Design Pattern of Hippo Service

Roger

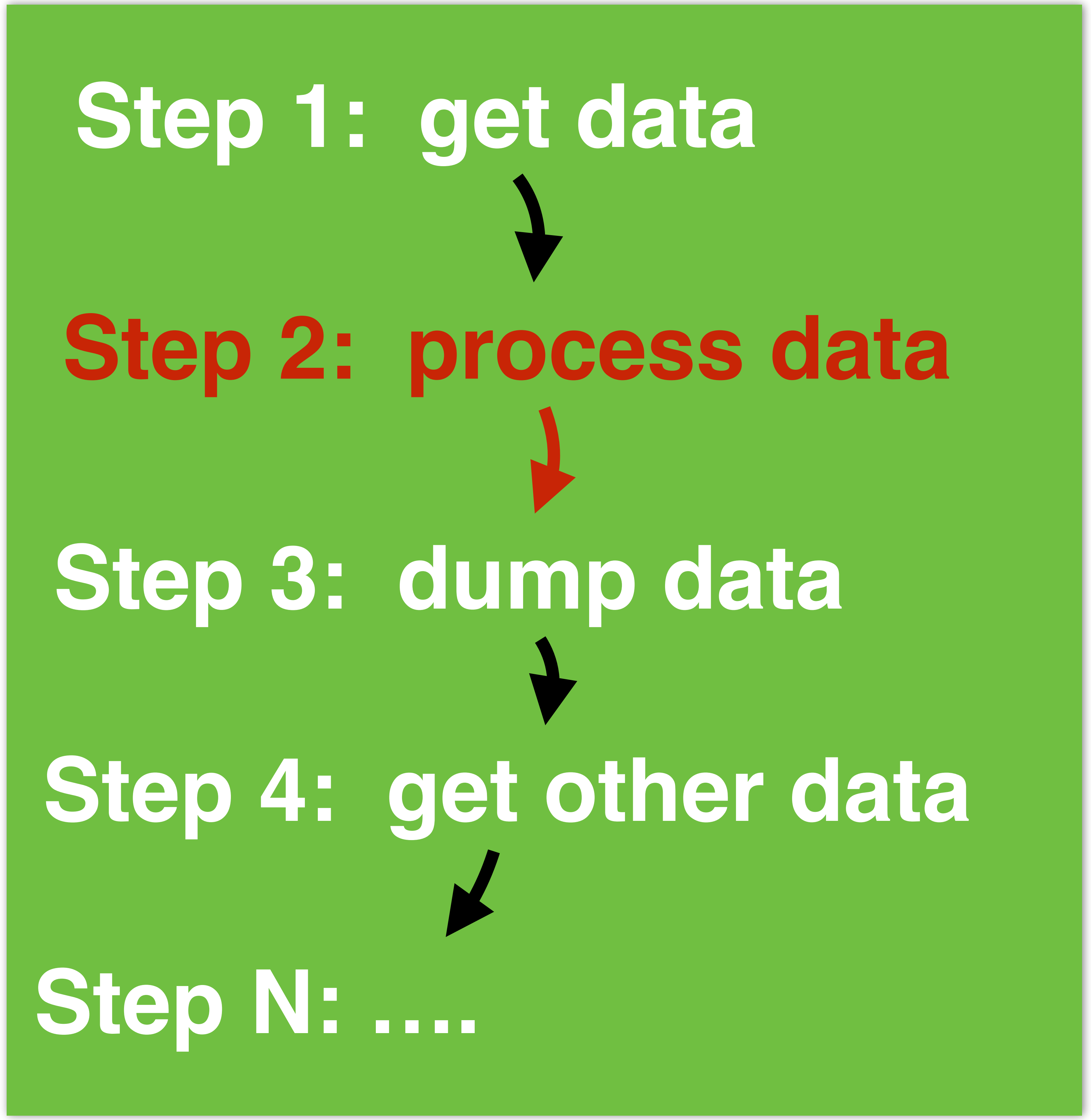
What is this?

Borrow the concept from

Microservice

是什麼？

單一大架構



Step 1: get data

Step 2: process data

Step 3: dump data

Step 4: get other data

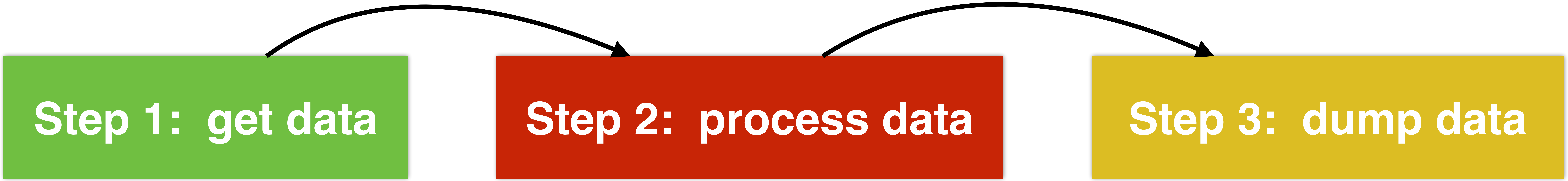
Step N: ....

Problem1: App太大包

Problem 2: 一步驟掛, 全掛

Problem 3: 每一步驟不能用不同技術

排程架構



Step 1: get data

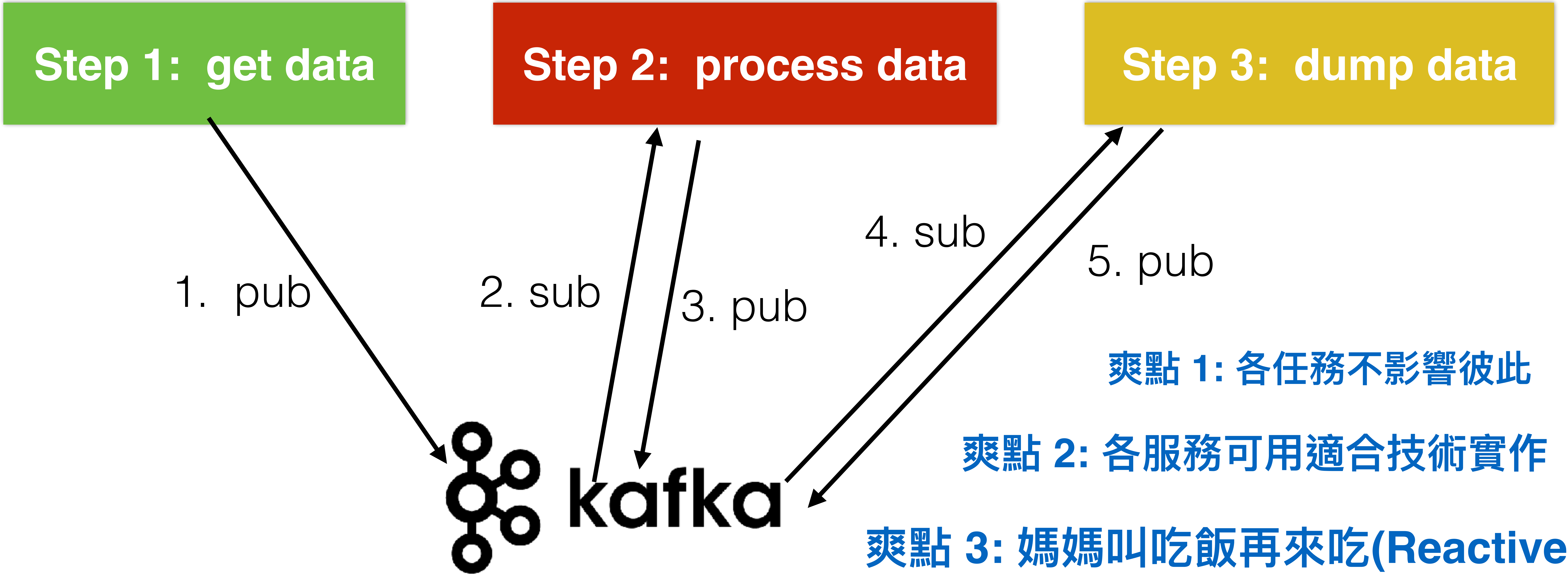
Step 2: process data

Step 3: dump data

Problem 1: 每一步驟該設幾點?

Problem 2: 一步驟掛, 還是接著跑!

微服務架構



Step 1: get data

Step 2: process data

Step 3: dump data

1. pub

2. sub

3. pub

4. sub

5. pub

kafka

爽點 1: 各任務不影響彼此

爽點 2: 各服務可用適合技術實作

爽點 3: 媽媽叫吃飯再來吃(Reactive)

訊息對列 (pubsub, 事件發送與訂閱...)

