Five things you need to know about serverless

gojko.net/assets/gotoberlin2019.pdf

"Server"

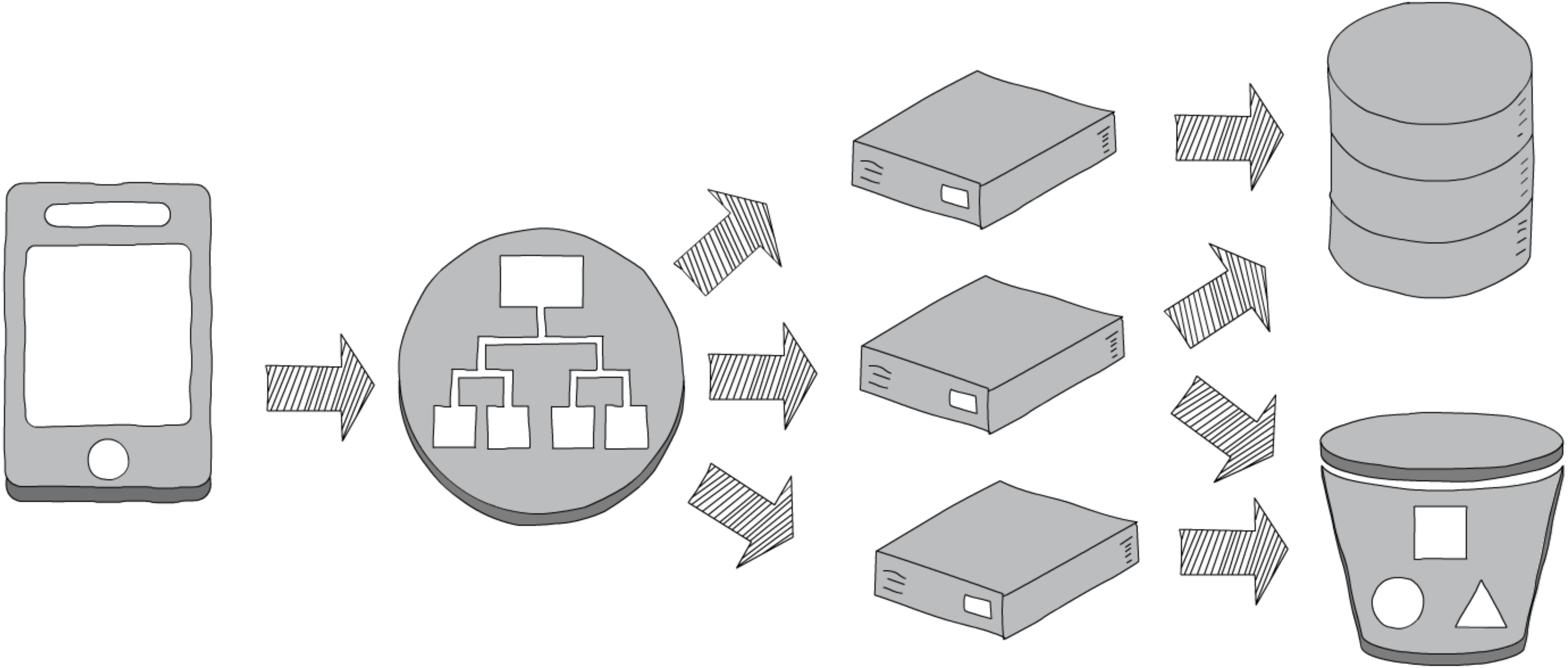
NGINX

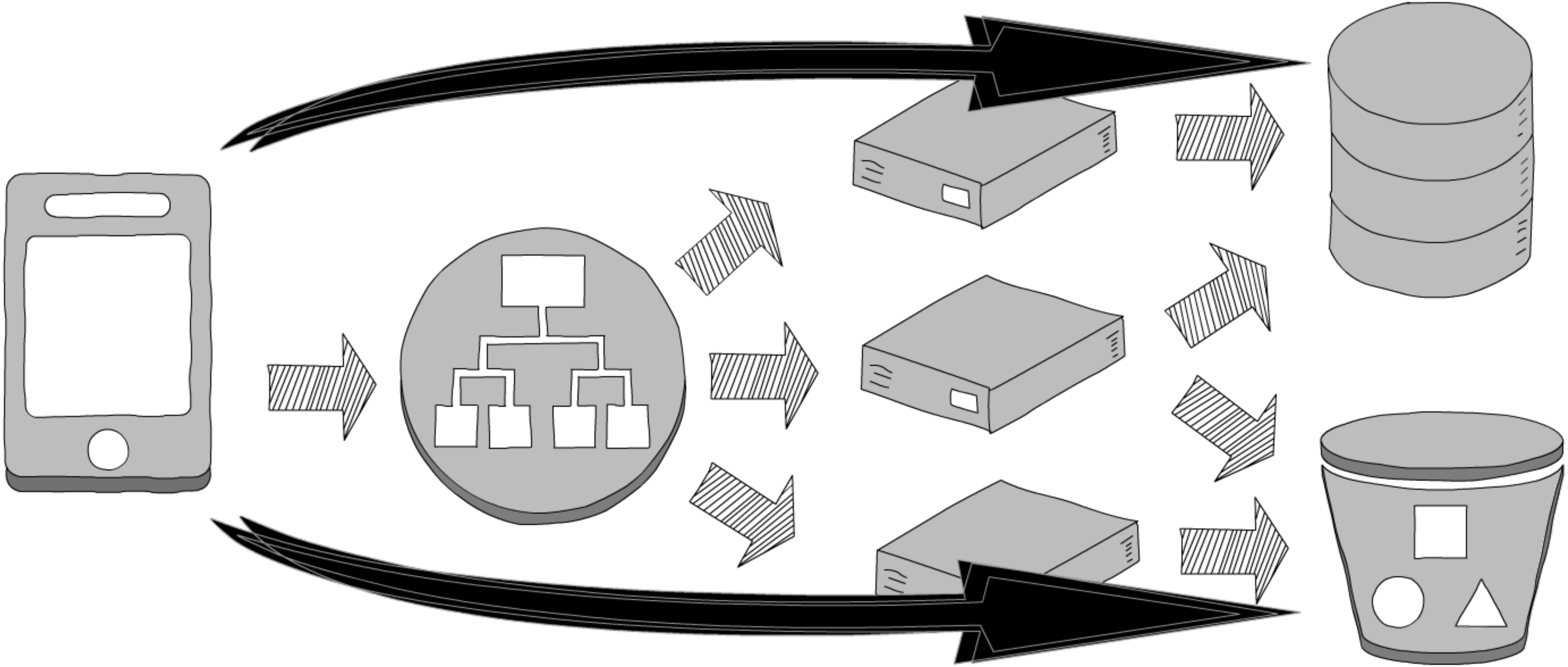
APACHE

ACTIVE MQ

JBoss®

