INTRODUCTION TO KAFKA - NOTES

linkedin.com/vaishnavi-muralidhar

KAFKA

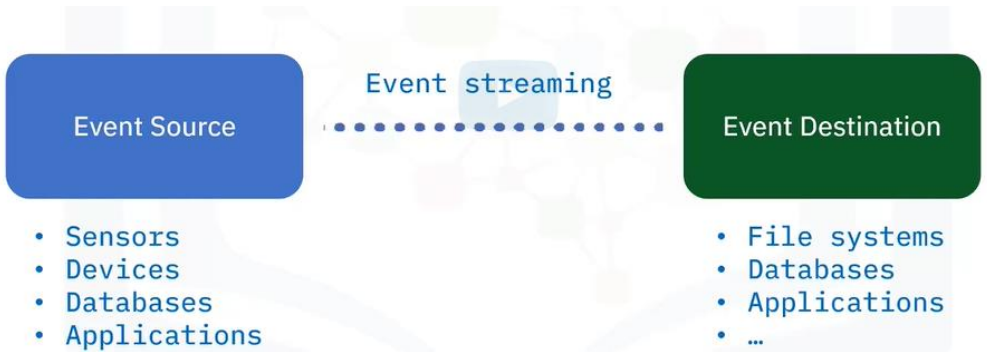
Monday, June 19, 2023
9:51 AM

EVERYTHING YOU NEED TO KNOW ABOUT KAFKA:

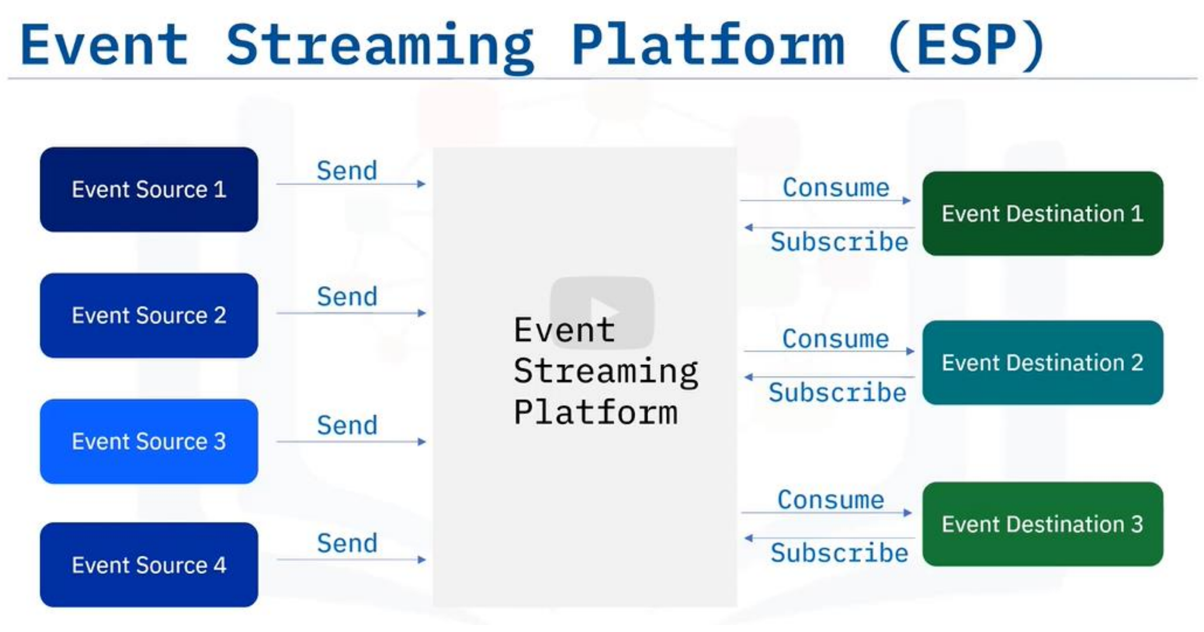
Apache Kafka is a very popular open-source event streaming pipeline.
Kafka Streams API is a client library supporting you with data processing in event streaming pipelines.

Event: Event is an entity with observable updates over time. Ex. Coordinates of moving car, wind speed at a place etc

