
Auto-check 無人商店

— Team 20 吳岱容 杜葳葳 陳騰鴻 —

Outline

- Introduction
- Demo
- 流程架構
- 實作介紹

Introduction

Motivation

- for 顧客：減去等候結帳的時間、提升購物體驗、個人化推薦
- for 商家：降低人力成本、提升庫存管理效率

Introduction

「無人商店雲端系統」，以「個人帳號」為中心的雲端服務，為顧客創造客製化的購物體驗。此外，系統會監控架上存貨，即時通知供貨商補貨，確保顧客都能買到想要的商品。最後，我們利用過去的購買紀錄，建置個人化的推薦系統。

Features

顧客

- 拍照、QRcode註冊
- 個人購物紀錄頁面
- 掃臉直接購物
- 客製化商品推薦

Auto-check
無人商店

商家

- 庫存管理
- 會員資料管理
- E-mail自動通知廠商補貨
- 購物流程全自動化、店裡面不用雇店員

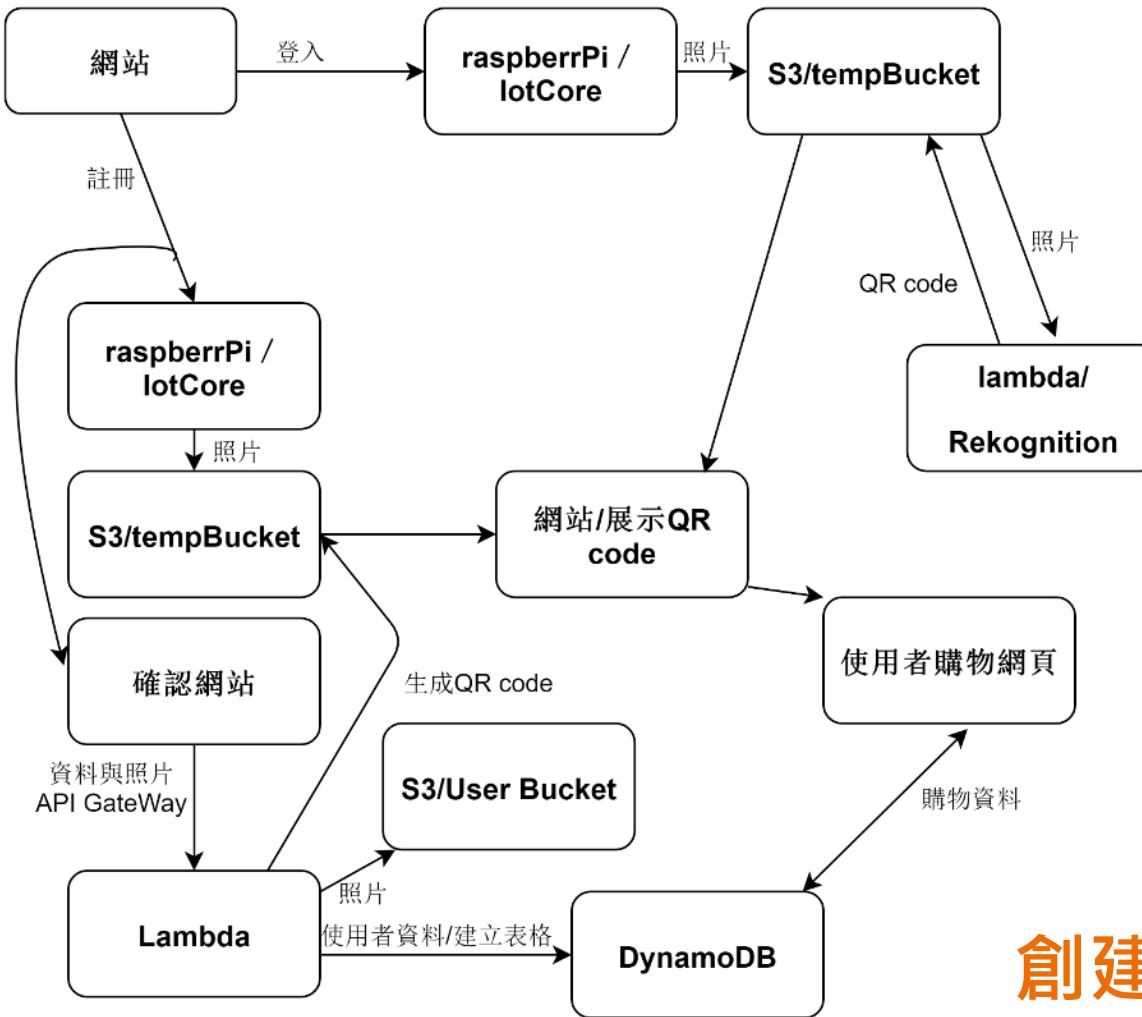
Demo

CLOUD PROGRAMMING
FINAL PROJECT
AUTO-CHECK

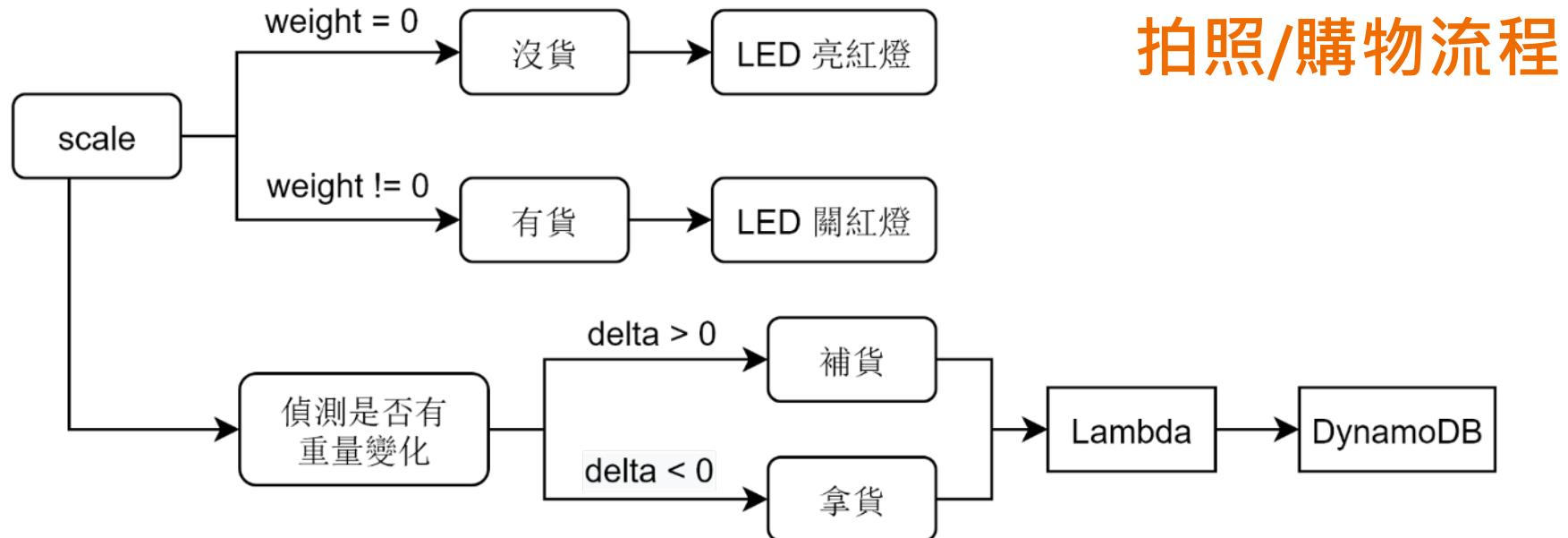
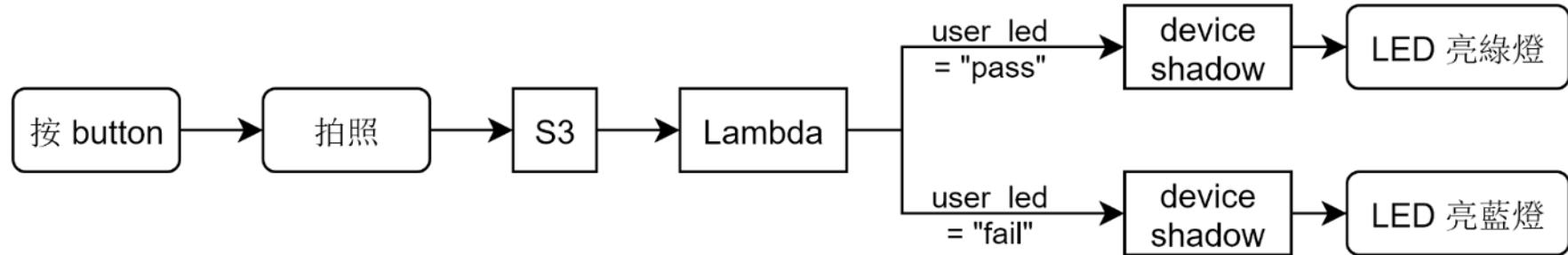
TEAM 20



