Apache Kafka

Apache Kafka is an open-source **stream-processing** software platform which is used to handle the real-time data storage. It works as a **broker** between two parties, i.e., a **sender and a receiver**. It can handle about trillions of data events in a day

Apache Kafka is a **distributed streaming platform** based on **publish/subscribe** messaging system. Another term which is used for Kafka is “distributed commit log”.

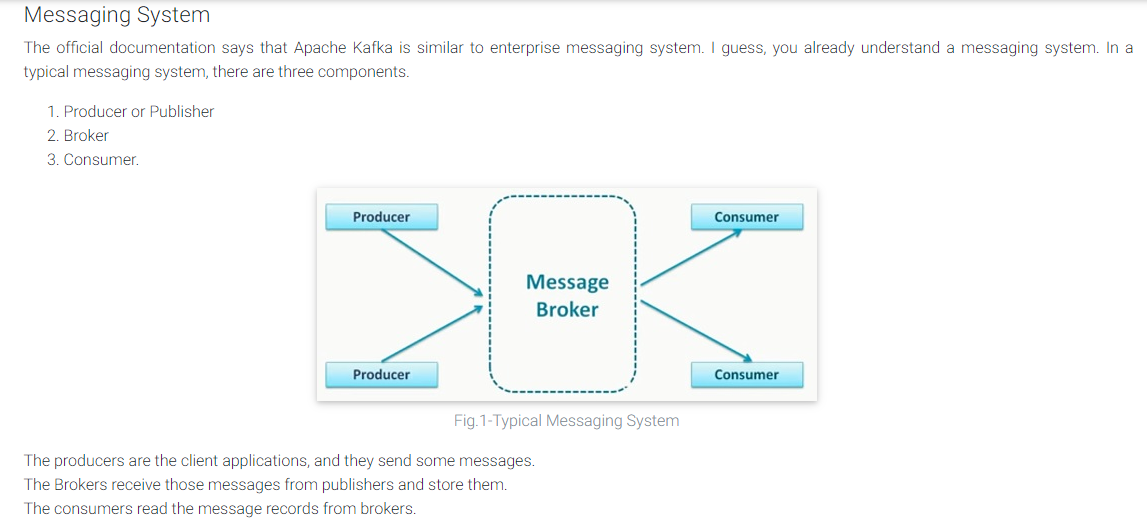
Just like we store the transactional data in database, so that we can retrieve it later to make some business decisions, Kafka also stores data in form of messages. Data within Kafka is stored **durably**, in order, and can be read deterministically.

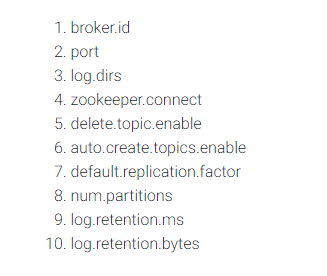
Apache Kafka is **publish-subscribe** based fault tolerant messaging system. It is fast, scalable and distributed by design.

Kafka is just like a messaging system.

**Apache Kafka is:**

* Scalable
* Fault tolerant
* A great publish-subscribe messaging system
* Capable of higher throughput compared with most messaging systems
* Highly durable
* Highly reliable
* High performant



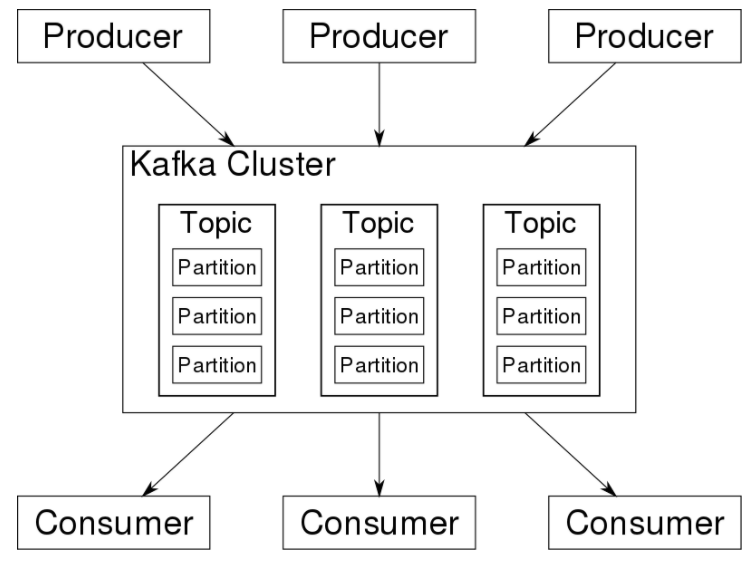


**Kafka Topic**

A Topic is a category/feed name to which records are stored and published.