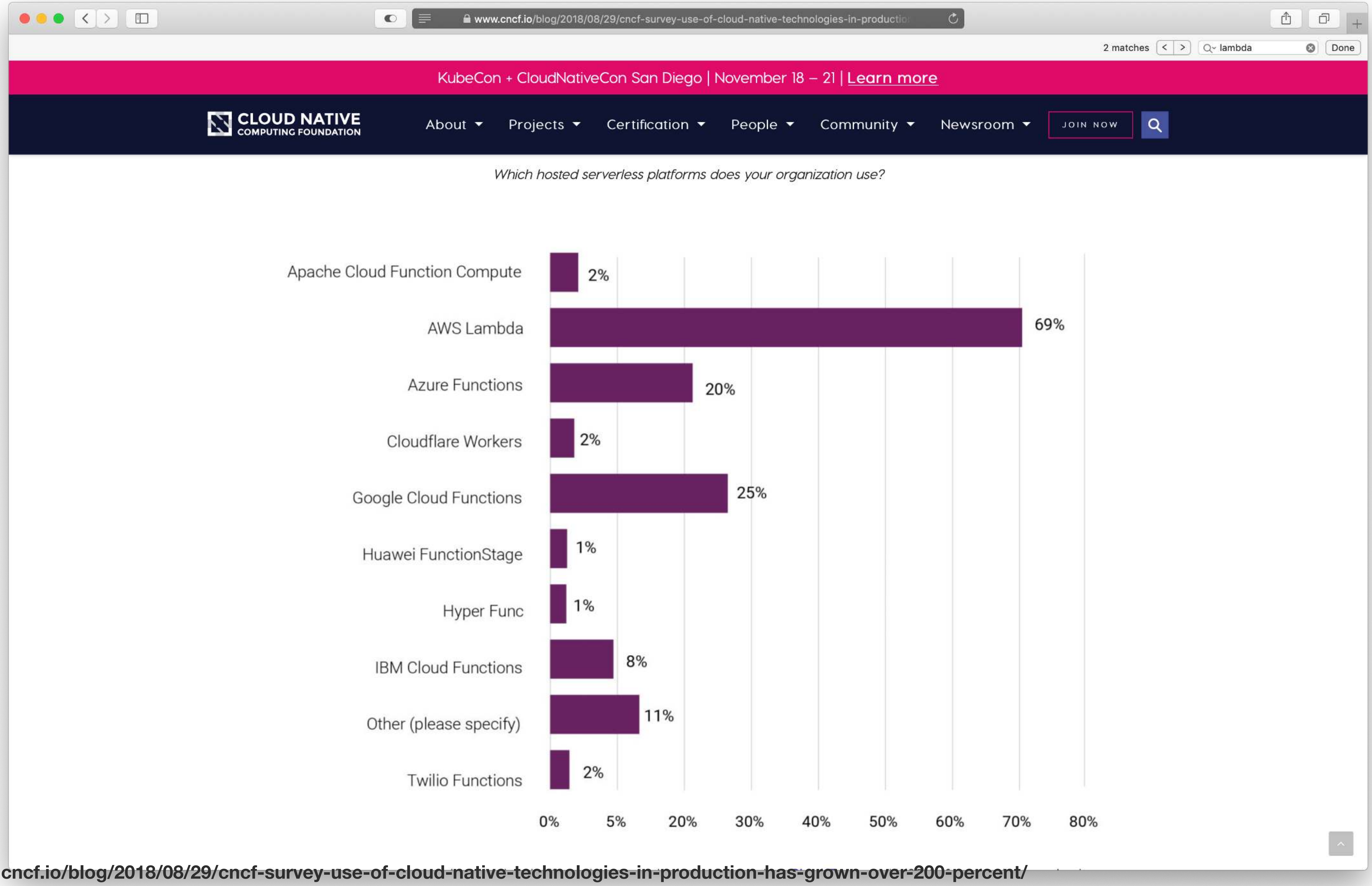
by Red Hat





~~Serverless~~ Socketless

- Why it's important
- Key tech constraints
- Key design constraints
- How to make the most of it
- Where it's going in the future