流程架構



註冊/登入
創建個人頁面



實作介紹

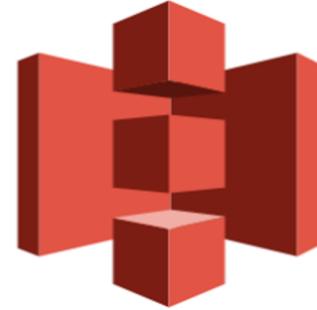
API Gateway

- 主要工作：
 - 傳遞資訊（**serverless**架構），負責和DynamoDB、Lambda溝通
 - 共用了三個資源分別接到兩個Lambda Function及DynamoDB
- 優點：
 - 沒有server的架構，維護上有便利之處
 - api gateway及lambda減少**server idle**的cost
- 缺點：
 - 功能限制較大



S3

- 會員 (user bucket)
 - 照片名稱是 user_id
- 暫存 (temp bucket)
 - 按button → Pi → S3
 - 會員註冊mode: → Lambda → S3/DynamoDB
 - 拿商品mode: → Lambda → Recognition 判斷是不是會員/是哪個會員



S3

會員 (user bucket)

The screenshot shows the Amazon S3 console interface for a bucket named "final-order-bucket-wei". The top navigation bar includes links for "Amazon S3", "final-order-bucket-wei", "物件 (12)", "属性", "許可", "指標", "管理", and "存取點". Below the navigation is a toolbar with buttons for "建立資料夾" (Create Folder) and "上傳" (Upload). A search bar is present. The main area displays a table of 12 objects, with two specific files highlighted by orange boxes: "chaewon.jpg" and "chaeyeon.jpg". The table columns include "名稱" (Name), "類型" (Type), "上次修改時間" (Last Modified), "大小" (Size), and "儲存體類別" (Storage Class).

名稱	類型	上次修改時間	大小	儲存體類別
chaewon.jpg	jpg	2021年6月17日 pm2:14:20 CST	145.6 KB	標準
chaeyeon.jpg	jpg	2021年6月17日 pm2:14:19 CST	84.7 KB	標準

照片名稱是user_id

暫存 (temp bucket)

The screenshot shows the Amazon S3 console interface for a bucket named "final-target-wei". The top navigation bar includes links for "Amazon S3", "final-target-wei", "物件 (4)", "属性", "許可", "指標", "管理", and "存取點". Below the navigation is a toolbar with buttons for "建立資料夾" (Create Folder) and "上傳" (Upload). A search bar is present. The main area displays a table of 4 objects, with two specific files highlighted by orange boxes: "jj.jpeg" and "mi.jpg". The table columns include "名稱" (Name), "類型" (Type), "上次修改時間" (Last Modified), "大小" (Size), and "儲存體類別" (Storage Class).