HIPPO types by purpose

ETL

- Frontier
- Batch-ETL
- DB-ETL
- Stream-ETL

Science

- ML Service

Application

- Query Service
- Web Service

Operation

- Doctor
- Integrator

One service may comprises mixed types

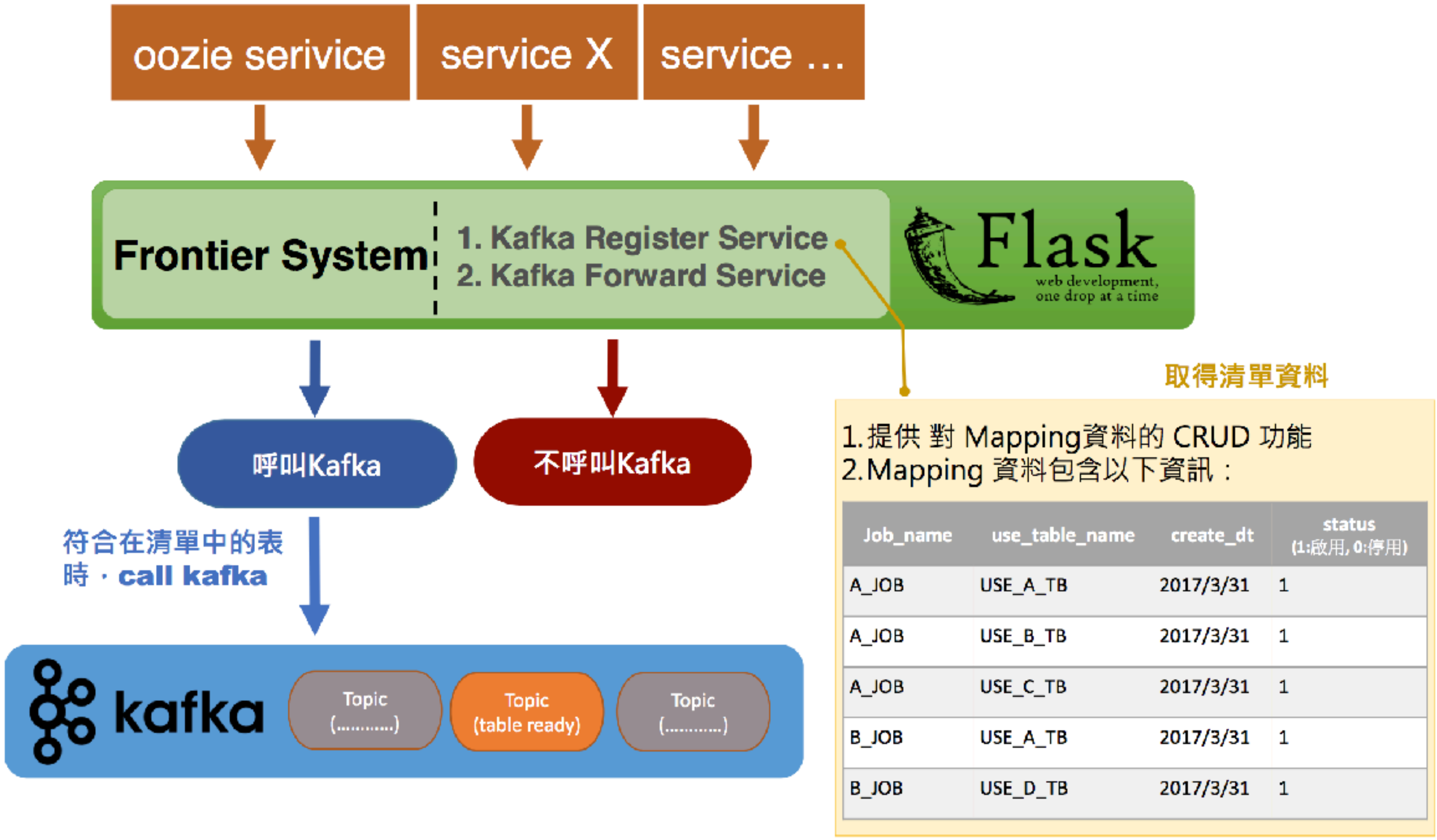
Future HIPPO Views

Operation
Doctor
Tractor
ML
Features
Training
Predictor
Evaluator
HIVE
ADW
Frontier
kafka
Queue、Stream
redis
Web
Batch ETL
DB ETL
Query
ETL
Stream ETL
API
mongoDB
NoSQL
Application
客經
IT
金服
理專
分行
…