All Kafka records are organized into topics. Producer applications write data to topics and consumer applications read from topics.

**Topics** are virtual groups of one or many [partitions](https://jaceklaskowski.gitbooks.io/apache-kafka/content/kafka-topics.html#partitions) across [Kafka brokers](https://jaceklaskowski.gitbooks.io/apache-kafka/content/kafka-brokers.html) in a Kafka cluster.



**n one line to help you out:** one could say that a topic in Kafka is the same ***concept*** as a table in a database. But it’s definitely not a table, and Kafka isn’t a database. Hence the word topic.

In a bit more details:

* a topic has a name, and it has to be unique across your Kafka cluster
* a topic is where data (messages) gets published to by a producer
* a topic is where data is pulled from by a consumer

To answer your question. You can have a topic “employees”, in which you publish some employee data, including their SSN. Or you can have a separate “ssn” topic, if you so dearly wish.

Behind the scene, your topic is divided in partitions, and each partition lives on one broker. Your topic is also replicated by a factor of your choice, for backup and high availability purposes.

### Kafka topic partition

Kafka topics are divided into a number of partitions, which contain records in an unchangeable sequence. Each record in a partition is assigned and identified by its unique offset. A topic can also have multiple partition logs. This allows multiple consumers to read from a topic in parallel.

Partitions are the main concurrency mechanism in Kafka. A topic is divided into 1 or more partitions, enabling producer and consumer loads to be scaled. Specifically, a consumer group supports as many consumers as partitions for a topic. Immutability

Default time is 7 days.

Offset are unique for respective partition numbers/integer number. (partition)

The offset is a simple integer number that is used by Kafka to maintain the current position of a consumer. That's it. The current offset is a pointer to the last record that Kafka has already sent to a consumer in the most recent poll. So, the consumer doesn't get the same record twice because of the current offset.

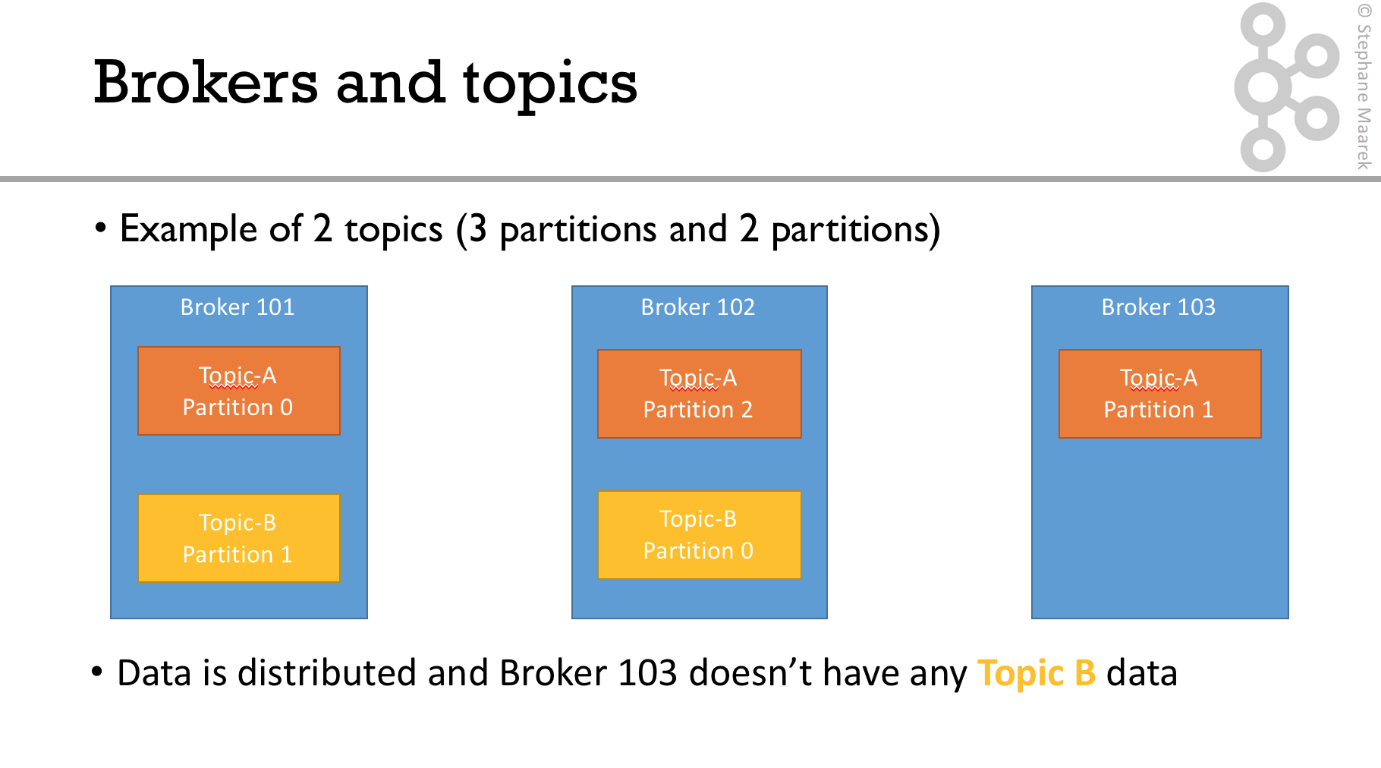
A Broker is a Kafka server that runs in a Kafka Cluster. Kafka Brokers form a cluster. The Kafka Cluster consists of many Kafka Brokers on many servers. Broker sometimes refer to more of a logical system or as Kafka as a whole.

