Ukraine

XP DAYS

Origins of Serverless

Andrii Soldatenko

10-11 November 2017

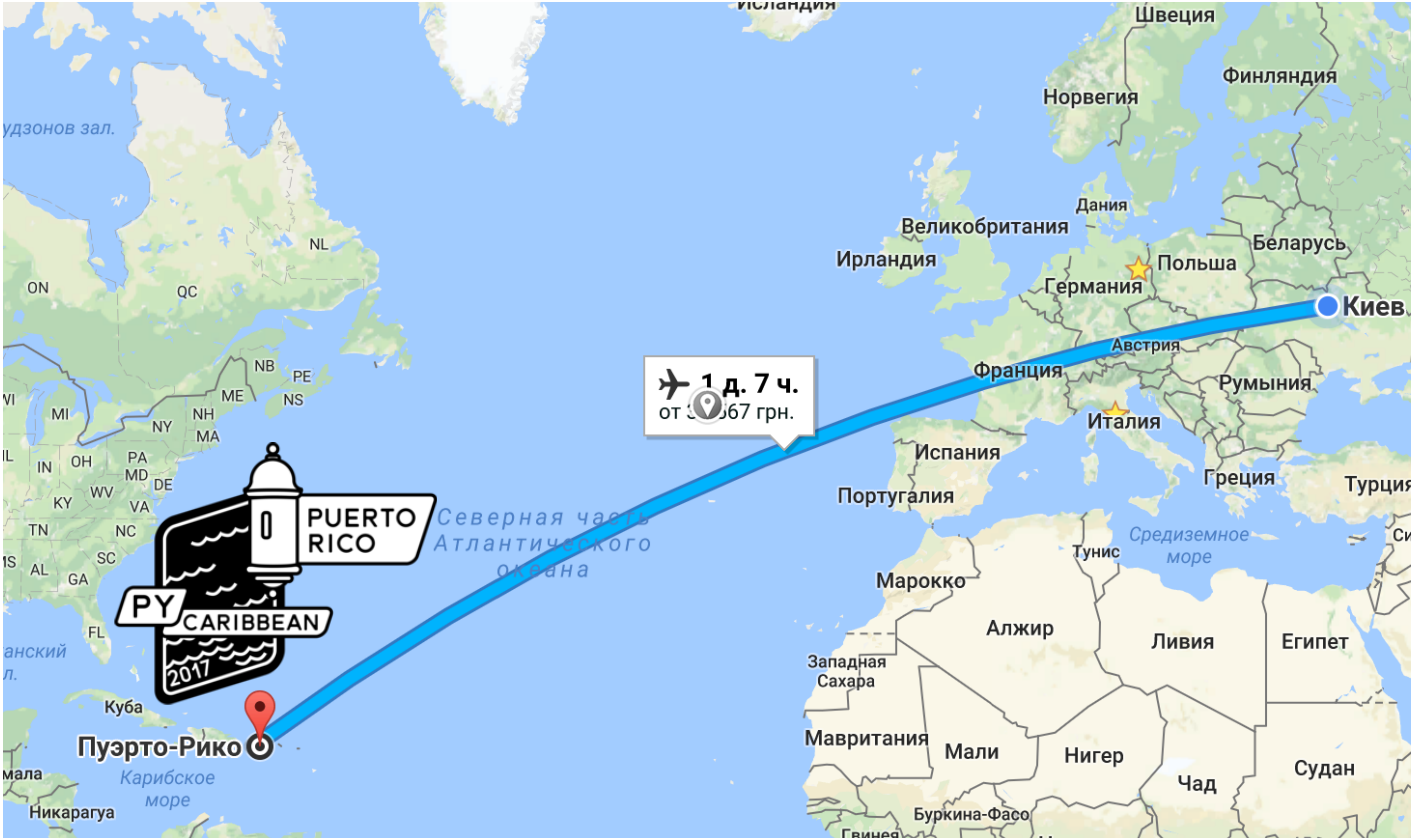
@a_soldatenko

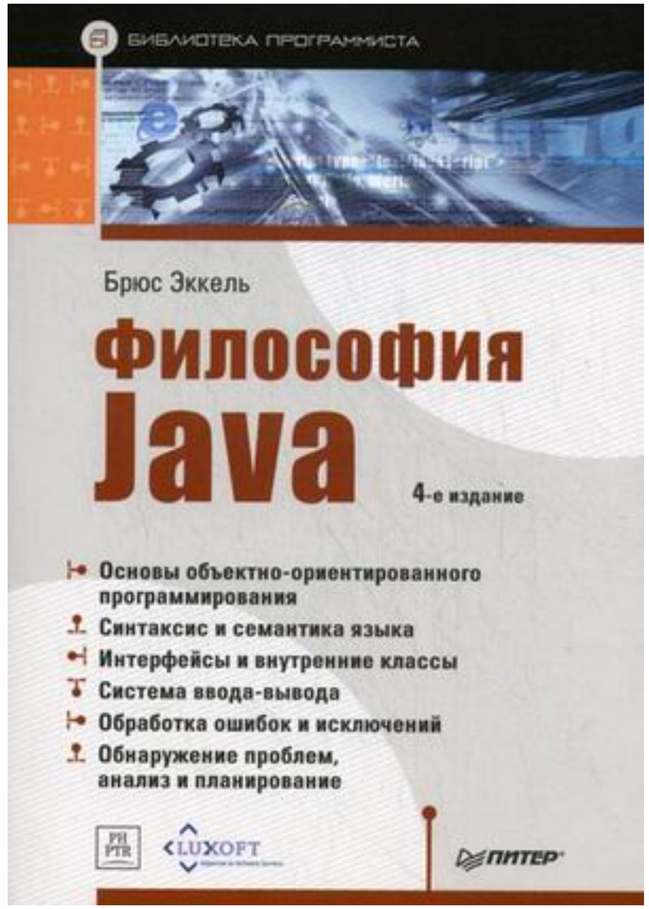
grep andrii /etc/passwd

- Gopher by night and Python dev by day
- Speaker at many conferences and open source contributor github.com/andriisoldatenko
- blogger at https://asoldatenko.com

https://asoldatenko.com/

@a_soldatenko


1 д. 7 ч.
PUERTO RICO
PY CARIBBEAN
2017
Швеция
Финляндия
Норвегия
Дания
Великобритания
Ирландия
Беларусь
Польша
Германия
Киев
Австрия
Франция
Румыния
Италия
Испания
Греция
Португалия
Северная часть Атлантического океана
Средиземное море
Тунис
Марокко
Алжир
Ливия
Египет
Западная Сахара
Мавритания
Мали
Нигер
Чад
Судан
Буркина-Фасо
Куба
Пуэрто-Рико
Карибское море
Никарагуа
NL
ON
QC
NB
PE
NS
ME
NH
MA
NY
MI
PA
MD
DE
OH
IN
WV
KY
VA
TN
NC
SC
AL
GA
FL

