Mar 21, 2022

Computer Engineering,
Chulalongkorn University

IoT Healthcare

Progression

ADVISOR
Assoc. Prof. Kultida Rojviboonchai

Expert (SnapLogic)
Thanawut Ananpiriyakul
Tanapat Ruengsatra
Kanya Kiatsakdawong

Member

Nitchakran Chaipojjana
6231322621

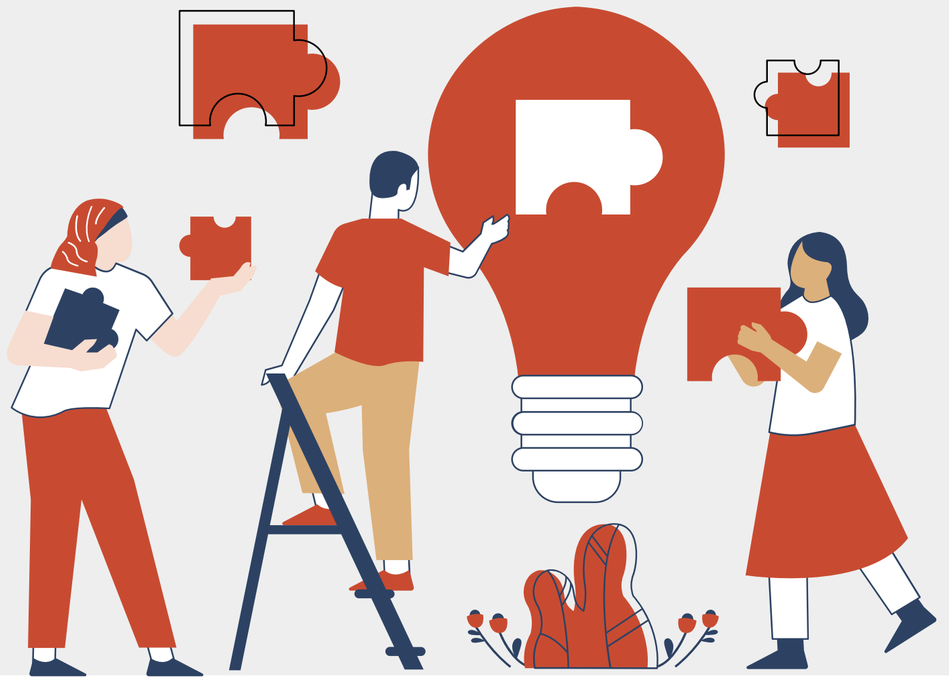
Siwagarn Jitwarodom
6231363321

Marineya Tajoparung
6231352421

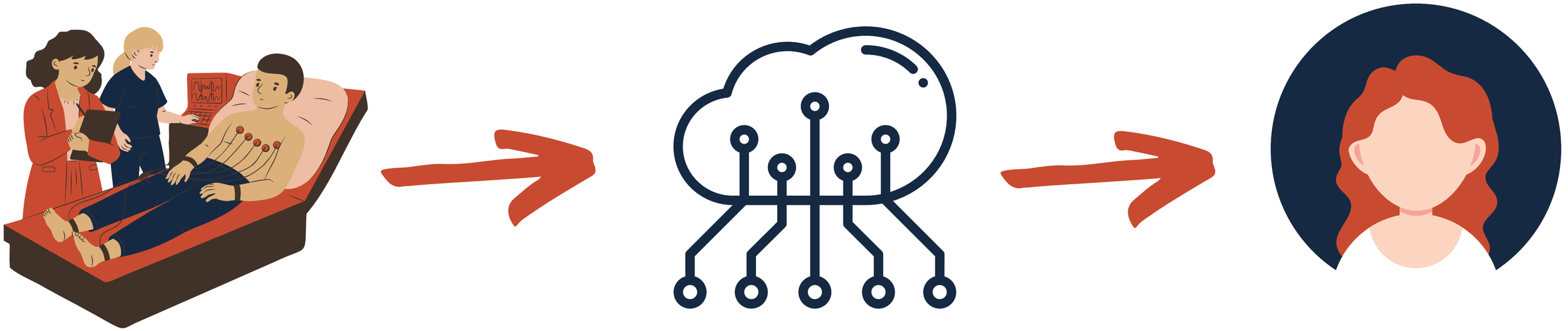
Nattapat Jaruchaisittikul
6230177821

Table of Contents

Project Overview



Open Healthcare IoT Platform

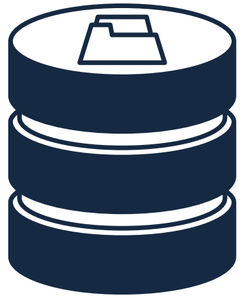


Requirement

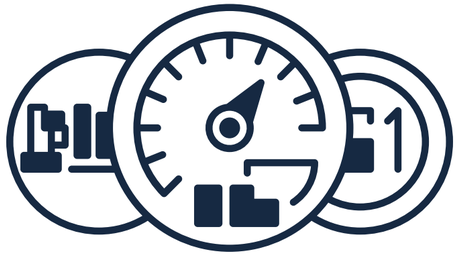
Modules

add keep delete

Data Receiving



Data Management



Developer Service

Admin Service

Logging

Functional requirements: Data Receiving System

Receive the IoT data from Kafka

Receive the IoT data from MQTT

Functional requirements: Data Management System

Save the received data into database

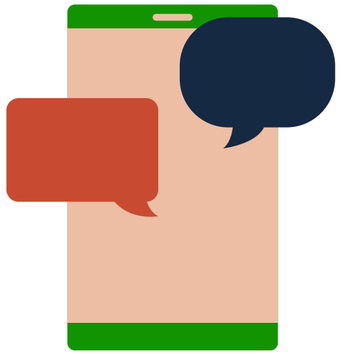
Upload data in google cloud storage to bigquery

Upload data in database to google cloud storage

Dashboard of IoT data for data analyst

Functional requirements: Developer Service System

Dashboard for CRUD an application



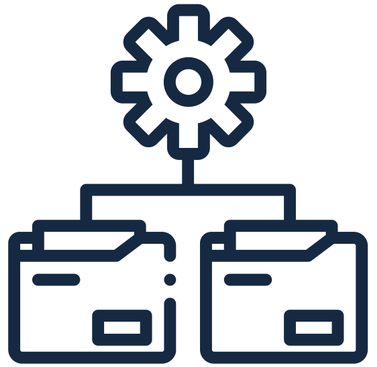
Dashboard to create and read a subscription

The Kafka cluster shall authenticated the application consumer.

Functional requirements: Developer Service System

The Kafka cluster shall allow applications to consume data according to permission

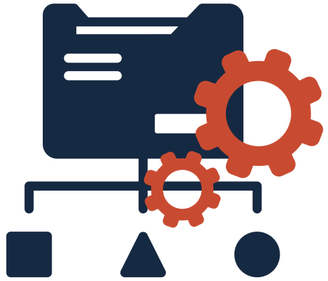
Authenticate the application api



Provide historical data and real-time data according to permission.

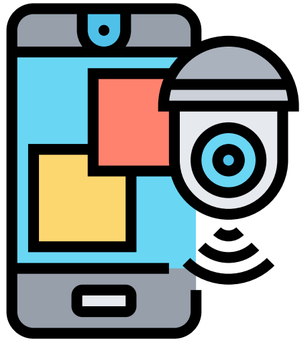
Functional requirements: Admin Service System

Dashboard for CRD developers



Dashboard to manage subscription's permission.

Dashboard to get all subscriptions .

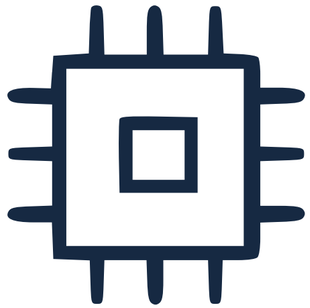


Dashboard for CRD medical models

Functional requirements: Logging System

Uploaded data status log.

All requested api log



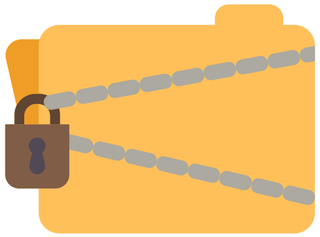
Monitor system's service uptime, cpu usage and memory usage

Logging dashboard

Non-functional requirements

Available 24/7

Backup site



Keep patient data confidential

Provide data based on user's authorization

Non-functional requirements

Response within 15 min

For historical data requests, the system should pass the load test within following criteria

- response 95% of requests within 3 seconds
- average data size = 6000 dp
  - throughput 100 requests/sec
  - 100 devices โดย 1 devices 1 ชั่วโมงจะผลิตข้อมูลมา 60 dp

15

For future data requests, the system should pass the load test within following criteria

- response 95% of processing requests within 15 seconds
- throughput = 80 message/s

Testing Guideline

- Test using 95th Percentile method
- Sort response time in ascending order
- Calculate 95th percentile of response time
- Repeat 10 times
- Find the average

Non-functional requirements

Response within 15 min

For historical data requests, the system should pass the load test within following criteria

- response 95% of requests within 3 seconds
- average data size = 6000 dp [throughput 100 requests/sec]
- 100 devices โดย 1 devices 1 ชั่วโมงจะผลิตข้อมูลมา 60 dp

15

For future data requests, the system should pass the load test within following criteria

- response 95% of processing requests within 15 seconds
- throughput = 80 message/s

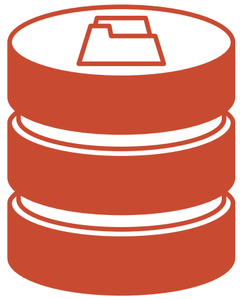
throughput = 80 message/s

- 40 ICU rooms / hospital
- 5 IoT healthcare devices / room
- 40x5 = 200 IoT healthcare devices in a hospital
- Some devices send data every 5 sec, some send data every 15 sec
- Assume every device send data every 5 sec.
- Frequency = 200/5 = 40 message/s
- Bound up value (multiply by 2) = 40*2 = 80 message/s

To-be System

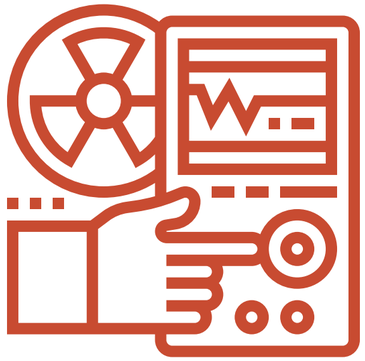
Systems

Data Recieving



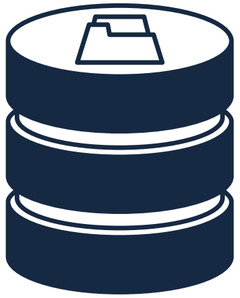
Data Management

User Service

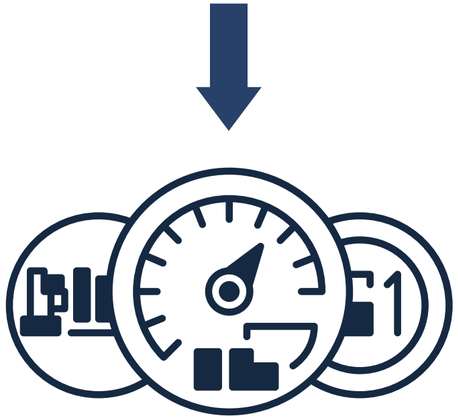


Admin System

Data Receiving



Data Management



Developer Service

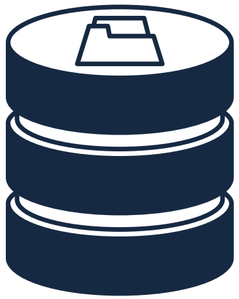
Admin Service

Logging

Systems

Data Receiving

- MQTT
- Kafka



Data Management

- Database: MongoDB, InfluxDB
- Data Lake & Data Warehouse
- Dashboard

Developer Service

- Data Transmission (real-time & historical)
- Dashboard: application & subscriptions
- application's customer authentication & authorization

Admin Service

- Dashboard: device, developer, developer's subscriptions

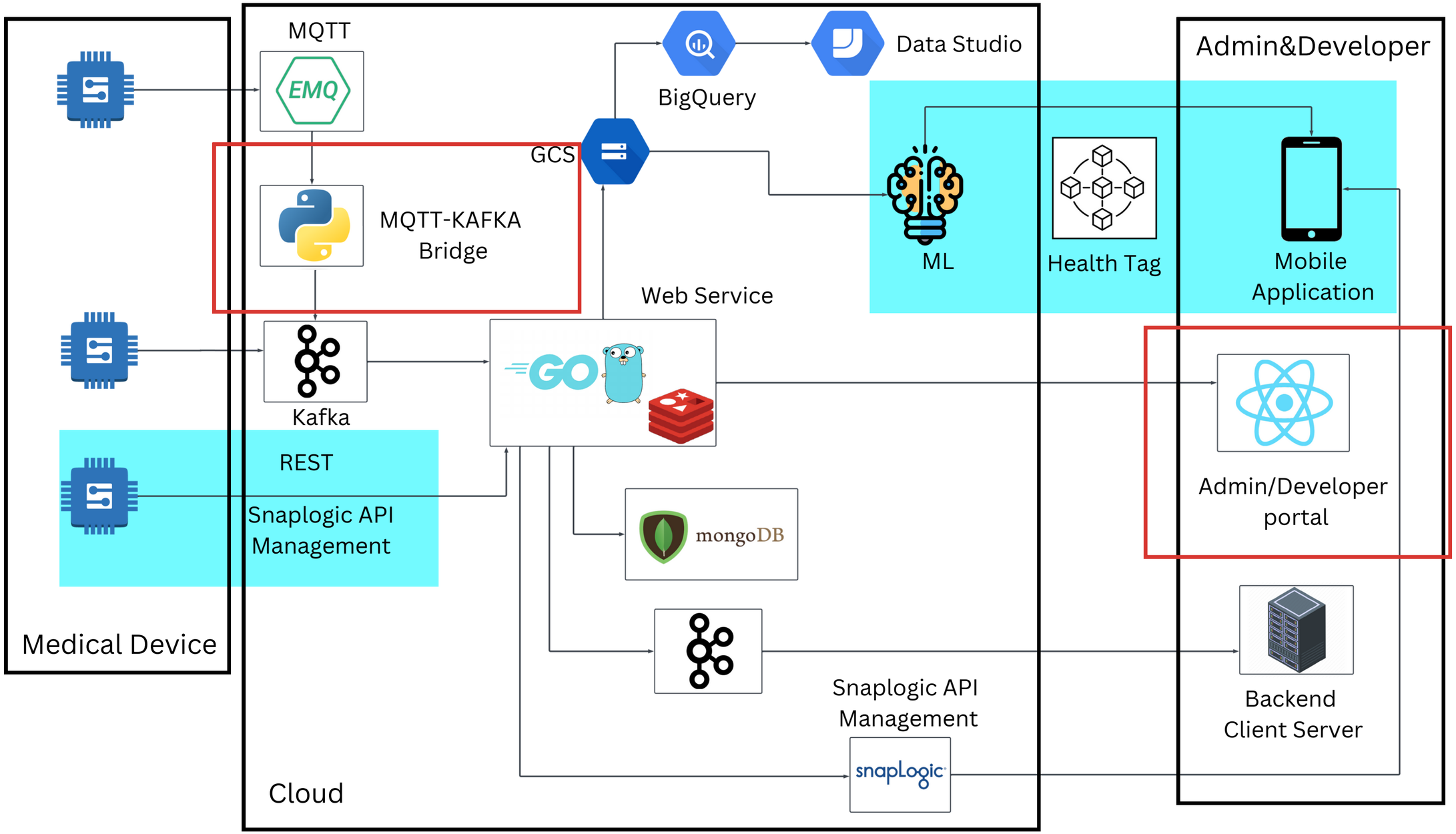
Systems

Logging

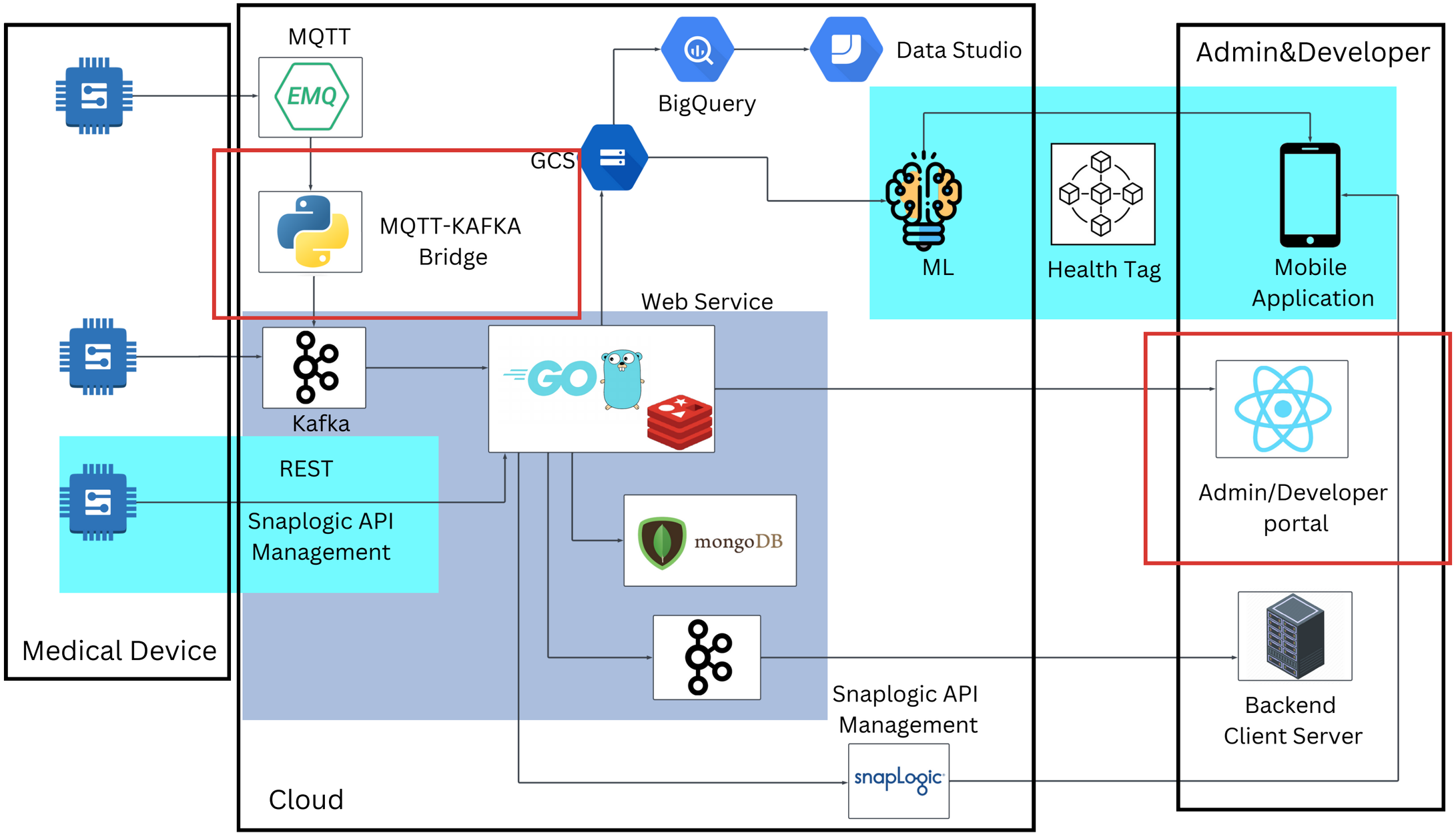
- Log of uploading to Data Lake & Data Warehouse
- api log
- system monitoring
- dashboard log

