Python in The Serverless Era

Hello!

I am Benny Bauer

- Chief Architect @ WESCOVER
- Ex- AUTODESK :
  Cloud Architect @ AutoCAD
- Find me at **@benikbauer**

What is Serverless

Dev SaaS	BaaS	FaaS
Auth0	Firebase	
twilio		
Algolia		CLOUD FUNCTIONS Create small, single-purpose functions that respond to events in the cloud
Cloudinary		
ALGORITHMIA		

What is FaaS - Function as a Service

Fully-managed compute

Provisioning, patching, scaling, monitoring, logging are provided out-of-the-box

LESS OPS

Deploy your code

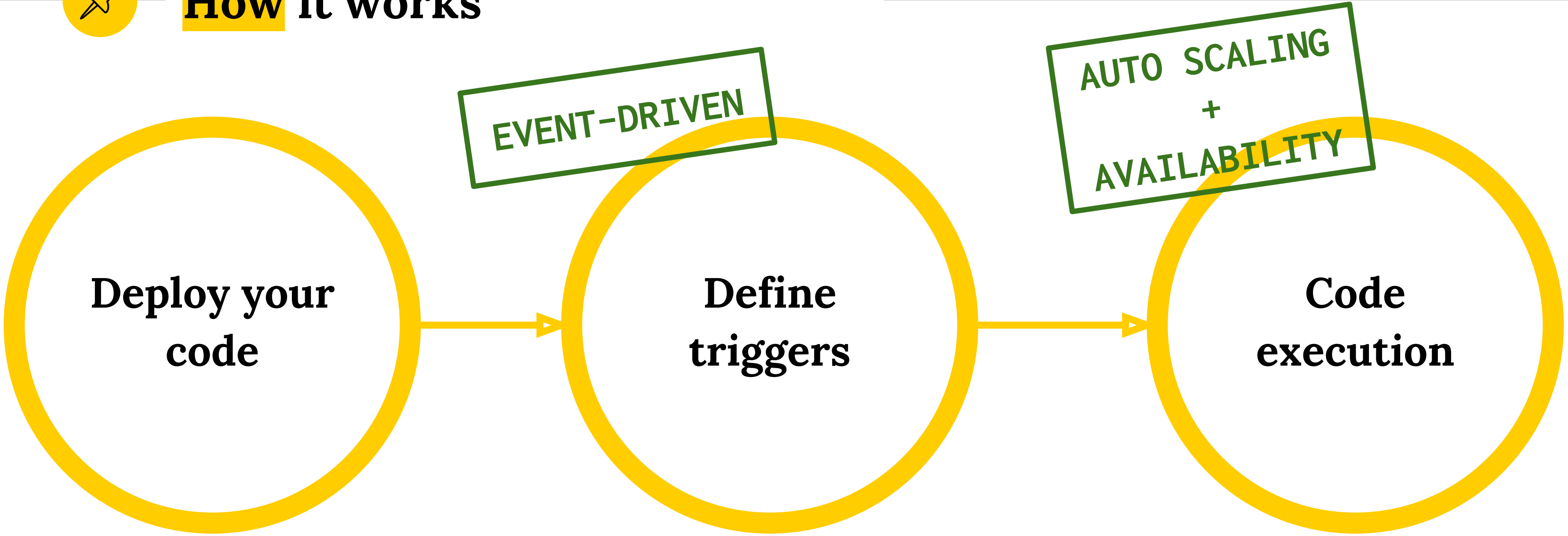
Just package and upload the code

Pay for actual usage

Getting charged only upon code execution, per 100ms

100% UTILIZATION

How it works



- HTTP requests
- Storage (e.g. file upload)
- DB (e.g. row insert)
- Scheduled tasks
- Messaging
- Many many more...

Use cases

API

- Web backend
- Mobile backend
- Bot

Data processing

- IoT
- Streams (analytics, logging, etc)
- Files (images, text, etc.)

Operations

- CI
- Policy enforcement
- Provisioning

Other

- Scheduled tasks
- Distributed compute

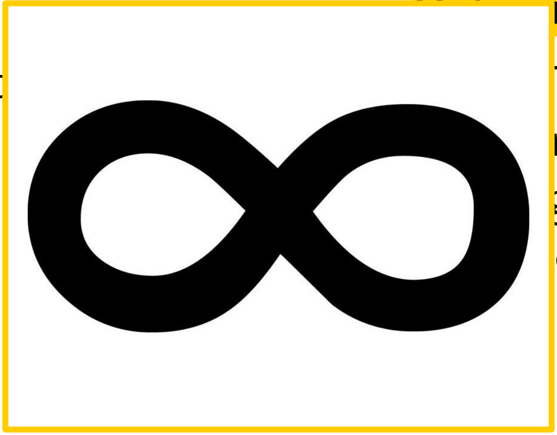
Use cases

API

- Web backenc
- Mobile backe
- Bot

rocessing

- T
- reams (analytics,
gging, etc)
- es (images, text, etc.)



Operations

- CI
- Policy enforcement
- Provisioning

- Scheduled tasks
- Distributed compute

Things to be aware of

Stateless

Instances are ephemeral. Store state on client-side, cache or db.

