

# Building Serverless applications with Python

Andrii Soldatenko

13 May 2017

Ukraine, Kyiv.py #20

 @a\_soldatenko

# Andrii Soldatenko

- Senior Python Developer at



- CTO at

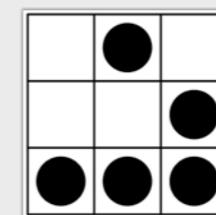


- Co-organizer PyCon Belarus 2017



- Speaker at many PyCons and open source contributor

- blogger at <https://asoldatenko.com>



...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



 @a\_soldatenko

# Origins of “Serverless”



# Origins of **serverless**

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

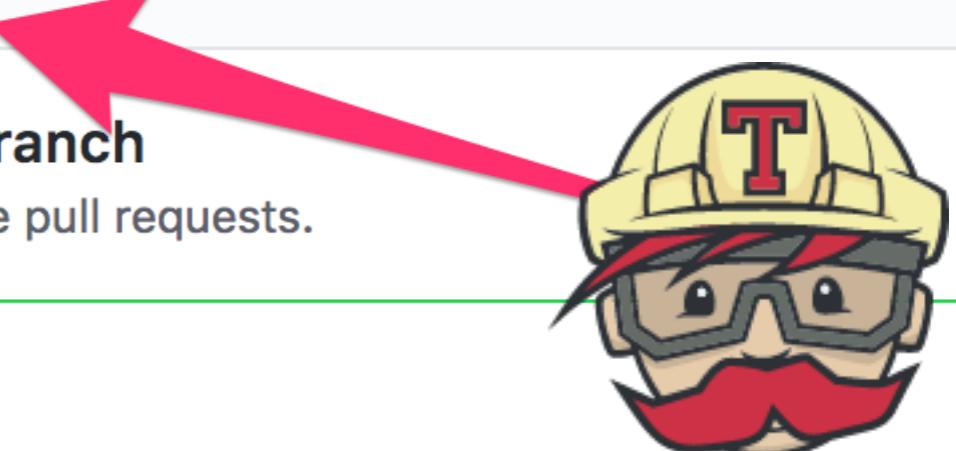
# Travis CI

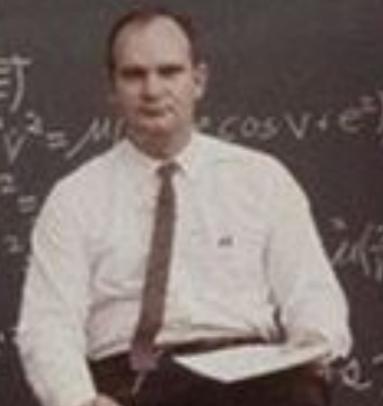
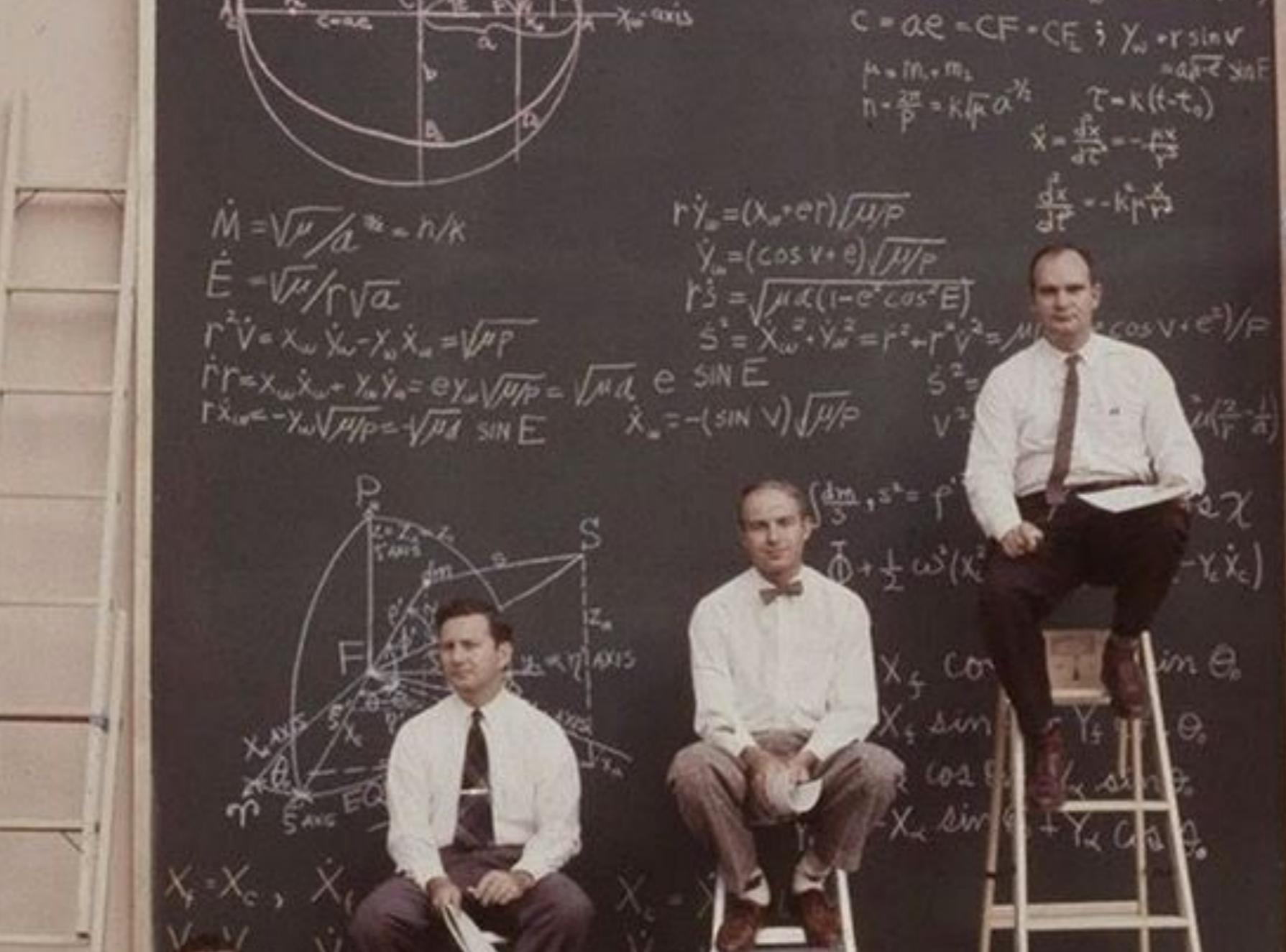
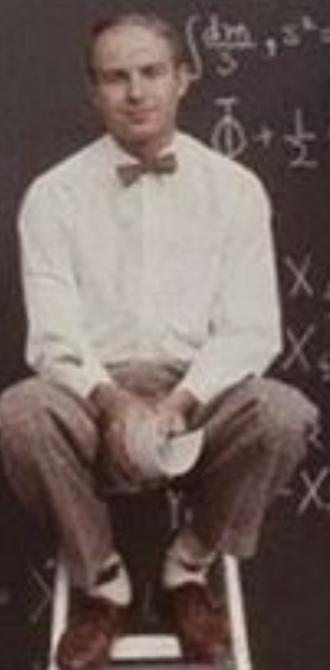
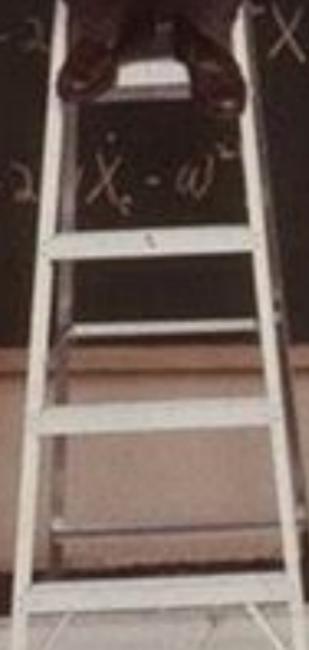
 All checks have passed [Hide all checks](#)