名稱	類型	上次修改時間	大小	儲存體類別
jj.jpeg	jpeg	2021年6月17日 pm2:46:16 CST	47.9 KB	標準
mi.jpg	jpg	2021年6月18日 pm8:38:42 CST	41.5 KB	標準

照片名稱隨意

DynamoDB

- 會員資料 (user)
 - key: user_id, value: email
- 每個會員的購物紀錄 ([user_id])
 - key: timestamp, value: item_id, count, isPay
- 目前存貨 (stock)
 - key: item_id, value: price, stock
- 每個商品的記錄 ([item_id])
 - key: timestamp, value: count

	名稱	狀態	分區鍵	排序鍵	索引
1	chaewon	作用中	timestamp (數字)	-	0
2	item_1	作用中	timestamp (數字)	-	0
3	item_2	作用中	timestamp (數字)	-	0
4	minju	作用中	timestamp (數字)	-	0
5	nako	作用中	timestamp (數字)	-	0
6	stock	作用中	item_id (字串)	-	0
7	user	作用中	user_id (字串)	-	0



DynamoDB 會員資料

會員資料 (user)

key: user_id, value: email

	user_id	email
	chaewon	chaewon@gmail.com
	minju	minju@gmail.com
	nako	dora1227@pchome.com.tw

每個會員的購物紀錄 ([user_id])

key: timestamp, value: item_id, count, isPay

掃描 [資料表] nako: timestamp
新增篩選條件
開始搜尋

	timestamp	count	isPay	item_id
	1624118951	1	false	item_1
	1624167923	3	false	item_2
	1624168373	2	false	item_1
	1624168377	1	false	item_1

DynamoDB 貨品資料

目前存貨 (stock)

key: item_id, value: price, stock

	item_id ⓘ	price	stock
<input type="checkbox"/>	item_1	20	0
<input type="checkbox"/>	item_2	10	3

每個商品的記錄 ([item_id])

key: timestamp, value: count

掃描 [資料表] item_1: timestamp

+ 新增篩選條件

開始搜尋

	timestamp ⓘ	count
<input type="checkbox"/>	1624117524	-2
<input type="checkbox"/>	1624118951	-1
<input type="checkbox"/>	1624167383	3
<input type="checkbox"/>	1624168324	3
<input type="checkbox"/>	1624168373	-2
<input type="checkbox"/>	1624168377	-1

SNS

- 當架上某個商品**沒貨**的時候寄信通知廠商
 - IoT core → Lambda → DynamoDB → **if (stock=0)** → SNS

AWS Notification Message 外部 收件匣 × 🖨️ ✉️

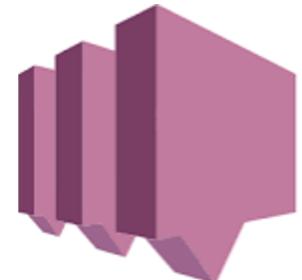
 AWS Notifications <no-reply@sns.amazonaws.com> 6月18日 週五 下午8:17 (17 小時前) ☆ ↶ ⋮

寄給我 ▾

Item item_1 is out of stock, please replenish the stock.

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:011825331023:final:28b9d5f2-b342-4c63-90bb-88a8e3e593f1&Endpoint=wwdu@gapp.nthu.edu.tw>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