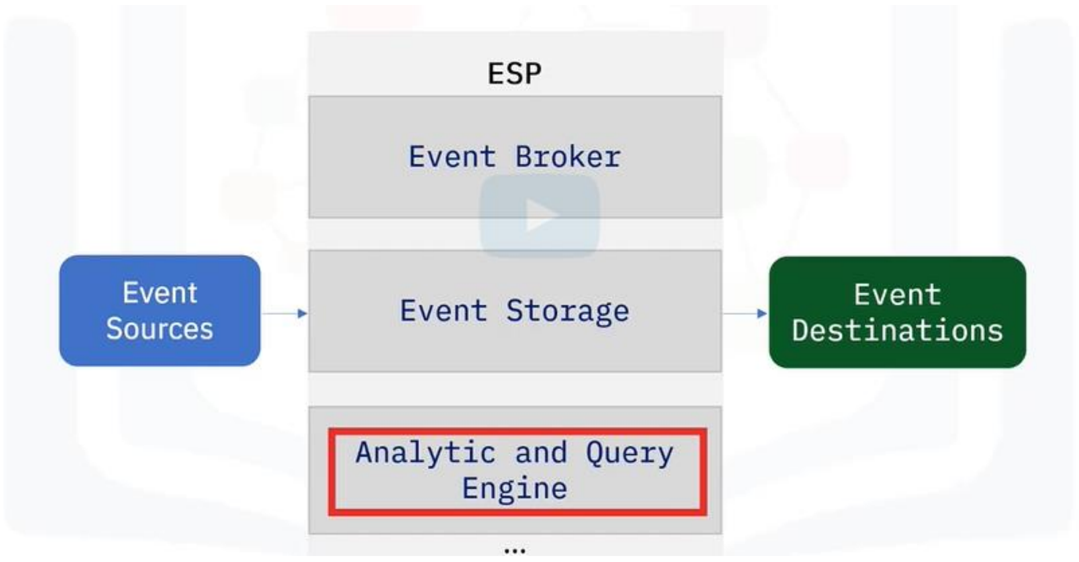
Event streaming: Continuous flow of events in near real time from source to destination.



Event Stream platform:
==> ESP acts as a middle man to handle the source & destination for the real time events that were generated by providing a platform.
==>Destination: It can subscribe to a particular event source so that it can consume data only from the source.

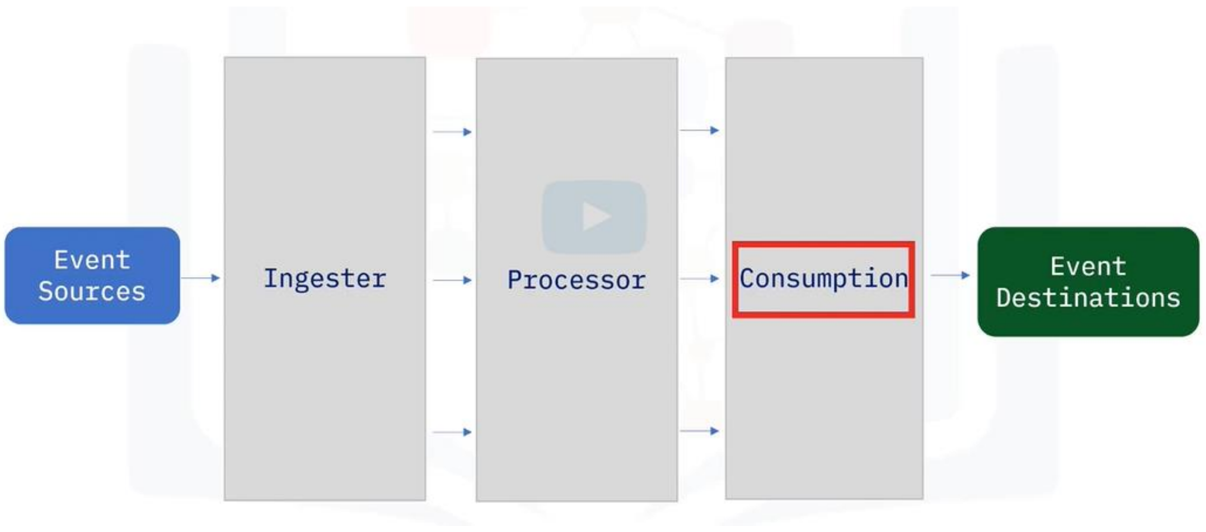


Components of ESP:
==>Event broker: Used to receive events from the event source
Event storage: Used the received events which can later be consumed by the destination
Analytics & query engine: used to analyze/query the stored events



Event Broker:

Ingester: Ingests the events from various sources
Processes: Processes the ingested data like serializing/de-serializing, compression/de-compression, encryption/de-encryption
Consumption: efficiently distributes the events consumed to their correct destinations.

