In the land of Serverless, who uses Zappa is king!

@jonatasbaldin
Developer at Cheesecake Labs

Serverless computing

From Wikipedia, the free encyclopedia

Serverless computing is a cloud computing execution model in which the cloud provider dynamically manages the allocation of machine resources, and bills based on the actual amount of resources consumed by an application, rather than billing based on pre-purchased units of capacity.[1] It is a form of utility computing.


VIRTUAL SERVERS
CLOUD COMPUTING
CONTAINERS
SERVERLESS

BaaS

Backend as a Service

FaaS

Function as a Service

Serverless Principles

Single Purpose Functions

Event-Driven Architectures

Abstracted Servers

Ephemeral Environments

Pay by Execution

Builtin Scalability

Commercial Providers





Open source Providers

IronFunctions


amazon
web services


amazon
web services

amazon web services

amazon web services

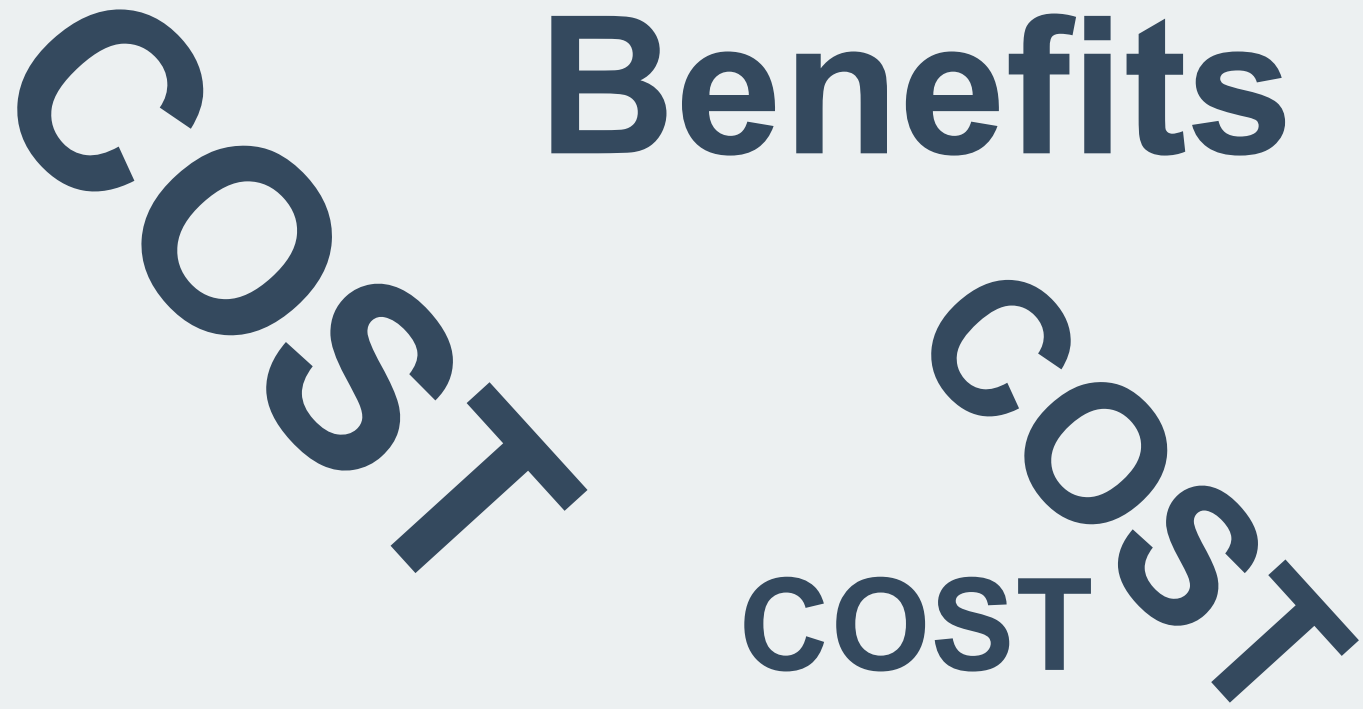
Use Cases

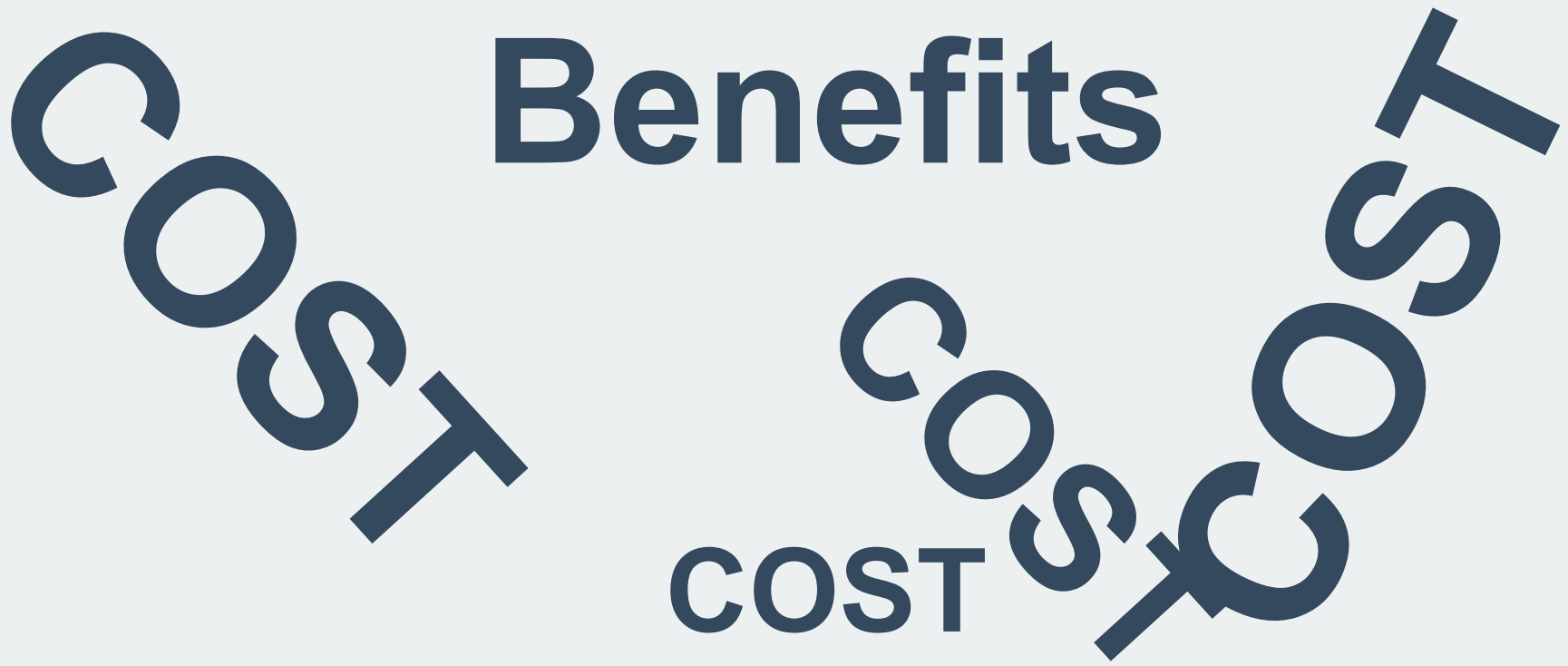
Web Backends
Bots
Data Processing
IoT

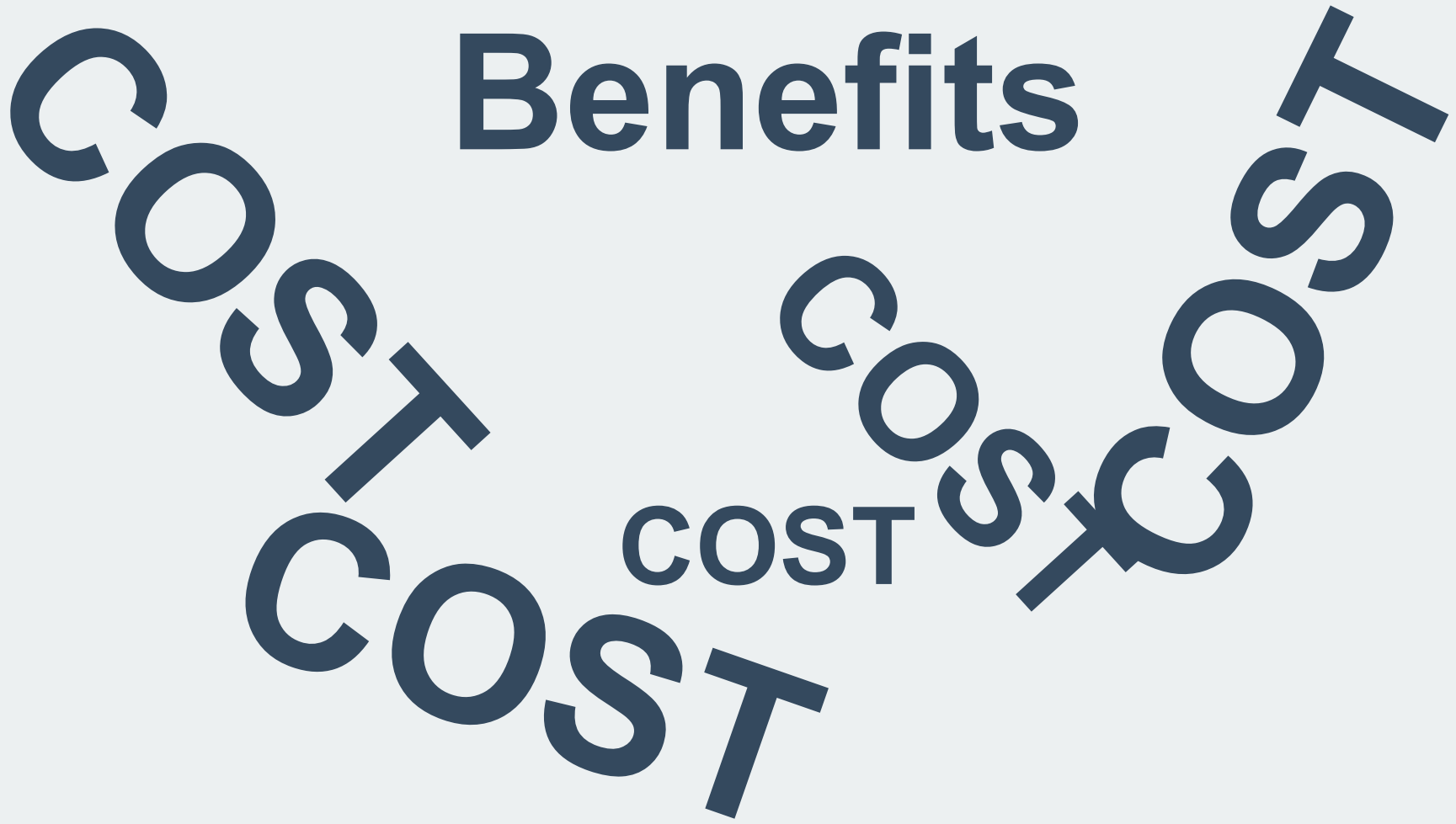
Benefits

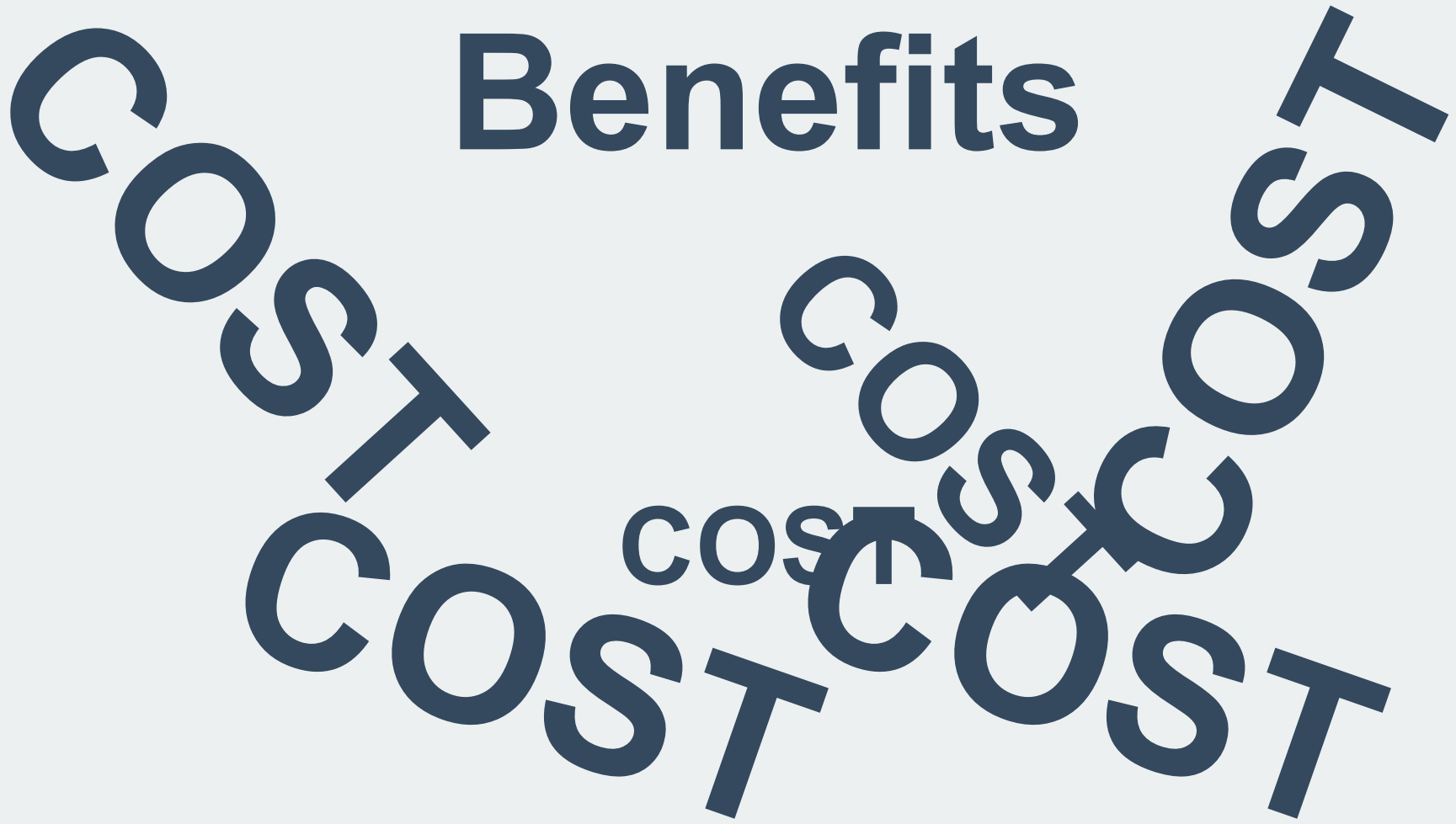
COST

Benefits


Benefits
COST
COST
COST


Benefits
COST
COST
COST
COST


Benefits
COST
COST
COST
COST
COST
COST

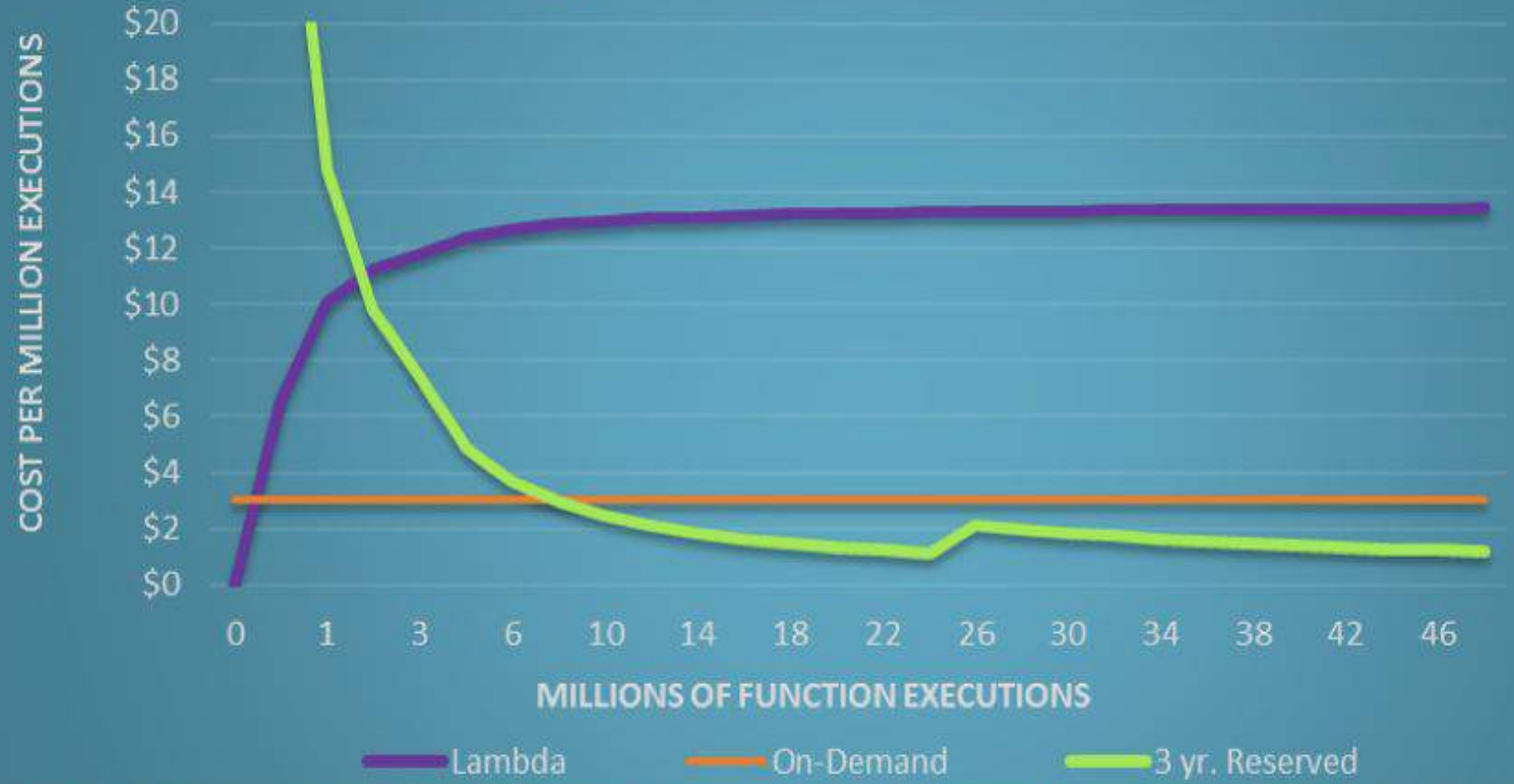

Benefits
COST
COST
cost
COST
COST
COST
COST


Benefits
COST
COST
COST
COST
COST
COST

* First 1 million requests per month are free

* $0.20 per 1 million requests thereafter ($0.0000002 per request)