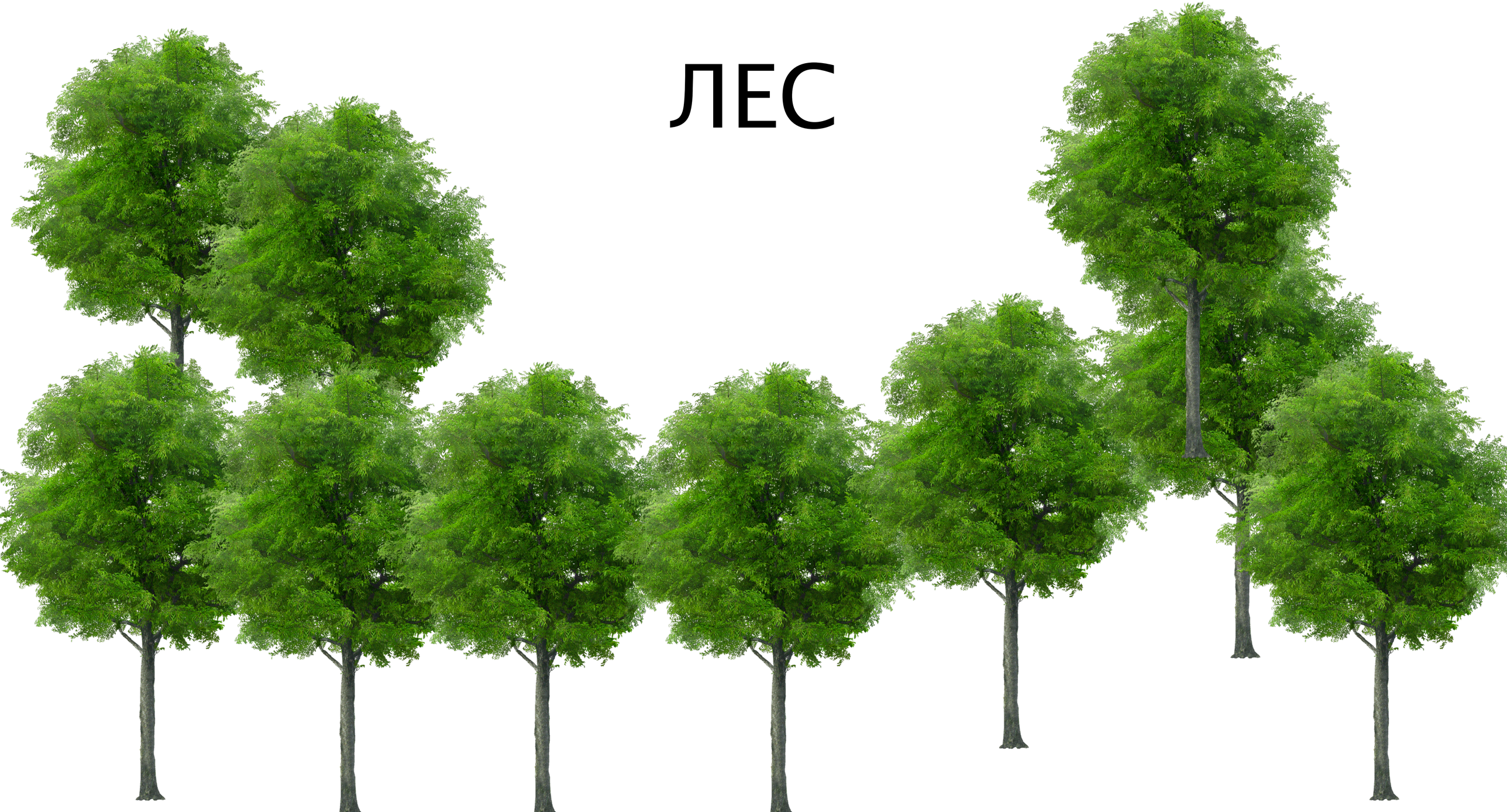


https://asoldatenko.com/

@a_soldatenko

Server

ЛЕС



SERVERLESS

…in the next few years we're going to see the first billion-dollar startup with a single employee, the founder, and that engineer will be using **serverless** technology.

James Governor
Analyst & Co-founder at RedMonk

Origins of “Serverless”

Origins of serverless

http://readwrite.com/2012/10/15/why-the-future-of-software-and-apps-is-serverless/

Python Geek blog (Graphs)-[:ARE]->(Everywhere)

Origins of term “Serverless”

June 21, 2017

Serverless is new trend in software development. It’s confusing many developers around the world, let’s try to find origins of term “Serverless”. First time Ken Fromm in 2012 use this term in his article. (Ken added a bit of clarification in the comments to give credit where credit is due.)

https://asoldatenko.com/building-serverless-applications-with-python-3.html

Origins of serverless

Andrii Soldatenko @a_soldatenko 30 Jul

Small note about origins of term “Serverless” in my blog asoldatenko.com/building-serve… #Serverless #python3

Ken Fromm
@frommww

1. First time I saw the term was in a blog post by Diego Basch on using Indextank and Iron to create a serverless search engine.

7:58 PM - Jul 30, 2017

1

Origins of serverless

Diego Basch • 3rd

Holder of Self-Referential Title

Cointipping.com • Carnegie Mellon University

San Francisco Bay Area • 500+

Origins of serverless

Andrii Soldatenko @a_soldatenko 30 Jul

Small note about origins of term “Serverless” in my blog
asoldatenko.com/building-serve… #Serverless #python3

