



**A CLOUD GURU**

## Serverless: Cloud functions and the future of software architectures

A Cloud Guru is our serverless on-demand video streaming platform built entirely on AWS.



Sam Kroonenburg  
Co-Founder & CTO  
A Cloud Guru

**@samkroon**



Peter Sbarski, PhD  
VP Engineering  
A Cloud Guru

**@sbarski**



serverless = compute + patterns

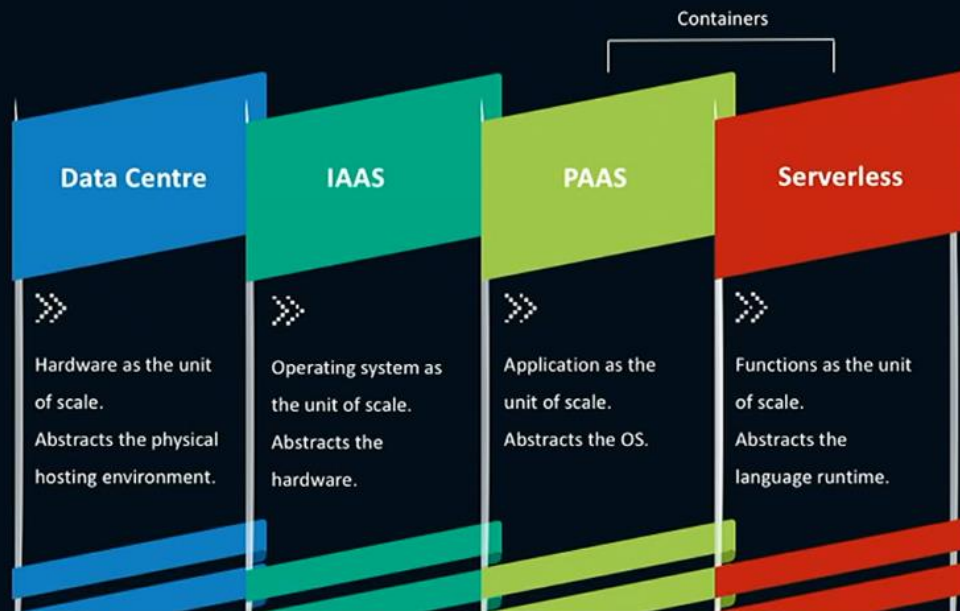
# Serverless Compute Technologies

FAAS



Event Driven

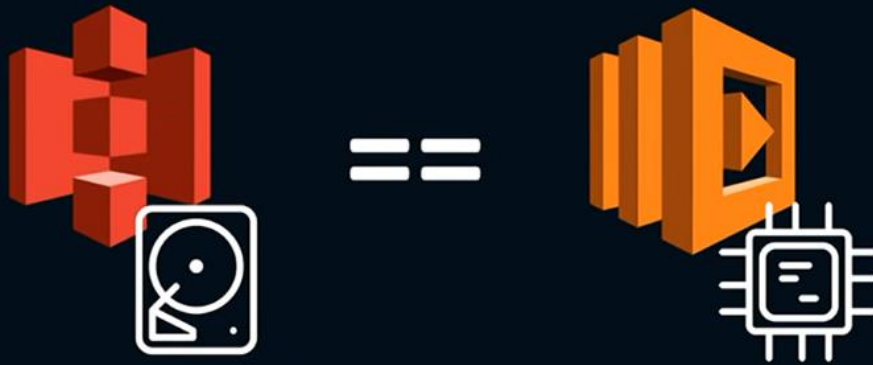
## A Brief History of Cloud



## AWS Lambda



Lambda is to compute, what S3 is to storage.



Is it really serverless?