Cold start

Latency (< 2 sec) when container is cold (upon first run or inactivity).

Granularity

Unit of deployment is nanoservice/function.

Vendor lock-in

Integrations with other services are the real bait.

Costs

Cost-effective up to a certain point.

Limitations

Execution time is 5 min. Payload & disk sizes are limited.

Python & FaaS

Let’s get to business!

Cloud providers

Python 2.7 & 3.6 are supported by:

- AWS
- OpenWhisk
- Microsoft Azure – experimental, still not GA

The need for frameworks

Configuration

- Event binding
- Resources definition
- Security roles definition

Deployment

- Package
- Upload
- Rollback

Frameworks

SERVERLESS

Zappa

Chalice

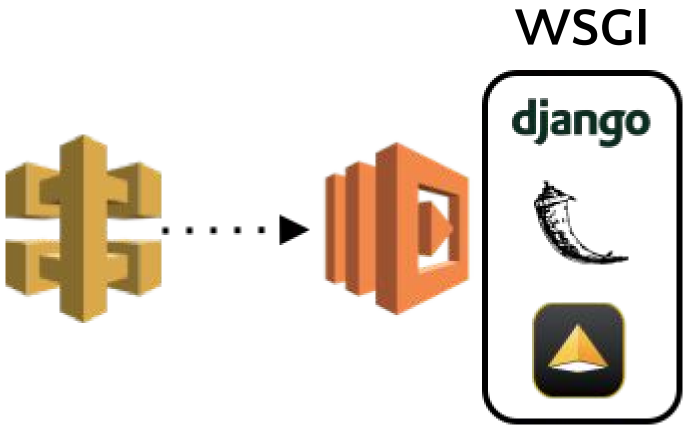
APEX

Gordon

Serverless Framework

SERVERLESS

Zappa



One function only. Serves as a WSGI server

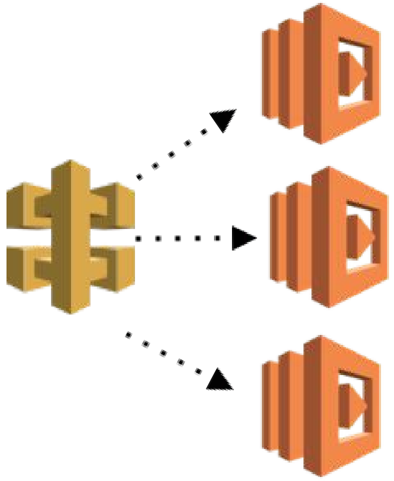
Zappa

Unique features:

- Global deployment
- “Keep warm” functionality
- SSL certification
- Support for AWS Lambda compatible python libraries (lambda-packages & Manylinux wheels)

Chalice

- Python Serverless Microframework for AWS
- Each endpoint is a separate function



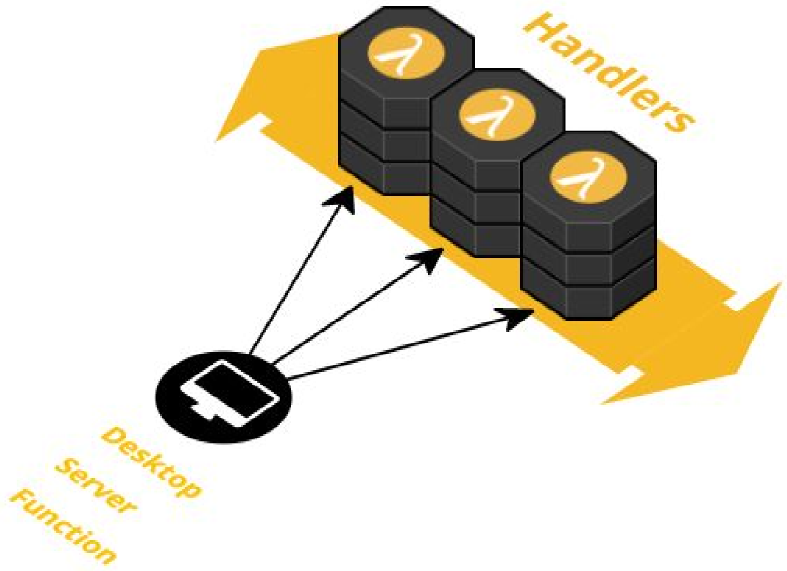
```
$ pip install chalice
$ chalice new-project helloworld && cd helloworld

$ cat app.py
from chalice import Chalice

app = Chalice(app_name="helloworld")

@app.route("/")
def index():
    return {"hello": "world"}

$ chalice deploy
```

Chalice

Unique features:

- Automatic IAM policy generation

Pywren



```
import pywren


def scrape(url):
    # scrape it...
  return data


wrenexec = pywren.default_executor()
futures = wrenexec.map(scrape, [url,...])
results = pywren.get_all_results(futures)
```

Credit: Sean Smith (https://blog.seanssmith.com/posts/pywren-web-scraping.html)

Summary

What was it all about?

Takeaways

- Serverless is a fast & cost-effective way to deliver many use cases with Python

- New frontiers for Python (frameworks, tooling, etc.)

Thanks!

Any questions?

Find me at **@benikbauer**