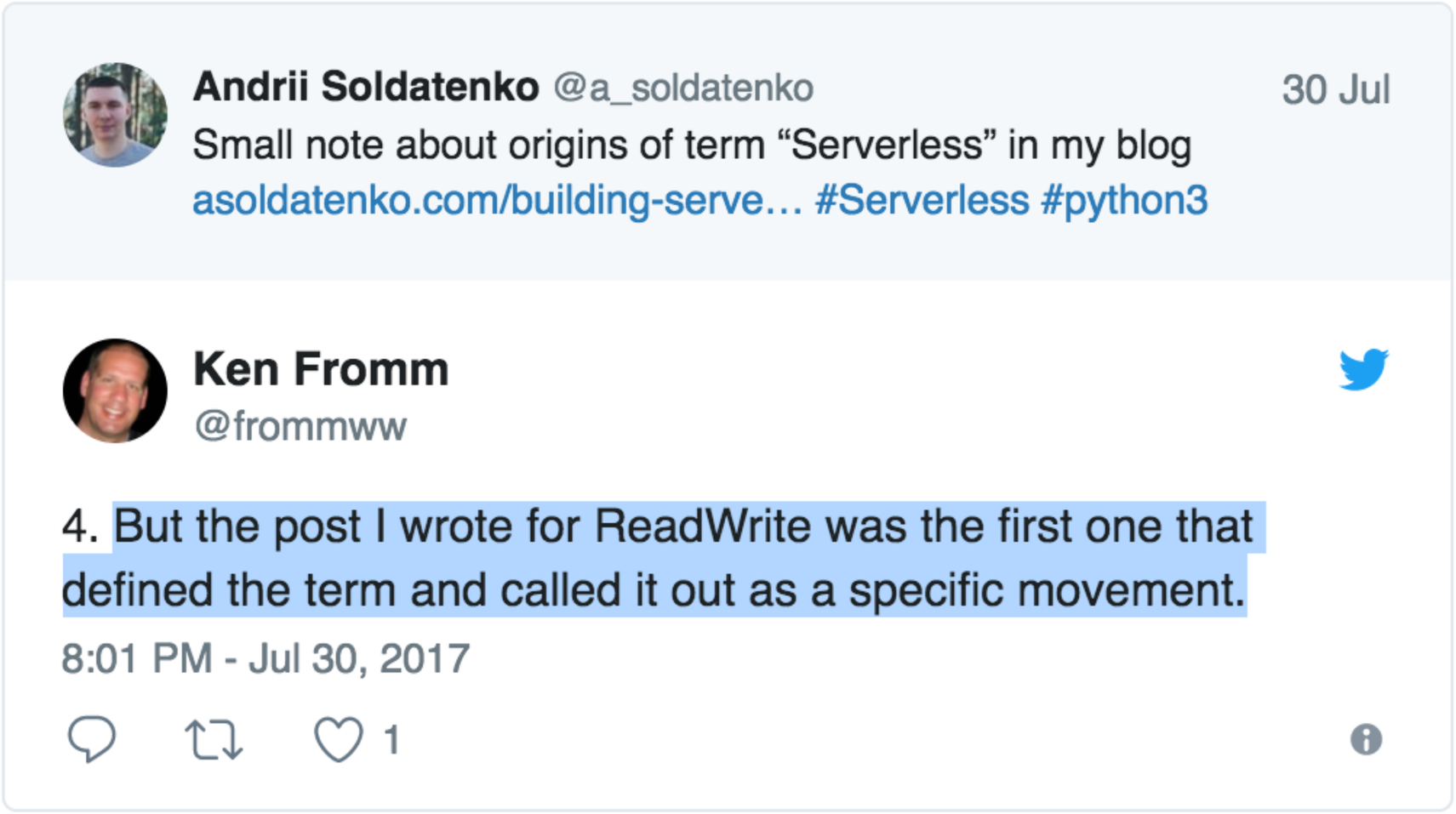
Ken Fromm
@frommww

2. We (Iron) reposted it and used the term a few more times in blogs prior to my October 2012 article.

7:59 PM - Jul 30, 2017

1 1

Origins of serverless



Andrii Soldatenko @a_soldatenko 30 Jul

Small note about origins of term “Serverless” in my blog asoldatenko.com/building-serve… #Serverless #python3

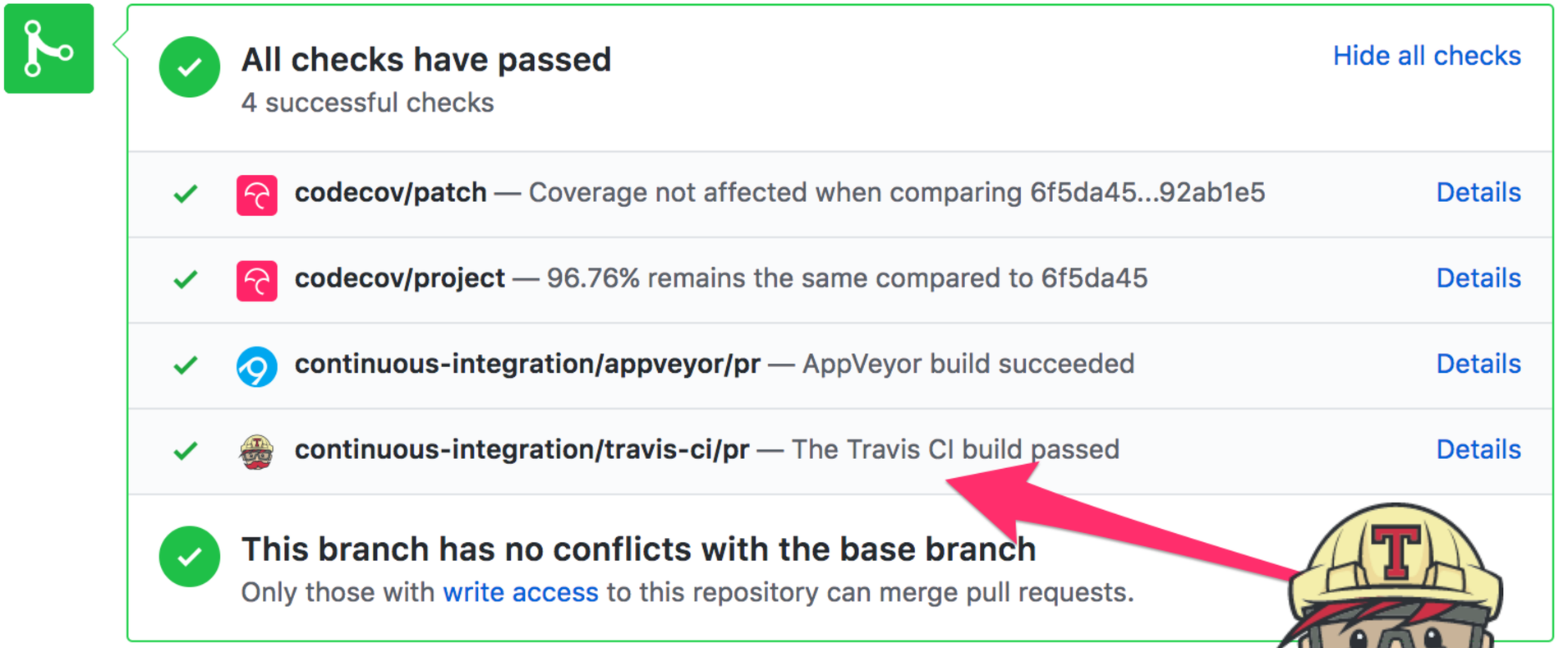
Ken Fromm
@frommww

4. But the post I wrote for ReadWrite was the first one that defined the term and called it out as a specific movement.

8:01 PM - Jul 30, 2017

1

Travis CI



https://asoldatenko.com/

@a_soldatenko

Later in 2014…

Amazon announced AWS Lambda

https://techcrunch.com/2015/11/24/aws-lamda-makes-serverless-applications-a-reality/

AWS Lambda
is one of the most popular
implementations of FaaS
(Functions as a service / FaaS)