Lambda Pricing vs. EC2: 100 ms excecution times



*Note: Lambda includes more management than EC2, and On-Demand instances don't provide equal performance.

Source: www.CloudServiceEvaluation.com

Benefits

Infinite Scale
Package and Deploy
Time to Market
Operational Management*

Do you mean...

#NoOps?

Every time you say NoOps, a sysadmin dies.

Serverless doesn't remove Ops complexity, it increases it exponentially

Enough concepts, let's talk about tooling

Serverless Frameworks*

It's been two years since the first one and we already have like 20+ frameworks

Frameworks

- Apex - Minimal AWS Lambda function manager with support for multiple languages including Nodejs, Golang, Python, Java, Rust and Clojure.
- Chalice - Python serverless microframework from Amazon for AWS lambda
- ClaudiaJS - Deploy Node.js microservices to AWS easily.
- DEEP - Full-stack Web Framework for Cloud-Native Applications and Platforms using Microservices Architecture.
- Gordon - λ Gordon is a tool to create, wire and deploy AWS Lambdas using CloudFormation
- Gestalt Framework - Gestalt's Lambda Application SERver (LASER)" for short, is a lambda service that supports running .Net, Javascript, Java, Scala, Ruby, and Python lambdas.
- IronFunctions - The Serverless Microservices platform
- Kappa - a command line tool that (hopefully) makes it easier to deploy, update, and test functions for AWS Lambda.
- Lambada Framework - JAX-RS implementation for AWS Lambda.
- Lambdoku - Heroku-like experience when using AWS Lambda
- Python-λ - A toolkit for developing and deploying serverless Python code in AWS Lambda
- Serverless Framework - Build and maintain web, mobile and IoT applications running on AWS Lambda and API Gateway (formerly known as JAWS).
- Shep - A framework for building APIs using AWS API Gateway and Lambda
- Sparta - A framework that transforms a Go application into an AWS Lambda powered microservice.
- Turtle - library for building functional and actor-driven NodeJS apps on Lambda
- Zappa - Serverless Python WSGI with AWS Lambda + API Gateway.
- λambdify - AWS Lambda automation and integration for Python
- Squeezer Framework - Event-driven APIs & Web apps on microservices, serverless.