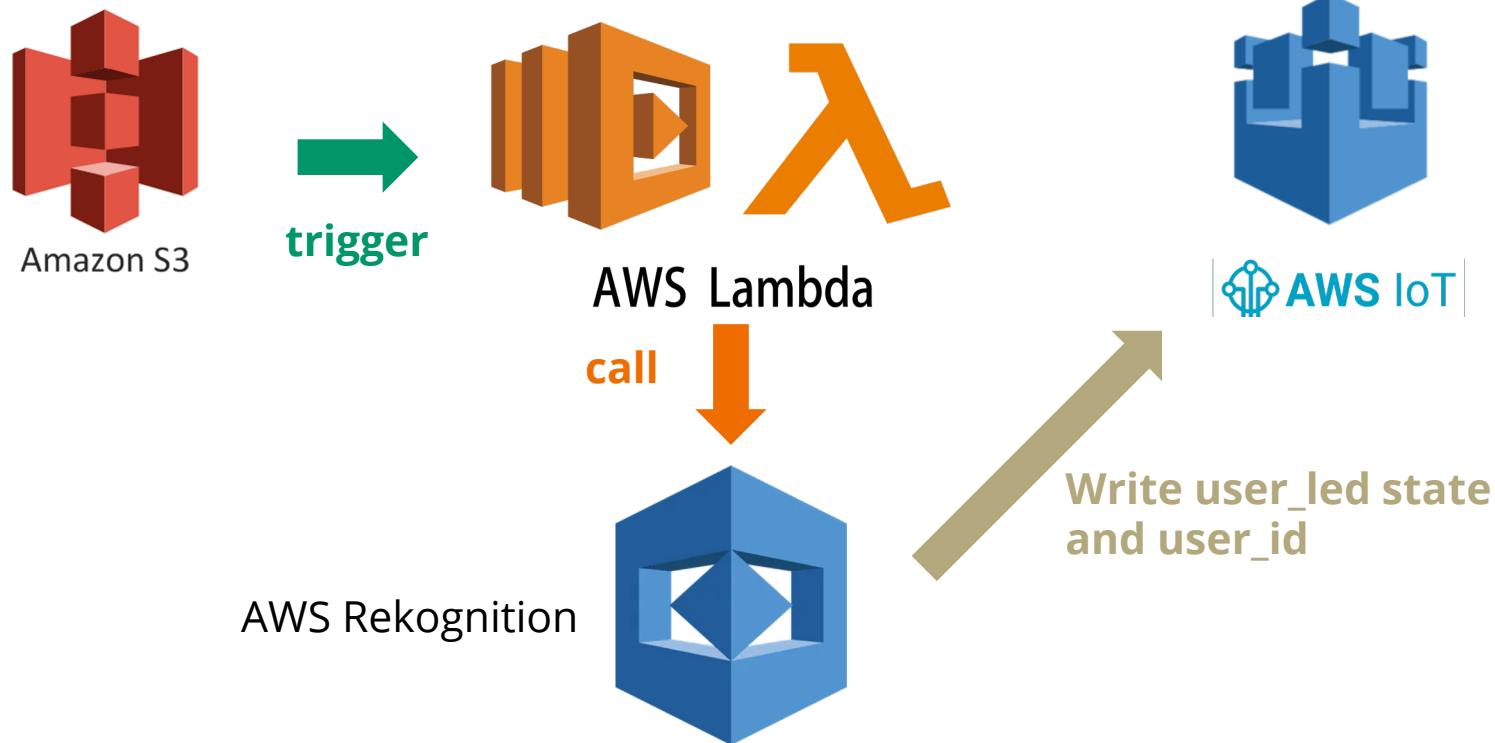


AWS Rekognition



- 主要工作：
 - 登入時的人臉識別
 - 購物時的人臉識別
- 演算法：
 - 從 trigger event 取得目標辨識照片 **T**
 - 依序將每一張會員 bucket 的照片 **S** 與 **T** 作比較：在 **T** 中尋找 **S** 裡最大的臉
 - 若成功配對一組臉，從 **S** 的檔案名稱提取 user_id

Lambda: 使用者人臉識別



Lambda: 庫存追蹤和購買紀錄



- **IoT Rule** 在 Device Shadow 更新時 trigger Lambda
 - `SELECT * FROM '$aws/things/final/shadow/update/accepted'`
- Lambda 從 Device Shadow 讀取資訊並寫入 databases
- 若缺貨，Lambda 使用 SNS 服務寄通知信

AWS IoT core

- Device Shadow
 - **AWS** (S3 → Recognition → Lambda) → **Pi**
 - **user_id**: 會員的id
 - **user_led**: 控制燈號(green/blue)
 - **Pi** → **AWS** (Lambda → DynamoDB)
 - **need_update**: 紀錄需不需要寫入 database
 - **user_id**: 會員的id(或店員的id)
 - **item_id**: 買的商品(或補貨的商品)
 - **count**: 買的數量(或補貨的數量)
 - 註: 補貨數值是+, 買貨數值是-



