The Serverless Revolution for JavaScript Developers

Pam Selle, IOpipe
@pamasaur | thewebivore.com

The Serverless Revolution ~~for JavaScript Developers~~

Pam Selle, IOpipe
@pamasaur | thewebivore.com

5 million requests for $5

(really $4.55)

http://serverlesscalc.com/

Q: Over how long?
A: Doesn’t matter

“NoOps”

“No server is easier to manage than ‘no server’”

- Matt Wood, GM Product Strategy, AWS

Source: http://www.businessinsider.com/amazon-web-services-lambda-explained-2015-11

Serverless architectures allow you to focus on deploying code that Does Things

Rather than focusing on server management ;)

What is serverless *really*?

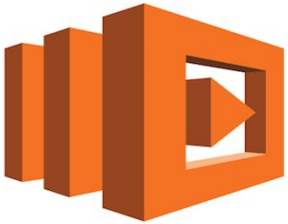
FaaS
&
BaaS

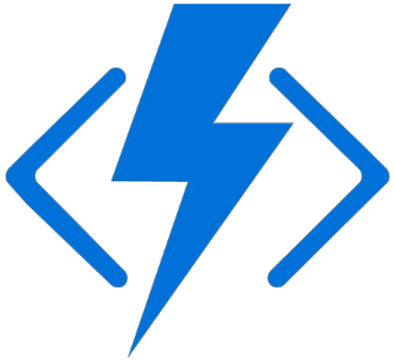
FaaS: Functions as a Service

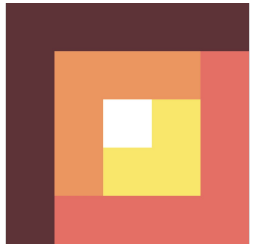
aka: “serverless computing”

FaaS:
Running code on-demand in response to a defined trigger

FaaS Providers







Left to right: Google Cloud Platform, AWS Lambda, Azure Functions, Webtask, Iron.io, IBM OpenWhisk

BaaS: Backend as a Service

aka: “going serverless”

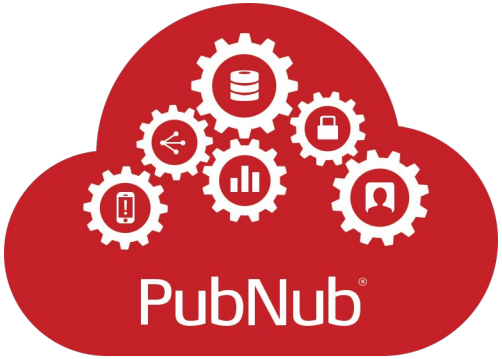
BaaS:
Relying on other services for your backend needs

