Serverless Applications

Richard Vasek

BaaS

BaaS

Backend as a Service

BaaS

Backend as a Service

- Cloud accessible DBs (Firebase)
- Auth Services (Auth0, AWS Cognito)

FaaS

FaaS

Function as a Service

FaaS

Function as a Service

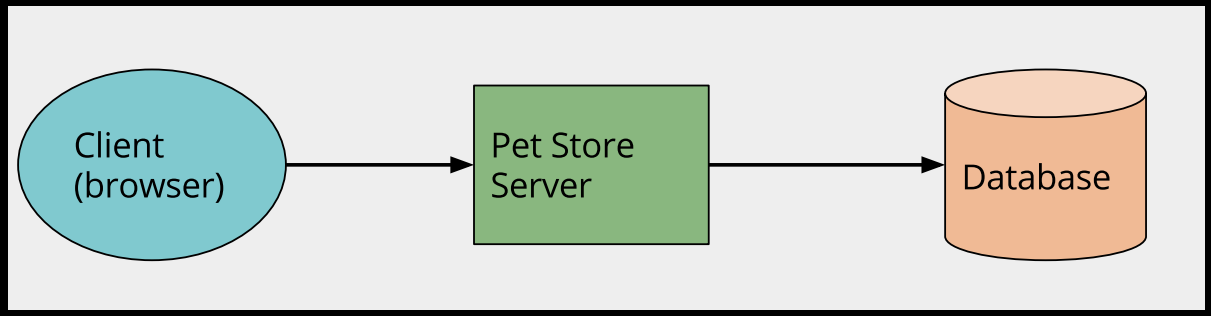
- Small code
- Stateless container
- Event triggered
- Managed by 3rd party

