

# 物聯網技術結合AI大數據 分析實作(2)

徐凡耘   
2019-09



## DAY2

### ▶ 上午

- ▶ 安裝樹莓派 **Kafka** 套件
- ▶ **Docker**環境建置
- ▶ 溫濕度資料收集與上傳

### ▶ 下午

- ▶ 啟動 **Message Queue** (套件 **Kafka/ Ksql**)
- ▶ 啟動 **NoSQL**儲存與**ETL** (套件 **Mongodb/ pyspark**)
- ▶ 查詢結果 (套件 **AdminMongo**)



# AIGO業界出題，考驗AI落地

- 官網

- [https://aigo.org.tw/ai-plus/home/new\\_index](https://aigo.org.tw/ai-plus/home/new_index)

- 出題案例

- 【天氣風險管理開發】 AI道路監視器影像判斷降雨：利用影像分析地面、天空、即時天氣資訊
  - 【台灣楓康超市】 自動辨識水果甜度挑選水果方案：以影像方式辨識水果甜度
  - 【邑流微測】 半導體產線AI智動化影像檢測工具：建立自動化影像分析系統連動品管關連性
  - 【佳穎精密】 AOI產線不良品影像自動檢測系統：端子、連接器生產品質
  - 【智慧領航教育科技】 AI頭皮檢測與診療方法判別：透過頭皮顯微影像進行問題分類
  - 【華碼數位】 食物熱量影像辨識：利用影像估算食物熱量

# 智慧音箱

- 語音辨識教學

- <https://makerpro.cc/2019/04/use-ros-for-speech-recognition/>

- Google AIY

- <https://www.raspberrypi.com.tw/tag/%E8%AA%9E%E9%9F%B3%E6%96%87%E5%AD%97%E8%BD%89%E6%8F%9B/>

# 關於影像辨識

- 樹莓派影像辨識 (必讀)

- <https://www.youtube.com/watch?v=Cgxsv1riJhl&feature=share>

- 博士實作 (辨識多人)

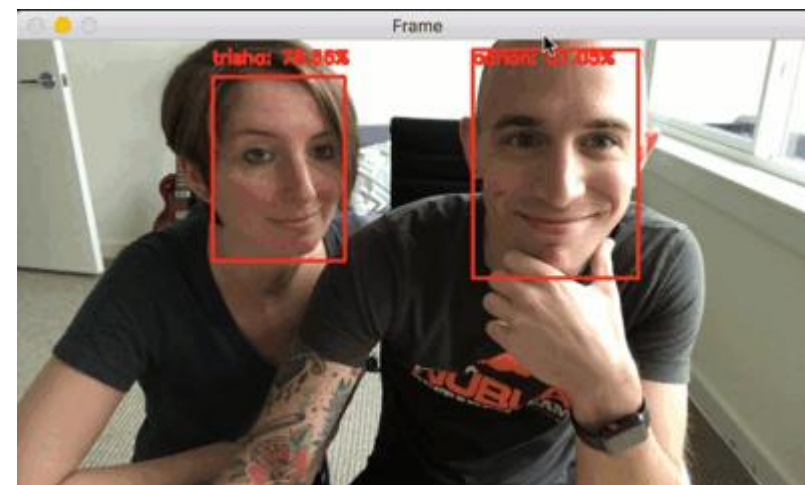
- <https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/?fbclid=IwAR2MYjcl3EsSZKyMWxDkZl1x8Etuq8NSW7qjdv-Jg9F2DzY7y5MLef1e-uc>

- 下載博士的免費電子書

- [https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/?fbclid=IwAR3E\\_uyFLeipDI2314xuMmBh-V6SRCVwzW3lWkO6Mon1NaD-Fbg-0vgOt2g](https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/?fbclid=IwAR3E_uyFLeipDI2314xuMmBh-V6SRCVwzW3lWkO6Mon1NaD-Fbg-0vgOt2g)

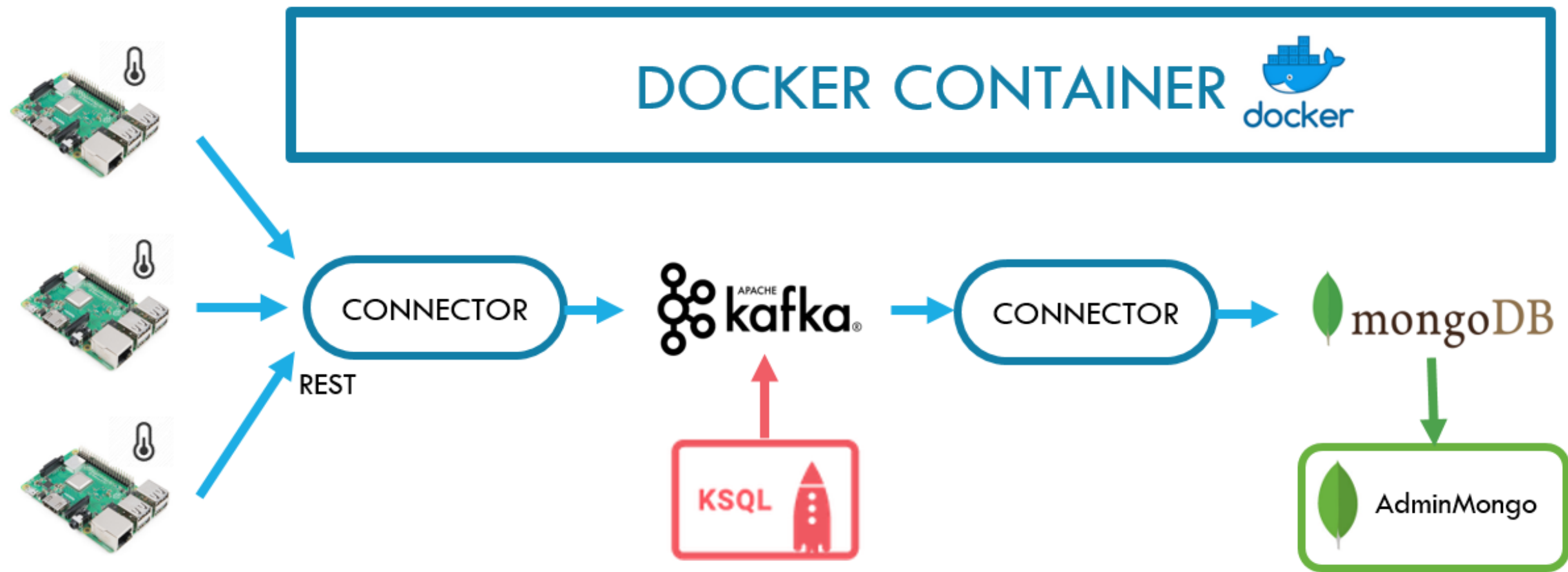
- OpenCV (ver.=4.0.1) + tensorflow 1.13 (pretrain-model)

