Sls

by Richie


Serverless All The Things
Paint X lite

SaaS

Software as a Service

SaaS

Software as a Service

- Buy, don't deploy
- Google Apps, Dropbox, Stripe (payments)

IaaS

Infrastructure as a Service

IaaS

Infrastructure as a Service

- Assemble before deploy
- Cloud servers (e.g. VMs with Ubuntu)
- Direct access to servers, storage,...
- Exoscale, parts of AWS

PaaS

Platform as a Service

PaaS

Platform as a Service

- Abstracts much of the work of dealing with servers
- Heroku, OpenShift

BaaS

Backend as a Service

BaaS

Backend as a Service

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

FaaS

Function as a Service

FaaS

Function as a Service

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

Serverless

- BaaS
- FaaS

What it is all about?

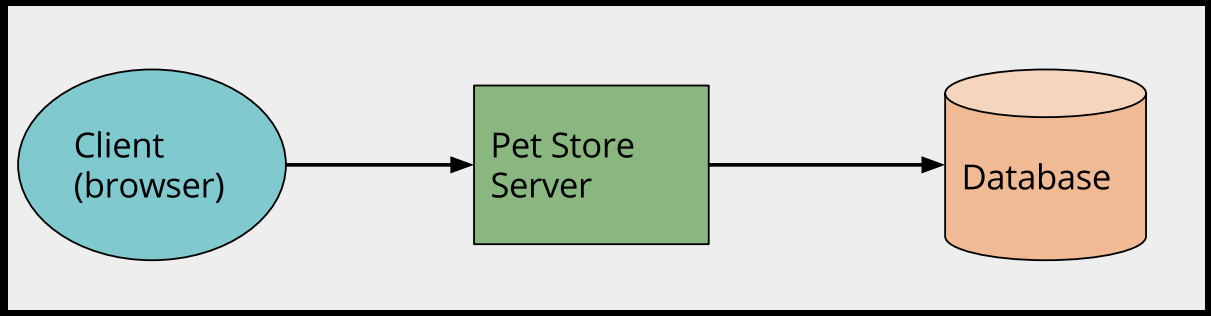
Getting shit done.


