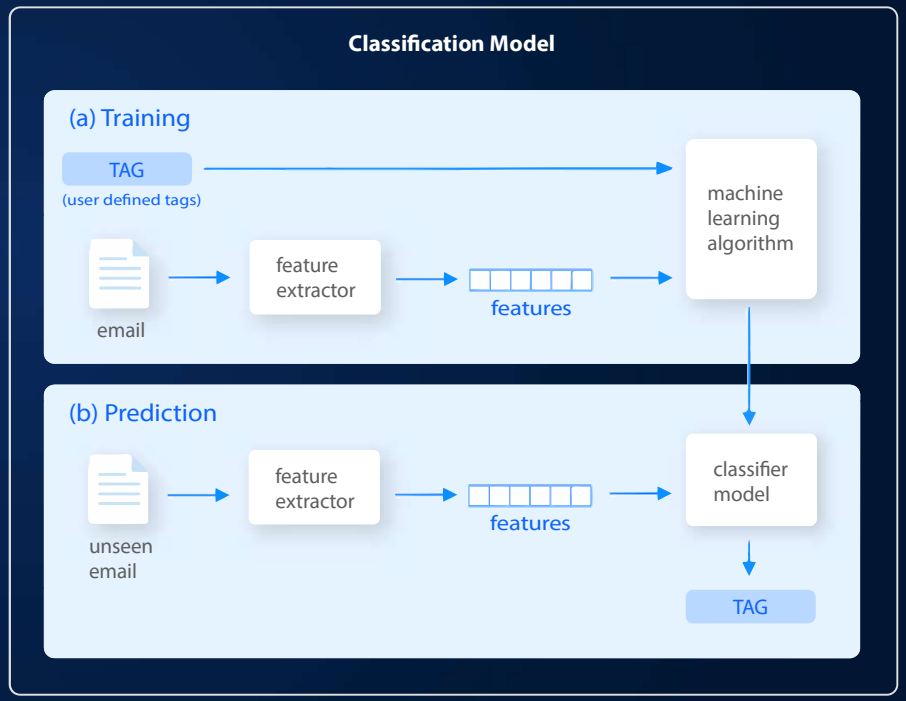
SOFTWARE GROUP PROJECT - GROUP 7B

Intelligent email archive

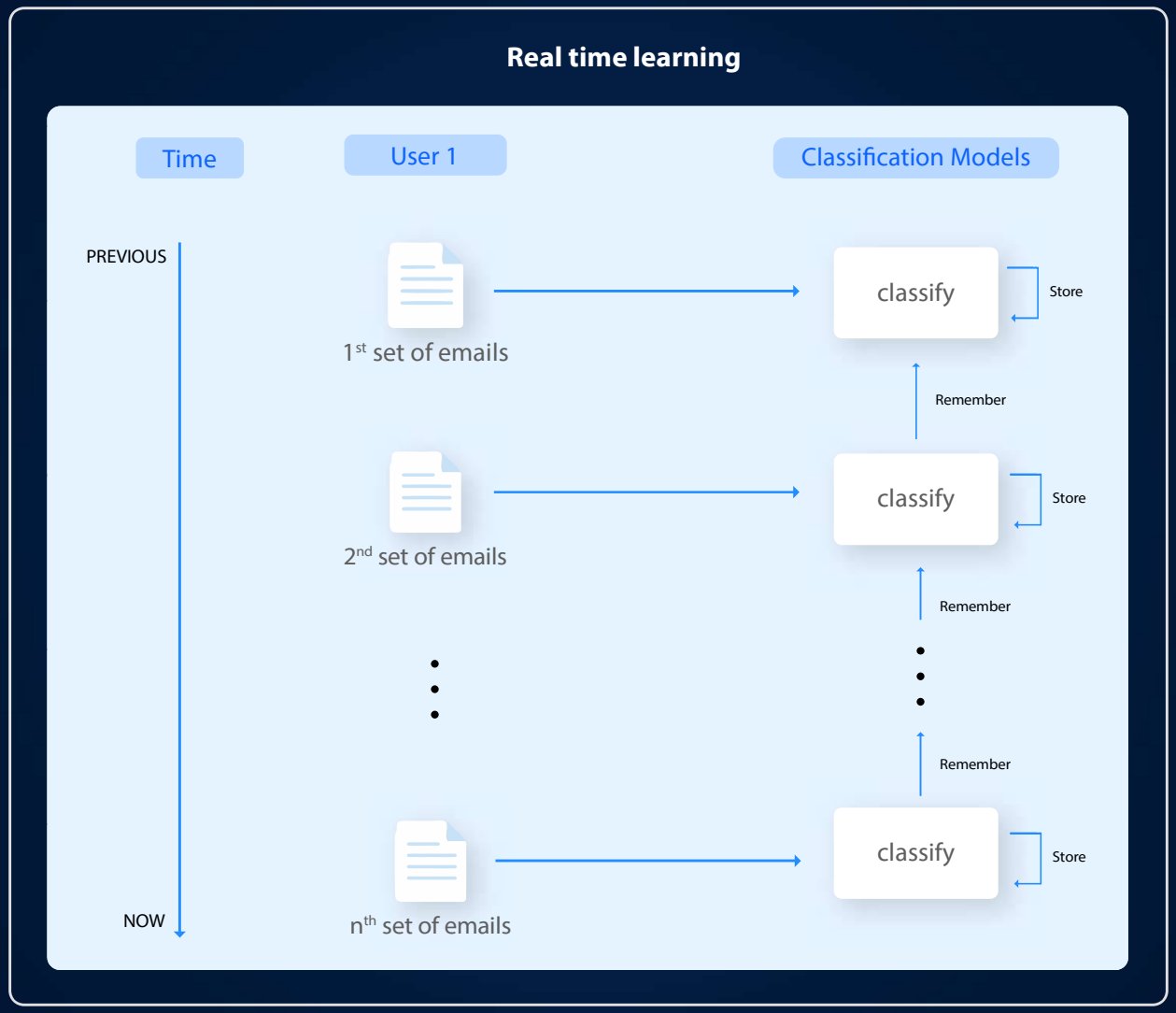
GOALS

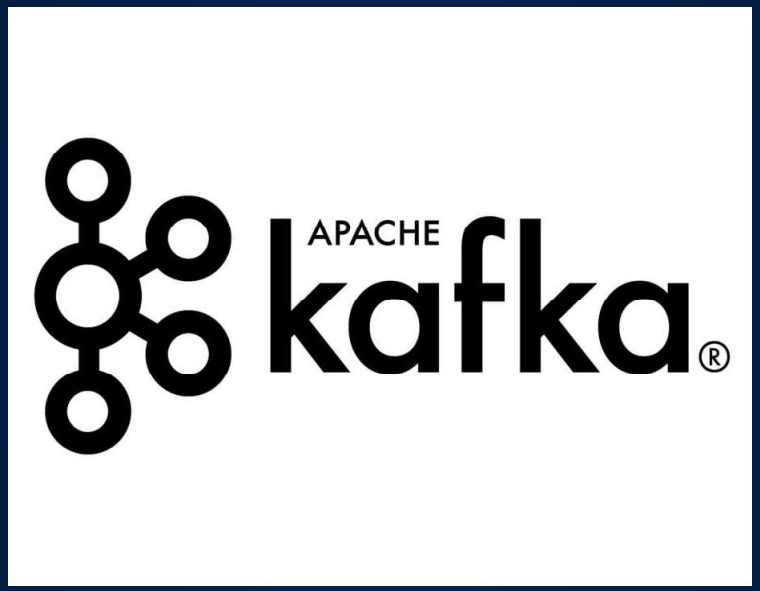
- Intelligent email archive system
- Gmail intergrated
- Use of a machine learning algorithm
- Natural language processing tools
- Search emails
- Saved emails PDF files.



Real time learning

Semi-supervised learning


Real time learning
Time
User 1
Classification Models
PREVIOUS
NOW
1st set of emails
2nd set of emails
nth set of emails
classify
classify
classify
Store
Store
Store
Remember
Remember
Remember


