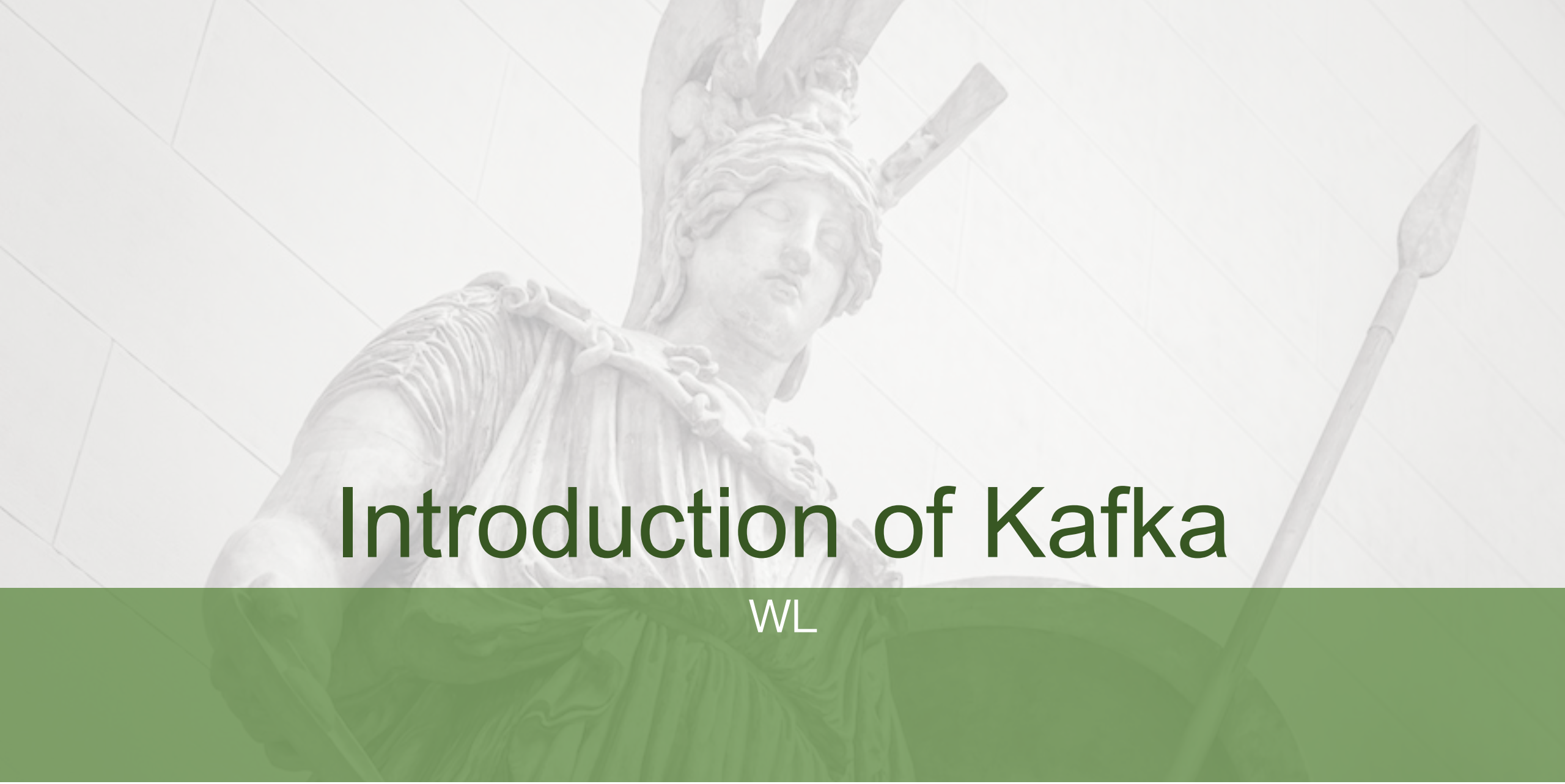


Introduction of Kafka

WL

Outline

- Introduction
- Why Streaming
- Architecture

Introduction

Kafka Outline

- Introduction
- Why streaming
- Streaming architecture

海邊的卡夫卡?!

法蘭茲·卡夫卡?!

What is Kafka

- 最初在 LinkedIn 領導 Kafka 開發的 Jay Kreps 解釋，Kafka 是一個 "system optimized for writing"，那麼使用作家的名字來命名它再好不過。他在大學時期上過很多文學課程，很喜歡作家 Franz Kafka，於是就以 "Kafka" 來命名這個項目
- Kafka 最初是由領英開發，並隨後於 2011 年初開源，並於 2012/10/23 由 Apache Incubator 孵化出站。2014/11 幾個曾在領英為 Kafka 工作的工程師，建立了名為 Confluent 的新公司，並著眼於 Kafka 應用開發