BUT SERVERLESS STILL HAS SERV...
THAT'S NOT THE POINT
imgflip.com

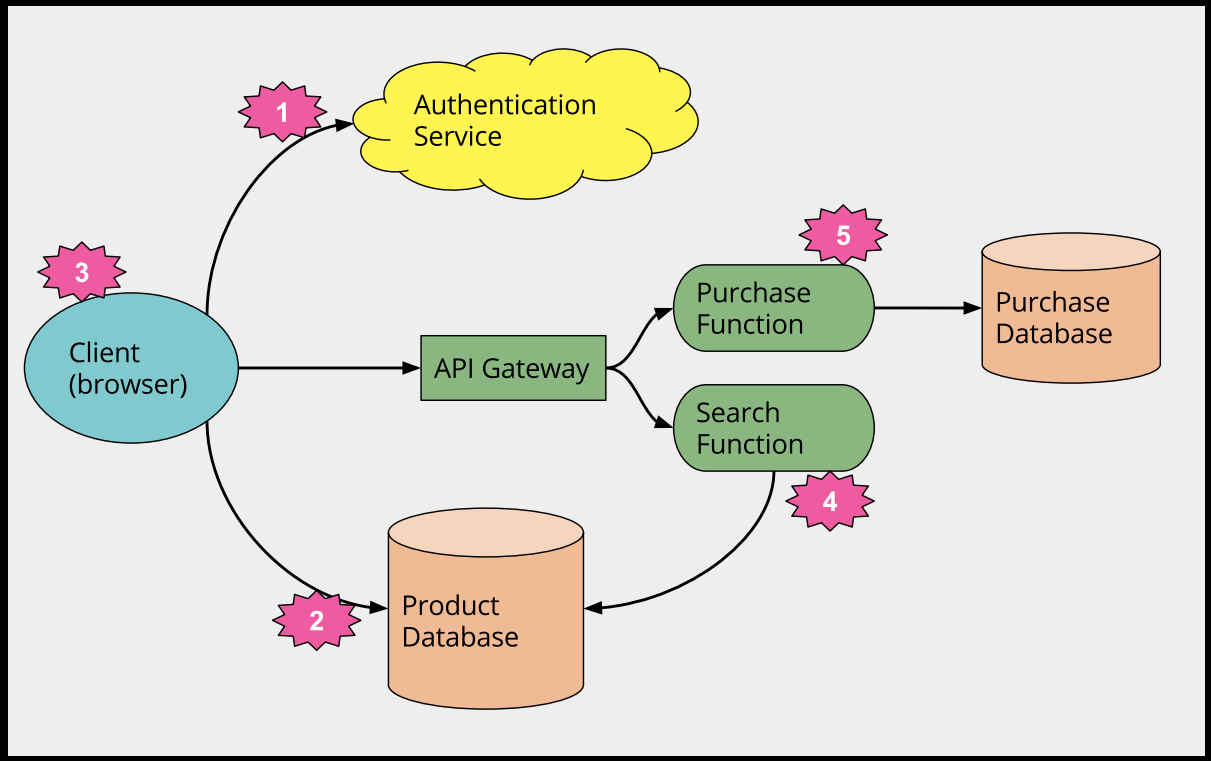
#NoOps

- There is always Ops, you just outsource it

3 Tier Example

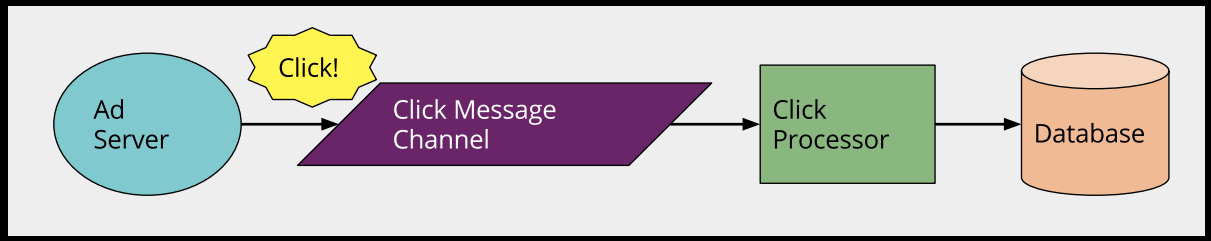


Serverless Example



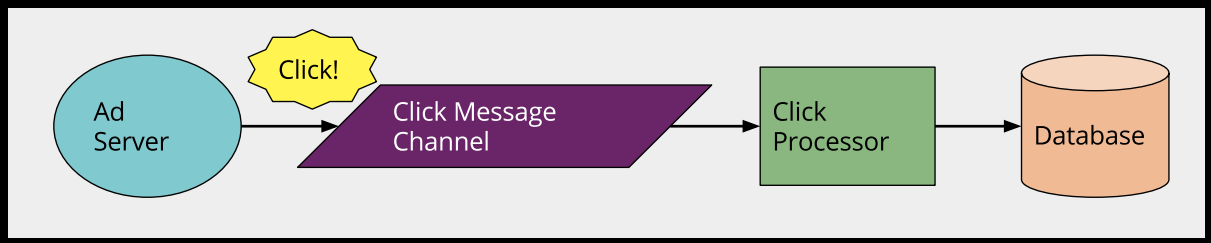
Message-driven App

Classic

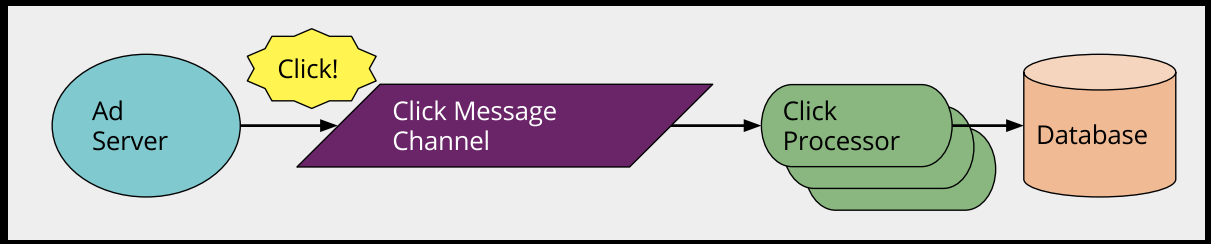


Message-driven App

Classic



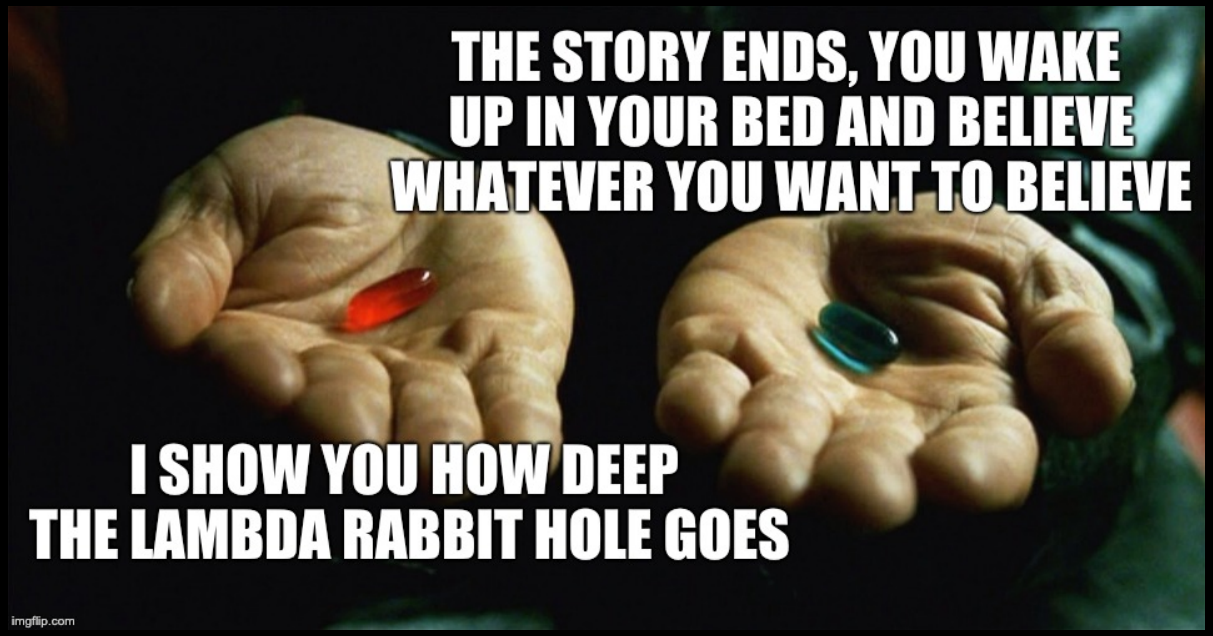
Serverless



Lambda Functions

WTF that even means?

After this, there is no turning back.


THE STORY ENDS, YOU WAKE
UP IN YOUR BED AND BELIEVE
WHATEVER YOU WANT TO BELIEVE
I SHOW YOU HOW DEEP
THE LAMBDA RABBIT HOLE GOES
imgflip.com

Lambda

Lambda

- Comes from Lambda calculus (1930s)

Lambda

- Comes from Lambda calculus (1930s)
- Formal system in mathematical logic for expressing computation based on function abstraction and application using variable binding and substitution.

Lambda

Lambda

- Functions: 1 -> 1

Lambda

- Functions: 1 -> 1
- and nothing else

Lambda

- Functions: 1 -> 1
- and nothing else
- (λx.λy.(λz.(λx.z x) (λy.z y)) (x y))