APACHE
kafka®

KAFKA KEY TERMS

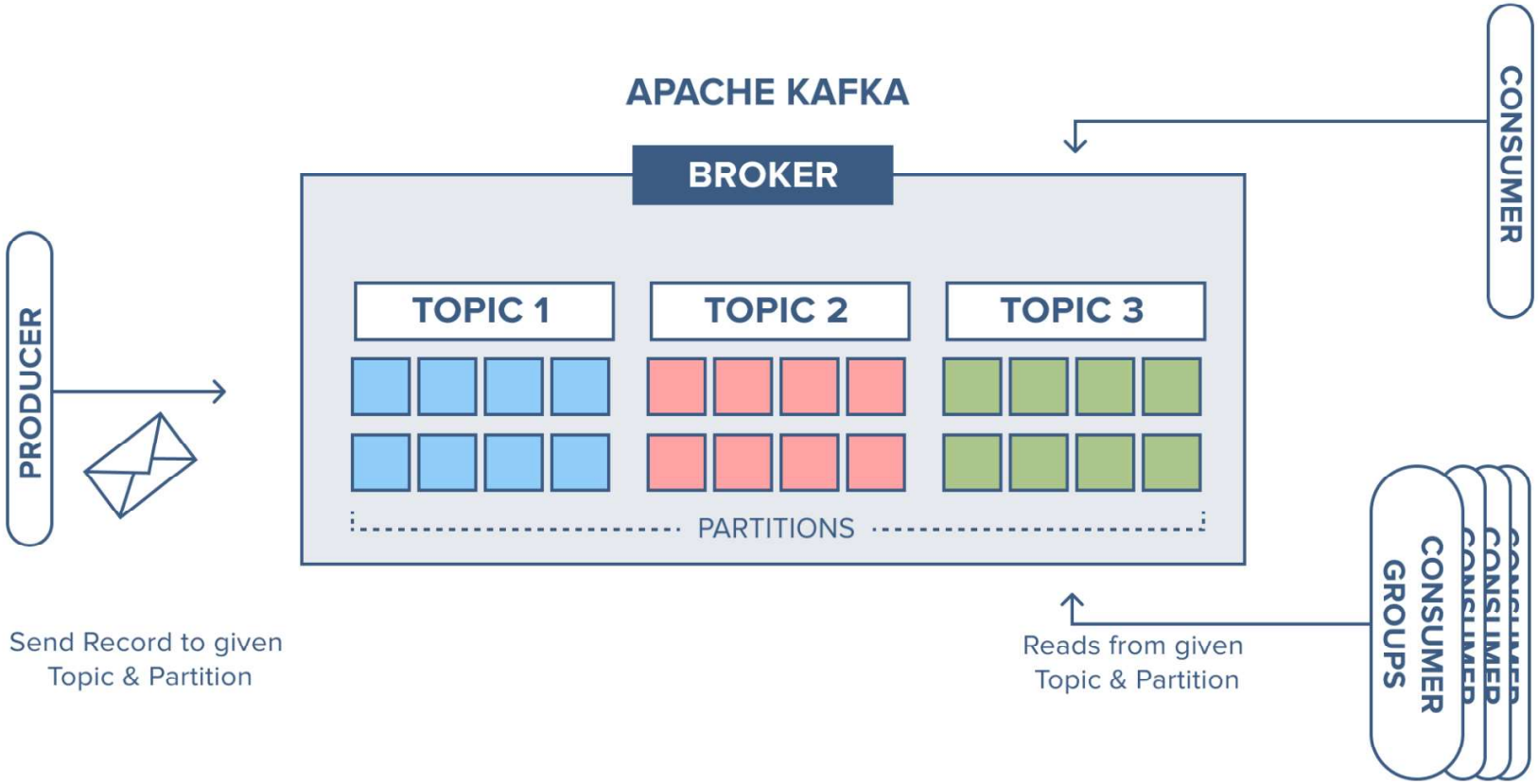
Event records the fact that "something happened" in the world or in your business. Also called record or message in the documentation. Read/write of data to Kafka is done in the form of events.

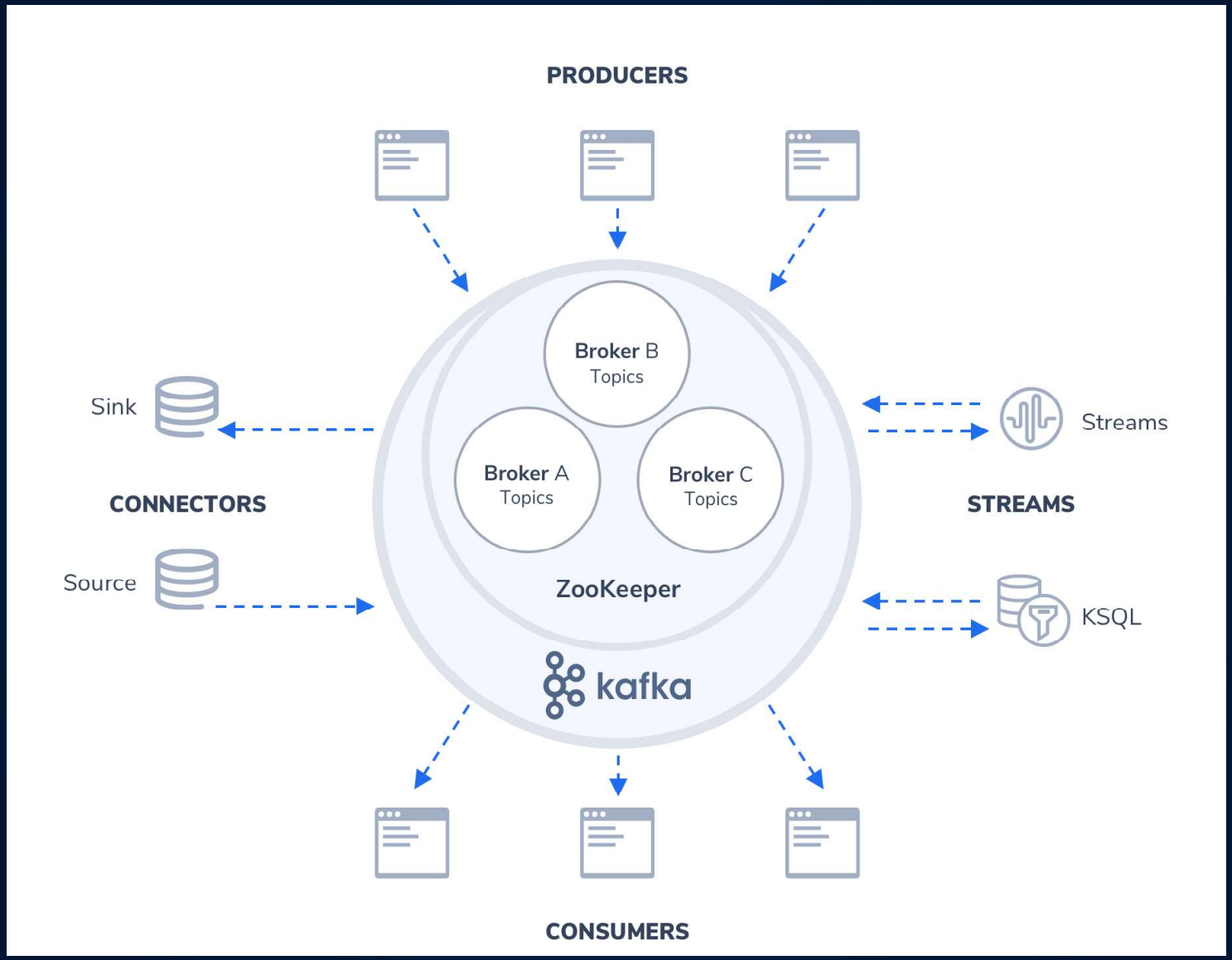
Producers are those client applications that publish (write) events to Kafka.