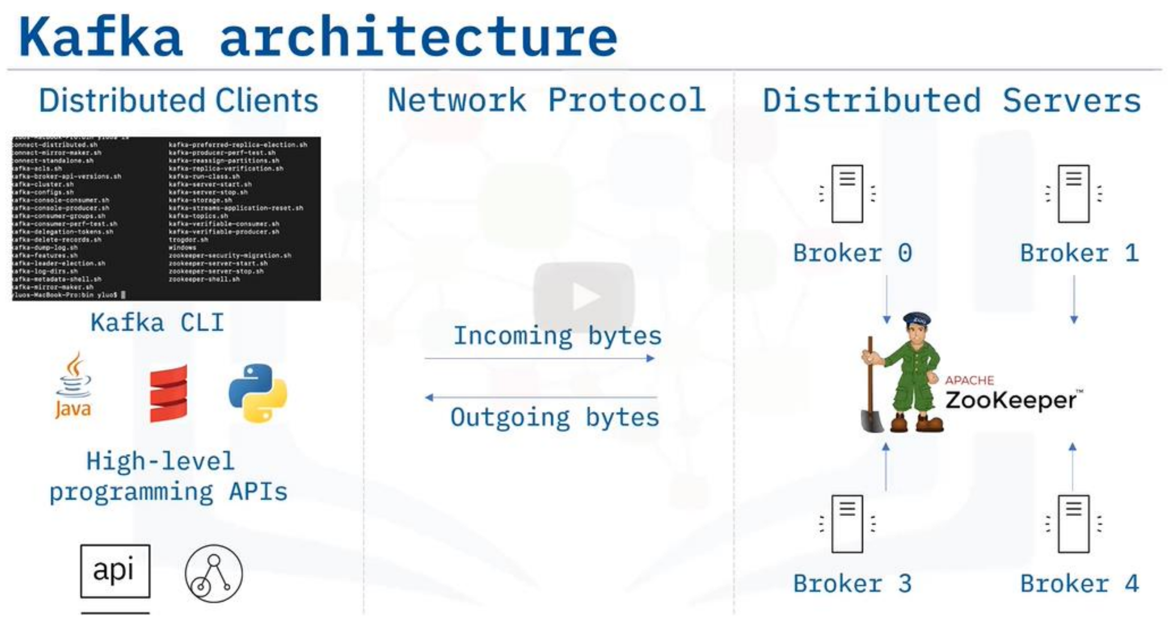


Popular ESP:

Kafka Architecture:

Distributed servers: They serve the purpose of receiving and storing the events from the sources.
Zoo keeper: To manage all the brokers and ensures they work in efficient way.
==> ZooKeeper is responsible for the overall management of Kafka cluster. It monitors the Kafka brokers and notifies Kafka if any broker or partition goes down, or if a new broker or partition goes up.

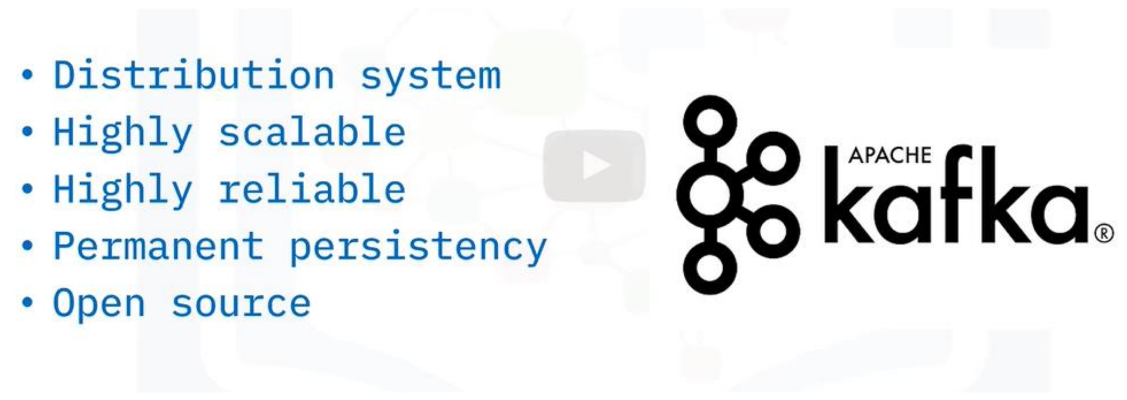
Network protocol: TCP (transmission control protocol) is used to facilitate sending & receiving of events.
Distributed clients: Facilitates to communicate with the brokers.



```
connect-distributed.sh               kafka-preferred-replica-election.sh
connect-mirror-maker.sh              kafka-producer-perf-test.sh
connect-standalone.sh                kafka-reassign-partitions.sh
kafka-acls.sh                        kafka-replica-verification.sh
kafka-broker-api-versions.sh         kafka-run-class.sh
kafka-cluster.sh                     kafka-server-start.sh
kafka-configs.sh                     kafka-server-stop.sh
kafka-console-consumer.sh            kafka-storage.sh
kafka-console-producer.sh            kafka-streams-application-reset.sh
kafka-consumer-groups.sh             kafka-topics.sh
kafka-consumer-perf-test.sh          kafka-verifiable-consumer.sh
kafka-delegation-tokens.sh           kafka-verifiable-producer.sh
kafka-delete-records.sh              trogdor.sh
kafka-dump-log.sh                    windows
kafka-features.sh                    zookeeper-security-migration.sh
kafka-leader-election.sh             zookeeper-server-start.sh
kafka-log-dirs.sh                    zookeeper-server-stop.sh
kafka-metadata-shell.sh              zookeeper-shell.sh
kafka-mirror-maker.sh
yluos-MacBook-Pro:bin yluo$
```

Features of kafka:

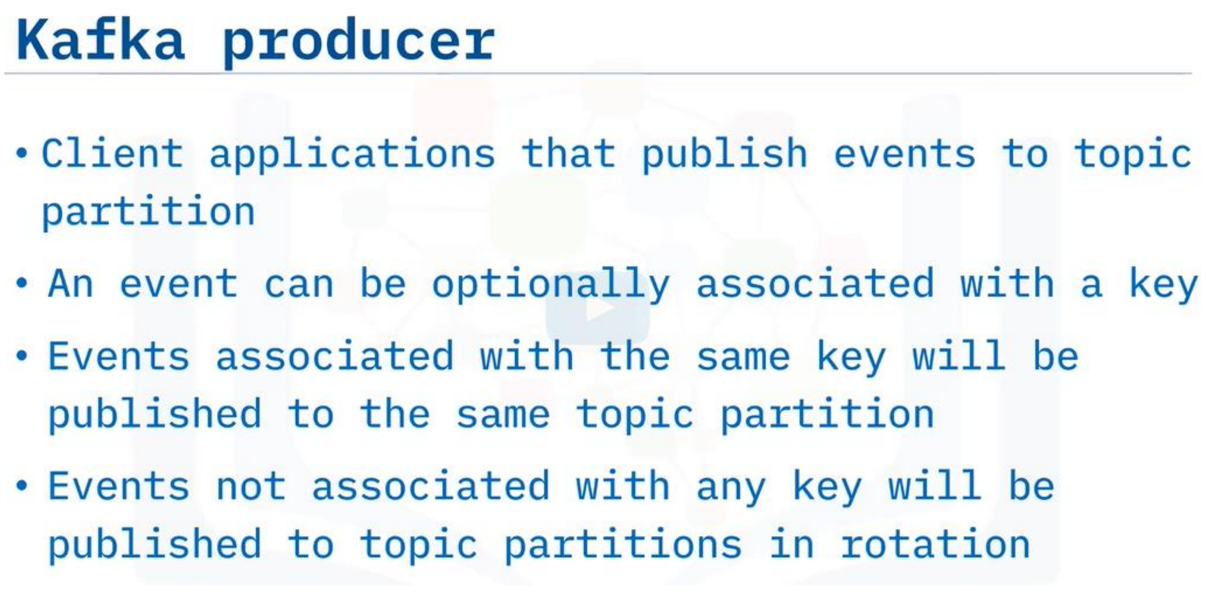
==>Distributed system: It can process parallelly and works in distributed manner.
==>Highly scalable : It can handle huge amount of data by scaling up on the go.
==>Highly reliable: Stores the data in partition and replicates the data to be fault tolerant.
==>Permanent persistency: It will store the data permanently (long time) and can be consumed whenever destination is ready.
==>Open source: it is a open source software and can be customized to ur needs.



- Distribution system
- Highly scalable
- Highly reliable
- Permanent persistency
- Open source

APACHE kafka®

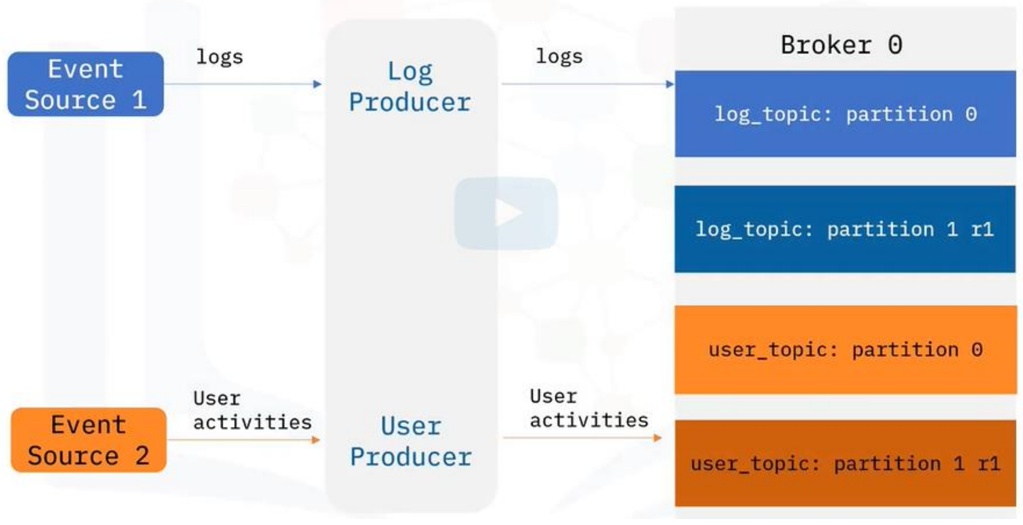
Topic: Database to store specific events like log topic, gps topic, sensor topic etc