- [https://omnixri.blogspot.com/2019/05/aigo2opencv.html?m=1&fbclid=IwAR3cw7B7zGFaV9MDgjON3rKhLH3jOWNgpTHjNQyYkUNPn\\_Oc\\_i1DxXUAgO](https://omnixri.blogspot.com/2019/05/aigo2opencv.html?m=1&fbclid=IwAR3cw7B7zGFaV9MDgjON3rKhLH3jOWNgpTHjNQyYkUNPn_Oc_i1DxXUAgO)





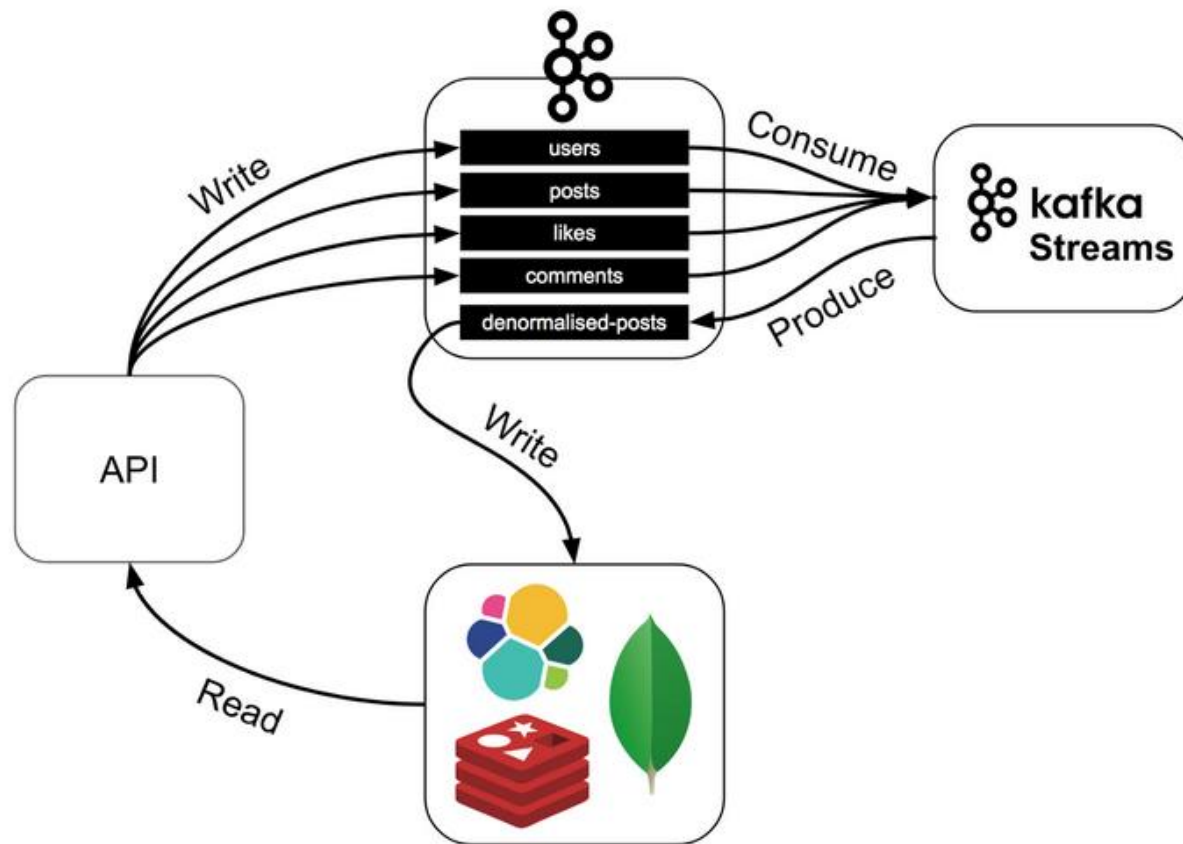
# 溫濕度資料收集與上傳





# KAFKA 介紹

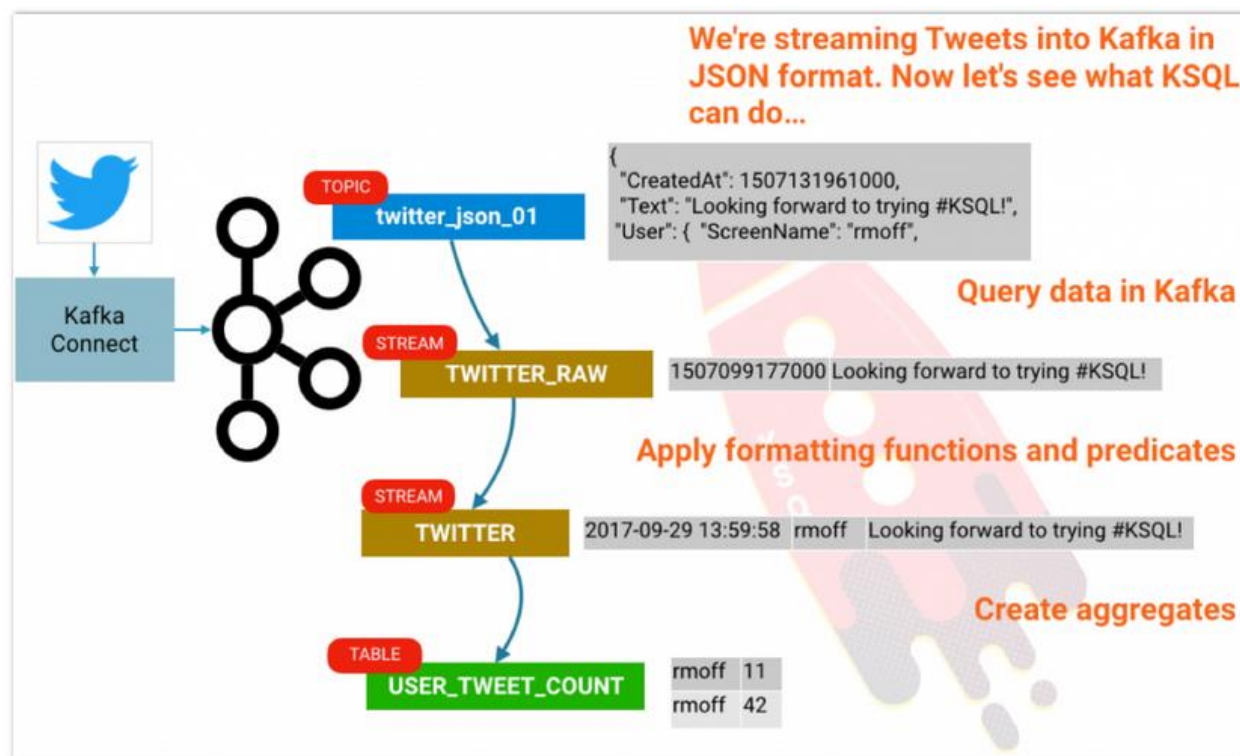
- Message Queue  
(發佈/訂閱)
- 可設定為單機或叢集
- 高吞吐率  
(普通的電腦上可處理 100K筆資料/每秒)
- 支援離線/即時資料處理
- 可設定個別 topic  
(將 streams 資料丟入該 topic 當中)
- 應用:
  - 埋 Code (Page view)
  - 收集機器 Log (CPU、IO使用率)