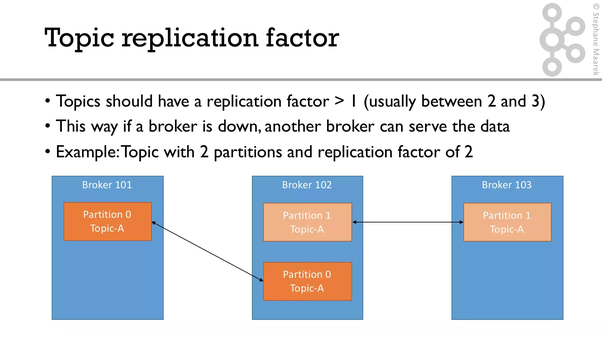
## Topic Replication

**Topic Replication** is the process to offer fail-over capability for a topic.

**Replication factor** defines the number of copies of a topic in a Kafka cluster.

Replication factor can be defined at topic level.





Based on Leader Consumer will access first

Partition - split the topic.

Replicating - make more than one copy of the messages in the topic.

Ex: if a topic has 2 partitions and 2 replications then it means the topic isn petitioned into 2 may be one broker or 2 brokers depending on your cluster config. And all the messages that come in is maintained as 2 copies.

## What is Zookeeper?

Zookeeper is a top-level software developed by Apache that acts as a centralized service and is used to maintain naming and configuration data and to provide flexible and robust synchronization within distributed systems. Zookeeper keeps track of status of the Kafka cluster nodes and it also keeps track of Kafka topics, partitions etc.

Zookeeper it self is allowing multiple clients to perform simultaneous reads and writes and acts as a shared configuration service within the system. The Zookeeper atomic broadcast (ZAB) protocol i s the brains of the whole system, making it possible for Zookeeper to act as an atomic broadcast system and issue orderly updates.

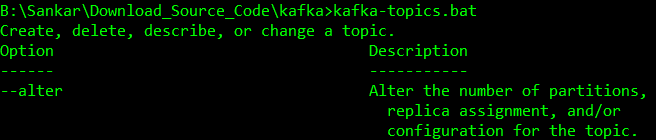
Coordination service in distributed system.