Kafka producer

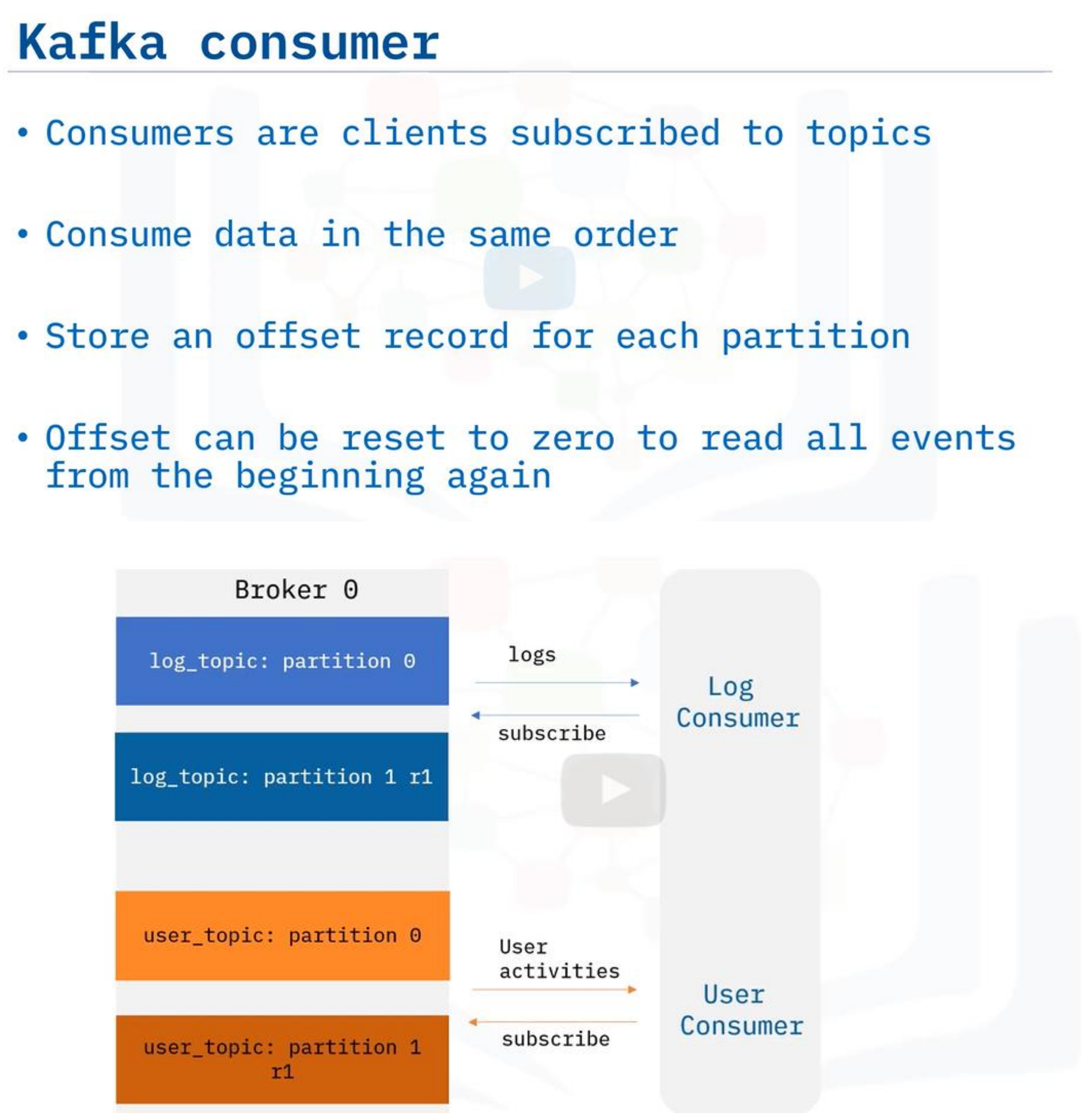
- Client applications that publish events to topic partition
- An event can be optionally associated with a key
- Events associated with the same key will be published to the same topic partition
- Events not associated with any key will be published to topic partitions in rotation

Kafka producer:

1. Events will be published to specific event topic.
2. The events will be partitioned and replicated to be spread across different brokers
3. If one of the broker is down then events will be sent to the replicated event topics.
4. Its good to associate every topic with a key



Kafka consumer: It consumes the events from the kafka producer.
==>Data will be read by the kafka consumer based on the events it is subscribed to



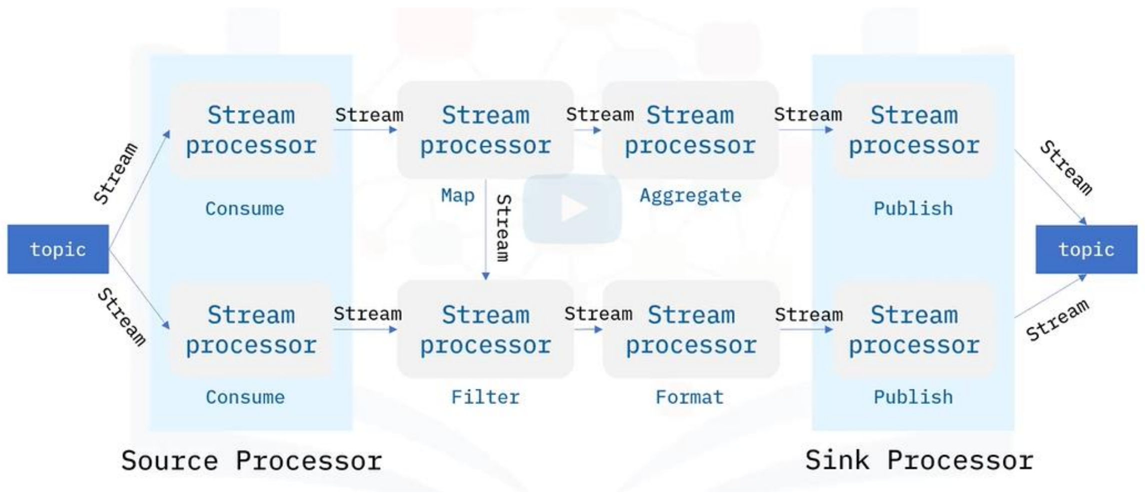
Kafka consumer

- Consumers are clients subscribed to topics
- Consume data in the same order
- Store an offset record for each partition
- Offset can be reset to zero to read all events from the beginning again