What is Kafka

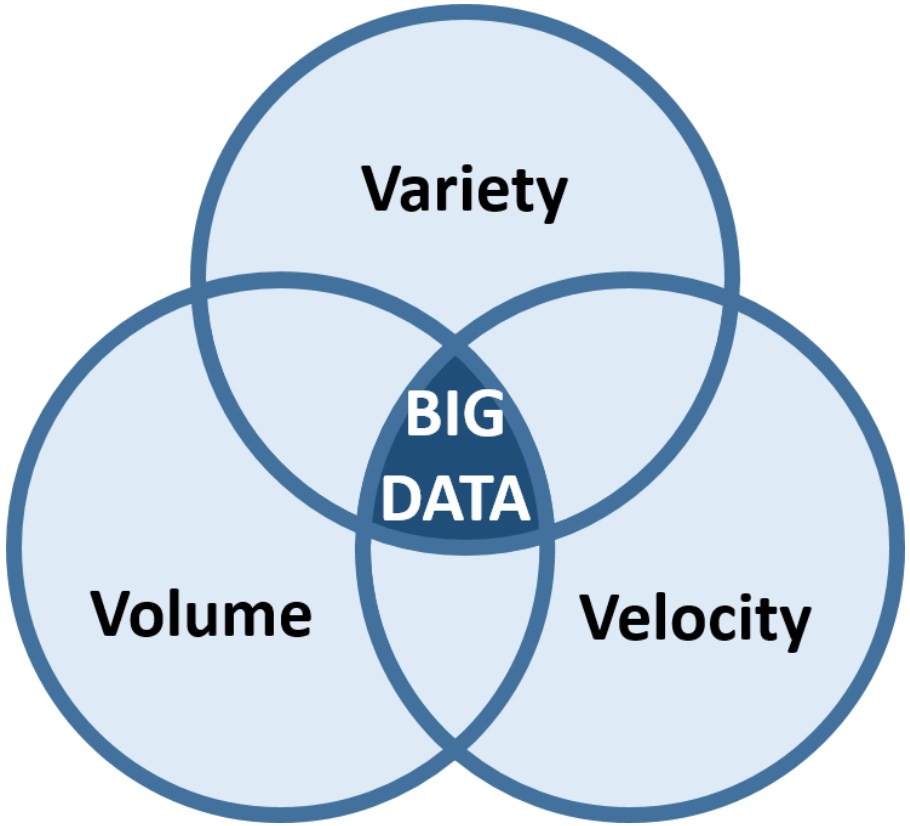
- Kafka 是由 Scala 和 Java 編寫。該專案的目標是為處理實時資料提供一個統一、高吞吐、低延遲的平台。其持久化層本質上是一個「按照分散式事務紀錄檔架構的大規模發布/訂閱訊息佇列」。