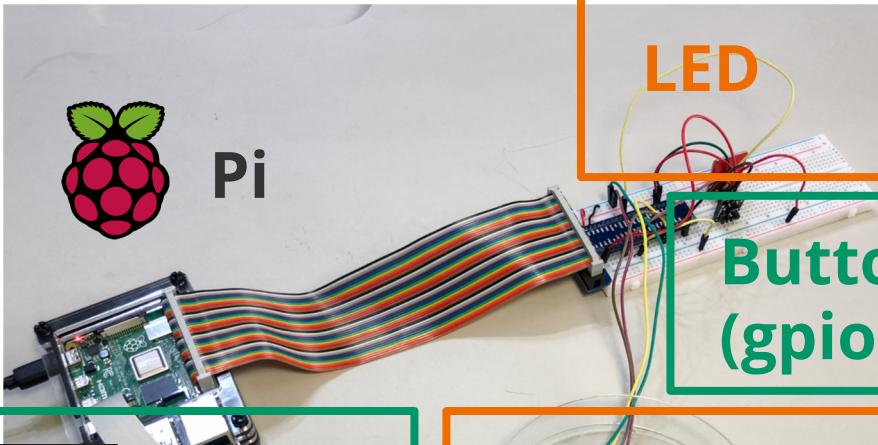
AWS IoT core

影子狀態：

```
{  
    "desired": {  
        "welcome": "aws-iot",  
        "拿(補)商品名稱" "item_id": "item_1",  
        "拿(補)多少商品" "count": 2,  
        "user_led": "pass",           控制是否為會員的燈號  
        "need_update": "False",      是否需要更新DynamoDB  
        "user_id": "nako"           會員的名字  
    },  
    "reported": {  
        "welcome": "aws-iot",  
        "item_id": "item_1",  
    }  
}
```



Raspberry Pi



Button
(gpiozero)

```
RED_LED_PIN = 16
BLUE_LED_PIN = 20
GREEN_LED_PIN = 25
PWM_FREQ = 200

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(RED_LED_PIN, GPIO.OUT)
GPIO.setup(BLUE_LED_PIN, GPIO.OUT)
GPIO.setup(GREEN_LED_PIN, GPIO.OUT)

red_pwm = GPIO.PWM(RED_LED_PIN, PWM_FREQ)
red_pwm.start(0)
blue_pwm = GPIO.PWM(BLUE_LED_PIN, PWM_FREQ)
blue_pwm.start(0)
green_pwm = GPIO.PWM(GREEN_LED_PIN, PWM_FREQ)
green_pwm.start(0)

def setColor(r=0, g=0, b=0):
    red_pwm.ChangeDutyCycle(100-int(r/255*100))
    blue_pwm.ChangeDutyCycle(100-int(b/255*100))
    green_pwm.ChangeDutyCycle(100-int(g/255*100))

setColor(0, 0, 0)
```

```
if button_is_pressed:
    camera = PiCamera()
    camera.resolution = (1024, 768)
    camera.start_preview()
    sleep(2)
    camera.capture("image.jpg")
    camera.close()
```

Camera
(picamera)



Scale
(HX711)

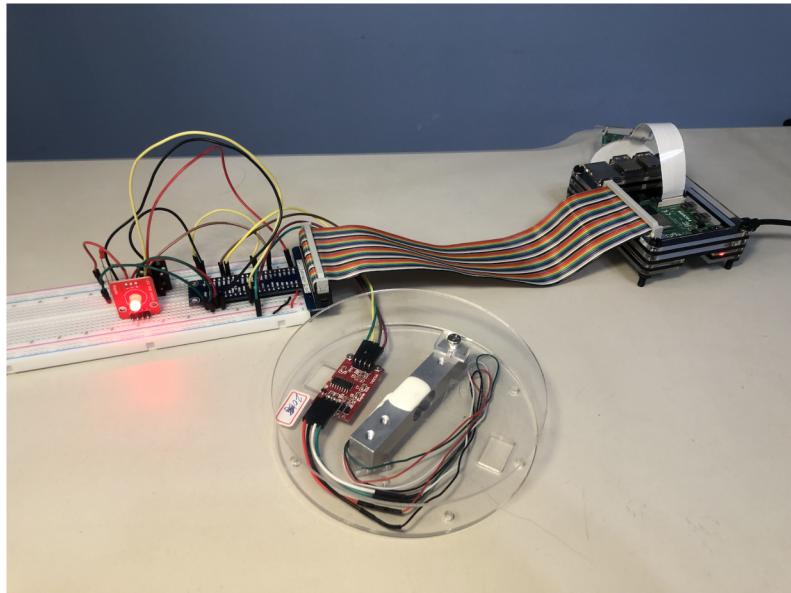
```
EMULATE_HX711=False
referenceUnit = 1
hx = HX711(5, 6)
hx.set_reading_format("MSB", "MSB")
hx.set_reference_unit(referenceUnit)
hx.reset()
hx.tare()
print("Tare done! Add weight now...")

val = hx.get_weight(5)
hx.power_down()
hx.power_up()
time.sleep(0.1)
```