Serverless

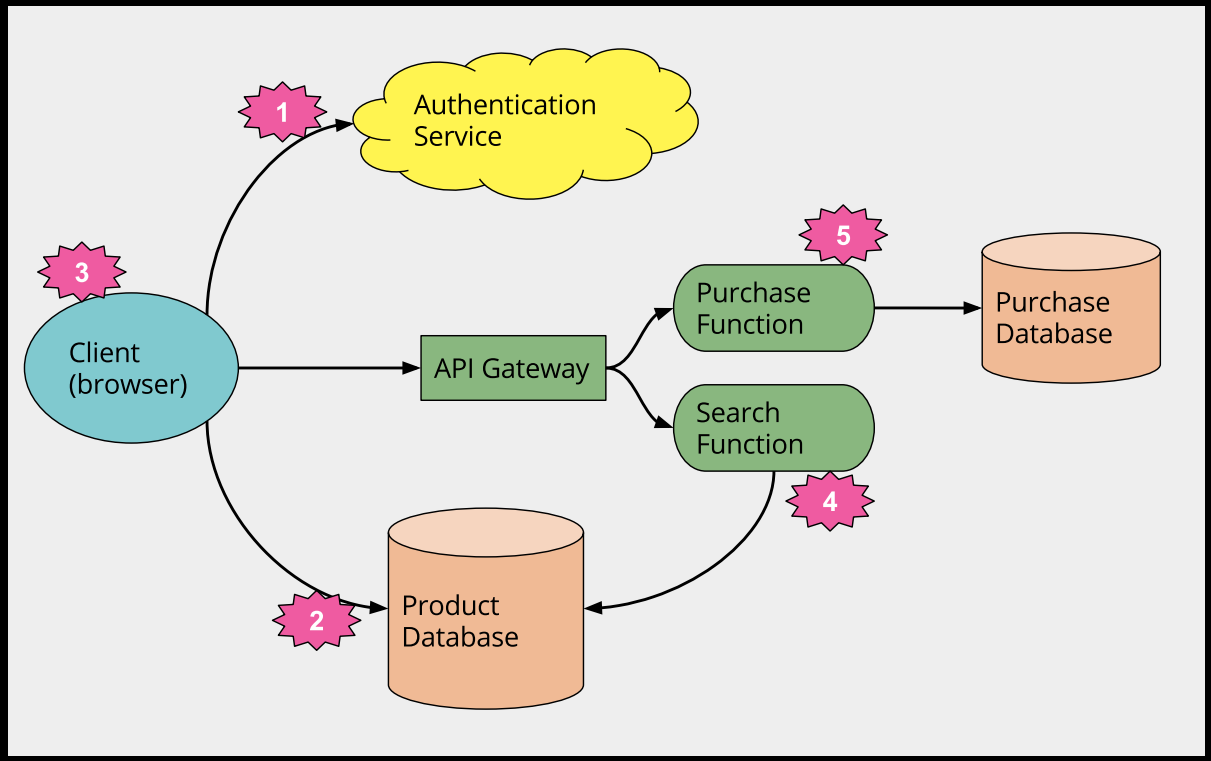
Serverless

- BaaS
- FaaS

3 Tier Example

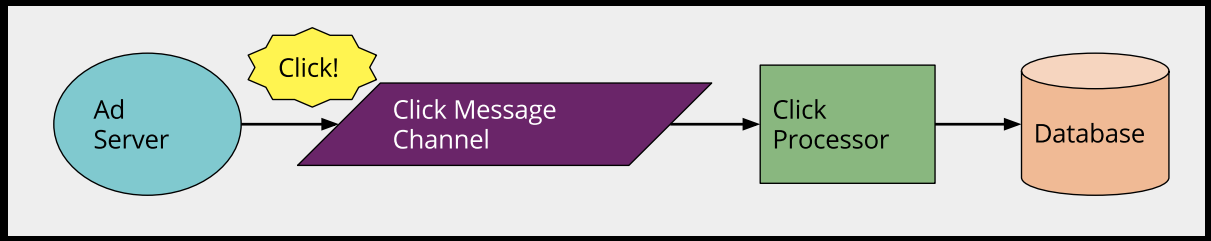


Serverless Example



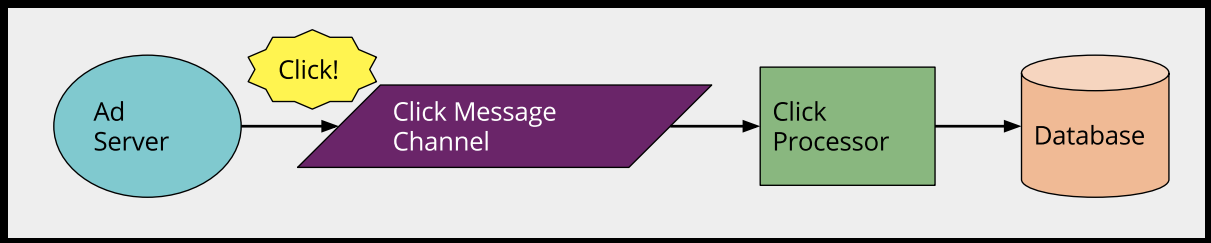
Message-driven App

Classic

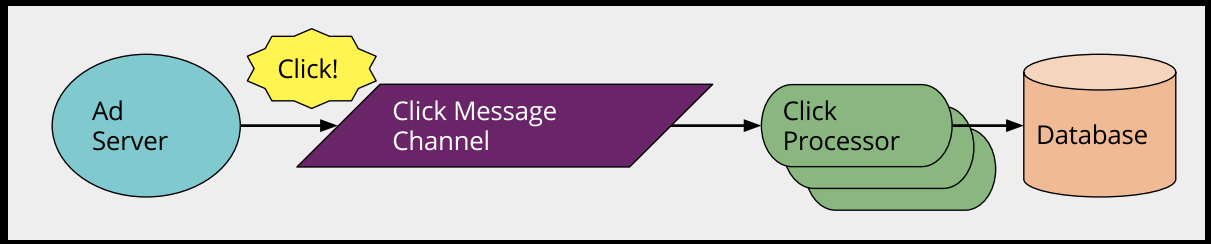


Message-driven App

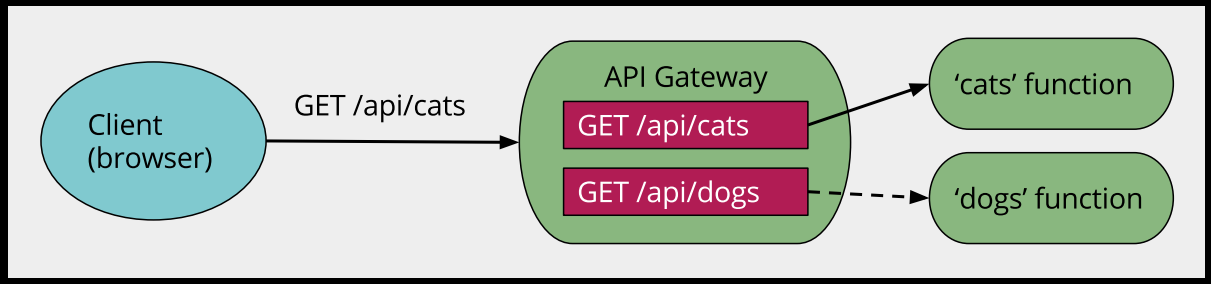
Classic



Serverless



API Gateways



- Routing requests
- Authorization
- Input validation
- Response code mapping

Scaling FaaS

- Automatically managed
- Transparent
- Fine grained

Costs

- Economy of Scale effect
- Reduced development cost
- Scaling costs
- Never pay for idle

AWS Pricing

- $0.0000002 per 100ms @ 128MB
- $0.20 per 1 million requests
- First 1M per month are free