KubeCon + CloudNativeCon San Diego | November 18 – 21 | Learn more

Which hosted serverless platforms does your organization use?

https://www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-in-production-has-grown-over-200-percent/

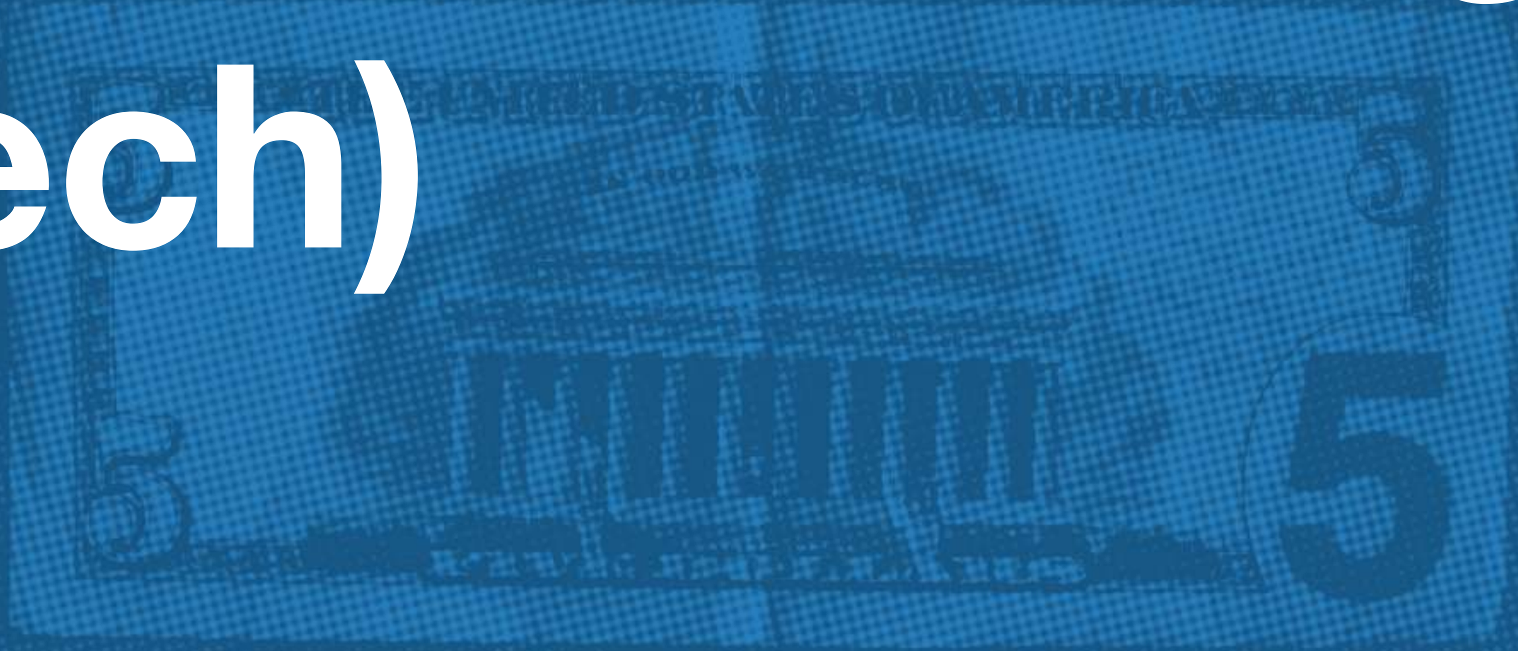
AWS Lambda model

```
exports.handler = function (event, context /*, *callback*/) {
  // do something useful with the event
}
```

Missverstandenfinanziellervorteil

~~Missverstandenfinanziellervorteil~~

Financial advantage (not tech)