Frameworks

- Apex - Minimal AWS Lambda function manager with support for multiple languages including Nodejs, Golang, Python, Java, Rust and Clojure.
- Chalice - Python serverless microframework from Amazon for AWS lambda
- ClaudiaJS - Deploy Node.js microservices to AWS easily.
- DEEP - Full-stack Web Framework for Cloud-Native Applications and Platforms using Microservices Architecture.
- Gordon - λ Gordon is a tool to create, wire and deploy AWS Lambdas using CloudFormation
- Gestalt Framework - Gestalt's Lambda Application SERver (LASER)" for short, is a lambda service that supports running .Net, Javascript, Java, Scala, Ruby, and Python lambdas.
- IronFunctions - The Serverless Microservices platform
- Kappa - a command line tool that [illegible] it easier to deploy, update, and test functions for AWS Lambda.
- Lambada Framework - JAX-RS implementation for AWS Lambda
- Lambdoku - Heroku-like experience when using AWS Lambda
- Python-λ - A toolkit for developing and deploying serverless Python code in AWS Lambda
- Serverless Framework - Build and maintain web, mobile and IoT applications running on AWS Lambda and API Gateway (formerly known as JAWS).
- Shep - A framework for building APIs using AWS API Gateway and Lambda
- Sparta - A framework that transforms a Go application into an AWS Lambda powered microservice.
- Turtle - library for building functional and actor-driven NodeJS apps on Lambda
- Zappa - Serverless Python WSGI with AWS Lambda + API Gateway.
- λambdify - AWS Lambda automation and integration for Python
- Squeezer Framework - Event-driven APIs & Web apps on microservices, serverless.

WHY?

