\kafka\libs\jetty-io-9.4.48.v20220622.jar;D:\kafka\
libs\jetty-security-9.4.48.v20220622.jar;D:\kafka\libs\jetty-server-9.4.48.v20220622.jar;D:\kafka\libs\jetty-servlet-9.4.48.v20220622.jar;D:\kafka\libs\jetty-servlets-9.4.4
8.v20220622.jar;D:\kafka\libs\jetty-util-9.4.48.v20220622.jar;D:\kafka\libs\jetty-util-ajax-9.4.48.v20220622.jar;D:\kafka\libs\jline-3.21.0.jar;D:\kafka\libs\jopt-simple-5.
0.4.jar;D:\kafka\libs\jose4j-0.7.9.jar;D:\kafka\libs\kafka-clients-3.3.1.jar;D:\kafka\libs\kafka-log4j-appender-3.3.1.jar;D:\kafka\libs\kafka-metadata-3.3.1.jar;D:\kafka\li
bs\kafka-raft-3.3.1.jar;D:\kafka\libs\kafka-server-common-3.3.1.jar;D:\kafka\libs\kafka-shell-3.3.1.jar;D:\kafka\libs\kafka-storage-3.3.1.jar;D:\kafka\libs\kafka-storage-ap
i-3.3.1.jar;D:\kafka\libs\kafka-streams-3.3.1.jar;D:\kafka\libs\kafka-streams-examples-3.3.1.jar;D:\kafka\libs\kafka-streams-scala_2.13-3.3.1.jar;D:\kafka\libs\kafka-stream
s-test-utils-3.3.1.jar;D:\kafka\libs\kafka-tools-3.3.1.jar;D:\kafka\libs\kafka_2.13-3.3.1.jar;D:\kafka\libs\lz4-java-1.8.0.jar;D:\kafka\libs\maven-artifact-3.8.4.jar;D:\kaf
ka\libs\metrics-core-2.2.0.jar;D:\kafka\libs\metrics-core-4.1.12.1.jar;D:\kafka\libs\netty-buffer-4.1.78.Final.jar;D:\kafka\libs\netty-codec-4.1.78.Final.jar;D:\kafka\libs\
netty-common-4.1.78.Final.jar;D:\kafka\libs\netty-handler-4.1.78.Final.jar;D:\kafka\libs\netty-resolver-4.1.78.Final.jar;D:\kafka\libs\netty-transport-4.1.78.Final.jar;D:\k
afka\libs\netty-transport-classes-epoll-4.1.78.Final.jar;D:\kafka\libs\netty-transport-native-epoll-4.1.78.Final.jar;D:\kafka\libs\netty-transport-native-unix-common-4.1.78
.Final.jar;D:\kafka\libs\osgi-resource-locator-1.0.3.jar;D:\kafka\libs\paranamer-2.8.jar;D:\kafka\libs\plexus-utils-3.3.0.jar;D:\kafka\libs\reflections-0.9.12.jar;D:\kafka\
libs\reload4j-1.2.19.jar;D:\kafka\libs\rocksdbjni-6.29.4.1.jar;D:\kafka\libs\scala-collection-compat_2.13-2.6.0.jar;D:\kafka\libs\scala-java8-compat_2.13-1.0.2.jar;D:\kafka
\libs\scala-library-2.13.8.jar;D:\kafka\libs\scala-logging_2.13-3.9.4.jar;D:\kafka\libs\scala-reflect-2.13.8.jar;D:\kafka\libs\slf4j-api-1.7.36.jar;D:\kafka\libs\slf4j-relo