ETL

Frontier

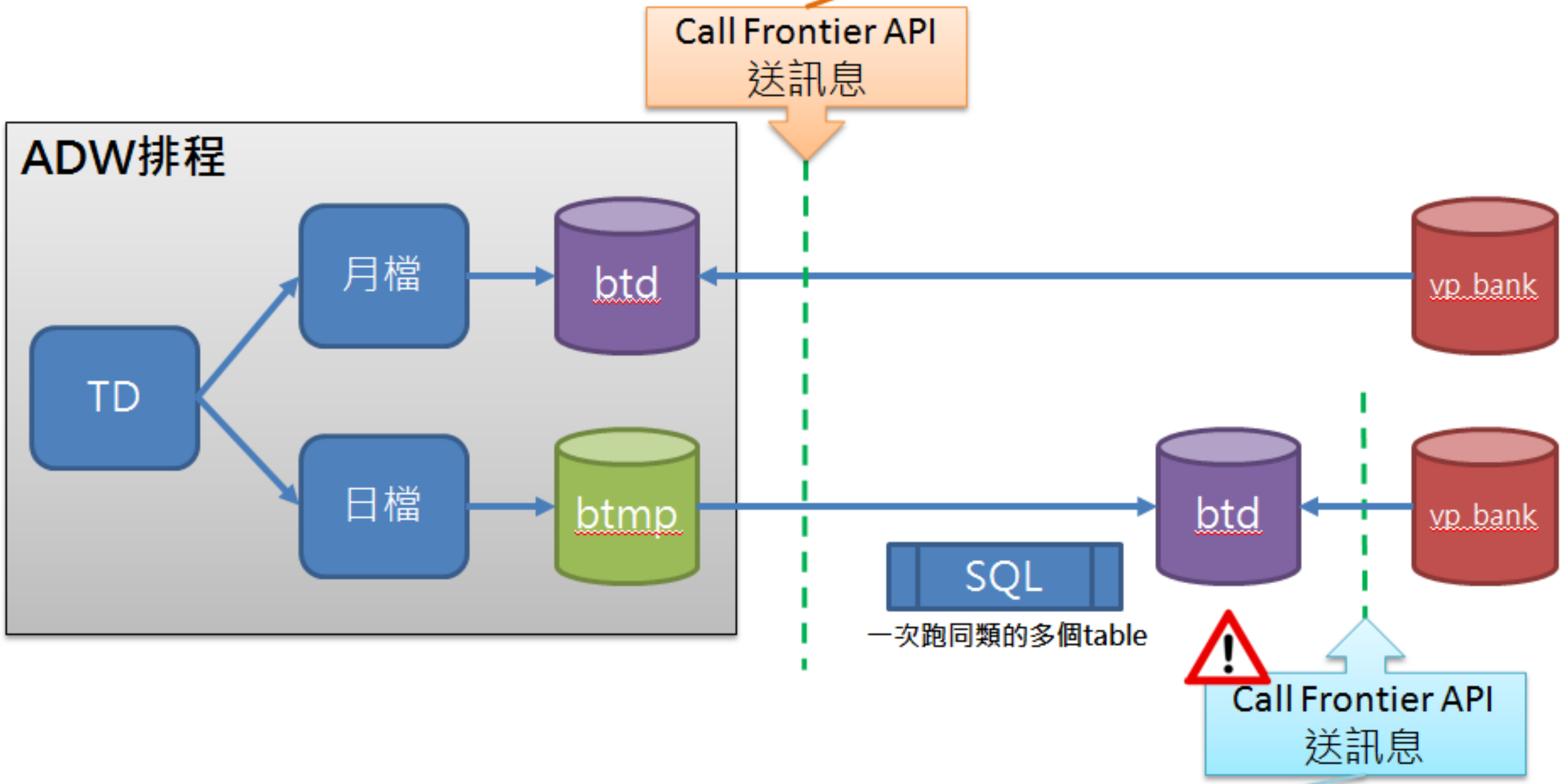
架構圖



1.提供 對 Mapping資料的 CRUD 功能
2.Mapping 資料包含以下資訊：

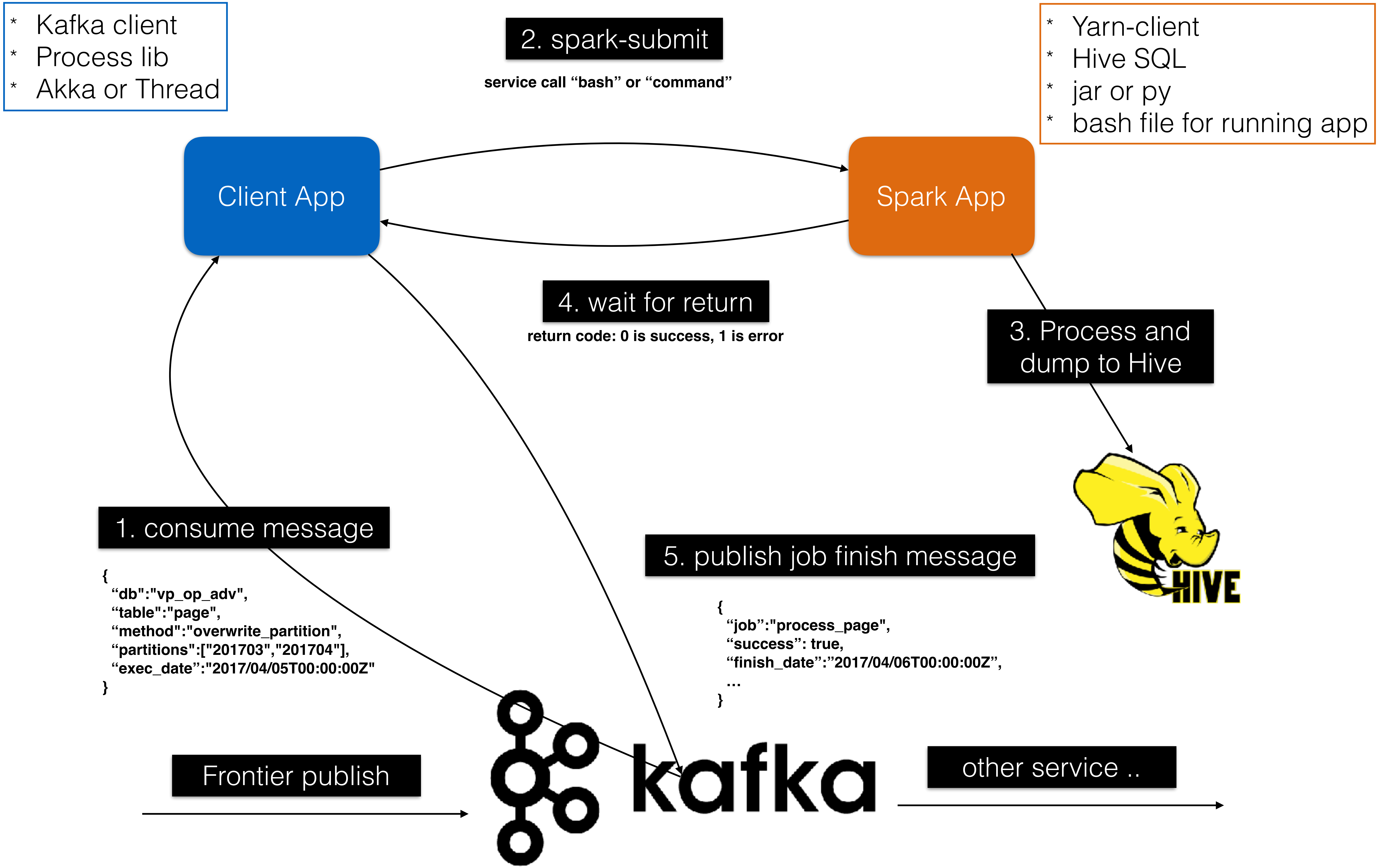
Job_name	use_table_name	create_dt	status (1:啟用, 0:停用)
A_JOB	USE_A_TB	2017/3/31	1
A_JOB	USE_B_TB	2017/3/31	1
A_JOB	USE_C_TB	2017/3/31	1
B_JOB	USE_A_TB	2017/3/31	1
B_JOB	USE_D_TB	2017/3/31	1

【傳送到kafka的訊息內容】
月檔：{"table": "PARTY_TRU_ACCUM_BAL_M", "partition": [{"name": "YYYYMM", "value": "201703"}], "db": "btd", "method": "overwrite_partition", "exec_date": "1493394946"}
日檔：{"table": "PARTY_EDW_CUST_INS_SALES", "partition": [{"name": "", "value": ""}], "db": "btmp", "method": "partition", "exec_date": "1493394946"}



【傳送到kafka的訊息內容】
日檔：{"table": "PARTY_EDW_CUST_INS_SALES", "partition": [{"name": "", "value": ""}], "db": “btd", "method": "partition", "exec_date": "1493394946"}

Batch-ETL



- Kafka client
- Process lib
- Akka or Thread

2. spark-submit

service call “bash” or “command”

- Yarn-client
- Hive SQL
- jar or py
- bash file for running app

Client App

Spark App

4. wait for return

return code: 0 is success, 1 is error

3. Process and dump to Hive

1. consume message

```
{
  “db":"vp_op_adv",
  “table":"page",
  “method":"overwrite_partition",
  “partitions":["201703","201704"],
  “exec_date”:"2017/04/05T00:00:00Z"
}
```

5. publish job finish message

```
{
  “job”:"process_page",
  “success”: true,
  “finish_date”:”2017/04/06T00:00:00Z”,
  …
}
```

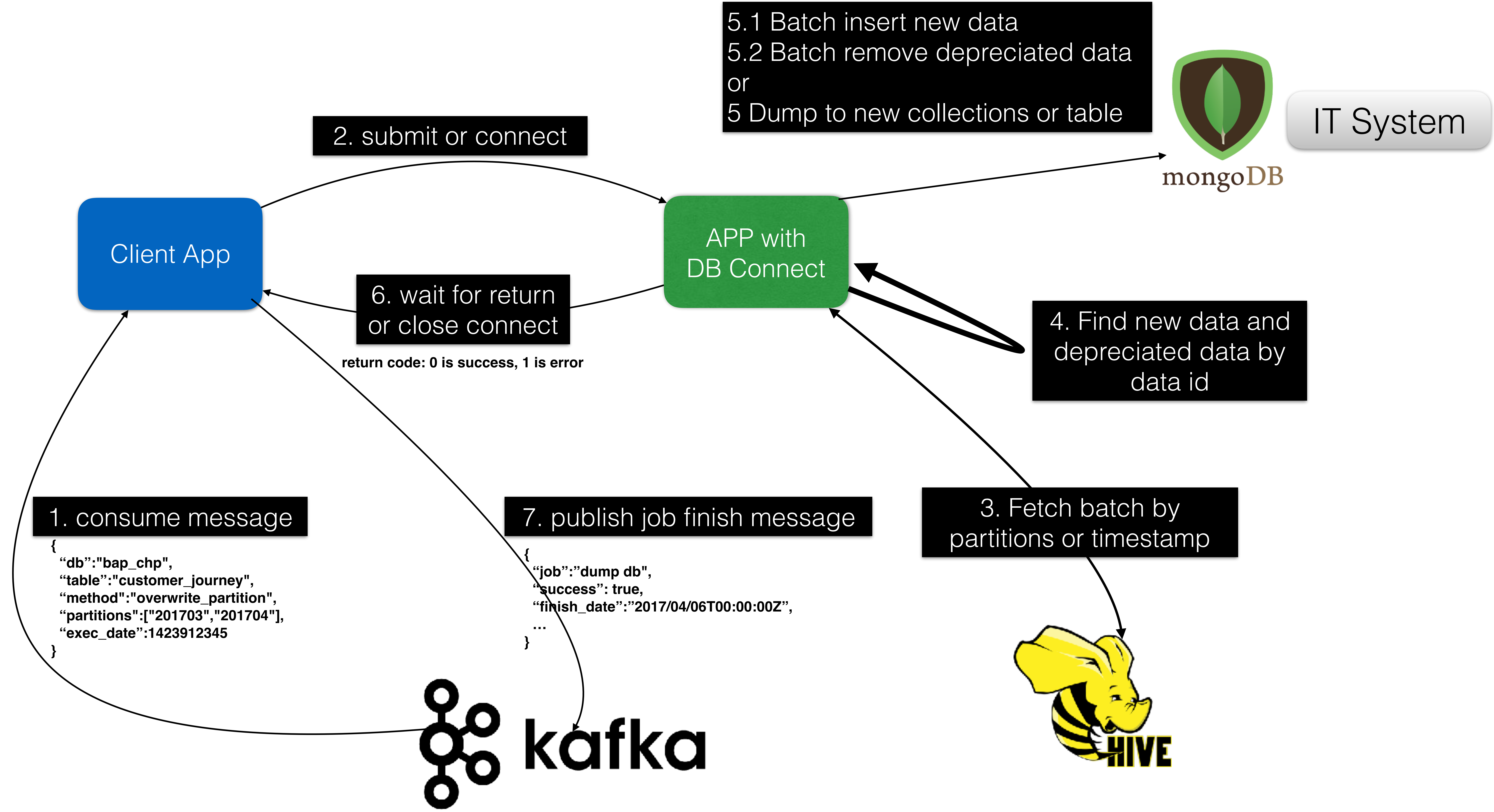
Frontier publish

other service ..

