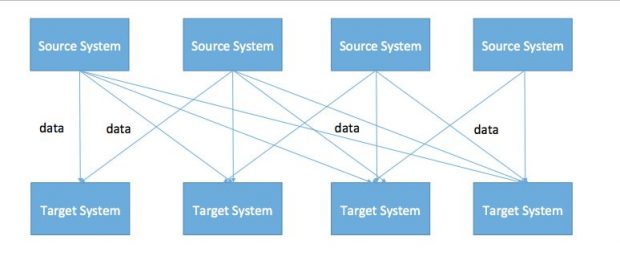
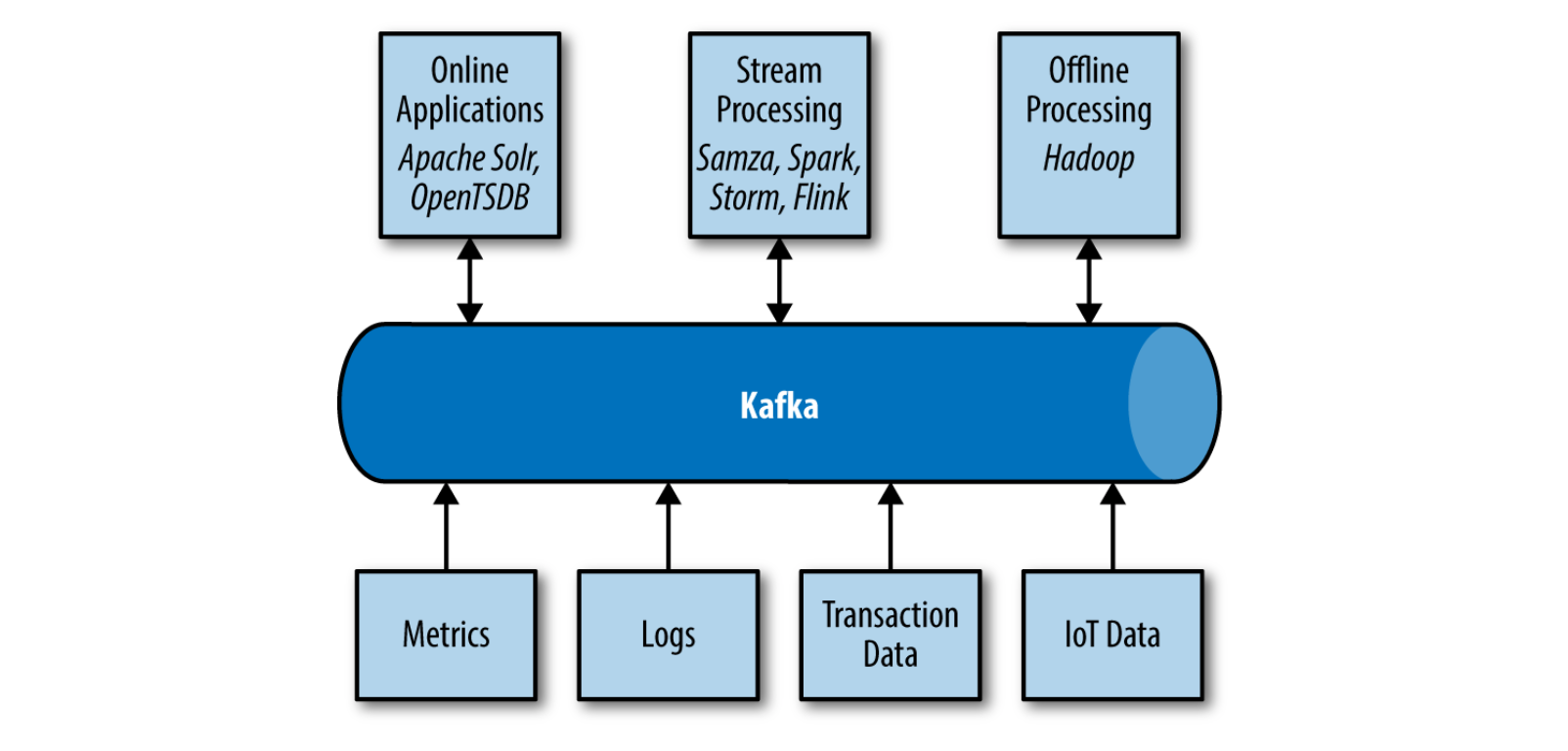
**Kafka Documentation**

Apache Kafka is an open-source distributed event streaming platform that is used to build real-time data pipelines and streaming applications. Kafka provides a scalable and fault-tolerant system for handling large volumes of data in real-time.