4 successful checks

-  **codecov/patch** — Coverage not affected when comparing 6f5da45...92ab1e5 [Details](#)
-  **codecov/project** — 96.76% remains the same compared to 6f5da45 [Details](#)
-  **continuous-integration/appveyor/pr** — AppVeyor build succeeded [Details](#)
-  **continuous-integration/travis-ci/pr** — The Travis CI build passed [Details](#)

 **This branch has no conflicts with the base branch**  
Only those with [write access](#) to this repository can merge pull requests.





Later in 2014...

Amazon introduce  
AWS Lambda

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

# What about conferences?



A CLOUD GURU

# SERVERLESSCONF

AUSTIN

LONDON

TOKYO

NEW YORK

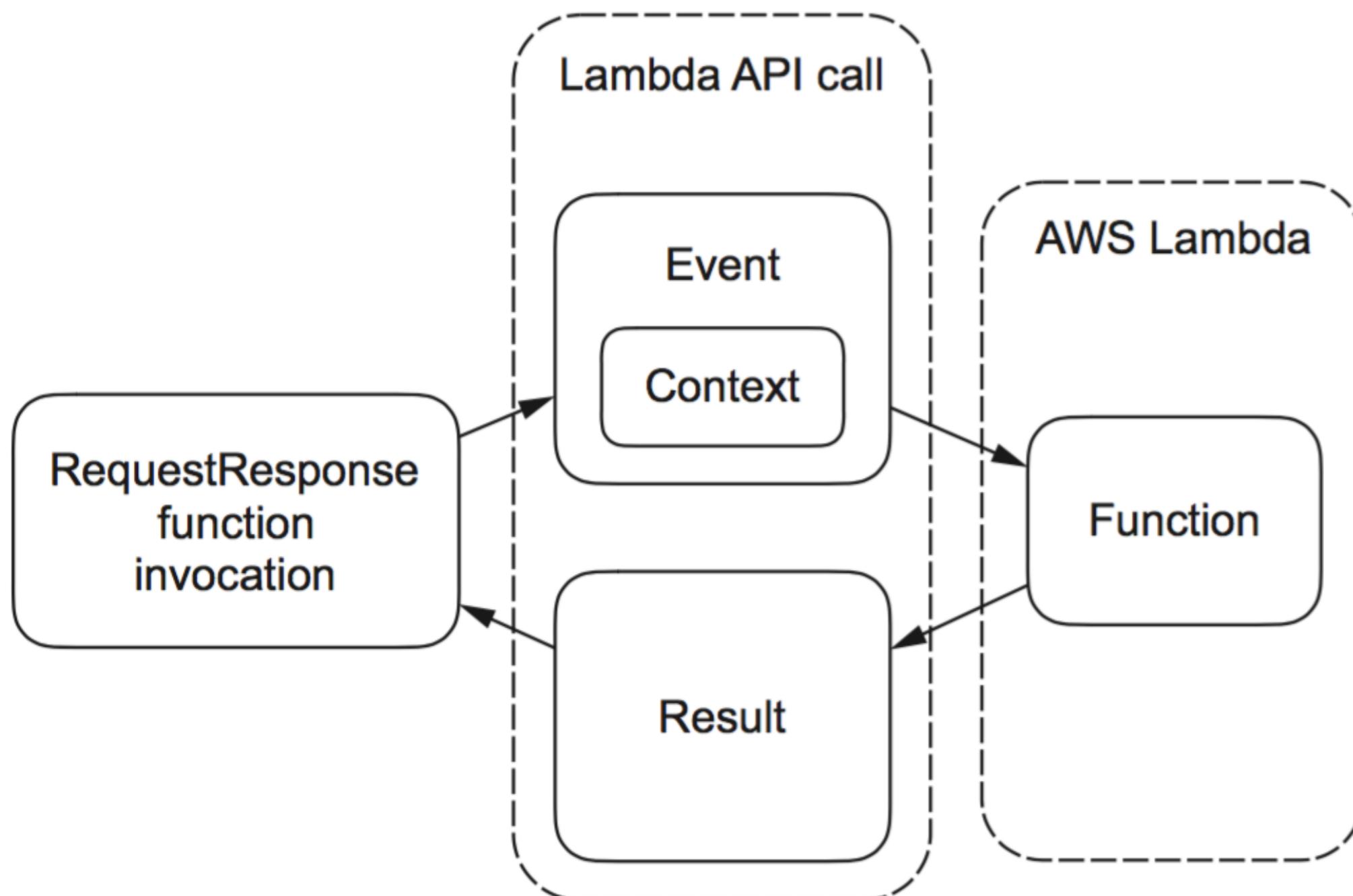
# How Lambda works



$\lambda$ 

@a\_soldatenko

# How Lambda works



# How Lambda works

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

# Your first $\lambda$ 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 file://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": "PyCon
Italy"}' output.txt

cat output.txt
{ "message": "Hello PyCon Italy!" }%
```

# Amazon API Gateway

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

# Chalice python micro framework

<https://github.com/awslabs/chalice>

# Chalice python micro framework

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

app = Chalice(app_name='hellopyconit')

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

...
...
...
```