Structure

- start-client.sh
- start-spark.sh
- client app folder
- spark app folder

DB-ETL



5.1 Batch insert new data
5.2 Batch remove depreciated data
or
5 Dump to new collections or table

2. submit or connect

6. wait for return or close connect

return code: 0 is success, 1 is error

4. Find new data and depreciated data by data id

1. consume message

```
{
  “db”:"bap_chp",
  “table”:"customer_journey",
  “method":"overwrite_partition",
  “partitions”:["201703","201704"],
  “exec_date”:1423912345
}
```

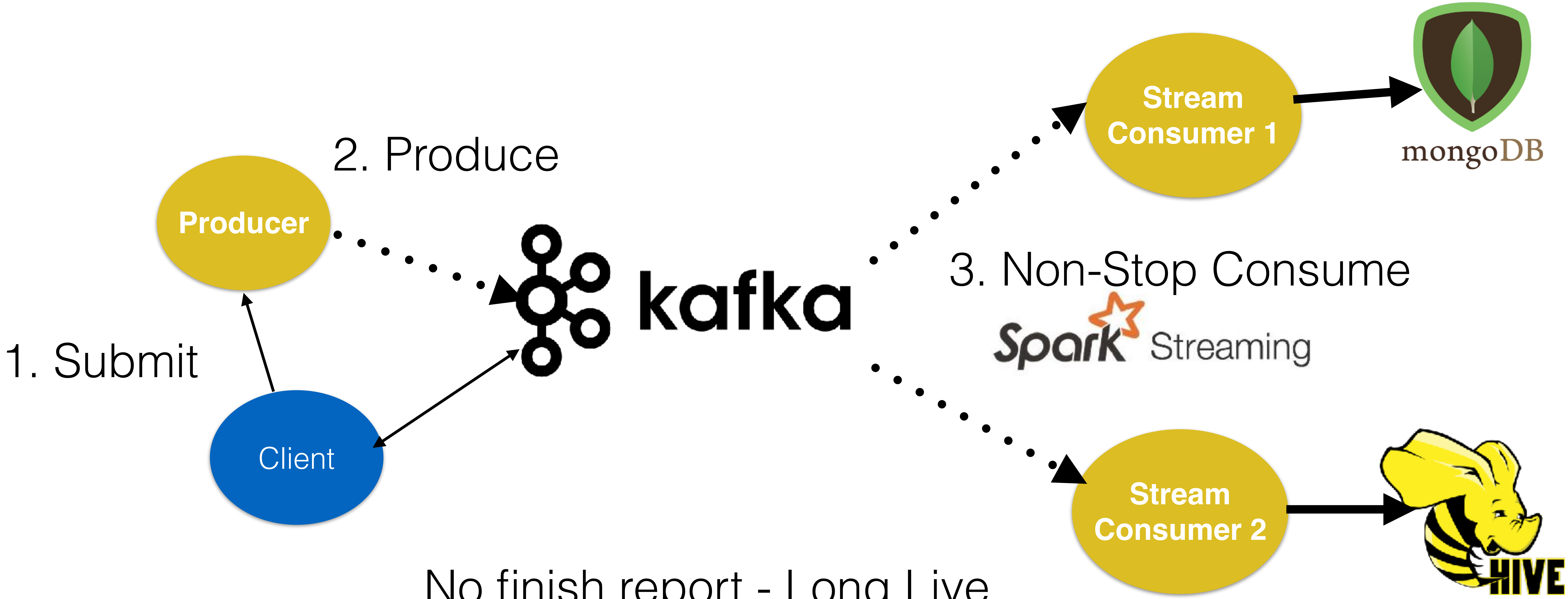
7. publish job finish message

```
{
 “job”:”dump db",
 “success”: true,
 “finish_date”:”2017/04/06T00:00:00Z”,
 …
}
```

3. Fetch batch by partitions or timestamp

Stream-ETL

Same source, but different consume purposes



2. Produce

Producer

1. Submit

Client

kafka

3. Non-Stop Consume

Spark Streaming

Stream Consumer 1

mongoDB

Stream Consumer 2

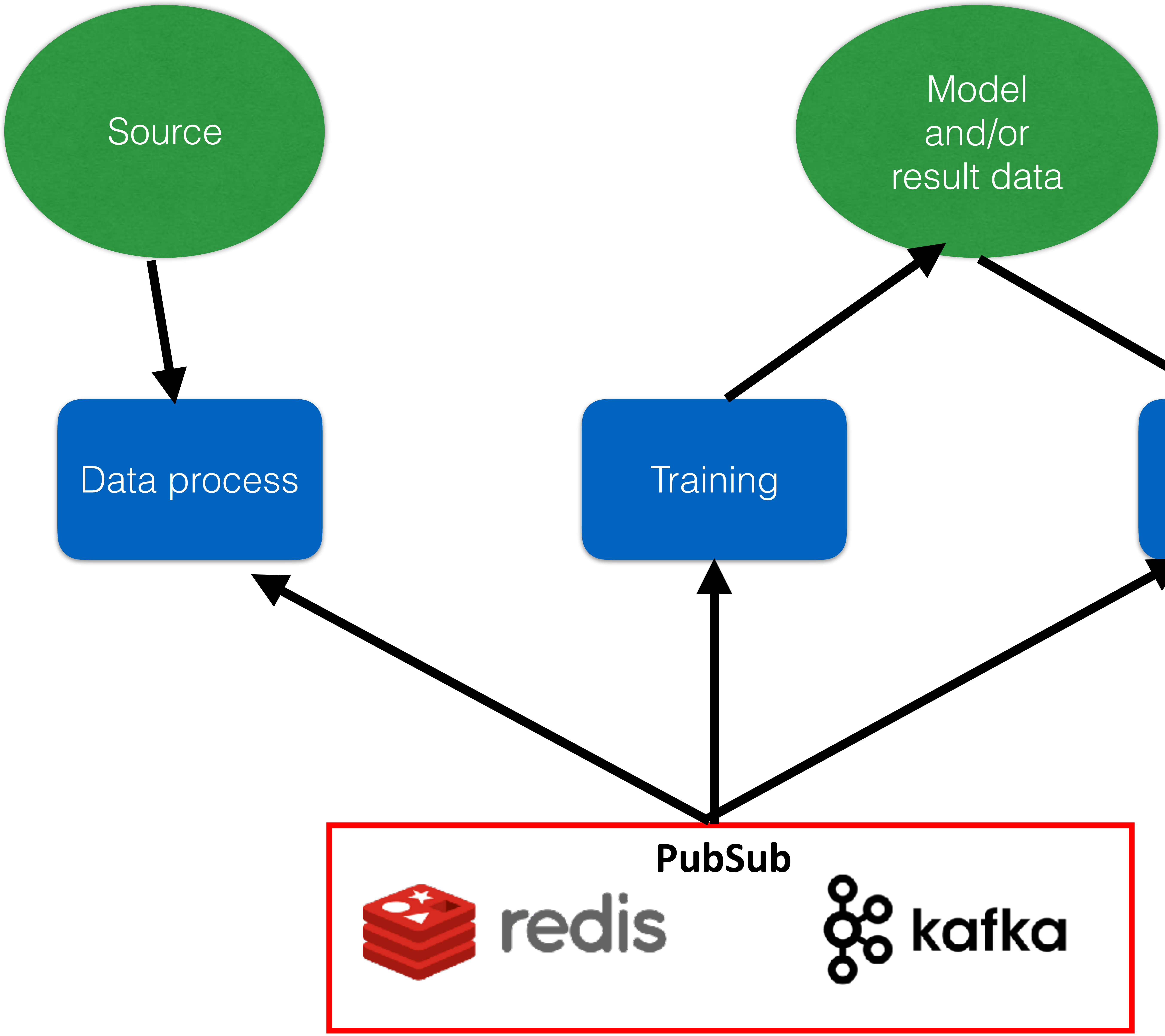
HIVE

No finish report - Long Live

Neha Narkhede, 2016
ETL Is Dead, Long Live Streams: real-time streams w/ Apache Kafka
https://www.youtube.com/watch?v=I32hmY4diFY&list=PLVeYbWw30yOKWUZOrLqeVjWf_Vfvy3AGf

