Beyond Microservices: Streams, State and Scalability

Gwen Shapira, Engineering Manager
@gwenshap
In the beginning...
We Have Microservices
We ❤ Microservices
They need to communicate
I know! I’ll use REST APIs

Orders

Validate Order

Inventory

Fulfill Order

Returns?
Synchronous request-response communication Leads to Tight point-to-point coupling
Clients know too much

- Returns
- Orders
- Inventory
- Validate
- Fulfill
shifts in responsibility, redundancy

Retry?

Retry?

Retry?

Fail
Making Changes is Risky
Adding Services Requires Explicit Calls
REST = HTTP + JSON = SLOW
We can do better.
Nice to meet you!

- Moving data around for 20 years
- Engineering Manager at Confluent
- Apache Kafka Committer
- Wrote a book or two
- Tweets a lot - @gwenshap
API Gateway
Clients know too much

Returns → Inventory
Inventory → Validate
Validate → Fulfill
Fulfill → Orders
Orders → Returns
Shift in responsibility, redundancy
API Gateway

Returns

Inventory

Orders
API Gateway Responsibility

- Authentication
- Routing
- Rate Limiting
- Logging and analytics
Anti-pattern
Service Mesh
North-South Traffic
East-West Traffic
Side-car
Proxy as sidecar:
I have a new IP now. YOLO!

I magically know all about it!

Who cares?
Error?
No worries!
Lets retry every millisecond forever

LOL. I’m dropping 99% of the retries.

I can recover from errors without drowning
Event Driven
Making Changes is Risky
Adding Services Requires Explicit Calls

Mistake Handler
1. Tell others what to do (commands)
2. Ask questions (queries)

Broadcast what I do

Others work out what to do

Queries use local cache
Events are both facts and triggers
Buying an iPad (with REST)

- Orders Service calls Shipping Service to tell it to ship item.
- Shipping service looks up address to ship to (from Customer Service)
Using events for Notification

- Orders Service no longer knows about the Shipping service (or any other service). Events are fire and forget.
Using events to share facts

- Call to Customer service is gone.
- Instead data is replicated, as events, into the shipping service, where it is queried locally.
DB for Each Microservice?

- It is safe: They are all derived from same stream of events
- Reduced dependencies
- Custom projection just the data each service needs.
- Low latency
Event Driven Microservices are Stateful
Schema
Request Driven

- 1. Tell others what to do (commands)
- 2. Ask questions (queries)

Event Driven

- Broadcast what I do
- Others work out what to do
- Queries use local cache
The medium is not the message.
This is a message

{
    sessionId: 676fc8983gu563,
    timestamp: 1413215458,
    viewType: "propertyView",
    propertyId: 7879,
    loyaltyId: 6764532
    origin: "promotion",
    ...... lots of metadata....
}
REST = HTTP + JSON = SLOW
Making Changes is Risky

Change

Change

Change
There are lots of dependencies

```
create table (use_id number, timestamp number)
```

```json
{user_id: 53, timestamp: 1497842472}
```
{user_id: 53, timestamp: "June 28, 2017 4:00pm"}

create table (
  use_id number,
  timestamp number)

new Date(timestamp)
Moving fast and breaking things

```
{user_id: 53, timestamp: “June 28, 2017 4:00pm”}
```

```
create table (  
    use_id number,  
    timestamp number)  
```
APIs between services are Contracts

In Event Driven World – Event Schemas ARE the API
So the flow is...

- **Dev**
- **Nightly build / merge**
- **Test Registry**
- **Prod Registry**
- **Prod**
Serverless
Function as a Service

Event

< / >

Response

Launch

VM

Function Instance

< / >
When nothing happens
At scale

Event

Launch

Response
Wait, this is super familiar
Up Next:
Stateful Serverless
State is required

- Dynamic Rules
- Event enrichment
- Joining multiple events
- Aggregation
How You Probably Do State

Create Order → Order Created

VM Function Instance

Launch

Cloud Data Store

select from inventory
... insert into Orders...
We can do a bit better

Select order_id, customer_name, product, quantity, price, state from orders
where state != "CLOSED"

What's the latest with my order?

Order is processing
Not shipped yet.

Maybe check DB?
But I really want this back:
Stateful Serverless

Create Order

Validate Order

Order Status

Inventory, Rules

VM
What’s Still missing?

- Durable functions everywhere
- Triggers and data from data stores to functions
- Unified view of current state
Resources

- [https://www.infoq.com/articles/service-mesh-promise-peril/](https://www.infoq.com/articles/service-mesh-promise-peril/)
- [https://blog.getambassador.io/api-gateway-vs-service-mesh-104c01fa4784](https://blog.getambassador.io/api-gateway-vs-service-mesh-104c01fa4784)
- [https://wecode.wepay.com/posts/migrating-apis-from-rest-to-grpc-at-wepay](https://wecode.wepay.com/posts/migrating-apis-from-rest-to-grpc-at-wepay)