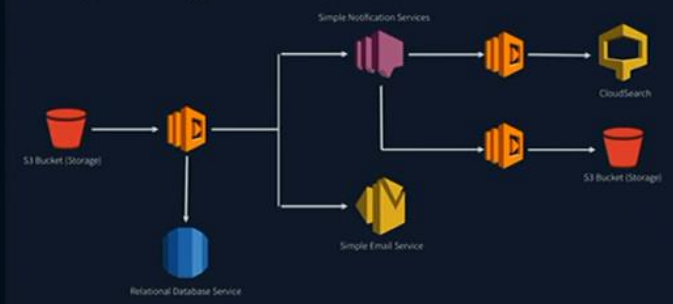
8

- Vendor takes care of provisioning and management of servers.
- Vendor is responsible for capacity provisioning and automated scaling.
- Moving away from servers and infrastructure concerns should be your goal.

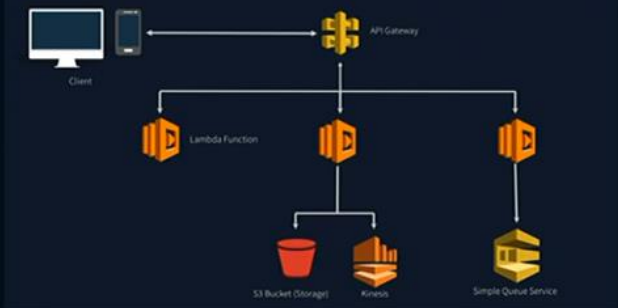


# Serverless Architectures

## Compute as glue



## Compute as back end



# Serverless is blowing up right now

10



# SERVERLESSCONF



**Mitch Garnaat**  
Creator of Java and AWS CLI



**Charity Majors**  
Engineer Co-Founder Band, formerly Percona/PostgreSQL



**Eric Windisch**  
Founder/CTO of Upex, formerly Docker



**Patrick Debois**  
Founder of OpenStack, CTO of Small Town Heroes

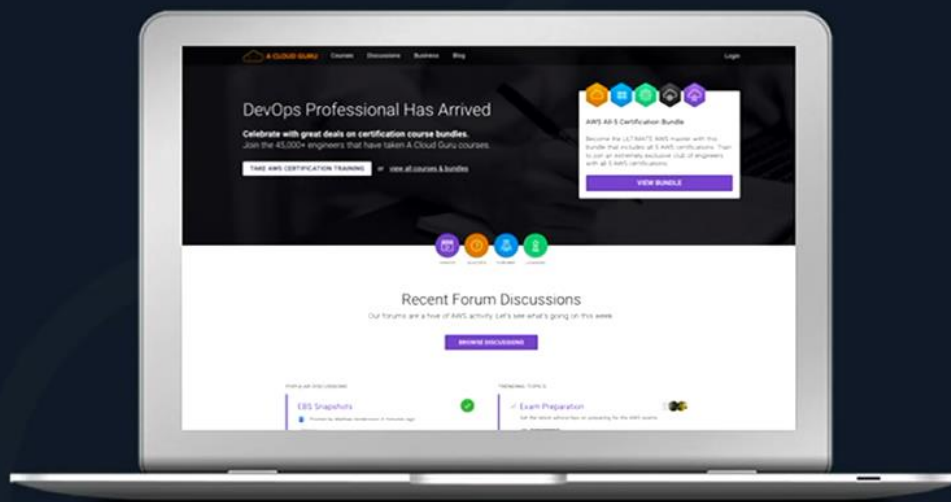
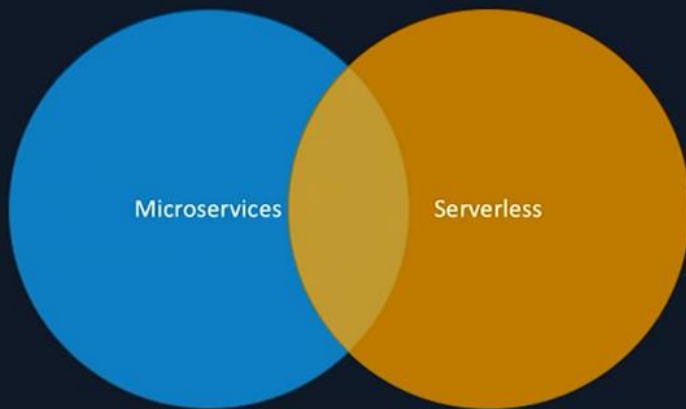


