Python & Kafka
餐點訂單分散式系統設計

郭益華

GitHub

目錄

1. 簡介說明

實作說明

- 使用Python & Kafka 模擬設計一個高擴展性的食物訂單系統

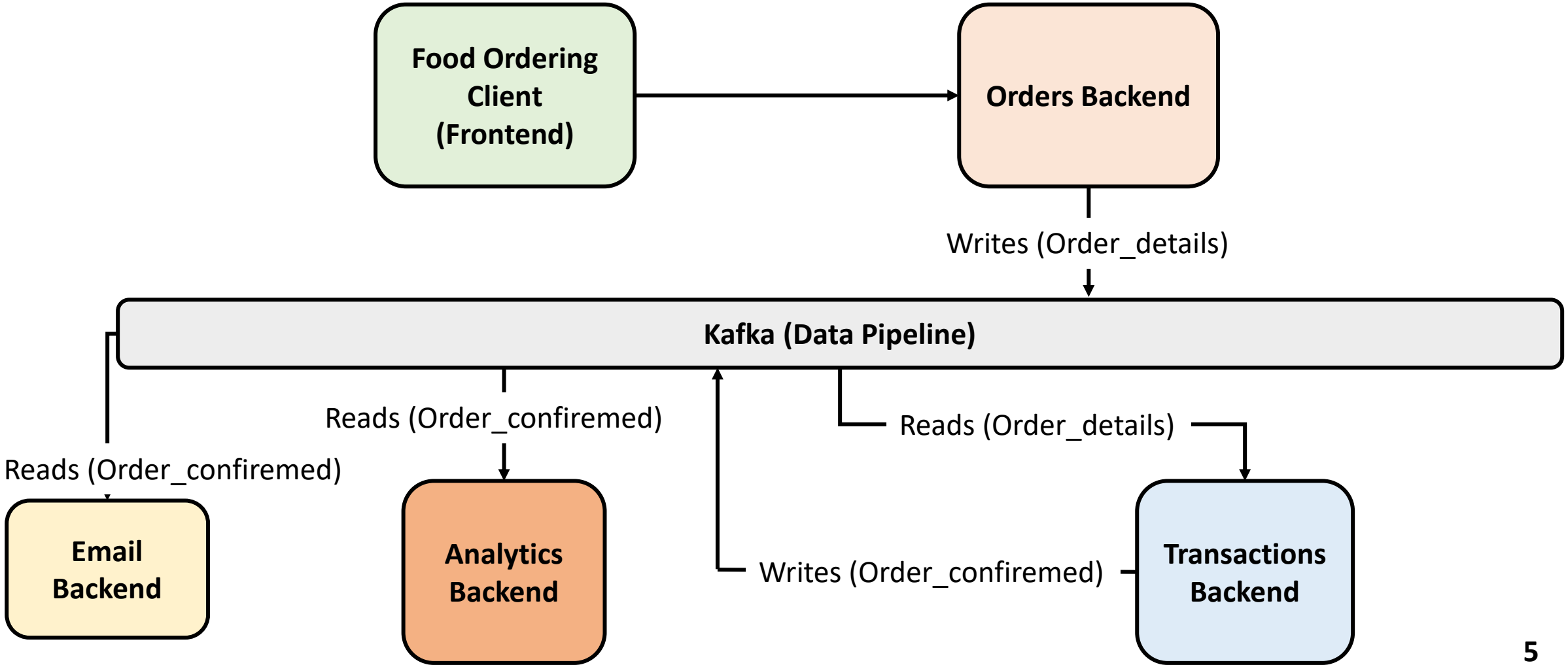
每件重要的事情皆為一個獨立事件，例如下訂單和確認訂單

- 實作專注於系統的設計並不會撰寫前端程式，會以迴圈的方式模擬訂單產生

- 系統優勢:

如果任何前端或後端系統出現問題，資料不會遭受任何遺失，因為每項功能皆為各自獨立

Flow


Food Ordering Client (Frontend)
Orders Backend
Writes (Order_details)
Kafka (Data Pipeline)
Reads (Order_confiremed)
Reads (Order_confiremed)
Reads (Order_details)
Email Backend
Analytics Backend
Writes (Order_confiremed)
Transactions Backend

專案架構

- Hierarchy

```
pythonkafka
├── analytics.py (計算訂單數及收入)
├── docker-compose.yml
├── email.py (信箱資訊)
├── order_backend.py (訂單資訊)
├── requirements.txt (使用套件)
├── transaction.py (交易資訊)
```