BaaS offerings

Auth0

Firebase

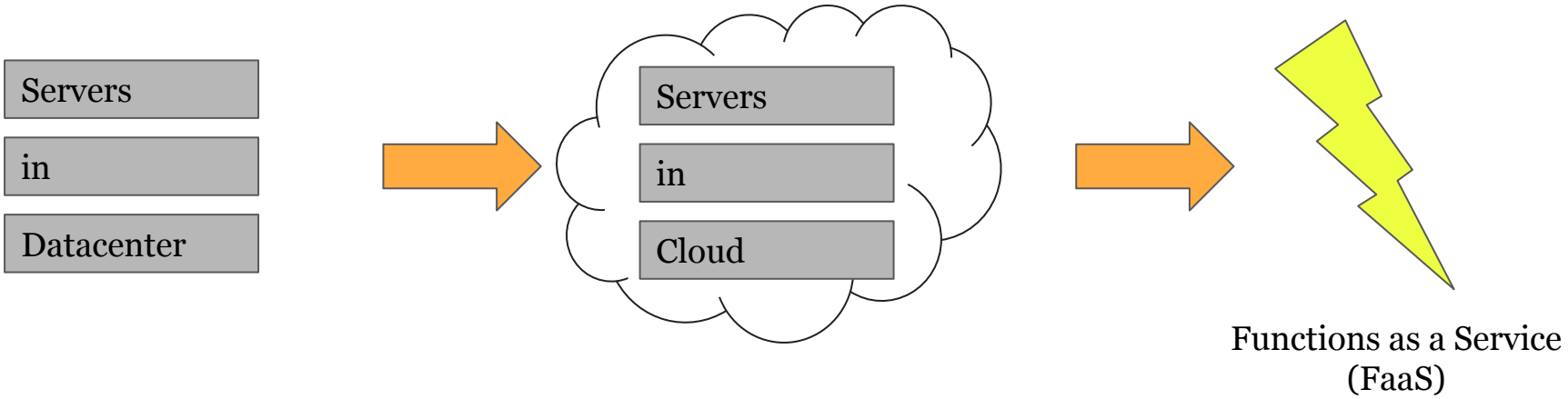
filestack



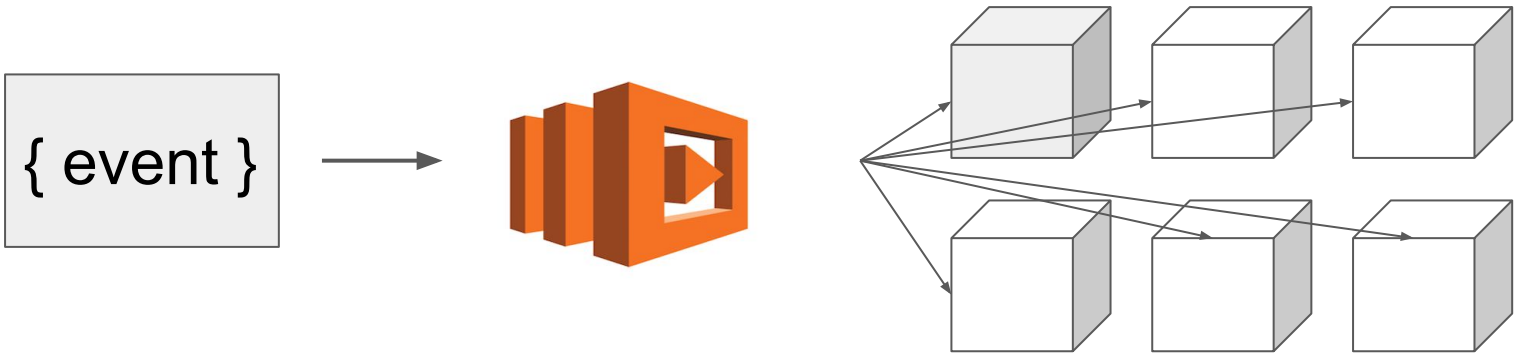
Lots more: https://github.com/anaibol/awesome-serverless

The rise of FaaS

How we got to now



How your code runs on FaaS



Up to 1000 concurrent
(can be raised)

Why does FaaS work now?

- Container technology enabled FaaS
- Friendly to providers for efficient use of infrastructure
- Metering model cost-friendly to customers
- NoOps/“no server” reduces operations costs

2014 Release

AWS Lambda is a compute service that **runs your code in response to events** and automatically manages the compute resources for you [...] With Amazon Lambda, you pay only for the requests served and the compute time required to run your code. Billing is **metered in increments of 100 milliseconds**, making it cost-effective and easy to scale automatically from a few requests per day to thousands per second.

(Source: https://aws.amazon.com/releasenotes/AWS-Lambda/8269001345899110, emphasis added)

FaaS Pricing

Cost = Invocations + Compute time

What's a GB-second?

- GB-seconds are a measure of compute time
- 1 second with 1GB of memory provisioned = 1 GB-s
- If you provision your funcs to use more memory, you use more GB-s each running second (billed in 100 ms)

Memory (MB)	% GB-s	Cost for 100ms (Lambda price)
128	12.5%	$0.000000208
1024	100%	$0.000001667

AWS Lambda

- Original, released in 2014
- Integrates seamlessly with many AWS services: API Gateway, Kinesis, DynamoDB, SNS, many more
- Languages: NodeJS, Python, Java, C#

Free tier invocations	1 million
Free tier compute time	400k GB-s
Invocations	$0.20/million ($0.0000002 each)
Compute time	$0.00001667/GB-second