Architecture

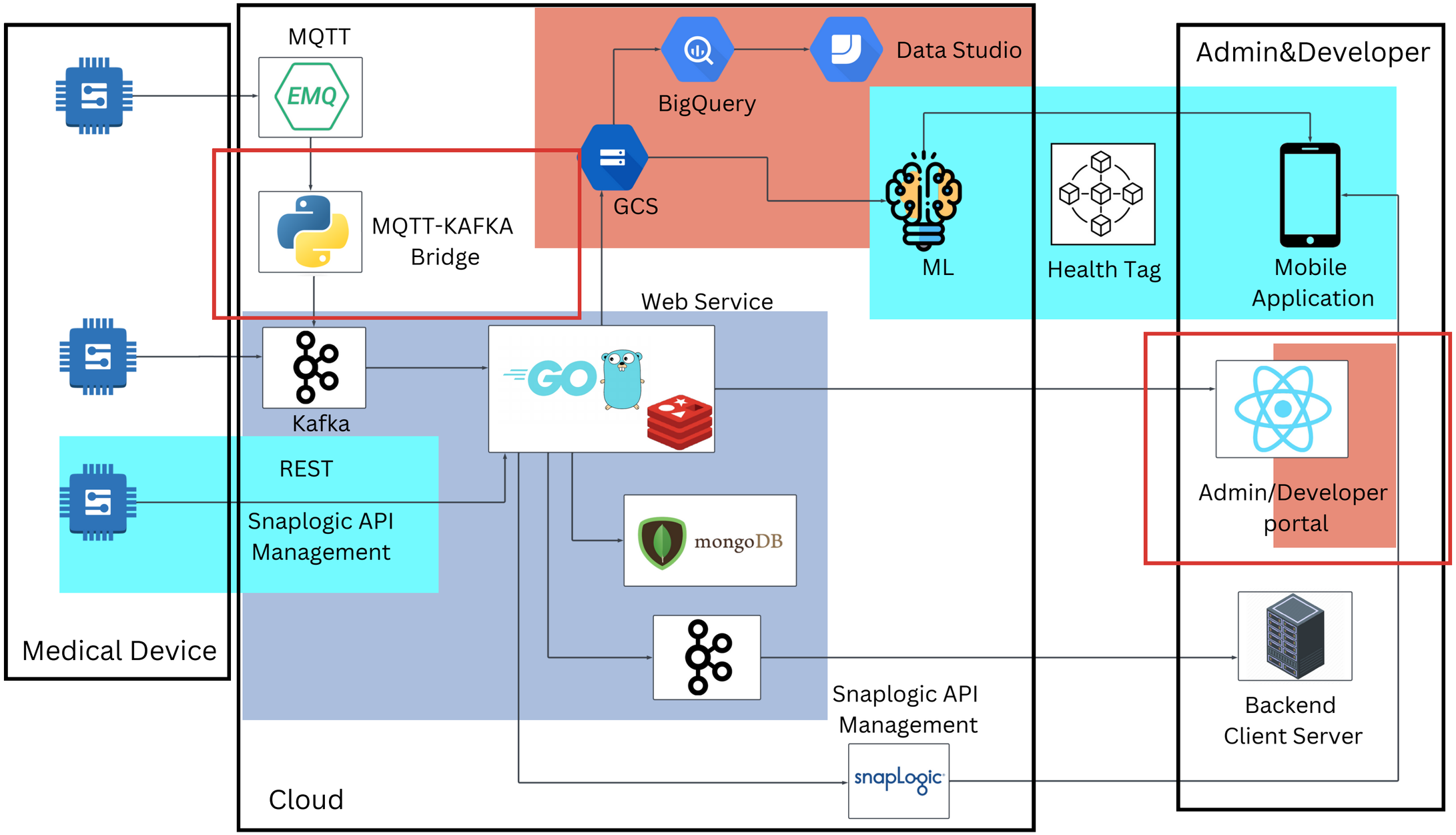
IV Architecture (Current)



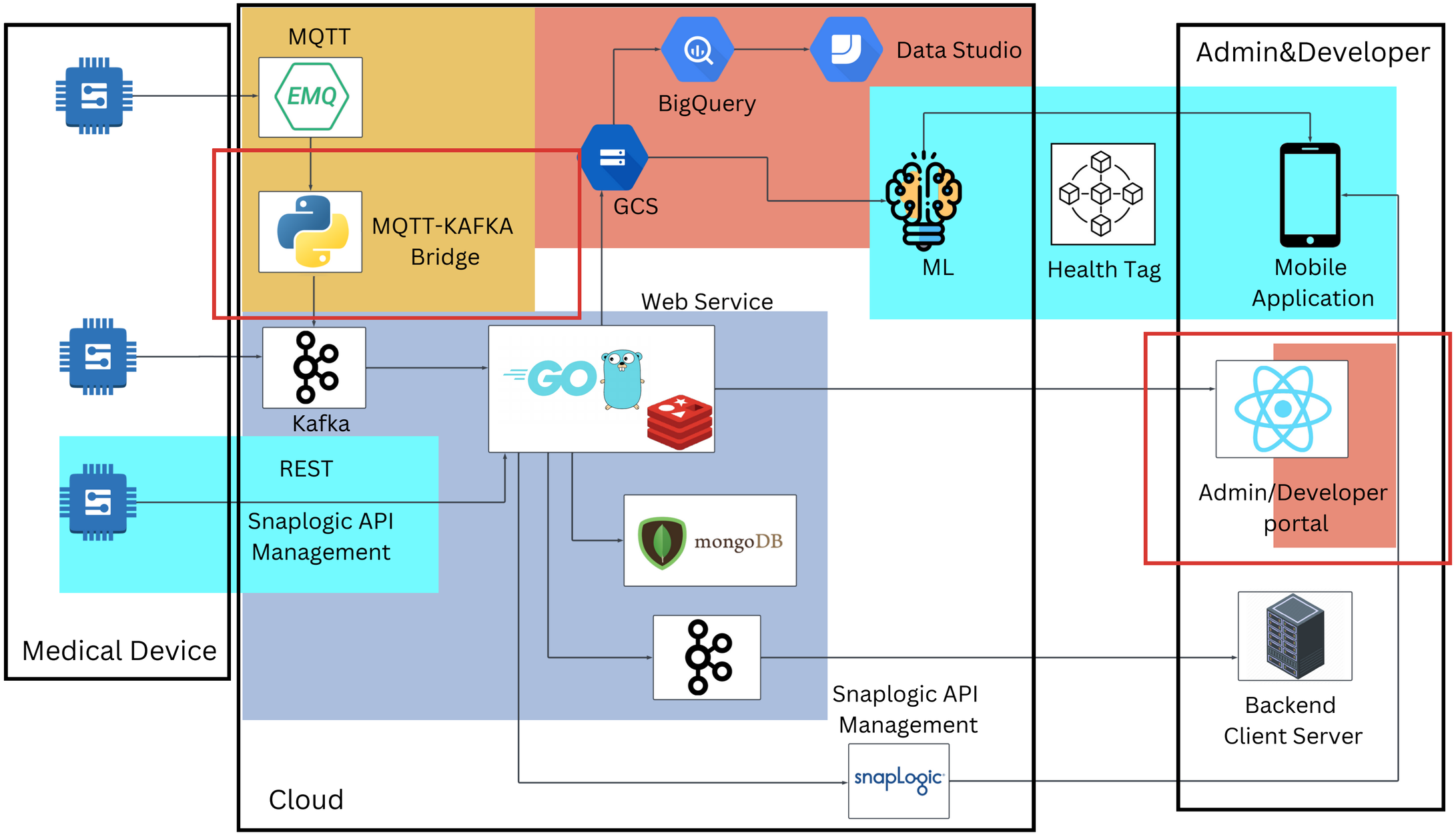
IV Architecture (Current)



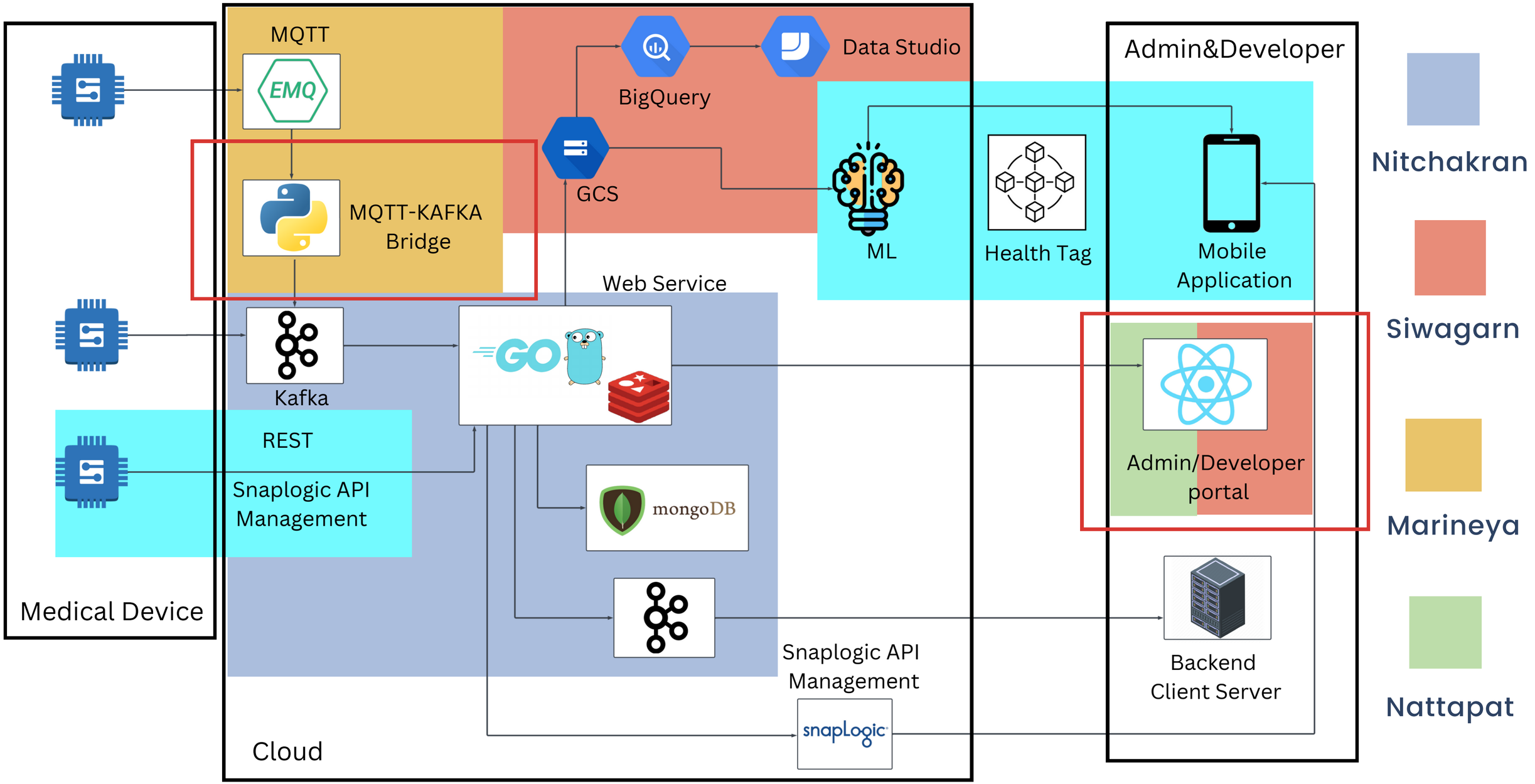
IV Architecture (Current)



IV Architecture (Current)



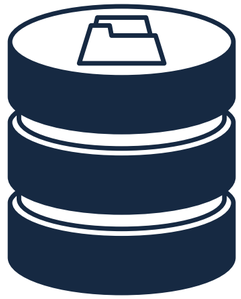
IV Architecture (Current)



Progression

Modules

Data Receiving



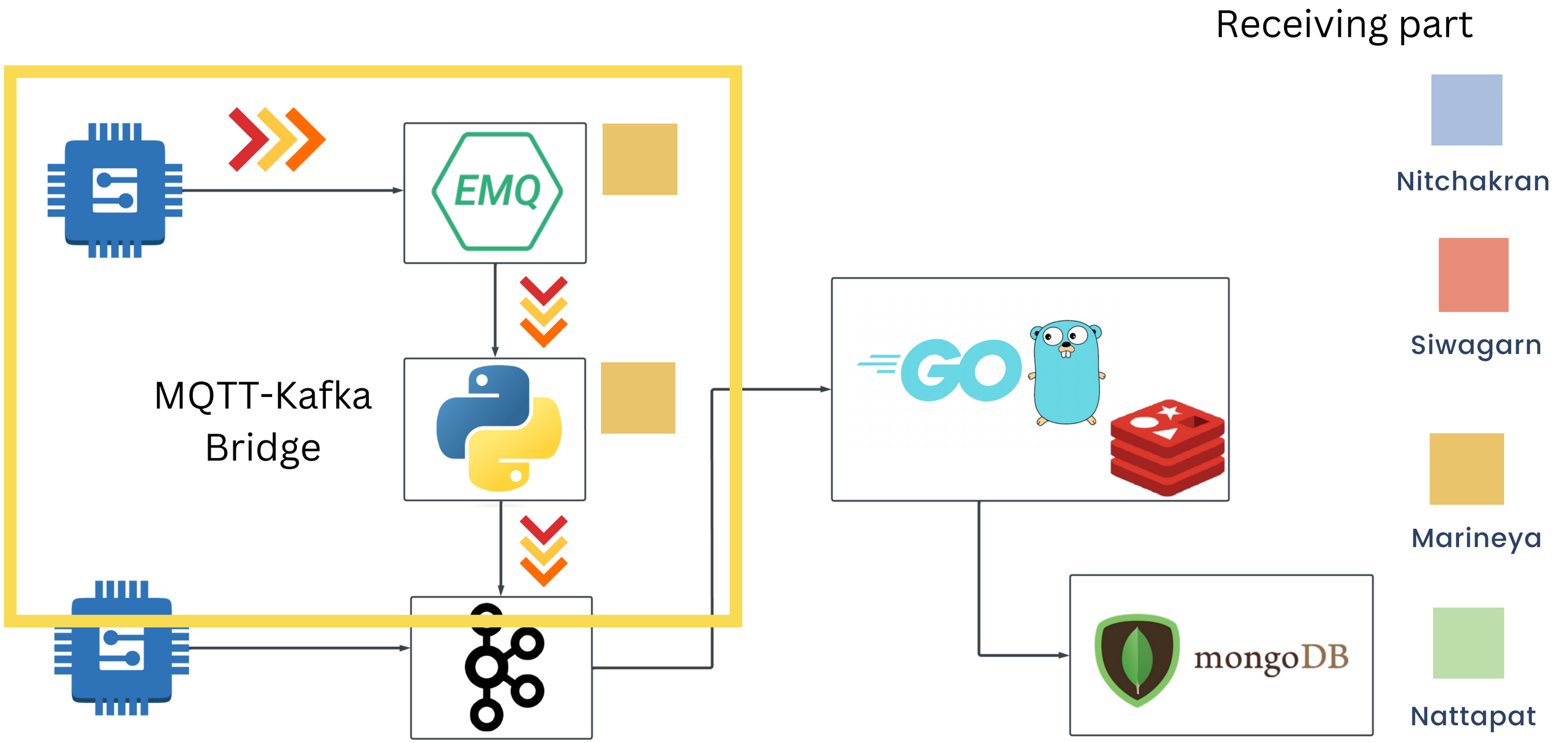
Data Management

Developer Service

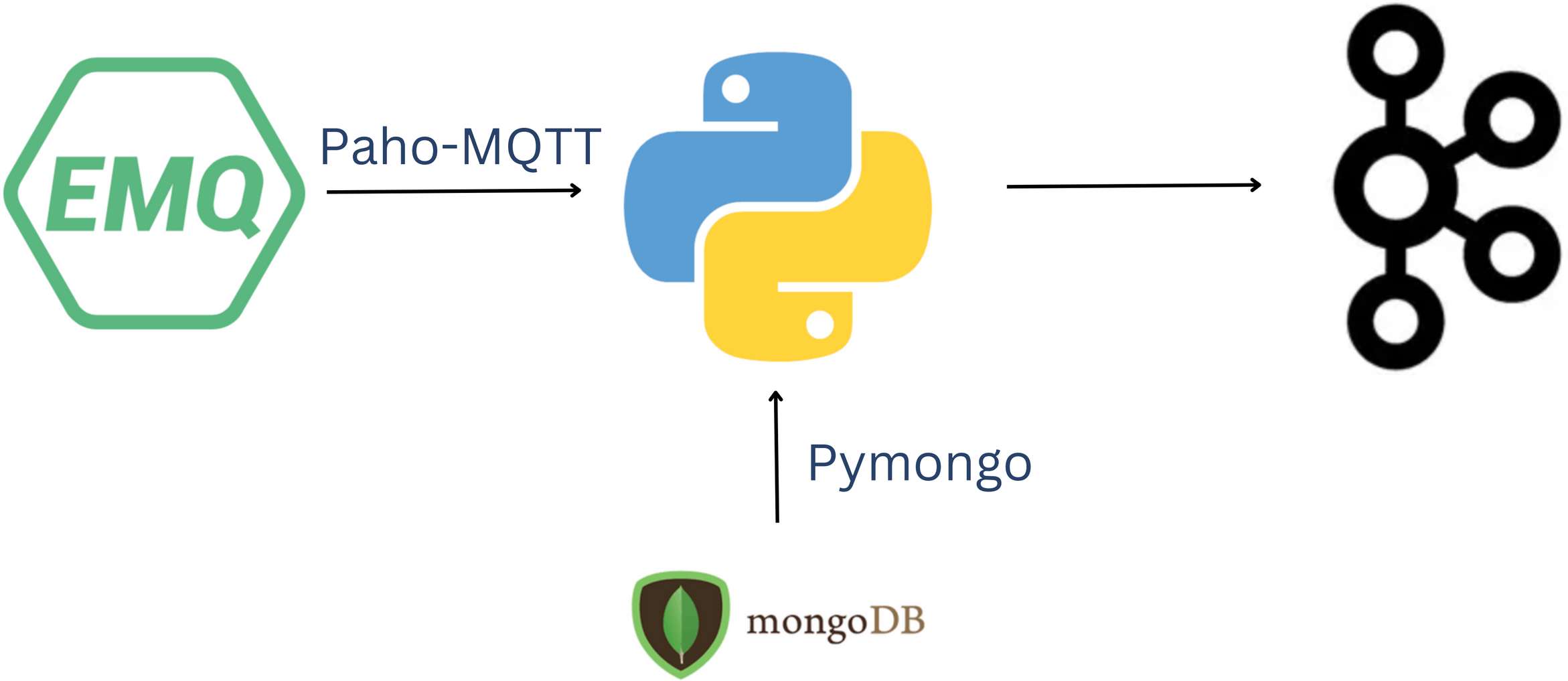
Admin Service

Logging

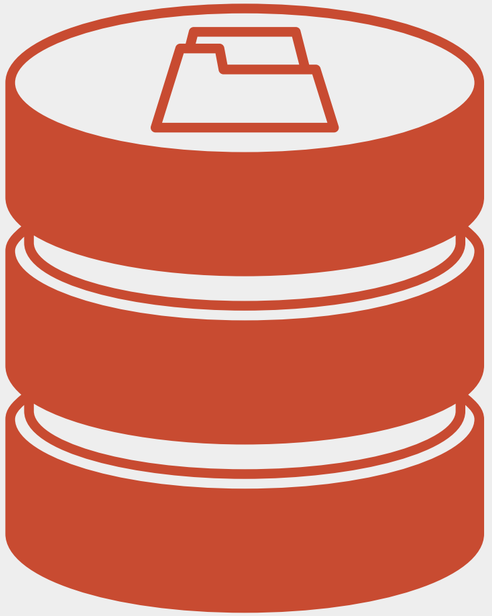
Data Receiving System


Receiving part
EMQ
MQTT-Kafka
Bridge
GO
mongoDB
Nitchakran
Siwagarn
Marineya
Nattapat