Consumers are those that subscribe to (read and process) these events.

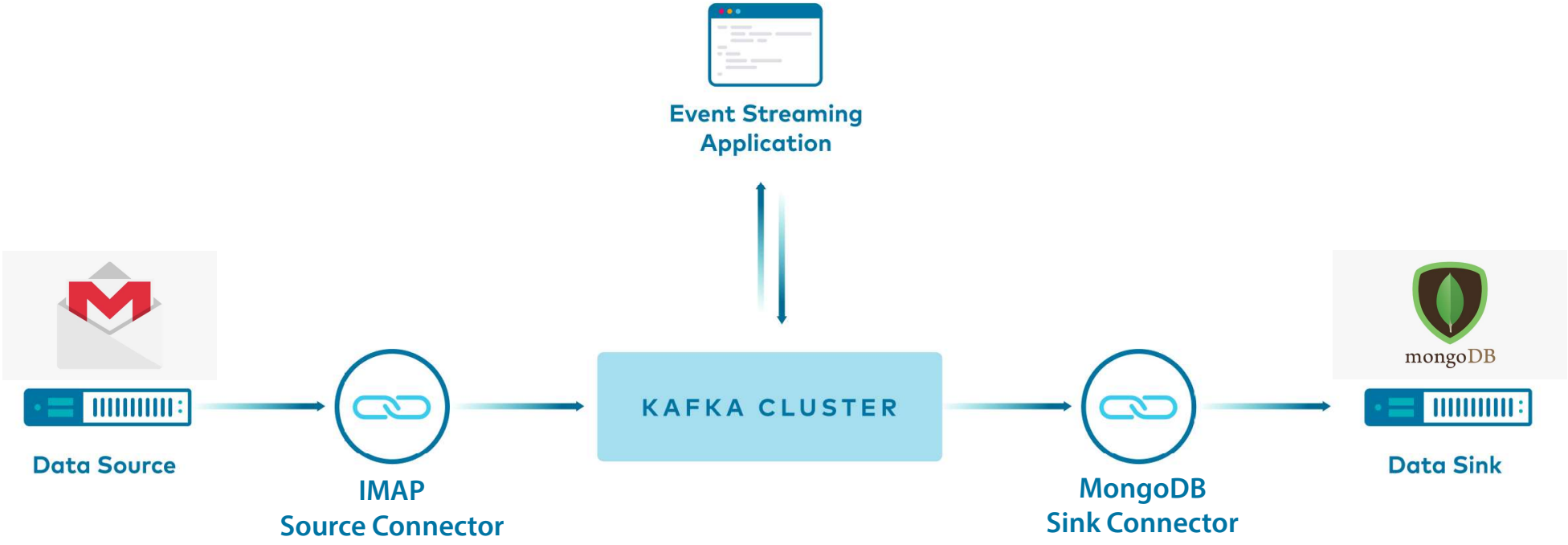
Events are organized and durably stored in **topics**. A topic is similar to a folder in a filesystem, and the events are the files in that folder.

Topics are **partitioned**, meaning a topic is spread over a number of "buckets" located on different Kafka brokers. This distributed placement of your data is very important for scalability because it allows client applications to both read and write the data from/to many brokers at the same time. When a new event is published to a topic, it is actually appended to one of the topic's partitions.