# KAFKA SQL 介紹

- 與 Kafka 一起運行
- KSQL 是 Apache Kafka 開源的 Streaming SQL 引擎
- 透過進來的資料，可以設定條件過濾與聚合
- 應用：
  - 統計由 **twitter** 進來的用戶每小時推文數量







# DOCKER 環境建置

- 準備一台可以與樹莓派同網段的電腦
- 記憶體: 8GB+
- CPU: 2 cores+
- 硬碟空間: 50 GB



# DOCKER 環境建置

- 安裝 git 套件

```
yum install git
```

- 下載 docker，完成之後需要重新開機

```
curl -sSL https://get.docker.com | sh
```

- 重新登入系統後，請啟動 docker 服務

```
systemctl start docker
```



# DOCKER 環境建置

- 安裝 docker-compose

```
yum install epel-release
```

```
yum install -y python-pip
```

```
pip install docker-compose
```

```
yum upgrade python*
```

```
pip install docker-compose
```



# DOCKER 環境建置

- 下載伺服器docker環境

```
cd /root; git clone https://github.com/orozcohsu/2019-ltu-iot.git
```

- 啟動伺服器環境

```
cd /root/2019-ltu-iot; docker-compose up -d
```

- docker容器介紹





# DOCKER 環境建置

- 進入 kafka container

```
docker exec -it kafka bash
```

- 建立 topic 名為 temperature

```
kafka-topics.sh --create --zookeeper zookeeper:2181 --topic temperature --partitions 1 --replication-factor 1
```

- 建立 topic 名為 humidity

```
kafka-topics.sh --create --zookeeper zookeeper:2181 --topic humidity --partitions 1 --replication-factor 1
```



# DOCKER 環境建置

- 列出有哪些 topic

```
kafka-topics.sh --list --zookeeper zookeeper:2181
```

- 檢視 temperature topic 狀態

```
kafka-topics.sh --describe --zookeeper zookeeper:2181 --topic temperature
```

- 進入ksql-cli container

```
docker exec -it ksql-cli bash
```



# DOCKER 環境建置

- 啟動 ksql

```
ksql http://ksql-server:8088
```

- 查看有哪些 topic

```
SHOW TOPICS;
```

- 檢視 topic=temperature 的資料 (會卡住，等待資料進來)

```
PRINT 'temperature';
```



# DOCKER 環境建置

- 檢視 topic=humidity 的資料資料 (會卡住，等待資料進來)

```
PRINT 'humidity';
```

- 建立一個 stream 為 temperature\_stream

```
CREATE STREAM temperature_stream (device_id varchar, timestamp varchar, \  
temperature double, rd varchar) WITH(kafka_topic='temperature', value_format='json');
```

- 建立一個 stream 為 humidity\_stream

```
CREATE STREAM humidity_stream (device_id varchar, timestamp varchar, \  
humidity integer, rd varchar) WITH(kafka_topic='humidity', value_format='json');
```





# DOCKER 環境建置

- 查看所有 stream

```
SHOW STREAMS;
```

- 查看 temperature\_stream 欄位資訊

```
DESCRIBE temperature_stream;
```

- 查看 humidity\_stream 欄位資訊

```
DESCRIBE humidity_stream;
```



# DOCKER 環境建置

- 建立 stream 名為 temp\_humidity\_enriched，內容是將 temperature\_stream 和 humidity\_stream 做 JOIN

```
CREATE STREAM temp_humidity_enriched AS SELECT t.device_id, t.timestamp, \  
temperature, humidity, t.rd FROM temperature_stream t JOIN humidity_stream h \  
WITHIN 4 HOURS ON t.rd= h.rd;
```