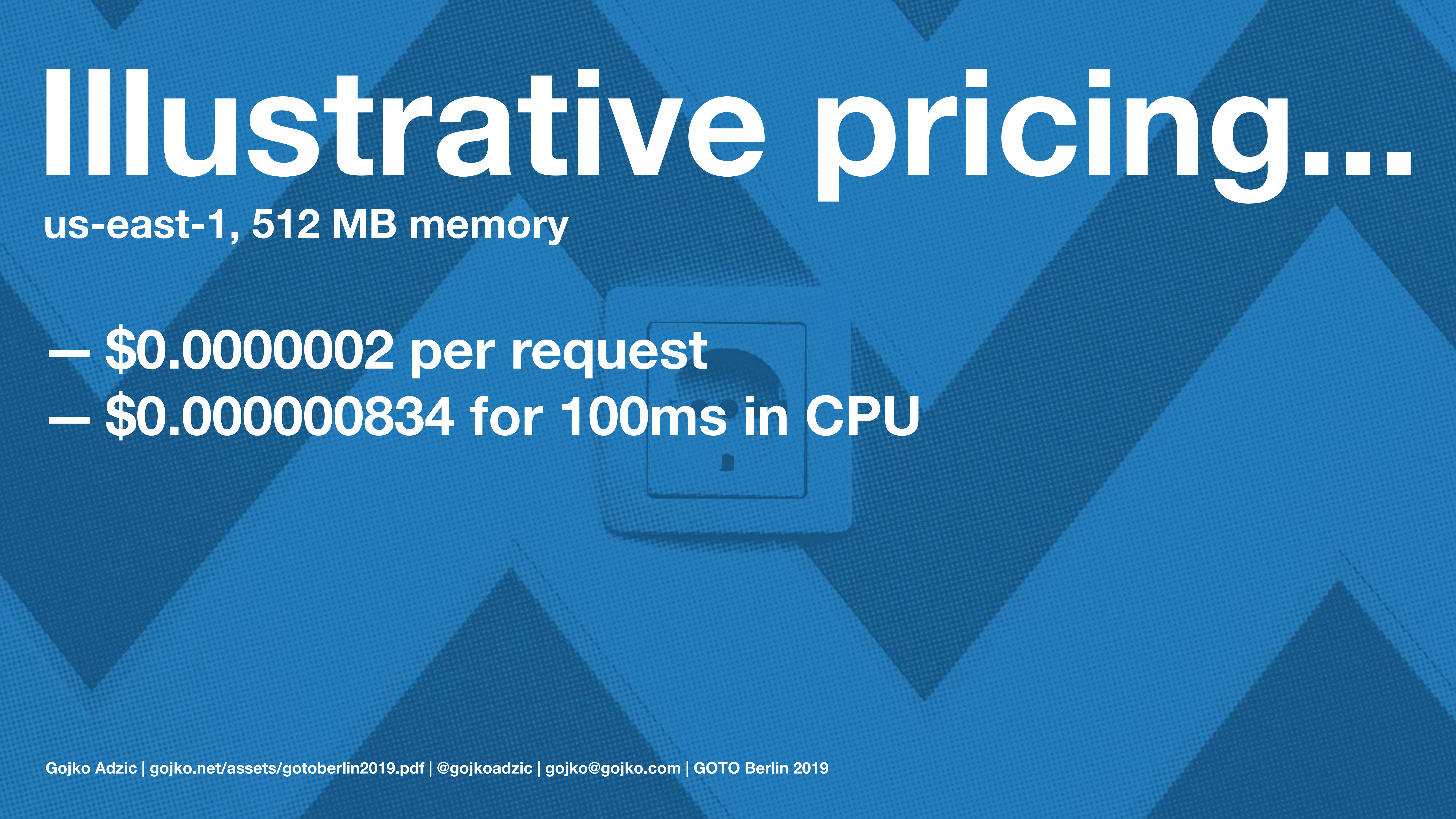


Paying for utilisation

not capacity

not environments

not service instances



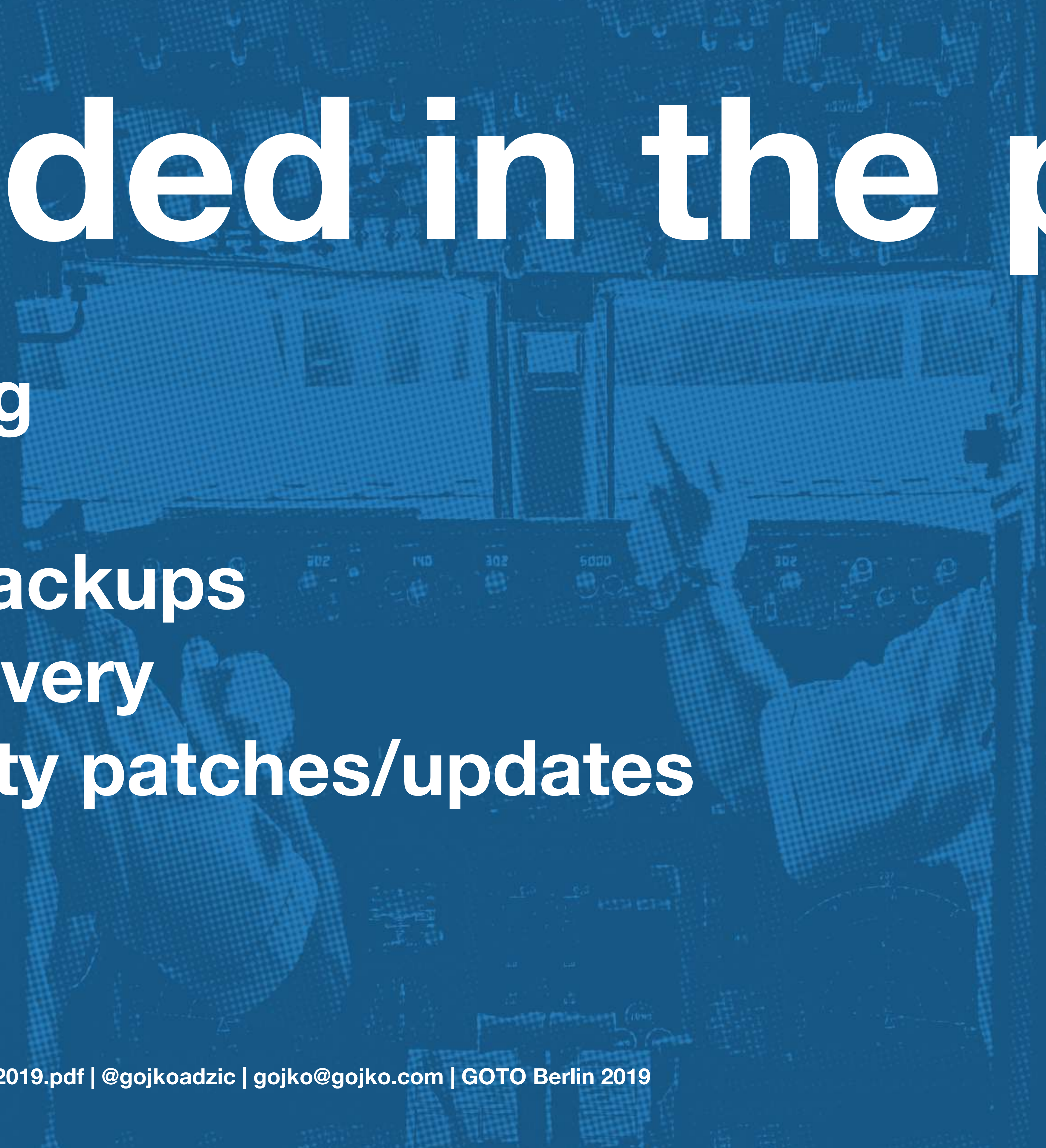
Illustrative pricing...

us-east-1, 512 MB memory

- $0.0000002 per request
- $0.000000834 for 100ms in CPU

monthly pricing...