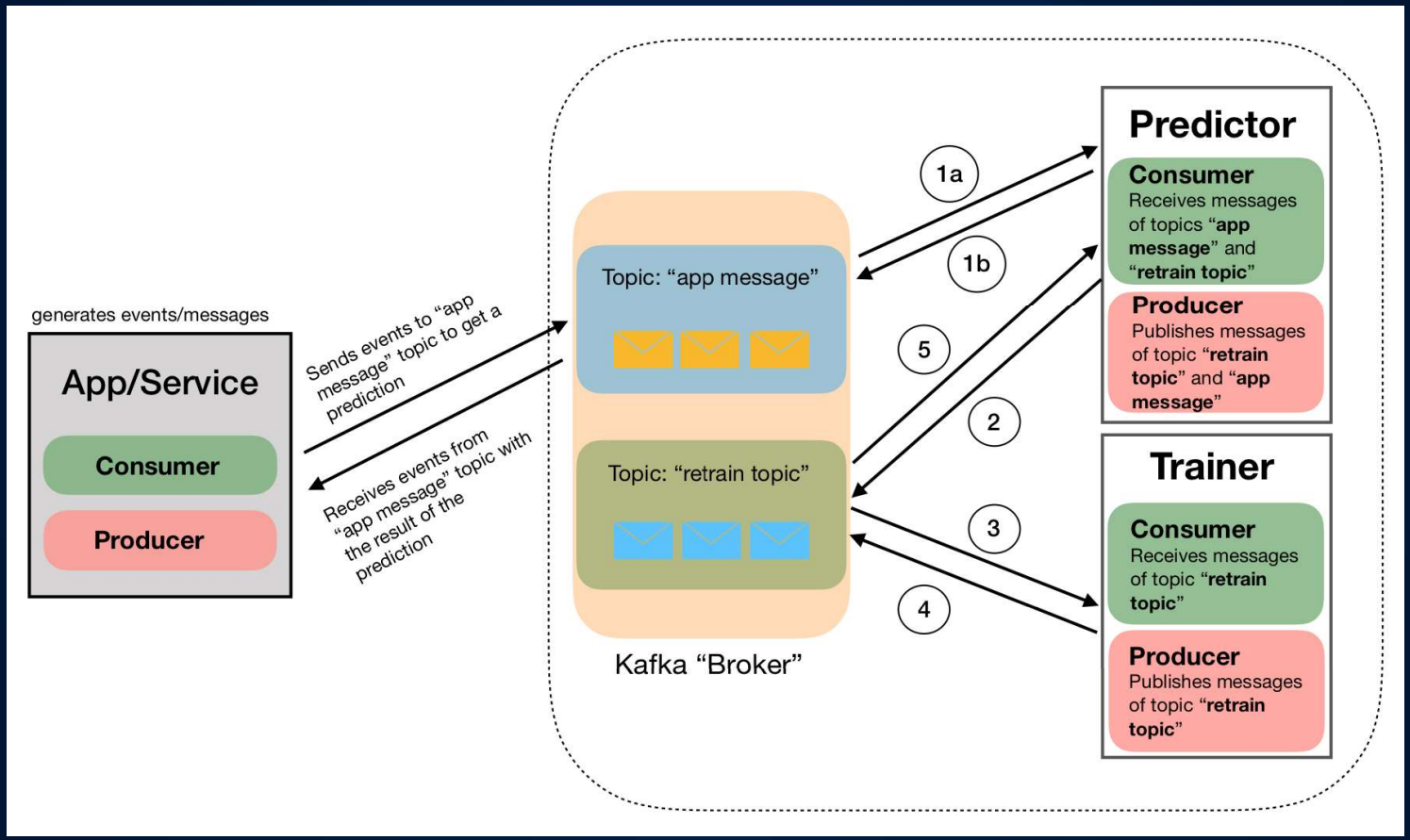

APACHE KAFKA
BROKER
TOPIC 1
TOPIC 2
TOPIC 3
PARTITIONS
PRODUCER
CONSUMER
CONSUMER GROUPS
Send Record to given Topic & Partition
Reads from given Topic & Partition


PRODUCERS
Broker B
Topics
Broker A
Topics
Broker C
Topics
ZooKeeper
kafka
Sink
Source
CONNECTORS
Streams
KSQL
STREAMS
CONSUMERS