**Ben Kehoe**  
Cloud Robotics Research Scientist at Google



**Dr. Andreas Nauerz**  
Technical Product Manager for IBM Bluemix OpenWhisk

# Did someone say **microservices**?



@samkroon



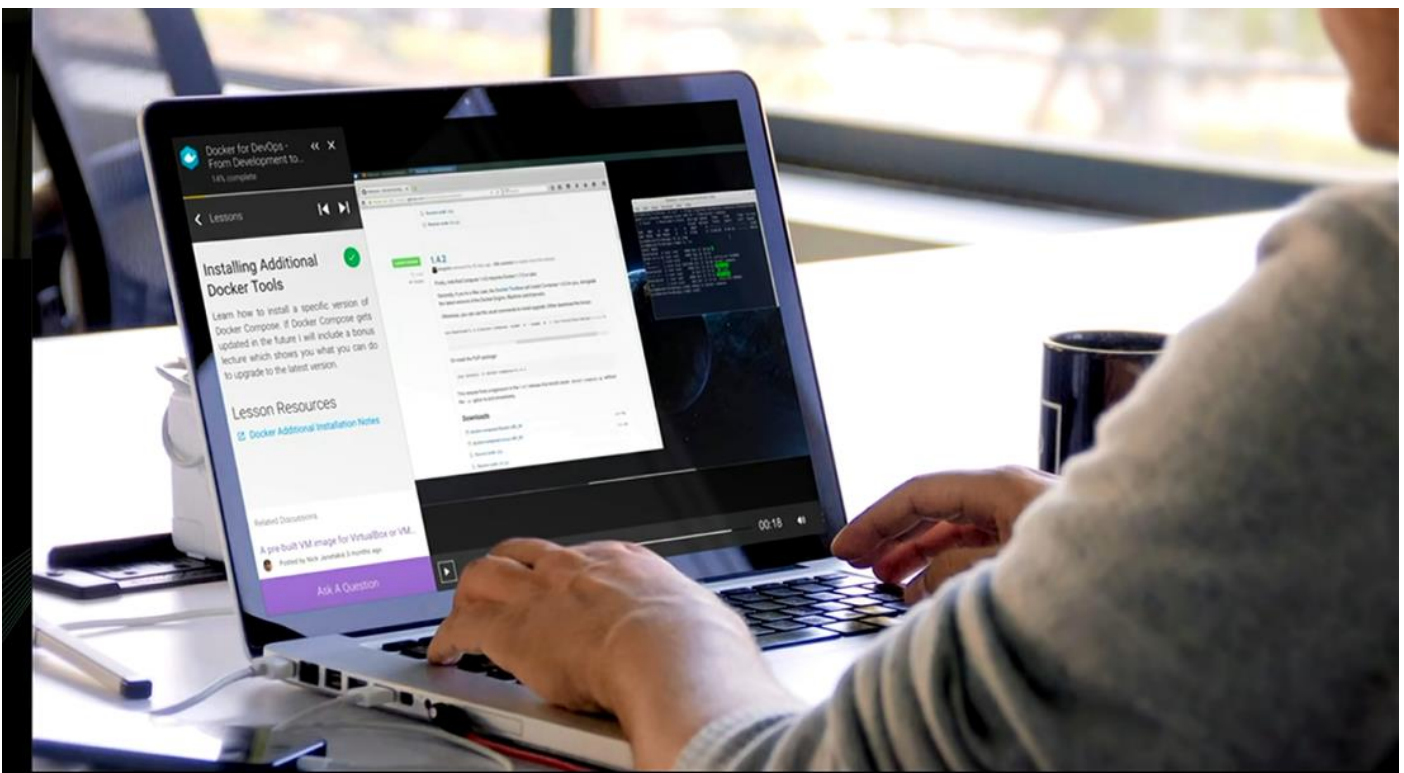
# 60,000+ engineers

Across 120 countries



@samkroon

And we don't run a single server.





