

# Serverless Architecture

## *Software Architecture*

Richard Thomas

April 14, 2025

## *Oxymoron 1.* Serverless

Logic running on someone else's server.

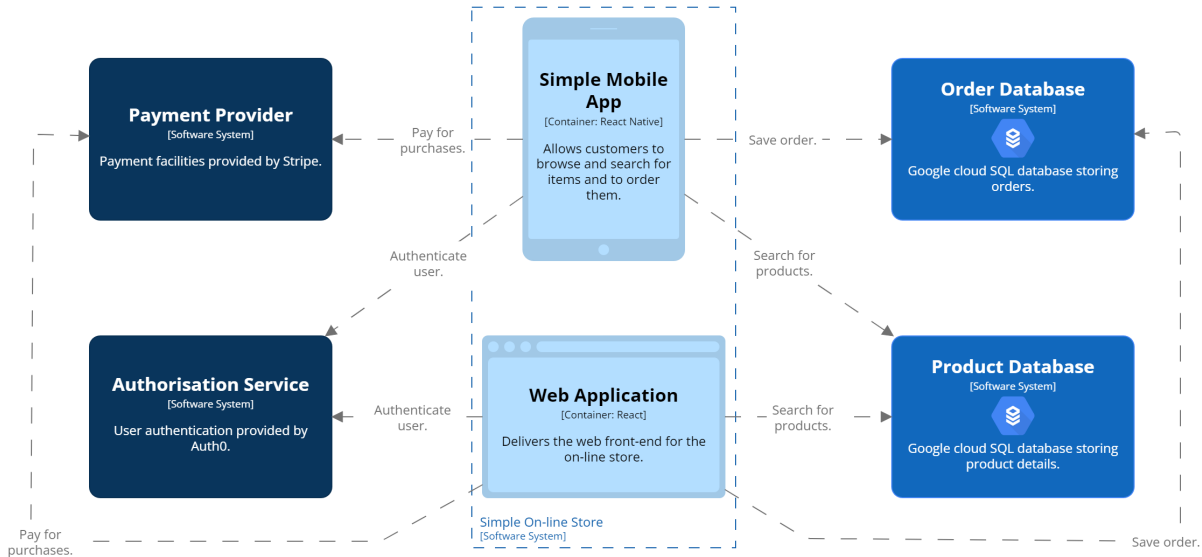
*Definition 0.* Backend as a Service (BaaS)

Cloud-hosted applications or services that deliver functionality used by an application front-end.

# BaaS Iceberg *[Brunko, 2019]*



# BaaS Example



*Definition 0.* Functions as a Service (FaaS)

Application logic that is triggered by an event and runs in a *transient*, *stateless* compute node.

# FaaS Iceberg *[Brunko, 2019]*



# FaaS Example





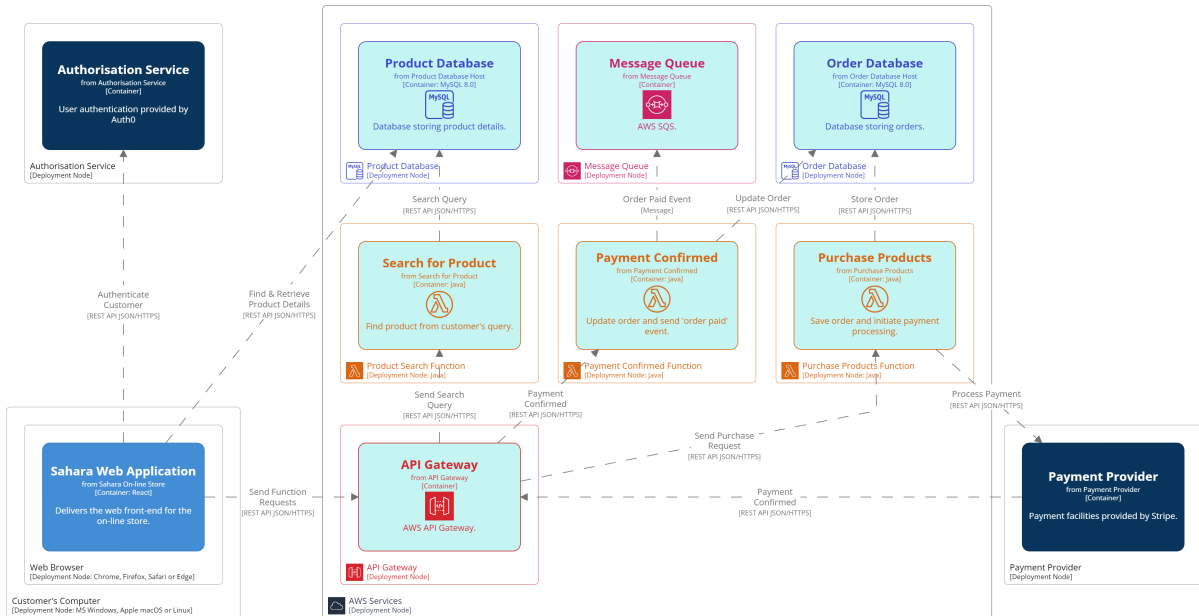
*Definition 0.* Serverless Architecture

Software system delivering functionality through BaaS or FaaS.