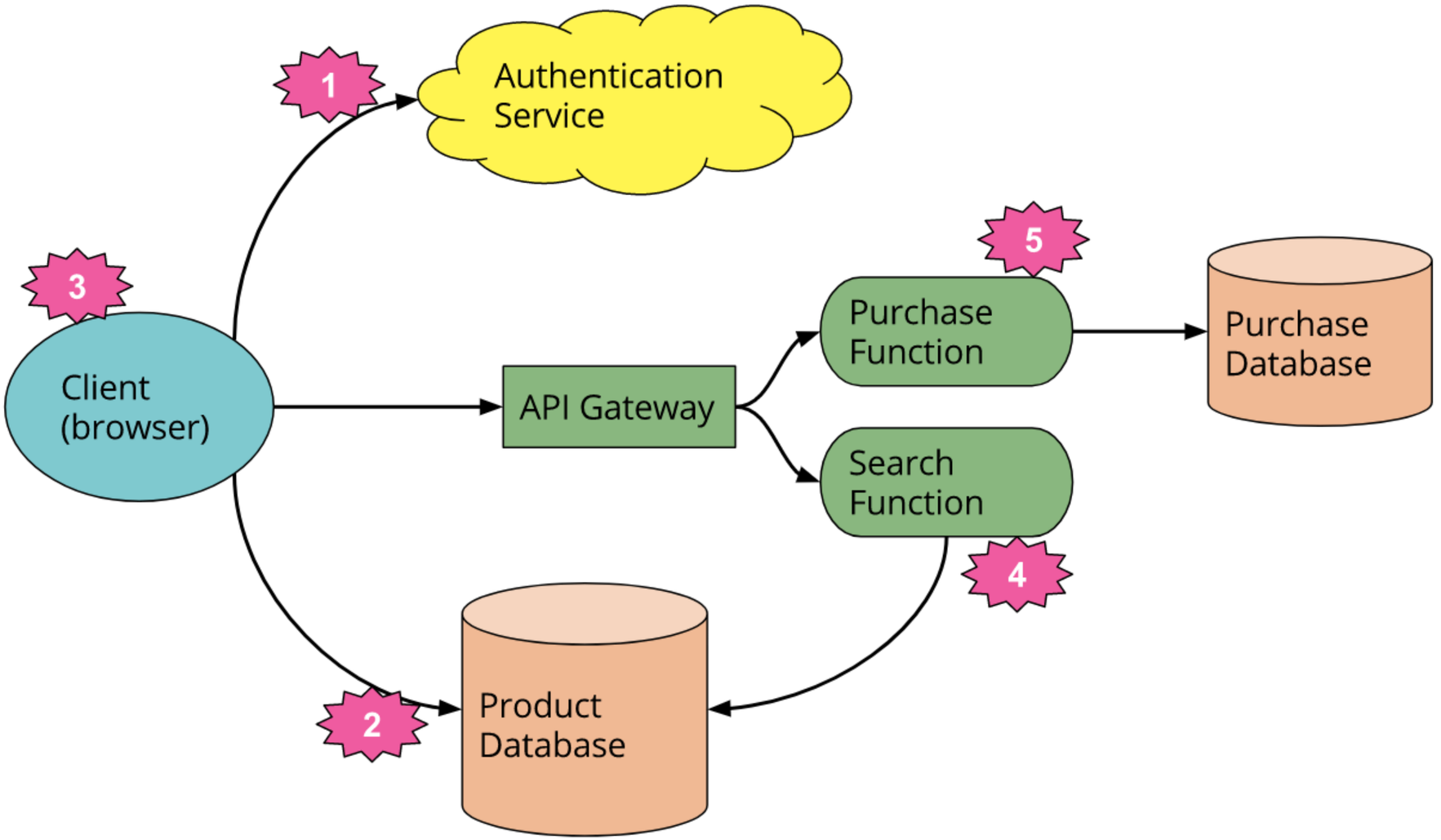
https://martinfowler.com/articles/serverless.html