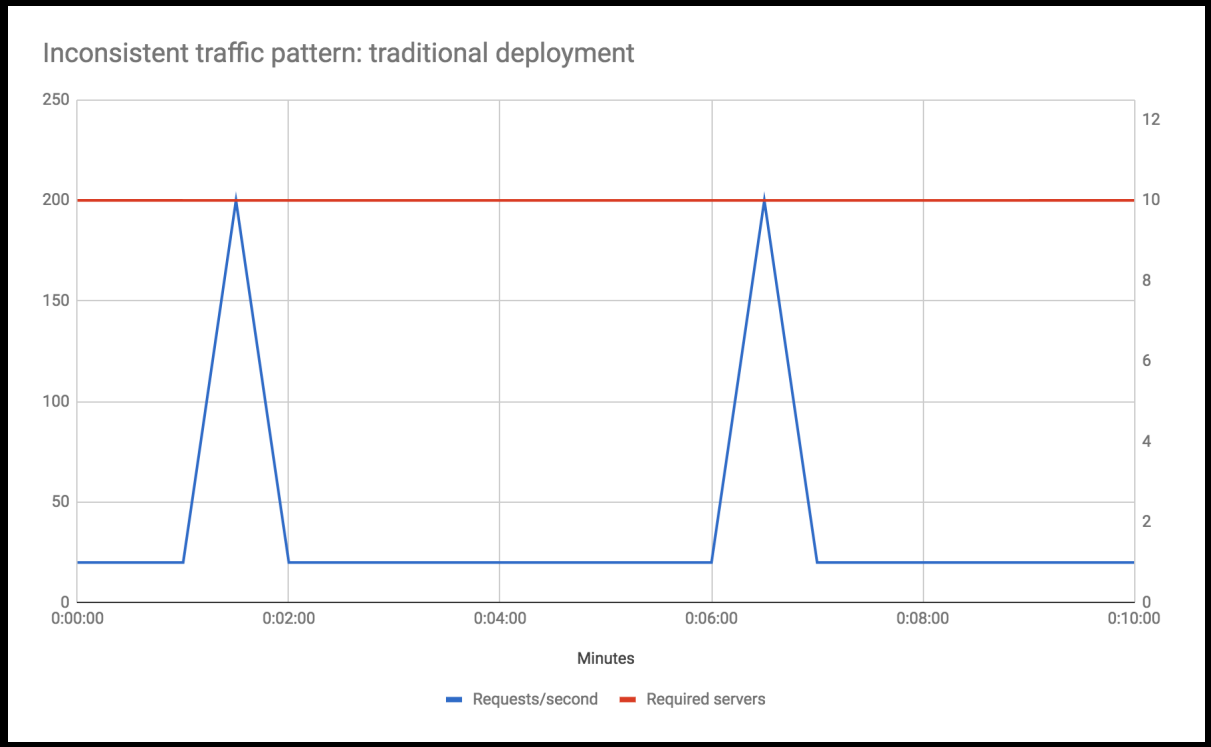
Optimization

1. You can clearly see which function is slow
2. Optimize 1s to 200ms
3. Immediately pay 80% less

Fine graded scaling

- Occasional requests
  - You don't pay when no requests
- Inconsistent traffic
  - Scale what's needed for time it's needed

Inconsistent traffic



Cheap experiments

- Pay for usage
- Replicate production for 0 cost
- Run multiple versions of code in production

Design around services

- Play arbitrage with different charging models
  - Lambda: #requests, time, memory, transfer
  - API GW: #requests, transfer
  - S3: transfer
  - Cognito: #users
  - IOT GW: #messages

e.g. Client file upload

1. Lambda returns secure S3 url
2. User uploads to S3 directly
3. You don't pay CPU time for S3, just transfer

Nice Right?

- Rainbows
- Unicorns
- All things shiny so far

But

Vendor control

- System downtime
- Unexpected limits
- Cost changes
- Loss of functionality
- Forced API upgrades

Vendor lock-in

- Hard to migrate to different vendor
- Multi-cloud is expensive

Startup Latency

- Can be from ms to s
- Cold
  - Create new container
  - (Run JIT)
- Warm
  - Reusing running instance

Security concerns

- Using BaaS database from client
- IAM policies

DoS yourself

1. AWS lambda instances limit is per AWS account (1000 by default)
2. Same account for production and test
3. Run load test on test env
4. DoS production

Memory vs CPU

- Need 50MB RAM

Memory vs CPU

- Need 50MB RAM
- So let's configure 128MB right?

Memory vs CPU

- Need 50MB RAM
- So let's configure 128MB right?
- Wrong

GCloud

128 MB	256 MB	512 MB	1 GB	2 GB
200 MHz	400 MHz	800 MHz	1.4 GHz	2.4 GHz
Testing	Small simple functions	Functions with moderate resource needs	Balance of speed and cost	Compute-intensive tasks

Testing

- Unit testing is easy
- Integration testing is hard
- Cloud-based testing not local

It's all still kinda new

- Not many patterns
- Not many best practices
- Incomplete tooling

Demo Time