Why Streaming

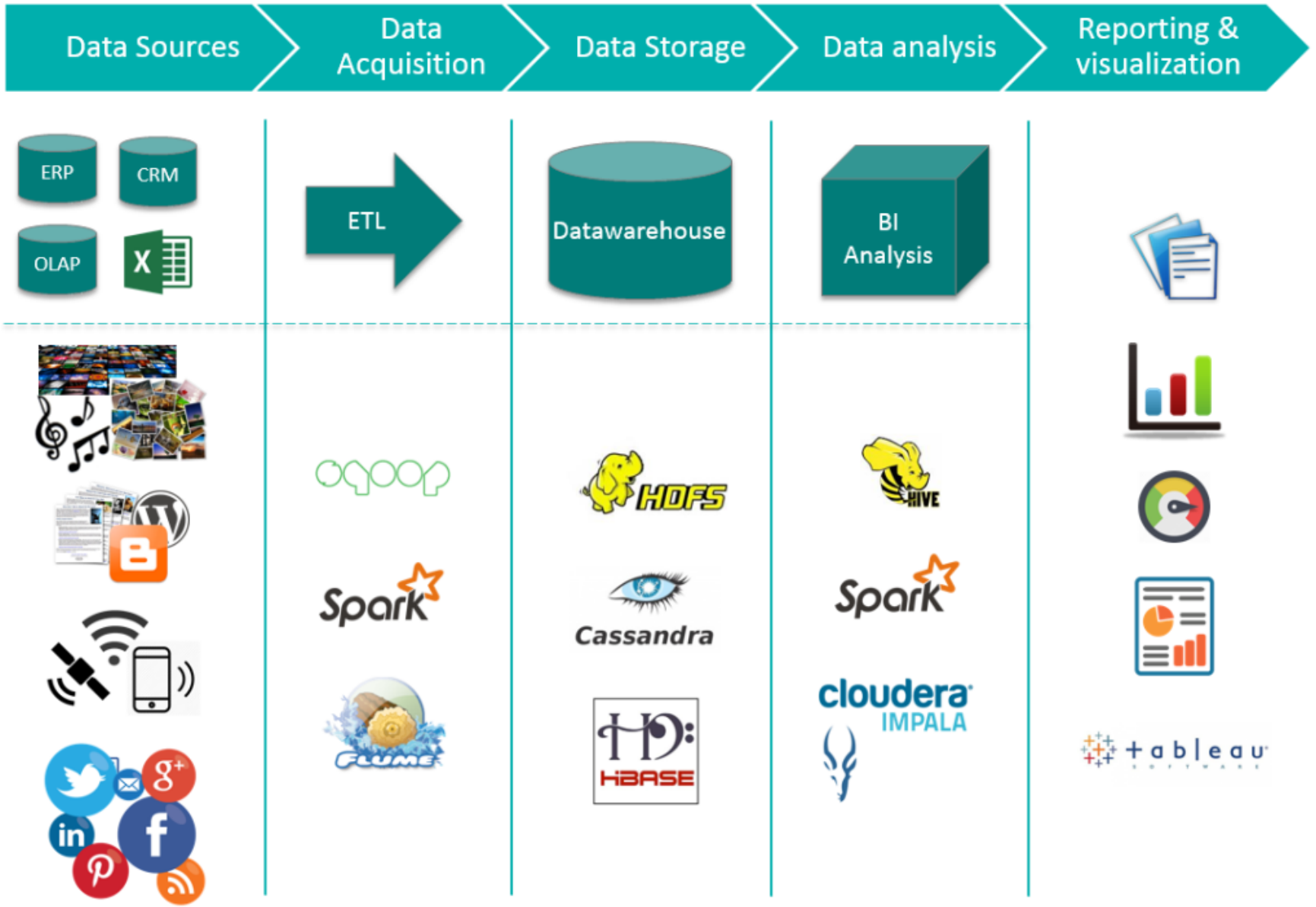
What is Big Data

- Big Data 又稱為大數據，指的是傳統資料處理應用軟體不足以處理它們的大或複雜的資料集的術語。大資料也可以定義為來自各種來源的大量非結構化或結構化資料

Big Data 3V



- Volume – 資料量
- Variety – 資料多元性
- Velocity – 資料即時性


Data Sources
Data Acquisition
Data Storage
Data analysis
Reporting & visualization
ERP
CRM
OLAP
ETL
Datawarehouse
BI Analysis
HDFS
HIVE
Spark
Cassandra
Spark
cloudera
IMPALA
HBASE
FLUME
tableau

ATHEMASTER

Batch vs. Streaming

- 若數據比做為水，批次處理和流處理的過程就如同用水桶裝水然後交付給客戶，而流處理則是直接將水流導向客戶來使用
- 然而很多時候，批次處理並沒有辦法完整的反應真實情況，也因此流的計算才會越來越受到關注

Batch vs. Streaming

	批次處理	串流處理
資料範圍	• 查詢或處理資料集中的所有或大部分資料	• 查詢或處理滾動式時窗內的資料或僅限最新的資料記錄
資料大小	• 大型資料批次	• 個別記錄或包含幾筆記錄的微型批次
效能	• 延遲從幾分鐘到數小時	• 需要幾秒或幾毫秒的延遲
分析	• 複雜分析	• 簡易回應功能、彙總和累積指標

What is Streaming

- 串流資料是由數千個資料來源持續產生的資料，通常會同時傳入資料記錄，且大小不大 (約幾 KB)。
- 串流資料包含各式各樣的資料
  - 客戶使用您的行動或 Web 應用程式產生的日誌檔
  - 電子商務採購、金融交易、遊戲中的玩家活動行為紀錄
  - 來自社交網路的紀錄
  - 連線裝置或資料中心儀器的遙測結果

Streaming Application Example

- 媒體出版商將來自其線上事業的數十億筆點擊流記錄製作成串流，透過使用者的人口統計資訊彙整資料來提供使用者最佳的相關性和體驗。
- 金融機構即時追蹤股市的變動、計算風險值，以及根據股價波動來自動重新平衡投資組合。

Streaming Application Example

- 在金融系統中，在客戶進行信用卡交易時，可以快速知道此筆交易是否為盜刷或偽造
- 線上遊戲公司收集關於玩家和遊戲的互動串流資料，並將這些資料傳送到遊戲平台。接著即時分析這些資料，提供獎勵和動態體驗來吸引玩家。

