Nov 22, 2022

Computer Engineering,
Chulalongkorn University

IoT Healthcare

ADVISOR
Assoc. Prof. Kultida Rojviboonchai

Expert (SnapLogic)
Thanawut Ananpiriyakul
Tanapat Ruengsatra
Kanya Kiatsakdawong

Member

Nitchakran Chaipojjana
6231322621

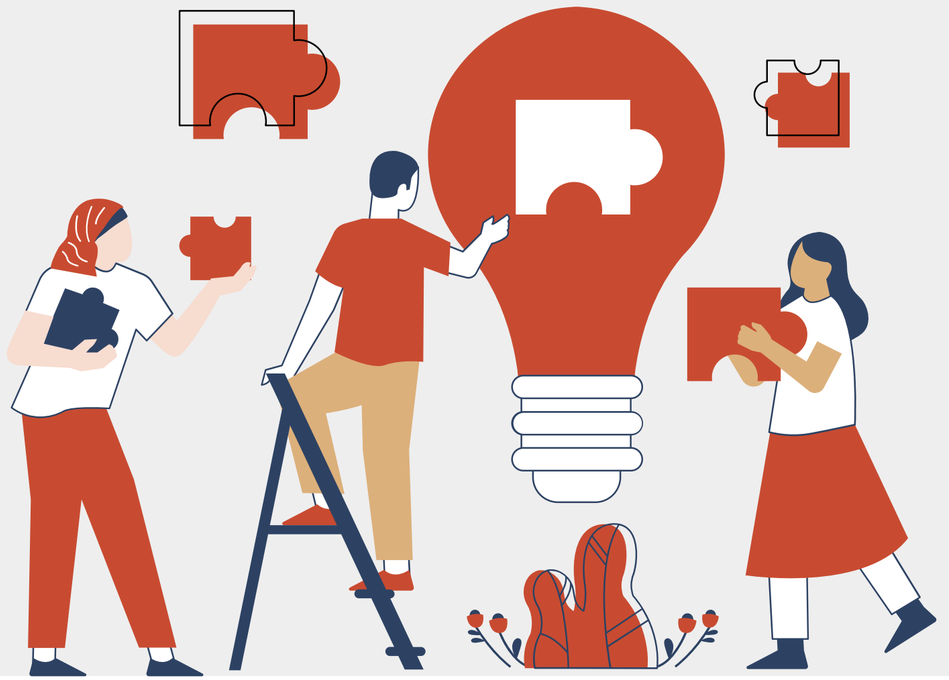
Siwagarn Jitwarodom
6231363321

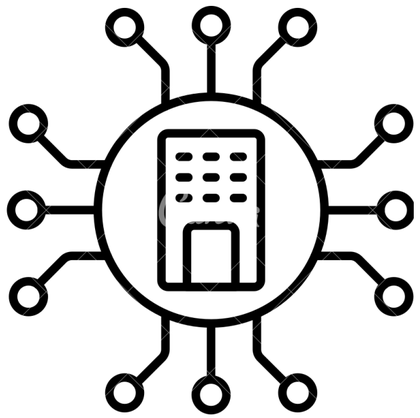
Marineya Tajoparung
6231352421

Nattapat Jaruchaisittikul
6230177821

Table of Contents

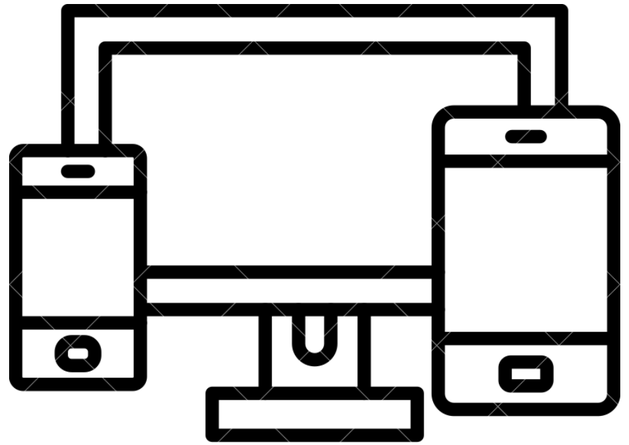
Project Overview



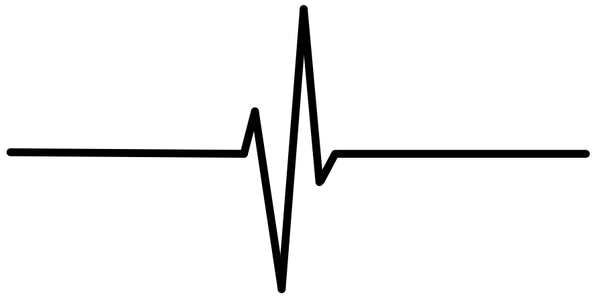


decentralized

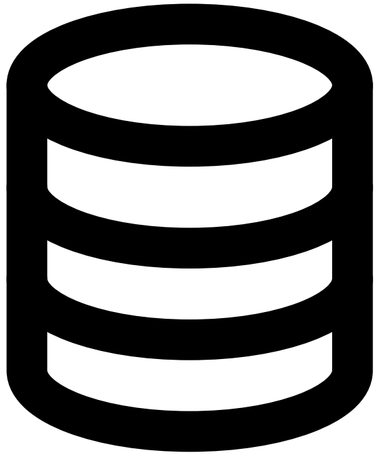
sending format



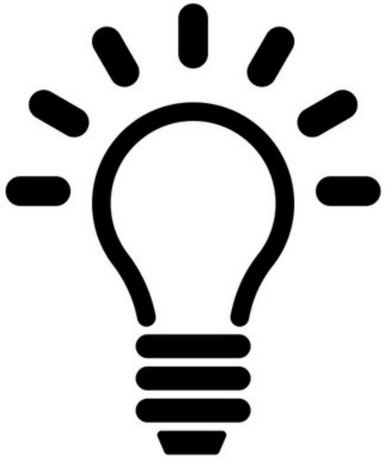
1 model 1 platform



tracking patient



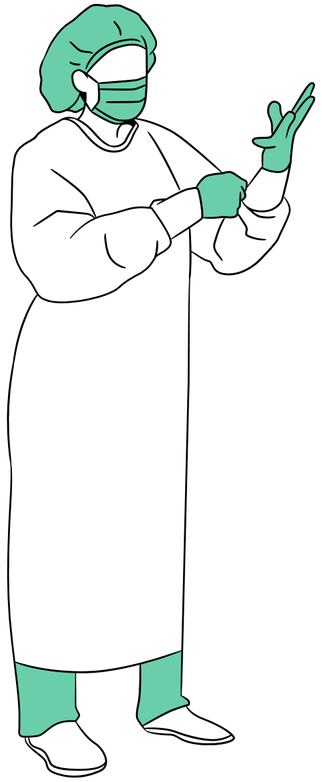
data center



Smart ICU

reduce cost

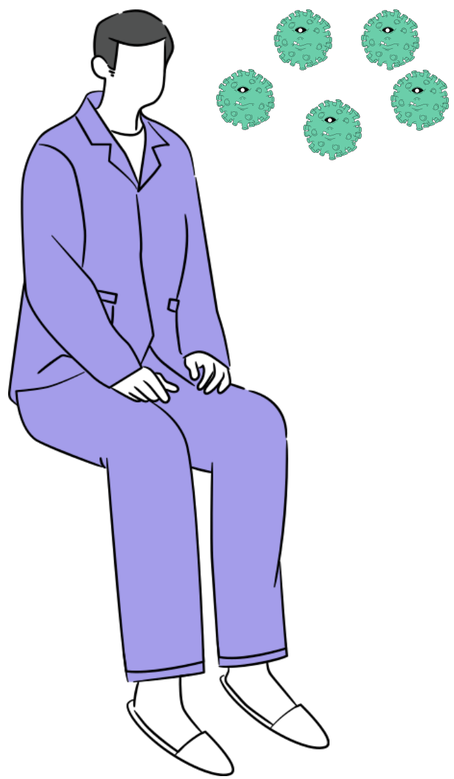
Medical staff



Easy to monitor patient symptom

Efficiently diagnose

Patient

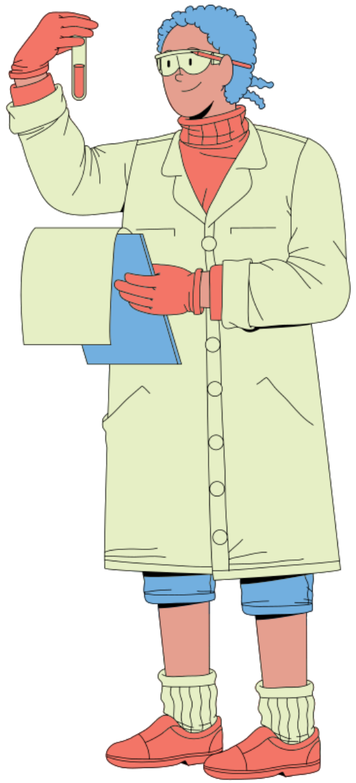


More Safely

More Convenient

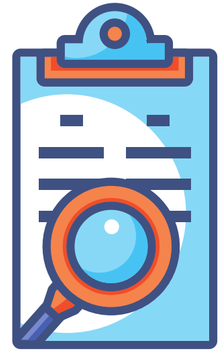
(since have historical data)

Researcher

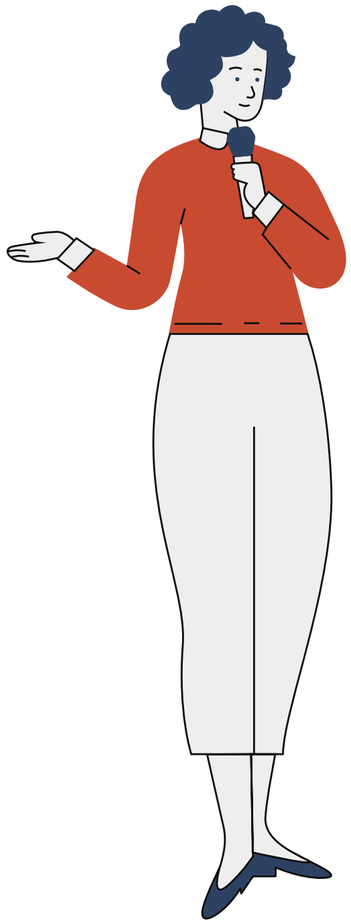


Easy to understand data

Convenient to analyze data



Requirement



Functional requirements

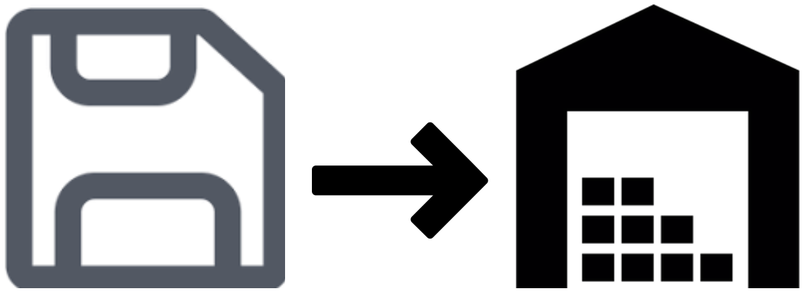
- The system shall be able to receive the input data of IOT from Kafka and MQTT
- the system shall be able to save the received data into datalake pipeline and dataware house
- the system shall provide the received data to other's server according to their permission
- the system shall manage permission for data of other's server
- the system should have a dashboard for user to monitor data

Non-Functional requirements

- The system shall be available 24/7
- the system shall keep patient data confidential
- the system should response within 15 min

Functional Requirements

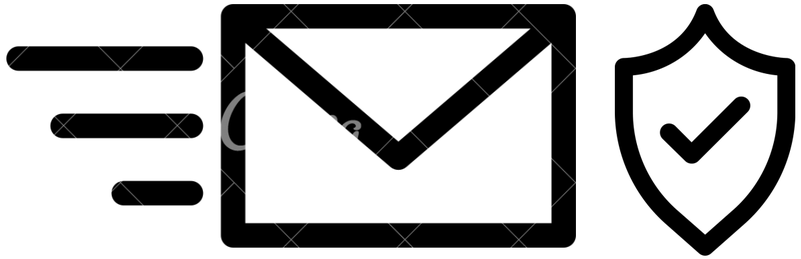
The system shall be able to receive data from kafka and MQTT



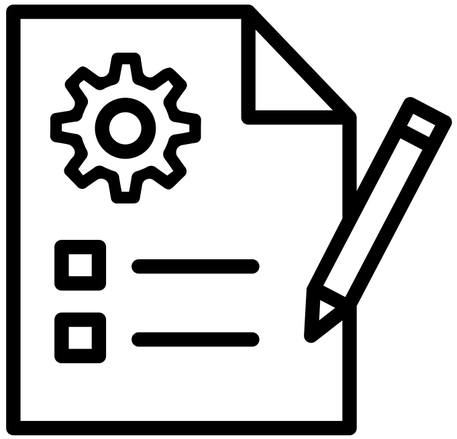
the system shall be able to save the received data to datalake and dataware house

Source: https://commons.wikimedia.org/wiki/Category:Apache_Kafka#/media/File:Apache_kafka.svg

Functional Requirements



the system shall provide the received data to other's server according to their permission

