# Background

The development of IOTs has increased the needs of real-time image data processing. The stream of images comes from various channels such as CCTV. This stream of images is a form of unstructured data that requires high processing power such as GPU to perform real-time image processing, and distributed processing using cluster that made of multiple nodes.

The company has a web application that collects images uploaded by customers and uses Apache Kafka stream to provide streaming of images into the system. Then, the software developers run some algorithms to process the images. The processed images are saved for a minimum of 7 days for archival purposes and the result is written into the database for analysis and statistical reports.

This documentation writes the details of the Section 3: System Design that explains the method to process the images and the technology stack used.

# Methodology

Apache Kafka is a high-throughput distributed messaging system (or streaming platform). It means it supports millions of messages in our modest hardware and scalable; there is no downtime. Apache Kafka allows us to build real-time streaming data pipeline that reliably get data between application. Apache Kafka builds, transforms and reacts to the stream of data. We have our data streams from our web application.

## Kafka Concepts

Apache Kafka runs on a cluster on one or more servers. The Kafka cluster stores streams of records in categories called topics. Each record consists of a key, a value, and a timestamp. This refers to the log of data structure. The database writes the change of events to a log, and derive the value of columns from that log. In Apache Kafka, messages are written to a topic that maintain this log from which subscribers can read and derive their representations of the data. You can think the message as an “activity” log.

## Four Core APIs

There are four core APIs:

* The Producer API allows an application to publish a stream of records to one or more Kafka topics.
* The Consumer API allows an application to subscribe to one or more topics and process the stream of records produced to them.
* The Streams API allows an application to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.
* The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.

**Producer and Consumer**

We can run the Kafka in a single node server (node) or a cluster mode with multiple nodes (Kafka broker). Producers are processes that publish data or a stream of records (push messages) into Kafka topics within the broker. A consumer pulls records off a or more Kafka topic and processes the streams of records produced to them. You can see it as Kafka has publisher, topics and subscribers. Messages are replicated across the cluster to provide support for multiple subscribers and balances the consumers in case of failures.

**Topic**

It is similar to a table in the database without any constraints. It has a name, the same way a table does have a name. You can have as many topics as you want in Kafka, just like as many tables as you want in a database. Topics in Kafka are always multi-subscriber, that means a topic can have zero, one or many consumers (subscribers) that subscribe to the data written to it. Topic can partition topics and enable parallel consumption.

**Broker**

Kafka broker handles all requests from consumers, and keeps data replicated within the cluster. There can be one or more brokers in a cluster. A good number to get started is 3 brokers for a cluster.

**Zookeeper**

Zookeeper keeps the state of the cluster. It is a system that manages the brokers, topics and users. The Zookeeper manages the leader and replicas.

**Apache Spark**

Apache Spark achieves high performance for both batch and streaming data, using a Directed Acyclic Graph (DAG) scheduler, a query optimizer, and a physical execution engine. Apache Spark can run on any cloud providers, and access data in any data sources.

# System Overview

Below is the proposed system overview of the image processing.