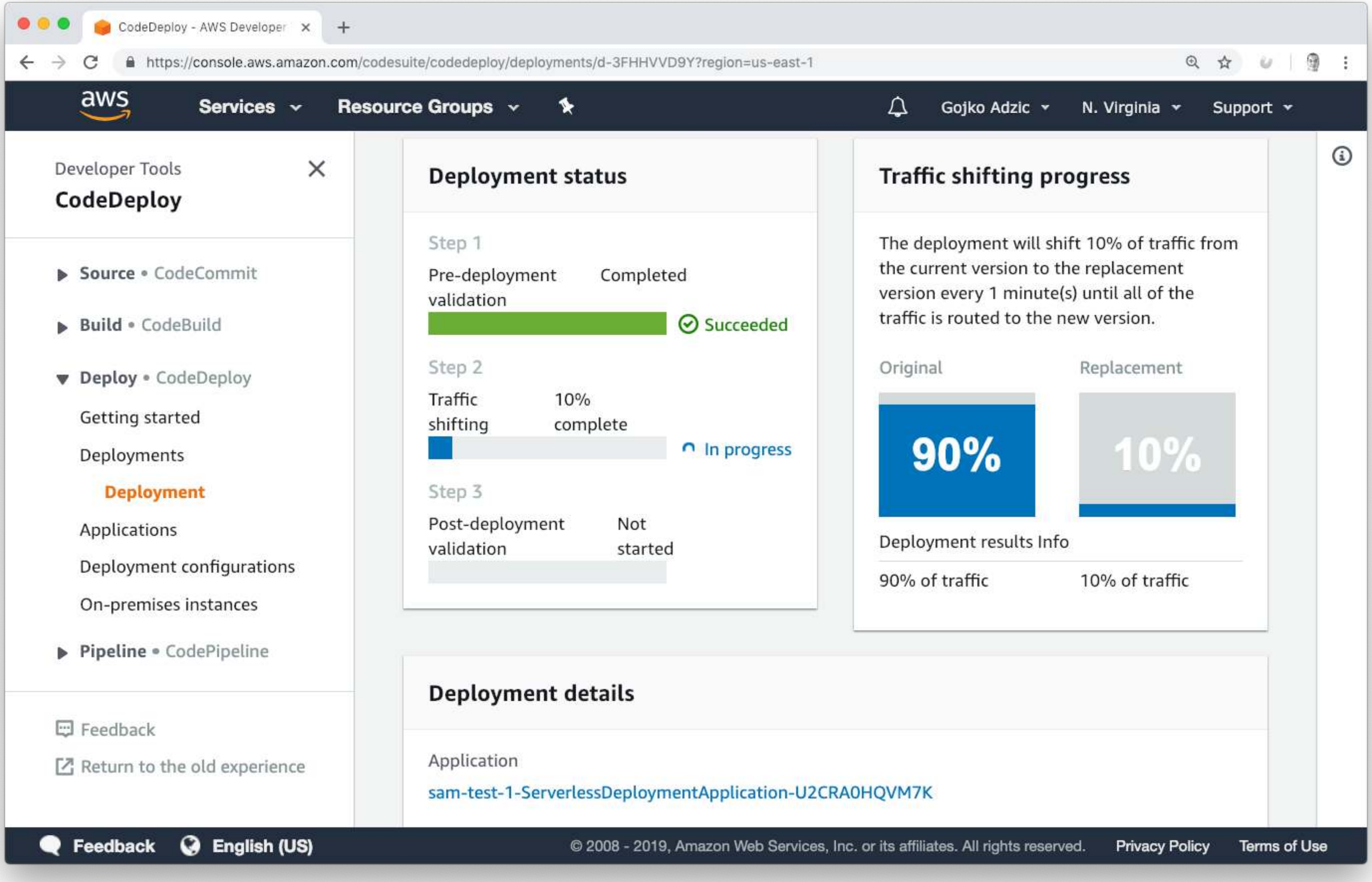
- 100ms every 5 minutes = 1¢
- non-stop = $27
- EC2 primary + failover = $9



Included in the price

— monitoring
— scaling
— failover/backups
— error recovery
— OS security patches/updates

Versions/environments have no effect on price



Deployment status

Step 1

Pre-deployment validation — Completed — ✓ Succeeded

Step 2

Traffic shifting — 10% complete — In progress

Step 3

Post-deployment validation — Not started

Traffic shifting progress

The deployment will shift 10% of traffic from the current version to the replacement version every 1 minute(s) until all of the traffic is routed to the new version.