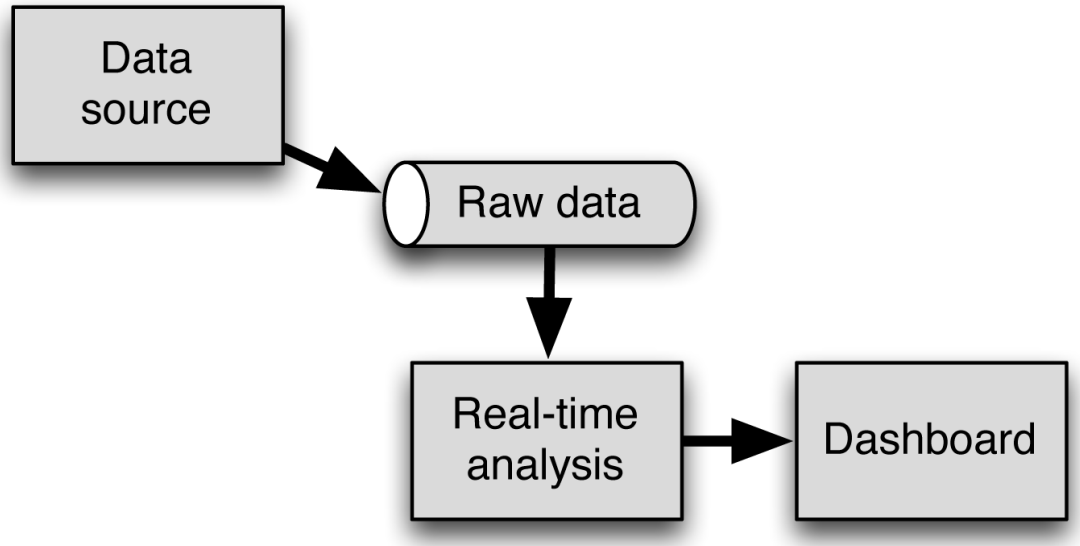
Streaming Application Example

- 運輸車輛、工業設備以及農業機械中的感應器會將資料傳送到串流應用程式。該應用程式會監控效能、預先偵測任何可能的故障，以及自動訂購零件以避免設備停機時間。

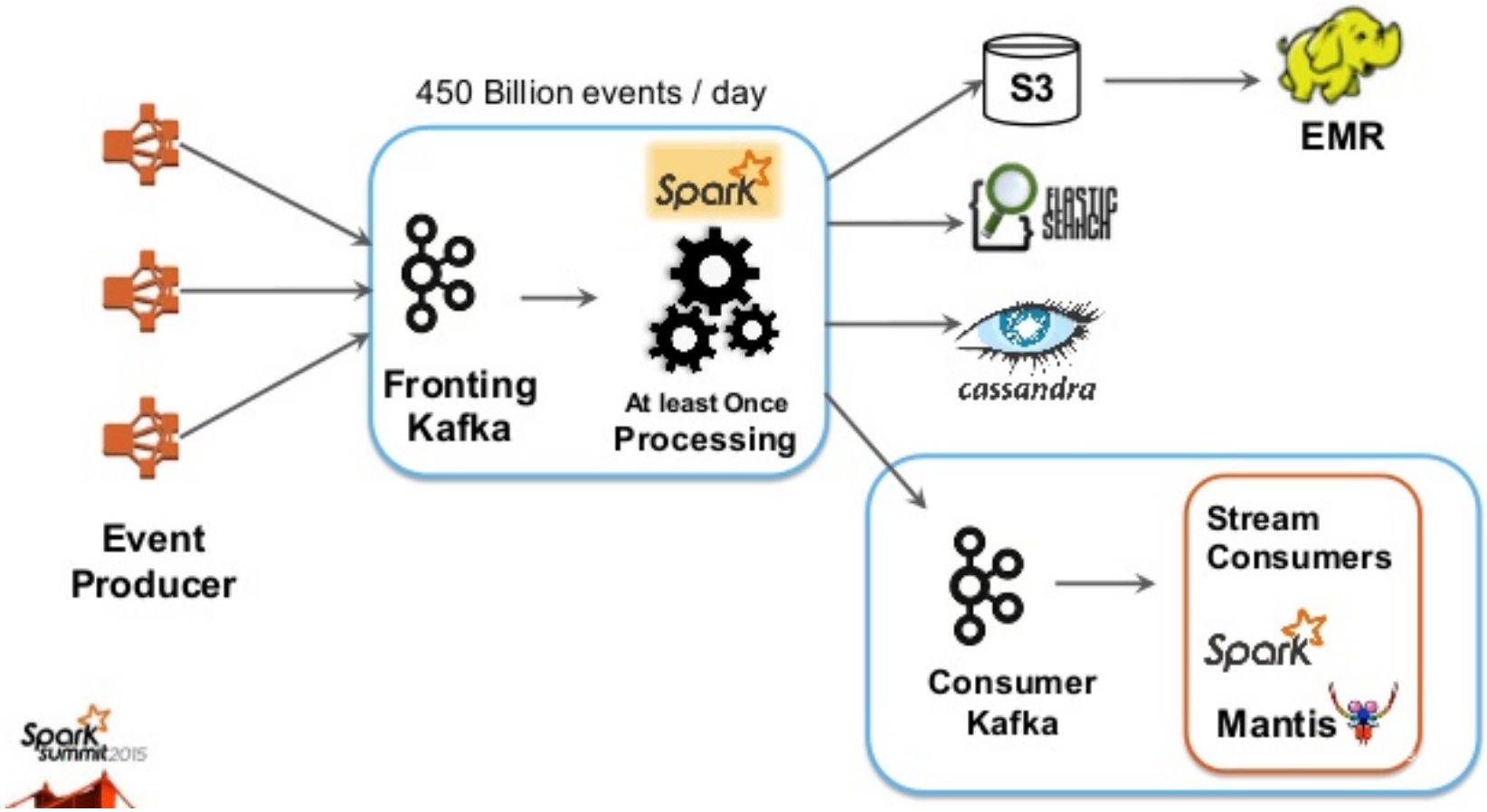