Science

ML Service



Source

Model
and/or
result data

Data process

Training

API engine

PubSub

redis

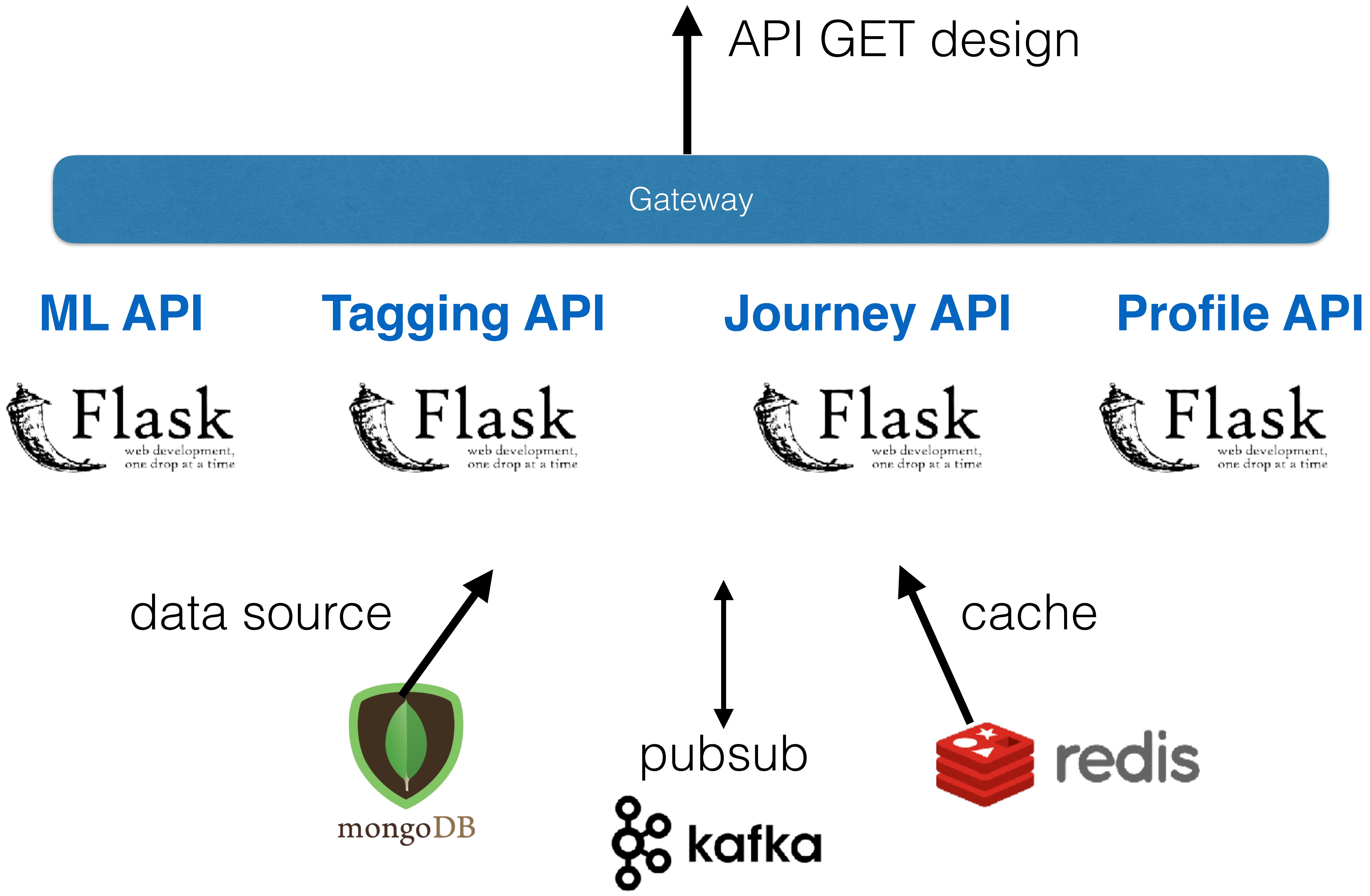
kafka

Repository

github.com/b96705008/MLServices

Application

Query Service



API GET design

Gateway

ML API

Tagging API

Journey API

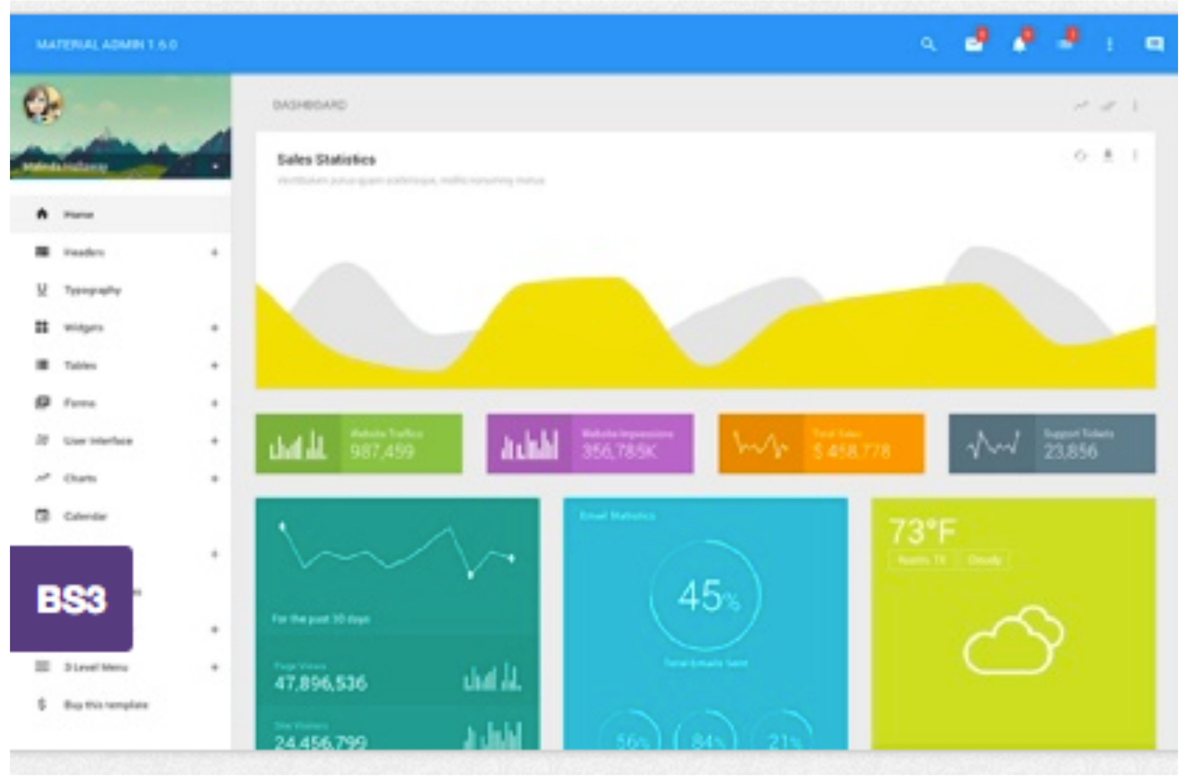
Profile API

data source

pubsub

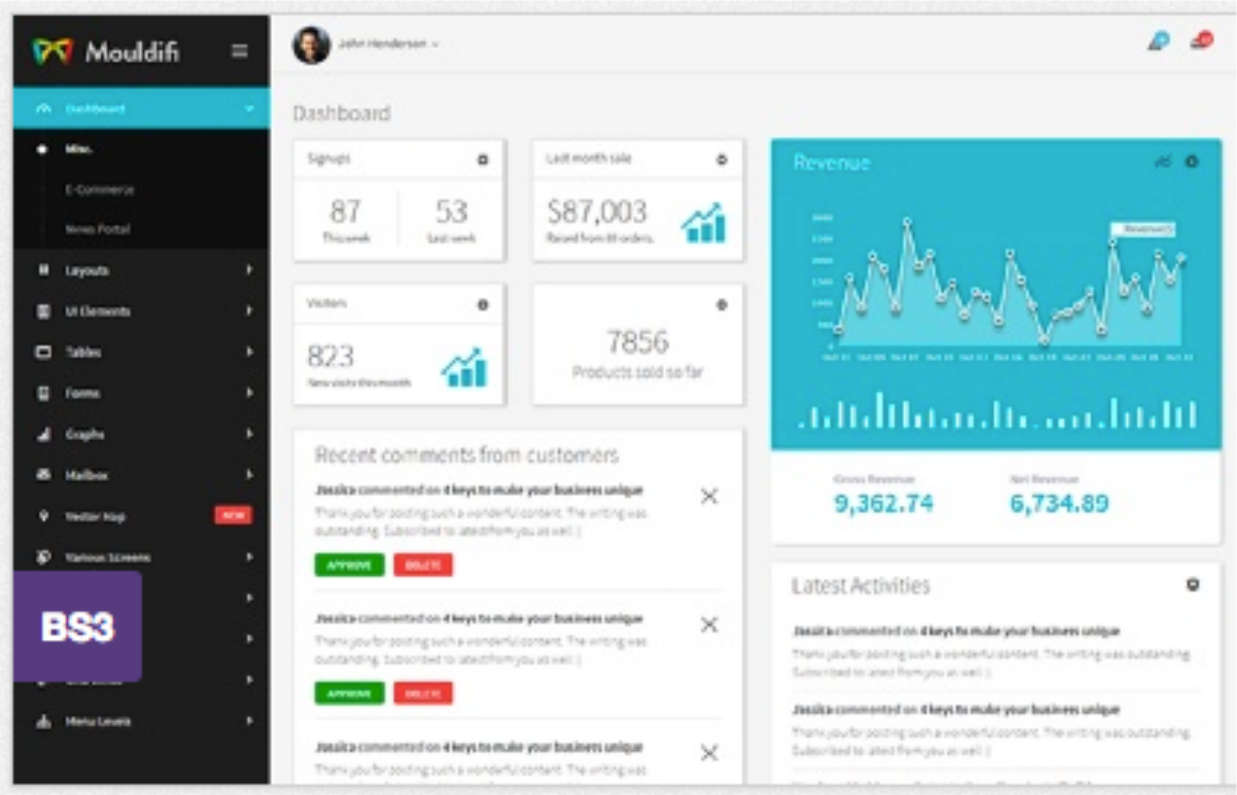
cache

Web Service

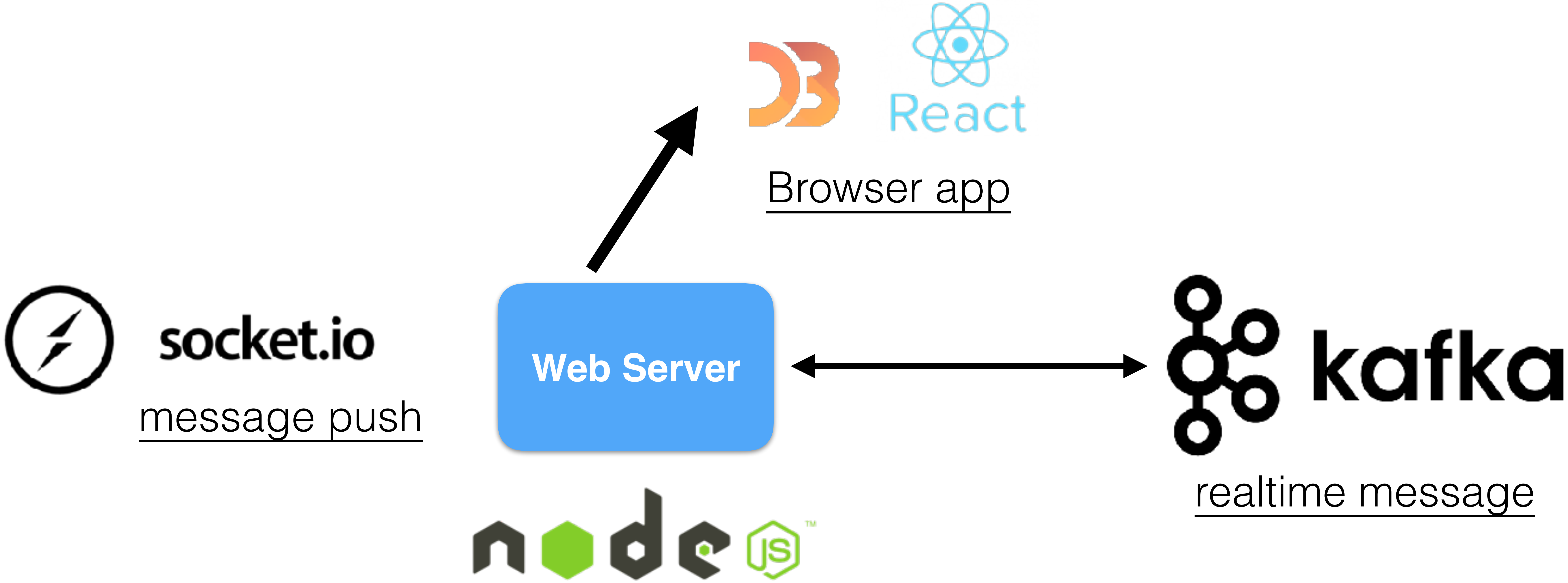