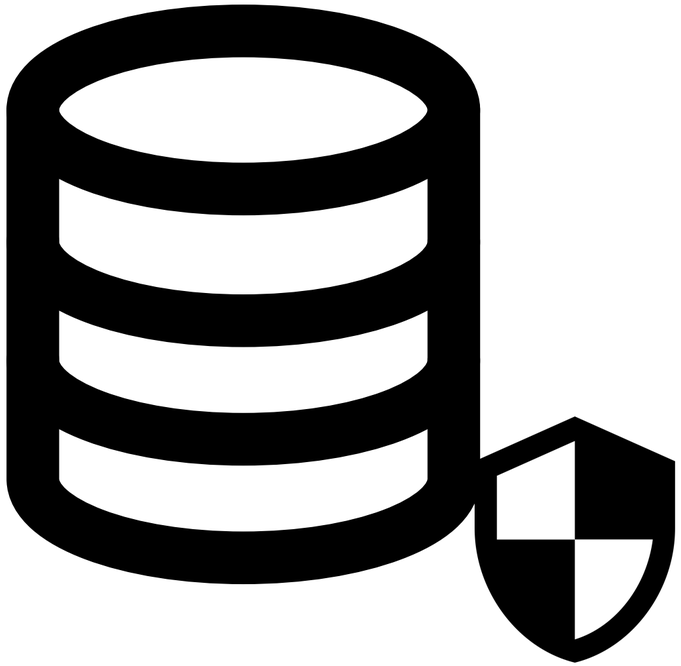


the system shall manage permission for data of other's server

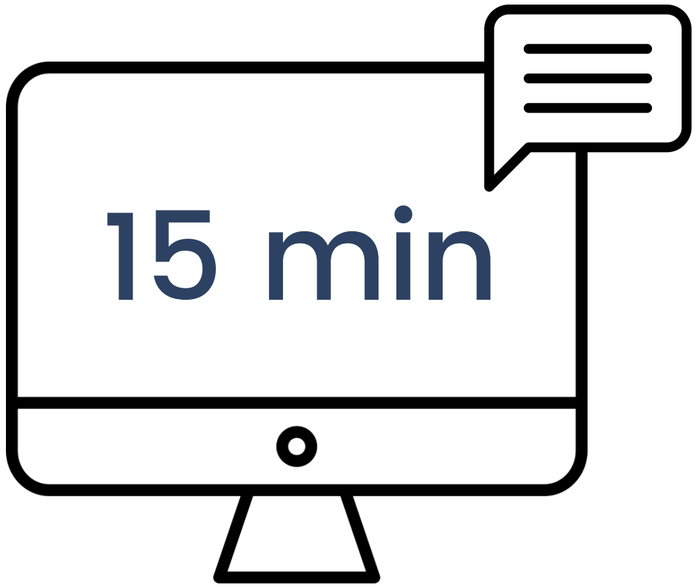
the system should have a dashboard for user to monitor data

Non-Functional Requirements

The system shall be available 24/7



the system shall keep patient data confidential

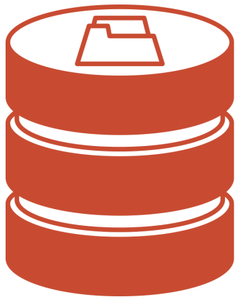


the system should response within 15 min

To-be System

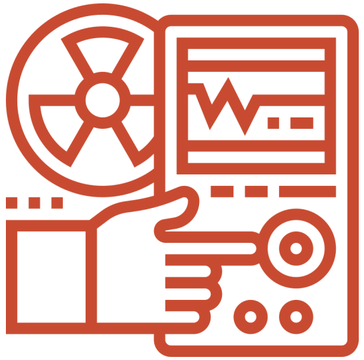
Data recieving system

- MQTT
- Kafka



Data management system

- Database: InfluxDB & MongoDB
- Data Lake & Data Warehouse
- Database Clustering



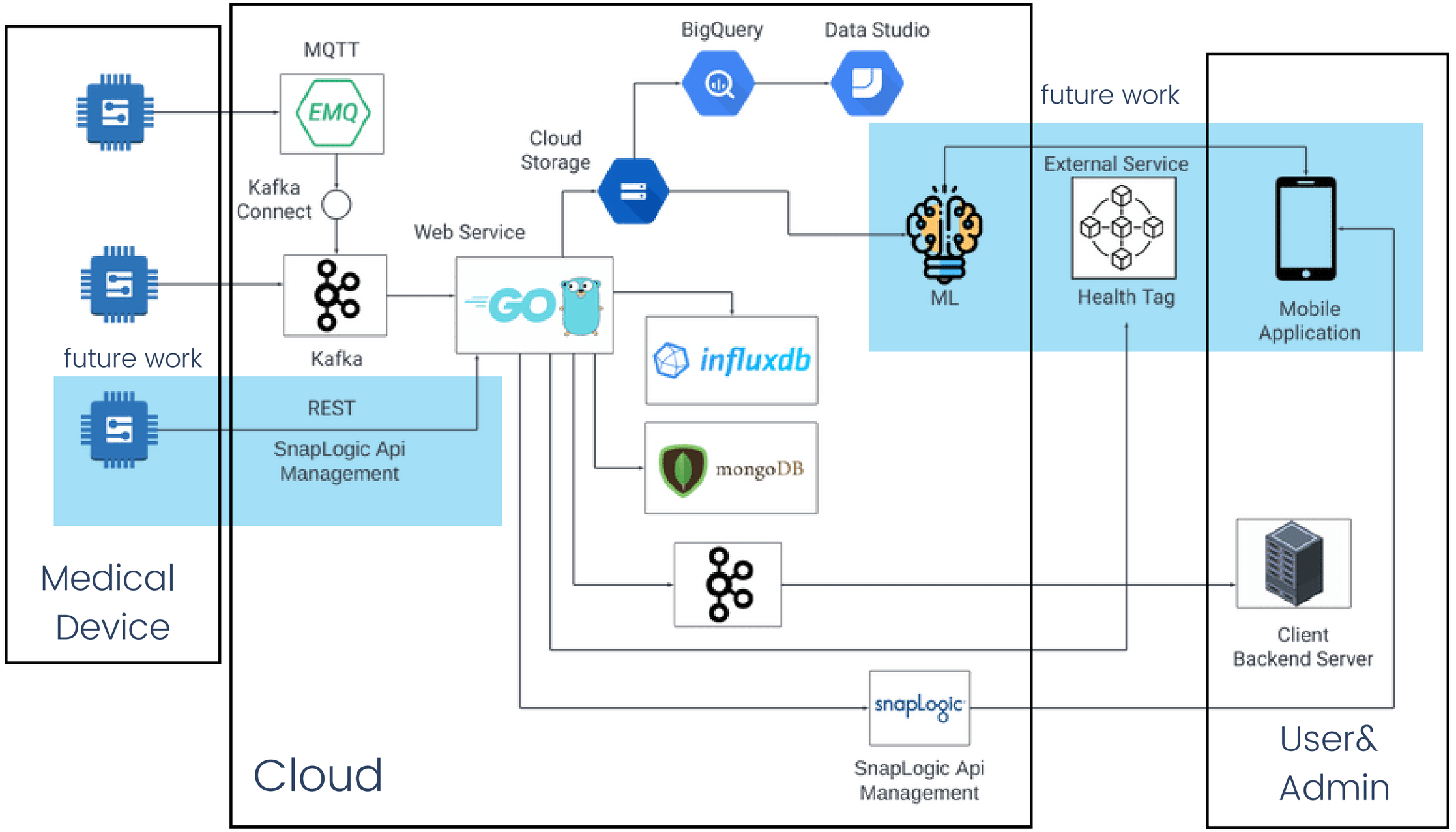
User Service system

- User registeration
- Dashboard
- Data transmission

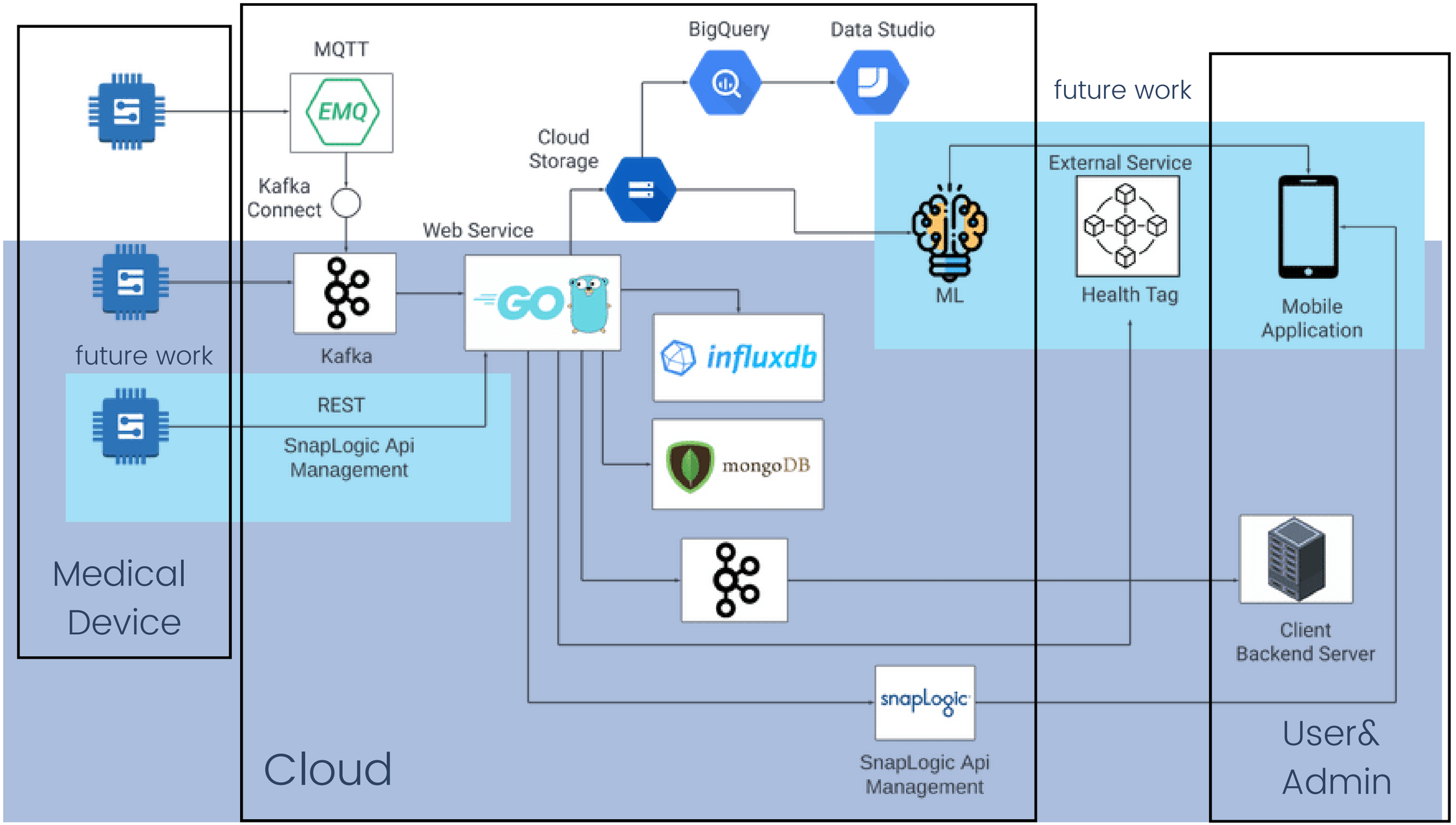
Admin system

- IoT device registeration
- User Authorization

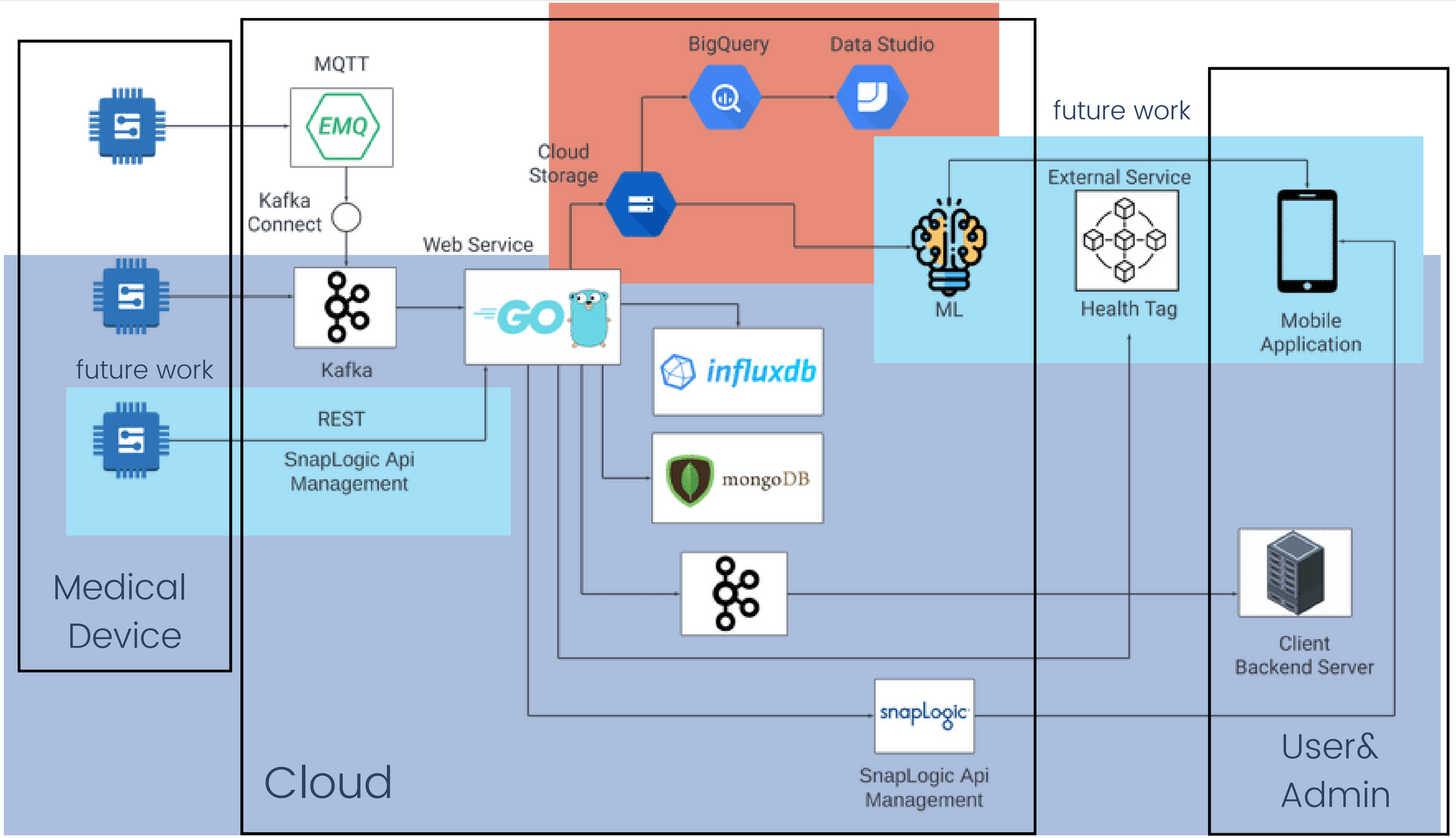
Architecture


MQTT
EMQ
Kafka
Connect
Kafka
future work
REST
SnapLogic Api
Management
Medical
Device
BigQuery
Data Studio
Cloud
Storage
Web Service
influxdb
mongoDB
snapLogic
SnapLogic Api
Management
Cloud
future work
ML
External Service
Health Tag
Mobile
Application
Client
Backend Server
User&
Admin