**Without Kafka –create a complexity in managing the system.**



**With Kafka –it become easier to handle complex management**



Kafka is made up of several components that work together to provide a complete streaming platform. The main components of Kafka are:

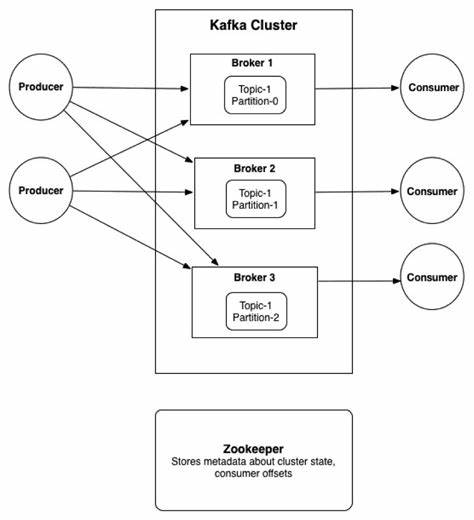
Before we move into the components we need to know about the messaging system

**Messaging system** is the responsible for transferring data from one application to another so the application can be focused on data without getting bogged down on data transmission and sharing there are two types of messaging system

:-

**Point to point messaging system** - under this the message are persisted in a queue Where there is a sender and a receiver in which a message can be received by only one receiver only. in this system a receiver can acknowledged back to sender.

**Publish subscribe messaging system**- In publish subscribe messaging system message are persisted in a topic and up publisher send data and it can be consumed by many subscribers but they are have some time limitation to consume that information and Can't acknowledged the publisher and this is system is followed by Kafka



**Brokers:** Storage node in Kafka Brokers are the Kafka servers that manage the storage and processing of messages. Each broker hosts one or more partitions of one or more topic and can communicate with other brokers to replicate and distribute data for fault tolerance.

**Topics:** A topic is a category or stream name to which messages are published by producers and from which messages are consumed by consumers. Each topic is divided into one or more partitions, which are distributed across the Kafka cluster for scalability and fault tolerance.

**Producers:** Producers are applications that generate data to be published to a Kafka topic. Producers publish messages to a topic and can choose to include a key with each message, which determines the partition to which the message will be sent.

**Consumers:** Consumers are applications that subscribe to one or more Kafka topics and consume messages from them. Consumers can be part of a consumer group, which allows them to share the load of consuming messages from a topic

**Zookeeper:** Zookeeper is a distributed coordination service that Kafka uses for store meta data like maintaining configuration data, tracking brokers and consumers, and managing cluster membership.

**Connectors:** Connectors are plugins that allow Kafka to integrate with other systems or data sources. Connectors can be used to import data from a database, export data to a data warehouse, or integrate with other messaging systems.

**Streams:** Kafka Streams is a library for building stream processing applications using Kafka. It allows developers to create complex data processing pipelines that can process streams of data in real-time.

Video Link reference - [What is Kafka | Messaging System | Distributed Environment | Apache Kafka Tutorial in Hindi |Video 1](https://youtu.be/TGKUYoegrDY)

[](https://youtu.be/TGKUYoegrDY)

To set up and use Kafka, we will need to meet certain prerequisites, such as:

**Java Development Kit**: Kafka is written in Java and requires a Java Development Kit (JDK) to be installed on the system.

**Zookeeper:** Kafka uses Zookeeper to coordinate cluster membership and requires a running Zookeeper instance to function properly.