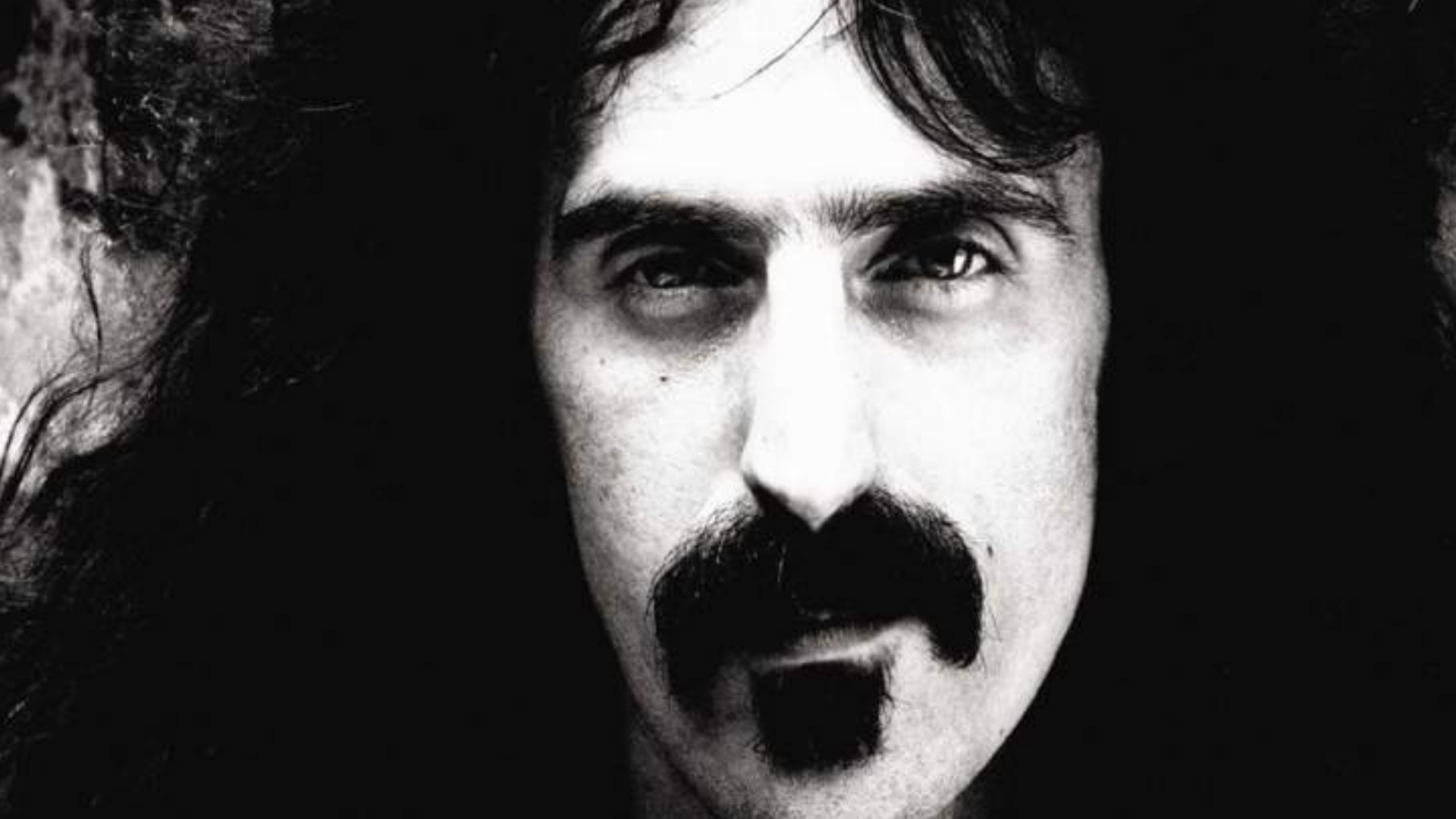
Everything is new

It's not *easy*

Lot of moving pieces

No specs whatsoever


Zappa




Zappa

FEATURES!

Python WSGI Applications
Great for micro AND macro services
AWS Lambda + AWS API Gateway
AWS Event Sources
Cron-like Events
Logs
Rollback

Environment variables from S3
Multiple stage deployments
Django Management Commands (yeah!)
Keep Warm
Free SSL
Globally Distributed Availability

```
$ pip install zappa
```

```
$ zappa init
```

zappa_settings.json

```
{
    "dev": {
        "aws_region": "us-east-1",
        "django_settings": "hello.settings",
        "profile_name": "default",
        "project_name": "hello",
        "runtime": "python3.6",
        "s3_bucket": "zappa-huyg6op0s"
    }
}
```

```
$ zappa deploy
```

```
# Deployment complete!: <some_url>
```

ARN - arn:aws:lambda:us-east-1:272997920446:function:helloworld-dev

helloworld-dev

Qualifiers ▼ Actions ▼ Test

Code | Configuration | Triggers | Tags | Monitoring

The deployment package of your Lambda function "helloworld-dev" is too large to enable inline code editing. However, you can still invoke your function right now.

Code entry type

Upload a .ZIP file ▼

Function package*

Upload

For files larger than 10 MB, consider uploading via S3.

Environment variables

You can define Environment Variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more.

Key | Value | Remove

Enable encryption helpers

For storing sensitive information, we recommend encrypting values using KMS and the console's encryption helpers.

helloworld-dev

Qualifiers ▼ | Actions ▼ | Test

Code | **Configuration** | Triggers | Tags | Monitoring

Basic information

Runtime

Python 3.6 ▼

Handler

The filename.handler-method value in your function. For example, "main.handler" would call the handler method defined in main.py.

handler.lambda_handler

Role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. Learn more about Lambda execution roles.

Choose an existing role ▼

Existing role

You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.

helloworld-dev-ZappaLambdaExecutionRole ▼

Description

Zappa Deployment

▼ Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. Learn more about how Lambda pricing works.

Memory (MB)
Your function is allocated CPU proportional to the memory configured.

512 MB

Timeout
0 min 30 sec

DLQ Resource Info
Choose the AWS service to send event payload to after exceeding maximum retries.