Discussion forums

# Benefits



Rapid Time To Market



Scale Effortlessly



Disruptive Cost Model

## Principles of Serverless Architecture



Use a compute service to execute code on demand



Write single-purpose stateless functions



Design push-based, event-driven pipelines



Create thicker, more powerful front ends



Embrace third party services

**1. Use a compute service to execute code on demand (aka don't run a server).**

2. Write single-purpose stateless functions



# Our Lambda Functions

A look at what we built

Submit Question



Submit Answer



Vote



Gurubot



Take Payment



Request Protected File



## 3. Design push-based, event-driven pipelines

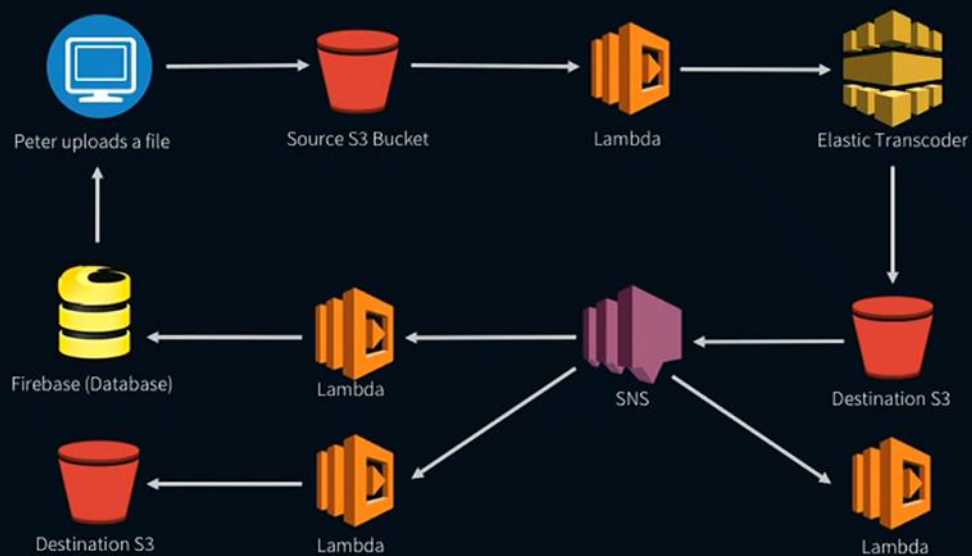
# Forum Comment



# Comment is free



# Encoding Media



## 4. Create thicker, more powerful front ends

01

User Interface

02

Client Side Model Binding

03

Client Side Service Layer

04

Server Side Service Layer