MQTT-Kafka Bridge



Data Management System

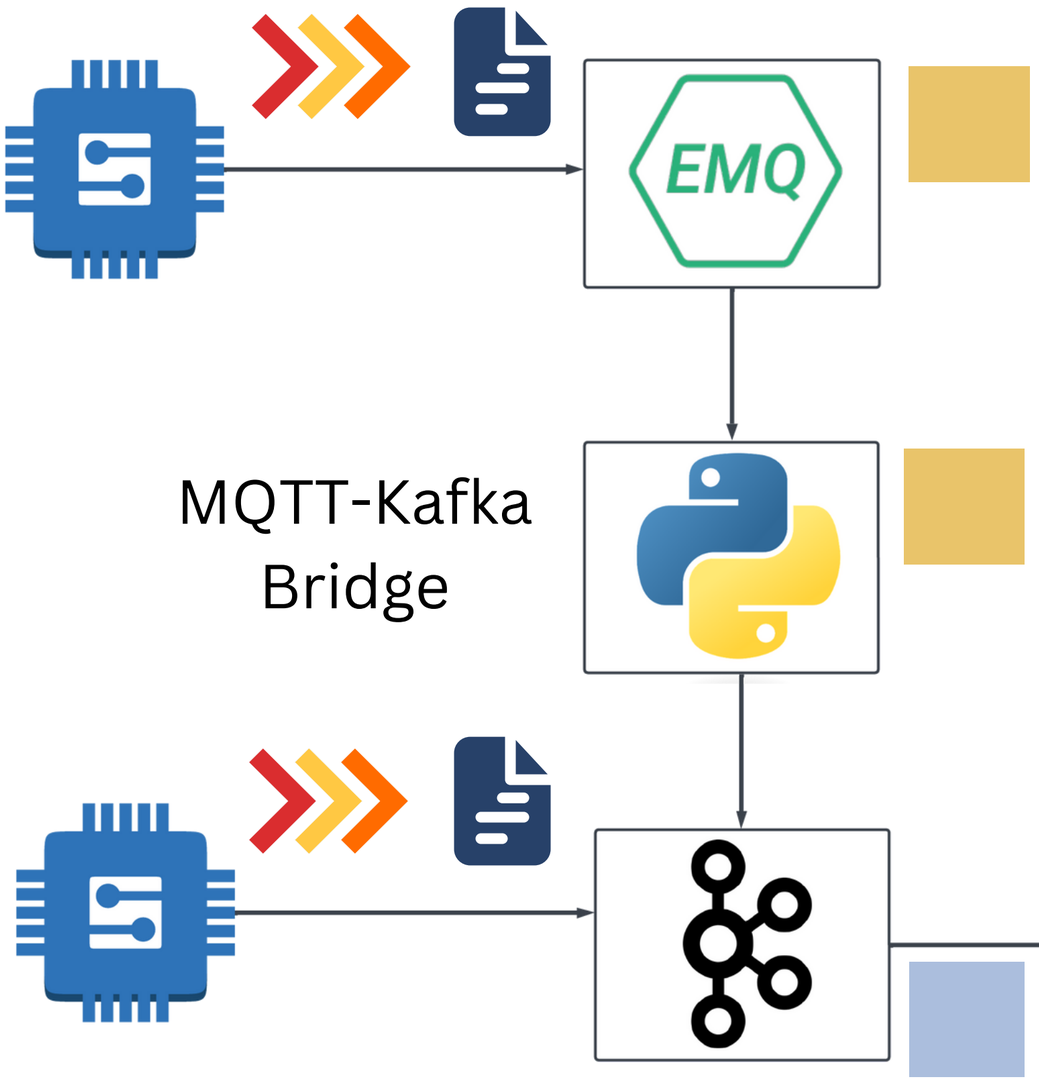


Receiving Part

Data Management System

Transmittion Part

Data flows



Receiving part

Web Service

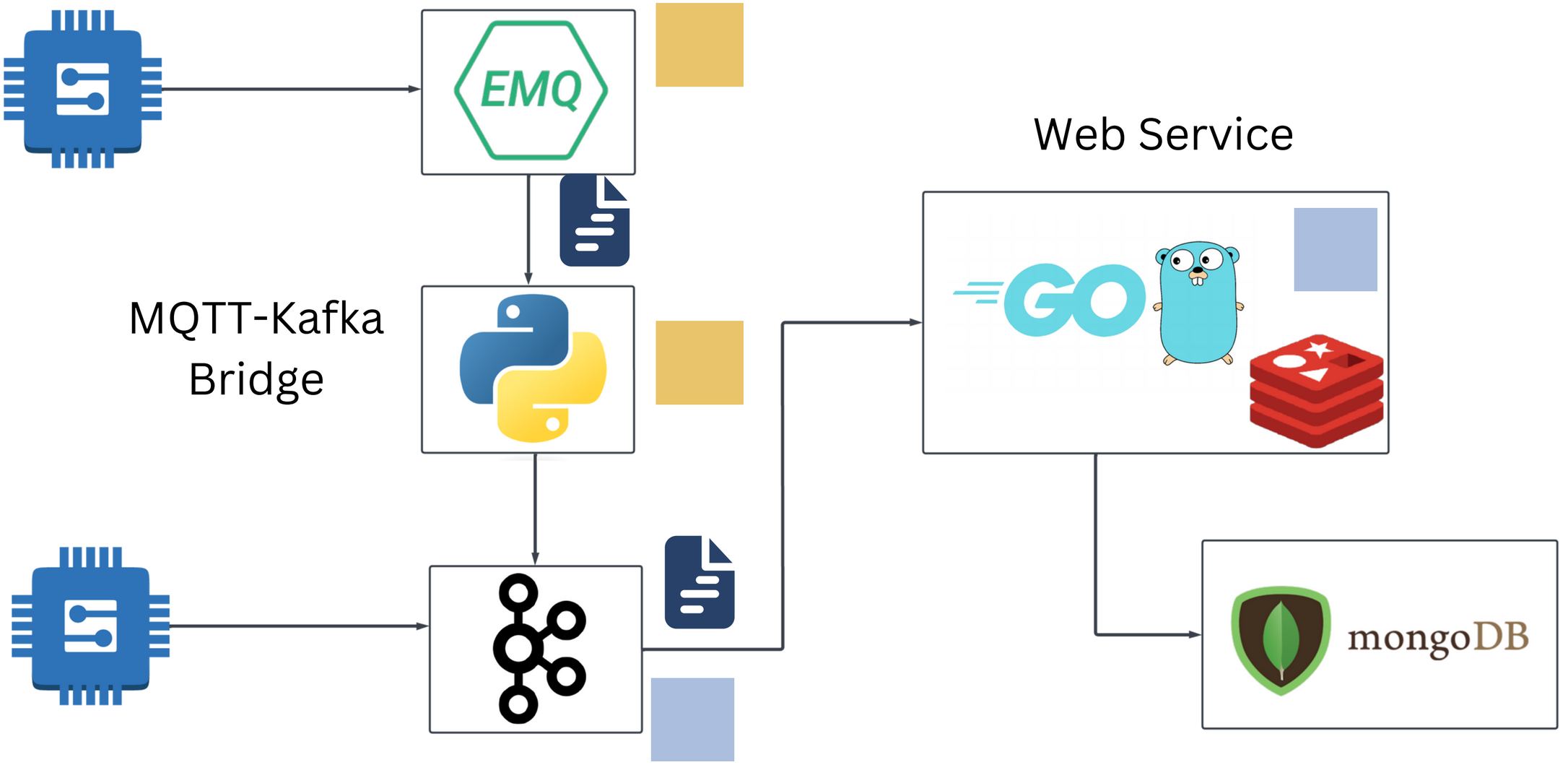
Nitchakran

Siwagarn

Marineya

Nattapat

Data flows



Receiving part

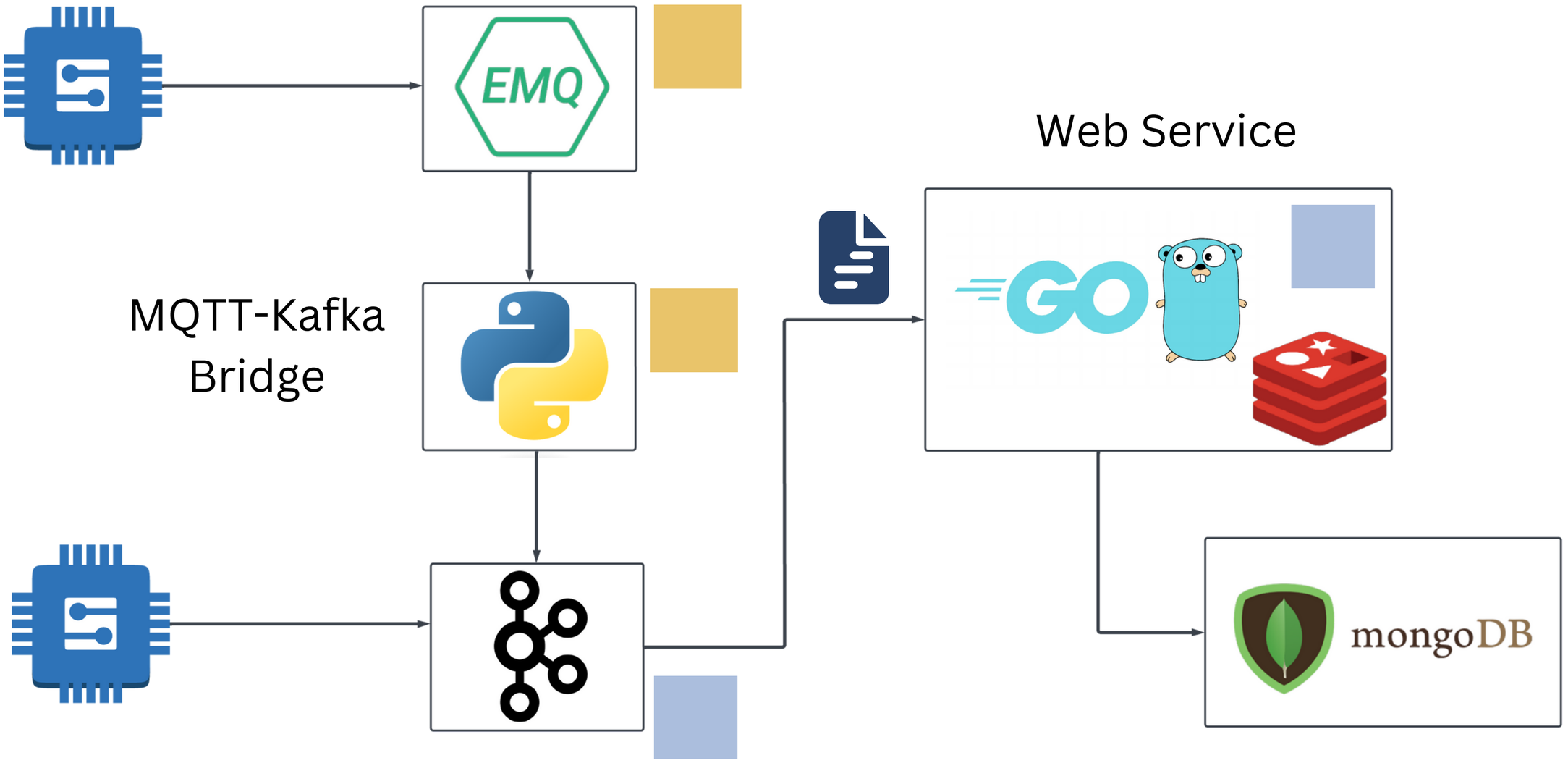
Nitchakran

Siwagarn

Marineya

Nattapat

Data flows



Receiving part

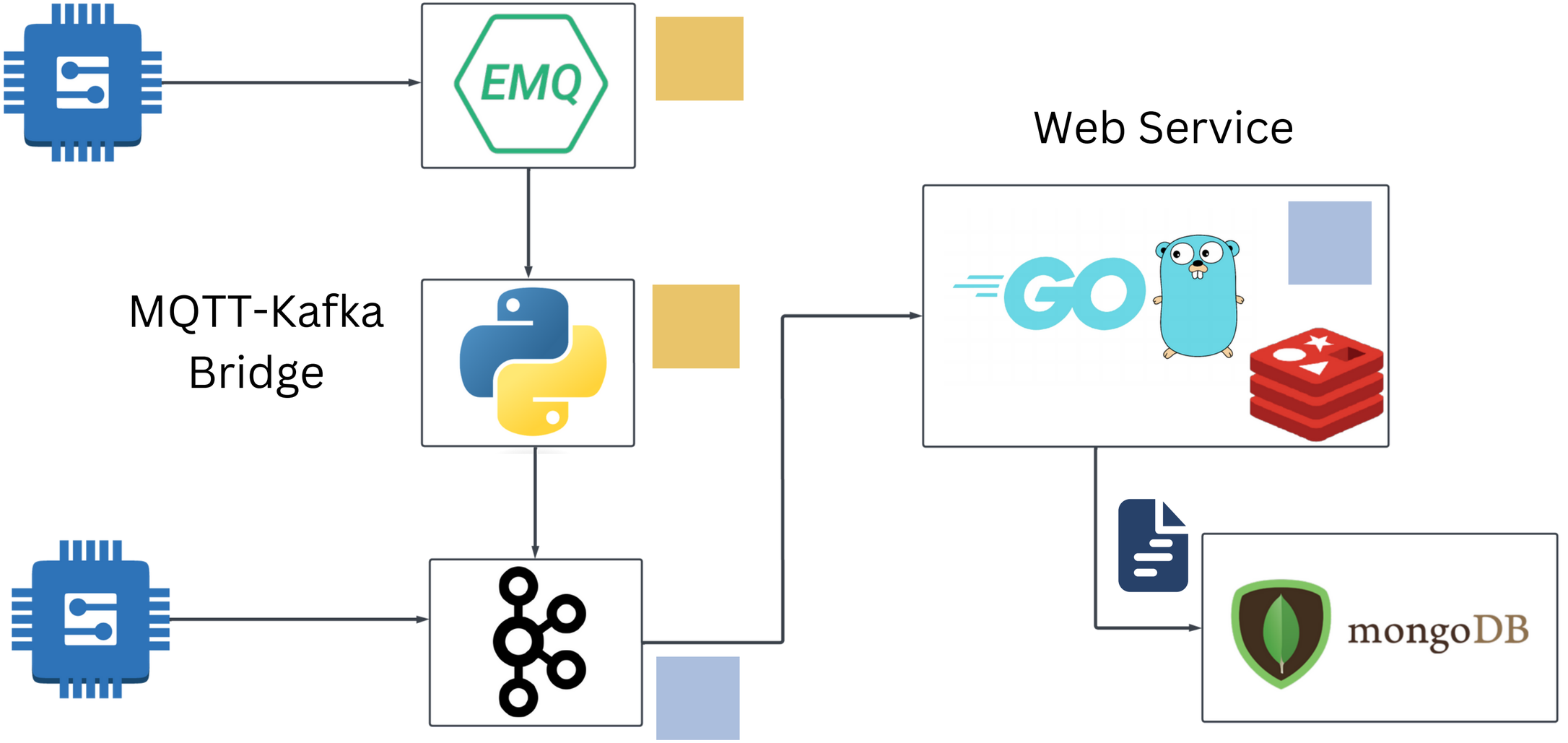
Nitchakran

Siwagarn

Marineya

Nattapat

Data flows



Receiving part

Nitchakran

Siwagarn

Marineya

Nattapat

Receiving part

1 collection ⟷ many models

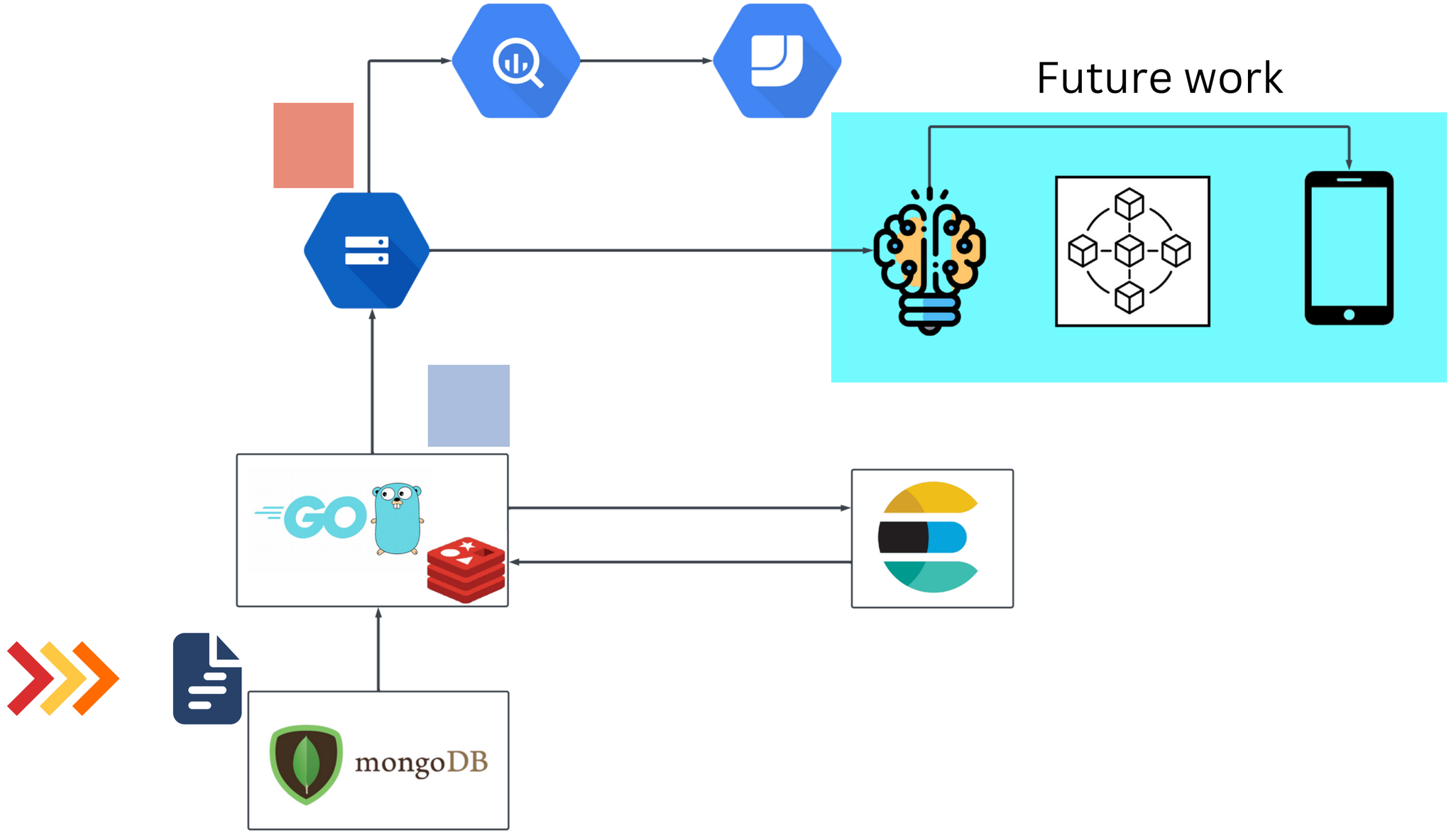
Identify device by deviceId and model name

Redis is used for keeping model structures