ad4j-1.7.36.jar;D:\kafka\libs\snappy-java-1.1.8.4.jar;D:\kafka\libs\swagger-annotations-2.2.0.jar;D:\kafka\libs\trogdor-3.3.1.jar;D:\kafka\libs\zookeeper-3.6.3.jar;D:\kafka
\libs\zookeeper-jute-3.6.3.jar;D:\kafka\libs\zstd-jni-1.5.2-1.jar (org.apache.zookeeper.ZooKeeper)
[2023-02-06 11:23:11,758] INFO Client environment:java.library.path=C:\Program Files\Java\jdk-11.0.11\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Program
```

Kafka Command-line Tools



- The Kafka cluster can run against the following broker setup:
  - Single Broker Cluster
  - Multiple Broker Cluster
- Some of the commonly used commands are:

Kafka Shell Script	Description
`zookeeper-server-start.sh`	It starts ZooKeeper using the properties configured under config/zookeeper.properties
`kafka-server-start.sh`	It starts the Kafka server using the properties configured under config/server.properties
`kafka-topics.sh`	It is used to create and list topics
`kafka-console-producer.sh`	A command-line client to send messages to the Kafka cluster
`kafka-console-consumer.sh`	A command line client to consume messages from the Kafka cluster

Our Locations

WISSEN

USA

2325 Parklawn Drive Suite G
Waukesha, Wisconsin 53186
United States

Canada

157 Adelaide St W #338,
Toronto, ON
M5H 1P9

Mexico

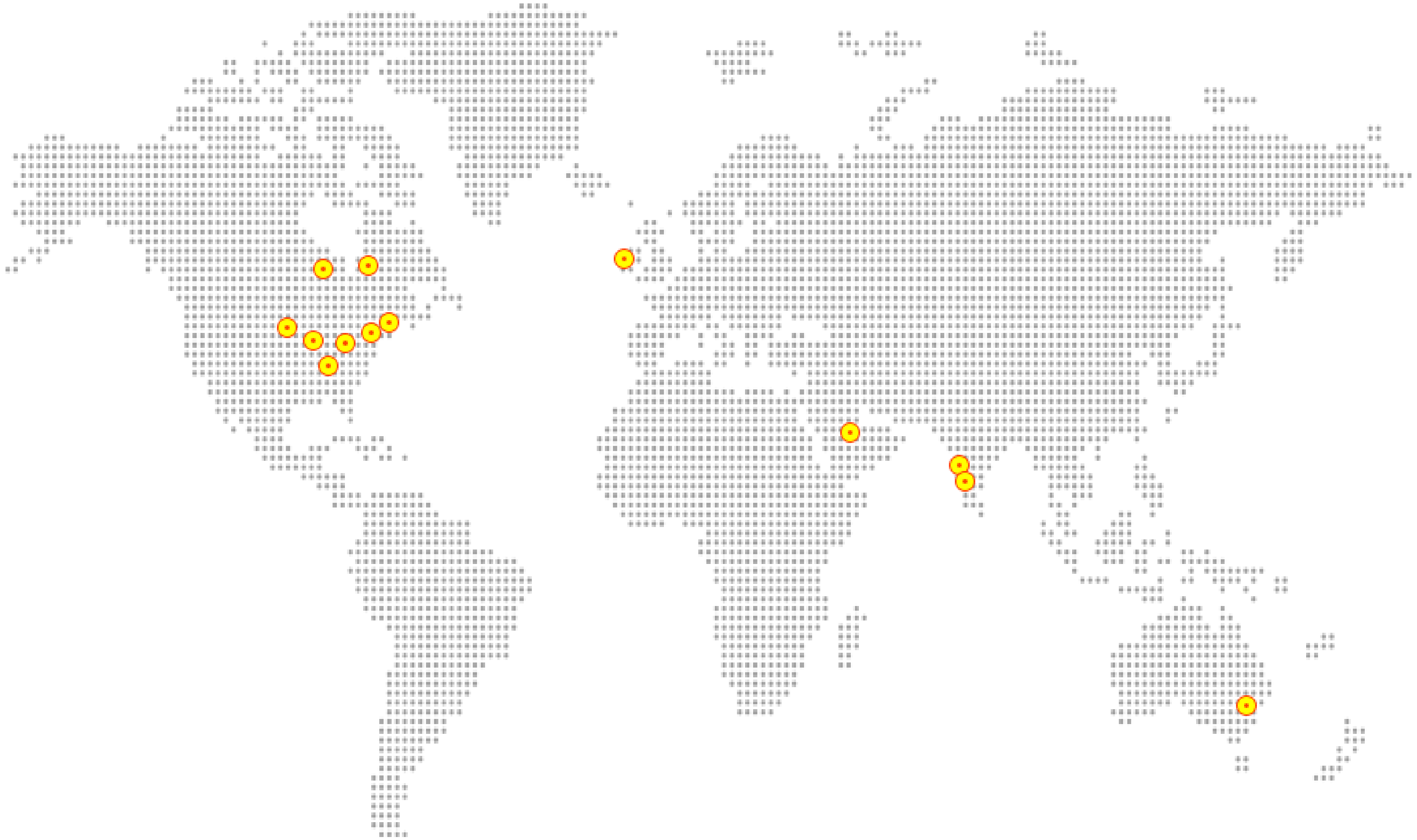
1600 Floor #4
Centre de Cuidad Santa Fe
Mexico City, D.F. 01210

Europe

10 Pappus House, Tollgate Business
park Tollgate West, Stanway
Colchester CO3 8AQ, United Kingdom

Australia

Wissen Australia Pty Ltd
19/105A Darling Point Road
DARLING POINT NSW 2027



Philippines

5TH Floor Zuellig Building, Paseo De
Roxas Avenue corner, Makati Avenue,
Makati City

India - Bangalore

#176, Adarsh Eco Place 4th Floor,
KIADB EPIP 2nd Phase Whitefield
Bangalore Karnataka, India

India - Mumbai

WeWork India Management Pvt Ltd,
5th Floor, Spectrum Tower, Mindspace,
Malad (W), Mumbai, Maharashtra
400064, India

India - Hyderabad

Q4, 9th Floor, Cyber Towers
Hi-tec City, Madhapur, Hyderabad
Telangana – 500081, India

India - Pune

Vatika Business Centre,
Level-5,C-Tower,Tech Park-I,
Airport Road,Yerwada,Pune-411006

USA | CANADA | MEXICO | UK | INDIA | AUSTRALIA | PHILIPPINES

Knowledge. Insight. Action