Kafka Streams API: Used to facilitate data processing in events streaming .
--> Used to process the data stored in kafka topics.
-->It can process only one record at a time

Stream processing flow:

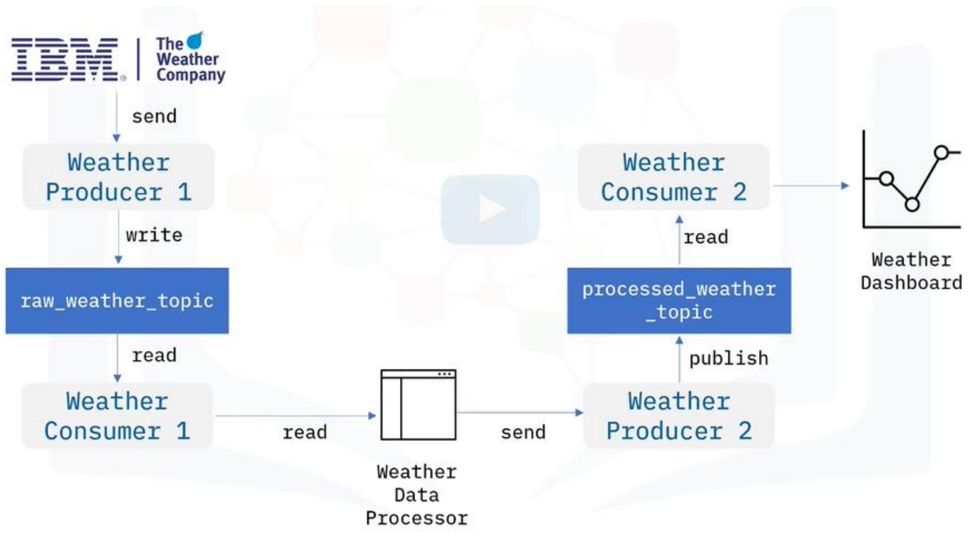
Source processor: It acts like a consumer where it consumes the events from the topic.
Processing: Map, filter,aggregate,fomat (list of processes that can transform the data received from source)
Sink processor: It acts like a producer and sends the final processed data to destination topic



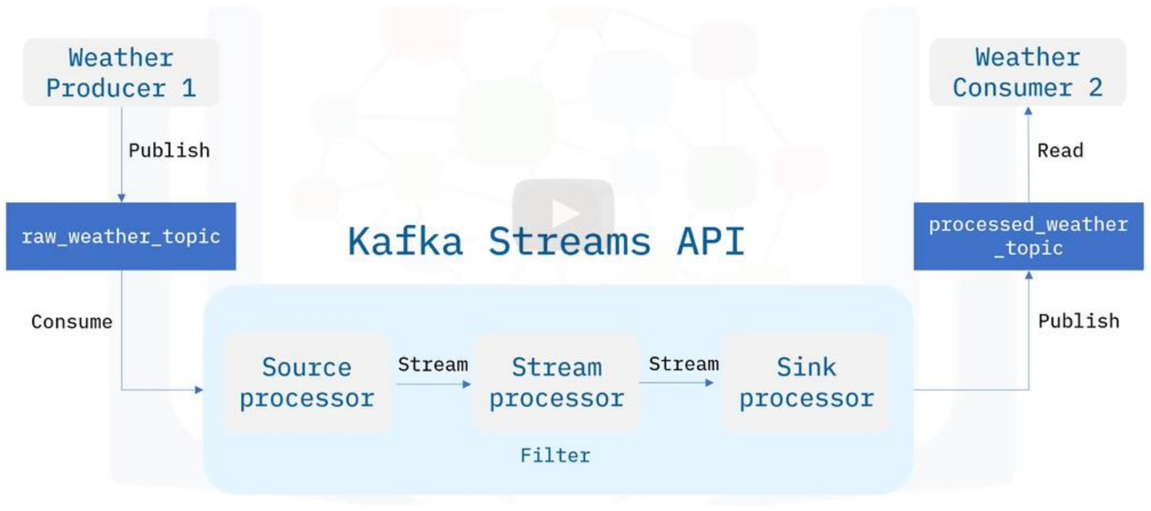
==> with the help of kafka streaming API it is easy to even process the data and store it in a desired format that suits our need.

Simple weather streaming project:

Project Flow diagram:



Using kafka streaming API to filter the highest temperature based on region on a given date time:



Kafka CLI options:

Options for kafka features:

```
Option                        Description
------                        -----------
--bootstrap-server <String: server to  REQUIRED: A comma-separated list of
  connect to>                       host:port pairs to use for
                              establishing the connection to the
                              Kafka cluster.
--command-config [String: command     Property file containing configs to be
  config property file]             passed to Admin Client. This is used
                              with --bootstrap-server option when
                              required.
--describe                       Describe supported and finalized
                              features from a random broker.
--downgrade-all                     Downgrades all finalized features to
                              the maximum version levels known to
                              the tool. This command deletes
                              unknown features from the list of
                              finalized features in the cluster,
                              but it is guaranteed to not add a
                              new feature.
--dry-run                        Performs a dry-run of
                              upgrade/downgrade mutations to
                              finalized feature without applying
                              them.
--help                         Print usage information.
--upgrade-all                     Upgrades all finalized features to the
                              maximum version levels known to the
                              tool. This command finalizes new
                              features known to the tool that were
                              never finalized previously in the
```

```
                                cluster, but it is guaranteed to not
                                delete any existing feature.
--version                          Display Kafka version.
```