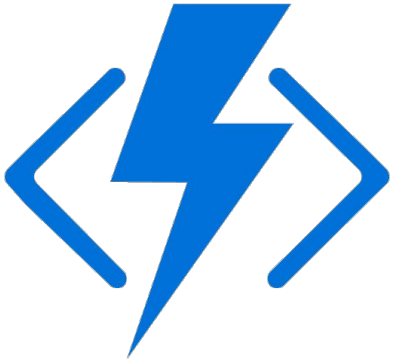
Google Cloud Functions

- HTTP triggers, Pub/Sub, Storage
- Local emulator (alpha)
- Languages: NodeJS

Free tier invocations	2 million
Free tier compute-time	400k GB-s
Invocations	$0.40/million ($0.0000004 each)
Compute time	$0.00001667/GB-second
Outgoing network requests	$0.12/GB (5GB free)

Azure Functions

- HTTP/webhooks (incl. Defaults for Slack, GitHub), Schedules, Storage, Azure Event Hub, Queues
- Languages: NodeJS, C# (but also F#, Python, PHP, Bash, Batch, and PowerShell)



Free grant invocations	1 million
Free grant compute-time	400k GB-s
Invocations	$0.20/million ($0.0000002 each)
Compute time	$0.00001667/GB-second

Don’t want to run on a public cloud?

- Apache OpenWhisk
- Run your own NoOps infrastructure
- https://github.com/openwhisk/openwhisk

APACHE
OpenWhisk™

Which one should I use?

- AWS is mature, focus of much tooling
- GCF has a larger free tier

¯_(ツ)_/¯ try them out!

Deploying a serverless function

Deploying functions

1. Upload the code (default limit 50mb for Lambda)

Deploying functions

1. Upload the code

(that’s it)

Aside: If what runs is what you upload, what does that mean about binaries?

Deploying functions (more realistic)

- Use tooling for CLI and CI/CD
- Provision and delegate resources
- Clean up resources as necessary

Deployment options

- Serverless Framework
- Apex
- Zappa (Python)
- Chalice (Python, AWS)
- ClaudiaJS (Node)
- Shep
- (there are more!)

Basic code for a FaaS

```
console.log('this prints on a coldstart')

exports.hello = function(event, context, callback) {
 console.log('processing event: %j', event)
 // callback(error, success)
 callback(null, { hello: 'world' })
 // context.fail()
 // context.succeed()
};
```

Basic code for a FaaS

```
console.log('this prints on a coldstart')

exports.hello = function(event, context, callback) {
 console.log('processing event: %j', event)
 // callback(error, success)
 callback(null, { hello: 'world' })
 // context.fail()
 // context.succeed()
};
```

Basic code for a FaaS

```
console.log('this prints on a coldstart')

exports.hello = function(event, context, callback) {
 console.log('processing event: %j', event)
 // callback(error, success)
 callback(null, { hello: 'world' })
 // context.fail()
 // context.succeed()
};
```

Basic code for a FaaS

```
console.log('this prints on a coldstart')

exports.hello = function(event, context, callback) {
 console.log('processing event: %j', event)
 // callback(error, success)
 callback(null, { hello: 'world' })
 // context.fail()
 // context.succeed()
};
```

Basic code for a FaaS

```
console.log('this prints on a coldstart')

exports.hello = function(event, context, callback) {
 console.log('processing event: %j', event)
 // callback(error, success)
 callback(null, { hello: 'world' })
 // context.fail()
 // context.succeed()
};
```