前置準備

- Docker下載安裝
- pip install –r requirement.txt

Kafka是什麼

- Apache Kafka 是一個分散式的 streaming 平台，提供以下幾種功能：

Message System：如常見的 AWS SQS, ActiveMQ…等 message queue 服務提供發布與訂閱資料串接資料流，應用場景如：收集各種指標（Metrics）、日誌（Log）。

Streaming processing：Kafka 提供 streaming API 直接在 Kafka 上撰寫 streaming 資料處理，如 word count, max, min 等統計分析。

Kafka 名詞概念解釋

- **Event**：事件，指的是當有什麼事情發生時所保留下的紀錄或訊息，當在對 Kafka 做讀寫時，就是以事件的形式來做這些事情。一個事件包含鍵(key)、值(value)、時間戳章(timestamp)，或是可選的元資料頭(metadata headers)

- **Producers** ：生產者，指的是向 Kafka 發佈事件的應用程式。

- **Consumers** ：消費者，指的是向 Kafka 訂閱事件的應用程式。

- **Topic**：主題，事件被組織化、持久的儲存在主題中。一個主題可以由零個到多個生產者和消費者去發佈或訂閱它。主題中的事件是隨時可以被讀取的，事件被消費後不被刪除。使用者可以透過配置設定每個主題中的事件可以保留多長的時間，而 Kafka 的性能相對於數據的大小是恆定的，因此長時間的儲存數據是完全沒問題的。

2. 使用docker compose 部署kafka

撰寫docker-compose.yml

docker-compose.yml

```yaml
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - 22181:2181

  kafka:
    image: confluentinc/cp-kafka:5.3.1
    depends_on:
      - zookeeper
    ports:
      - 29092:29092
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://localhost:29092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
```

於專案目錄下
啟動指令:
docker-compose up

指令可以加上-d參數使容器
於背景執行

3. 撰寫 order_backend.py

order_backend.py × | transaction.py | email.py | analytics.py

order_backend.py > ...

```python
import json
import time

from kafka import KafkaProducer

ORDER_KAFKA_TOPIC = "order_details"
ORDER_LIMIT = 5

producer = KafkaProducer(bootstrap_servers="localhost:29092")

print("Going to be generating order after 10 seconds")
print("Will generate one unique order every 10 seconds")

# 模擬訂單數據
for i in range(1, ORDER_LIMIT):
    data = {
        "order_id":1,
        "user_id":f"tom_{i}",
        "total_cost": i * 2,
        "items": "burger, sandwich"
    }
# 將數據發送到所定義的主題
    producer.send(
        ORDER_KAFKA_TOPIC,
        json.dumps(data).encode("utf-8")
    )
    print(f"Done sending...{i}")
```

使用迴圈模擬訂單

測試

```
Windows PowerShell    Windows PowerShell
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python order_backend.py
Going to be generating order after 10 seconds
Will generate one unique order every 10 seconds
Done sending...1
Done sending...2
Done sending...3
Done sending...4
```

4. 撰寫 transaction.py

order_backend.py | transaction.py | email.py | analytics.py

transaction.py > ...

```python
import json

from kafka import KafkaConsumer
from kafka import KafkaProducer

ORDER_KAKFA_TOPIC = "order_details"
ORDER_CONFIRMED_KAFKA_TOPIC = "order_confirmed"

consumer = KafkaConsumer(
    ORDER_KAKFA_TOPIC,
    bootstrap_servers="localhost:29092"
)
producer = KafkaProducer(
    bootstrap_servers="localhost:29092"
)

print("Gonna start listening...")
# 接收order_backend.py的資料並轉換為json印出
while True:
    for message in consumer:
        print("Ongoing transaction...")
        consumed_message = json.loads(message.value.decode())
        print(consumed_message)

        user_id = consumed_message["user_id"]
        total_cost = consumed_message["total_cost"]
```

將從order_backend.py收到的資料user_id, total_cost 轉換為 custome_id, customer_email, total_cost，並發送

```python
# 將從order_backend.py收到的資料user_id, total_cost 轉換為 custome_id,
# 並發送

        data = {
            "coustomer_id": user_id,
            "customer_email": f"{user_id}@gmail.com",
            "total_cost": total_cost
        }

        print("Successful transaction...")
        producer.send(
            ORDER_CONFIRMED_KAFKA_TOPIC,
            json.dumps(data).encode("utf-8")
        )
```

同步測試

order_backend.py

```
Windows PowerShell    Windows PowerShell
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python order_backend.py
Going to be generating order after 10 seconds
Will generate one unique order every 10 seconds
Done sending...1
Done sending...2
Done sending...3
Done sending...4
```

transaction.py

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python transaction.py
Gonna start listening...
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_1', 'total_cost': 2, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_2', 'total_cost': 4, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_3', 'total_cost': 6, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_4', 'total_cost': 8, 'items': 'burger, sandwich'}
```

5. 撰寫 email.py

收來自 transaction.py 中的信箱資訊

email.py > ...