Managing all the brokers

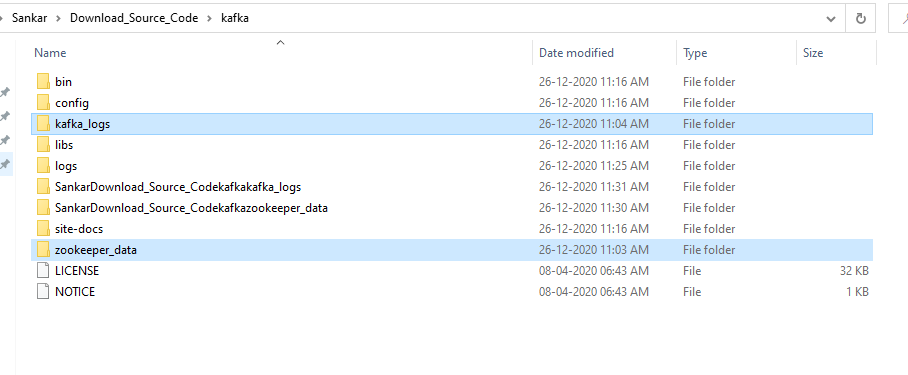
Finalizing the leaders of brokers.

Contain metadata of brokers.

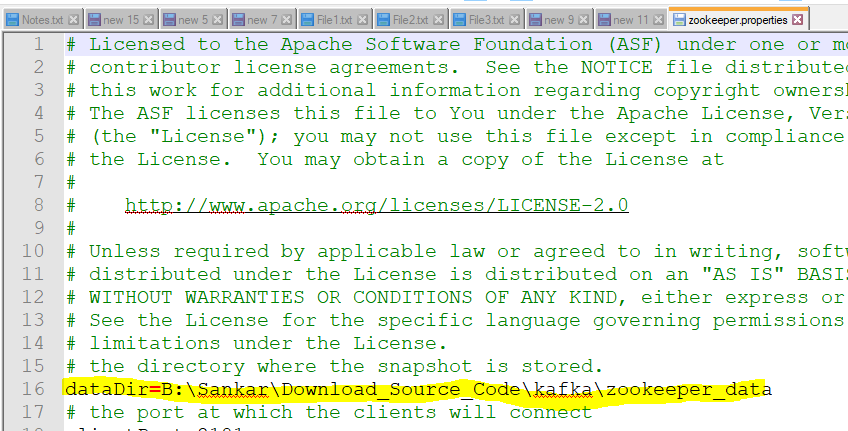
Reference to Download & Install <https://www.learningjournal.guru/article/kafka/installing-kafka-on-windows/>