Architecture

What Can Kafka Do

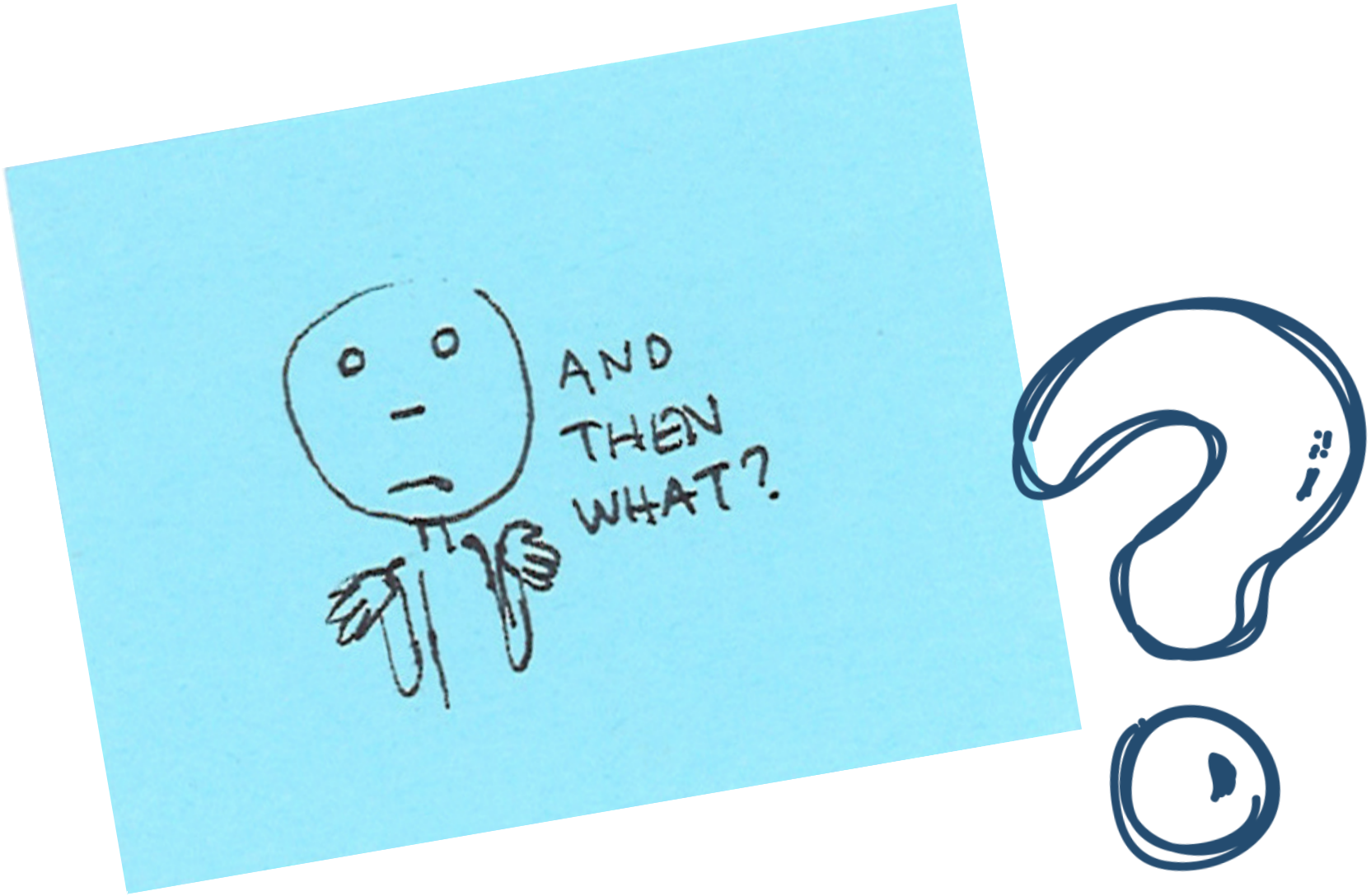
- 作為訊息佇列使用 (Message Queue)，若分析程序臨時中斷，或運行分析的處理速度變慢，可以用來作為分析程序之前安排訊息陣列，當作系統的安全緩存