How Lambda (λ) works

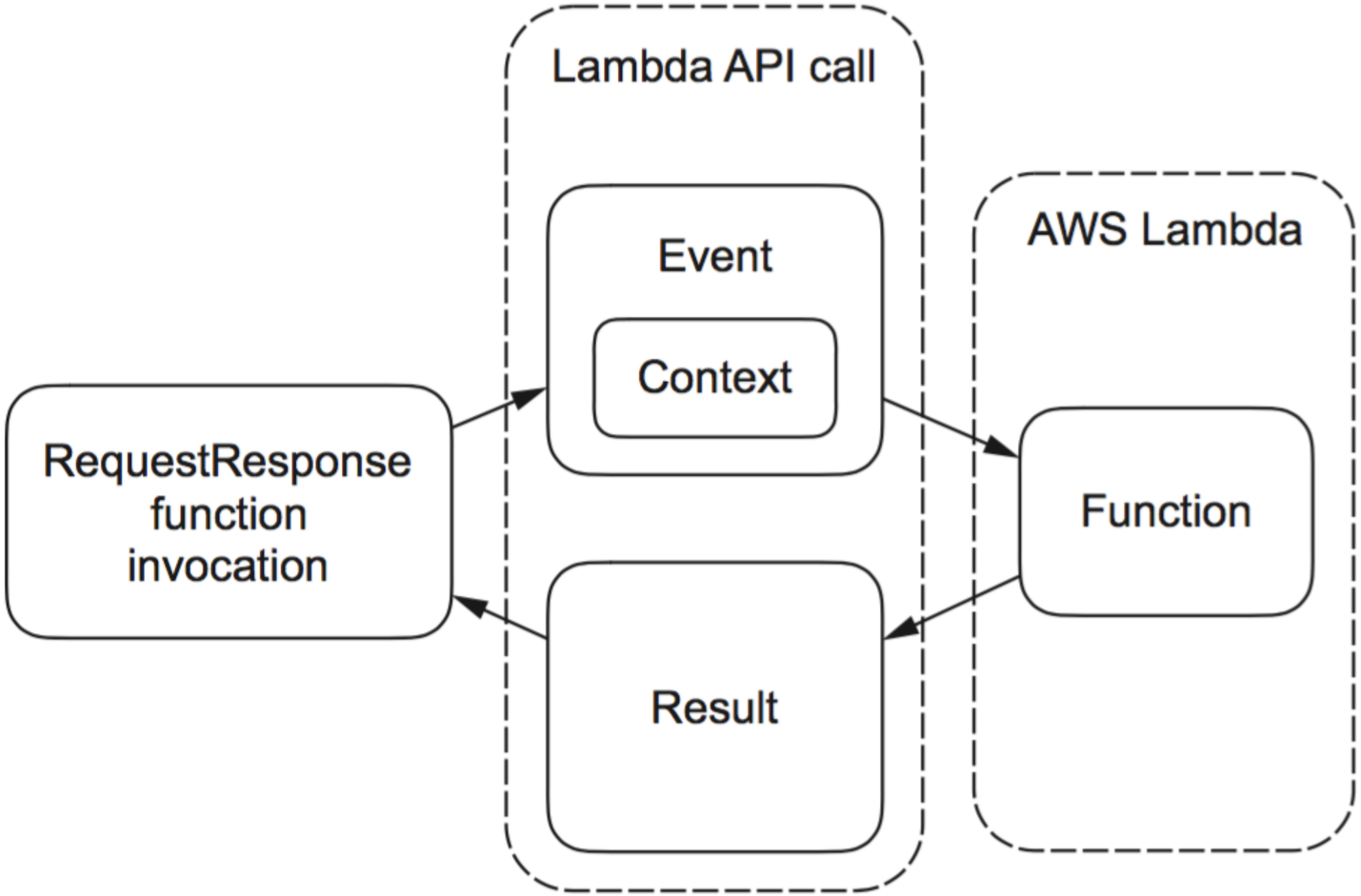
@a_soldatenko

Traditionally the architecture

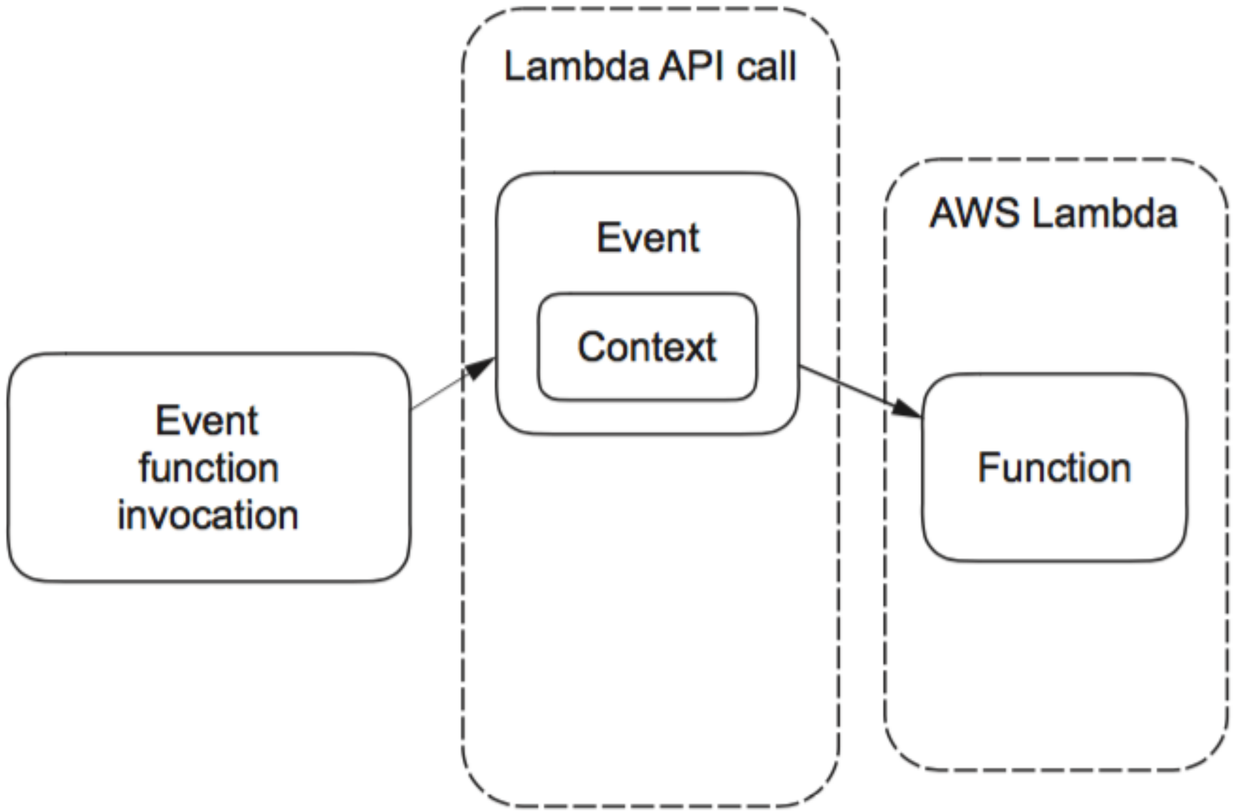
Serverless architecture



How Lambda works

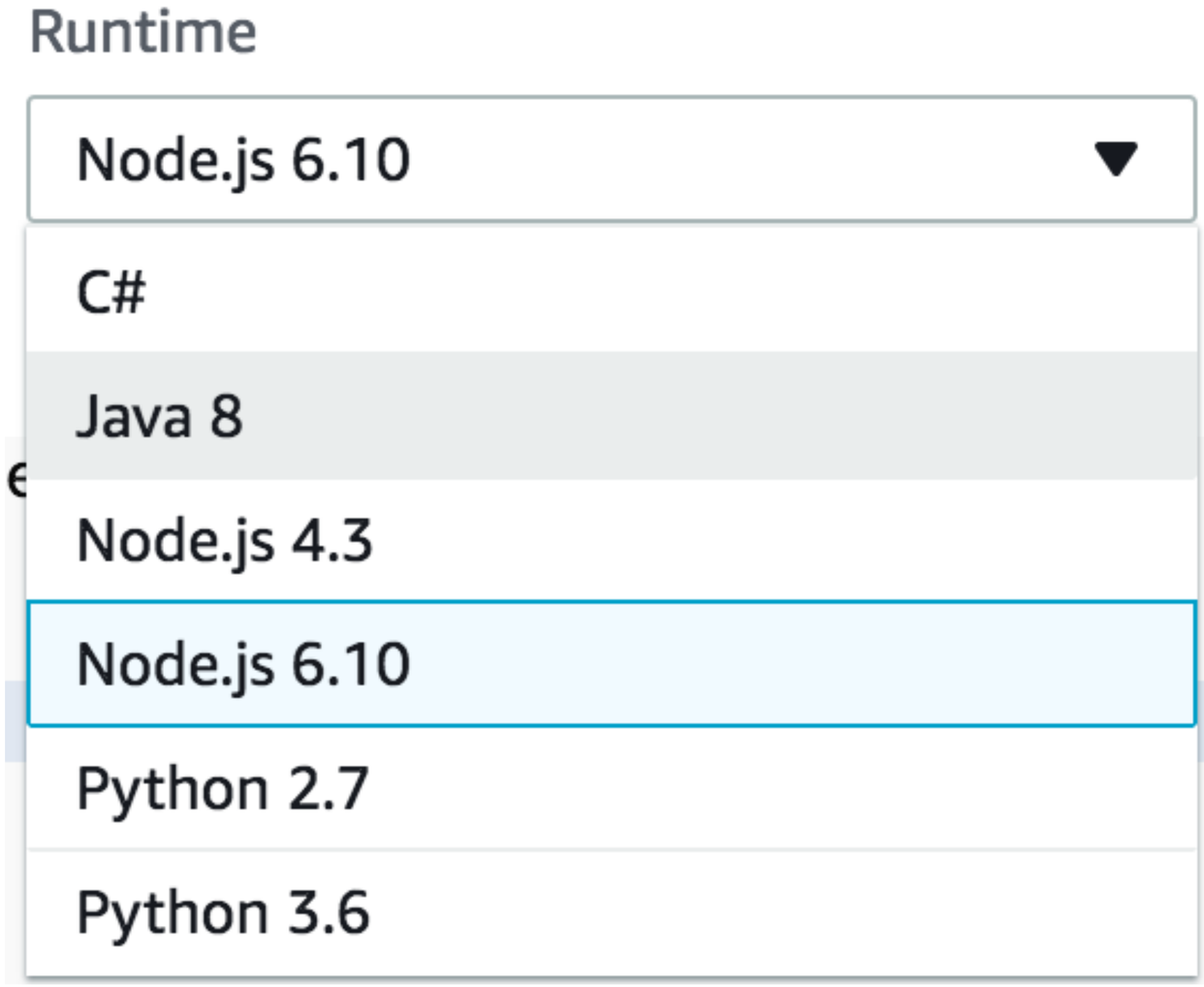


How Lambda works



@a_soldatenko

Runtimes


Runtime
Node.js 6.10
C#
Java 8
Node.js 4.3
Node.js 6.10
Python 2.7
Python 3.6

How Lambda works

- function name;
- memory size;
- timeout;
- role;

Your first λ function

```
def lambda_handler(event, context):
    message = 'Hello {}!'.format(event['name'])
    return {'message': message}
```