Web service —fetch→ model structures

Web service uses model structure for mapping data before stored data into MongoDB

Data flows



Transmition part

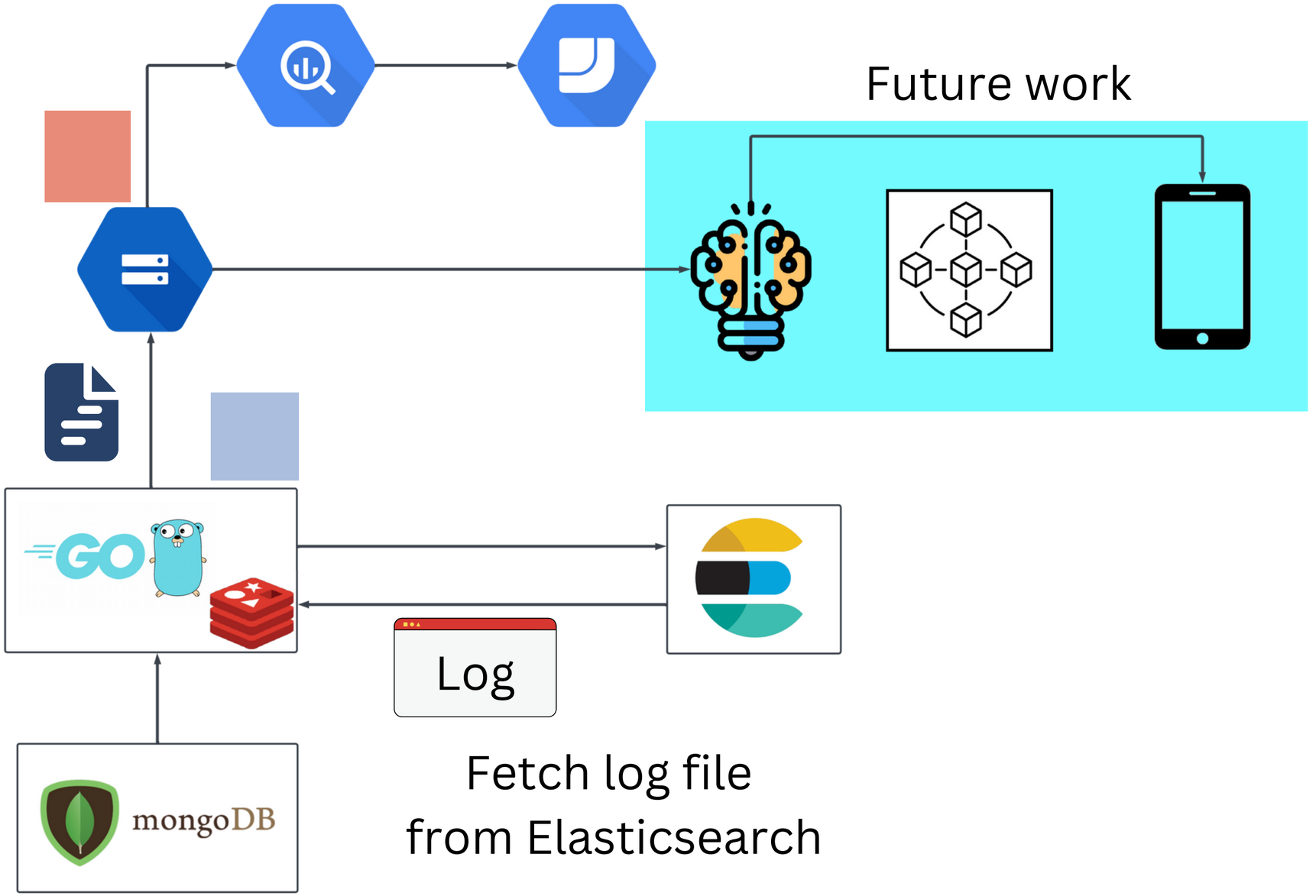
Nitchakran

Siwagarn

Marineya

Nattapat

Data flows



Transmition part

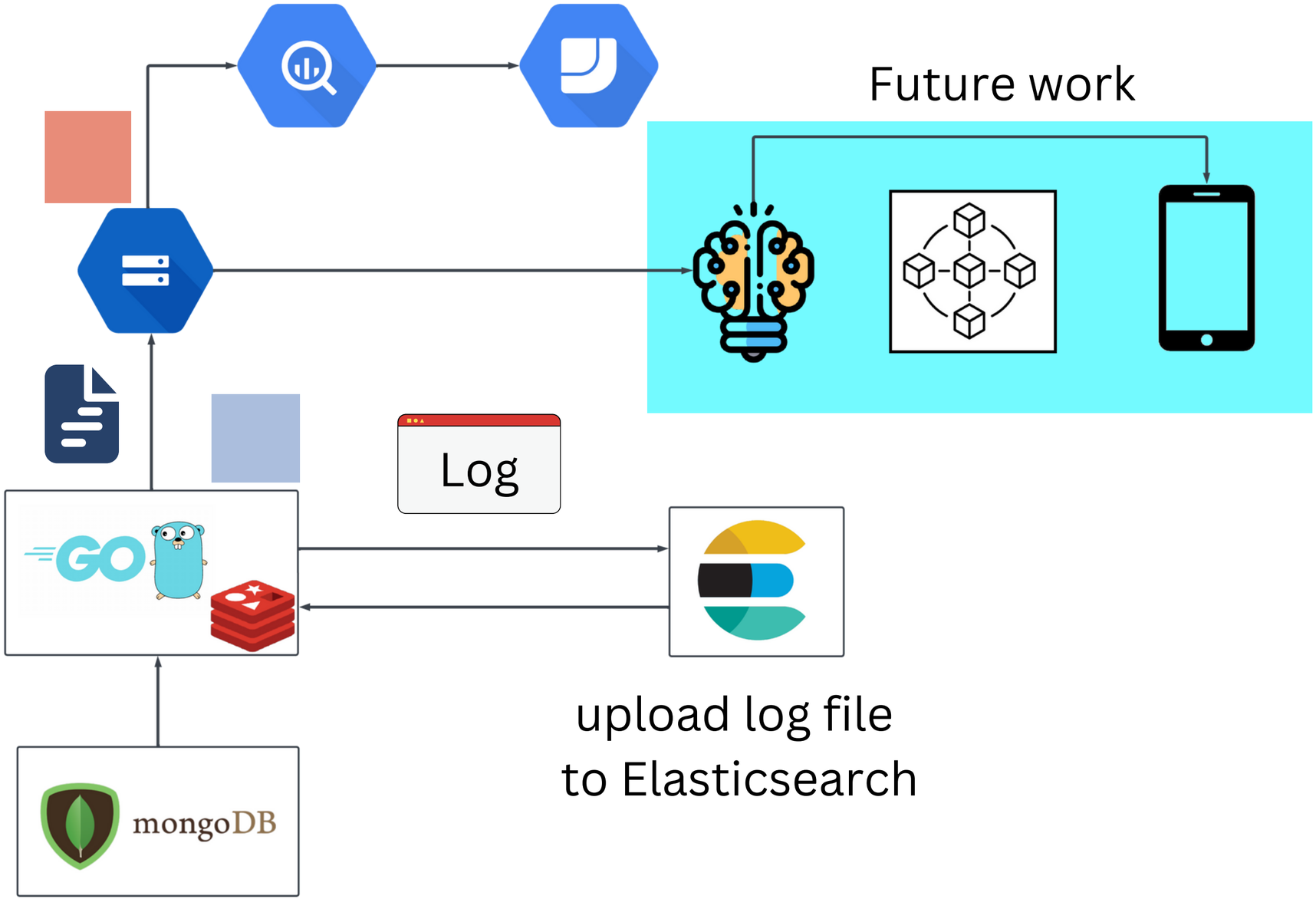
Nitchakran

Siwagarn

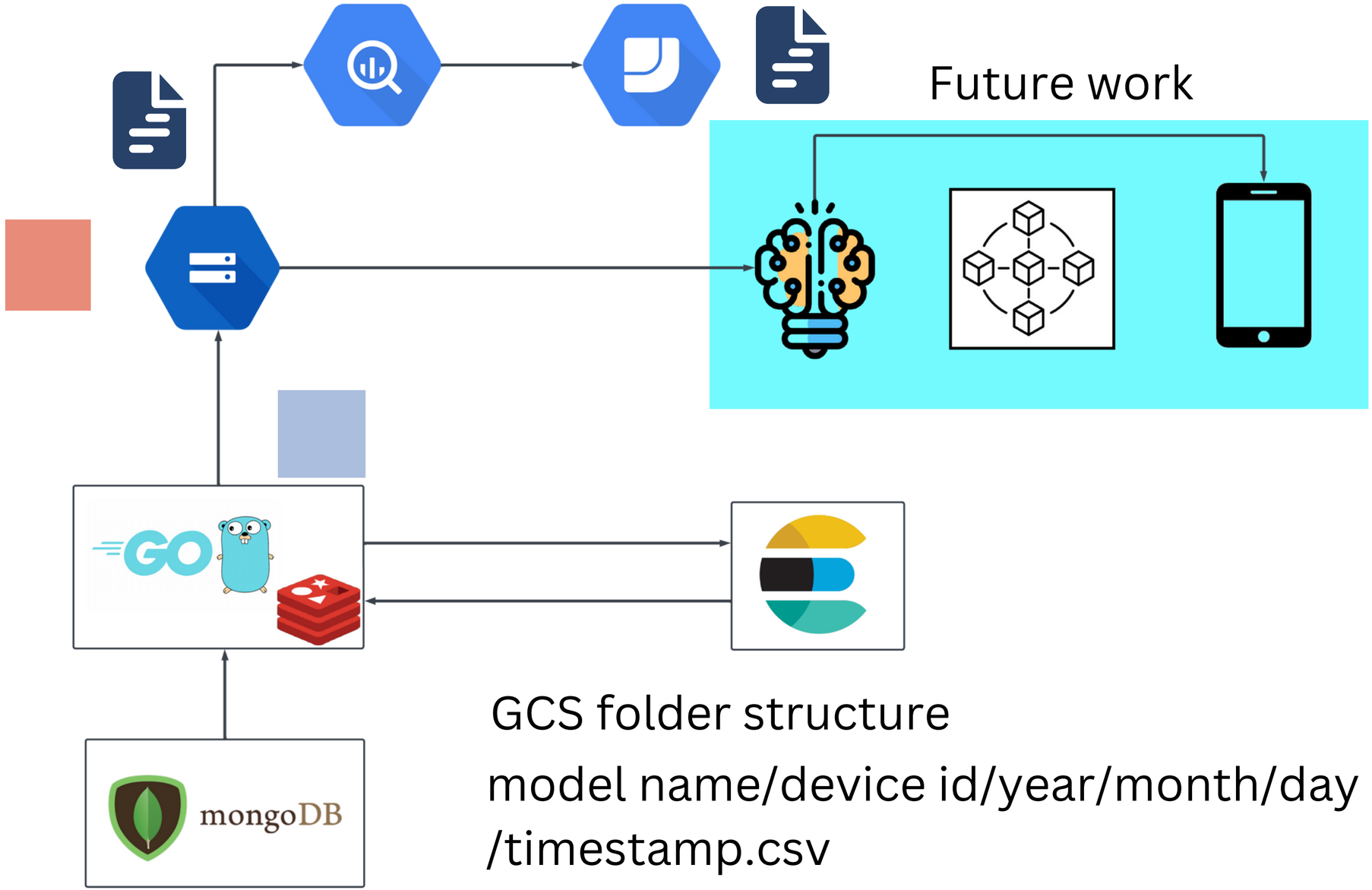
Marineya

Nattapat

Data flows


Transmition part
Future work
Log
upload log file
to Elasticsearch
mongoDB
Nitchakran
Siwagarn
Marineya
Nattapat

Data flows



Transmition part

Nitchakran

Siwagarn

Marineya

Nattapat

GCS folder structure

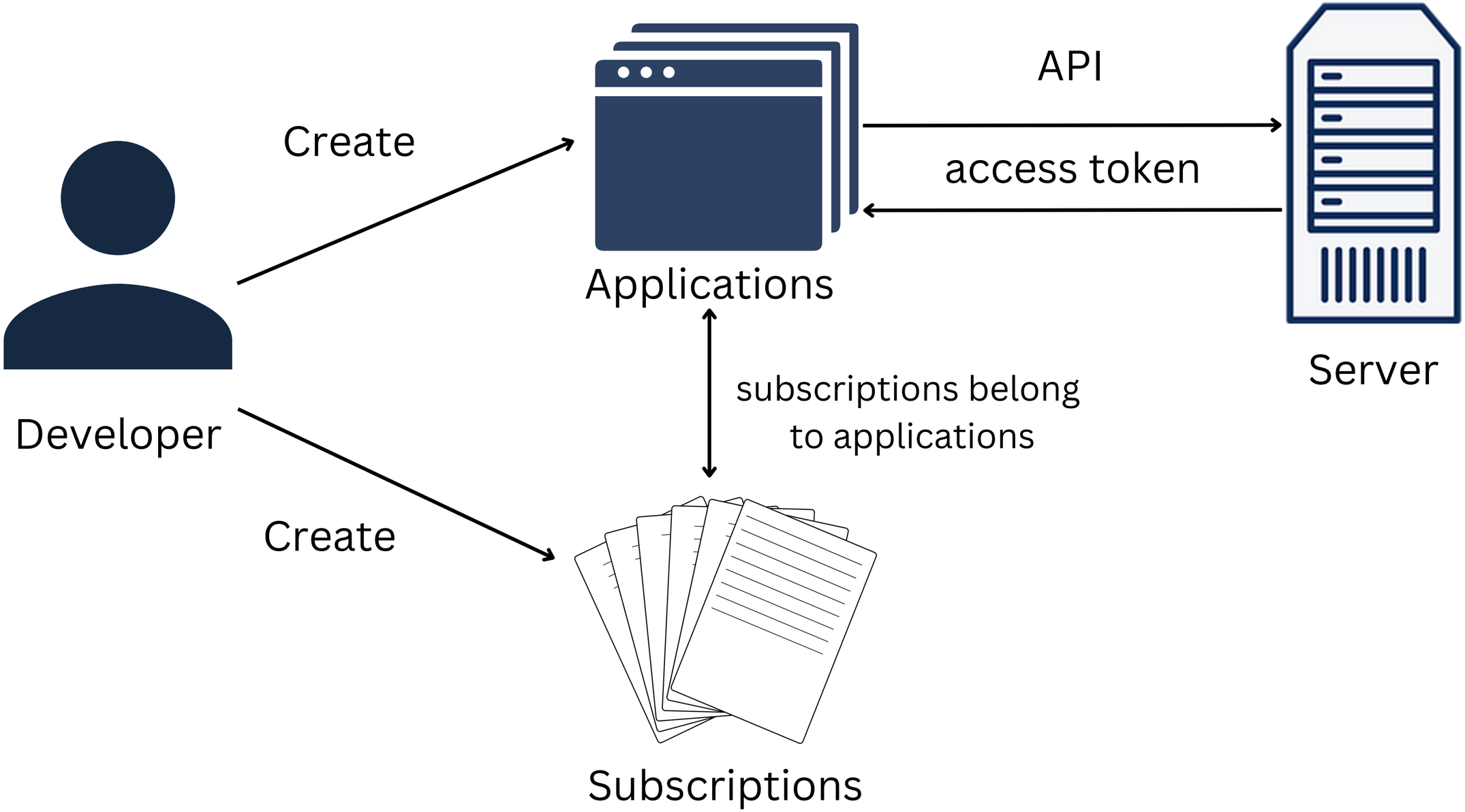
model name/device id/year/month/day /timestamp.csv