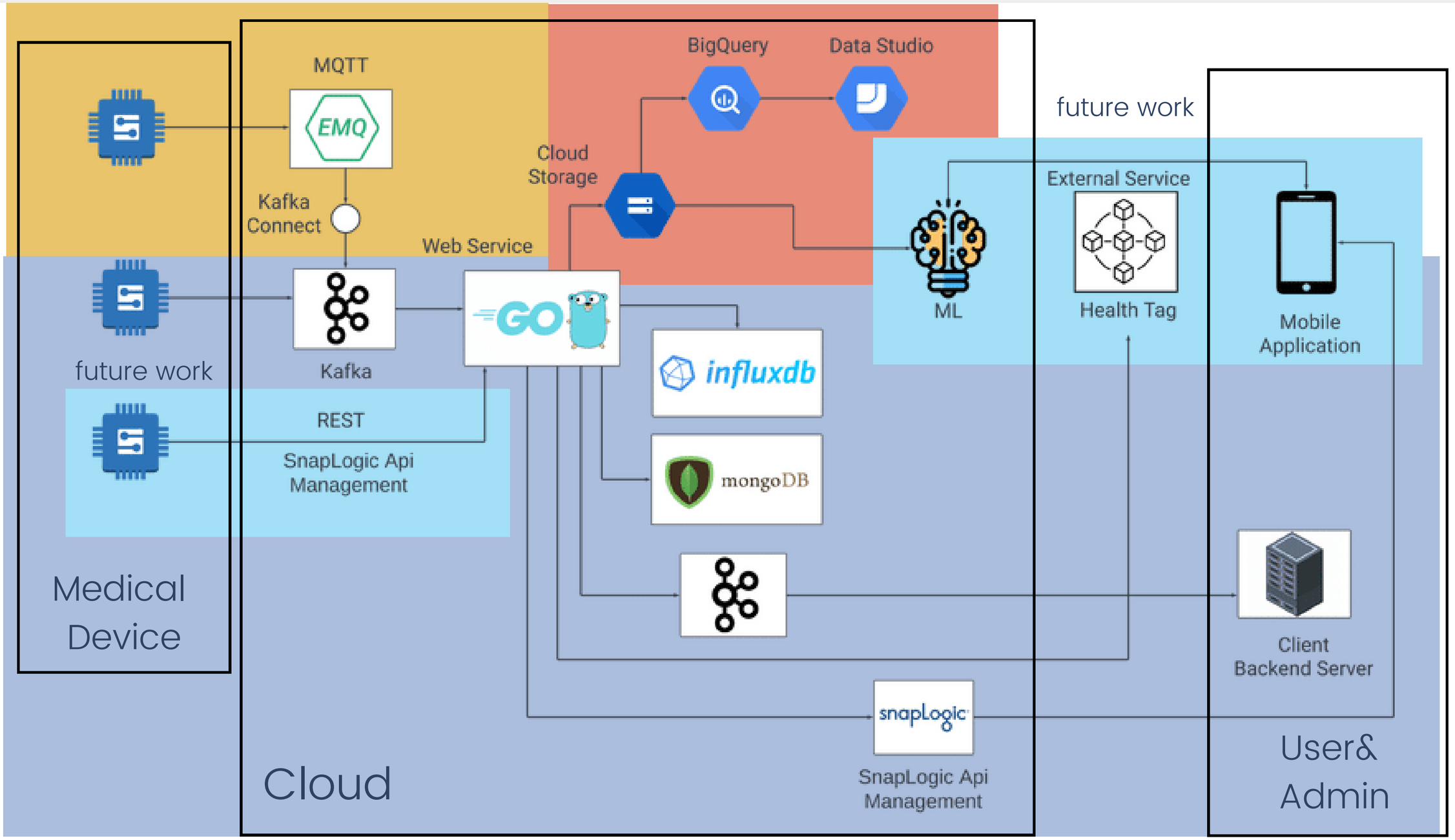
Architecture: Responsibility



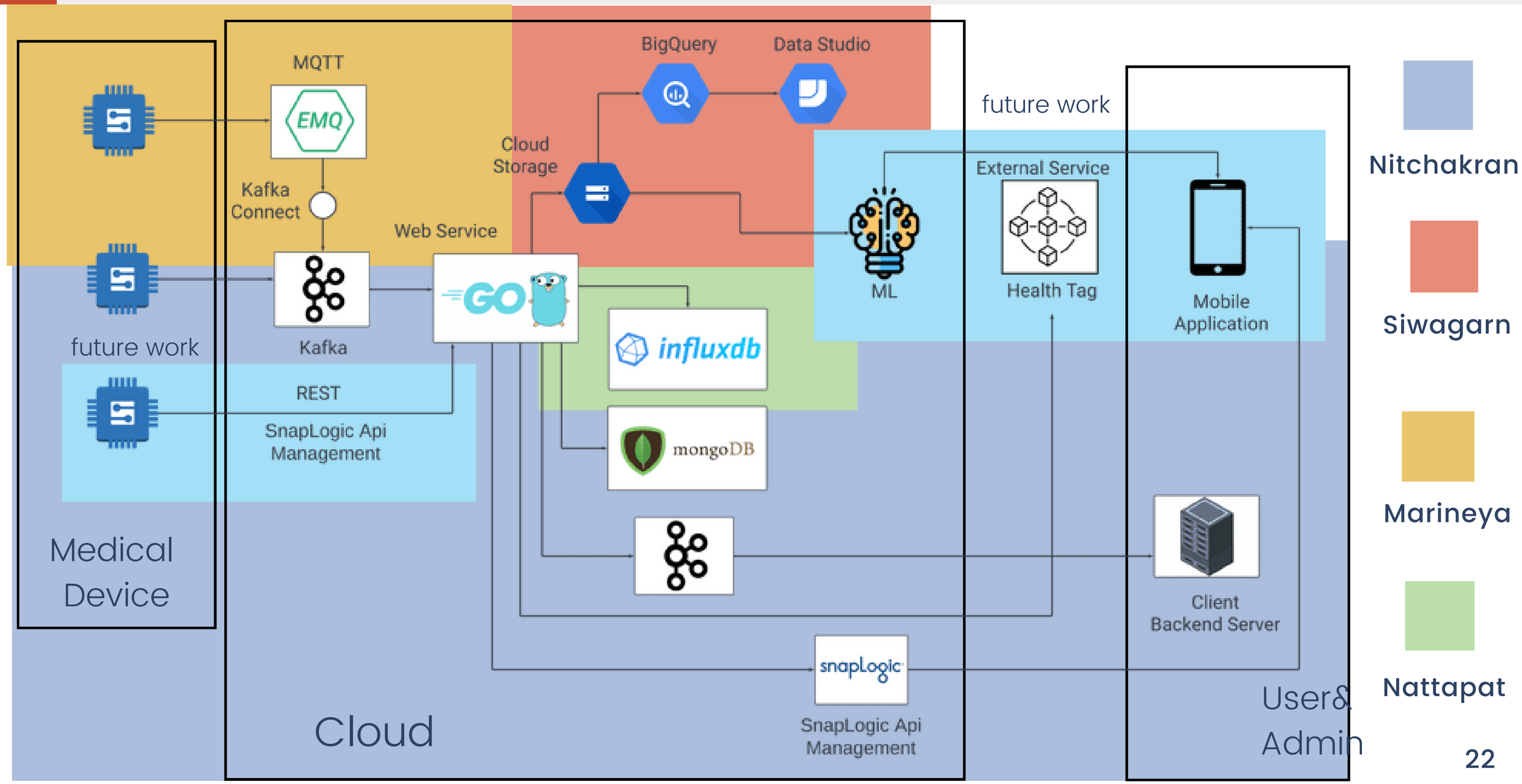
Architecture: Responsibility



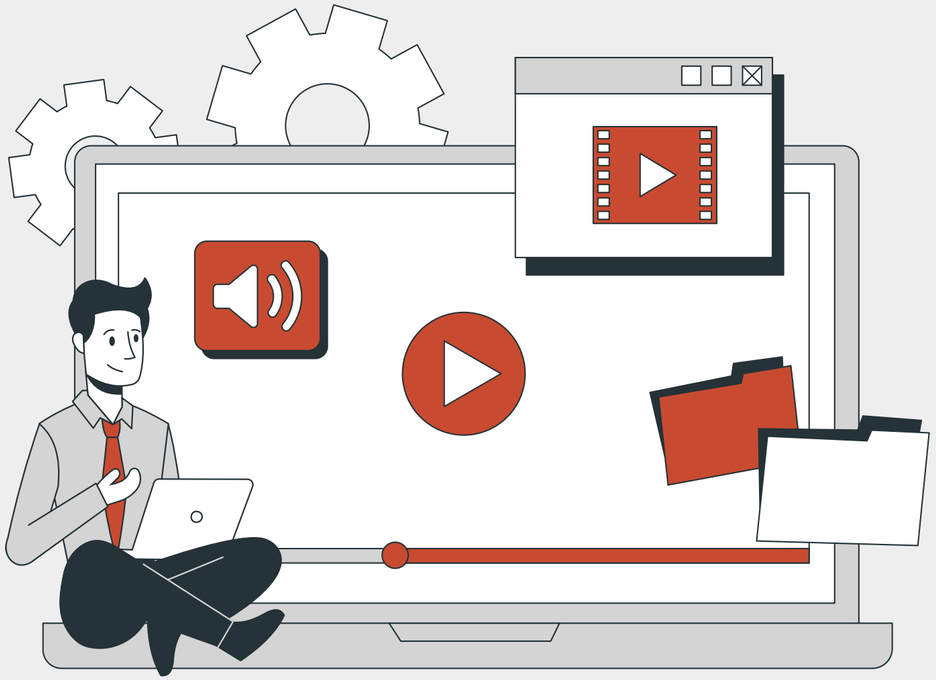
Architecture: Responsibility



Architecture: Responsibility

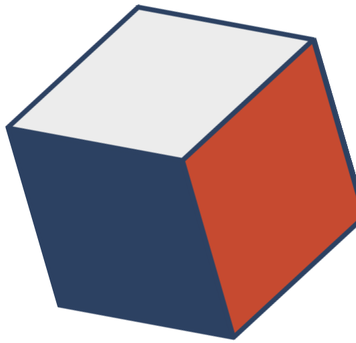


System



Data Receiving System

Data Management System

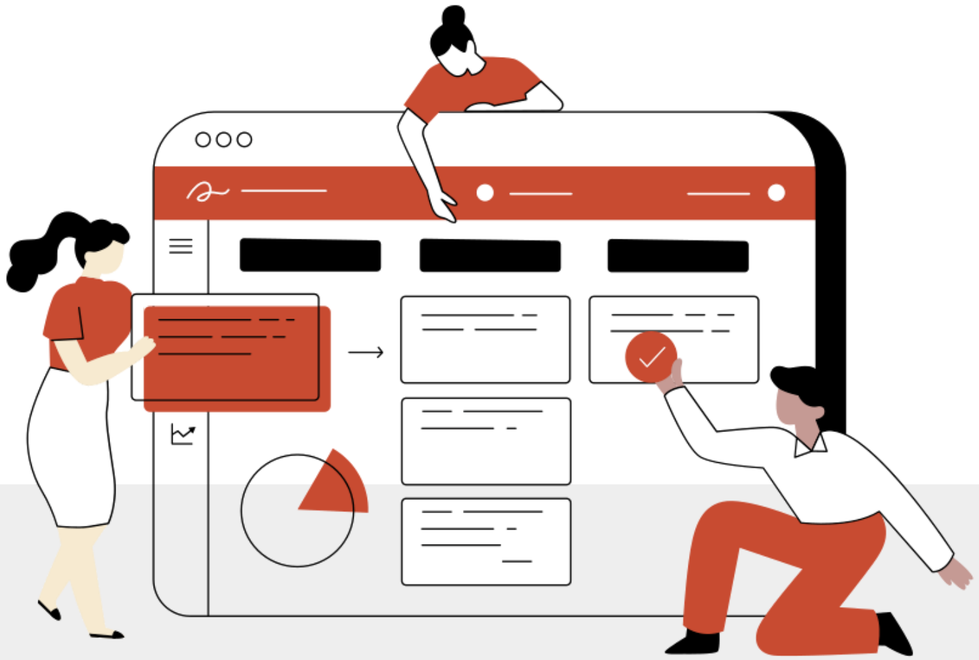


System

User Service System

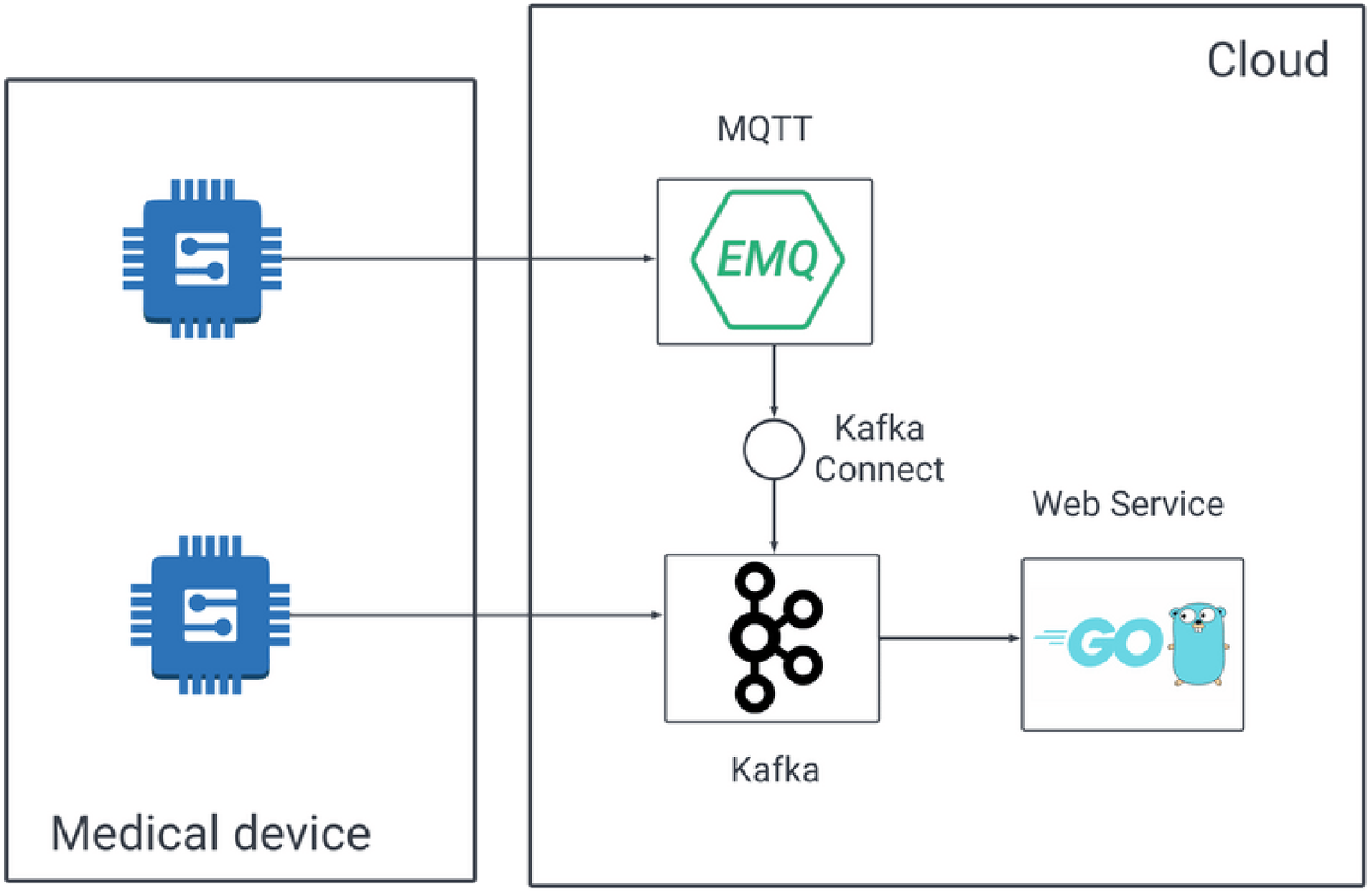
Admin System

Data Receiving System



MQTT Protocol

Kafka

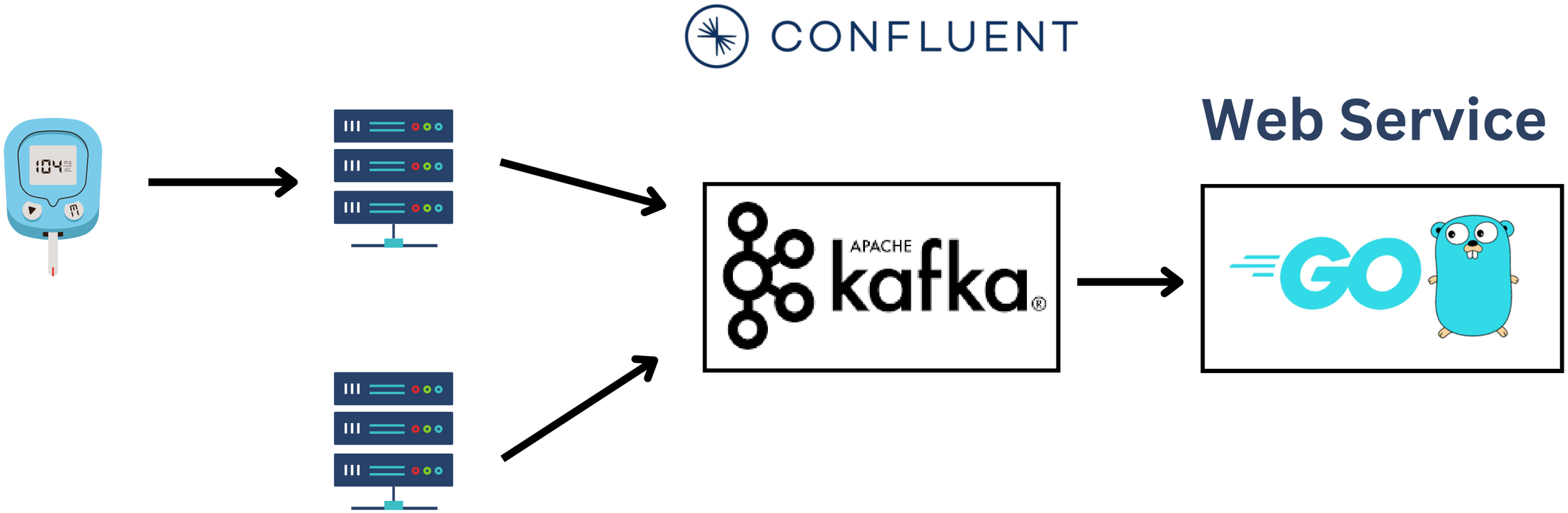


Data Receiving System: Receiving Data By MQTT

Connector

Source: https://www.confluent.io/lp/confluent-kafka

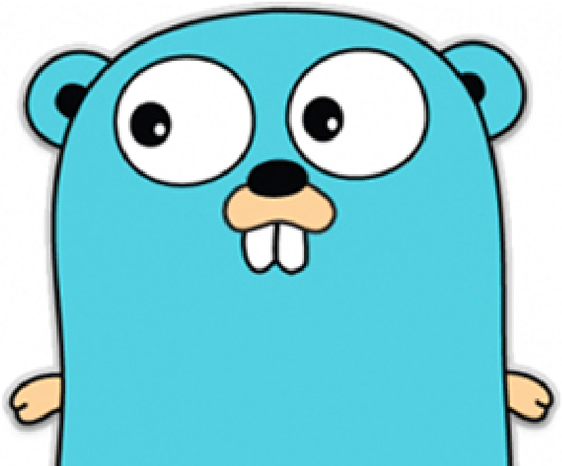
Data Receiving System: Receiving Data By Kafka