# DOCKER KAFKA REST PROXY 測試

- 登入 Kafka REST 容器

```
docker exec -it restproxy bash
```

- 查看目前有多少個 topic

```
curl http://localhost:8082/topics
```

- 查看 temperature 相關資訊

```
curl http://localhost:8082/topics/temperature
```



# DOCKER KAFKA REST PROXY 測試

- 執行測試溫度測試

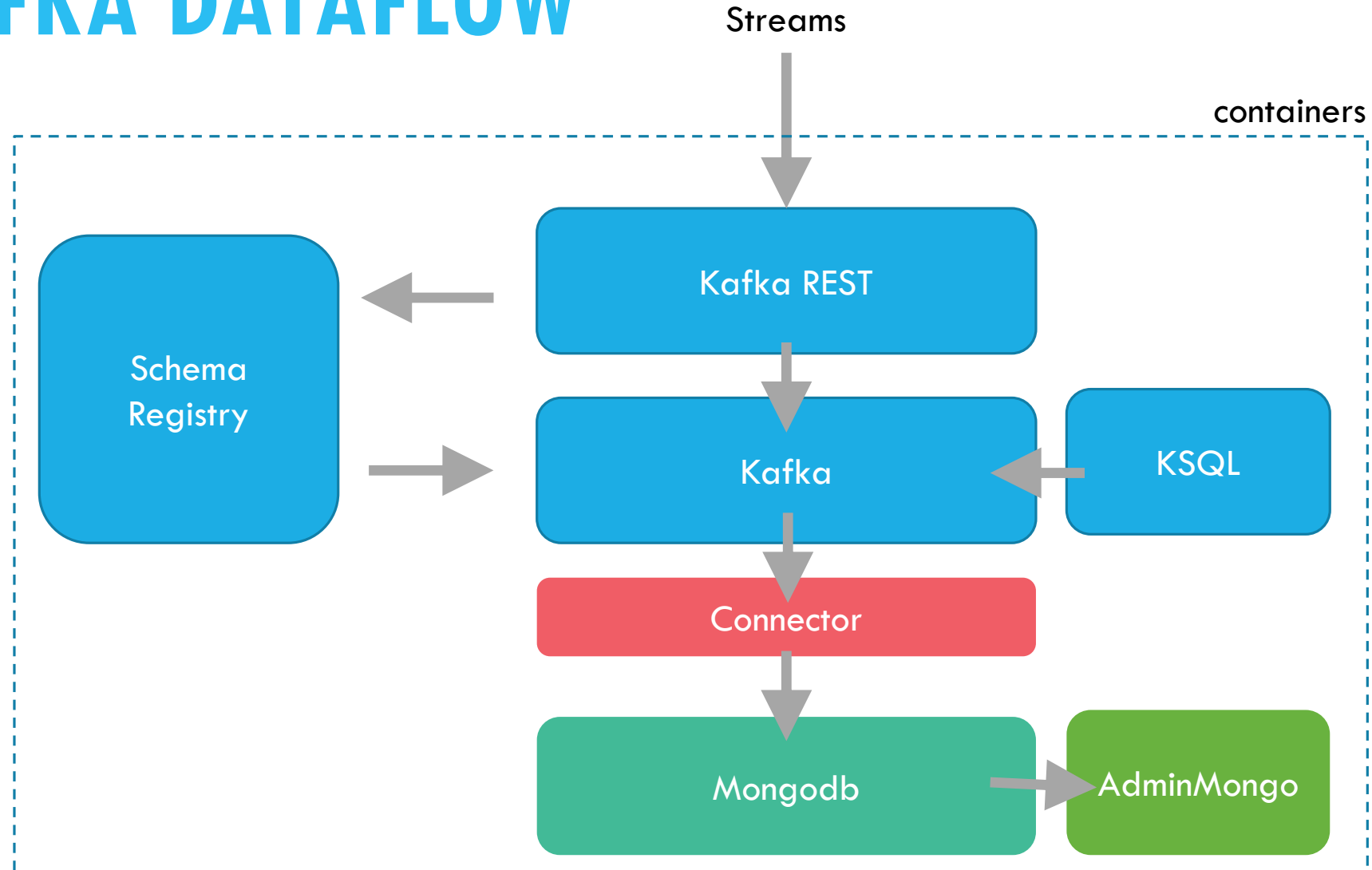
```
curl -X POST -H "Content-Type: application/vnd.kafka.json.v2+json" --data '{"records":[{"value":{"device_id": "999", \
"timestamp": "2019-07-02 12:26:44", "Temperature": 99}}]}' http://localhost:8082/topics/temperature
```

```
curl -X POST -H "Content-Type: application/vnd.kafka.json.v2+json" --data '{"records":[{"value":{"device_id": "999", \
"timestamp": "2019-07-02 12:26:44", "Humidity": 10}}]}' http://localhost:8082/topics/humidity
```

- 觀察 Adminmongo 是否有資料進來



# KAFKA DATAFLOW





# MONGODB

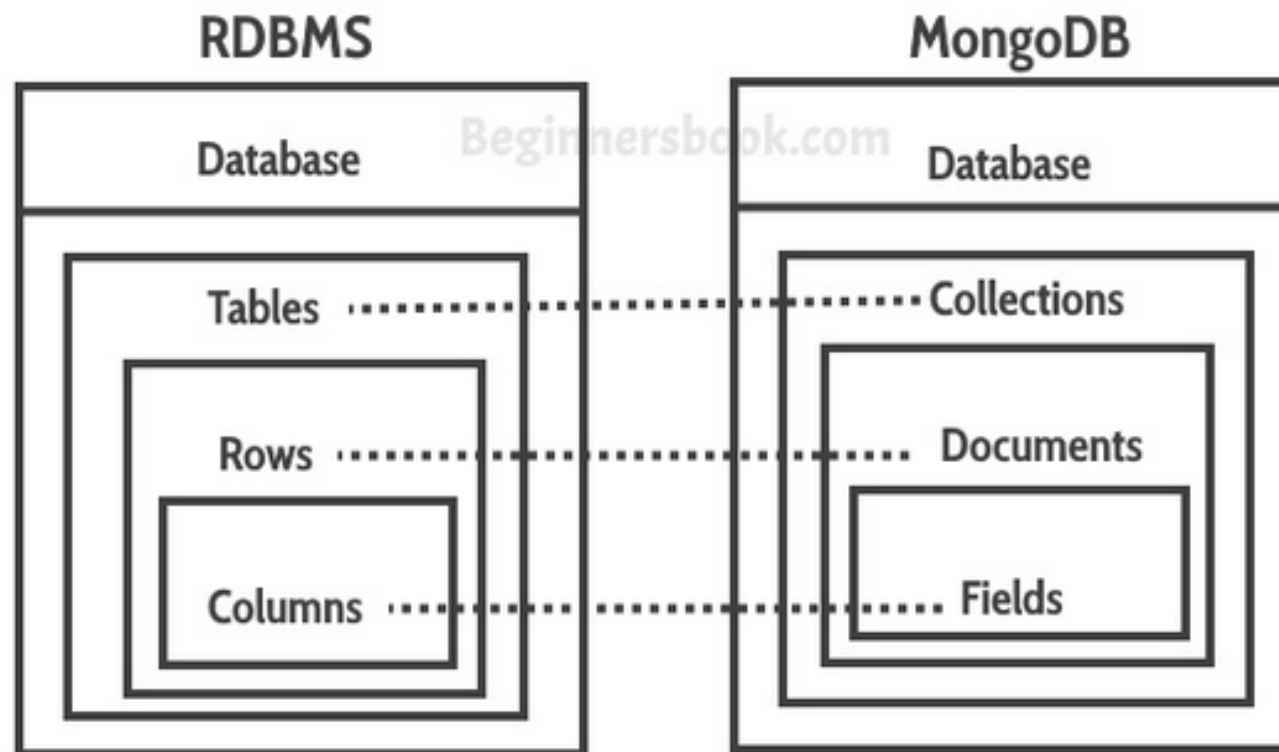
- MongoDB 屬於文件資料庫 ( Document Database ) ，以文本方式儲存
- 可設定為單機或叢集
- 本身沒有 Schema ，所以在架構上很好調整
- 資料樣貌

```
{  
  _id: "948794777",  
  name: "Robby",  
  age: 30,  
  email: "Robby",  
  skill: [  
    'javascript',  
    'java'  
  ]  
}
```