1. Download and unzip and rename folder as kafka.
2. 

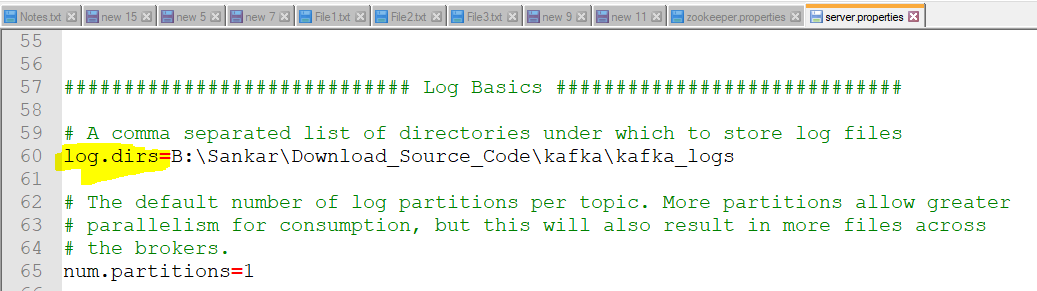
Create folders



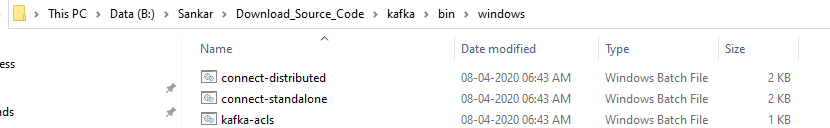
Edit zookeeper.properties

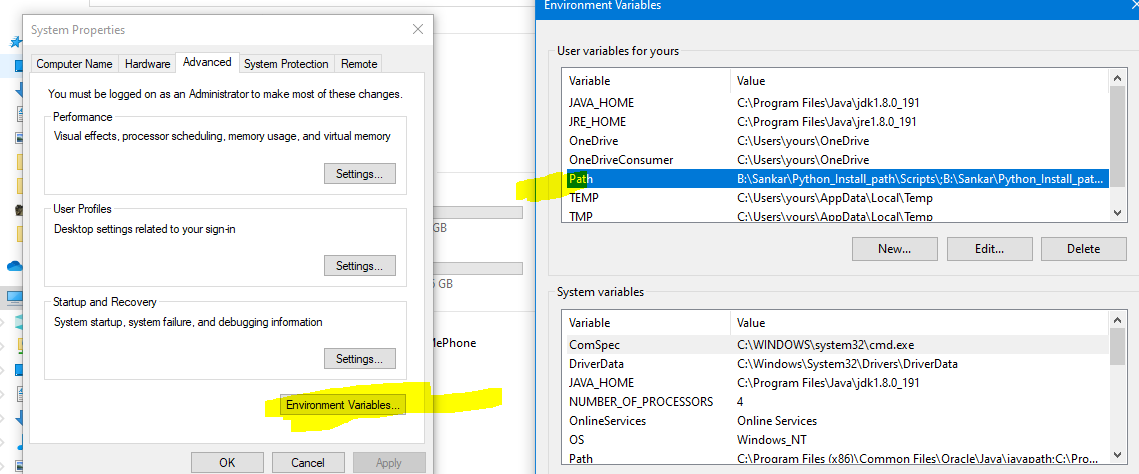


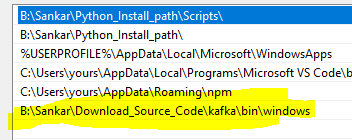
Edit server.properties



Copy path

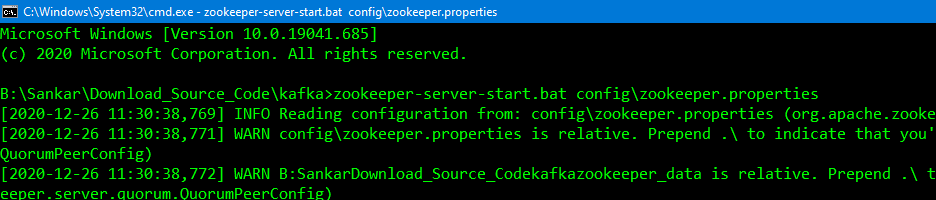






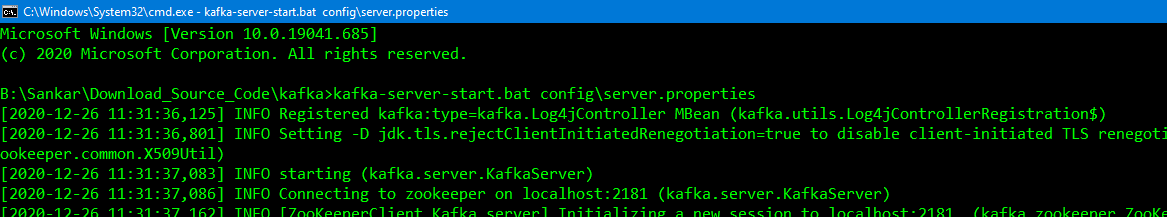
Step 1: START THE ZOOKEEPER WITH NEW CMD INSTANCE

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



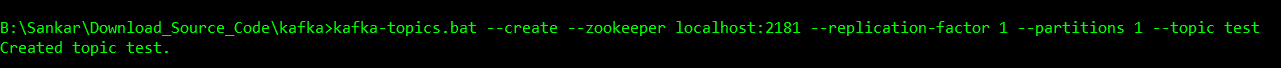
Step 2: START THE KAFKA WITH NEW CMD INSTANCE