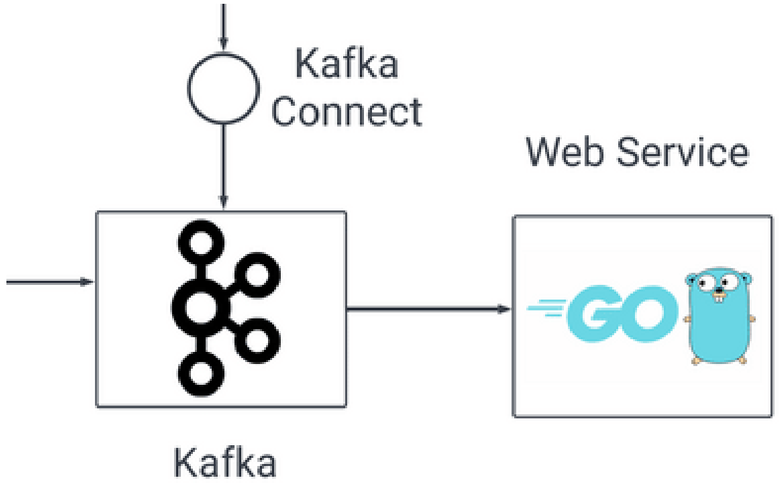
Fast

Concurrent Programming & Multithreading Support

Standard Library



Source: https://go.dev/#



Client Library

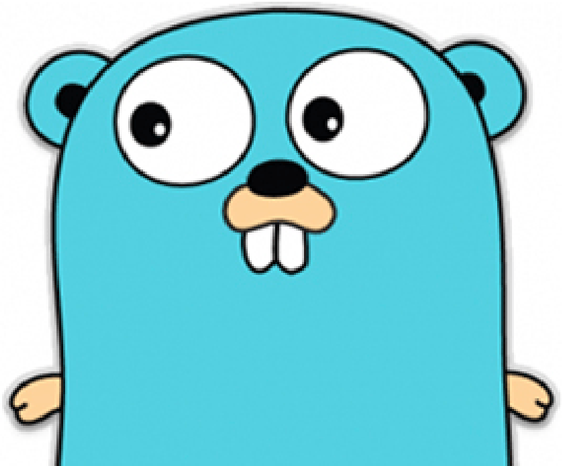
sarama

reference | build passing | codecov unknown

Sarama is an MIT-licensed Go client library for Apache Kafka version 0.8 (and later).

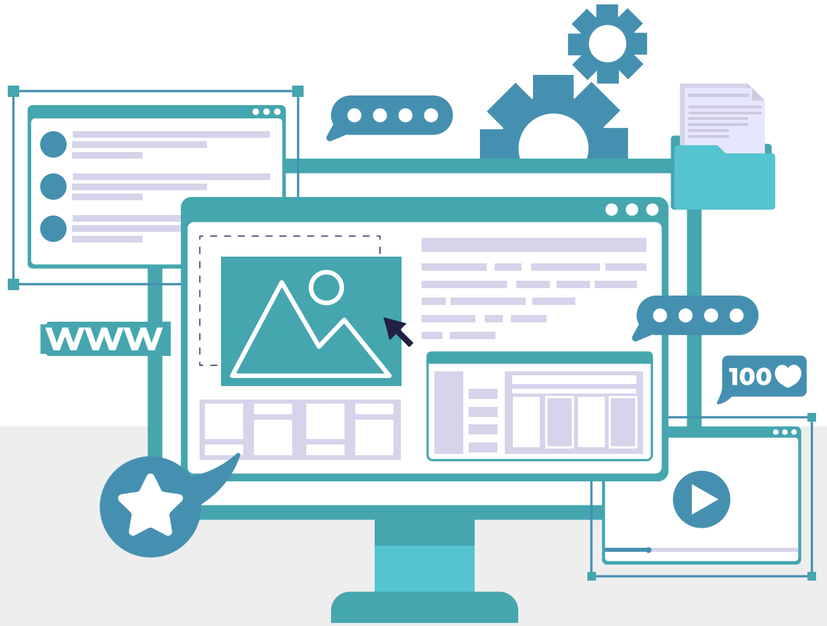
Provide Low-level API
& High-Level API

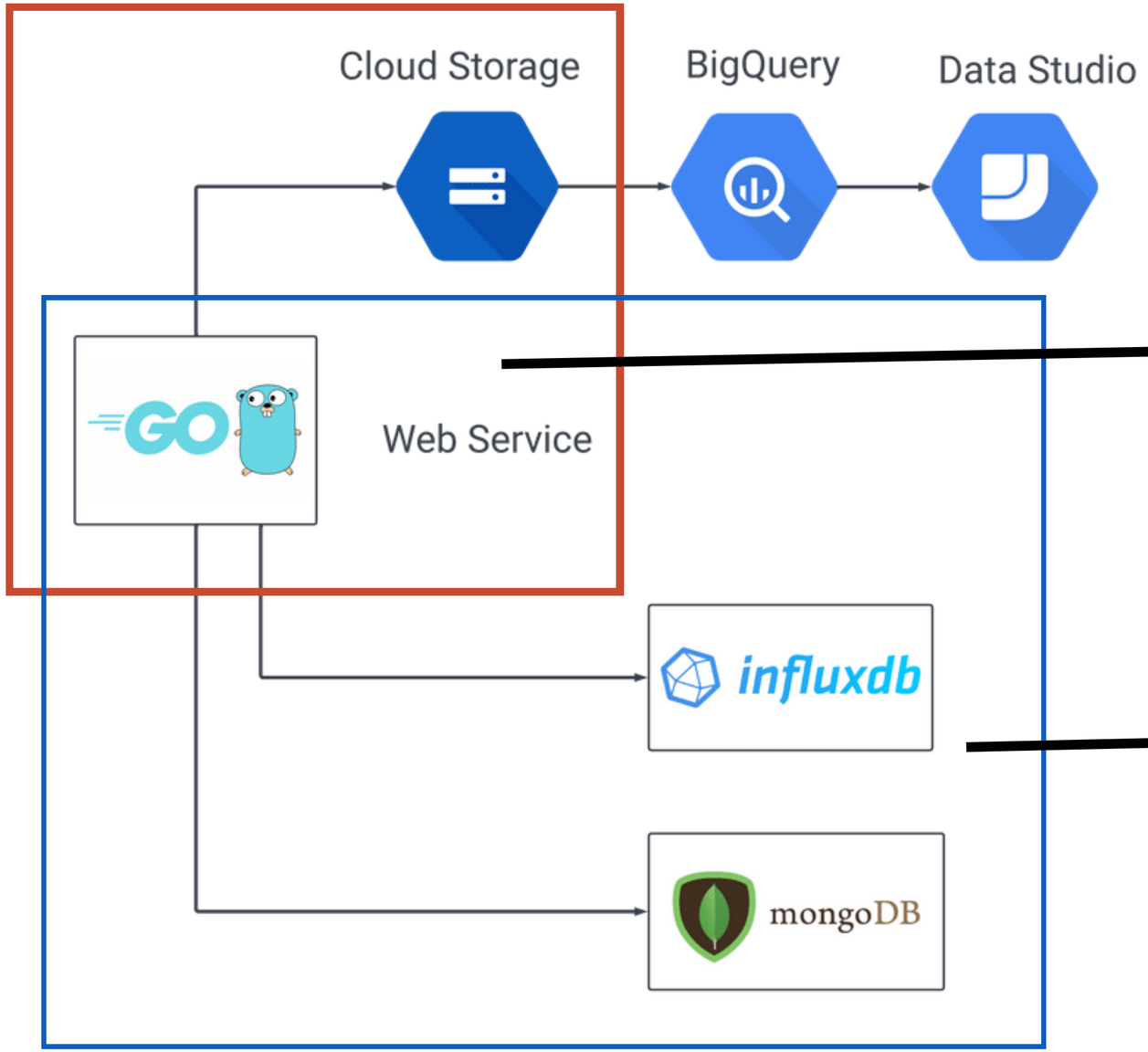
Beginner Friendly



Source: https://pkg.go.dev/github.com/shopify/sarama#section-readme

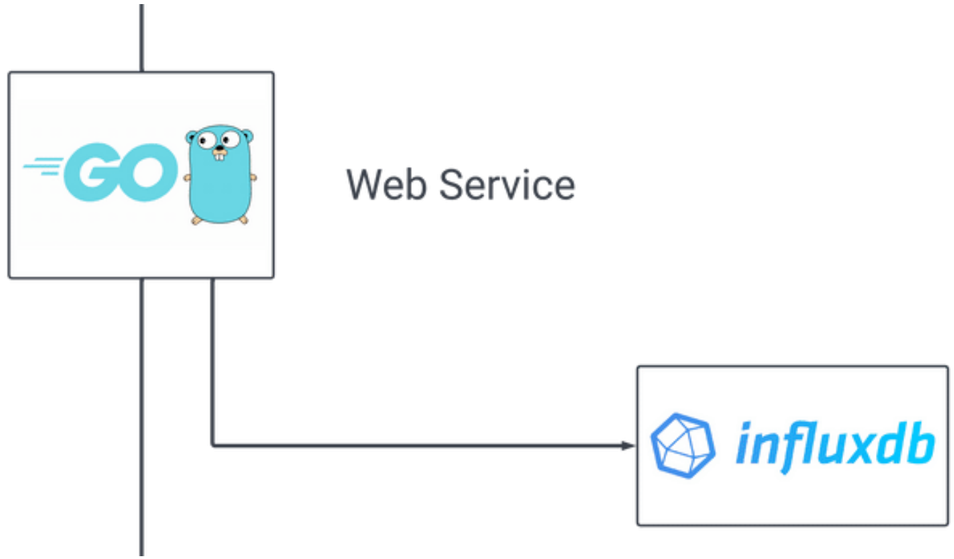
Data Management System





Upload Data To GCS

Save data into influxDB and mongoDB



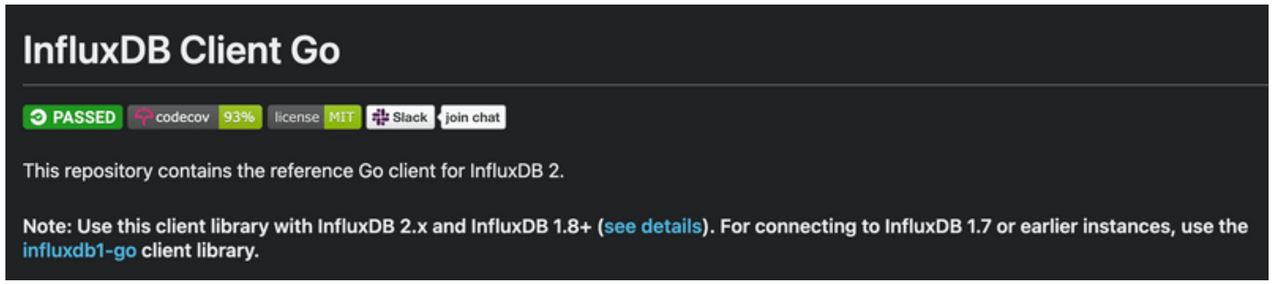
Why use InfluxDB?