What?

What?

- x - variable
- M - lambda term
- (λx.M) - function definition
- (M N) - apply a function to an argument

Where are all the stuff?

Where are all the stuff?

- TRUE
- FALSE
- IF-ELSE
- FOR LOOP
- BINARY OPERATORS
- NUMBERS???

Here you go

Here you go

- TRUE := λx.λy.x
- FALSE := λx.λy.y
- AND := λp.λq.p q p
- OR := λp.λq.p p q
- NOT := λp.p FALSE TRUE
- IFTHENELSE := λp.λa.λb.p a b

Numbers?

Church numerals

- 0 := λf.λx.x
- 1 := λf.λx.f x
- 2 := λf.λx.f (f x)
- 3 := λf.λx.f (f (f x))
- 4 := λf.λx.f (f (f (f x)))

Nah, this is all just academic nonsense right?

Well

Let me show you some JS

Demo time!

What is lambda today?

- `() => {}`
- AWS FaaS is called Lambda

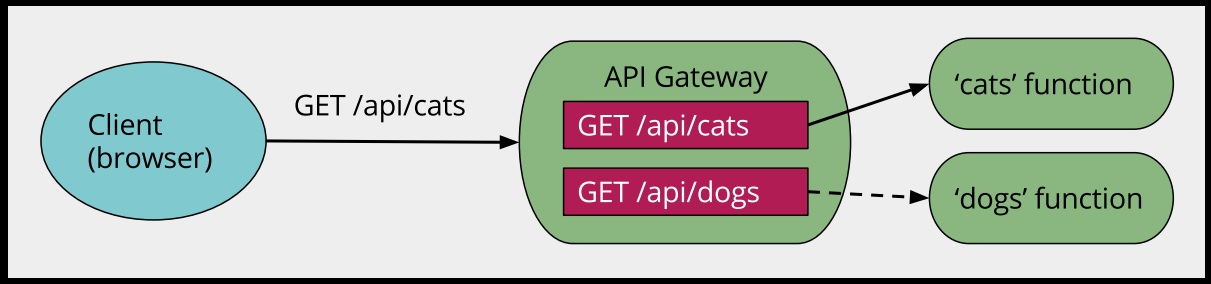
Function State

- Stateful, but
- External state (AWS S3, DB)

Startup Latency

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

API Gateways



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

GCloud Demo

PaaS and Serverless

- If your PaaS can efficiently start instances in 20ms that run for half a second, then call it serverless.