Deploy λ function

```
#!/usr/bin/env bash
python -m zipfile -c hello_python.zip hello_python.py

aws lambda create-function \
    --region us-west-2 \
    --function-name HelloPython \
    --zip-file fileb://hello_python.zip \
    --role arn:aws:iam::278117350010:role/lambda-s3-execution-role  \
    --handler hello_python.my_handler \
    --runtime python2.7 \
    --timeout 15 \
    --memory-size 512
```

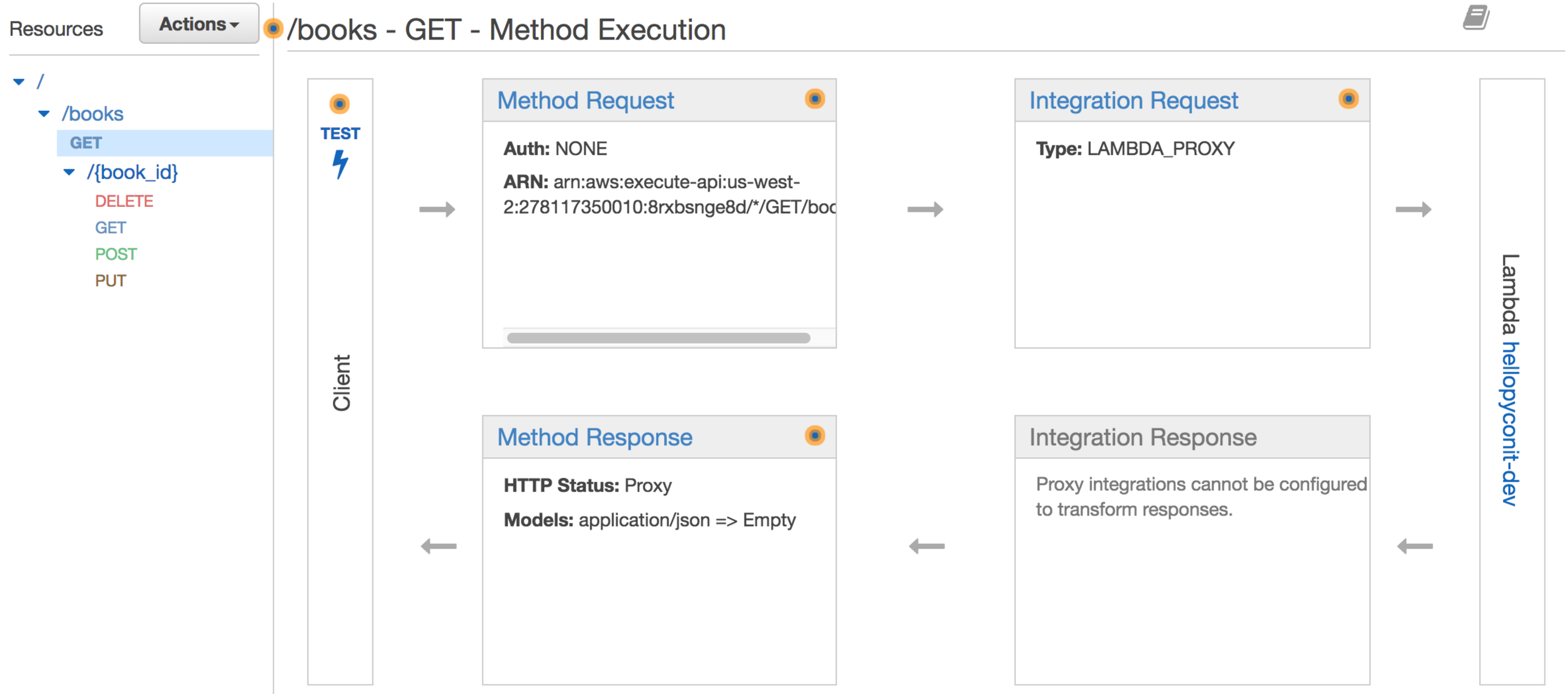
Invoke λ function

```
aws lambda invoke \
        --function-name HelloPython \
        --payload '{"name":"XP Days!"}' output.txt

cat output.txt
{"message": "Hello XP Days!"}%
```

Amazon API Gateway

Resource	HTTP verb	AWS Lambda
/books	GET	get_books
/book	POST	create_book
/books/{ID}	PUT	chage_book
/books/{ID}	DELETE	delete_book


Resources
Actions
/books - GET - Method Execution
/
/books
GET
/{book_id}
DELETE
GET
POST
PUT
TEST
Client
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-west-2:278117350010:8rxbsnge8d/*/GET/boo
Integration Request
Type: LAMBDA_PROXY
Lambda hellopyconit-dev
Method Response
HTTP Status: Proxy
Models: application/json => Empty
Integration Response
Proxy integrations cannot be configured to transform responses.

Chalice python micro framework

https://github.com/awslabs/chalice

@a_soldatenko

Chalice python micro framework

```
$ cat app.py
from chalice import Chalice

app = Chalice(app_name='helloxpdays')


@app.route('/books', methods=['GET'])
def get_books():
    return {'hello': 'from python library'}
...
...
...
```

https://github.com/awslabs/chalice

Chalice python micro framework

```
...
...
...
@app.route('/books/{book_id}', methods=['POST', 'PUT', 'DELETE'])
def process_book(book_id):
    request = app.current_request
    if request.method == 'PUT':
        return {'msg': 'Book {} changed'.format(book_id)}
    elif request.method == 'DELETE':
        return {'msg': 'Book {} deleted'.format(book_id)}
    elif request.method == 'POST':
        return {'msg': 'Book {} created'.format(book_id)}
```

@a_soldatenko

Chalice deploy

```
chalice deploy
Updating IAM policy.
Updating lambda function...
...
...
API Gateway rest API already found.
Deploying to: dev
https://8rxbsnge8d.execute-api.us-
west-2.amazonaws.com/dev/
```

@a_soldatenko

Try our books API

```
curl -X GET https://8rxbsnge8d.execute-api.us-
west-2.amazonaws.com/dev/books
{"hello": "from python library"}%

curl -X GET https://8rxbsnge8d.execute-api.us-
west-2.amazonaws.com/dev/books/15
{"message": "Book 15"}%

curl -X DELETE https://8rxbsnge8d.execute-api.us-
west-2.amazonaws.com/dev/books/15
{"message": "Book 15 has been deleted"}%
```

@a_soldatenko

Chalice under the hood

chalice delete #40

Open **pauldeden** opened this issue on Jul 12, 2016 · 5 comments

pauldeden commented on Jul 12, 2016 Contributor +😀

I love the simplicity of the chalice CLI, but feel there is one more command to add to complete the feature set. That is a `chalice delete` (or named something similar) command.