High-speed write database

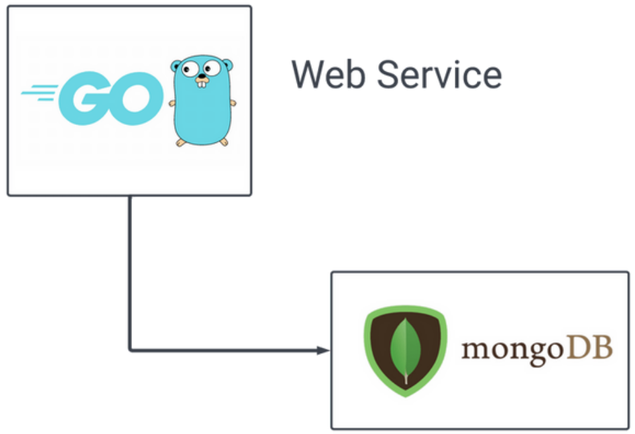
Time series data support

Query engine support time series data

Go client library support
(official library)



Source: https://www.influxdata.com/
source: https://pkg.go.dev/github.com/influxdata/influxdb

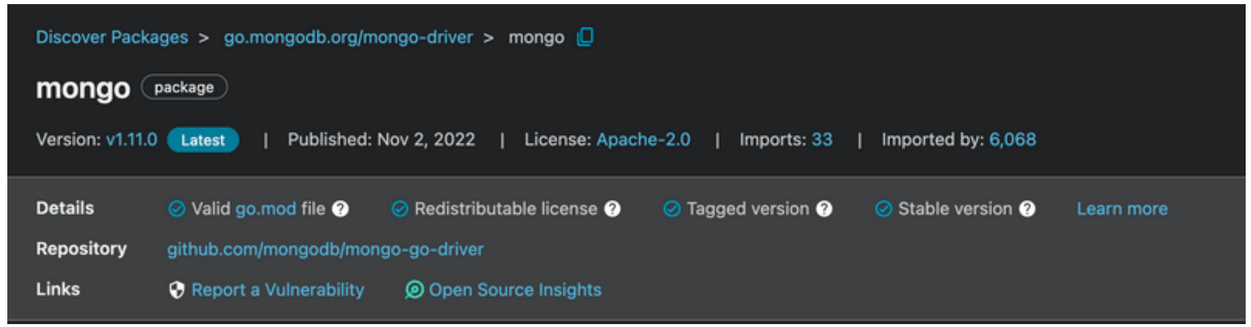


Why use MongoDB?

Schemaless

Easy to scale

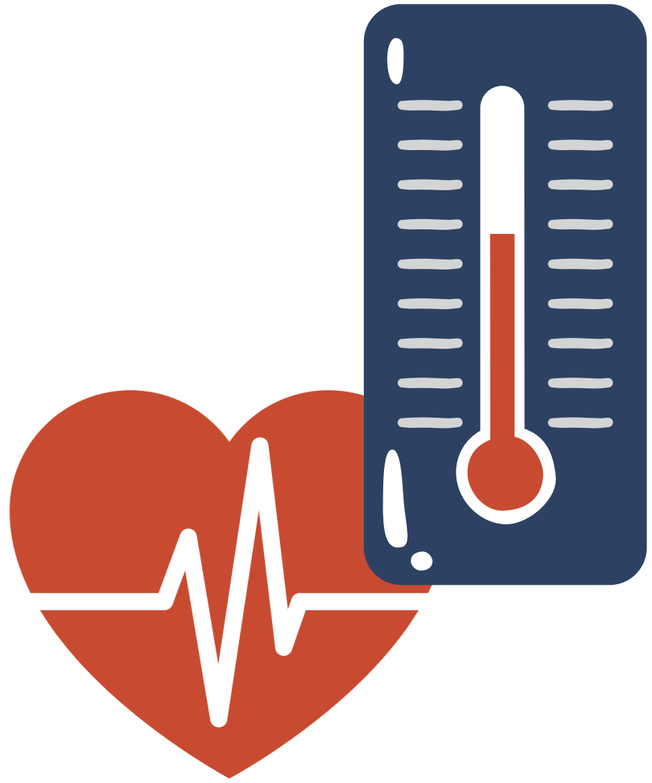
Go client library support
(official library)



Source: https://www.mongodb.com/why-use-mongodb
source: https://pkg.go.dev/go.mongodb.org/mongo-driver/mongo

1 model = 1 type of device

- 1st: temperature, heart rate
- 2nd: temperature, heart rate, o2



1 Bucket ⟷ 1 Model

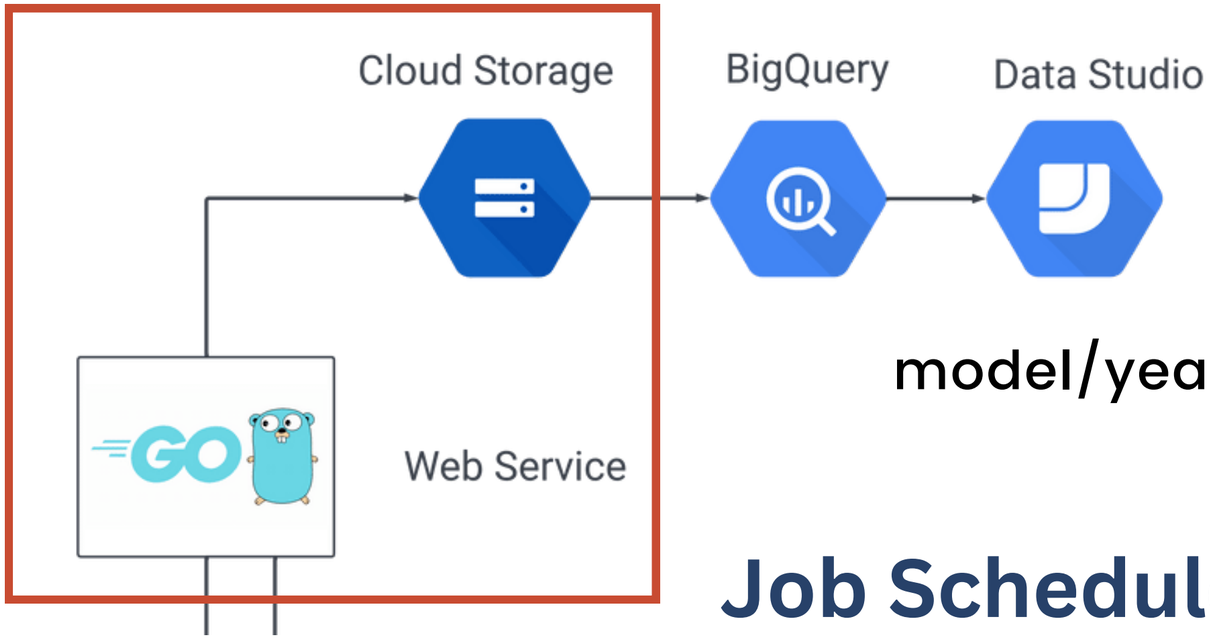
Identify device by deviceId
and keep it into _measurement
(influx structure)

Keep device value
into _field (influx structure)

1 Collection ⟷ 1 Model

Identify device by deviceId

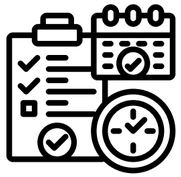
- raw data
- user data
- permission
- request real-time data



model/year/month/day/device_id_timestamp.csv

GCS folder structure

Job Scheduler



Query data from InfluxDB → Write CSV file → Upload file to GCS & Save log to Elasticsearch

Visualize

Elasticsearch

Kibana

Fast [used BKD tree]
Provide docker image
Provide API to interact with the server
Go client library Support

Elasticsearch and Kibana are compatible

Source: https://www.elastic.co/elasticsearch/
source: https://www.elastic.co/what-is/kibana