Original	Replacement
90%	10%

Deployment results Info

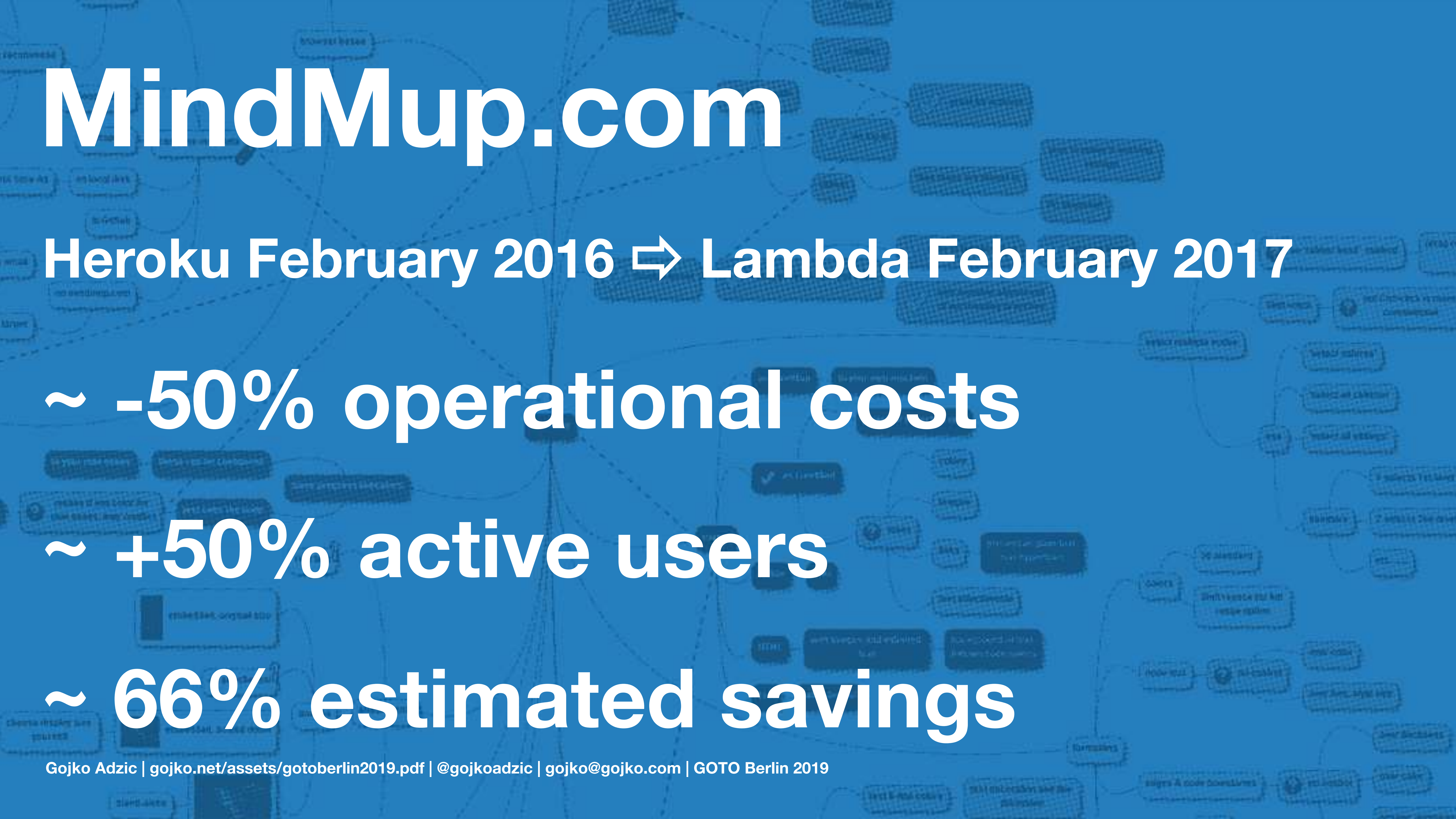
90% of traffic	10% of traffic

Deployment details

Application

sam-test-1-ServerlessDeploymentApplication-U2CRA0HQVM7K

```
ConvertFileFunction:
  Type: AWS::Serverless::Function
  Properties:
    DeploymentPreference:
     Type: Canary10Percent10Minutes
     Alarms:
       - !Ref CheckForLambdaErrors
       - !Ref CheckForDropInSales
       - !Ref CheckForDropInConversion
     Hooks:
       PreTraffic: !Ref ClearStatisticsLambda
       PostTraffic: !Ref NotifyAdminsLambda
```