Data Track

Decision Making Support



Supervisor



Browser app

message push

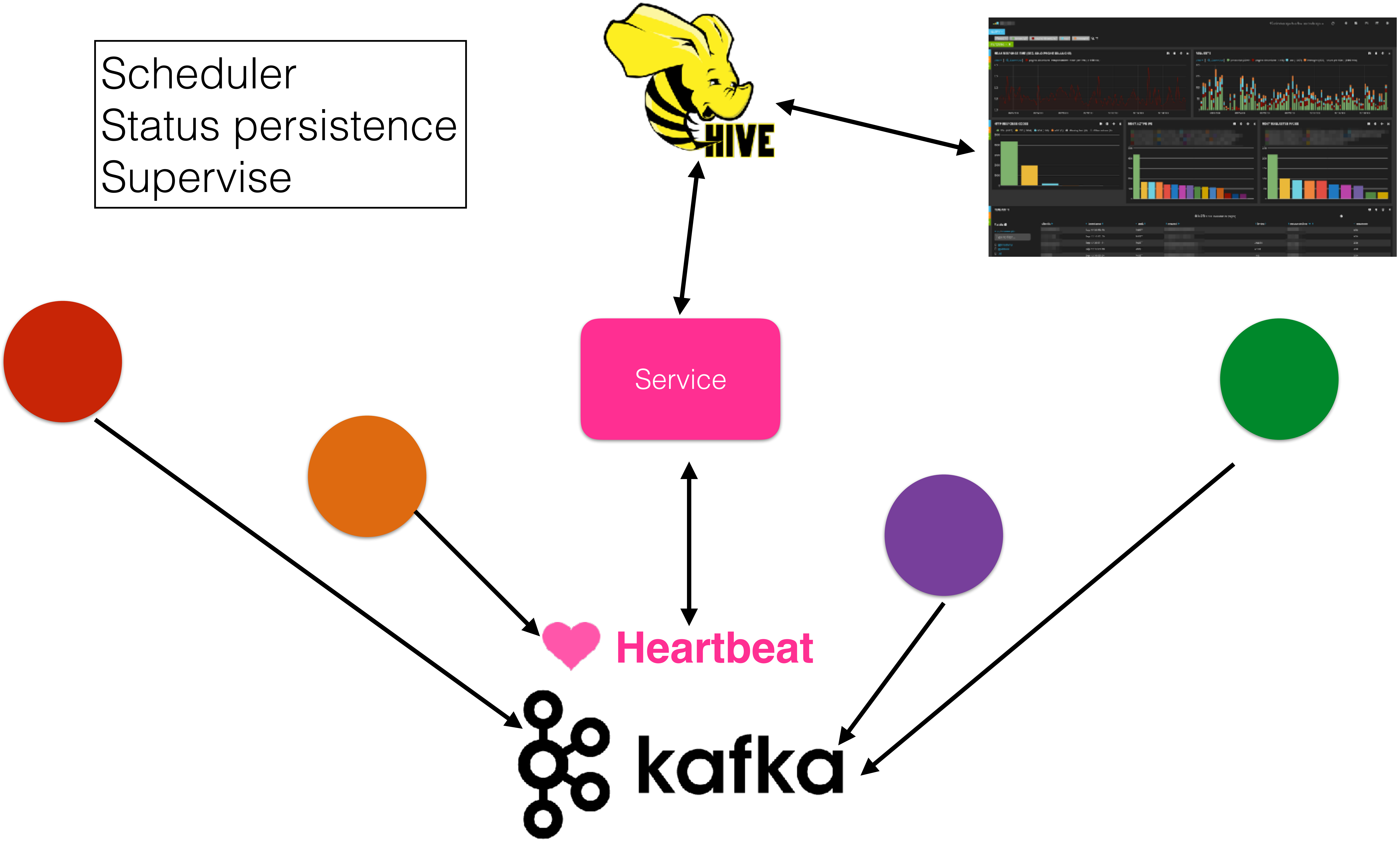
Web Server

realtime message

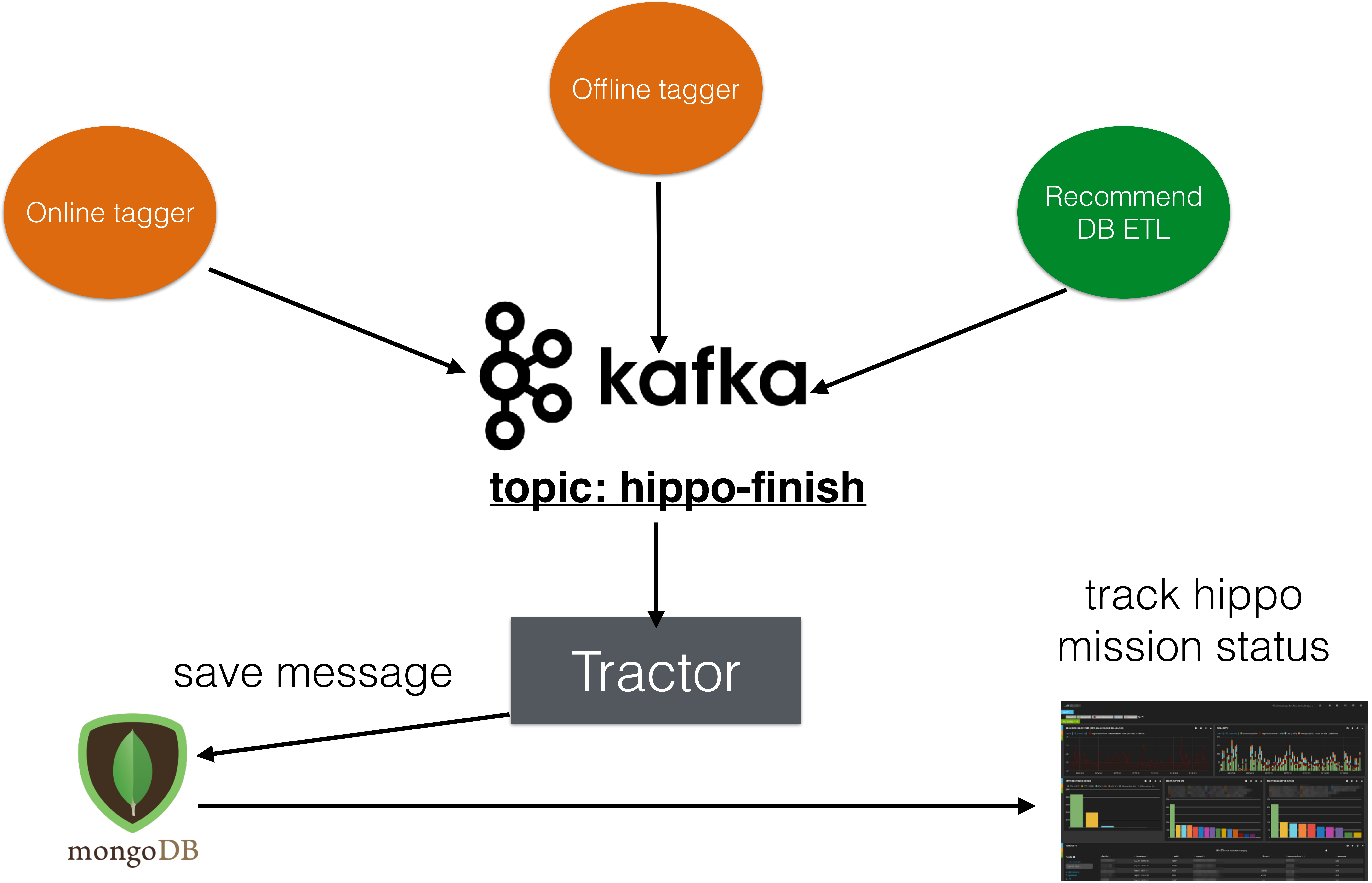
Operation

Doctor

Scheduler
Status persistence
Supervise



Tractor



Offline tagger

Online tagger

Recommend DB ETL

kafka

topic: hippo-finish

track hippo mission status

save message

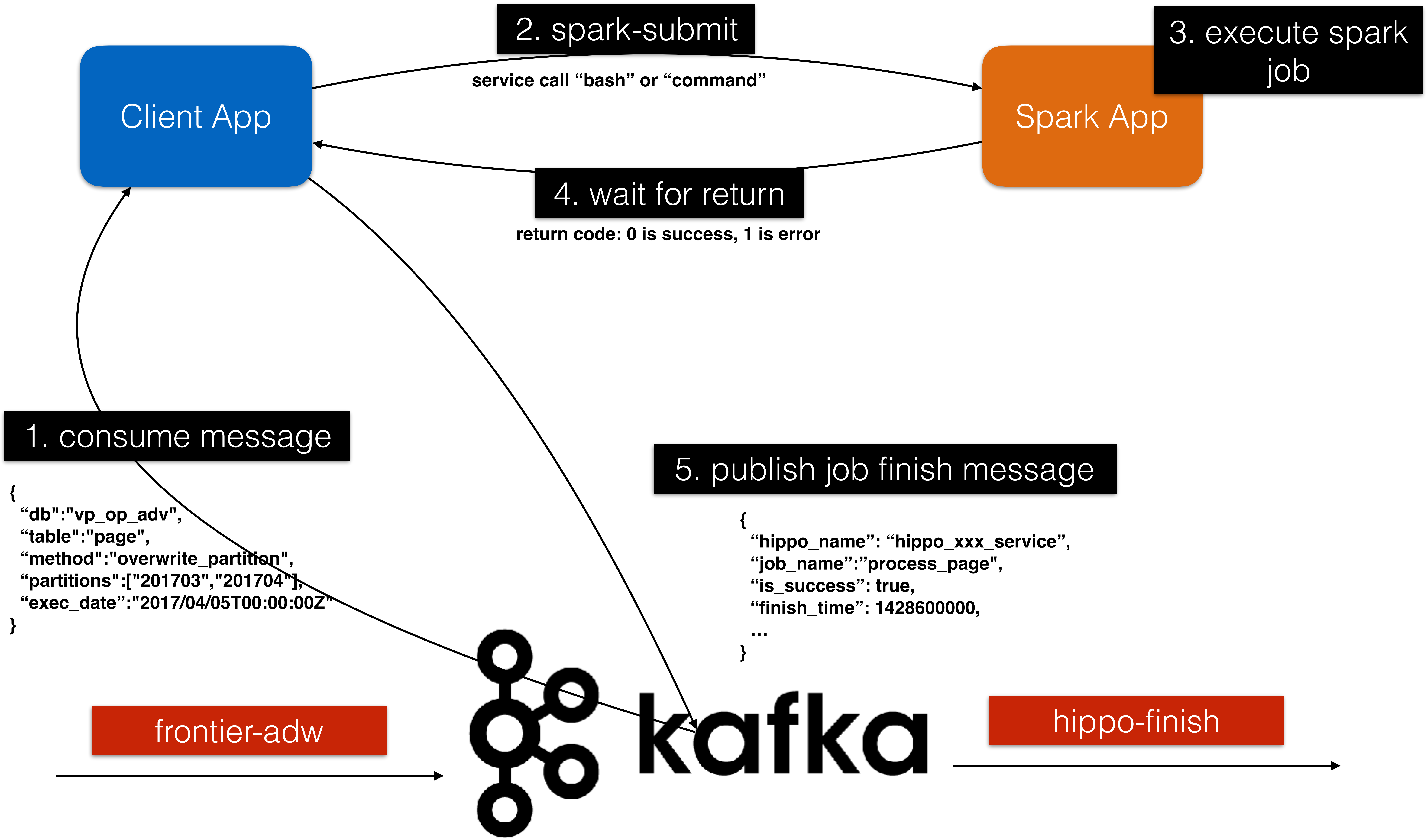
Tractor

mongoDB