# MONGODB

- 與結構化資料庫的術語比較





# ADMINMONGO

- Mongodb 網頁查詢平台 (連線輸入: mongodb://mongodb:27017)

 adminMongo 🔌 Connections

MongoDB Connections

No connections. Please add one below

Connection name	Connection string	Connection options (See <a href="#">docs</a> )	Action
ltu	<div><div></div><div><code>mongodb://mongodb:27017</code></div></div>	<div>1 { }</div>	<div>Add connection</div>







# ADMINMONGO

Database Objects

ltu

test

data\_20190630

temp\_humidity

Connections

ltu

Monitoring

Connections

Database: test / Collection - data\_20190630

Home / ltu (connection) / test (database) / data\_20190630 (collection)

New document

Indexes

Search

Query

Reset

Docs per page

5

Total records: 42

Delete all

<pre>{   "_id": "5d1b8a6cbe567be7cc90350d",   "device_id": "001",   "timestamp": "2019-07-03 00:36:58",   "Temperature": 27,   "Humidity": 65 }</pre>	<div>DeleteLinkEdit</div>
<pre>{   "_id": "5d1b8a6cbe567be7cc90350e",   "device_id": "001",   "timestamp": "2019-07-03 00:37:06",   "Temperature": 27,   "Humidity": 66 }</pre>	<div>DeleteLinkEdit</div>
<pre>{   "_id": "5d1b8a6cbe567be7cc90350f",   "device_id": "001",   "timestamp": "2019-07-03 00:37:22",   "Temperature": 27,   "Humidity": 66 }</pre>	<div>DeleteLinkEdit</div>



# DOCKER 環境建置

- 開啟 jupyter 網頁

<http://IP:8889>

- 開啟 producer 程式
  - 新增 rd 亂數資料

**kafka\_producer.ipynb**

- 開啟 consumer 程式

**kafka\_consumer\_temperature.ipynb**

**kafka\_consumer\_humidity.ipynb**

**kafka\_consumer.ipynb**



# 建立DATABASE

- Database = Itu\_demo

The screenshot displays the adminMongo web interface. On the left, a dark sidebar contains the 'adminMongo' logo and a 'Database Objects' section. This section lists three databases: 'Itu', 'config', and ' Itu\_demo'. The 'config' database is expanded, showing 'system.sessions'. The ' Itu\_demo' database is also expanded, showing 'data\_2019063020'. The 'test' database is expanded, showing 'data\_20190630' and 'temp\_humidity'. Below this is a 'Connections' section with 'Itu' listed. The main content area on the right has a header 'Database: Not s'. Below this is a 'Server status' section with the following information: Version: 4.0.10, Host: mongod, PID: 1, Uptime: 28 minutes, and Local time: Tue Jul 02 2019 16:59:01 GMT+0000 (Coordinated Universal Time). Below the server status is a 'Database maintenance' section. It includes a 'Create new database' form with a text input containing ' Itu\_demo' and a green 'Create' button. Below that is a 'Delete database' form with a dropdown menu showing 'config' and a red 'Delete' button.



# 調整 KAFKA\_CONSUMER 時間

- Pyspark 要用這個資訊 ETL
- 若配合 ETL 可以考慮建立 collection
  - 例如: collection = db.data\_20190630

```
jupyter kafka_consumer (read only)
File Edit View Insert Cell Kernel Help
+ - * / < > Run Code



In [*]: from kafka import KafkaConsumer
import sys, json, pymysql, pymongo

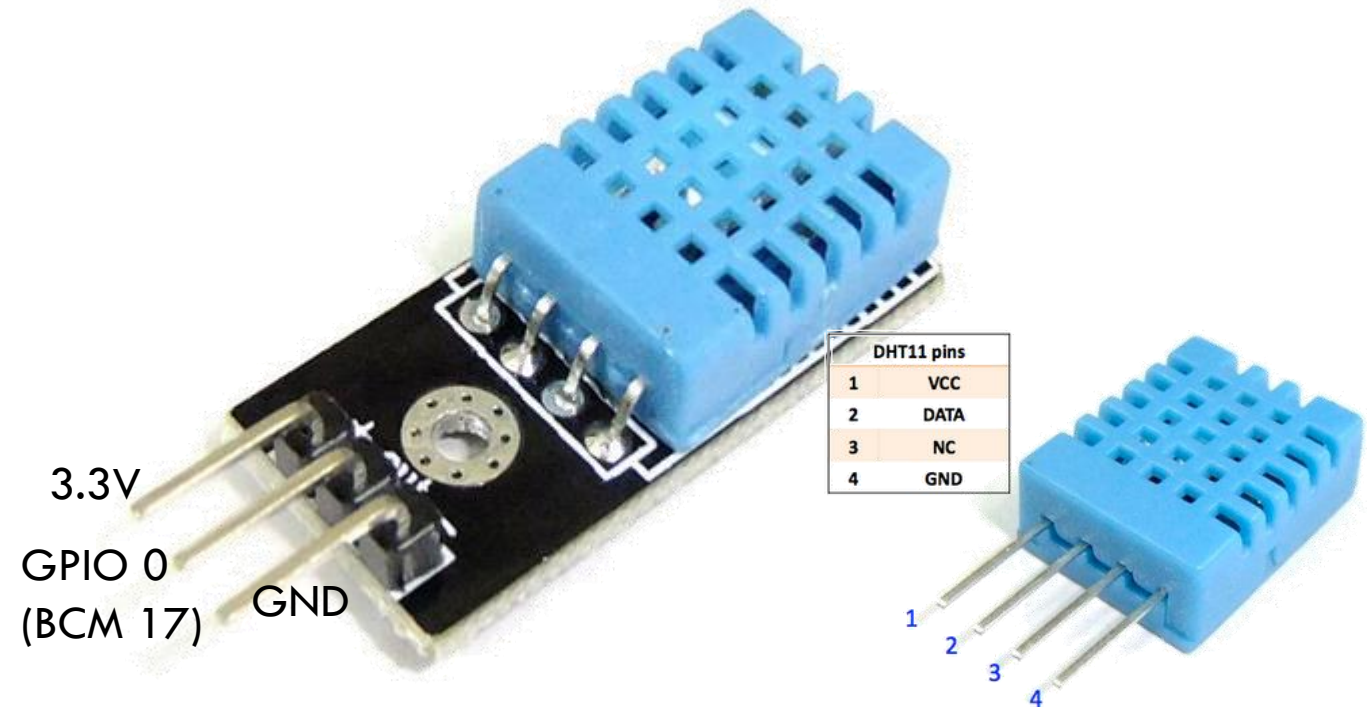
if __name__ == "__main__":
    # 與 MongoDB連線
    client = pymongo.MongoClient(host="mongodb", port=27017)
    # 指定為 test 資料庫
    db = client.ltu_demo
    # 指定 temp_humidity 集合, MongoDB的每個資料庫又包含許多集合(collection), 類似於關聯性資料庫中的表
    collection = db.data_2019063020

    # 設定要連線到Kafka集群的相關設定, 產生一個Kafka的Consumer的實例
    consumer = KafkaConsumer(
        # 指定Kafka集群伺服器
        bootstrap_servers=["kafka:9092"],
        # ConsumerGroup的名稱, 可以不指定
        #group_id="cg_001",
        # 指定msgKey的反序列化器, 若Key為None, 無法反序列化
        # key_deserializer=bytes.decode,
        # 指定msgValue的反序列化器
        #value_deserializer=bytes.decode,
        value_deserializer=lambda m: json.loads(m.decode('ascii')),
        # 是否從這個ConsumerGroup尚未讀取的partition / offset開始讀
        auto_offset_reset="earliest",
    )
```



