Building Event Driven Services with Apache Kafka, Kafka Streams & KSQL

Ben Stopford
@benstopford
There is a book!

Event Driven Architectures
Business Events
Event Sourcing

DDD

Stream Processing
Today's ecosystems get pretty big

- 2.2 trillion messages per day (6 Petabytes)
- Up to 400 Microservices per cluster.
- 20-200 Brokers per cluster
Today’s ecosystems get pretty big

- 1 billion messages per day
- 20,000 messages per second
- 100 teams
Event Driven Architectures
Business Events
Event Sourcing
DDD

Stream Processing
Streaming Platforms

KAFKA

Kafka Streams / KSQL

Serving Layer (Cassandra etc.)

High Throughput Messaging

Data is embedded in each engine

Clustered Java App

confluent
Streaming Pipeline

(a) Apps Opened, Per App, Per day

(b) Crashes, Per App, Per day

(c) Unstable Applications

apps_opened

opened_per_day

crashed_per_day

unstable_apps

app_crashes
Streaming Platforms

KAFKA

Kafka Streams / KSQL

High Throughput Messaging

Data is embedded in each engine

Clustered Java App

Serving Layer (Cassandra etc.)
An event log is a simple idea

Messages are added at the end of the log
Readers have a position all of their own

George
is here
Scan

Old
Fred
is here
Scan
New
Sally
is here
Scan
You can rewind and replay, just like Tivo!
The hard part: Tying it all together!
Many "logs" over many machines

Producing Services

Kafka

Consuming Services
Resistant to Failure

Producing Services

Kafka

Consuming Services
Streaming Platforms

KAFKA

Kafka Streams / KSQL

High Throughput Messaging

Data is embedded in each engine

Clustered Java App

Serving Layer (Cassandra etc.)
Streaming Example

apps_opened

open_per_day
CREATE TABLE opened_per_day AS
SELECT app_id, count(*)
FROM apps_opened
WINDOW TUMBLING (SIZE 1 DAY)
GROUP BY app_id;
CREATE TABLE opened_per_day AS
SELECT app_id, count(*)
FROM apps_opened
WINDOW TUMBLING (SIZE 1 DAY)
GROUP BY app_id;
CREATE TABLE opened_per_day AS
SELECT app_id, count(*)
FROM apps_opened
WINDOW TUMBLING (SIZE 1 DAY)
GROUP BY app_id;
CREATE TABLE opened_per_day AS
SELECT app_id_id, count(*)
FROM apps_opened
WINDOW TUMBLING (SIZE 1 DAY)
GROUP BY app_id_id;
CREATE TABLE opened_per_day AS
SELECT app_id, count(*)
FROM apps_opened
WINDOW TUMBLING (SIZE 1 DAY)
GROUP BY app_id;
CREATE TABLE opened_per_day AS
SELECT app_id, count(*)
FROM apps_opened
WINDOW TUMBLING (SIZE 1 DAY)
GROUP BY app_id;
Streaming is manipulating events in flight, at scale.
Event Driven Architectures
Business Events
Event Sourcing
DDD

Stream Processing
Increasingly we build ecosystems
SOA / Microservices / EDA
The Problem is DATA

YOUR DATA
Most services share the same core facts.

Most services live in here.
Events have two hats

Notification

Data replication
Buying an iPad (with REST/RPC)
Events for Notification Only

Notification

Submit Order

Orders Service

Shipping Service

Customer Service

Order Created

Webserver

getCustomer()

REST

KAFKA
Pluggability

Webserver

Submit
Order

Notification

Orders
Service

Shipping
Service

Customer
Service

getCustomer()

REST

KAFKA

Orders Created

Confluent
Repricing

Orders

payments

Confluent
Events for Data Locality

Data is replicated

Submit Order

Orders Service

Shipping Service

Customer Service

KAFKA
Events have two hats

Notification

Data replication
Stateless / Stateful Stream Processing Relates to these hats
Stateless Stream Processing

Notification

Submit Order

Orders Service

Kafka Streams / KSQL

Webserver

KStreams Shipping Service

Customer Service

REST/RPC

getCustomer()

Order Created

Order

KAFKA
Stateful Stream Processing

Data replication

Order Created

Order Updated

Kafka
Streams & Tables

KStreams
Shipping Service

Join

Orders Stream (Buffer)

Customers (Buffer All)

KAFKA
KSQL ~ KStreams
Streaming is about

1. Joining & Operating on Streams

2. Joining & Operating on Materialized Tables

On Notification

Data Replication
Kafka: a Streaming Platform

Producer → The Log → Consumer

Connectors

Streaming Engine

Connectors
8 Steps to Streaming Services
1. Use events to decouple and collaborate
2. Use Connect (Δ CDC) to evolve away from legacy
Make Legacy Datasets Available via the Log
3. Use the Single Writer Principal
State changes to a topic owned by one service

- Browser
- Webserver
- Orders Service
- Order Received
- Order Validated
- Order Completed
- Products
- KAFKA
- Connect
Local consistency points in the absence of Global Consistency
4. Use Kafka as a Shared Source of Truth (Messaging that Remembers)
Shared Source of Truth

Browser

Webserver

Orders Service

Order Received
Order Validated
Order Completed

Products

KAFKA

Connect

Reporting
Product Catalogue stored in 3 places

- Browser
- Webserver
- Orders Service

Orders Services:
- Order Received
- Order Validated
- Order Completed

Products

Connect

KAFKA

Reporting view may be “thinner”
5. Move Data to Code
Materialize Stock ‘View’ Inside Service

Browser

Webserver

Orders Service

Stock

KAFKA

Order Received

Order Validated

Order Completed

Products

Stock

Data Replication

Connect

Browser

Webserver

Orders Service

KAFKA

Order Received

Order Validated

Order Completed

Products

Stock

Data Replication
Kafka has several features for reducing the need to move data on startup

- Standby Replicas
- Disk Checkpoints
- Compacted topics
6. Write to State Stores, just like a local ‘database’, backed up in Kafka
State stores behave like local databases

Diagram showing connections between Browser, Webserver, Orders Service, Reserved Stocks, Stock, Products, and KAFKA.
7. Use Transactions to tie All Interactions Together
8. Evolve and Grow
Tiered Contexts
Span regions or clouds
Handle Disconnectedness
So...
Optimize for complexity vs optimize for scale

Event Driven Architectures

Stream Processing
Events provide the key to evolutionary architectures

Notification

Data replication
Spectrum of use cases

Finer Grained, Collaborative, Connected

Courser Grained, Non-collaborative, Disconnected

Notification

Data Replication
Streaming is the toolset for dealing with events at scale
Event Driven Services

- Broadcast events
- Retain them in the log
- Evolve the event-stream with streaming functions
- Recasting the event stream into views when you need to query.
Find out more


Software: https://confluent.io/download/


Twitter: @benstopford