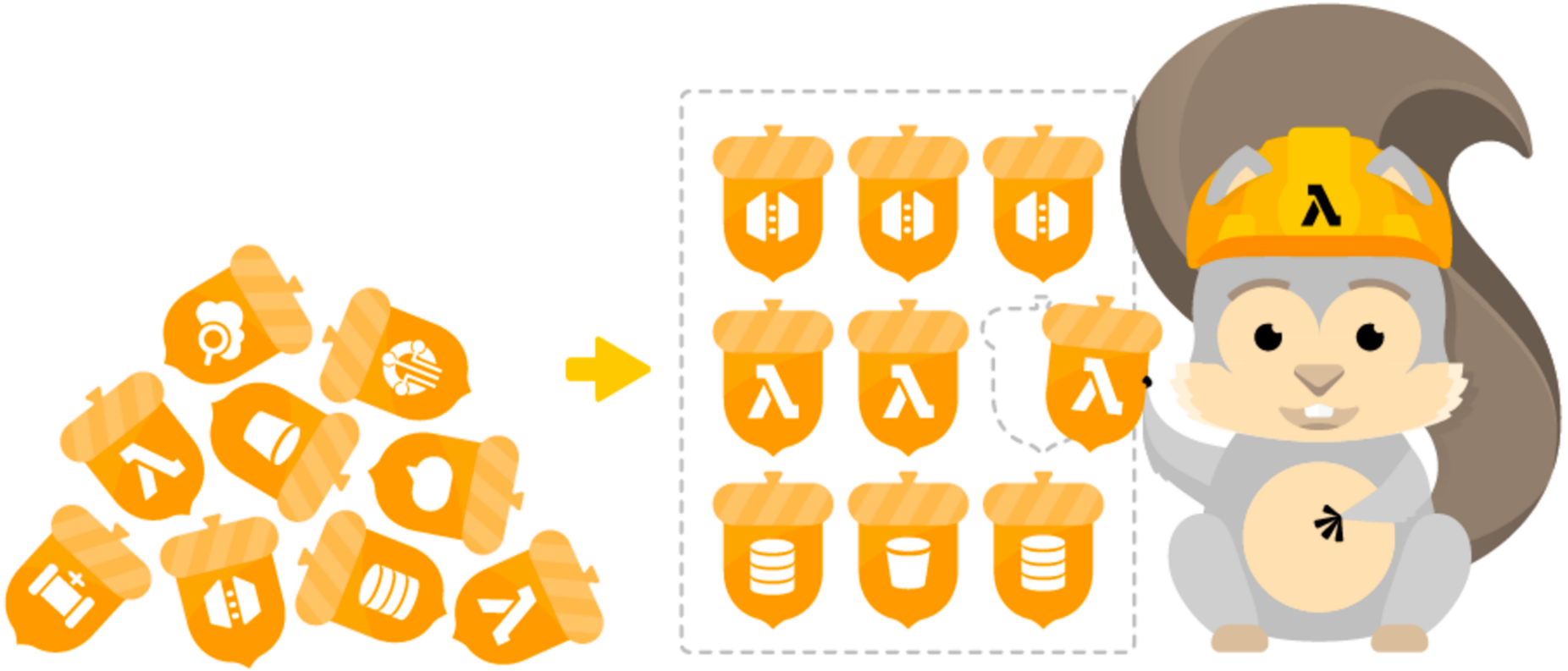
@a_soldatenko

AWS Serverless Application Model

MEET SAM.

@a_soldatenko

AWS SAM



USE SAM TO BUILD TEMPLATES THAT DEFINE
YOUR SERVERLESS APPLICATIONS.

@a_soldatenko

AWS lambda Limits

AWS Lambda Resource Limits

Resource	Default Limit
Ephemeral disk capacity ("/tmp" space)	512 MB
Number of file descriptors	1,024
Number of processes and threads (combined total)	1,024
Maximum execution duration per request	300 seconds
Invoke request body payload size (RequestResponse)	6 MB
Invoke request body payload size (Event)	128 K
Invoke response body payload size (RequestResponse)	6 MB

@a_soldatenko

What about conferences?

http://serverlessconf.io/

AWS Lambda costs

- The first 400,000 seconds of execution time with 1 GB of memory

One missing return; ended up costing me $206.

https://sourcebox.be/blog/2017/08/07/serverless-a-lesson-learned-the-hard-way/

And again what do you mean “serveless”?

@a_soldatenko

Serverless - это сервер который живет **40** миллисекунд

What if you want to run your Django or any WSGI app?

@a_soldatenko

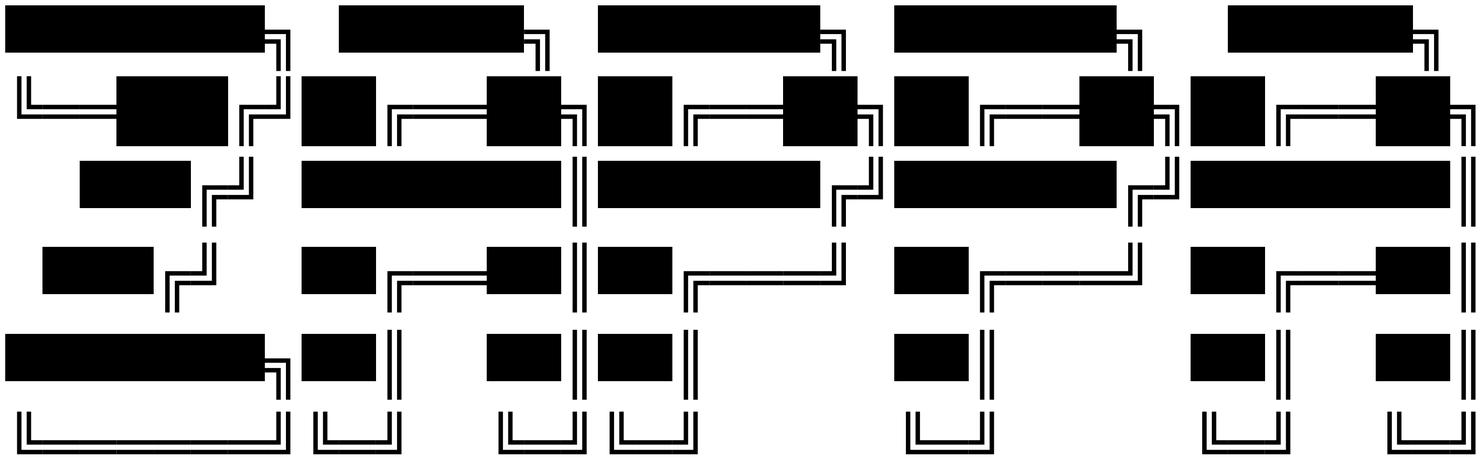
@a_soldatenko

Top 10 Python libraries of 2016

1. Zappa

Since the release of **AWS Lambda** (and **others** that **have followed**), all the rage has been about **serverless architectures**. These allow microservices to be deployed in the cloud, in a fully managed environment where one doesn't have to care about managing any server, but is assigned stateless, ephemeral *computing containers* that are fully managed by a provider. With this paradigm, events (such as a traffic spike) can trigger the execution of more of these *containers* and therefore give the possibility to handle "infinite" horizontal scaling.