# 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) }
```

# Chalice deploy

**chalice deploy**

Updating IAM policy.

Updating **lambda** function...

Regen deployment package...

Sending changes to **lambda**.

API Gateway rest API already found.

Deploying to: dev

<https://8rxbsnge8d.execute-api.us-west-2.amazonaws.com/dev/>

# Try our books API

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

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

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

# Chalice under the hood

- botocore;
- typing;

# 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



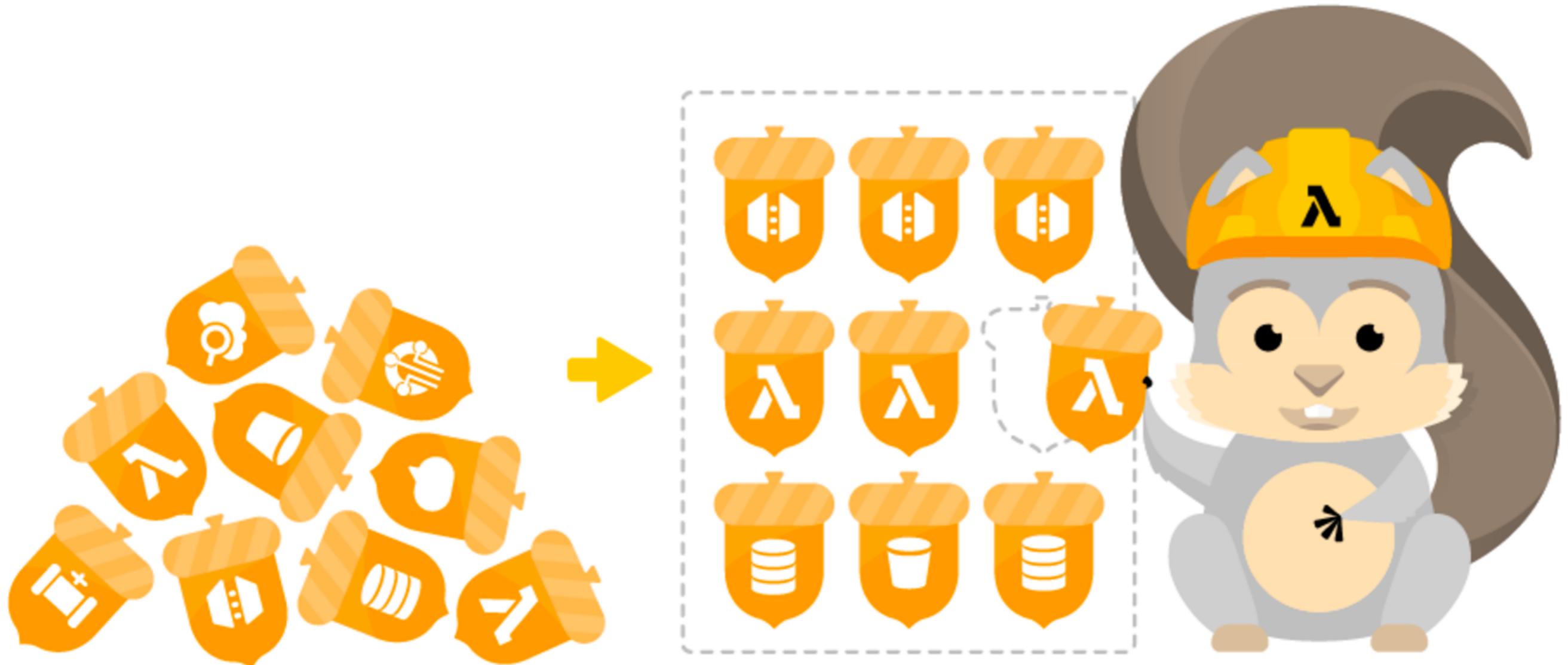