```python
import json

from kafka import KafkaConsumer


ORDER_CONFIRMED_KAFKA_TOPIC = "order_confirmed"


consumer = KafkaConsumer(
    ORDER_CONFIRMED_KAFKA_TOPIC,
    bootstrap_servers="localhost:29092"
)
# 接收來自transaction.py中的信箱資訊
emails_sent_so_far = set()
print("Gonna start listening")
while True:
    for message in consumer:
        consumed_message = json.loads(message.value.decode())
        customer_email = consumed_message["customer_email"]
        print(f"Sending email to {customer_email} ")
        emails_sent_so_far.add(customer_email)
        print(f"So far emails sent to {len(emails_sent_so_far)} unique emails")
```

同步測試

```
Windows PowerShell    Windows PowerShell
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python order_backend.py
Going to be generating order after 10 seconds
Will generate one unique order every 10 seconds
Done sending...1
Done sending...2
Done sending...3
Done sending...4
```

order_backend.py

transaction.py

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python transaction.py
Gonna start listening...
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_1', 'total_cost': 2, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_2', 'total_cost': 4, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_3', 'total_cost': 6, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_4', 'total_cost': 8, 'items': 'burger, sandwich'}
```

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python email.py
Gonna start listening
Sending email to tom_1@gmail.com
So far emails sent to 1 unique emails
Sending email to tom_2@gmail.com
So far emails sent to 2 unique emails
Sending email to tom_3@gmail.com
So far emails sent to 3 unique emails
Sending email to tom_4@gmail.com
So far emails sent to 4 unique emails
```

email.py

6. 撰寫 analytics.py

接收 order_backemd.py的訂單資訊
計算訂單總數與收入

order_backend.py | transaction.py | email.py | analytics.py

analytics.py > ...

```python
import json

from kafka import KafkaConsumer


ORDER_CONFIRMED_KAFKA_TOPIC = "order_confirmed"


consumer = KafkaConsumer(
    ORDER_CONFIRMED_KAFKA_TOPIC,
    bootstrap_servers="localhost:29092"
)

# 接收 order_backemd.py的訂單資訊
# 計算訂單總數與收入
total_orders_count = 0
total_revenue = 0
print("Gonna start listening")
while True:
    for message in consumer:
        print("Updating analytics..")
        consumed_message = json.loads(message.value.decode())
        total_cost = float(consumed_message["total_cost"])
        total_orders_count += 1
        total_revenue += total_cost
        print(f"Orders so far today: {total_orders_count}")
        print(f"Revenue so far today: {total_revenue}")
```

同步測試(1/2)

```
Windows PowerShell    Windows PowerShell
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python order_backend.py
Going to be generating order after 10 seconds
Will generate one unique order every 10 seconds
Done sending...1
Done sending...2
Done sending...3
Done sending...4
```

order_backend.py

transaction.py

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python transaction.py
Gonna start listening...
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_1', 'total_cost': 2, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_2', 'total_cost': 4, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_3', 'total_cost': 6, 'items': 'burger, sandwich'}
Ongoing transaction...
{'order_id': 1, 'user_id': 'tom_4', 'total_cost': 8, 'items': 'burger, sandwich'}
```

同步測試(2/2)

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python email.py
Gonna start listening
Sending email to tom_1@gmail.com
So far emails sent to 1 unique emails
Sending email to tom_2@gmail.com
So far emails sent to 2 unique emails
Sending email to tom_3@gmail.com
So far emails sent to 3 unique emails
Sending email to tom_4@gmail.com
So far emails sent to 4 unique emails
```

email.py

```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\kafkaPython\pythonkafka> python analytics.py
Gonna start listening
Updating analytics..
Orders so far today: 1
Revenue so far today: 2.0
Updating analytics..
Orders so far today: 2
Revenue so far today: 6.0
Updating analytics..
Orders so far today: 3
Revenue so far today: 12.0
Updating analytics..
Orders so far today: 4
Revenue so far today: 20.0
```

analytics.py

End