Elasticsearch

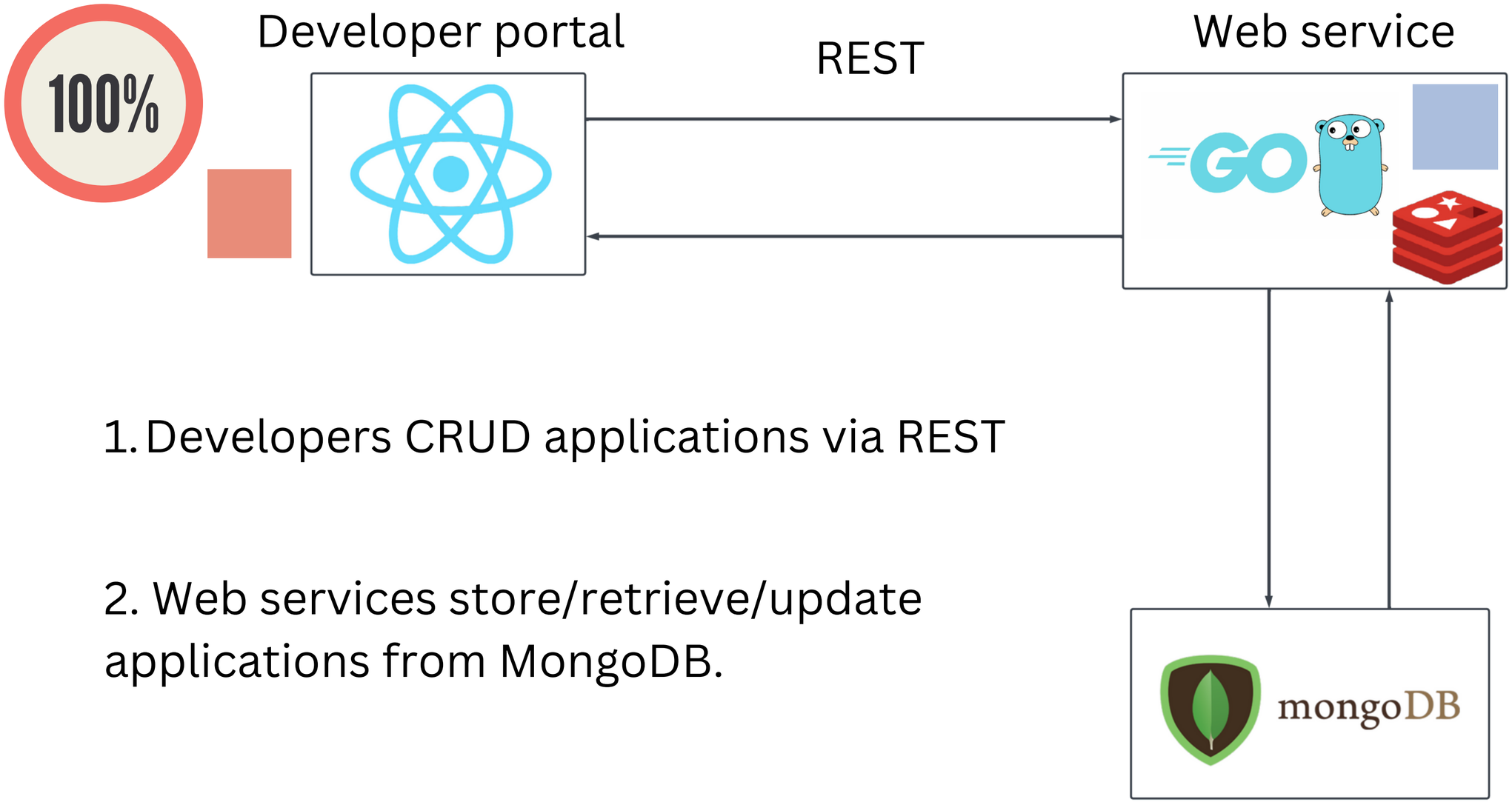
```
type UploadMedicalDataLog struct
    {
      ModelName string
      DeviceId string
      StartTime int64
      EndTime int64
      CreatedDate int64
      IsCompleted bool
    }
```

Start time for the next batch is based on the End time of the previous batch.

Developer Service System


API
Create
access token
Applications
Server
subscriptions belong
to applications
Developer
Create
Subscriptions

Developer CRUD applications



1. Developers CRUD applications via REST

2. Web services store/retrieve/update applications from MongoDB.

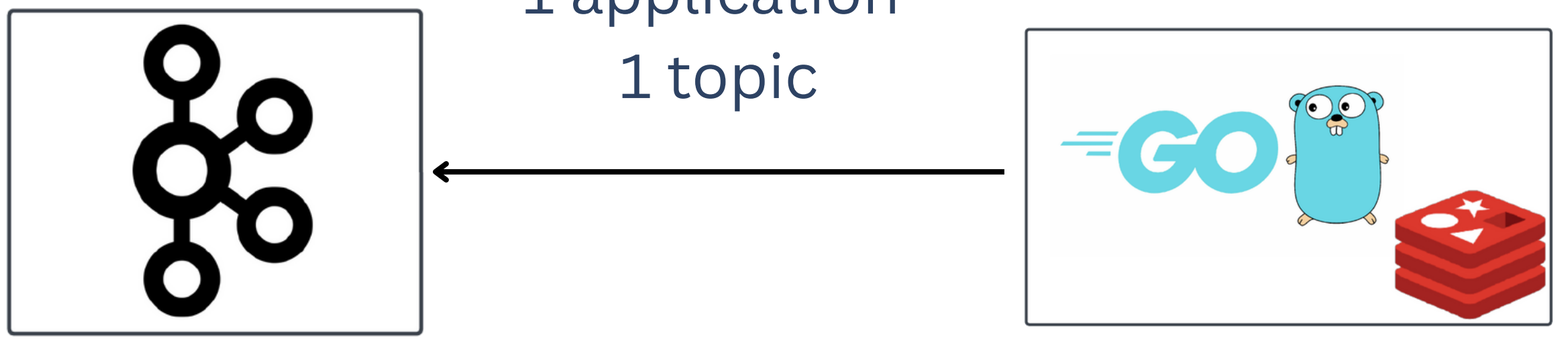
Nitchakran

Siwagarn

Marineya

Nattapat

Developer CRUD applications



3. Web services creates a new topic for the application.

4. Web services creates new consumer's username and alter permission for the application

Nitchakran

Siwagarn

Marineya

Nattapat

Developer create, read and update subscriptions

100%

1. Developer create, read and update subscriptions via REST

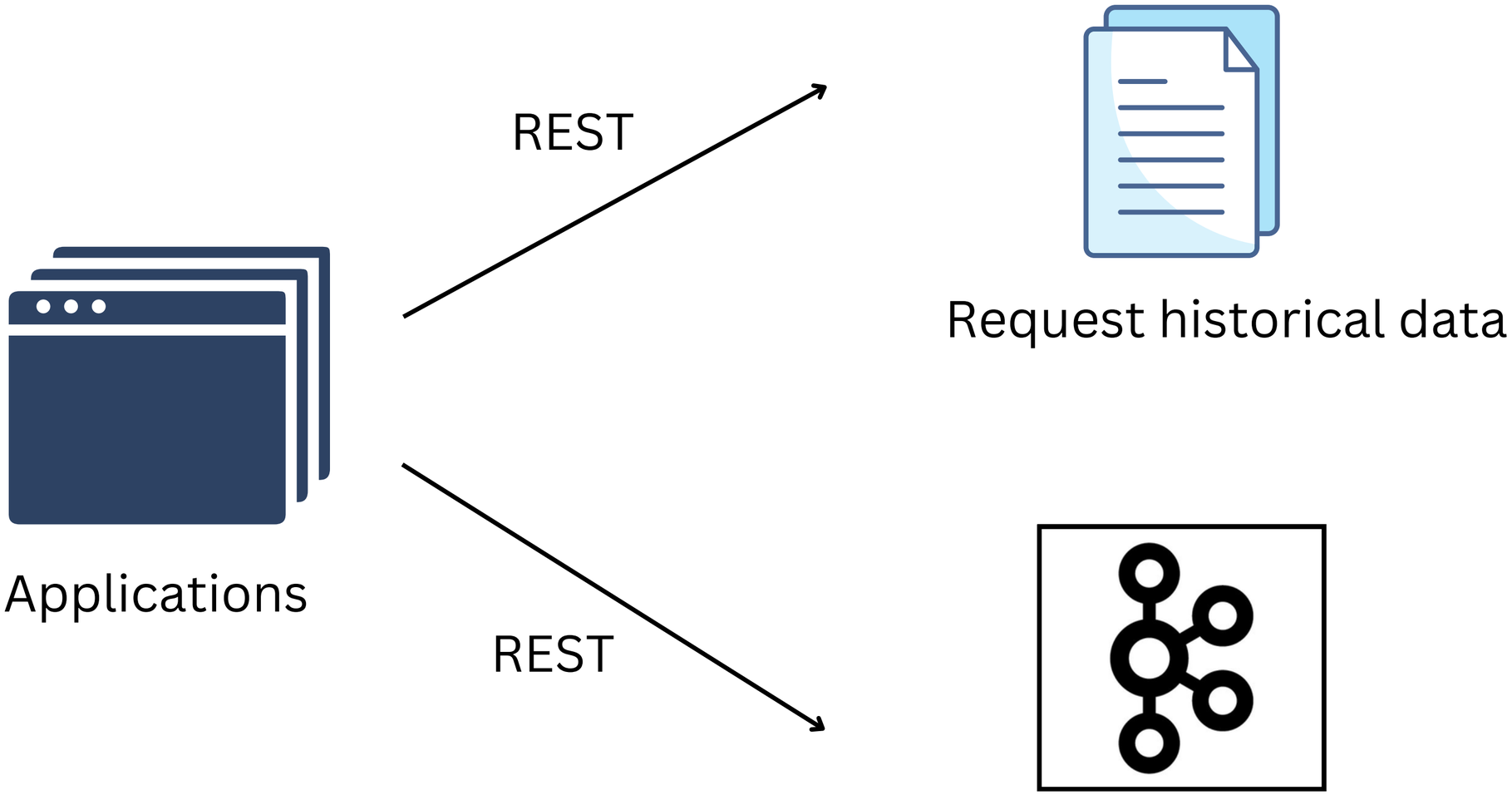
2. Web service store/retrieve/update subscription from MongoDB