Message Queue

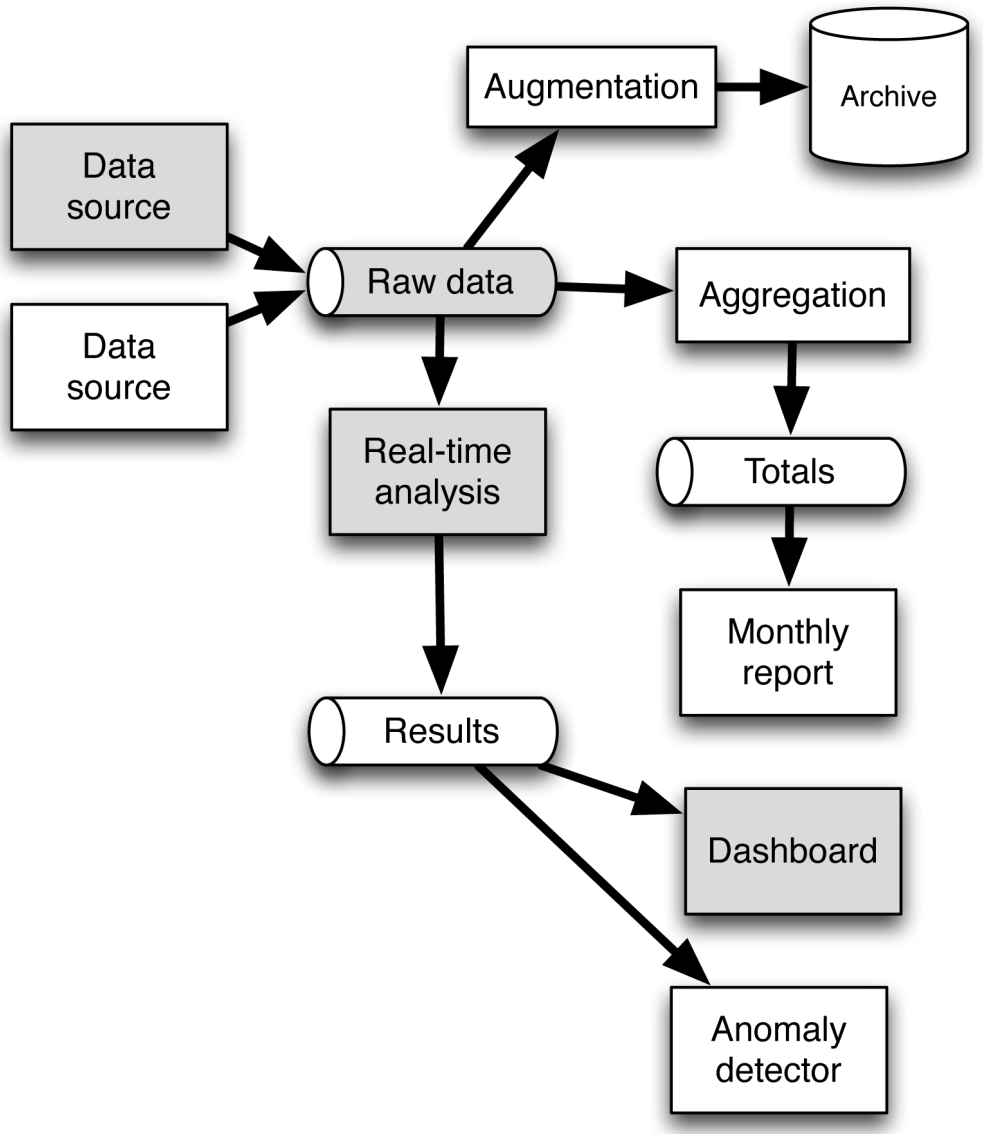


ATHEMASTER



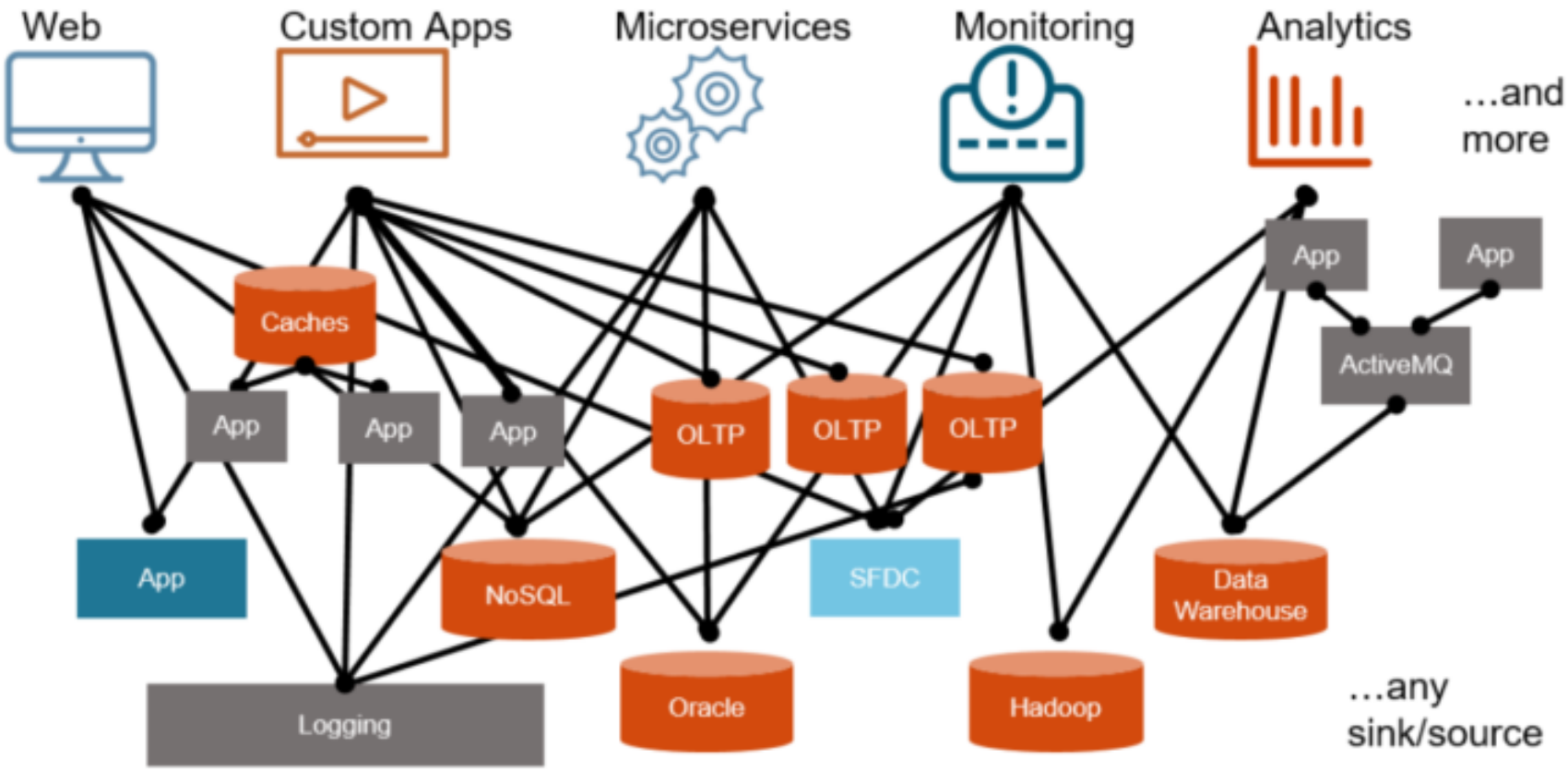
Data Integration

- *Data Integration—Making all of an organization's data easily