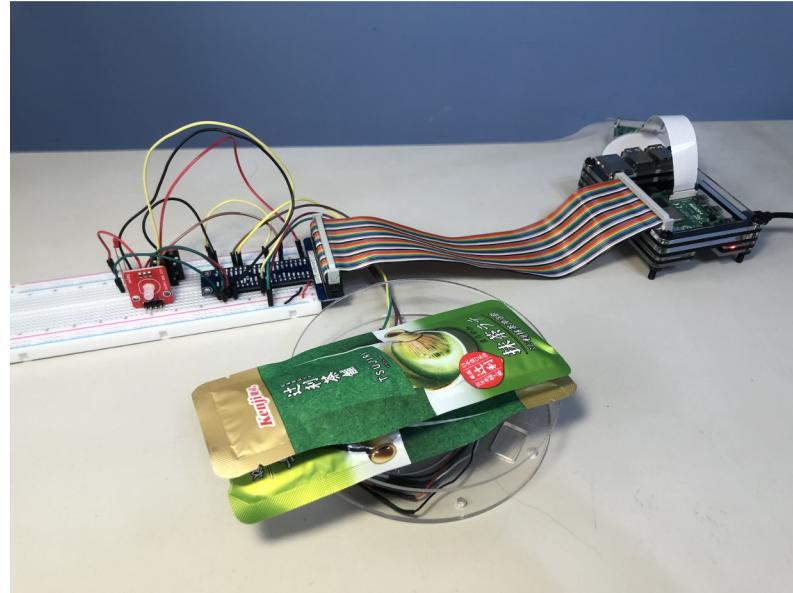
```
EMULATE_HX711=False
referenceUnit = 1
hx = HX711(5, 6)
hx.set_reading_format("MSB", "MSB")
hx.set_reference_unit(referenceUnit)
hx.reset()
hx.tare()
print("Tare done! Add weight now...")
```

Raspberry Pi

缺貨 (亮紅燈)



有貨 (紅燈熄滅)



AWS Personalize

recommendation_system

user_id	item_id	count	timestamp
1	1	5	874965758
1	2	3	876893171
1	3	4	878542960
1	4	3	876893119
1	5	3	889751712
1	7	4	875071561
1	8	1	875072484
1	9	5	878543541
2	1	4	888550871
6	1	4	883599478
6	7	2	883599102
6	8	4	883600657

Recommendations

Recommendation ID

RID-abf34aac-48fb-4f0d-89cc-f9a8afdf64e3

Item ID

235

70

402

234

640

949

Test campaign results

User ID Info

This is the user ID of the user you want to see campaign results for. This user ID needs to be obtained from your user-interactions or user dataset.