- Store raw data
- Different model of IoT devices
- Semi-structure data

Tool:  Google Cloud Storage

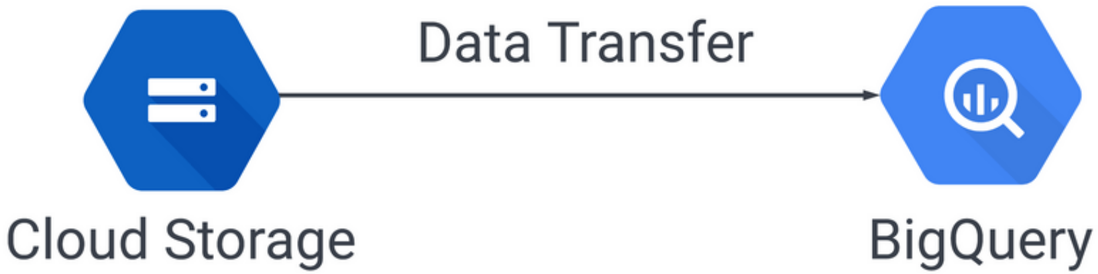
- Data Retention

Source: https://cloudmounter.net/amazon-s3-vs-google-cloud-storage/

- Fetch data from GCS to Bigquery
- 1 Table = 1 Model

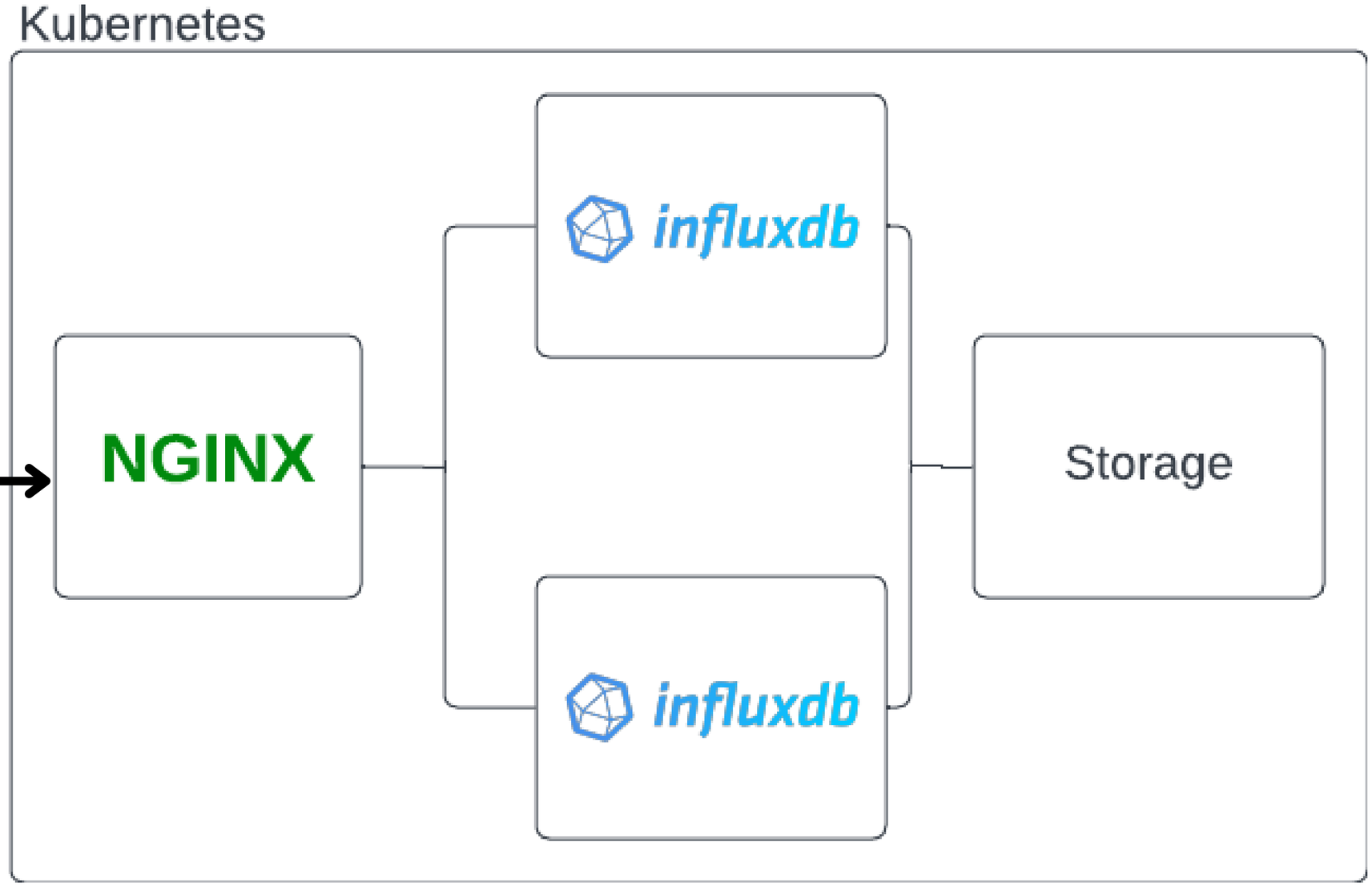
Tool: Google BigQuery

- Google service
- No additional tools
- Data transfer

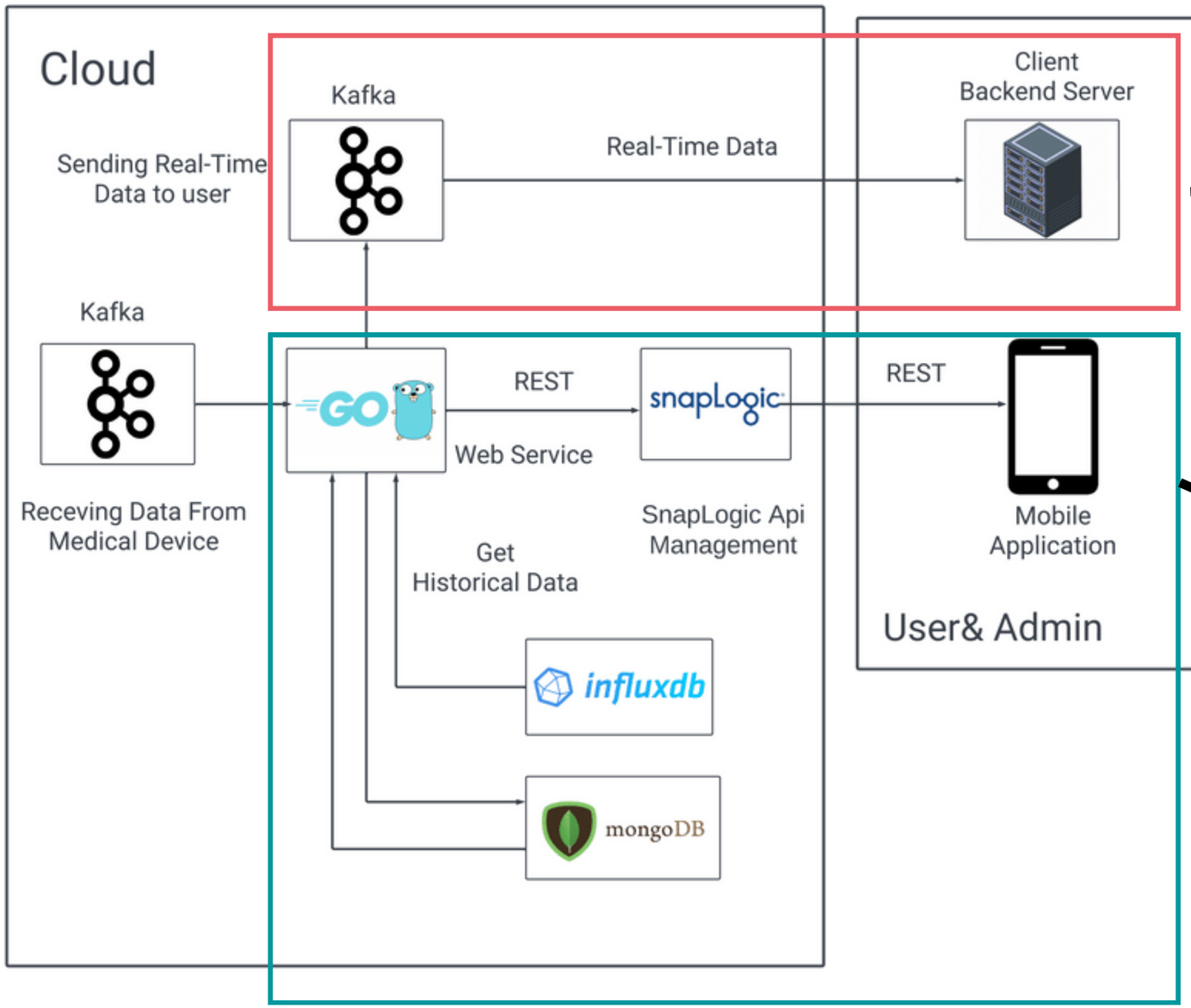


Source: https://cloud.google.com/bigquery/docs/reference/datatransfer/libraries#client-libraries-install-go

Data input from server



User Service System



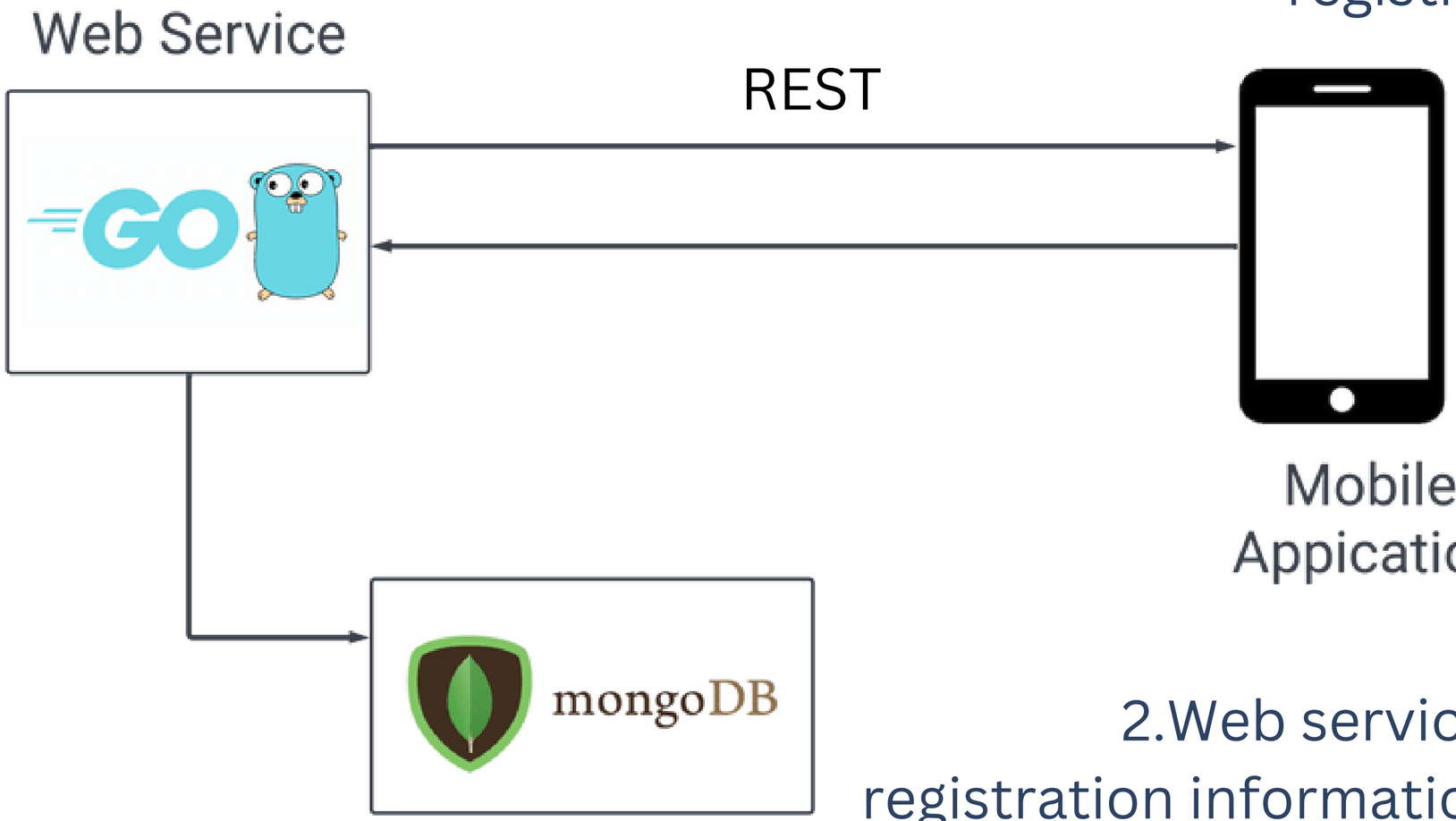
Sending real-time data &dashboard

User Registration

Sending historical data

User registration

1.User send
registration information

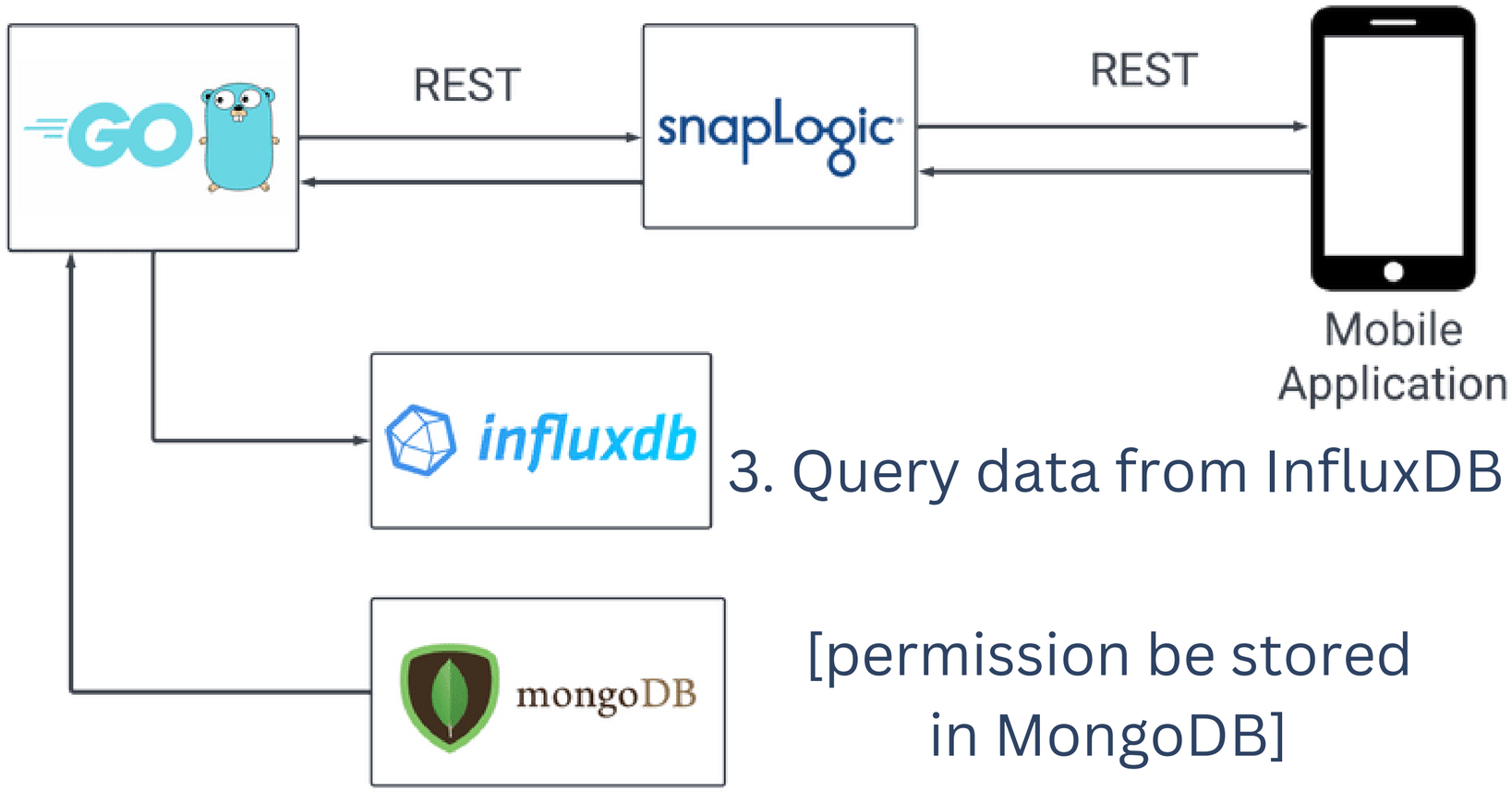


2.Web service store
registration information into MongoDB

Sending historical data

1. User request data to web service

4. Send data to user



2. Web service check user permission

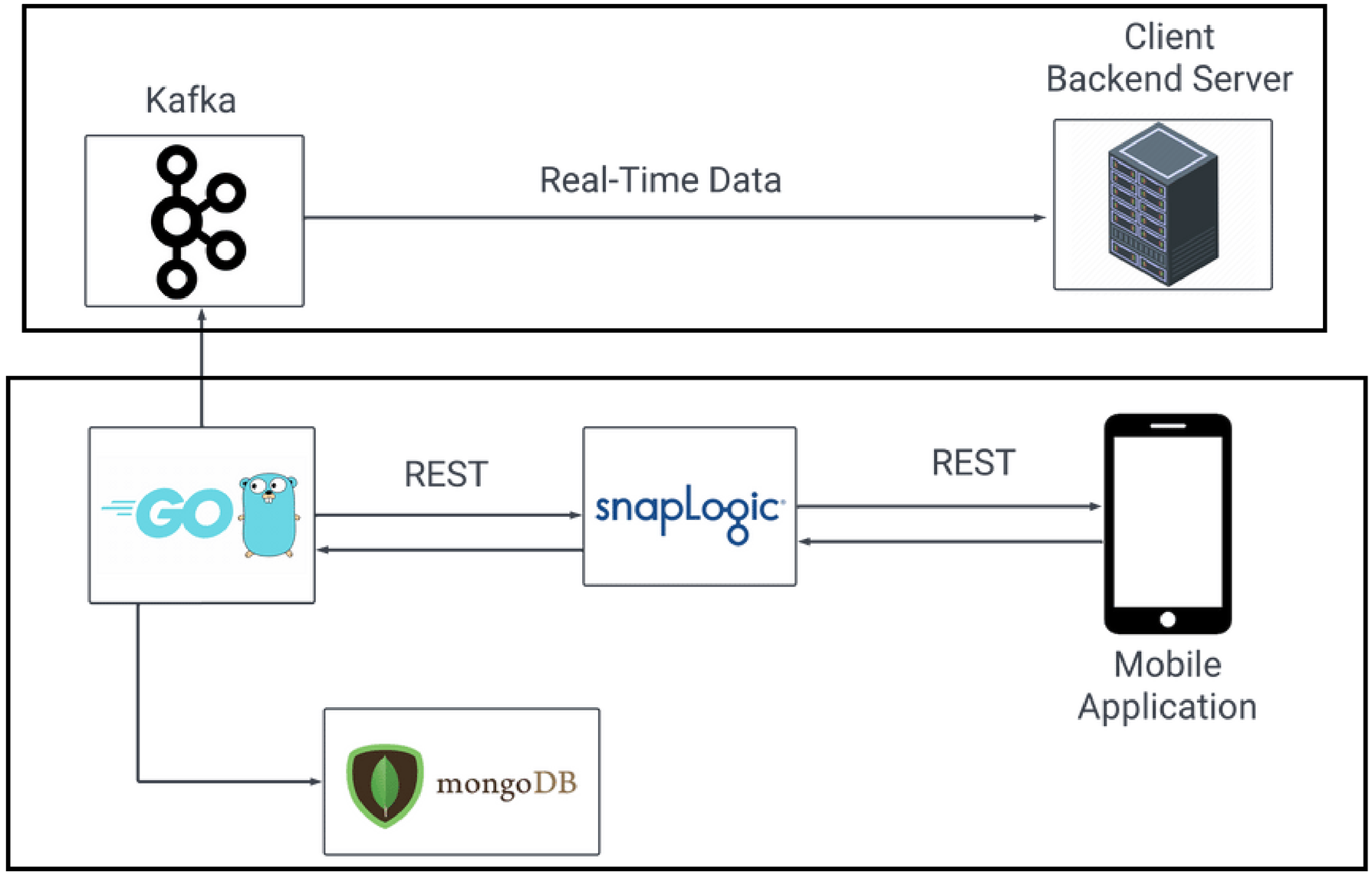
3. Query data from InfluxDB

[permission be stored in MongoDB]