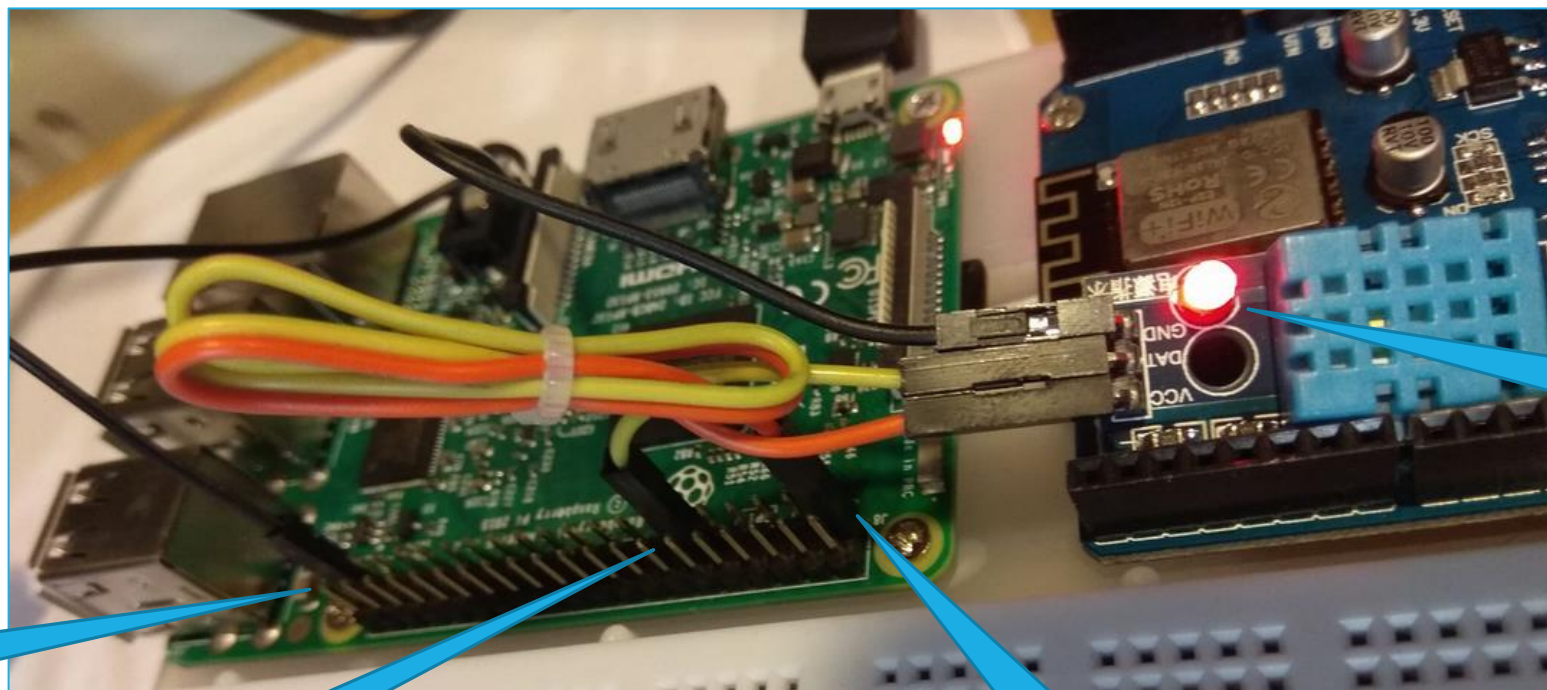
# 溫濕度資料收集與上傳

DHT11	DHT22
	
3 - 5.5V	3.3 - 6V
20 - 80 %	0 - 100 %
5%	5%
0 - 50 °C	-40 - 125 °C
+/- 2 °C	+/- 0,5 °C
1s	2s





# 溫濕度資料收集與上傳



第39隻腳  
(接地)

第11隻腳  
(GPIO0)

第1隻腳  
(3.3V)

指示燈





# 樹莓派上執行

- 開啟 consumer 程式，並修改
  - database = ltu\_demo
  - collection = db.data\_20190630
- python sensor\_kafka.py

```
if __name__ == "__main__":  
    # 與 MongoDB連線  
    client = pymongo.MongoClient(host="mongodb", port=27017)  
    # 指定為 test 資料庫  
    db = client.ltu_demo  
    # 指定 temp_humidity 集合, MongoDB的每個資料庫又包含許多集  
    collection = db.data_20190630
```

```
root@raspberrypi:/home/pi/2019-ltu-airbox# python sensor_kafka.py  
temperature updated  
humidity updated  
wait next 5 seconds  
temperature updated  
humidity updated  
wait next 5 seconds  
temperature updated  
humidity updated  
wait next 5 seconds  
temperature updated  
humidity updated  
wait next 5 seconds
```