Options for kafka producer:

```
Option                                Description
------                                -----------
--batch-size <Integer: size>            Number of messages to send in a single
                                  batch if they are not being sent
                                  synchronously. (default: 200)
--bootstrap-server <String: server to    REQUIRED unless --broker-list
  connect to>                            (deprecated) is specified. The server
                                  (s) to connect to. The broker list
                                  string in the form HOST1:PORT1,HOST2:
                                  PORT2.
--broker-list <String: broker-list>      DEPRECATED, use --bootstrap-server
                                  instead; ignored if --bootstrap-
                                  server is specified.  The broker
                                  list string in the form HOST1:PORT1,
                                  HOST2:PORT2.
--compression-codec [String:             The compression codec: either 'none',
  compression-codec]                     'gzip', 'snappy', 'lz4', or 'zstd'.
                                  If specified without value, then it
                                  defaults to 'gzip'
--help                             Print usage information.
--line-reader <String: reader_class>     The class name of the class to use for
                                  reading lines from standard in. By
                                  default each line is read as a
                                  separate message. (default: kafka.
                                  tools.
                                  ConsoleProducer$LineMessageReader)
--max-block-ms <Long: max block on       The max time that the producer will
  send>                                  block for during a send request
                                  (default: 60000)
--max-memory-bytes <Long: total memory   The total memory used by the producer
  in bytes>                              to buffer records waiting to be sent
                                  to the server. (default: 33554432)
--max-partition-memory-bytes <Long:      The buffer size allocated for a
  memory in bytes per partition>         partition. When records are received
                                  which are smaller than this size the
                                  producer will attempt to
                                  optimistically group them together
                                  until this size is reached.
                                  (default: 16384)
--message-send-max-retries <Integer>     Brokers can fail receiving the message
                                  for multiple reasons, and being
```

```
                                   unavailable transiently is just one
                                   of them. This property specifies the
                                   number of retries before the
                                   producer give up and drop this
                                   message. (default: 3)
--metadata-expiry-ms <Long: metadata     The period of time in milliseconds
  expiration interval>                   after which we force a refresh of
                                   metadata even if we haven't seen any
                                   leadership changes. (default: 300000)
--producer-property <String:             A mechanism to pass user-defined
  producer_prop>                         properties in the form key=value to
                                   the producer.
--producer.config <String: config file>  Producer config properties file. Note
                                   that [producer-property] takes
                                   precedence over this config.
--property <String: prop>                A mechanism to pass user-defined
                                   properties in the form key=value to
                                   the message reader. This allows
                                   custom configuration for a user-
                                   defined message reader. Default
                                   properties include:
                                       parse.key=true|false
                                       key.separator=<key.separator>
                                       ignore.error=true|false
--request-required-acks <String:         The required acks of the producer
  request required acks>                 requests (default: 1)
--request-timeout-ms <Integer: request   The ack timeout of the producer
  timeout ms>                            requests. Value must be non-negative
                                   and non-zero (default: 1500)
--retry-backoff-ms <Integer>             Before each retry, the producer
                                   refreshes the metadata of relevant
                                   topics. Since leader election takes
                                   a bit of time, this property
                                   specifies the amount of time that
                                   the producer waits before refreshing
                                   the metadata. (default: 100)
--socket-buffer-size <Integer: size>     The size of the tcp RECV size.
                                   (default: 102400)
--sync                                   If set message send requests to the
                                   brokers are synchronously, one at a
                                   time as they arrive.
--timeout <Integer: timeout_ms>          If set and the producer is running in
                                   asynchronous mode, this gives the
                                   maximum amount of time a message
                                   will queue awaiting sufficient batch
                                   size. The value is given in ms.
                                   (default: 1000)
--topic <String: topic>                  REQUIRED: The topic id to produce
```

```
                                       messages to.
--version                                  Display Kafka version.
```