05

Server Side Model Binding

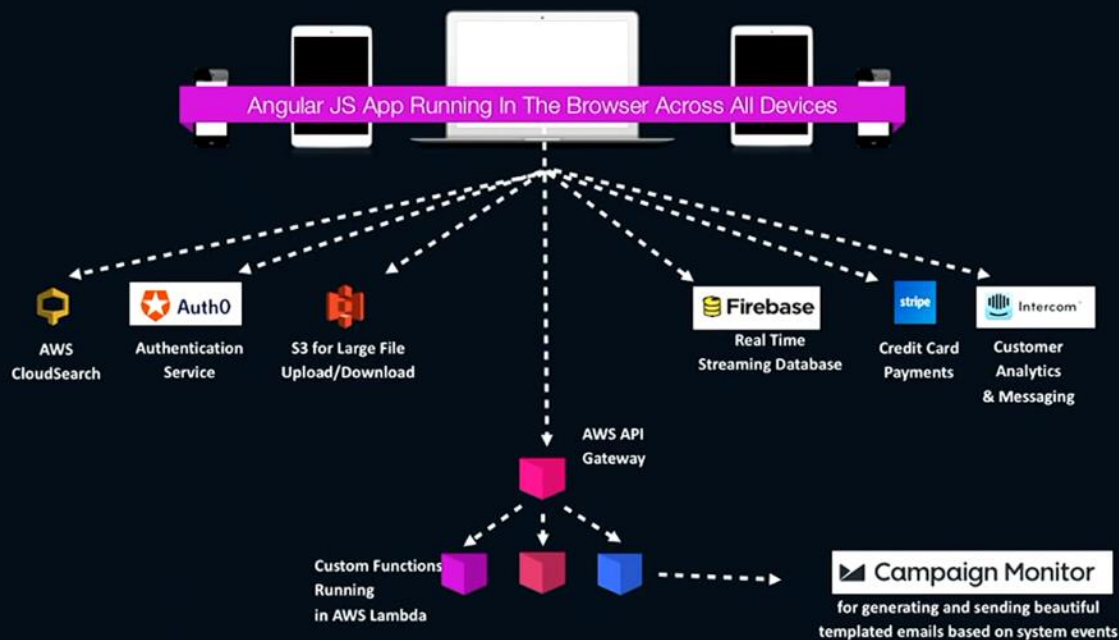
06

Server Side DB Mapping

07

Database Storage

@samkroon



01

User Interface

02

Client Side Model Binding

03

Client Side Service Layer

04

Database Storage

05

Cloud Functions

## 5. Use third party services

### Reasons Not To Go Serverless



Performance



Service levels and customization



Vendor lock-in



Decentralization

**SERVERLESSCONF**

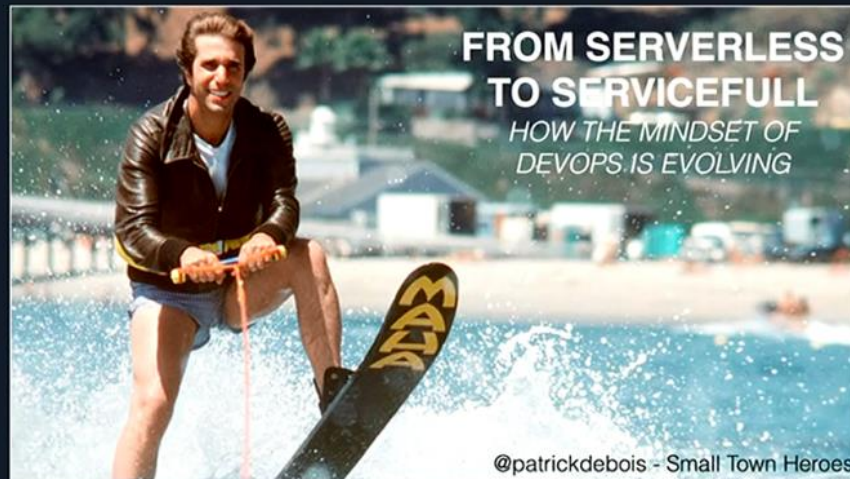
*BROOKLYN, NYC. MAY 2016.*



# From Serverless to Servicefull

## Patrick Debois

40



### “Backend” services



### “IT support” services



## Our “Office” services



## “Frontend” services

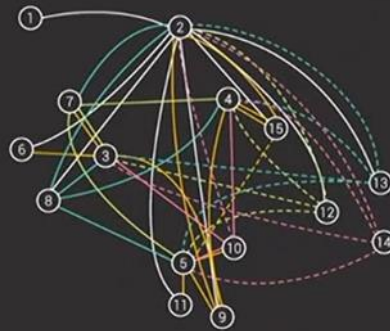


## “Mobile” services



# Promise Theory

## Principles and Applications



Jan A. Bergstra and Mark Burgess



**Jonathan Clarke**  
@jooooooooon42



Following

GitHub's broken. Everyone take the afternoon off.