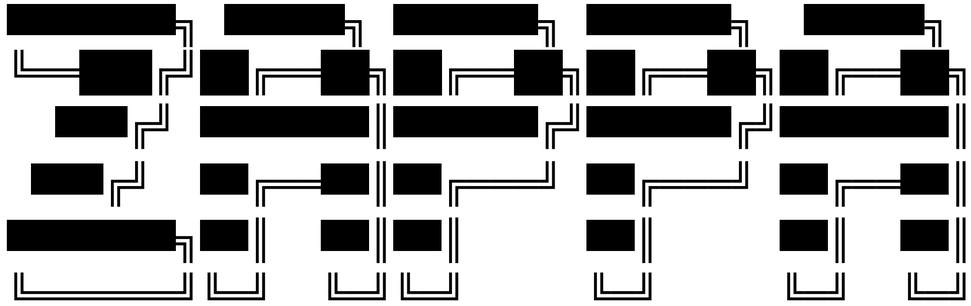
```
➜ pip install zappa
➜ zappa init
```



```
Welcome to Zappa!
...
➜ zappa deploy
```

https://github.com/Miserlou/Zappa

@a_soldatenko



- deploy;
- tailing logs;
- run django manage.py
- …

https://github.com/Miserlou/Zappa

@a_soldatenko

AWS lambda and Python 3

```
import os

def lambda_handler(event, context):
    txt = open('/etc/issue')
    print txt.read()
    return {'message': txt.read()}
```

Amazon Linux AMI release 2016.03
Kernel \r on an \m

@a_soldatenko

AWS lambda and Python 3

```
import subprocess

def lambda_handler(event, context):
    args = ('uname', '-a')
    popen = subprocess.Popen(args,
stdout=subprocess.PIPE)
    popen.wait()
    output = popen.stdout.read()
    print(output)
```

Linux ip-10-11-15-179 4.4.51-40.60.amzn1.x86_64 #1 SMP Wed Mar 29 19:17:24 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux

@a_soldatenko

AWS lambda and Python 3

```
import subprocess

def lambda_handler(event, context):
    args = ('which', 'python3')
    popen = subprocess.Popen(args,
stdout=subprocess.PIPE)
    popen.wait()
    output = popen.stdout.read()
    print(output)
```

YAHOOO!!:

python3: /usr/bin/python3 /usr/bin/python3.4m /usr/bin/python3.4 /usr/lib/python3.4 /usr/lib64/python3.4 /usr/local/lib/python3.4 /usr/include/python3.4m /usr/share/man/man1/python3.1.gz

@a_soldatenko

Run python 3 from python 2

@a_soldatenko

Prepare Python 3 for AWS Lambda

```
virtualenv venv -p `which python3.4`

pip install requests

zip -r lambda_python3.zip venv

python3_program.py lambda.py
```

@a_soldatenko

Run python 3 from python 2

```
cat lambda.py
import subprocess

def lambda_handler(event, context):
    args = ('venv/bin/python3.4', 'python3_program.py')
    popen = subprocess.Popen(args,
stdout=subprocess.PIPE)
    popen.wait()
    output = popen.stdout.read()
    print(output)
```

@a_soldatenko

Run python 3 from python 2

START RequestId: 44c89efb-1bd2-11e7-bf8c-83d444ed46f1 Version: $LATEST

Python 3.4.0

END RequestId: 44c89efb-1bd2-11e7-bf8c-83d444ed46f1

REPORT RequestId: 44c89efb-1bd2-11e7-bf8c-83d444ed46f1

@a_soldatenko

Thanks Lyndon Swan for ideas about hacking python 3

@a_soldatenko

Future of serverless

The biggest problem with Serverless FaaS right now is tooling. Deployment / application bundling, configuration, monitoring / logging, and debugging all need serious work.

MEET SAM.

https://github.com/awslabs/serverless-application-model/blob/master/HOWTO.md

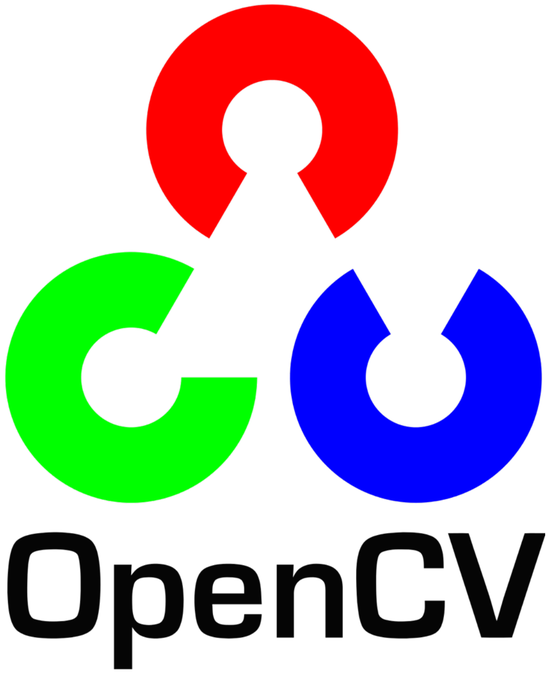
@a_soldatenko

From Apr 18, 2017 AWS Lambda Supports Python 3.6

@a_soldatenko

Face detection Example:

Using binaries with your function



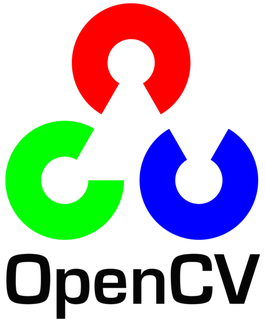
Prepare wheels

```
yum update -y
yum install -y git cmake gcc-c++ gcc python-devel chrpath
mkdir -p lambda-package/cv2 build/numpy
...
pip install opencv-python -t .
...
# Copy template function and zip package
cp template.py lambda-package/lambda_function.py
cd lambda-package
zip -r ../lambda-package.zip *
```

lambda function

```
import cv2

def lambda_handler(event, context):
  print(“OpenCV installed version:", cv2.__ver
  return "It works!"
```



START RequestId: 19e6a8f9-4eea-11e7-a662-299188c47179 Version: $LATEST

OpenCV installed version: 3.2.0

END RequestId: 19e6a8f9-4eea-11e7-a662-299188c47179

REPORT RequestId: 19e6a8f9-4eea-11e7-a662-299188c47179 Duration: 0.46 ms

Billed Duration: 100 ms Memory Size: 512 MB Max Memory Used: 35 MB


This is the original image.


Here faces are marked with white rectangles.

Debug lambda locally

```
import importlib; sys

# Import your lambda python module
mod = importlib.import_module(sys.argv[1])
function_handler = sys.argv[2]

lambda_function = getattr(mod, function_handler )

event = {'name': 'Andrii'}
context = {'conference_name': ‘XP Days!’}

try:
    data = function_handler(event, context)
    print(data)
except Exception as error:
    print(error)
```

https://goo.gl/2VNPyA

How to run lambda locally

```
$ python local_lambda.py lambda_function \
lambda_handler

{'message': 'Hello Andrii!'}
```

Conclusion

Reduce operational cost and complexity

Patrick Brandt
Solutions Architect at The Coca-Cola Company

What about websockets?

A: 2 years later and nothing yet on this?

https://forums.aws.amazon.com/thread.jspa?threadID=205761

AWS IoT Now Supports WebSockets

http://stesie.github.io/2016/04/aws-iot-pubsub

Serverless are the new NoSQL 🎉💩

- everybody claims it's the next big thing

- a few people have actually worked with the technology

- and in the end it's just a crippled (one-way) database

Thank You

https://asoldatenko.com

 @a_soldatenko

Questions

@a_soldatenko