Apache Kafka

Kafka - simply saying, Kafka is used to transporting data b/w different source systems to different
target systems by acting as a middle man

-> Kafka contains Topics -> while creating a topic need to specify the number of partitions
-> Topis -> Partions -> offset
-> Offset is specific to Kafka Partitions
-> Order is guaranteed within the partition and not across partitions
-> Data is kept only for a limited time in Kafka(default - one week)
-> Data written in the partition is immutable (can't be changed)

-> Kafka Cluster is composed of multiple brokers
-> Kafka Topic is distributed across the brokers
-> In case of replication factor is used -> rf = 2 , each partition in a topic will present in two brokers

-> At any time only one broker can be a leader for a given partition
-> only that leader can receive and serve data for a partition, other brokers sync data
-> Each Partition has one leader and multiple ISR (in sync replica)
-> Leader and ISR is determined by zookeeper

-> Kafka Producer -> produces data into Kafka topics
-> Producers can choose to receive acknowledgments
-> acks =0 producer won't wait for acknowledgments (possible data loss)
-> acks =1 producer waits for acknowledgments (limited data loss)
-> acks=all Leader + waits for replica acknowledgments (no data loss)

--> Producer can send key along with a message
-> if the key is null -> data is sent out in round-robin, if sent - all msg with that key always go to same
partition
-> If we need messaging order for a specific field -> producer can send key

-> Consumer consumes data from a topic (they know from the broker they need to read)
-> data is read in order within an each partition

Consumer Groups
-> consumer read data in consumer groups
-> Each consumer within a group read from an exclusive partition
-> if you have more consumers than partitions, some consumers will be inactive

Consumer Offsets
->Kakfa consumers in a consumer group post-processing from topic, they write offsets
in to kafkatopicname__consumer_offset table
->If a consumer dies it will be able to read back from where it is left

Delivery Semantics of consumer offsets - consumer chooses when to commit offsets
-> Atmost Once - Offsets are committed as soon as they receive the msg (if the processing goes the wrong msg
will be lost)
-> At least Once (preferred) - consumer commits offset only after it processes the msg(if the processing goes
wrong, msg will be read again, make sure your processing is idempotent- should not impact system)
-> Exactly once - can be achieved for Kafka to Kafka workflows

Kafka Broker Discovery

-> Every Kafka broker is a "bootstrap server"
-> Once u connect to the broker u will be connected to the entire cluster
-> When Kafka client connects to a broker -> broker send backs the metadata (list of broker, topic, partitions)
and Kafka client is intelligent enough now to connect to the desired broker

Zookeeper:
-> It manages brokers and helps in performing leader elections for partitions
-> Sends notifications to Kafka in case of changes(new broker, topic, topic delete broker delete)
-> Kafka can not work without a zookeeper
    -> zookeeper is designed to operate with an odd number of servers
    -> zookeeper is a leader(handles writes) and the rest of the servers are followers(handle reads)
    -> Zookeeper does not store consumer offsets with Kafka > v0.10
-> Kafka uses a zookeeper to manage its metadata

Kafka Guarantees:
-> msgs are appended in the order they are sent

-> consumer read msgs in the order stored in partition
-> With Replication factor N , Kafka can tolerate up to N-1 brokers being down, that's why replication
factor 3 is a good idea.
-> As long as the number of partitions remains constant for a topic, the same key will always go to the same partition

Kafka installtion on windows & start:
-> create data folder under Kafka and create two subfolder that one for Kafka and one for zookeeper
-> refer zookeeper folder path in zookeeper.properties under config
-> refer Kafka folder path in server/properties under config
-> start zookeeper first (zookeeper-server-start.bat zookeeper.properties)
-> start Kafka then (Kafka-server-start.bat server.properties)
Kafka CLI commands with explanation
-> Create kafka topic (kafka-topics --zookeeper 127.0.0.1:2181 --topic first_topic --create --partitions 3
--replication-factor 1)
-> List kafka topics (kafka-topics --zookeeper 127.0.0.1:2181 --list)
-> kafka-console-producer --broker-list 127.0.0.1:9202 --topic first_topic
-> kafka-console-consumer --bootstrap-server 127.0.0.1:9202 --topic first_topic --group my-first-application
--> kafka-consumer-groups --bootstrap-server 127.0.0.1:9202 --group my-first-application --reset-offsets
earliest --execute --topic first_topic
--> kafka-consumer-groups --bootstrap-server 127.0.0.1:9202 --group my-first-application --describe