Select resource

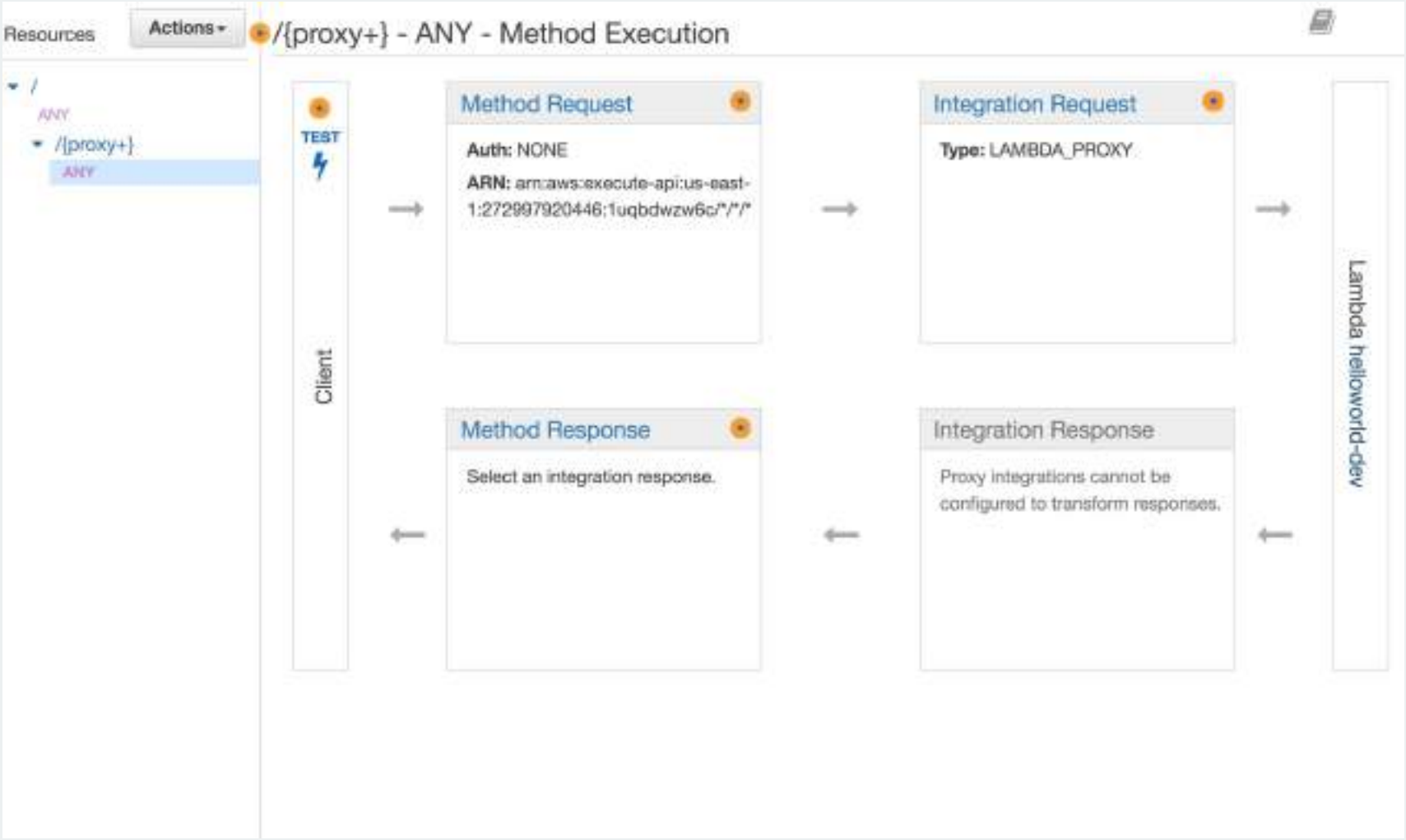
VPC Info
Select a VPC that your function will access.

No VPC

Enable active tracing Info

KMS key Info
Select a KMS key to encrypt the environment variables at rest, or simply let Lambda manage the encryption.

(default) aws/lambda | Enter value


Resources
Actions
/{proxy+} - ANY - Method Execution
/
ANY
/{proxy+}
ANY
TEST
Client
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:272997920446:1uqbdwzw6c/*/*/*
Integration Request
Type: LAMBDA_PROXY
Lambda helloworld-dev
Method Response
Select an integration response.
Integration Response
Proxy integrations cannot be configured to transform responses.

```
$ zappa update dev
```

```
$ zappa rollback dev -n 2
```

```
$ zappa tail dev
```

```
$ zappa invoke dev "print('PyConZA')" --raw
```

```
$ zappa manage dev migrate
```

```
# zappa_settings.json

{
    "dev": {
        ...
    },
    "staging": {
        ...
    },
    "production": {
        ...
    }
}
```

```
$ zappa deploy staging
```

And much more!
https://github.com/Miserlou/Zappa

Drawbacks

Vendor control/lock-in*
No server optimizations
No in-server state

Opportunities

Tooling
Open source projects
Learning and teaching
LOTS OF improvements

And the question that everyone is wondering....

Is it production ready?

YES!

but *try* if first

Serverless Weekly

http://eepurl.com/cUU8sD

In the land of Serverless, who uses Zappa is king!

@jonatasbaldin

Obrigado!