OUR PIPELINE



EXAMPLE CONFIGURATION

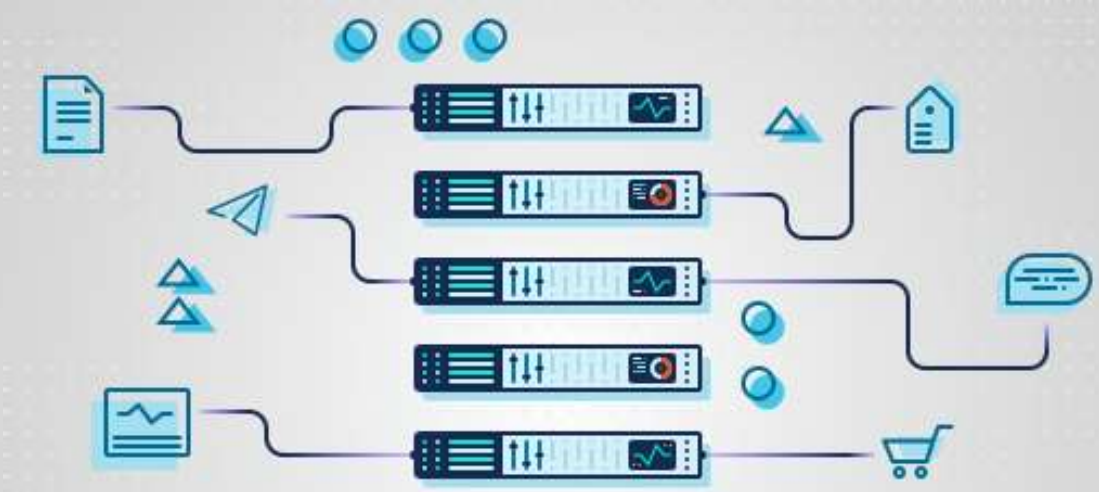


SCENARIO

A company collects data using a series of services that generate events as the users/customers interact with the company's website or app. As these interactions happen, an algorithm needs to run in real time and some immediate action needs to be taken based on the algorithm's outputs (or predictions). On top of that, after N interactions (or observations) the algorithm needs to be retrained without stopping the prediction service, since users will keep interacting.

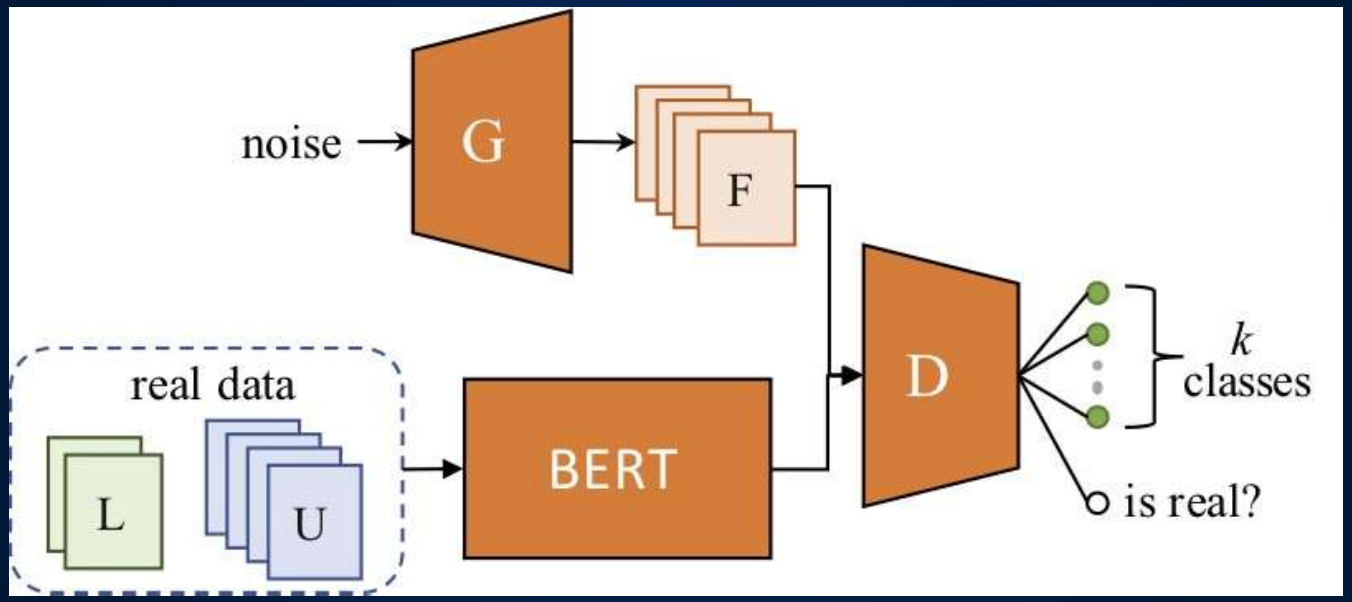
DETAILED EXPLANATION OF SAMPLE PIPELINE

0. The App/Service will send messages into the pipeline. These will be processed and App/Service will then get the results of the predictions.
1. 1a) The messages from App/Service will be published to Kafka and, eventualy, received by the Predictor
   1b) The Predictor will process the data and run the algorithm publishing the message with the prediction result back to Kafka, which will be eventually received by App/Service
2. After N messages the Predictor will publish a "retrain topic" message
3. The Trainer will receive the "retrain topic" message and start retraining the algorithm. In the mean time, the Predictor will not stop serving predictions.
4. Once the algorithm is retrained, the Trainer will publish a message with the corresponding information (namely: *"retraining completed"*)
5. The Predictor will receive the message that retraining is complete, it will load the new model and proceed as usual.