Basic deployment code for a FaaS using Serverless Framework

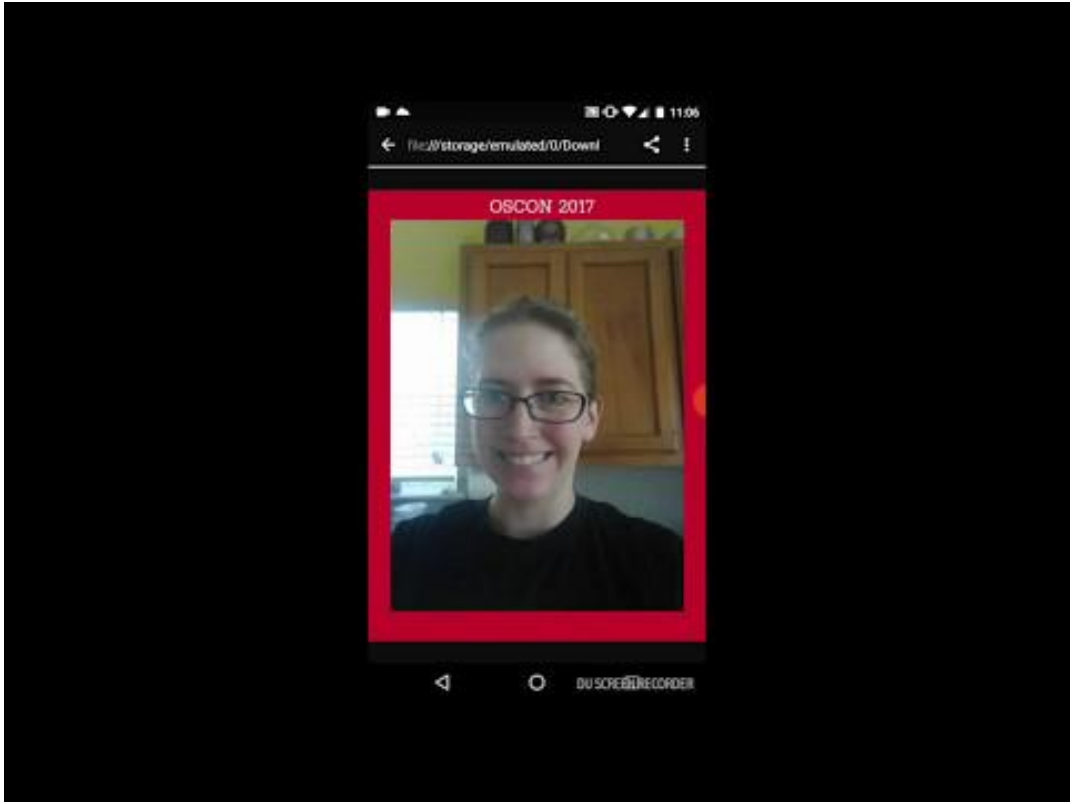
```
service: ourService

provider:
  name: aws
  runtime: nodejs6.10

functions:
  ourFunction:
    handler: handler.hello # assuming file is called “handler”
    memorySize: 128
    events:
      - http:
          path: hello
          method: get
```

```
$ sls create --template aws-nodejs
```

Demo:
OSCON Photo Booth!



https://youtu.be/YjHxAlllFLo

Make your own photo booth selfie:
pselle.github.io/oscon-photo-booth

Disclaimer! It's slow!

Check out the code:
github.com/pselle/oscon-photo-booth

Ways to use serverless computing

- Serving APIs
- Mobile apps
- IoT
- Data analysis
- Operations tasks (ex. cleanup)

… or a serverless longboard :)

https://twitter.com/nodebotanist/status/851571198983122945

FaaS in production

Bustle

Women's interest website with 50MM+ readers/month

- Converted from a monolith architecture that was difficult to scale
- ~12 APIs in production
- 100 million events per day
- Leveraging: GraphQL, Kinesis, API Gateway

"The size of our team is half of what is normally needed to build and operate a site of this scale."

- Tyler Love, CTO

https://aws.amazon.com/solutions/case-studies/bustle/
https://thenewstack.io/bustle-migrated-100-million-events-per-day-product-serverless/

Coca-Cola

Hundreds of brands, vending powered by serverless

Major productivity gains from going serverless!

- Went from 24% unplanned work to 6%
- 68% of time spent on biz projects (vs 39%)

re:Invent
https://www.youtube.com/watch?v=yErmil00DYs

MLB Advanced Media

Captures and analyzes every play

"Lambda is really clever. It's where we take the raw data, do some cleaning up and error detection, then create the metrics that bring more insights into plays—the throws, the player's acceleration rate, the top running speeds … We're **accessing a truly big data mine**, and have yet to scratch the surface."

https://aws.amazon.com/solutions/case-studies/major-league-baseball-mlbam/

What should you check out next?

- Deploy a serverless function (or lots of them)
- Explore event sources – the options are endless
- Look into GraphQL
- The awesome-serverless list
- More further reading: https://martinfowler.com/articles/serverless.html

Challenges of Serverless (what's next)

- State
- Orchestration
- Deployment
- Testing
- Debugging
- Networking
- Monitoring

Thank you!

@pamasaur

@iopipes

www.iopipe.com