Nitchakran

Siwagarn

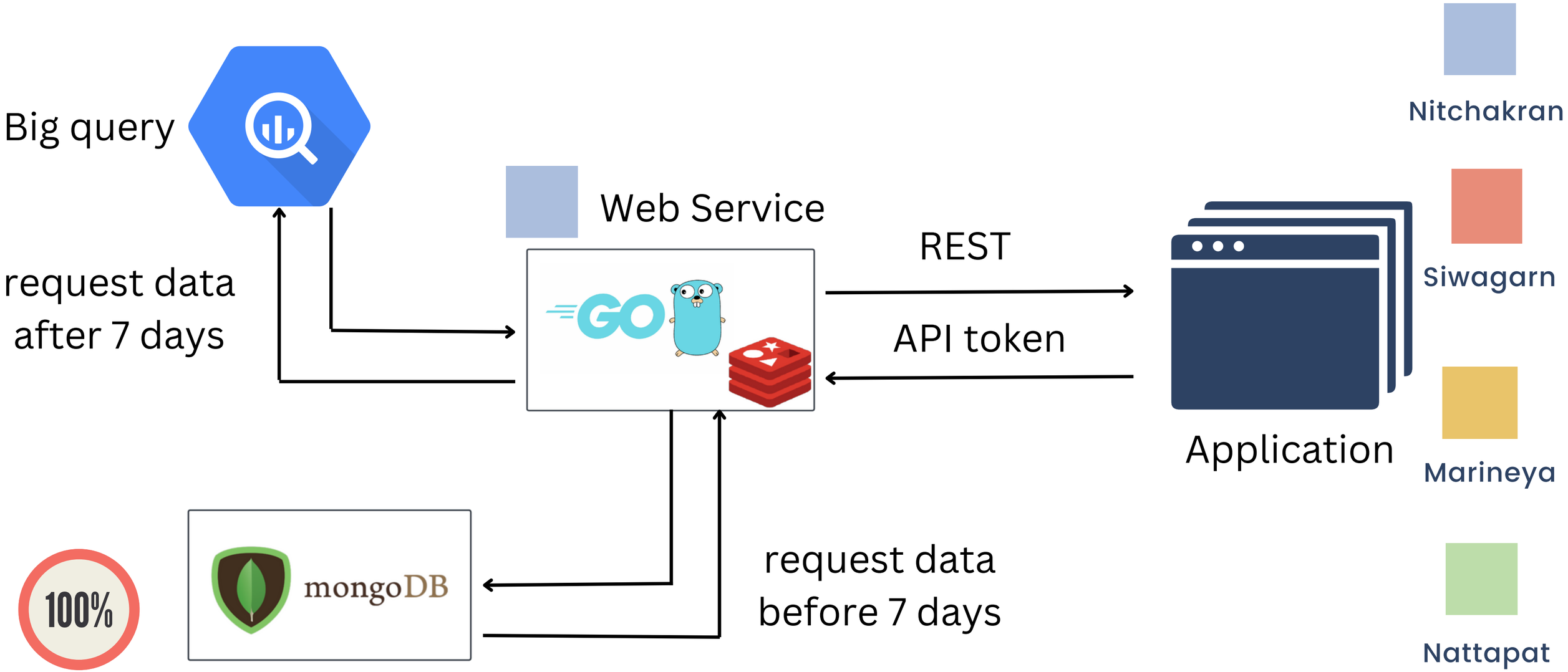
Marineya

Nattapat


REST
Request historical data
Applications
REST
Request real-time data

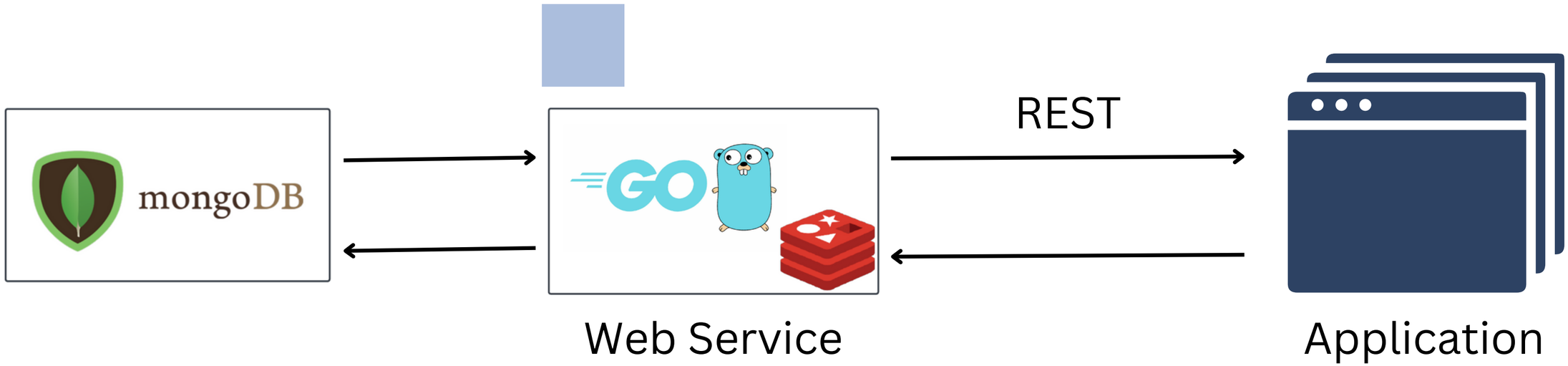
Request historical data

update implementation logic



Request real-time data

update implementation logic



Nitchakran

Siwagarn

Marineya

Nattapat

100%

Request real-time data

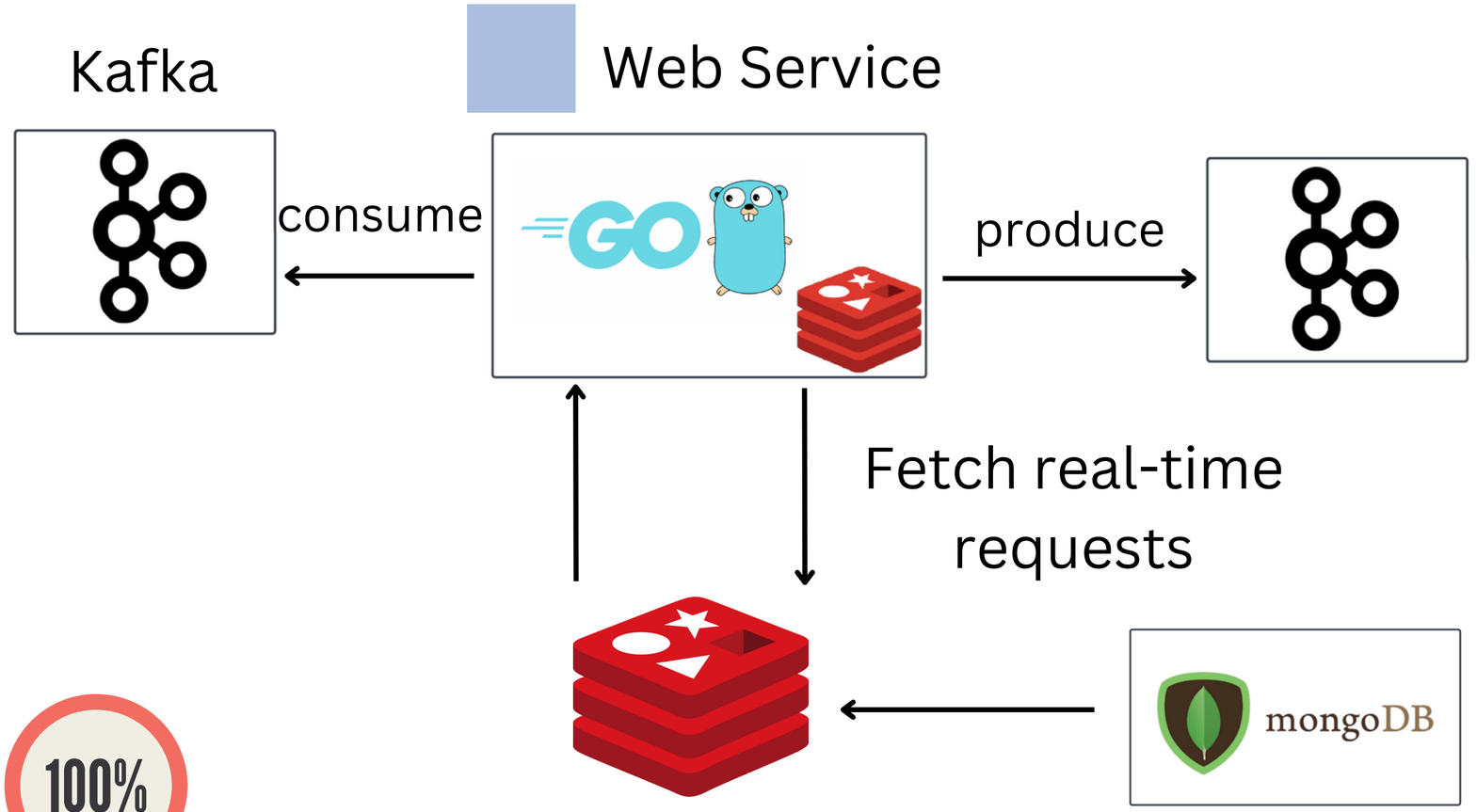
update implementation logic

Nitchakran

Siwagarn

Marineya

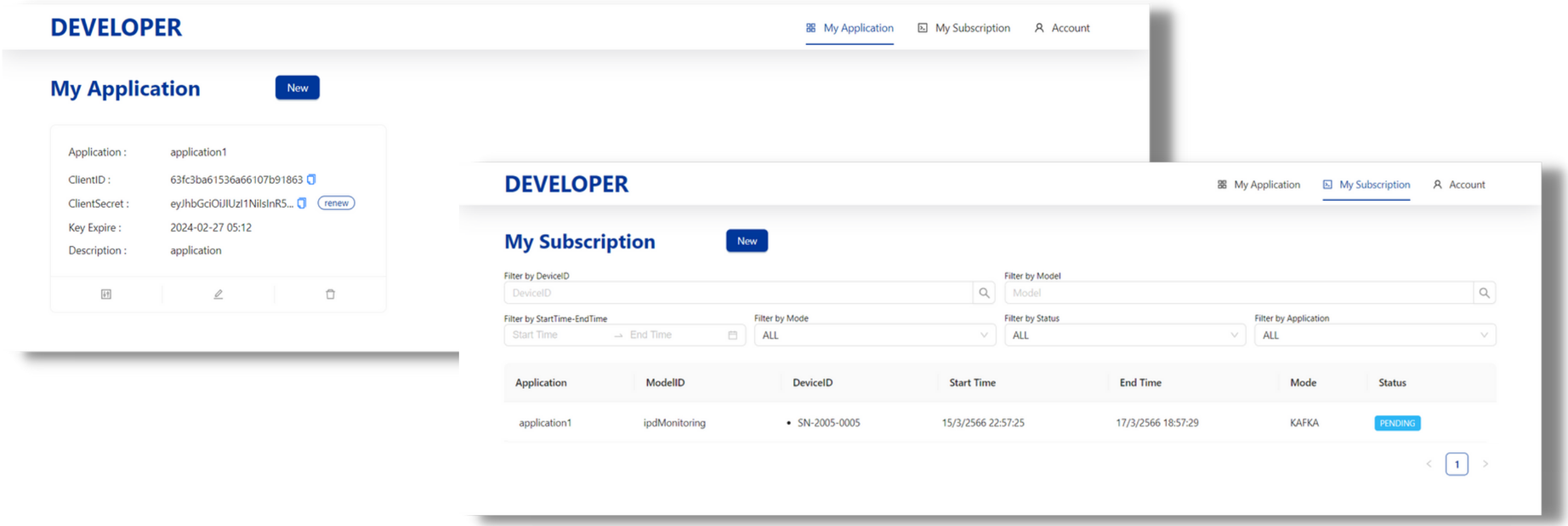
Nattapat



100%

Dashboard

- CRUD application
- Create and read subscriptions
- Account



100%

Nitchakran

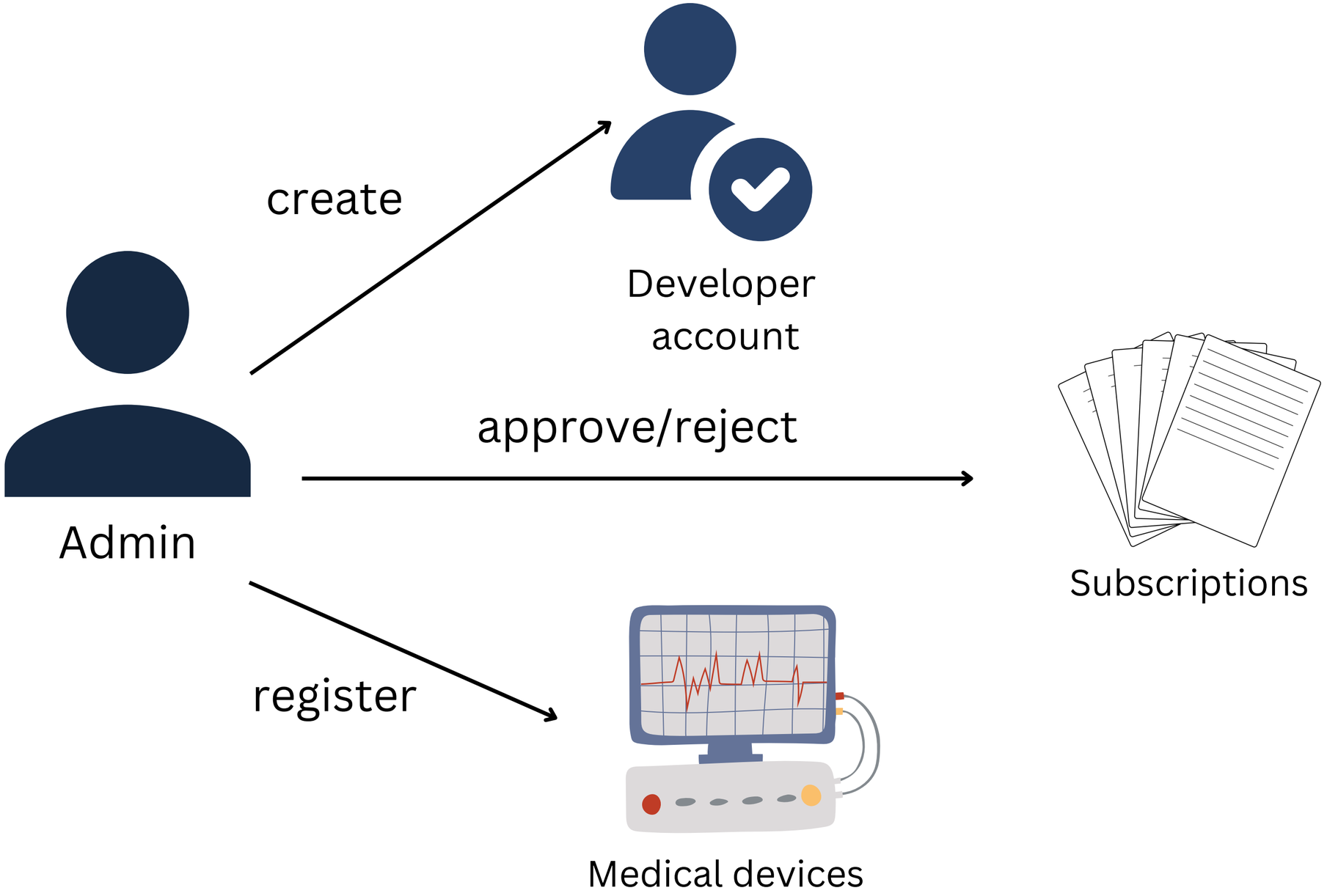
Siwagarn

Marineya

Nattapat

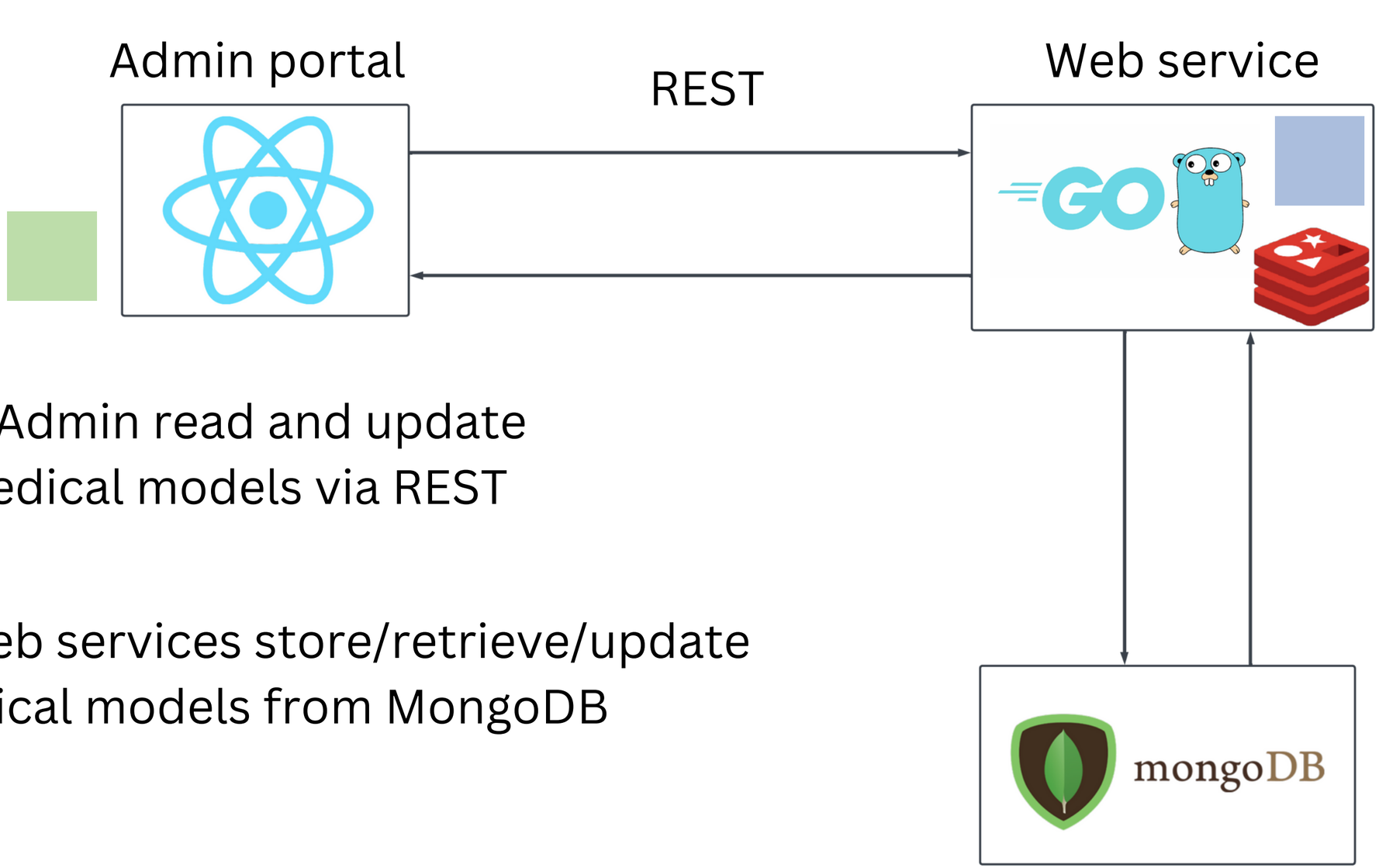
Admin Service System

Progression: Admin Service System



Admin CRUD medical models

100%



1. Admin read and update medical models via REST

2. Web services store/retrieve/update medical models from MongoDB

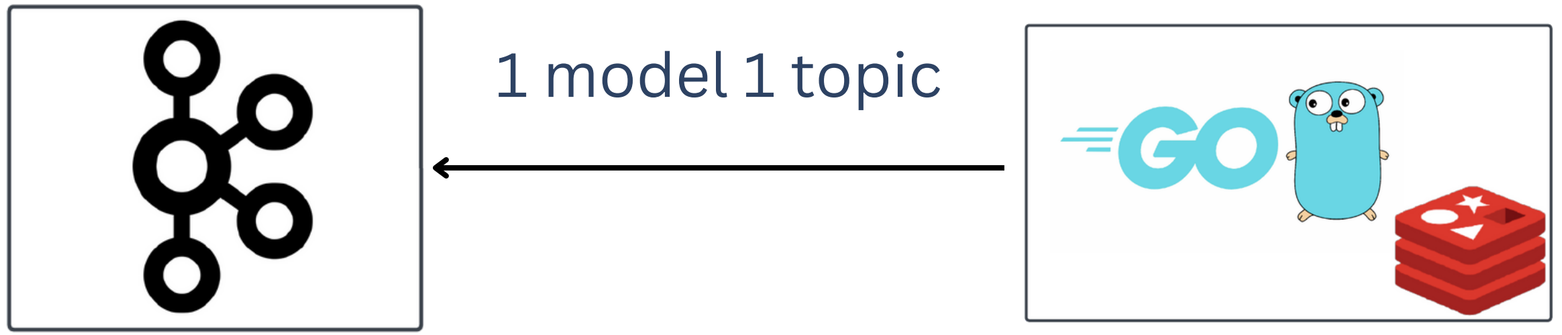
Nitchakran

Siwagarn

Marineya

Nattapat

Admin CRUD medical models



3. Web services create the new topic for the medical device

4. Web services stop the routine and restart to subscribe the new topic

Nitchakran

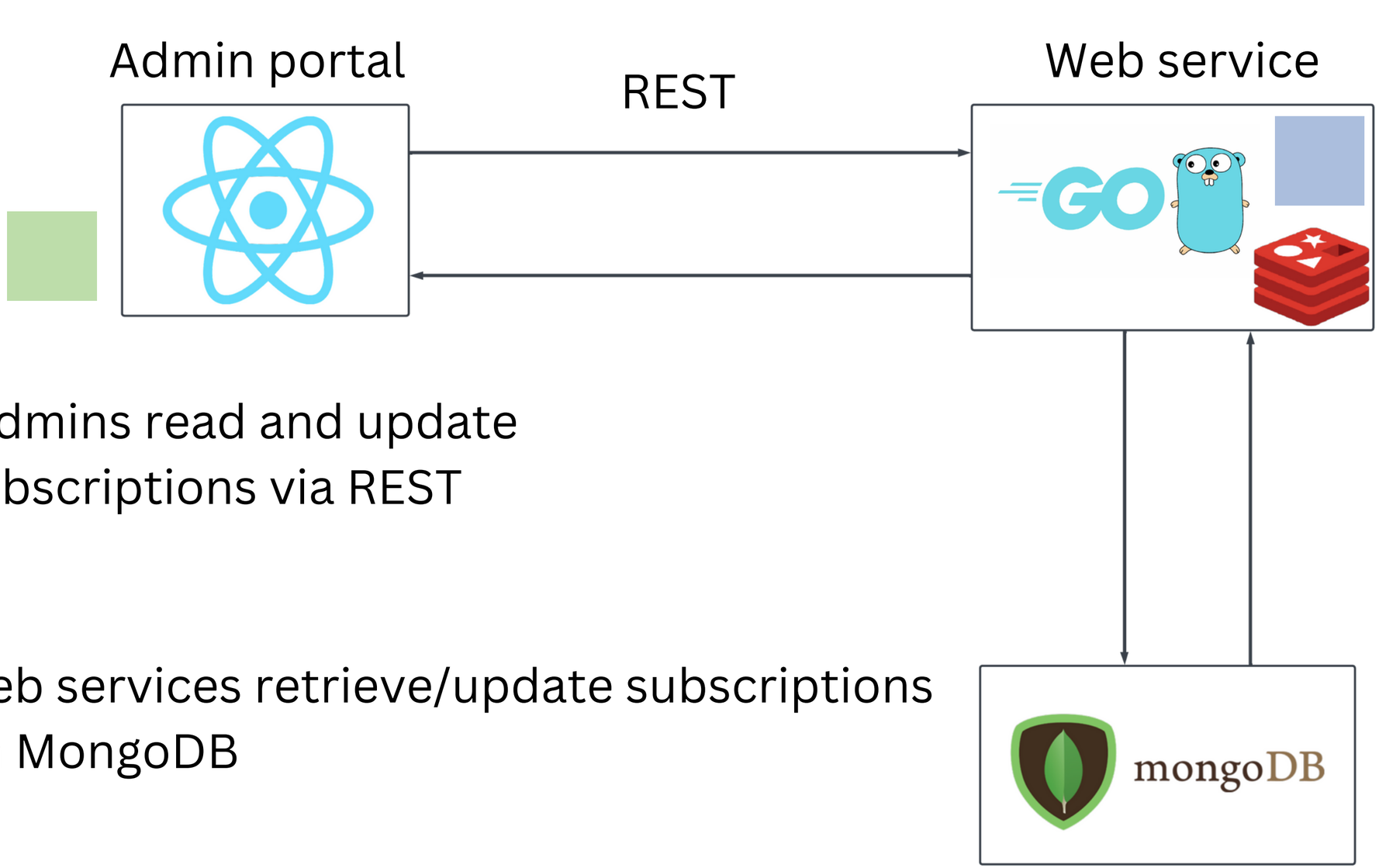
Siwagarn

Marineya

Nattapat

Admin read and update subscriptions

100%



1. Admins read and update subscriptions via REST

2. Web services retrieve/update subscriptions from MongoDB

Nitchakran

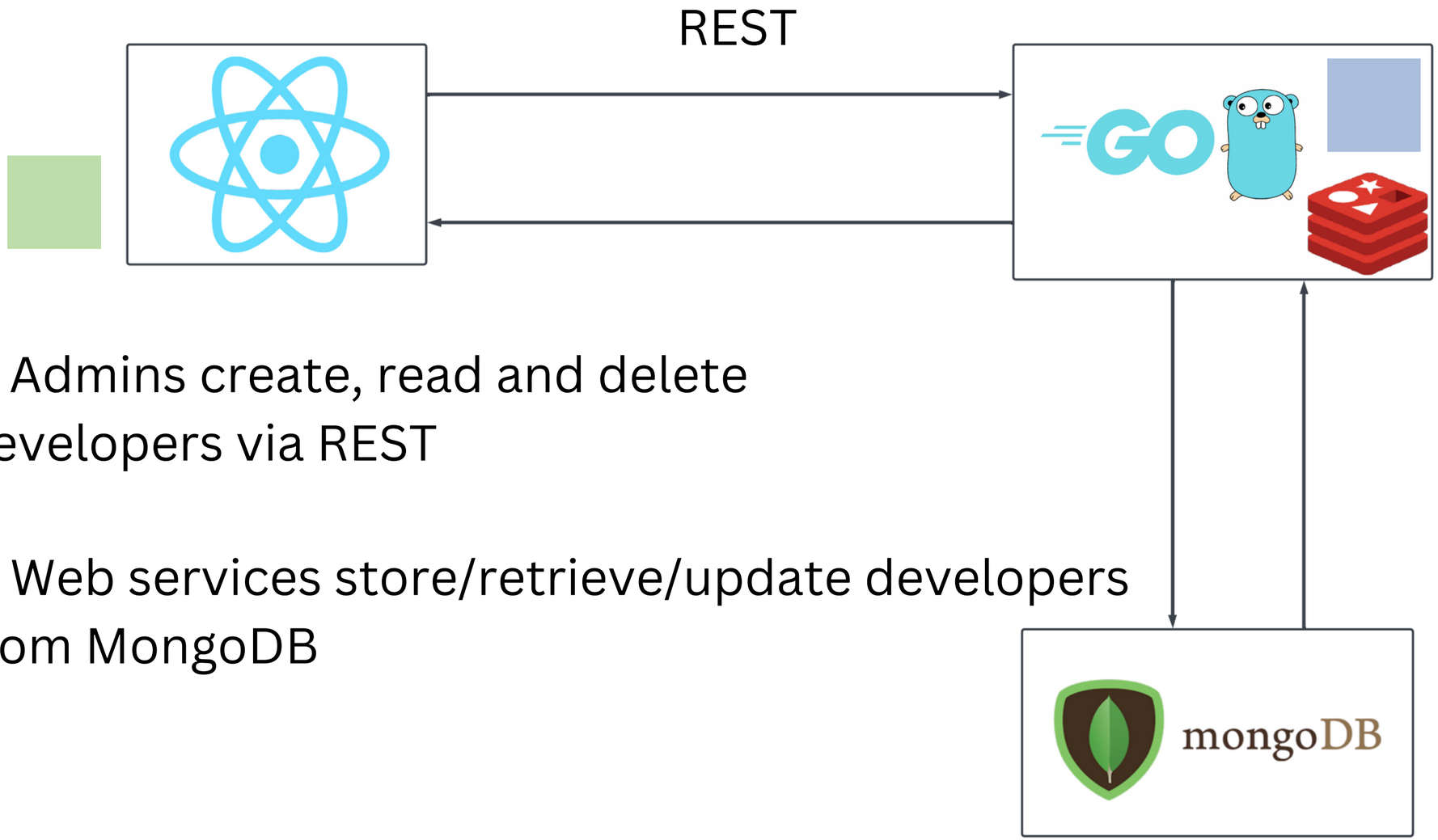
Siwagarn

Marineya

Nattapat

Admin CRD developers

100%



1. Admins create, read and delete developers via REST

2. Web services store/retrieve/update developers from MongoDB

Nitchakran

Siwagarn

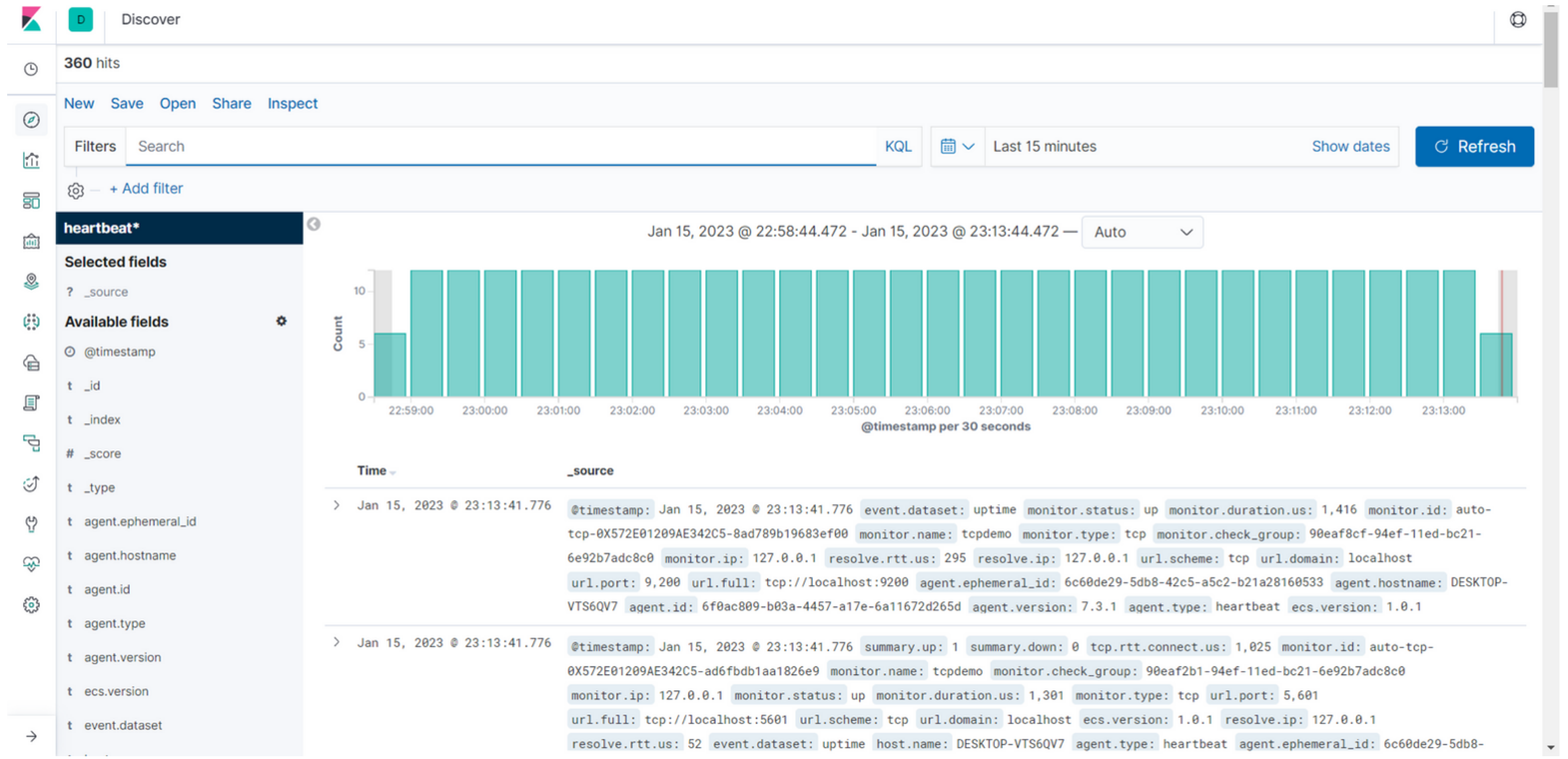
Marineya

Nattapat

Logging

Monitor system's service: Uptime

Heartbeat



100%

Nitchakran

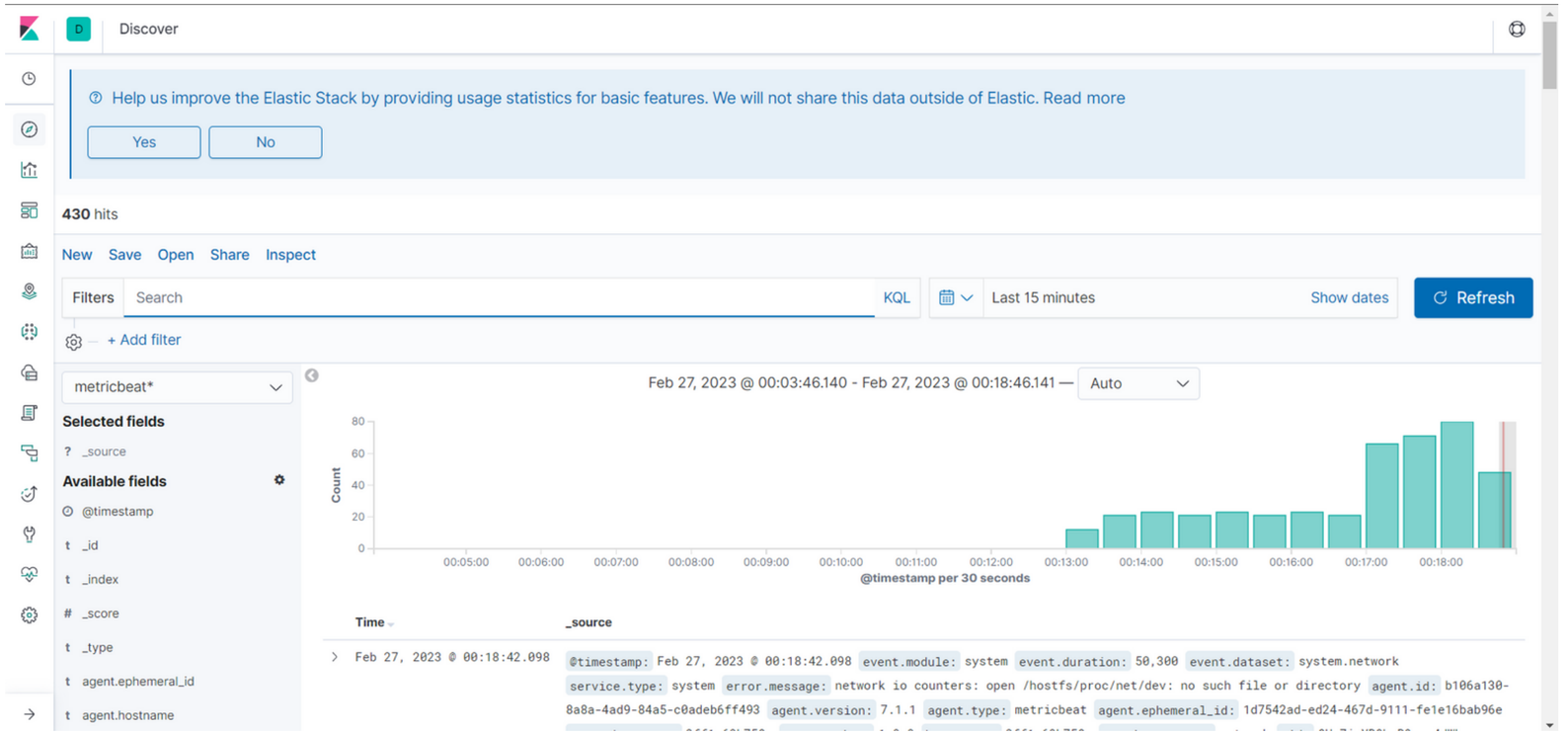
Siwagarn

Marineya

Nattapat

Monitor system's service: CPU&Mem usage

Metricbeat



100%

Nitchakran

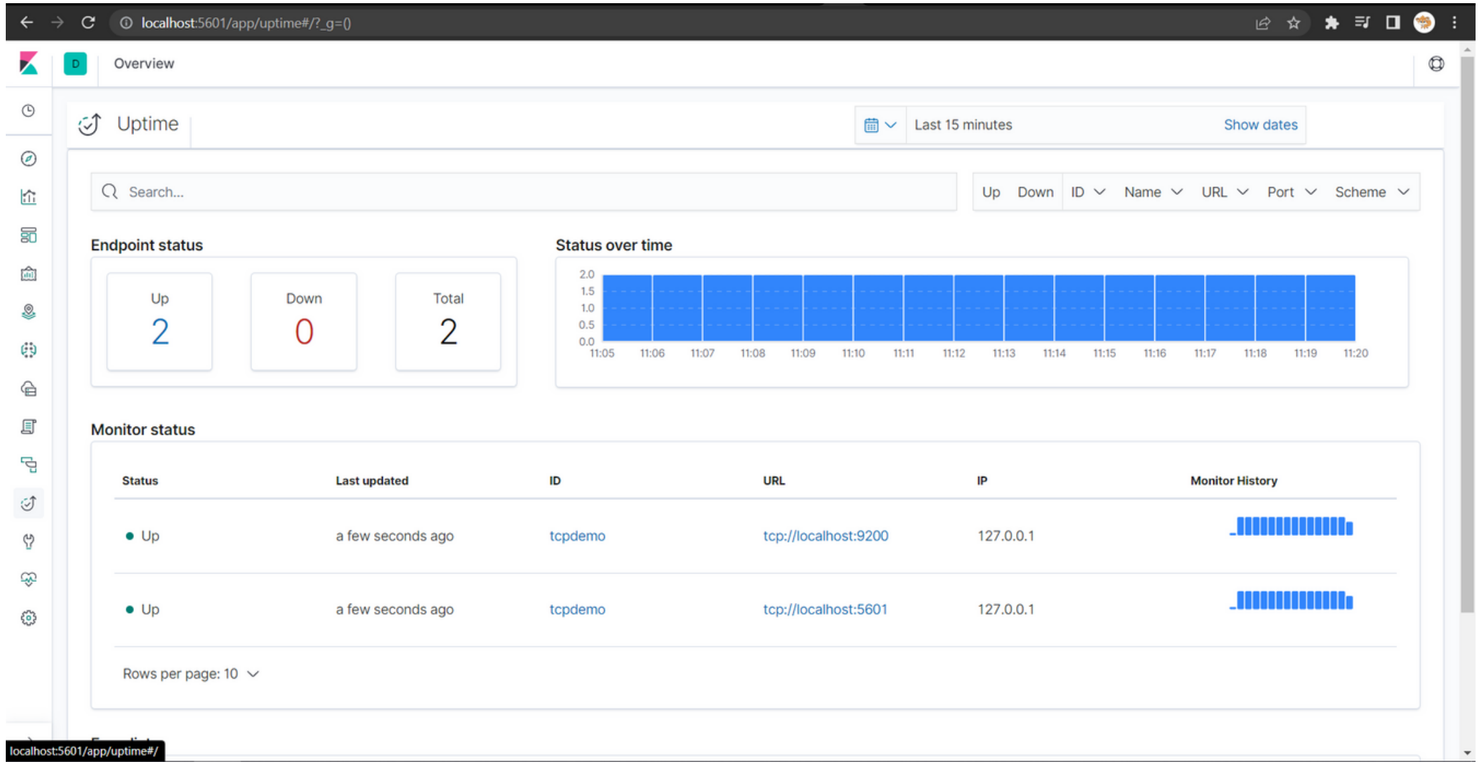
Siwagarn

Marineya

Nattapat

Logging dashboard

Heartbeat



100%

Nitchakran

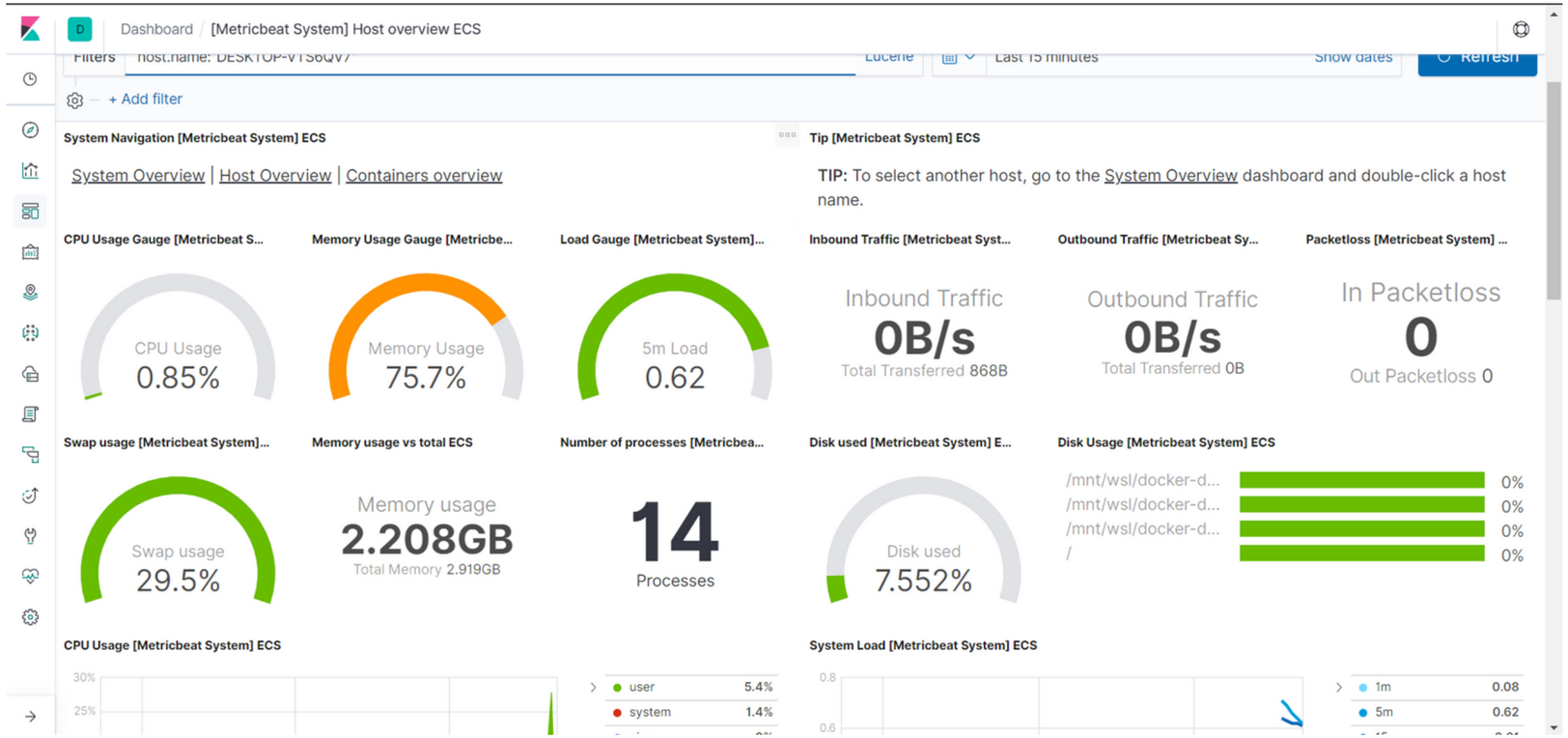
Siwagarn

Marineya

Nattapat

Logging dashboard

Metricbeat



100%

Nitchakran

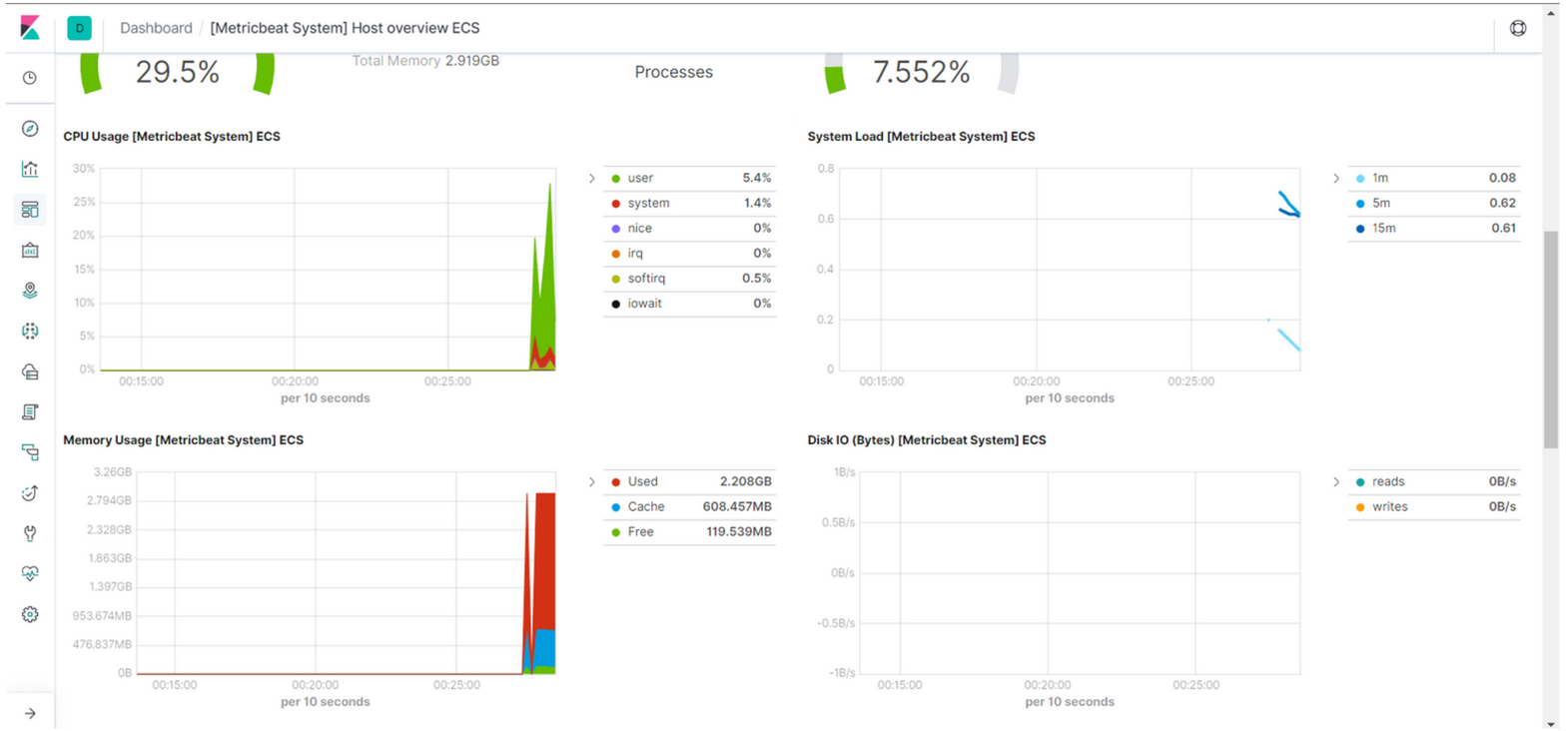
Siwagarn

Marineya

Nattapat

Logging dashboard

Metricbeat



100%

Nitchakran

Siwagarn

Marineya

Nattapat

Log every request & Uploaded status log

```go
func Init() *elasticsearch8.Client {
    configs.InitConfig()
    rootCAs, _ := x509.SystemCertPool()
    if rootCAs == nil {
        rootCAs = x509.NewCertPool()
    }
    certs, err := ioutil.ReadFile(viper.GetString("elasticsearch.caCert"))

    if err != nil {
        fmt.Println("error read file")
    }

    rootCAs.AppendCertsFromPEM(certs)
    config := elasticsearch8.Config{
        Addresses: viper.GetStringSlice("elasticsearch.address"),
        Username:  viper.GetString("elasticsearch.username"),
        Password:  viper.GetString("elasticsearch.password"),
        Transport: &http.Transport{
            MaxIdleConnsPerHost:   10,
