Implementing an Event-Driven Microservices Architecture: A case study of Jet.com

NIKHIL BARTHWAL

SENIOR ENGINEER, JET.COM

Background

- Launched in July 2015
- 26 Million visitors a month, 25K orders daily
- 8 Million customers
- Have 15 million SKU's in inventory

Acquired by Walmart for $3.3 Billion in Sept 2016!

Architecture

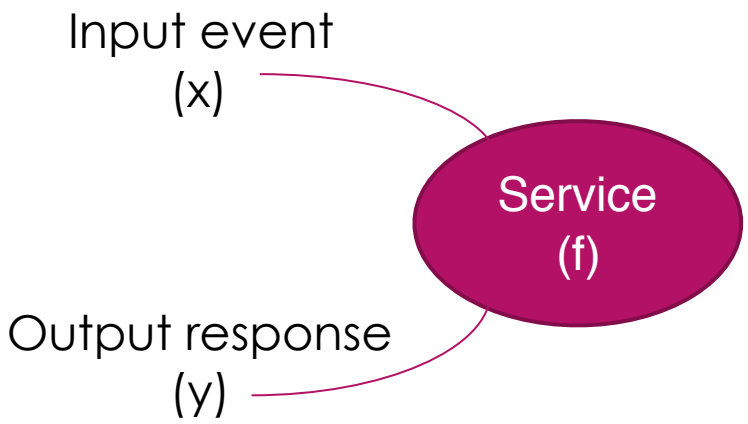
- Microservices based (Over 700+ in production)
- Event-Driven Architecture
- Event Sourcing

Technology Stack

- Runs on Microsoft Azure
- Uses .Net framework
- Use a mix of Kafka, Redis, Splunk, Event Store, ...

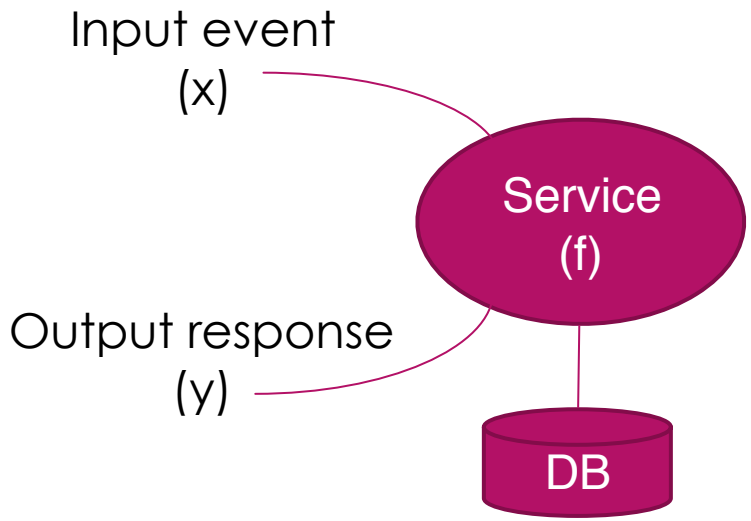
Bulk of backend implemented in F#!

A view of Microservice



Mathematical representation: y=f(x)

Pure Service (Majority)
No side effects except Logging



Mathematical representation: y, s_{out}=f(x, s_{inp})

Impure service (Minority)
Side effects like I/O to DB etc.

Why did we not use OOP?

Arguments	Counter-Arguments
OOP models application as objects with state	Microservices mostly don't have states
OOP extends imperative with encapsulation and polymorphism	
Imperative was designed for Van-Neumann style hardware	Microservices hosted on the cloud don't interact with hardware directly
Being dominant paradigm, OOP Languages like Java/C# have very good ecosystem	Languages like F#/Scala can inter-operate with C#/Java seamlessly

Modelling a Service using FP

FP Construct	Mapping to the Service world
Algebraic Data Type	Events Modelling
Functions	Services
Immutability	Events are immutable

Implementation: Example in F#

```
type input =
    | ItemName of string
    | ItemSKU of int
```

Define Input type

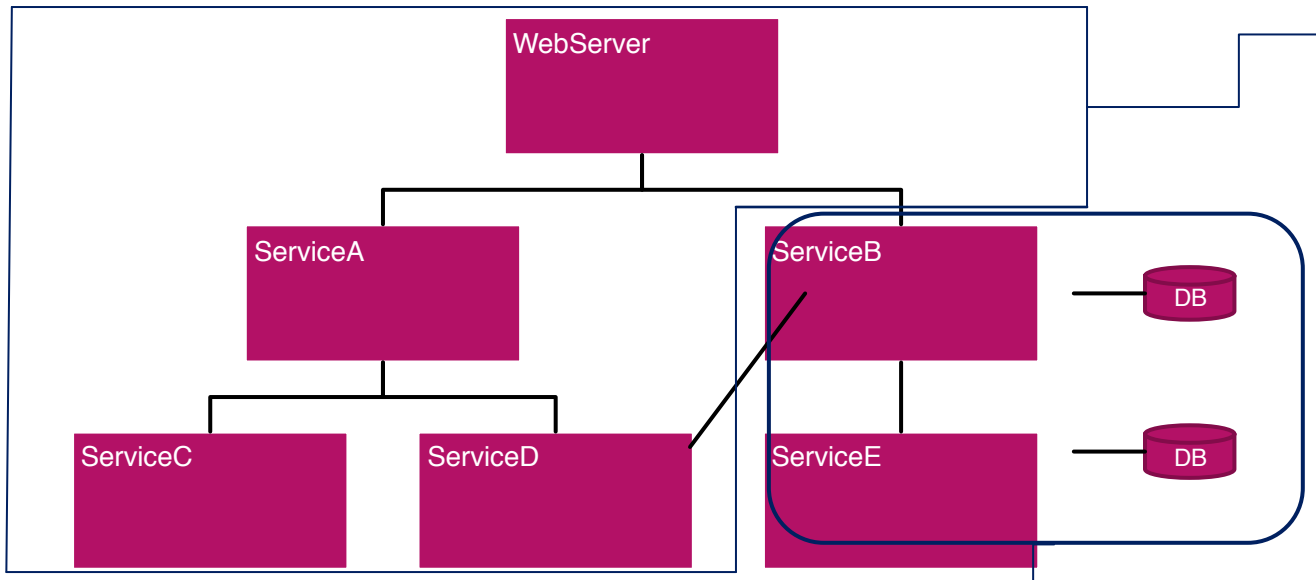
```
type output =
    | ItemInInventory of int
    | ItemNotInInventory of (System.DateTime option)
    | ItemNotSold
```

Define Output type

```
let CatalogSearch (query:input) : output =
    // the code ...
```

Write a function to convert input to output…

Testing Microservices



Pure Services:

- Being pure, behavior is very predictable
- Can be tested exhaustively!

Impure Services:

- Unpredictable, due to external state
- Certain services can't be tested exhaustively (e.g. Payment gateway)!

Benefits of using F#

- Scalability
- Productivity
- Code Correctness

Conclusion

- Very few startups scaled to Jet's size in same time
- Using F# was the most forward looking decision
- Scalability, parallelism & productivity

Questions?

NIKHIL BARTHWAL

SENIOR ENGINEER, JET.COM