I love the simplicity of the chalice CLI, but feel there is one more command to add to complete the feature set. That is a `chalice delete` (or named something similar) command.

# AWS Serverless Application Model



MEET SAM.

# AWS SAM



USE SAM TO BUILD TEMPLATES THAT DEFINE  
YOUR SERVERLESS APPLICATIONS.

# 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

# AWS Lambda without any costs

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

**What if you want to run  
your Django app?**



**zappa**



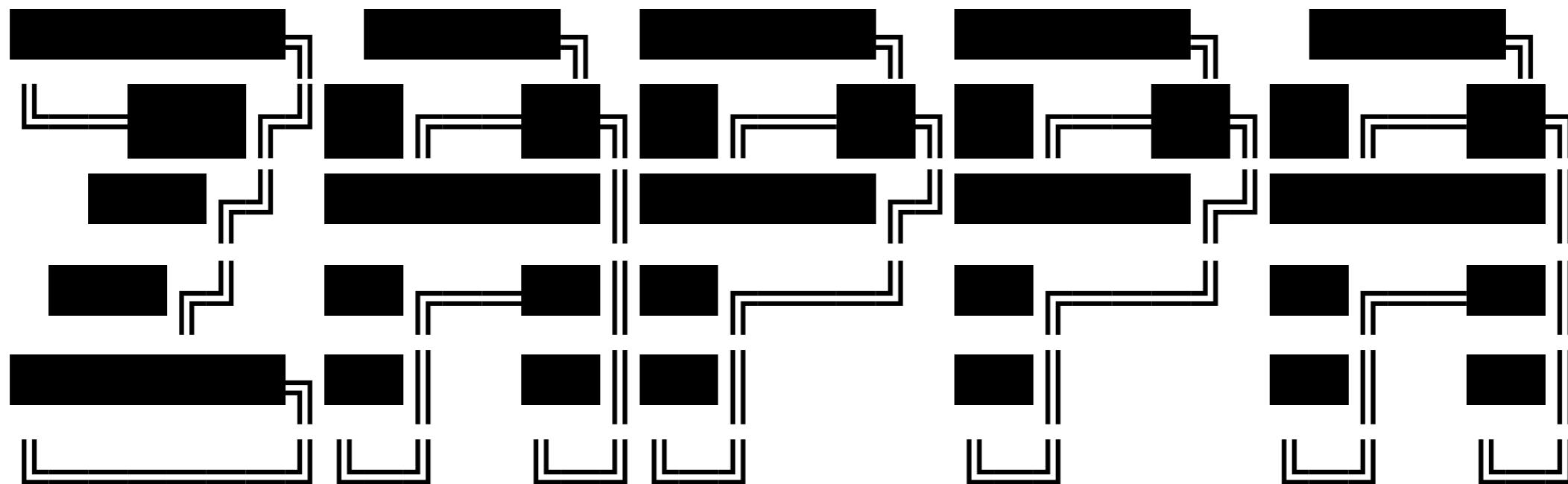
# Top 10 Python libraries of 2016

## 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.

And again what do you  
mean “serveless”?

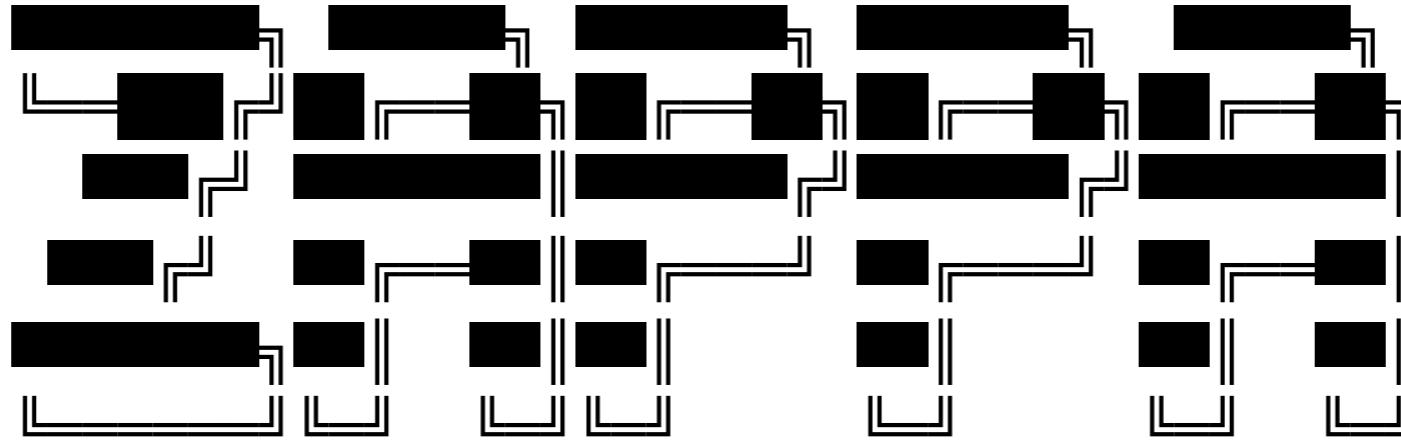
```
→ pip install zappa  
→ zappa init
```



Welcome to Zappa!

...