            ResponseHeaderTimeout: time.Second,
            DialContext:           (&net.Dialer{Timeout: time.Second}).DialContext,
            TLSClientConfig: &tls.Config{
                MinVersion: tls.VersionTLS12,
                RootCAs:    rootCAs,
            },
        },
    }

    client, err := elasticsearch8.NewClient(config)

    if err != nil {
        panic(err)
    }

    return client
```

```go
var elasticClient = Init()

func UploadLog(index string, body interface{}) {
    _, err := elasticClient.Index(index, esutil.NewJSONReader(&body))
    fmt.Println("called")
    if err != nil {
        fmt.Println(err.Error())
    }
}
```

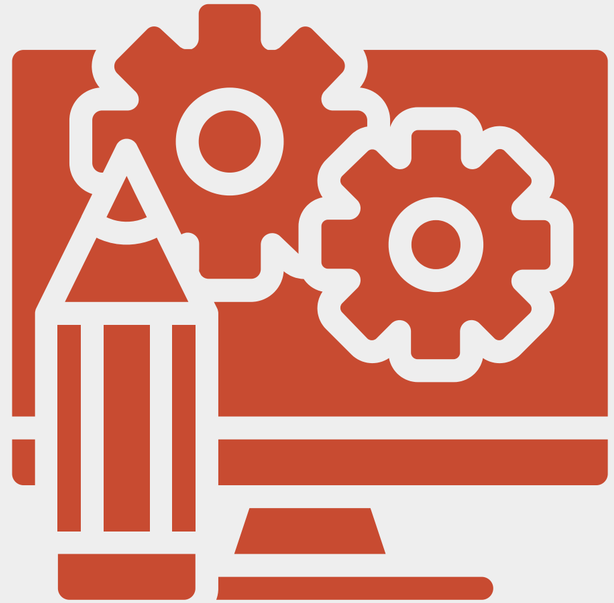
Nitchakran

Siwagarn

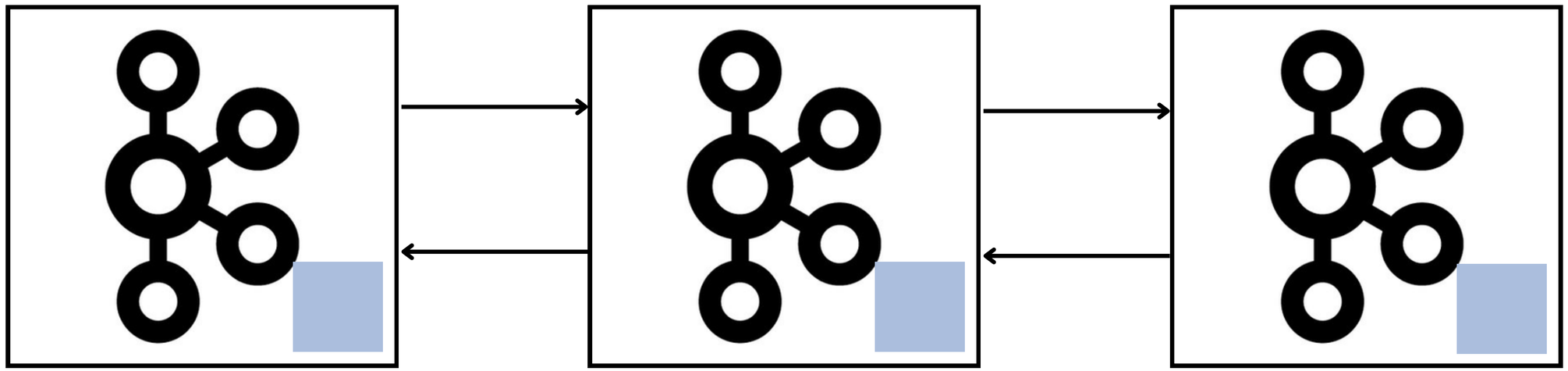
Marineya

Nattapat

Deployment



Kafka clusters



Each instances: 1-broker, 1-zookeeper

Security configuration: SASL_SSL

Nitchakran

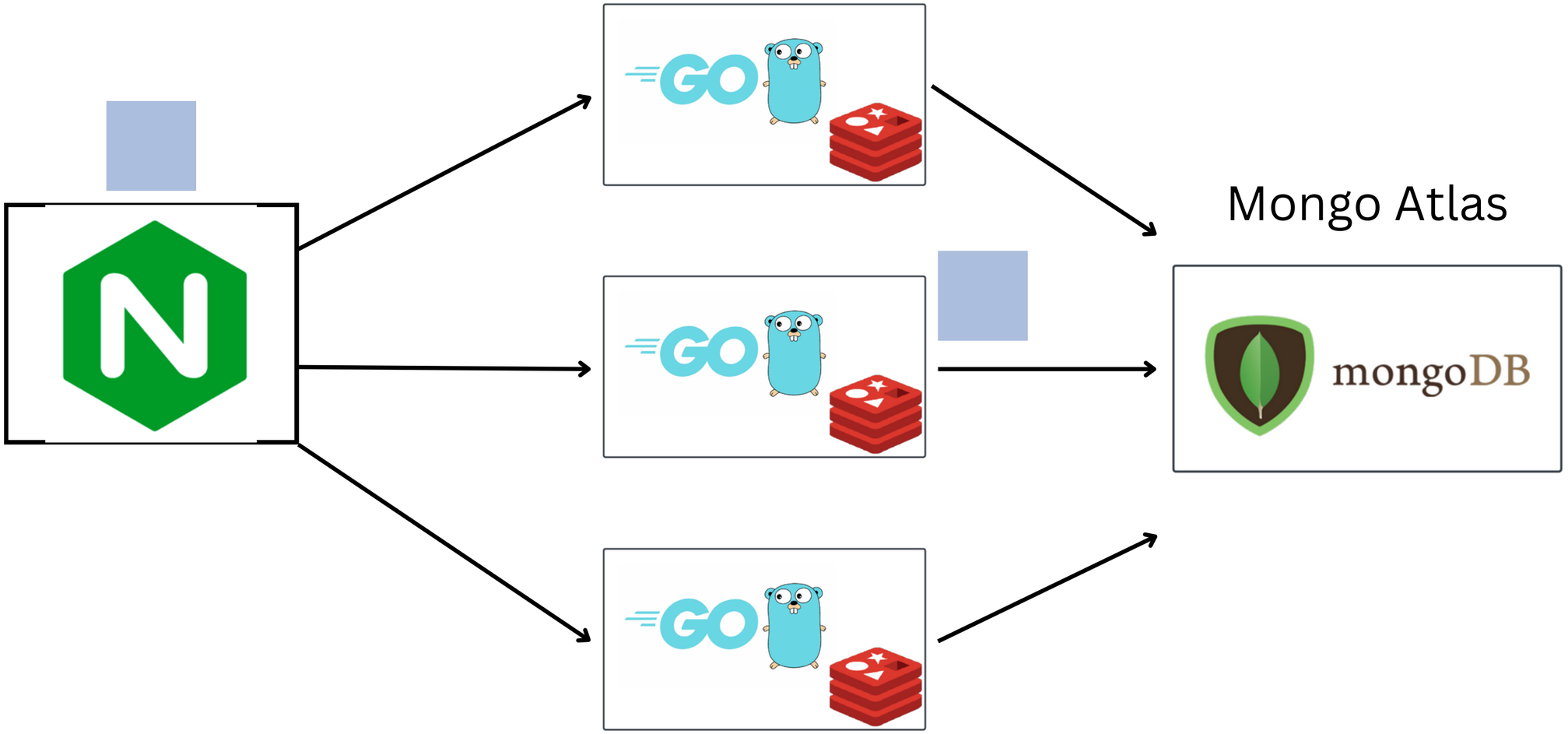
Siwagarn

Marineya

Nattapat

100%

Web service clusters



Nitchakran

Siwagarn

Marineya

Nattapat

Demo

Demo

I Developer Dashboard

II Admin Dashboard

III Application API

IV Uploading data to GCS

Demo


N
progression presentation (demo)
Share
Demo
Watch on YouTube

Summary

Done

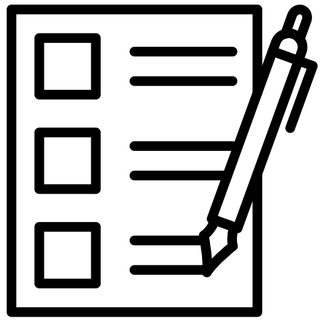
	Task	Responsibility	%
		2023: JAN, FEB, MAR, APR	
1	Data receiving system		100
	learn zookeeper and Kraft	มารินญา	100
	learn confluent kafka bridge	มารินญา	100
	docker compose file for confluent	มารินญา	100
	EMQX bridge kafka using EMQX Enterprise	มารินญา	100
	docker compose file for EMQX Enterprise	มารินญา	100
	learn MQTT,Kafka client library	มารินญา	100
	Build an MQTT Bridge to Kafka(hard code)	มารินญา	100
	learn pymongo	มารินญา	100
	connect bridge to mongoDB	มารินญา	100
	thread and coroutine concept	มารินญา	100
	trigger function when notice new modelName	มารินญา	100
	Build an MQTT Bridge to Kafka to get all data from DB	มารินญา	100
	MQTT Bridge authen to Kafka	มารินญา	100
	config EMQX cluster to Kafka	มารินญา	100
	Kafka - configure SASL_SSL	ณิชกานต์	100
2	Data management system		
	upload data from mongodb to cloud storage	ณิชกานต์	100
	receiver - implement receiver consumer group with new library	ณิชกานต์	100
3	Developer service system		
	Dashboard - Login	ศิวกาญจน์	100
	Dashboard - My Application (Get)	ศิวกาญจน์	100
	Dashboard - My Application (New)	ศิวกาญจน์	100
	Dashboard - My Application (Update)	ศิวกาญจน์	100
	Dashboard - My Application (RenewSecret)	ศิวกาญจน์	100
	Dashboard - My Application (Delete)	ศิวกาญจน์	100
	Dashboard - My Subscription (Get&Filter)	ศิวกาญจน์	100
	Dashboard - My Subscription (New)	ศิวกาญจน์	100
	Dashboard - My Account (Get)	ศิวกาญจน์	100
	Dashboard - My Account (Update)	ศิวกาญจน์	100
	Developer - Authenticate developers [Login]	ณิชกานต์	100
	Developer - Create developer	ณิชกานต์	100
	Developer - Update developer	ณิชกานต์	100
	Developer - Get all developer	ณิชกานต์	100
	Application - Create	ณิชกานต์	100
	Application - Get app by developer id	ณิชกานต์	100
	Application - Update app	ณิชกานต์	100
	Application - Update application secret	ณิชกานต์	100