LIKE

1



5:27 PM - 23 May 2016



Reply to @jooooooooon42

**Amazon Web  
Services**

Feb 9, 2016  
01:27 PM -0500



Hello,

I'm glad you've found my information helpful, I hope that your appeal process goes smoothly. As soon as you send a response to [ses-enforcement@amazon.com](mailto:ses-enforcement@amazon.com) outlining the changes you're making, you can normally see around a 24-hour turn-around on your appeal, of course this is assuming that ses-enforcement views your appeal and sees that you've appropriately addressed the bounces.

If you've not received a response from ses-enforcement after a day, feel free to reach out to me and I'll work towards get your account re-enabled.

As always, if you have any further questions or concerns feel free to contact us.

Best regards,

service got disabled because of too many bounces



Amazon Web  
Services  
Mar 29, 2016  
03:26 PM -0400



Hello,

Unfortunately, the conditions for container re-use is non-deterministic, but it generally involves when a Lambda function is executed, the container for the Lambda function is created. To save time on this setup, the container can be reused for other Lambda functions if the next invocation is within a short period.

For example, you have 2 people invoking your Lambda function. Each function will have it's own container. After some time, one of those people is finished invoking, but the other is not finished before a third person invokes your Lambda function. This third person will take up the container that the first person utilized, rather than create their own, if that container is available. There would be some effects left over from the previous execution. If enough time passes before another execution, the container would be deleted, and any new people would have to create a new container when invoking the Lambda function.

container re-use  
non-deterministic

“The **collaboration**  
between dev & ops is now  
**extended** to external 3rd parties”

Even when you outsource a problem,  
**you are still responsible for the results.**

## Facts:

No one will ever care as much about your product  
and your problems as much as you do.

No one else is thinking holistically  
about your systems and their  
interdependencies.

Labor can be outsourced.  
Caring can't.