```
→ zappa deploy
```



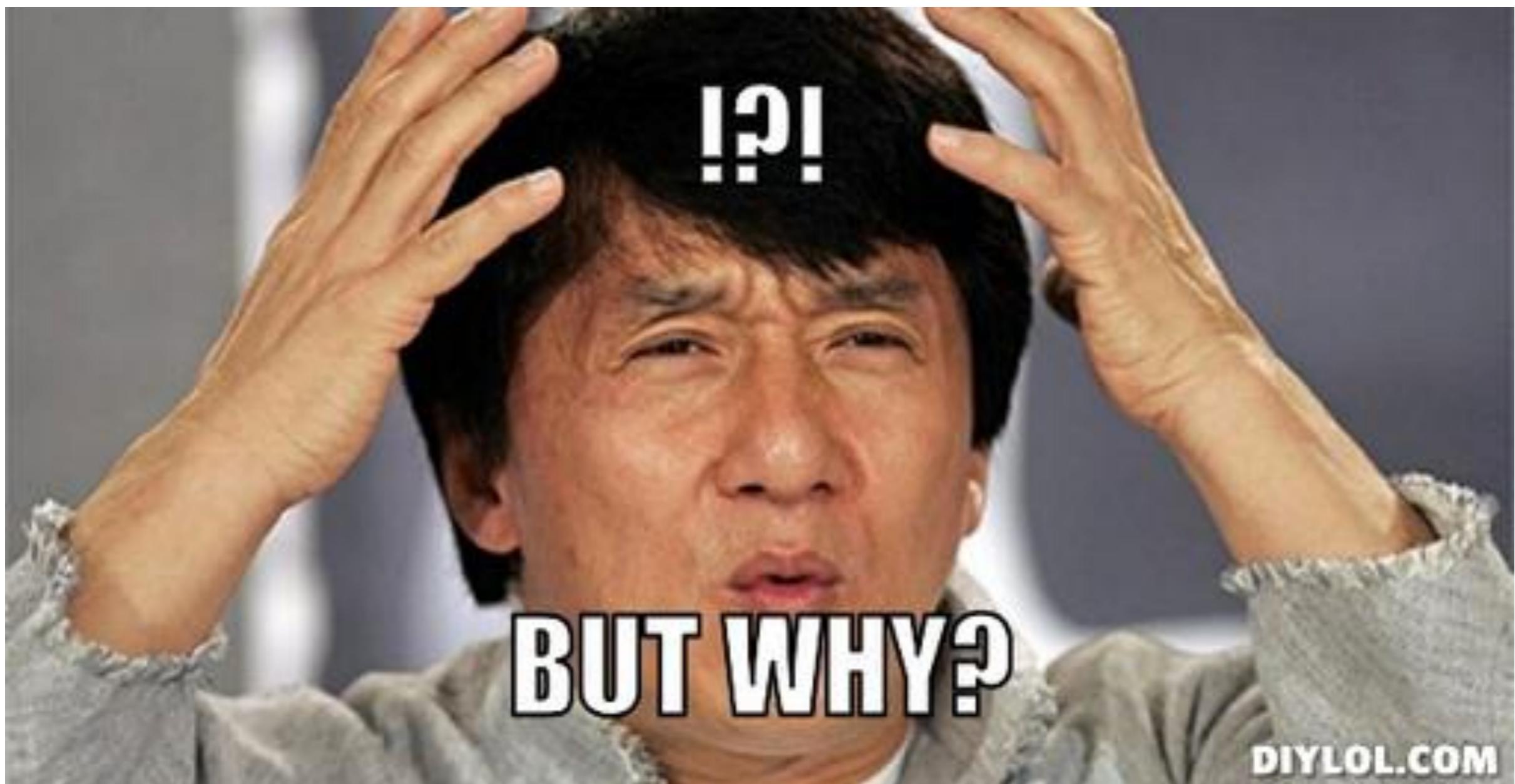
- deploy;
  - tailing logs;
  - run django manage.py
- ...

# AWS Lambda Limits

AWS Lambda supports the following runtime versions:

- Node.js – v4.3.2 and 6.10
- Java – Java 8
- Python – Python 2.7
- ~~.NET Core~~ – .NET Core 1.0.1 (C#)

# Python 2.7



# Python 2.7 will retire in...



<https://pythonclock.org/>

 @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

# 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

# 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

# Run python 3 from python 2



В рот мне ноги, это же Дэвид Блейн!

 @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
```

# 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)
```

# 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

Thanks Lyndon Swan  
for ideas about hacking  
python 3

# 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>

From Apr 18, 2017 AWS Lambda  
Supports Python 3.6

# Thank You

**andrii.soldatenko@toptal.com**

**https://asoldatenko.com**

# Questions



We are hiring



<https://www.toptal.com/#connect-fantastic-computer-engineers>