	Subscription - Create subscription	ณิชกานต์	100
	Subscription - Get all subscription according to application and developer	ณิชกานต์	100
	Application - Delete app	ณิชกานต์	100
	Application api - Authenticate application api	ณิชกานต์	100
	Application api - Request historical data [code improvement]	ณิชกานต์	100
	Application api - Request real time data	ณิชกานต์	100
	Application api - [real-time data] sender consumer group [implement with new library]	ณิชกานต์	100

	Task	Responsibility	%
4	Admin service system	ณัฐภัทร	
	Dashboard - Get Developer's Subsription	ณัฐภัทร	90
	Dashboard - Update Developer's Subsription	ณัฐภัทร	90
	Dashboard - Delete Developer's Subsription	ณัฐภัทร	90
	Dashboard - Get Developer's Account	ณัฐภัทร	90
	Dashboard - Create Developer's Account	ณัฐภัทร	90
	Dashboard - Delete Developer's Account	ณัฐภัทร	90
	Dashboard - Get Device	ณัฐภัทร	100
	Dashboard - Create Device	ณัฐภัทร	100
	Dashboard - Delete Device	ณัฐภัทร	100
	Admin - Authentication [Login]	ณิชกานต์	100
	Developer - Create developers	ณิชกานต์	100
	Developer - Get all developers in the system	ณิชกานต์	100
	Developer - Delete developers	ณิชกานต์	100
	Medical model - Get all model in the systems [code improvement]	ณิชกานต์	100
	Medical model - Get all model and device id list	ณิชกานต์	100
	Medical model - Create medical model [code improvement]	ณิชกานต์	100
	Medical model - Update medical model	ณิชกานต์	100
	Medical model - Delete medical model	ณิชกานต์	100
	Subscription - Update subscription status	ณิชกานต์	100
5	Log system		
	Logging gcs upload status	ศิวกาญจน์	25
	Logging all request	ศิวกาญจน์	25
	Uptime monitoring	ศิวกาญจน์	100
	CPU/Mem monitoring	ศิวกาญจน์	100
	Kibana	ศิวกาญจน์	100
	Improve log when file was uploaded on google cloud storage	ณิชกานต์	100
6	Deployment		
	learned how to write nginx config file	ณิชกานต์	100
	Write nginx config file and create docker image for application	ณิชกานต์	100
	Write docker compose file for web service cluster	ณิชกานต์	100
	Deploy on AWS [3 instance 1 for nginx server and others for web service]	ณิชกานต์	100
	Create docker file for web service [need to have java runtime]	ณิชกานต์	0
	Configure Kafka with SASL_SSL with 3 zookeeper and 3 broker using 3 instance	ณิชกานต์	0

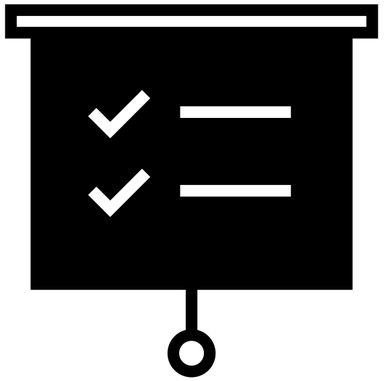
Todo

	Task	Responsibility	2023								%
			MAR				APR				
7	Testing										
	List data receiving test case	มารีนญา									0
	List admin dashboard test case	ณัฐภัทร									0
	List developer dashboard test case	ศิวกาญจน์									0
	Fundamental testing - Data receiving system	ทุกคน									0
	Fundamental testing - Developer service system [not include application api]	ทุกคน									0
	Fundamental testing - Application api	ทุกคน									0
	Fundamental testing - Admin service system	ทุกคน									0
	Web service high availability testing	ทุกคน									0

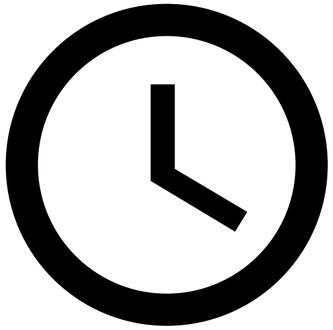
Testing



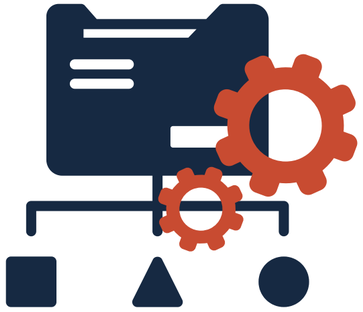
Fundamental Testing



High Availability Testing



Real-time data request



Kafka cluster security

Thank you for listening