MindMup.com

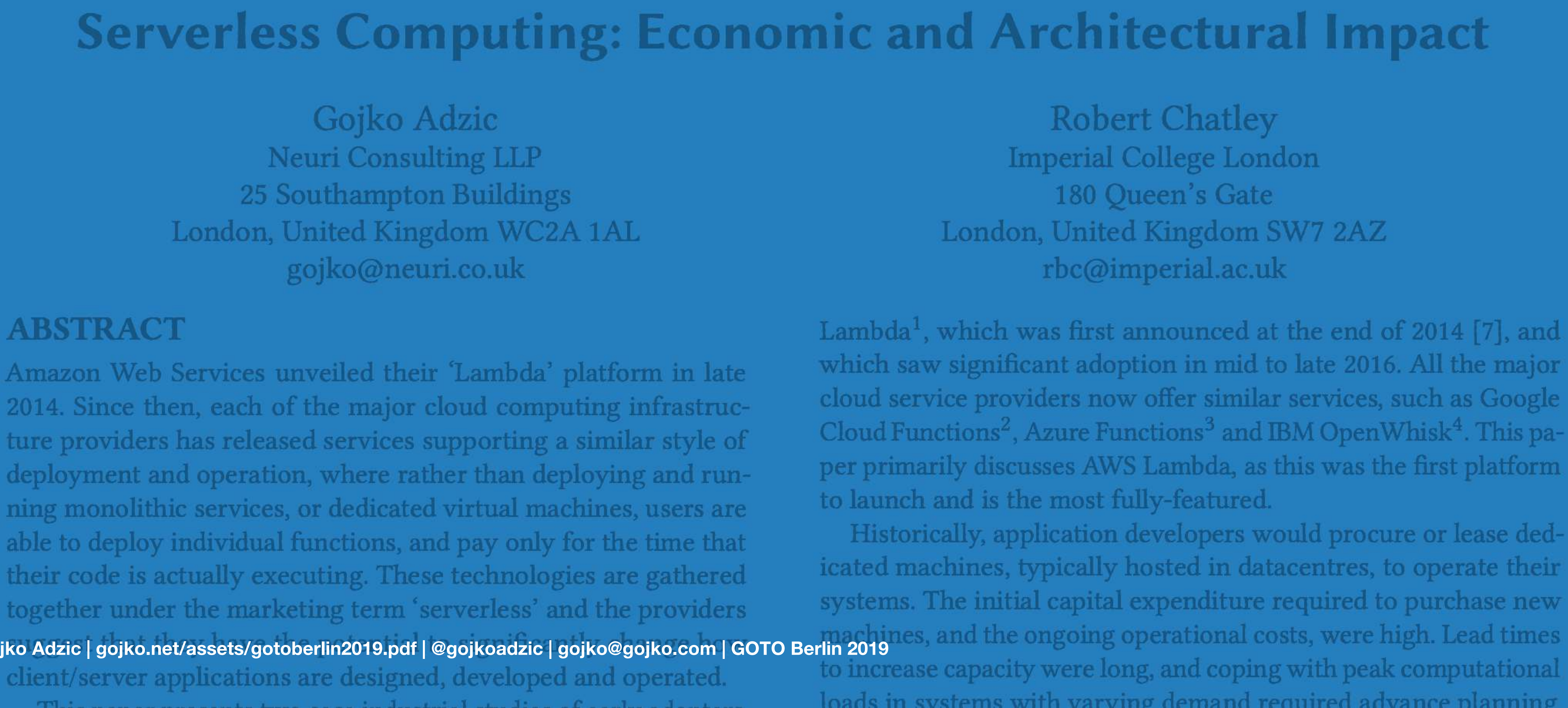
Heroku February 2016 ⇨ Lambda February 2017

~ -50% operational costs

~ +50% active users

~ 66% estimated savings

https://dl.acm.org/citation.cfm?id=3117767



Serverless Computing: Economic and Architectural Impact

Gojko Adzic
Neuri Consulting LLP
25 Southampton Buildings
London, United Kingdom WC2A 1AL
gojko@neuri.co.uk

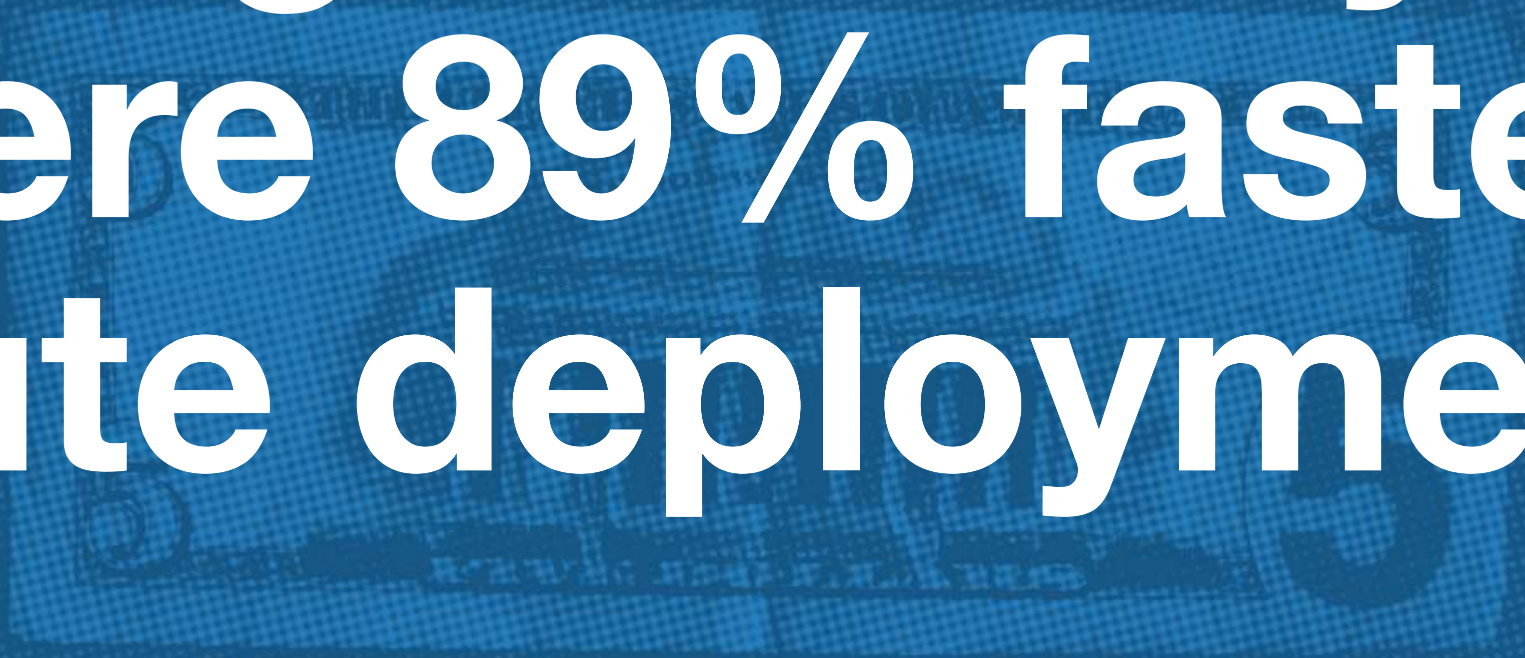
Robert Chatley
Imperial College London
180 Queen's Gate
London, United Kingdom SW7 2AZ
rbc@imperial.ac.uk