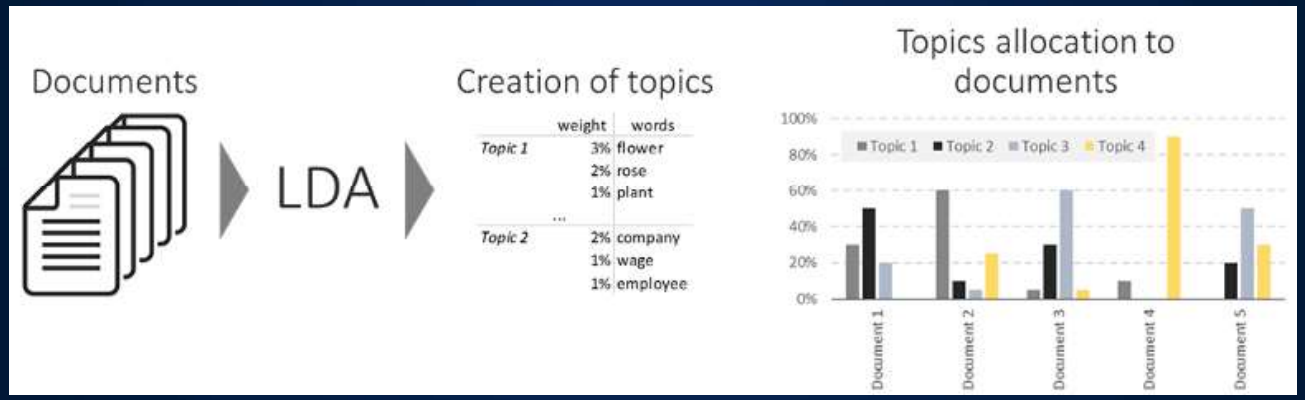


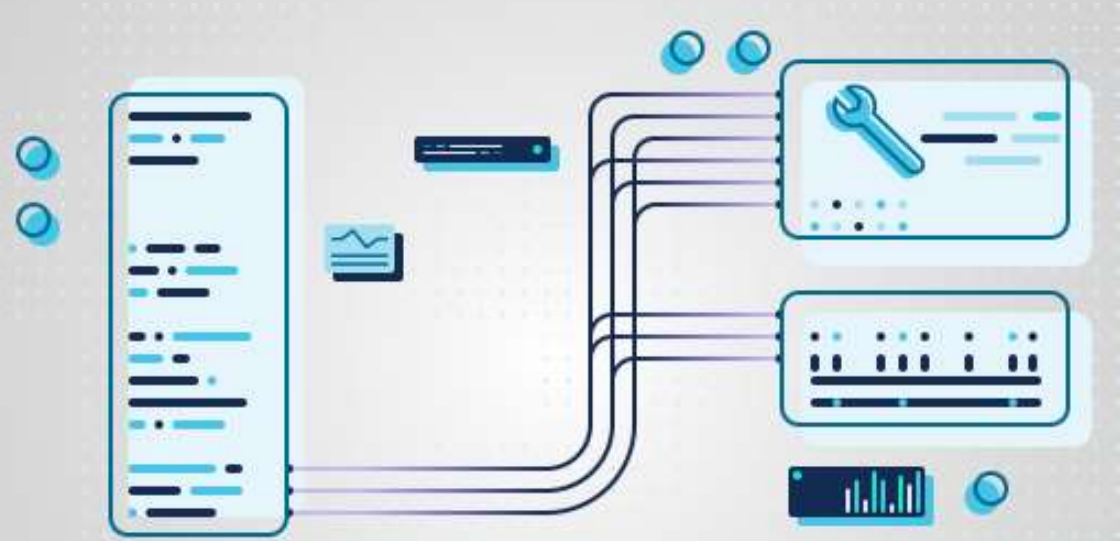
MACHINE LEARNING MODEL

GAN BERT (SEMI SUPERVISED)



LATENT DIRICHLET ALLOCATION (UNSUPERVISED)