Let’s try!

Hippo basic config

- hippo name
- subscribe topics
- publish topic
- port [optional]
- host [optional]

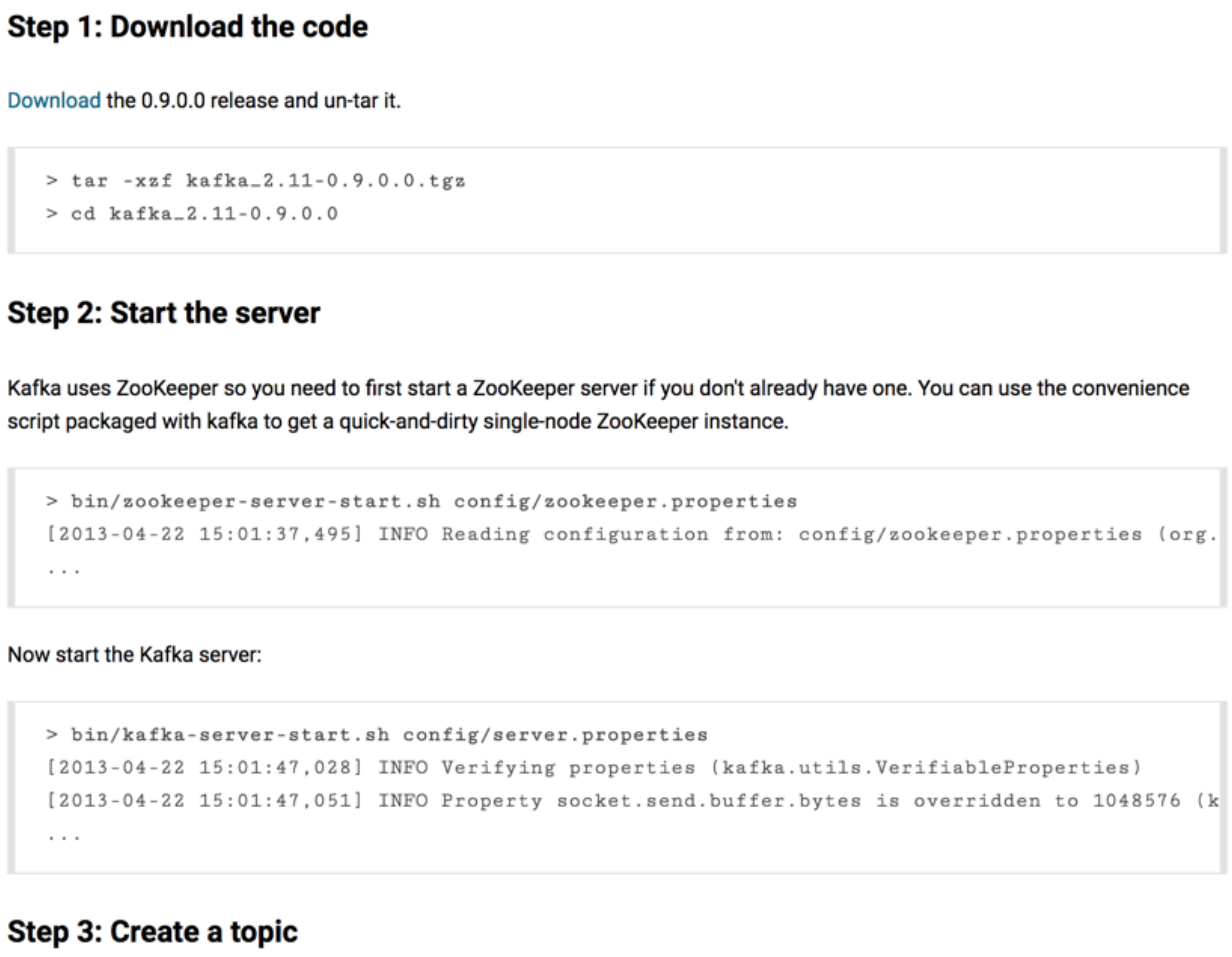


```
{
  “db":"vp_op_adv",
  “table":"page",
  “method":"overwrite_partition",
  “partitions":["201703","201704"],
  “exec_date”:"2017/04/05T00:00:00Z"
}
```

```
{
  “hippo_name”: “hippo_xxx_service”,
  “job_name”:”process_page",
  “is_success”: true,
  “finish_time”: 1428600000,
  …
}
```