# Webtaskalifragilisticex Tomasz Janczuk

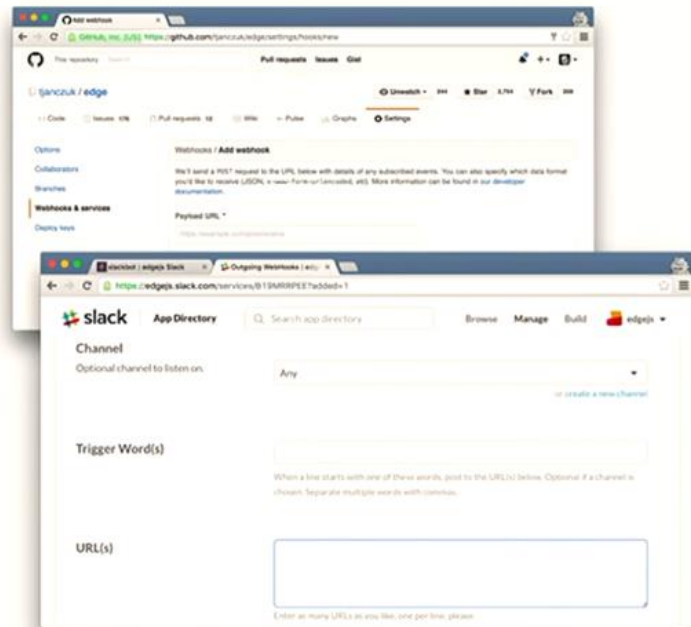
61



## Webtaskalifragilisticexpialidocious

Tomasz Janczuk (@tjanczuk)  
Auth0

Webhooks offer  
ultimate flexibility  
for SaaS  
extensibility



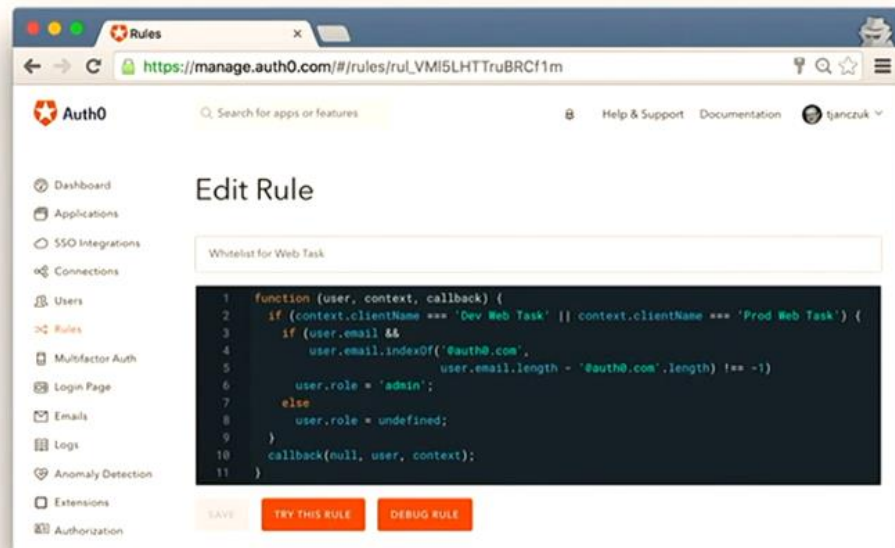
Webhooks  
are so 2015...

- Mostly async
- Servers
- Maintenance
- Scaling
- Monitoring
- Separate billing
- SSL
- Toil, grief, and tears



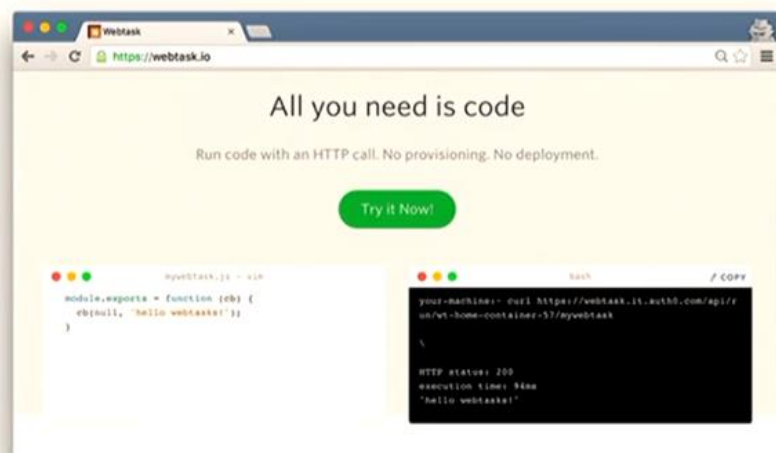
Webtask is  
a webhook  
with server  
included

Serverless  
webhook



Webtask design  
priorities

- Isolation
- HTTP fidelity
- Node.js
- Low latency
- Cloud and on-premise
- Scalability
- Developer love



# Server Optional Development

## Mike McDonald & Frank van Puffelen

66



## Server Optional Development

Lessons learned while building Firebase



Mike McDonald  
@asciimike



Frank van Puffelen  
@puf



# IBM Bluemix OpenWhisk

## Michael Behrendt & Dr. Andreas Nauerz

69



## Usage Scenarios



Microservices-based apps / APIs



Mobile Backends



Data (Stream) Processing



IoT



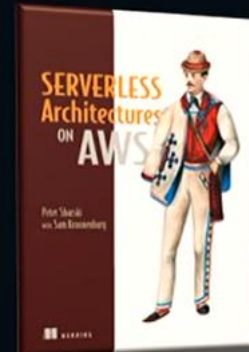
Cognitive



Bots

# How can you get started?

- <http://acloud.guru> - Have a course on lambda, and we'll post serverless meetup talks here.
- Follow [@acloudguru](https://twitter.com/acloudguru) for serverlessconf videos
- Book: "Serverless Architectures on AWS"  
<http://serverless.acloud.guru> - 40% off code



@samkroon