Scaling FaaS

- Automatically managed
- Transparent
- Fine grained

Time to Recover -> Time to Start

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

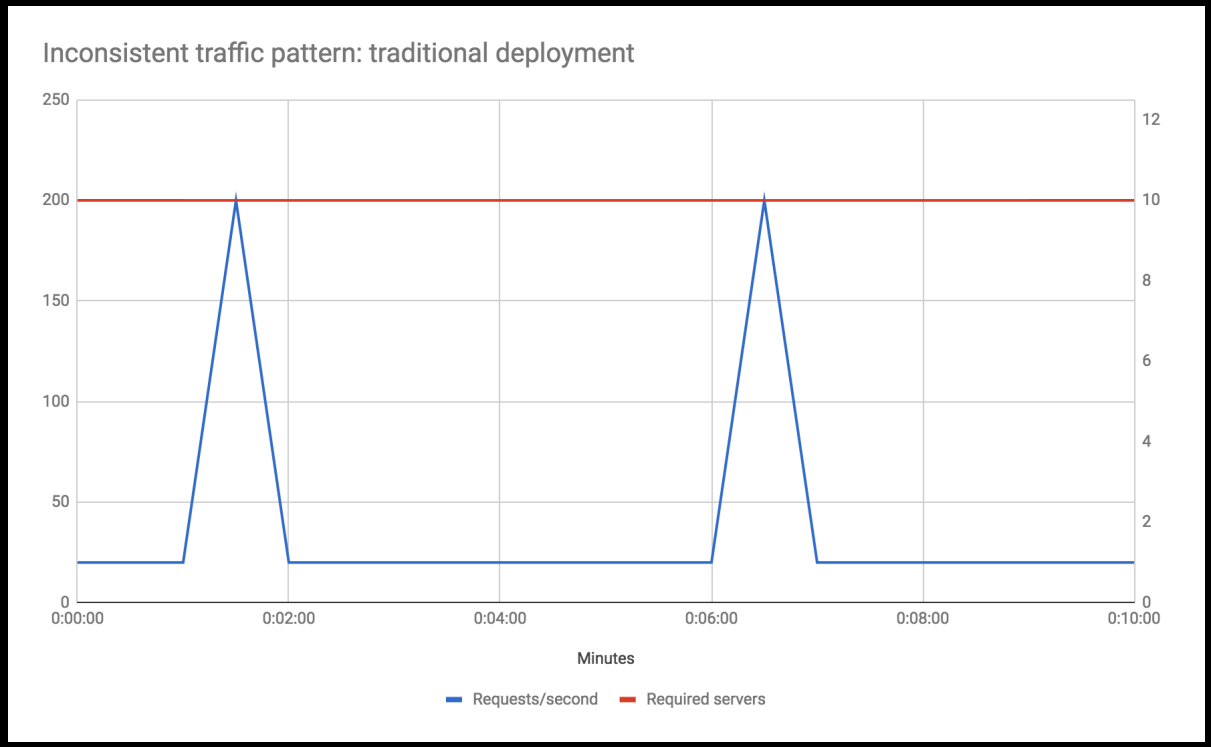
Optimalization

1. You can clearly see which function is slow
2. Optimize 1s to 200ms
3. Imidiately 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
  - 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

Cognito and IOT GW

- State in Cognito
- IOT GW does not care about size

Let clients connect to "back-end" resources

And use 1000s of CPUs for free

Nice Right?

- Rainbows
- Unicorns
- All things shiny so far

Nice Right?

- Rainbows
- Unicorns
- All things shiny so far
- About to get slapped around the face by the wet fish of reality

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

Security concerns

- Using BaaS database from client
- IAM policies
- Identity-Based 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

Execution duration

- Limited to ~5 minutes
- No signs of changing it

Startup latency

- Cold starts
- Significant concern for JVM

Memmory vs CPU

- Need 50MB
- 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

AWS

- No CPU guaranties
- Experience similar to GCloud

Testing

- Unit testing is easy
- Integration testing is hard
- Cloud-based testing not local
- Should we switch to cloud IDE?

Over-ambitious API gateways

1. You pay per request not CPU time
2. You put a lot of logic to API gateway
3. Yaml hell
4. Hard to maintain

It's all still kinda new

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

Serverless FW Demo