available in all its storage and processing systems, by Jay Kreps*

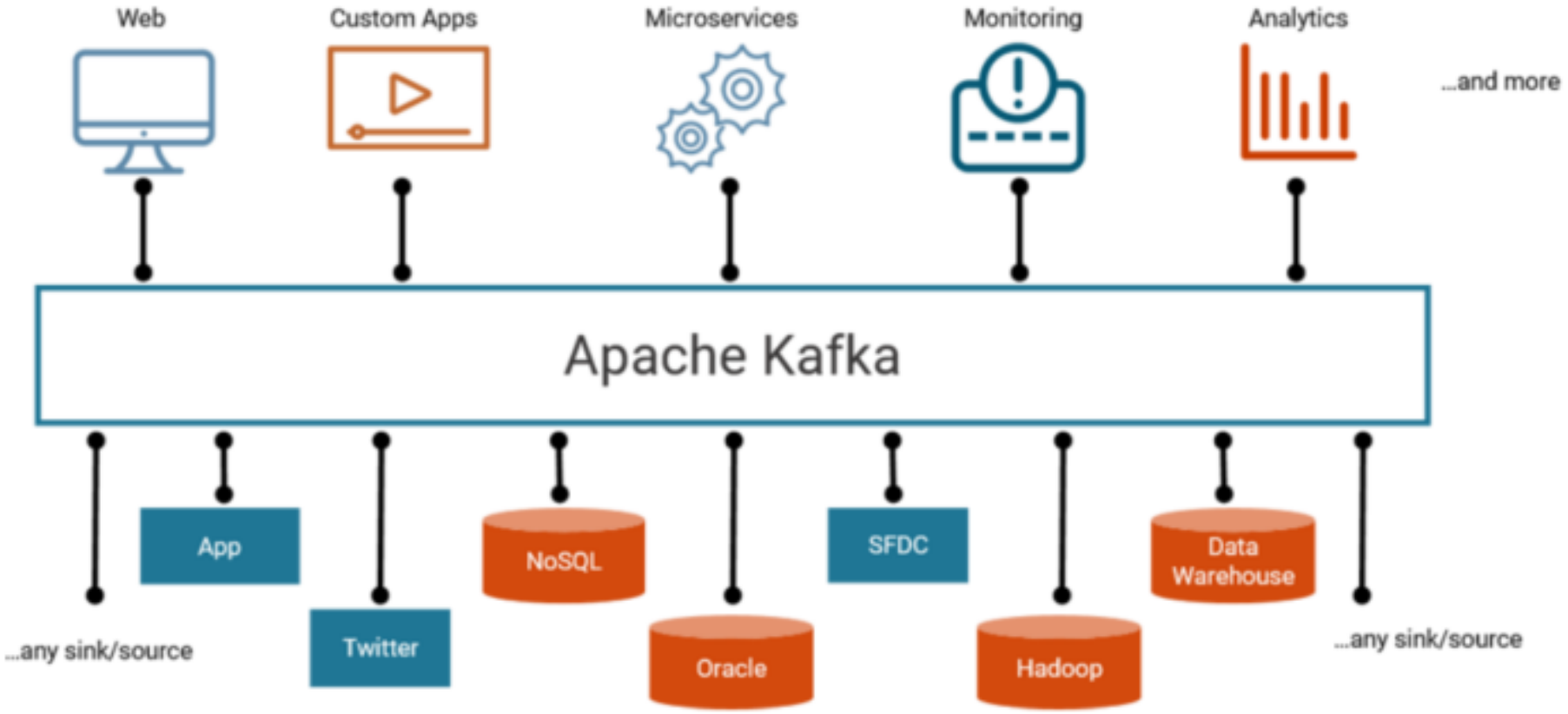


- 傳統的即時分析典型設計，來自單一資料源的數據，用於更新 dashboard
- 在 streaming 架構中，多個組件可以使用同一個訊息流，除了統計外，也可以運用在不同的用途上。