Options for kafka consumer:

```
Option                                    Description
------                                    -----------
--bootstrap-server <String: server to    REQUIRED: The server(s) to connect to.
  connect to>
--consumer-property <String:             A mechanism to pass user-defined
  consumer_prop>                          properties in the form key=value to
                                          the consumer.
--consumer.config <String: config file>  Consumer config properties file. Note
                                          that [consumer-property] takes
                                          precedence over this config.
--enable-systest-events                  Log lifecycle events of the consumer
                                          in addition to logging consumed
                                          messages. (This is specific for
                                          system tests.)
--formatter <String: class>              The name of a class to use for
                                          formatting kafka messages for
                                          display. (default: kafka.tools.
                                          DefaultMessageFormatter)
--from-beginning                         If the consumer does not already have
                                          an established offset to consume
                                          from, start with the earliest
                                          message present in the log rather
                                          than the latest message.
--group <String: consumer group id>      The consumer group id of the consumer.
--help                                   Print usage information.
--isolation-level <String>               Set to read_committed in order to
                                          filter out transactional messages
                                          which are not committed. Set to
                                          read_uncommitted to read all
                                          messages. (default: read_uncommitted)
--key-deserializer <String:
  deserializer for key>
--max-messages <Integer: num_messages>   The maximum number of messages to
                                          consume before exiting. If not set,
                                          consumption is continual.
--offset <String: consume offset>        The offset id to consume from (a non-
                                          negative number), or 'earliest'
                                          which means from beginning, or
                                          'latest' which means from end
                                          (default: latest)
--partition <Integer: partition>         The partition to consume from.
                                          Consumption starts from the end of
```

```
                                         the partition unless '--offset' is
                                         specified.
--property <String: prop>                The properties to initialize the
                                         message formatter. Default
                                         properties include:
                                        print.timestamp=true|false
                                        print.key=true|false
                                        print.offset=true|false
                                        print.partition=true|false
                                        print.headers=true|false
                                        print.value=true|false
                                        key.separator=<key.separator>
                                        line.separator=<line.separator>
                                        headers.separator=<line.separator>
                                        null.literal=<null.literal>
                                        key.deserializer=<key.deserializer>
                                        value.deserializer=<value.
                                         deserializer>
                                        header.deserializer=<header.
                                         deserializer>
                                       Users can also pass in customized
                                         properties for their formatter; more
                                         specifically, users can pass in
                                         properties keyed with 'key.
                                         deserializer.', 'value.
                                         deserializer.' and 'headers.
                                         deserializer.' prefixes to configure
                                         their deserializers.
--skip-message-on-error                  If there is an error when processing a
                                         message, skip it instead of halt.
--timeout-ms <Integer: timeout_ms>       If specified, exit if no message is
                                         available for consumption for the
                                         specified interval.
--topic <String: topic>                  The topic id to consume on.
--value-deserializer <String:
  deserializer for values>
--version                                Display Kafka version.
--whitelist <String: whitelist>          Regular expression specifying
                                         whitelist of topics to include for
                                         consumption.
```

COMMANDS TO FOLLOW:

==> All the shell scripts are available in the cloud lab. You can go through by

1.cd /home/project/kafka_2.12-2.8.0/bin

2.ls

3.you can view the list of shell scripts available along with the various options available within them

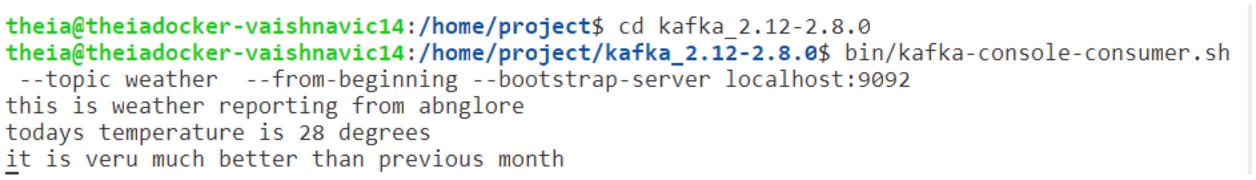
CREATE A TOPIC:
bin/kafka-topics.sh --create --topic news --bootstrap-server localhost:9092 => creates a topic named "news" which runs on the bootstrap server which runs on the localhost.

CREATE PRODUCER
bin/kafka-console-producer.sh --topic news --bootstrap-server localhost:9092

CREATE CONSUMER:
bin/kafka-console-consumer.sh --topic news --from-beginning --bootstrap-server localhost:9092

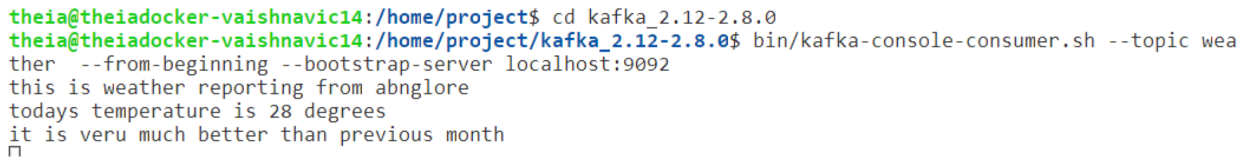
In the cloud labs, you will be able to create a topic and start a producer where you can send the messages as events
Then start a consumer in different terminal where you can receive the messages sent from the producer

Producer:



```
theia@theiadocker-vaishnavic14:/home/project$ cd kafka_2.12-2.8.0
theia@theiadocker-vaishnavic14:/home/project/kafka_2.12-2.8.0$ bin/kafka-console-consumer.sh
 --topic weather  --from-beginning --bootstrap-server localhost:9092
this is weather reporting from abnglore
todays temperature is 28 degrees
it is veru much better than previous month
```

Consumer:



```
theia@theiadocker-vaishnavic14:/home/project$ cd kafka_2.12-2.8.0
theia@theiadocker-vaishnavic14:/home/project/kafka_2.12-2.8.0$ bin/kafka-console-consumer.sh --topic wea
ther  --from-beginning --bootstrap-server localhost:9092
this is weather reporting from abnglore
todays temperature is 28 degrees
it is veru much better than previous month
```