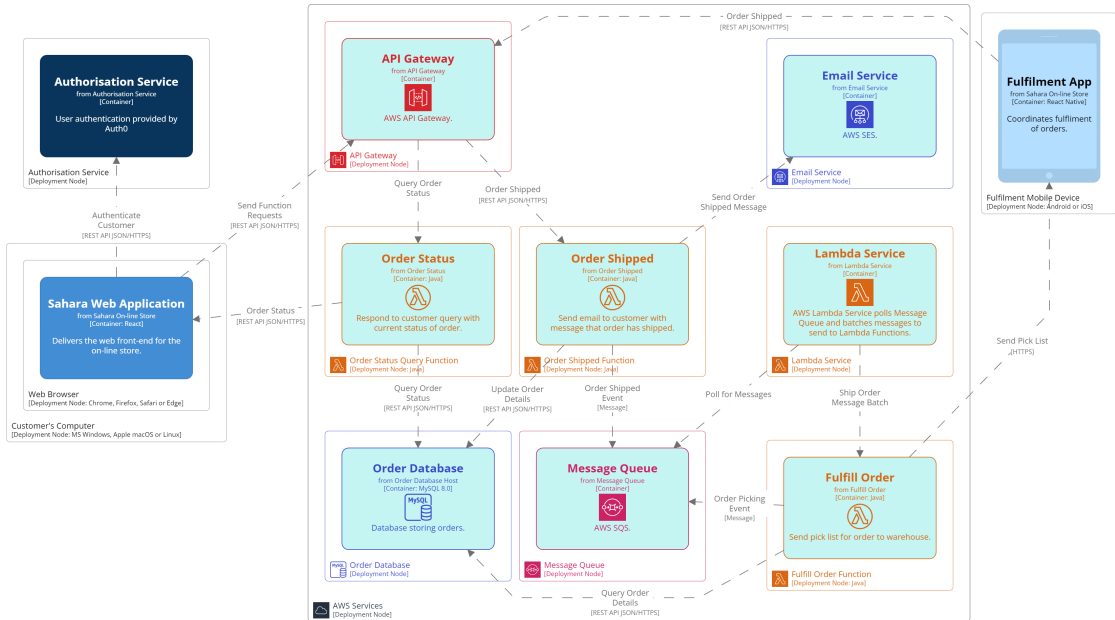
# Sahara Browse & Order — Serverless



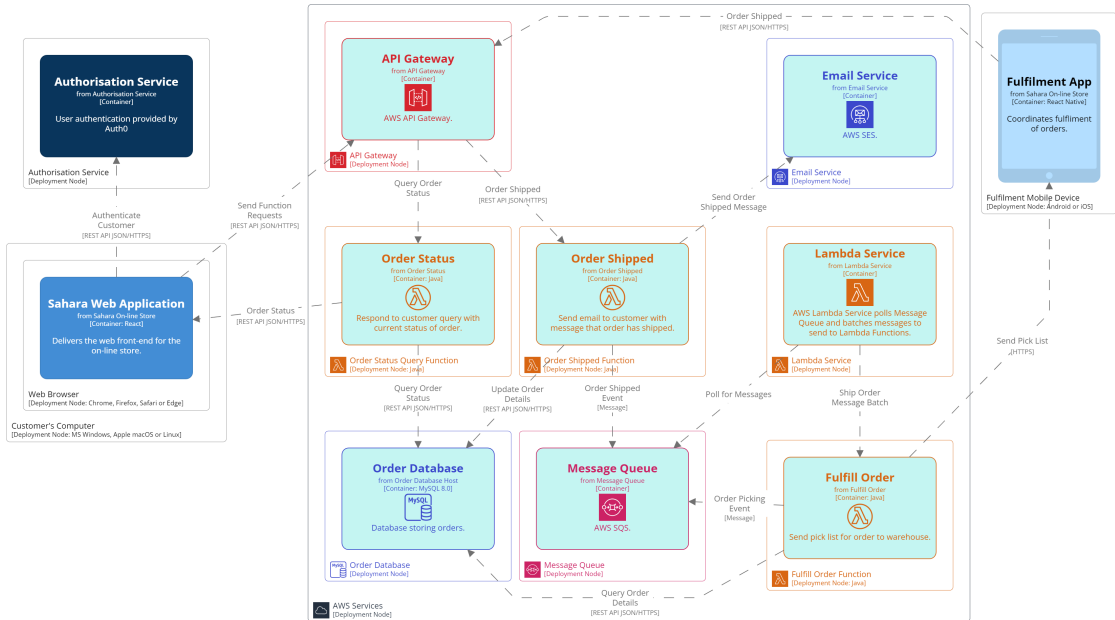
# Sahara Browse & Order — Serverless



## Sahara Fulfilment — Serverless



## Sahara Fulfilment — Serverless



## Serverless Benefits

- Automatic scaling
  - Multiple instances of function

## Serverless Benefits

- Automatic scaling
  - Multiple instances of function
- Reduced cost for dynamic loads
  - No server idle time

## Serverless Benefits

- Automatic scaling
  - Multiple instances of function
- Reduced cost for dynamic loads
  - No server idle time
- Reduced server management



## Serverless Benefits

- Automatic scaling
  - Multiple instances of function
- Reduced cost for dynamic loads
  - No server idle time
- Reduced server management
- Easier to run closer to client
  - Launch in same zone as client

## BaaS Tradeoffs

- Front-end accesses database directly
  - Front-end needs to sanitise inputs
  - Easy to spoof messages from front-end
    - Hope DB provider is secure

## BaaS Tradeoffs

- Front-end accesses database directly
  - Front-end needs to sanitise inputs
  - Easy to spoof messages from front-end
    - Hope DB provider is secure
- Application logic is in front-end
  - Less modularisation
  - Duplication of logic with multiple front-ends
    - Web, mobile, ...

## BaaS Tradeoffs

- Front-end accesses database directly
  - Front-end needs to sanitise inputs
  - Easy to spoof messages from front-end
    - Hope DB provider is secure
- Application logic is in front-end
  - Less modularisation
  - Duplication of logic with multiple front-ends
    - Web, mobile, ...
- No control over server optimisation

## FaaS Tradeoffs

- No server state
  - All state needs to be saved (e.g. Redis, S3, ...)
    - Not just persistent state

## FaaS Tradeoffs

- No server state
  - All state needs to be saved (e.g. Redis, S3, ...)
    - Not just persistent state
- Execution duration
  - Can't be long running process
    - AWS Lambda – up to 15 minutes

## FaaS Tradeoffs

- No server state
  - All state needs to be saved (e.g. Redis, S3, ...)
    - Not just persistent state
- Execution duration
  - Can't be long running process