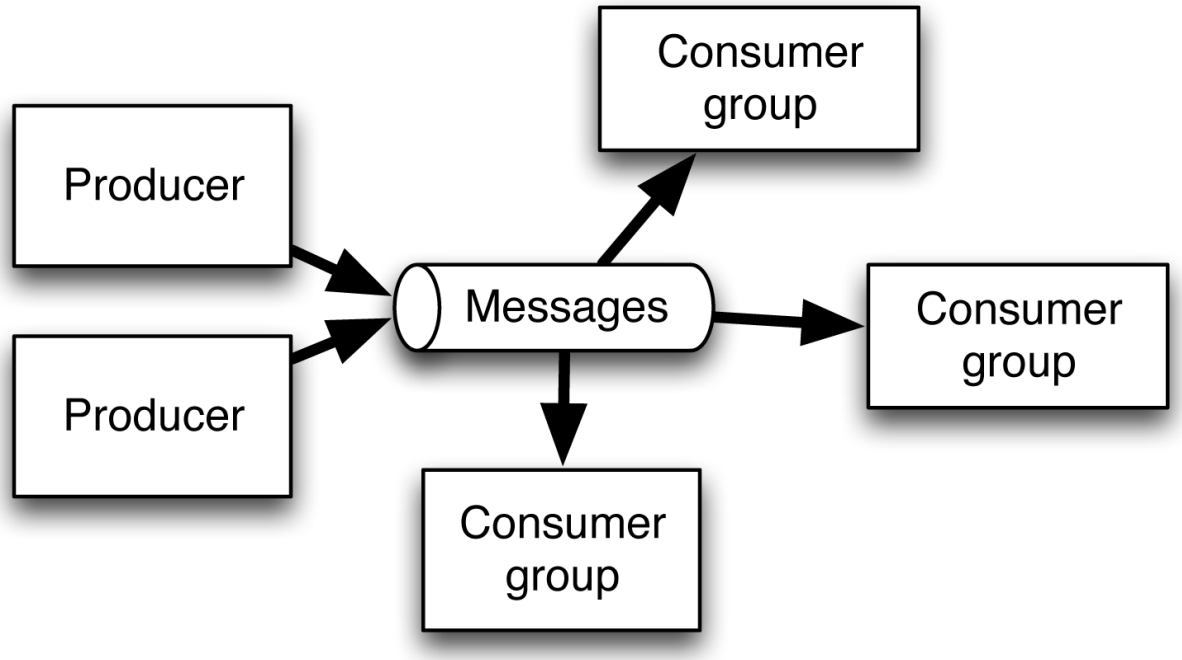
Decoupling



Decoupling



Message-passing Infrastructure



- Message-passing infrastructure 的核心則是以這種方法運作，處理來自多個數據源 (producer)，並以解耦合的方式提供 consumer 使用

Message-passing Infrastructure

- Message-passing infrastructure 需運用於大規模數據、要求彈性、快速數據處理能力，通常需要滿足下列條件
  - 解耦合
  - 持久性 & 冗餘
  - 擴展性
  - 靈活性和峰值處理能力
  - 可恢復性
  - 順序保證
  - 緩衝
  - 異步通信

Message-passing Infrastructure

- 解耦合
  - 消息系統在處理過程中間插入了一個隱含的、基於數據的接口層，兩邊的處理過程都要實現這一接口。這允許你獨立的擴展或修改兩邊的處理過程
- 持久性 & 冗餘 (redundant)
  - 有些情況下，處理數據的過程會失敗。除非數據被持久化，否則將造成丟失。消息隊列把數據進行持久化直到它們已經被完全處理，通過這一方式規避了數據丟失風險

Message-passing Infrastructure

- 擴展性
  - 因為消息隊列解耦了你的處理過程，所以增大消息入隊和處理的頻率是很容易的，只要另外增加處理過程即可。不需要改變代碼
- 靈活性 & 峰值處理能力
  - 在訪問量劇增的情況下，應用仍然需要繼續發揮作用，但是這樣的突發流量並不常見；如果為以能處理這類峰值訪問為標準來投入資源隨時待命無疑是巨大的浪費。使用消息隊列能夠使關鍵組件頂住突發的訪問壓力，而不會因為突發的超負荷的請求而完全崩潰

Message-passing Infrastructure

- 可恢復性
  - 系統的一部分組件失效時，不會影響到整個系統。消息隊列降低了進程間的耦合度，所以即使一個處理消息的進程掛掉，加入隊列中的消息仍然可以在系統恢復後被處理。
- 順序保證
  - 在大多使用場景下，數據處理的順序都很重要。大部分消息隊列本來就是排序的，並且能保證數據會按照特定的順序來處理。 Kafka 保證一個 Partition 內的消息的有序性。

Message-passing Infrastructure

- 緩衝
  - 消息隊列通過一個緩衝層來幫助任務最高效率的執行，寫入隊列的處理會盡可能的快速。該緩衝有助於控制和優化數據流經過系統的速度。
- 異步通信
  - 用戶不想也不需要立即處理消息。消息隊列提供了異步處理機制，允許用戶把一個消息放入隊列，但並不立即處理它

Q & A

– THE PATHFINDER TO OPENSOURCE –

info@athemaster.com
FB: Athemaster