**Network Configuration:** Kafka is a distribution system that requires proper network configuration to function. The Kafka brokers and Zookeeper nodes must be able to communicate with each other over the network, and clients must be able to connect to the Kafka brokers to produce and consume messages. Need to config file to set that up server of zookeeper and Kafka

**Installation Step**

Before everything we need to check the system requirements: Ensure that our system meets the minimum system requirements for installing and running Kafka, Zookeeper, and Schema Registry.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Nodes** | **Storage** | **Memory** | **CPU** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Broker** | 3 | * 12 X 1 TB disk. RAID 10 is optional * Separate OS disks from Apache Kafka® storage | 4 GB RAM | 2 core sockets |
| **Connect** | 2 | Only required for installation | 0.5 - 4 GB heap size depending on connectors | Typically not CPU- bound. More cores is better than faster cores. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Schema Registry** | 2 | Only required for installation | 1 GB heap size | Typically not CPU- bound. More cores is better than faster cores. |
| **ZooKeeper** | 3-5 | * Transaction log: 512 GB * Storage: 2 X 1 TB SATA, RAID 10   Each write to ZooKeeper must be persisted in the transaction log before the client gets an ack.  Using SSD reduces the ZooKeeper write latency. | 2-4 GB RAM | 2 cores |

Process 1:- for Apache kafka

Installation of Java because Kafka is built on top of it: Kafka, Zookeeper, and Schema Registry require Java to be installed on our system. Install the latest version of Java from the official website <https://www.java.com/en/download/>.

*Java 17 is the recommended version in this version Java 11 and Java 8 are also supported.*

*Java 8 is now deprecated and will be removed in a future version.*

**Download Kafka**: Download the latest version of Kafka from the official website <https://kafka.apache.org/downloads>.

After that we need to select binary package which is based on scala here just need to choose kafka latest version of 3.4.0 on scala version 1.13

**Extract the Kafka files:** Once you have downloaded Kafka, extract the files to a directory on your system. Using GUI or CLI.

After extraction, navigate to the Kafka directory using the following command:cd kafka\_<version> (in my case it will be 3.4.0)

It's a zip/tar file next here unzip the file with the given command:- tar -xzf kafka\_<version>.tgz

**Configure Kafka**: Before running Kafka, we need to configure it by editing the server. Properties file in the config directory of the Kafka installation. You can set various configurations such as the broker ID, port, log directory, and more.

**Install Zookeeper**: Kafka uses Zookeeper to manage its cluster. Download and install the latest version of Zookeeper from the official website.

**Extract the zookeeper files:** Once you have downloaded zookeeper, as like kafka we need to extract the zookeeper files to a directory on our system. Using GUI or CLI.

Here I am using command is :-tar -xzf zookeeper-<version>.tar.gz

**Configure Zookeeper:** Edit the zoo.cfg file in the conf directory of the Zookeeper installation to configure Zookeeper. You can set the data directory, client port, and more.

For example

Open the zoo.cfg file using a text editor. The file is typically located in the conf directory of your ZooKeeper installation.

Specify the port number on which ZooKeeper will listen for client connections. For example, to listen on port 2181, add the following line to the file:  *clientPort=2181*

Specify the data directory where ZooKeeper will store its data. For example, to use the directory "/var/lib/zookeeper/data", add the following line to the file: *dataDir=/var/lib/zookeeper/data*

Specify the directory where ZooKeeper will store its transaction log. For example, to use the directory "/var/lib/zookeeper/logs", add the following line to the file:  *dataLogDir=/var/lib/zookeeper/logs*

If you are running a ZooKeeper ensemble (a group of ZooKeeper servers that work together), specify the server IDs and their corresponding IP addresses and port numbers. For example, to configure a three-server ensemble with server IDs 1, 2, and 3, and IP addresses 192.168.1.1, 192.168.1.2, and 192.168.1.3, add the following lines to the file:

*server.1=192.168.1.1:2888:3888*

*server.2=192.168.1.2:2888:3888*

*server.3=192.168.1.3:2888:3888*

The first number in each line is the server ID, followed by the IP address and two port numbers. The first port number (2888) is the port that the server uses to communicate with other servers in the ensemble, while the second port number (3888) is the port used for leader election.