1

大

Score

0.0090934

0.0077030

0.0054251

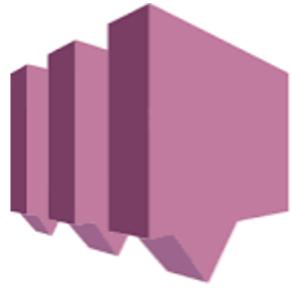
0.0046498

0.0039225

0.0038401

小





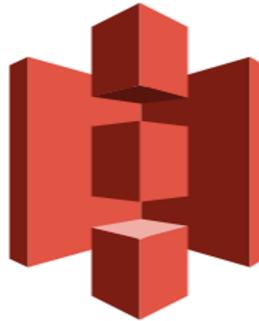
SNS



Amazon
API Gateway



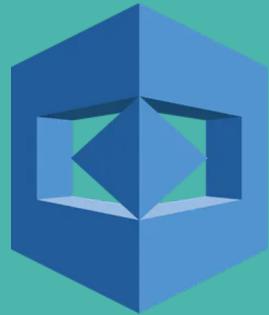
Summary



Amazon S3



Amazon DynamoDB



Rekognition



AWS Lambda



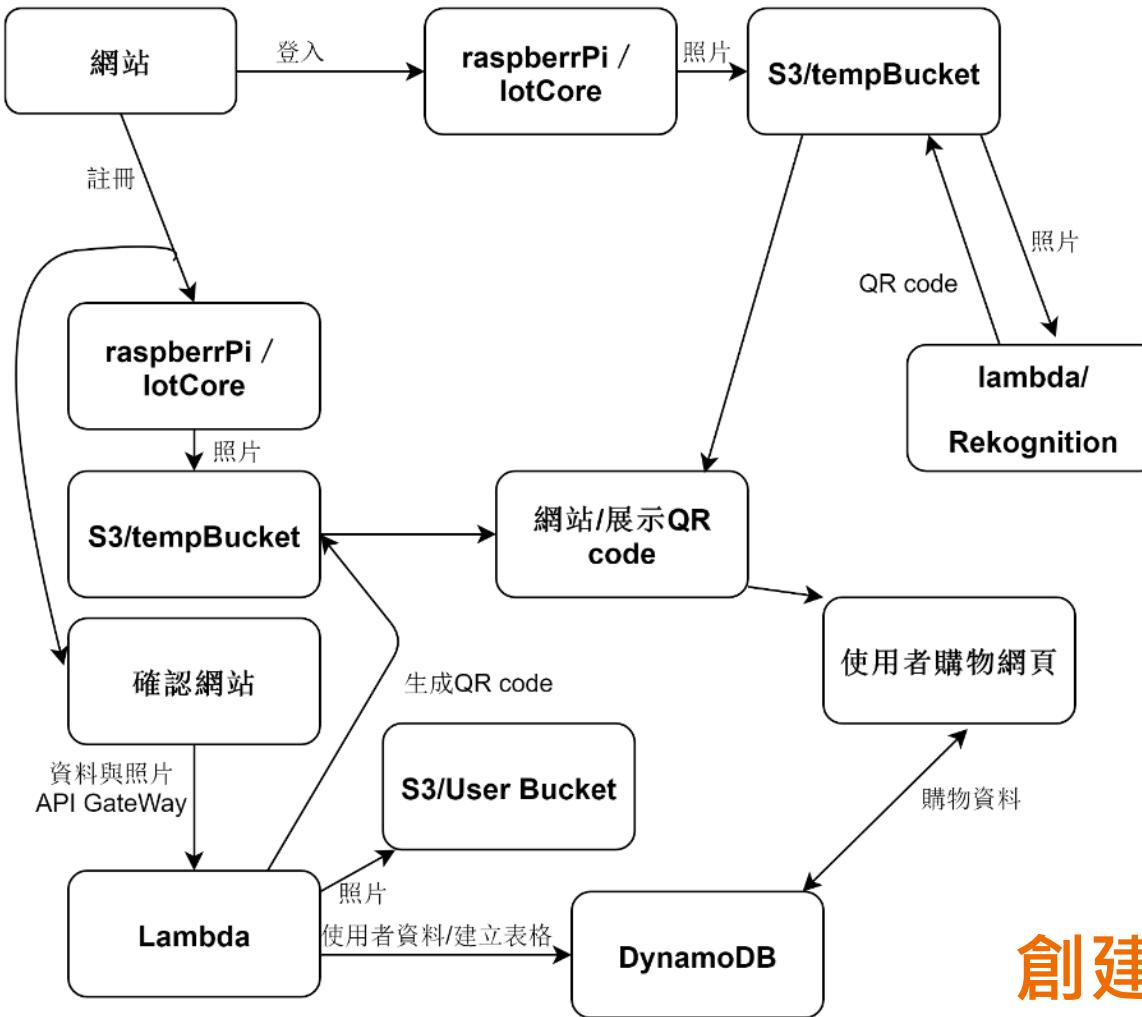
AWS IoT core



AWS Personalize

Thank you

Team 20



註冊/登入
創建個人頁面