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

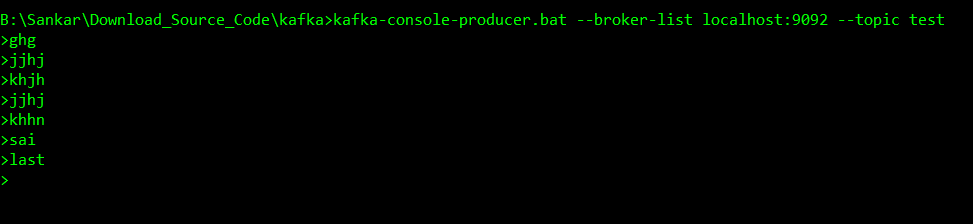


Step 3: **CREATE THE TOPIC WITH NEW CMD INSTANCE**

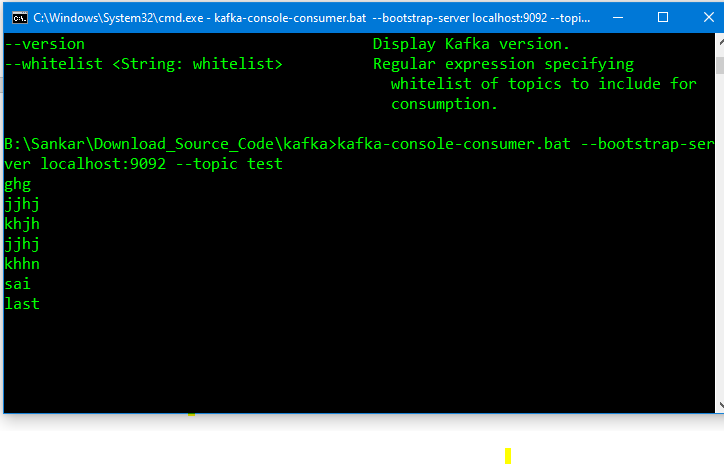
kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test



kafka-console-producer.bat --broker-list localhost:9092 --topic test



kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test



To check available topics

B:\Sankar\Download\_Source\_Code\kafka>kafka-topics.bat --list --zookeeper localhost:2181

SupplierTopic

\_\_consumer\_offsets

test

test1

test2

test5

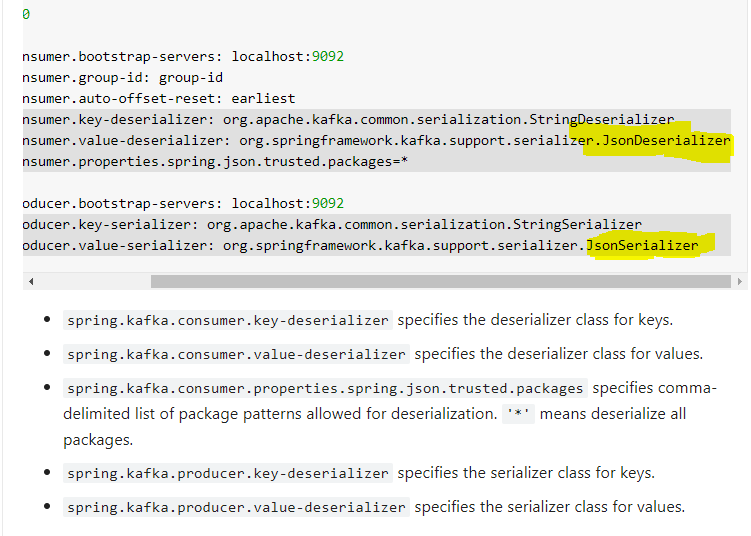
test8686

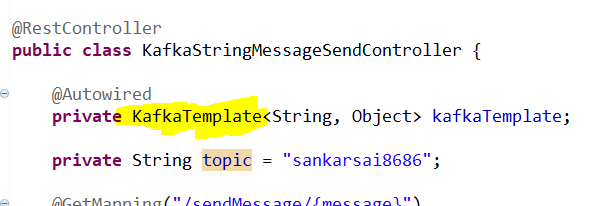
Kafka is a message queue product. Based on Topic partitions design, it can achieve very high performance of message sending and processing.

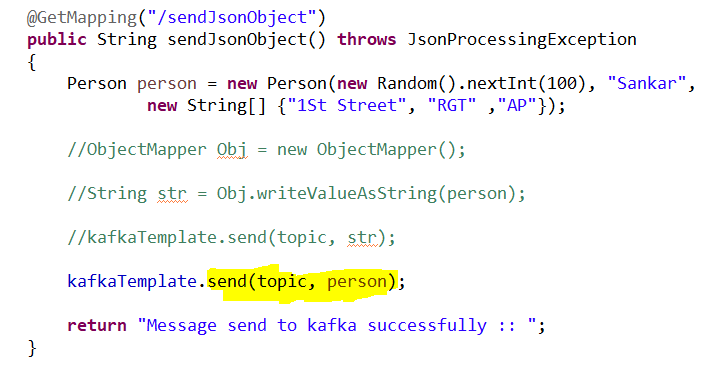
Kafka producer in SpringBoot.

1. Add kafka dependency
2. Create model class which we want to send
3. Mention properties like bootstrap servers and key & Value serilizers
4. Use kafka template to send data.

<https://howtodoinjava.com/kafka/spring-boot-jsonserializer-example/>







Without Using application.properties then option is java config

