Kafka - message broker (สามารถใช้ service bus แทนได้ Rabbit MQ, Active MQ)
Java, scala ผสม

1. มี even ex. การคลิก, การกรอกข้อมูล
2. ประมวลผลข้อมูลที่มาก

- message, even, data = ข้อมูลที่ส่งต่อ
- Kafka เป็น Broker = รับส่งหรือกระจายข้อมูล (ตัวกลาง) (สามารถใช้ queue, sqs ของ AWS ประมวลผลได้)
- Architect สามารถทำ hight to put, low latency = อัด data เข้าไปได้เยอะในเวลาที่ความเร็วสูง
  - Kafka สามารถ scale ให้รองรับการเก็บได้ขยาย storage ได้
  - Kafka สามารถทำการ redundancy เก็บหลายๆ copy ได้ ถ้าจุดใดจุดหนึ่งสามารถดึง data จากอีกจุดแทนได้ ทำให้ data ไม่หาย
- เหมาะสำหรับ ปริมาณข้อมูลที่ใหญ่ที่มีการคุยกันหลาย module ได้

ปกติโครงสร้างของการทำงาน

input ⇨ process ⇨ output

1 data = 1 even → 1 even

ปกติของการเรียกใช้ function

user ⇄ request / response ⇄ server

Kafka ของการเรียกใช้ function

even stream → hidemap result surely

* even เกิดมาก่อส่งต่อกันเป็นลำดับ = even streaming
  - ทุกสิ่งที่ user ทำ
  - ex. website realtime activity tracking

source even (ex. log ต่างๆ ใน service) ⇨ chart, graph, matrix

โยน → Kafka
- consumer
- admin
- operation
→ ดู log ใน microservice เพื่อดูความสัมพันธ์เกี่ยวข้องระหว่าง

Kafka ทำงานเป็น Cluster หลาย Kafka brokers รวมกัน
- ตัวที่ทำงานเรียกว่า Kafka broker — เก็บข้อมูลแบบกระจายโดยการ copy

cluster 1 A, B, C
cluster 2 B, D, E
} เป็นการป้องกันการพังของ Broker ซึ่งก็คือ การที่ข้อมูลแบบ topic

topic เก็บข้อมูลแบบที่มีตัวซ้ำกัน โดยใน topic จะประกอบด้วย message ที่มี Data ก้อนข้อมูล ที่จะโยนเข้า topic

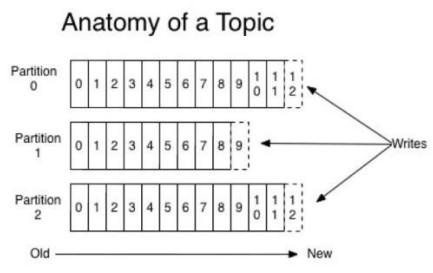
มีการทำงาน คือ producer → topic: message → consumer

โครงสร้างคร่าวๆ

Kafka
- topic 1
  - partition1 message1
  - partition2 message2
  } เกิดการ parallel ขณะ process
  - publicer: produser → message → subscriber consumer
  - produser → mesage → consumer
- topic 2
  - message โดยมี (key), value
    - ทำให้เข้าใจง่ายและทำงานเร็วขึ้น

- ทำงานแบบ parallel รองรับ load ได้มากขึ้น
- consumer access ข้อมูลได้เร็ว

RabbitMQ: queue [q1 | q2 | q3 | q4 | q5 | q6] ออกทีละ 1



Kafka: partition queue 1, partition queue 2, partition queue 3 (ข้อมูลออกทีละ 3)
producer ⇨
- topic เดียวกัน หรือต่าง topic ก็ได้
- ถ้า service ออกจากกัน เป็นการทำงานแบบขนาน

* topic เดียวกันทำให้ แต่ละ partition queue ทำงานพร้อมกันตามหน้าที่ของตนเองได้ ⇨ consumer รับ even ไปทำงานต่อ
* ตรงนี้ partition = การทำงานพร้อมกัน 3 คน
* เป็น process แบบ pallalel

ข้อจำกัดของ Kafka
- Kafka ไม่เหมาะกับ software แบบ sequencecial
- Kafka เหมาะกับ even deviant แบบ คาดไม่ถึงว่าจะมี
  ex. 1 even ที่สามารถเกิดพร้อมกันได้
- ms design ทำในลักษณะที่ each even แบ่งเป็น s with แล้วทำงานพร้อมกัน
- ข้อเสียของ rabbit MQ ใช้ที่ในระบบใหญ่ๆ
  ต้องการ load bartency ต่ำ, too put สูง, handle realtime data หนัก

ข้อดี
1. ทำงานเร็ว, Data ไม่หาย
2. ตัดแบ่ง flow ได้ชัดเจนและแบ่งแต่ละ flow (topic) ทำหน้าที่ของตัวเองได้ ซึ่งเป็น massage broker

Kafka เป็น lib ในลงมา set up ได้
เป็น Open sorurce ใช้กับ aws ใช้ manage service เป็น MSK
Amazon managed streaming kafka → Apache Kafka