# SPARK (PYSPARK)



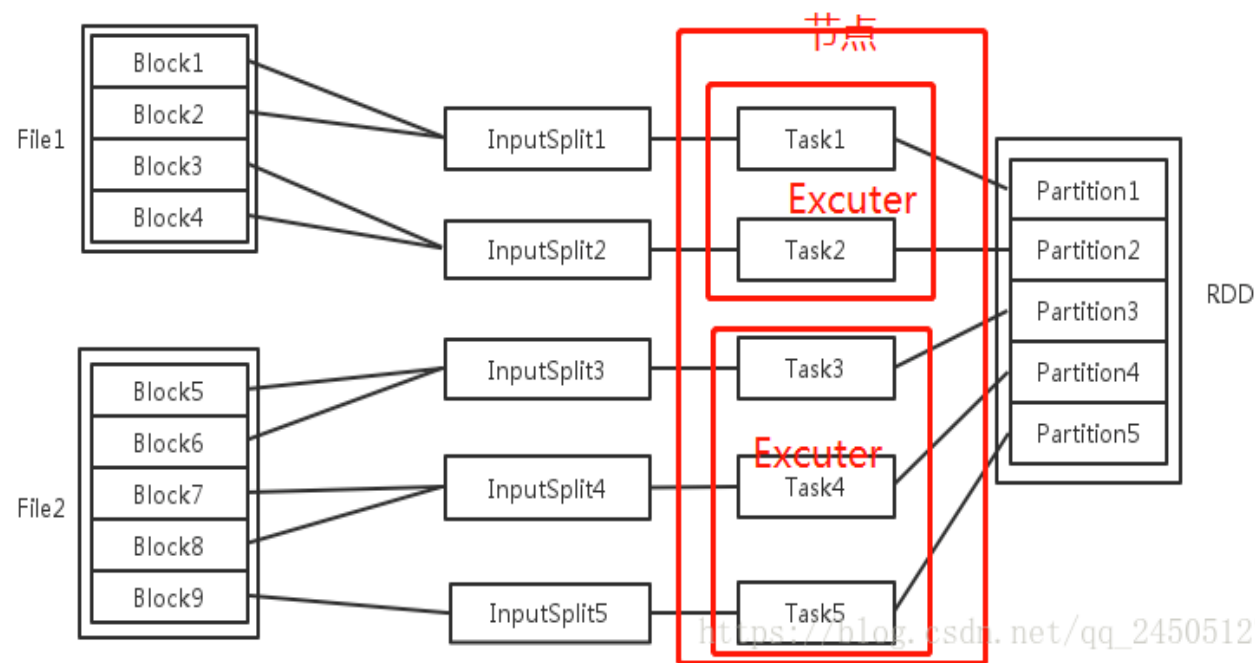
- 分散式運算框架，資料透過記憶體叢集完成
- 為 Hadoop MapReduce 的取代品
- 能處理大於記憶體叢集總和之資料量
- 支援 Spark (Scala)、PySpark、SparkR 工具
- Spark SQL、Spark MLlib、Spark Streaming、Spark GraphX





# RDD

- 資料儲存於記憶體叢集之資料格式
- 透過 Transformation 轉變為一個新的 RDD
- 透過 Action 將 RDD 計算後，求得一個新的值 (純量或陣列)





# PYTHON (PANDAS 套件)

- Python 常用套件之一
- 適合用來整理資料
- 資料格式為 DataFrame，與 Spark SQL、RDD 可以互轉



# 執行 PYSPARK

- 開啟 Spark jupyter

<http://IP:8890>

- 執行Pyspark程式

**pyspark-read-mysql.ipynb**

**pyspark\_mongodb\_read1.ipynb**

**pyspark\_mongodb\_read2.ipynb**

**pyspark\_mysql\_mongodb\_join\_etl.ipynb**



# 查看最終結果

Database: *Not selected* / Collection - *Not selected*

Server status

Version: 4.0.10

Host: mongodb

PID: 1

Uptime: 34 minutes

Local time: Tue Jul 02 2019 17:05:01 GMT+0000 (Coordinated Universal Time)

Connection statistics

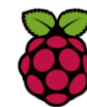
config		
system.sessions	Storage: 396 Bytes	Documents: 4
ltu_demo		
data_2019063020	Storage: 20.2 KB	Documents: 182
<u>final</u>	Storage: 32.2 KB	Documents: 179
test		
data_20190630	Storage: 4.66 KB	Documents: 42
temp_humidity	Storage: 4.44 KB	Documents: 40

Database maintenance

Create new database

Create





# 查看最終結果

Database: **ltu\_demo** / Collection - final

Home / ltu (connection) / ltu\_demo (database) / final (collection)