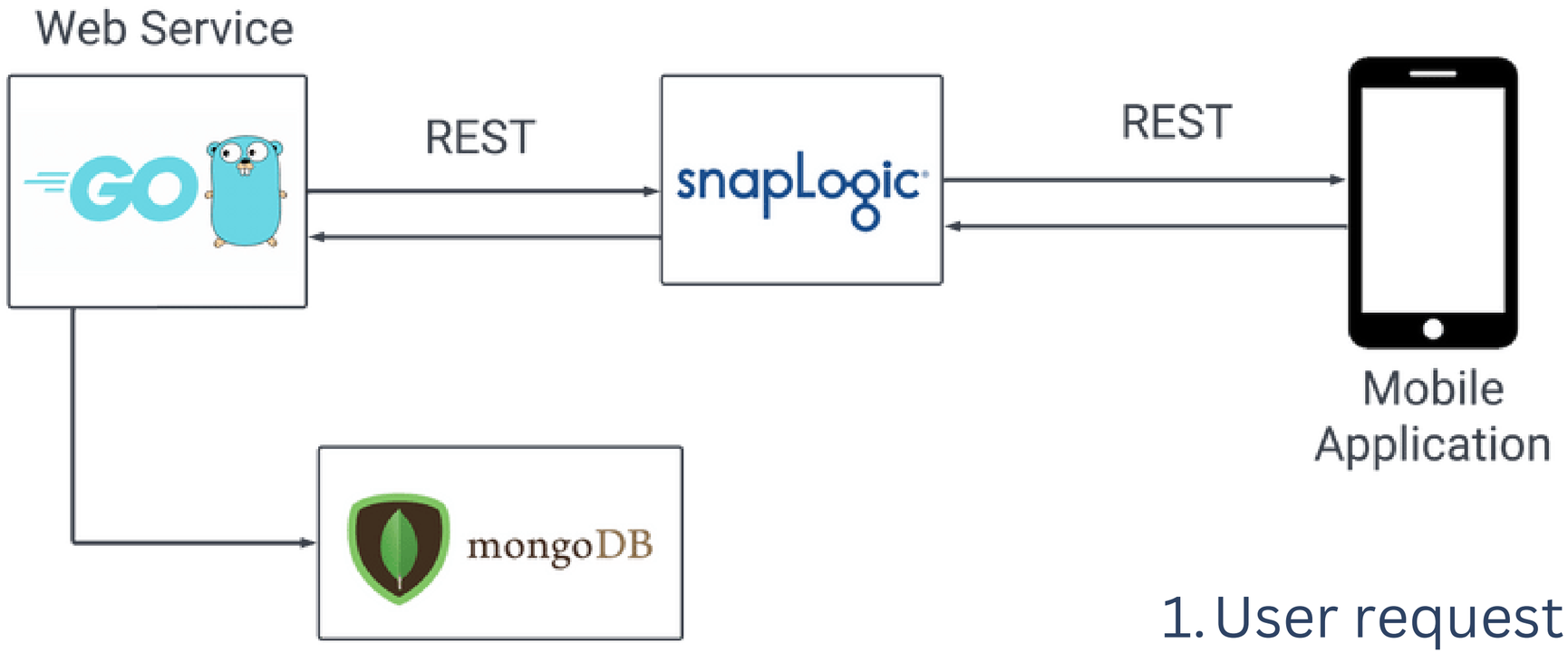
Sending real-time data

2

1



User Service System: Sending Data [Real-time Data]

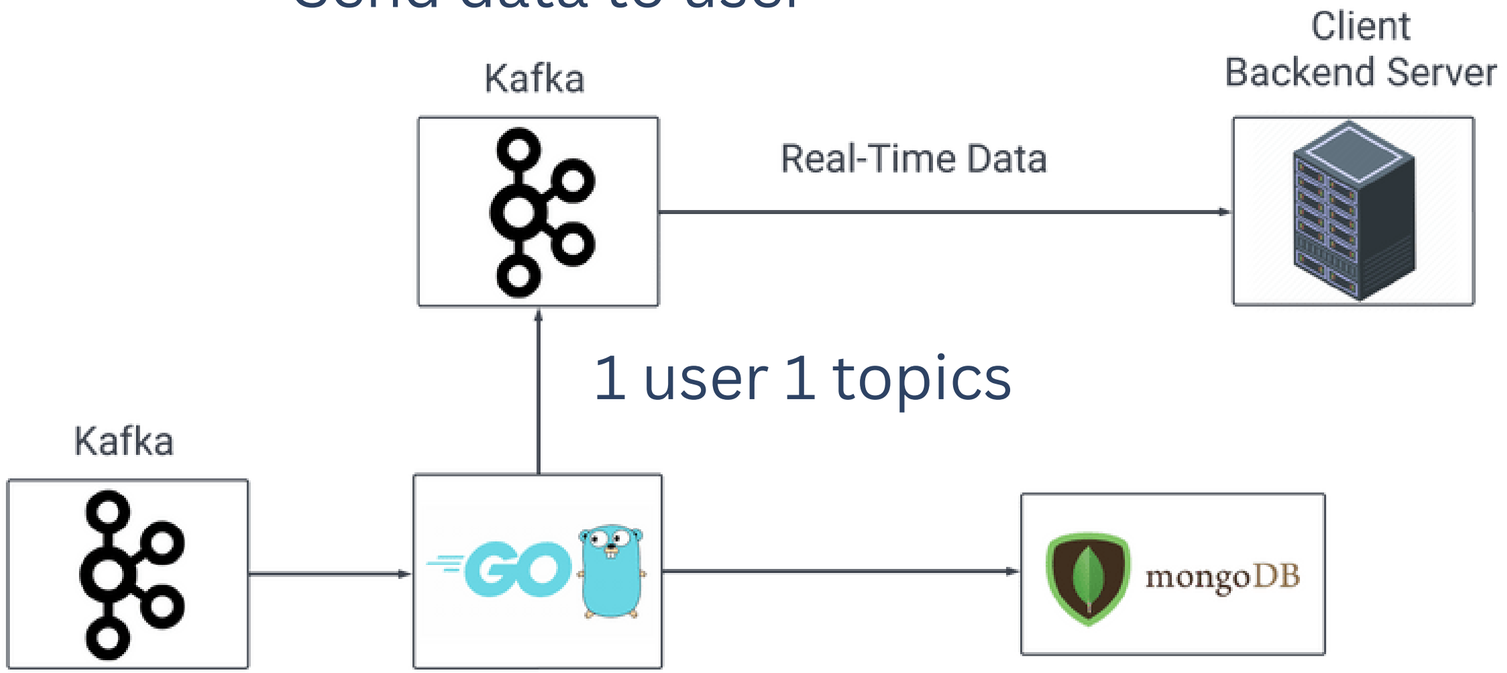



```
type RequestRealTimeData struct {
    ID          primitive.ObjectID `bson:"_id" json:"_id"`
    AppClientID string             `bson:"appClientId" json:"appClientId"`
    ModelName   string             `bson:"modelName" json:"modelName"`
    DeviceID    string             `bson:"deviceId" json:"deviceId"`
    StartTime   string             `bson:"startTime" json:"startTime"`
    EndTime     string             `bson:"endTime" json:"endTime"`
    IsUsed      bool               `bson:"isUsed" json:"isUsed"`
}
```