Step 1: Kafka Quickstart

- https://kafka.apache.org/090/documentation.html#gettingStarted



Step 1: Download the code

Download the 0.9.0.0 release and un-tar it.

```
> tar -xzf kafka_2.11-0.9.0.0.tgz
> cd kafka_2.11-0.9.0.0
```

Step 2: Start the server

Kafka uses ZooKeeper so you need to first start a ZooKeeper server if you don't already have one. You can use the convenience script packaged with kafka to get a quick-and-dirty single-node ZooKeeper instance.

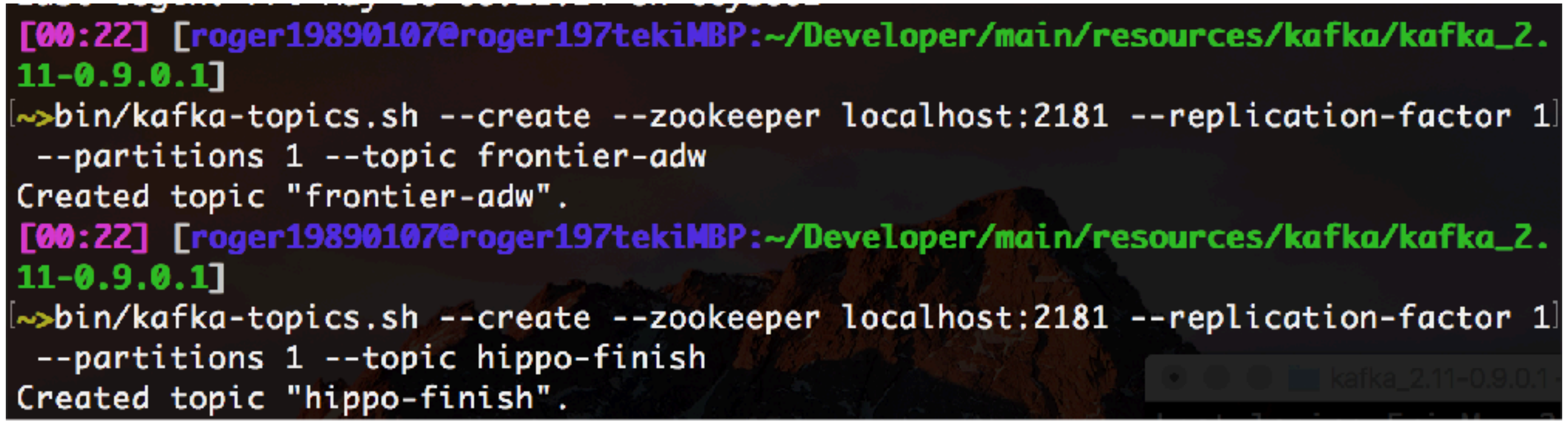
```
> bin/zookeeper-server-start.sh config/zookeeper.properties
[2013-04-22 15:01:37,495] INFO Reading configuration from: config/zookeeper.properties (org.
...
```

Now start the Kafka server:

```
> bin/kafka-server-start.sh config/server.properties
[2013-04-22 15:01:47,028] INFO Verifying properties (kafka.utils.VerifiableProperties)
[2013-04-22 15:01:47,051] INFO Property socket.send.buffer.bytes is overridden to 1048576 (k
...
```

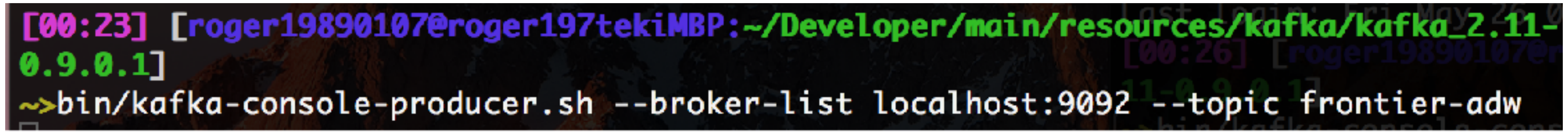
Step 3: Create a topic

Step 2: Prepare Kafka for Hippo



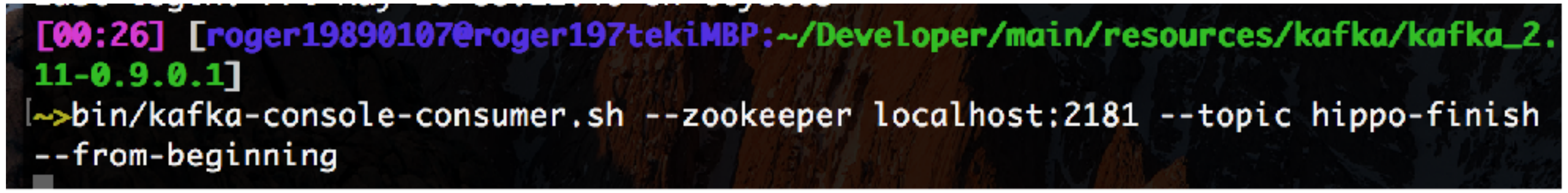
```
[00:22] [roger19890107@roger197tekiMBP:~/Developer/main/resources/kafka/kafka_2.
11-0.9.0.1]
~>bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1
 --partitions 1 --topic frontier-adw
Created topic "frontier-adw".
[00:22] [roger19890107@roger197tekiMBP:~/Developer/main/resources/kafka/kafka_2.
11-0.9.0.1]
~>bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1
 --partitions 1 --topic hippo-finish
Created topic "hippo-finish".
```

create topics: frontier-adw, hippo-finish



```
[00:23] [roger19890107@roger197tekiMBP:~/Developer/main/resources/kafka/kafka_2.11-
0.9.0.1]
~>bin/kafka-console-producer.sh --broker-list localhost:9092 --topic frontier-adw
```

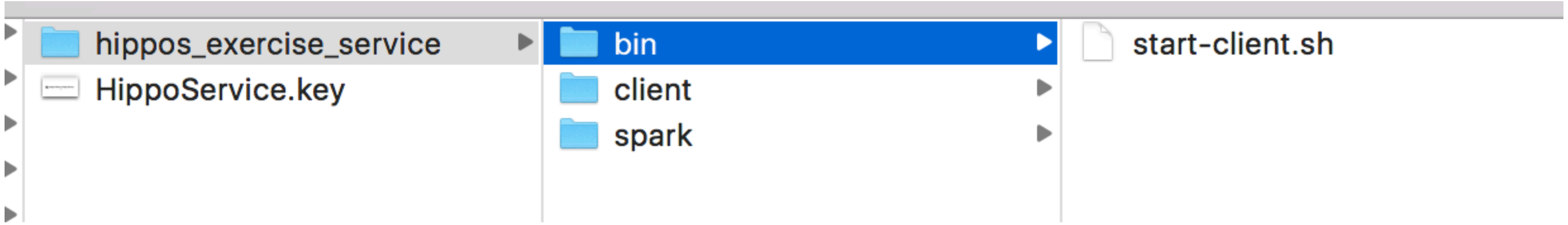
start producer for frontier-adw



```
[00:26] [roger19890107@roger197tekiMBP:~/Developer/main/resources/kafka/kafka_2.
11-0.9.0.1]
~>bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic hippo-finish
--from-beginning
```

start consumer for hippo-finish

Step 3: Write Hippo!



hippos_exercise_service

HippoService.key

bin

client

spark

start-client.sh

Thanks!