New document

Indexes

Search

Query

Reset

Docs per page

5

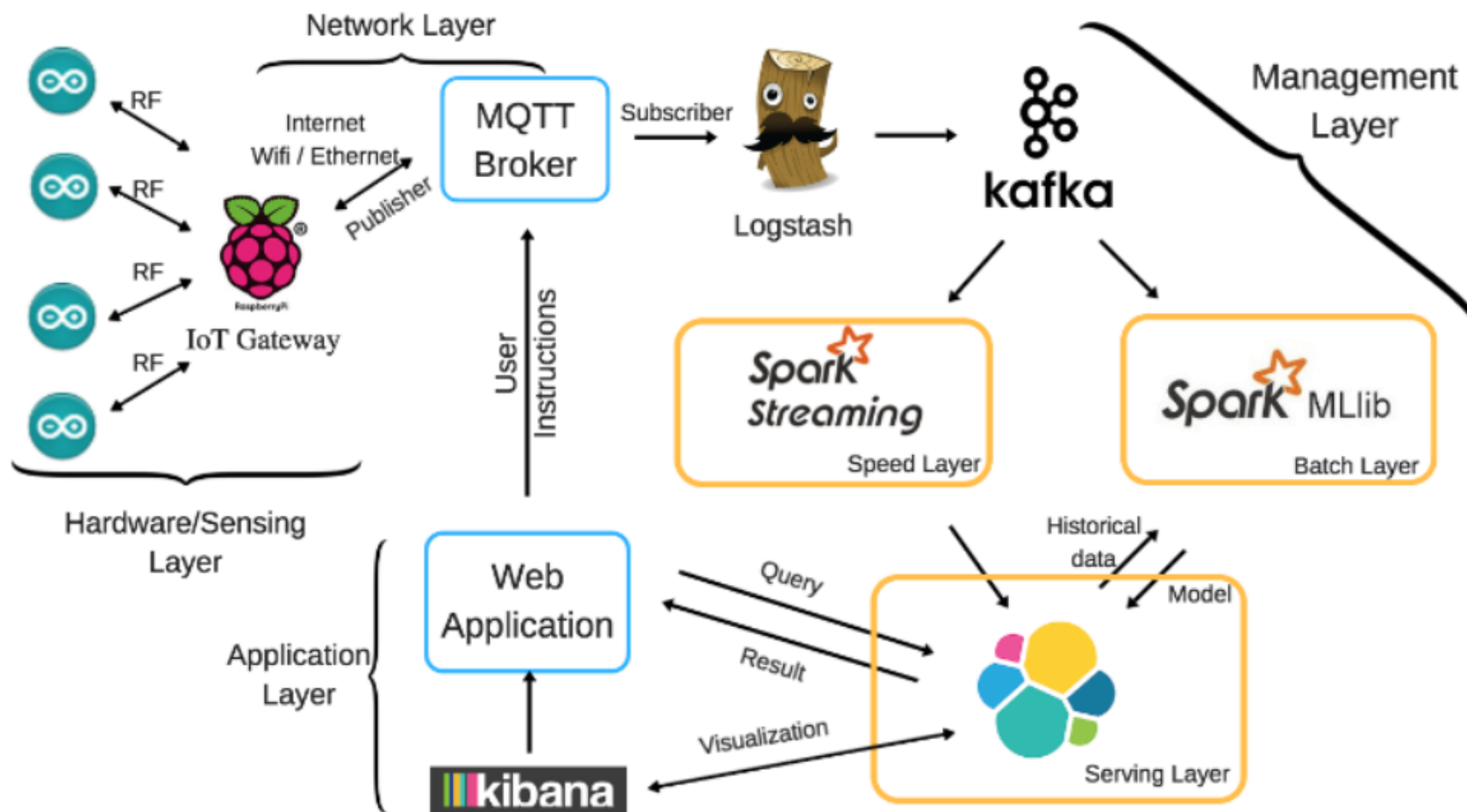
Total records: 179

Delete all

<pre>{   "_id": "5d1b8d40be567be7cc90353b",   "device_id": "001",   "location": "Taichung",   "description": "Semiconductor-Machine-01",   "Humidity": 66 }</pre>	<div>Delete</div> <div>Link</div> <div>Edit</div>
<pre>{   "_id": "5d1b8d40be567be7cc90353c",   "device_id": "001",   "location": "Taichung",   "description": "Semiconductor-Machine-01",   "Humidity": 66 }</pre>	<div>Delete</div> <div>Link</div> <div>Edit</div>
<pre>{   "_id": "5d1b8d40be567be7cc90353d",   "device_id": "001",   "location": "Taichung",   "description": "Semiconductor-Machine-01",   "Humidity": 64 }</pre>	<div>Delete</div> <div>Link</div> <div>Edit</div>



# 大數據物聯網系統架構回顧 (總結)



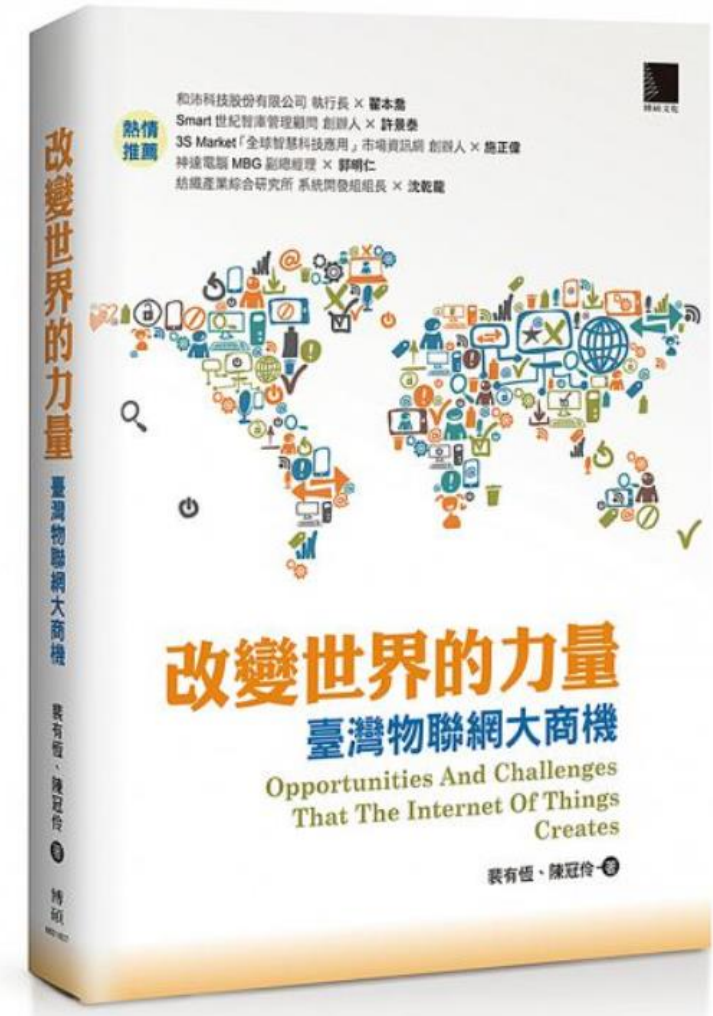
# 作業

- 何謂 NodeRED? 如何與樹莓派應用?
- <https://oranwind.org/-aws-she-ding-node-red-lai-shi-jue-hua-xian-shi-gan-ce-zi-liao-jiao-xue/>



# 參考資料

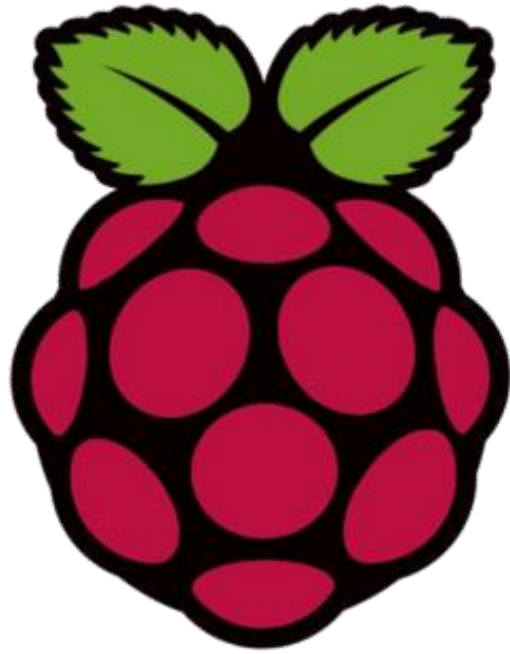
- ✓ AIoT產業-多組學的精準醫療
- ✓ AIoT產業-從 Google I/O 跟微軟 Build 談 Edge computing 發展的迫切性
- ✓ AIoT產業-微軟和 Google 為何都在台灣佈局人工智慧-微軟篇
- ✓ AIoT產業-Google 的人工智慧將在台灣往下一個世代開始佈局
- ✓ AIoT產業-AIoT 服務沒有資安，會出這三個大問題



<http://rich4innovation.blogspot.tw>



# 享受開源的樹莓派應用



歡迎聯繫數數科技資訊社  
[orozcohsu@hotmail.com](mailto:orozcohsu@hotmail.com)