**Kafka server properties: kafka** need to run itself it needs some property server properties using that it will run itself by doing some changes in server.properties file --

Open the server.properties file using a text editor. The file is typically located in the config directory of your Kafka installation.

Specify the broker ID. Each broker in a Kafka cluster must have a unique ID. For example, to set the broker ID to 1, add the following line to the file: broker.id=1

Specify the ZooKeeper connection string. Kafka brokers use ZooKeeper to manage cluster membership and maintain topic configuration information. For example, to connect to a ZooKeeper ensemble running on three servers with IP addresses 192.168.1.1, 192.168.1.2, and 192.168.1.3, and listening on the default port (2181), add the following line to the file: zookeeper.connect=192.168.1.1:2181,192.168.1.2:2181,192.168.1.3:2181

Specify the Kafka data directory. This is where the broker stores its log files and other persistent data. For example, to use the directory /var/lib/kafka/data, add the following line to the file: log.dirs=/var/lib/kafka/data

(Optional) Specify other configurations such as the advertised listeners, the maximum message size, the number of partitions, and the default replication factor. These configurations can be added as needed based on your use case. For example:

advertised.listeners=PLAINTEXT://localhost:9092

message.max.bytes=10485760

num.partitions=3

default.replication.factor=2

**Start Zookeeper:** Start Zookeeper by running the zkServer.sh script in the bin directory of the Zookeeper installation. We can use to start in foreground or background

**Start Kafka brokers**: Start Kafka brokers by running the kafka-server-start.sh script in the bin directory of the Kafka installation. You can specify the server. Properties file to use with the --config option.

**Install Schema Registry**: Schema Registry is an optional component that you can install to manage the schemas of messages sent to Kafka. Download and install the latest version of Schema Registry from the official website.

**Configure Schema Registry**: Edit the schema-registry. Properties file in the config directory of the Schema Registry installation to configure Schema Registry. You can set the Kafka broker URL, port, and more.

{ Once you have configured your Kafka broker and ZooKeeper ensemble, you can proceed with configuring the Schema Registry. Here's an example of how to configure the Schema Registry:

Open the schema-registry.properties file using a text editor. The file is typically located in the config directory of your Schema Registry installation.

Specify the Kafka broker URL. This should point to the URL of the Kafka broker that the Schema Registry should use. For example: *kafkastore.bootstrap.servers=localhost:9092*

Specify the ZooKeeper URL. This should point to the URL of the ZooKeeper ensemble that the Schema Registry should use. For example: *kafkastore.connection.url=localhost:2181*

Specify the Schema Registry URL. This should point to the URL where the Schema Registry will be accessible. For example: *listeners=http://localhost:8081*

(Optional) Specify any other configurations such as the port number, the number of Schema Registry workers, and the maximum number of versions for a given schema. These configurations can be added as needed based on your use case. For example:

*port=8081*

*schema.registry.num.workers=1*

*max.schemas.per.subject=10}*

**Start Schema Registry:** Start Schema Registry by running the schema-registry-start.sh script in the bin directory of the Schema Registry installation.

Process2:- Apache kafka with confluent

After checking all the prerequisites and configurations of our system next thing what we have to do is

Step1: first of all is go to confluent website and logging into that or creator one account mentioned website

[Try Confluent for Free. Fully Managed Kafka Service on Any Cloud](https://www.confluent.io/get-started/?product=software)

Step 2: after creating the account in my case download tar file and untar using mention command :- tar –zxvf confluent-x.x.x.tar.gz (x.x.x -7.4.0)

Step3: now we have to create an environment variable (access confluent Kafka from anywhere using CLI) for that open :- vim .profile file in the linux home directory and type these commad code

export CONFLUENT\_HOME=/home/shashank-s-mac/confluent-7.4.0

export PATH=$PATH:$CONFLUENT\_HOME/bin

Step 4: restart os and type these command to check path

>> echo $CONFLUENT\_HOME

Step 5:Now finally installtion of confluent cli to work with confluent kafka using these command

curl -sL https://raw.githubusercontent.com/confluentinc/cli/main/install.sh | sudo sh -s -- -b /usr/local/bin