1. User request data to web service

2. Web Service check permission and save request into MongoDB

Send data to user



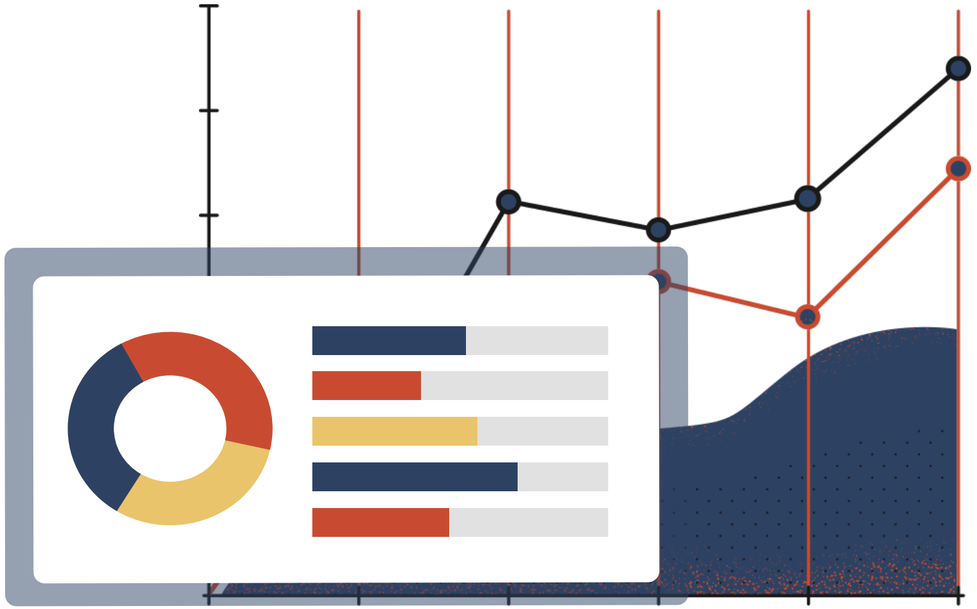
Receive data from device

2 Thread

1. Fetch request from MongoDB every 15 minutes

2. Consume and Produce data follow request information

- Monitoring healthcare data
- Monitoring IoT device
- Real-time & History

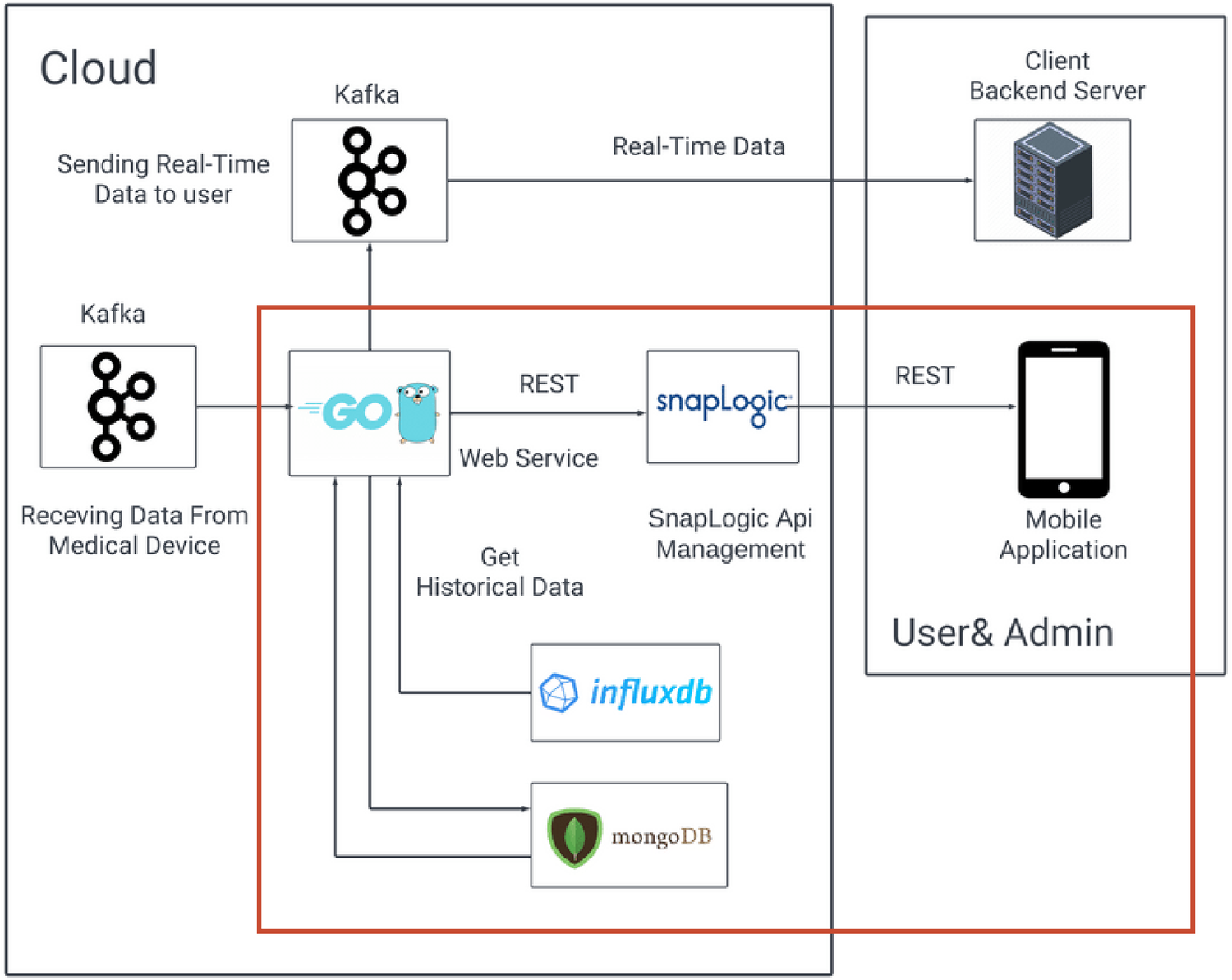


Tool: Google Data Studio

- Google service
- No additional tools
- automatically updated

Source: https://stepstraining.co/analytics/what-is-google-data-studio

Admin System

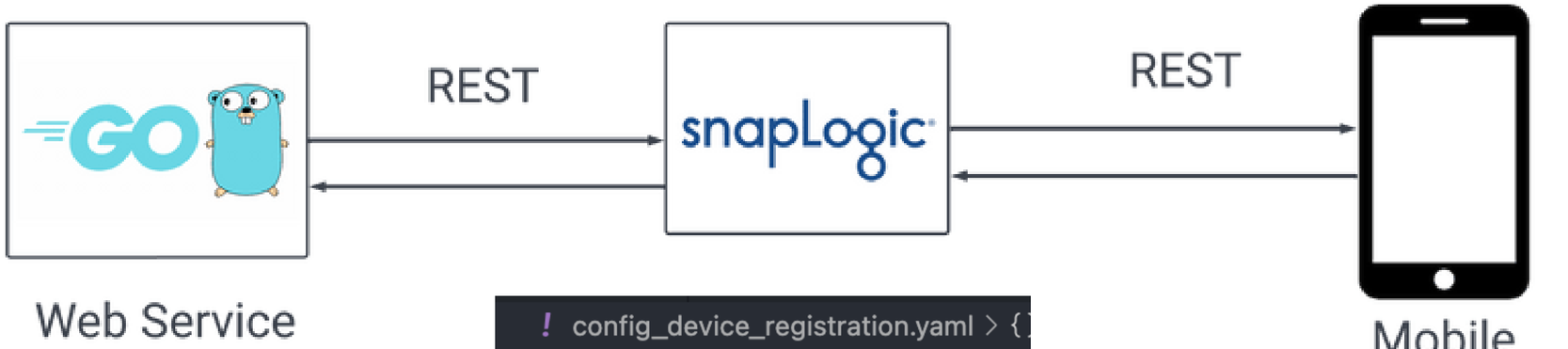


IoT device Registration

Manage User Permission

2. Web Service write and update configuration file, influxdb bucket and kafka topic

1. Admin send device information [ie what value that can measure]

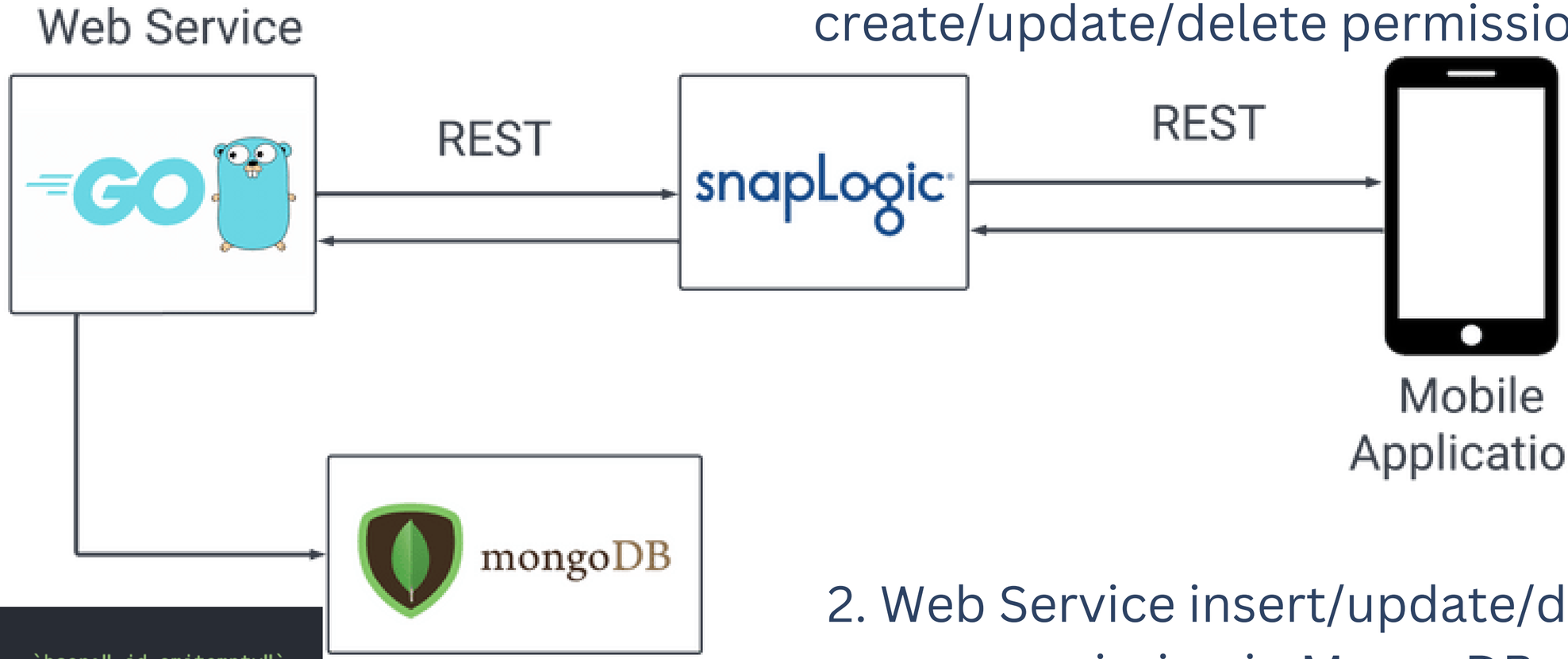


Configuration file

config_device_registration.yaml

```yaml
ipd-monitoring:
  keys:
  - bloodPressure
  - heartRate
  - o2Saturation
  - respirationRate
  - temperature
  datatypes:
  - NUMBER
  - NUMBER
  - NUMBER
  - NUMBER
  - NUMBER
  fieldLength: 5
```

1. User requests to create/update/delete permission



2. Web Service insert/update/delete permission in MongoDB

```
type Role struct {
    ID        string    `bson:"_id,omitempty"`
    ModelName string    `bson:"modelName"`
    DeviceID  string    `bson:"deviceId"`
    StartTime string    `bson:"startTime"`
    EndTime   string    `bson:"endTime"`
    Mode      dto.Mode `bson:"mode"`
}
```

Demo

Admin System

- Register device
- Manage user permission

User Service System

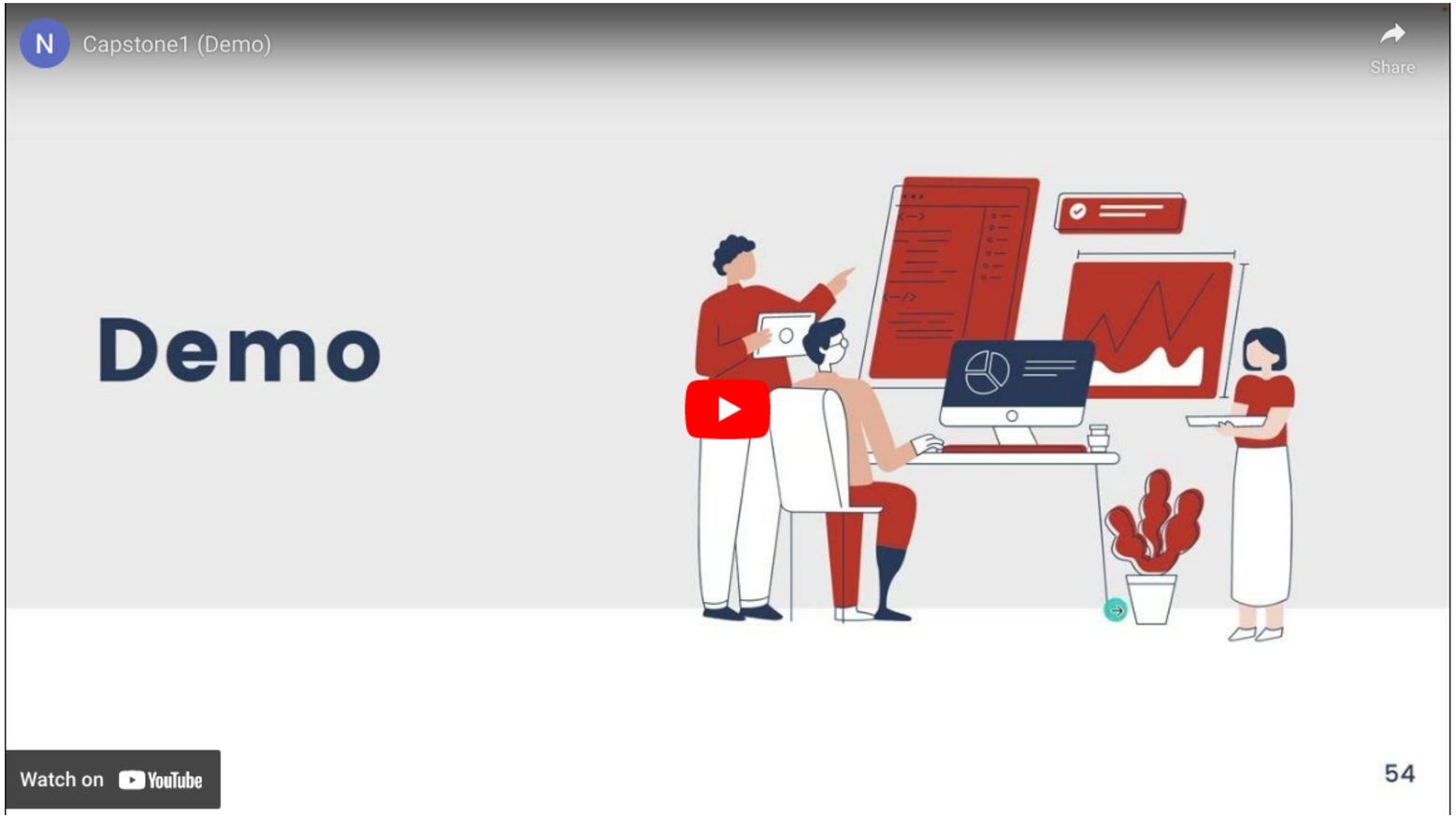
- Registration
- Login
- Request historical data
- Request real-time data

Data Management System

- Upload file to GCS and save log into elasticsearch
- Biquery and Data dashboard
- Store data into Influxdb and mongoDB

Data Receiving System

- Connect MQTT to Kafka



https://youtu.be/wpKDtCOFo9M

Summary

Done

	Task	Responsibility	Duration(Weeks)	2022 AUG	SEP	OCT	NOV
1	เลือกหัวข้อโปรเจค		4				
2	ศึกษาโครงสร้างของระบบ		4				
2.1	AWS IoT Core	ทุกคน	1				
2.2	MQTT	ทุกคน	3				
2.3	Kafka	ทุกคน	3				
2.4	InfluxDB	ทุกคน	1				
2.5	Flutter	ทุกคน	2				
2.6	HealthTag	ทุกคน	1				
3	ระบบการรับข้อมูลจากอุปกรณ์การแพทย์		7				
3.1	รองรับข้อมูลที่ส่งผ่าน MQTT protocol	มารีนญา	4				
3.2	รองรับการรับข้อมูลจาก Kafka	ณิชกานต์	5				
4	ระบบการจัดการข้อมูล		8				
4.1	การบันทึกข้อมูลลง influxDB และ mongoDB	ณิชกานต์	4				
4.3	การบันทึกข้อมูลลง data lake, data warehouse	ณิชกานต์	5				
4.5	การทำ time series database clustering	ณัฐภัทร	4				
5	ระบบการติดต่อผู้ใช้งาน		5				
5.1	การลงทะเบียนผู้ใช้งาน	ณิชกานต์	5				
5.2	การทำ dashboard สำหรับผู้ใช้	ศิวกาญจน์	3				
5.3	การส่งข้อมูล IoT ไปให้ผู้ใช้งาน	ณิชกานต์	2				
6	ระบบแอดมิน		5				
6.1	การลงทะเบียนอุปกรณ์ IoT	ณิชกานต์	2				
6.2	การจัดการสิทธิการเข้าถึงของผู้ใช้งานในการเข้าถึงข้อมูล	ณิชกานต์	3				

Todo

	Task	Responsibility	Duration(Weeks)	2022																			
				DEC				JAN				FEB				MAR				APR			
7	ทดสอบระบบการทำงานของระบบ		5																				
7.1	ทดสอบระบบการรับข้อมูล	มาริษญา	2																				
7.2	ทดสอบระบบการจัดการข้อมูล	ณิชกานต์	2																				
7.3	ทดสอบระบบการติดต่อผู้ใช้งาน	ศิวกาญจน์	2																				
7.4	ทดสอบระบบแอดมิน	ณัฐภัทร	2																				
8	เพิ่มประสิทธิภาพของระบบ		7																				
8.1	เรียนรู้การเพิ่มประสิทธิภาพของระบบ	ทุกคน	3																				
8.2	เพิ่มประสิทธิภาพระบบการรับข้อมูล	มาริษญา	3																				
8.3	เพิ่มประสิทธิภาพระบบการจัดการข้อมูล	ณิชกานต์	2																				
8.4	เพิ่มประสิทธิภาพระบบการติดต่อผู้ใช้งาน	ศิวกาญจน์	2																				
8.5	เพิ่มประสิทธิภาพระบบแอดมิน	ณัฐภัทร	2																				
9	วัดผลการเพิ่มประสิทธิภาพของระบบ		4																				
9.1	วัดผลเพิ่มประสิทธิภาพระบบการรับข้อมูล	มาริษญา	1																				
9.2	วัดผลเพิ่มประสิทธิภาพระบบการจัดการข้อมูล	ณิชกานต์	1																				
9.3	วัดผลเพิ่มประสิทธิภาพระบบการติดต่อผู้ใช้งาน	ศิวกาญจน์	1																				
9.4	วัดผลเพิ่มประสิทธิภาพระบบแอดมิน	ณัฐภัทร	1																				
10	รายงานและเอกสารต่างๆ	ทุกคน	3																				

The End

Thank you for listening