FRONT END

DOCKER

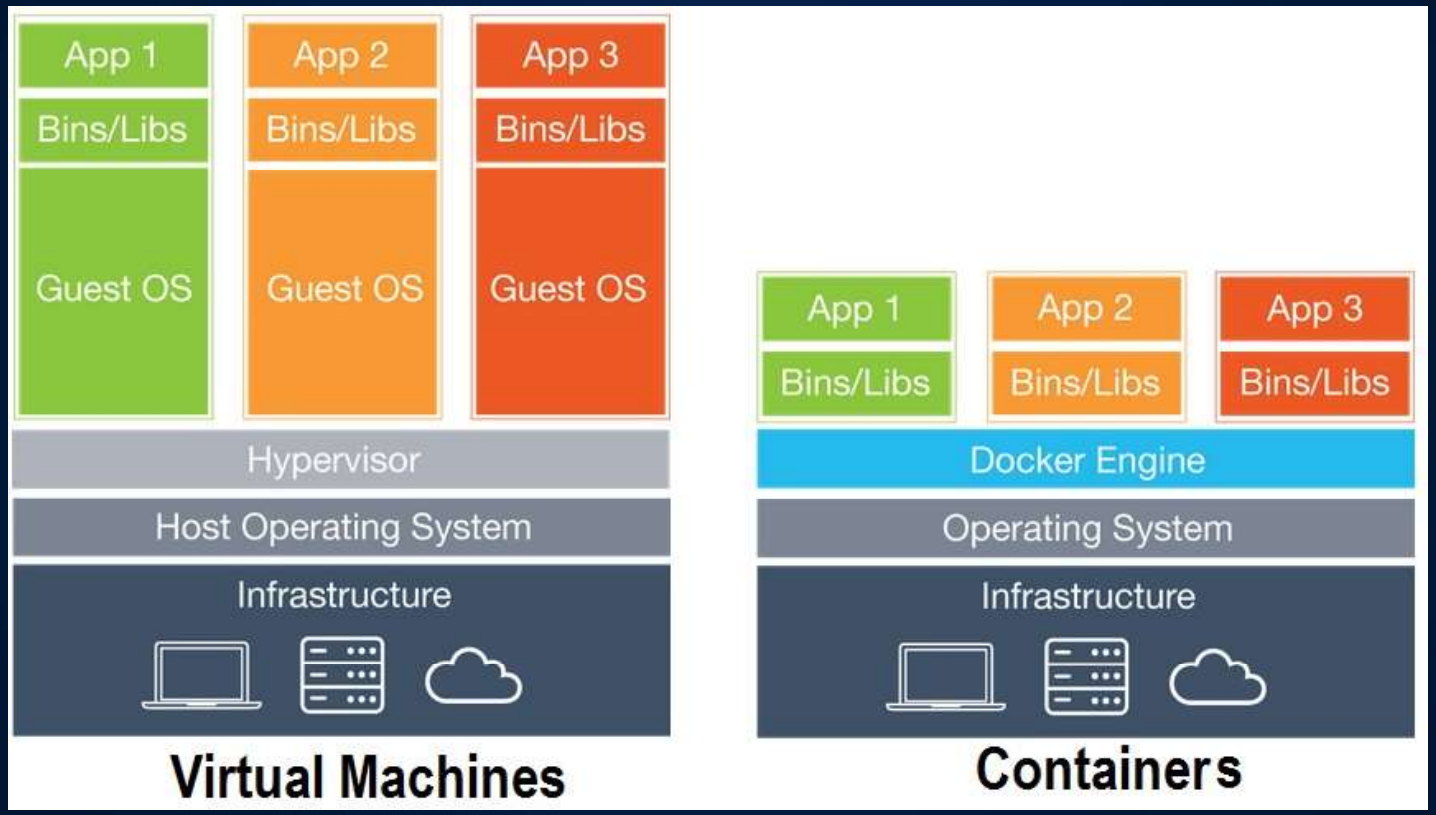
Used for automating the deployment of applications as portable, self-sufficient containers that can run on the cloud or on-premises.

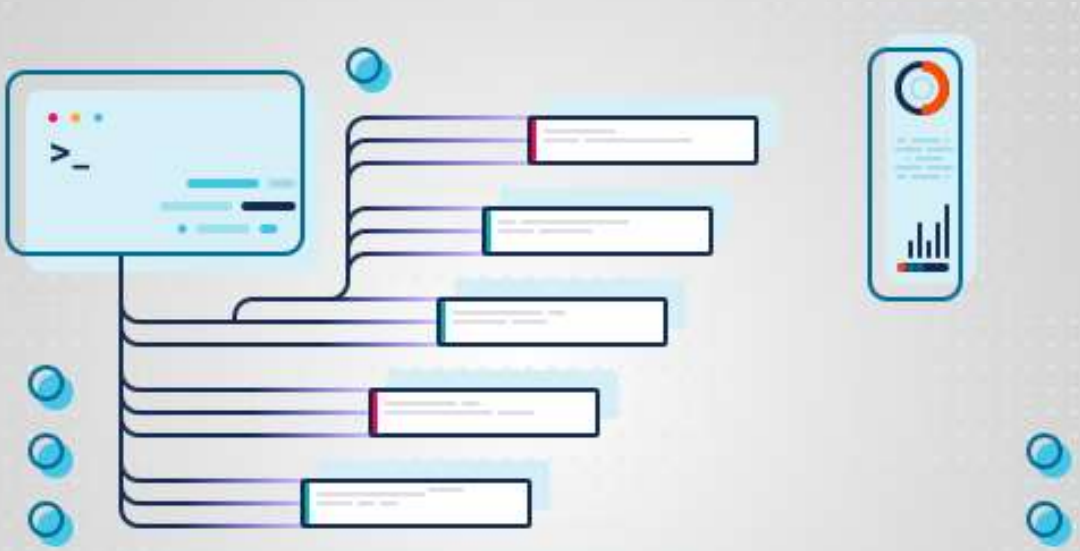
A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

A containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

DOCKER VS VIRTUAL MACHINES





THANK YOU