    - AWS Lambda – up to 15 minutes
- Startup latency
  - Functions take time to start
    - Some languages worse than others (e.g. Java)

## FaaS Tradeoffs

- No server state
  - All state needs to be saved (e.g. Redis, S3, ...)
    - Not just persistent state
- Execution duration
  - Can't be long running process
    - AWS Lambda – up to 15 minutes
- Startup latency
  - Functions take time to start
    - Some languages worse than others (e.g. Java)
- Proliferation of functions
  - Loss of encapsulation



*Question*

When is serverless appropriate?

### *Question*

When is serverless appropriate?

### *Answer*

- Rich client apps with common backend
  - BaaS

### *Question*

When is serverless appropriate?

### *Answer*

- Rich client apps with common backend
  - BaaS
- High latency processing
  - Within function duration constraints

### *Question*

When is serverless appropriate?

### *Answer*

- Rich client apps with common backend
  - BaaS
- High latency processing
  - Within function duration constraints
- Apps with variable load
  - Take advantage of auto-scaling

*Question*

When is serverless *not* appropriate?

*Question*

When is serverless *not* appropriate?

*Answer*

- Quick response required
  - Can't wait for FaaS to start

### *Question*

When is serverless *not* appropriate?

### *Answer*

- Quick response required
  - Can't wait for FaaS to start
- Compute intensive processing

### *Question*

When is serverless *not* appropriate?

### *Answer*

- Quick response required
  - Can't wait for FaaS to start
- Compute intensive processing
- Apps with steady load
  - Server-based approaches are cheaper



### *Self-Study Exercise*

- Redesign your scalability assignment to be serverless.
  - What parts of your design would benefit from being serverless?
- Implement your revised design.

## Pros & Cons

Extensibility



Reliability



Interoperability



Scalability



Deployability



Modularity



Testability



Maintainability



Security



Simplicity



## References

[Brunko, 2019] Brunko, P. (2019).

Serverless architecture: When to use this approach and what benefits it gives.

<https://apiko.com/blog/serverless-architecture-benefits/>.