ABSTRACT

Amazon Web Services unveiled their 'Lambda' platform in late 2014. Since then, each of the major cloud computing infrastructure providers has released services supporting a similar style of deployment and operation, where rather than deploying and running monolithic services, or dedicated virtual machines, users are able to deploy individual functions, and pay only for the time that their code is actually executing. These technologies are gathered together under the marketing term 'serverless' and the providers suggest that they have the potential to significantly change how client/server applications are designed, developed and operated.

Lambda[1], which was first announced at the end of 2014 [7], and which saw significant adoption in mid to late 2016. All the major cloud service providers now offer similar services, such as Google Cloud Functions[2], Azure Functions[3] and IBM OpenWhisk[4]. This paper primarily discusses AWS Lambda, as this was the first platform to launch and is the most fully-featured.

Historically, application developers would procure or lease dedicated machines, typically hosted in datacentres, to operate their systems. The initial capital expenditure required to purchase new machines, and the ongoing operational costs, were high. Lead times to increase capacity were long, and coping with peak computational loads in systems with varying demand required advance planning



“lowered five-year operating costs by 60% and were 89% faster at compute deployment”

— IDC white paper on AWS Serverless

https://pages.awscloud.com/Gated_IDC_Generating_Value_Through_IT_Agility.html

“fourth quarter of 2017… serverless adoption grew by 667%”

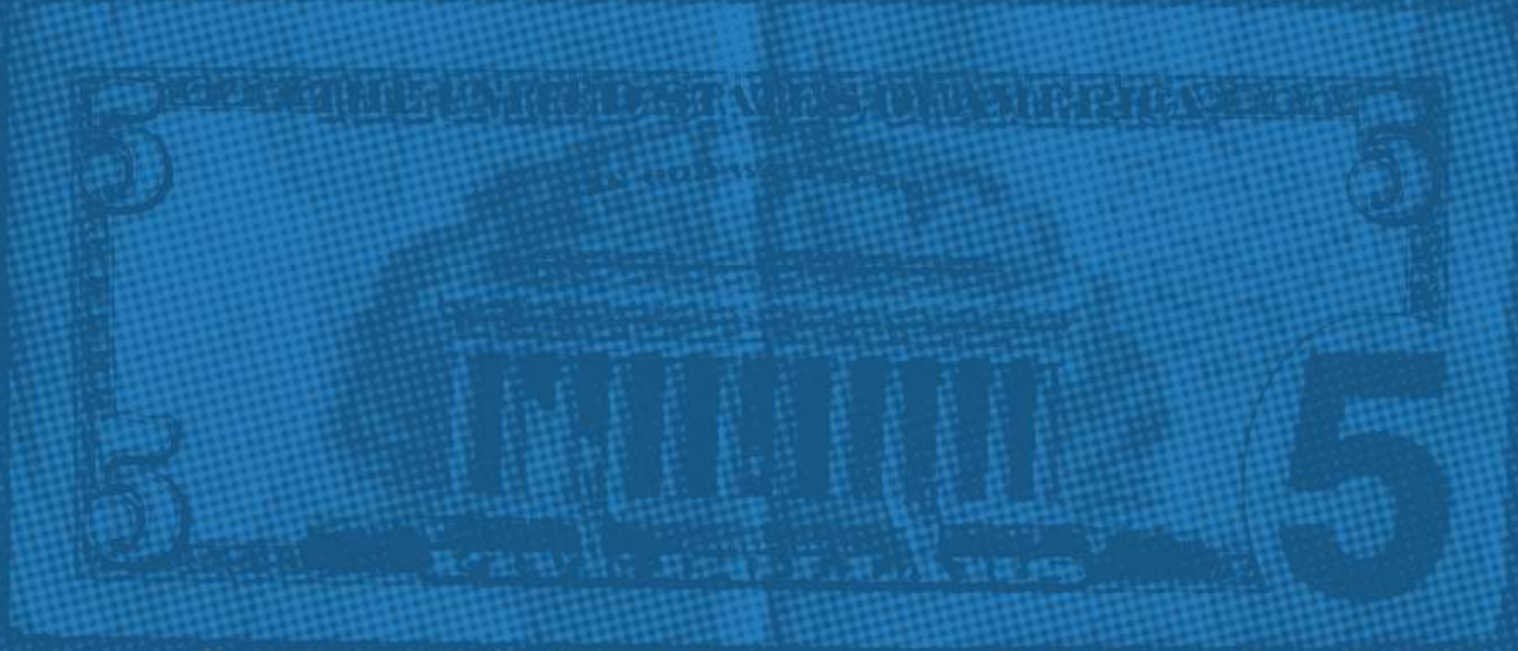
— Cloudability research

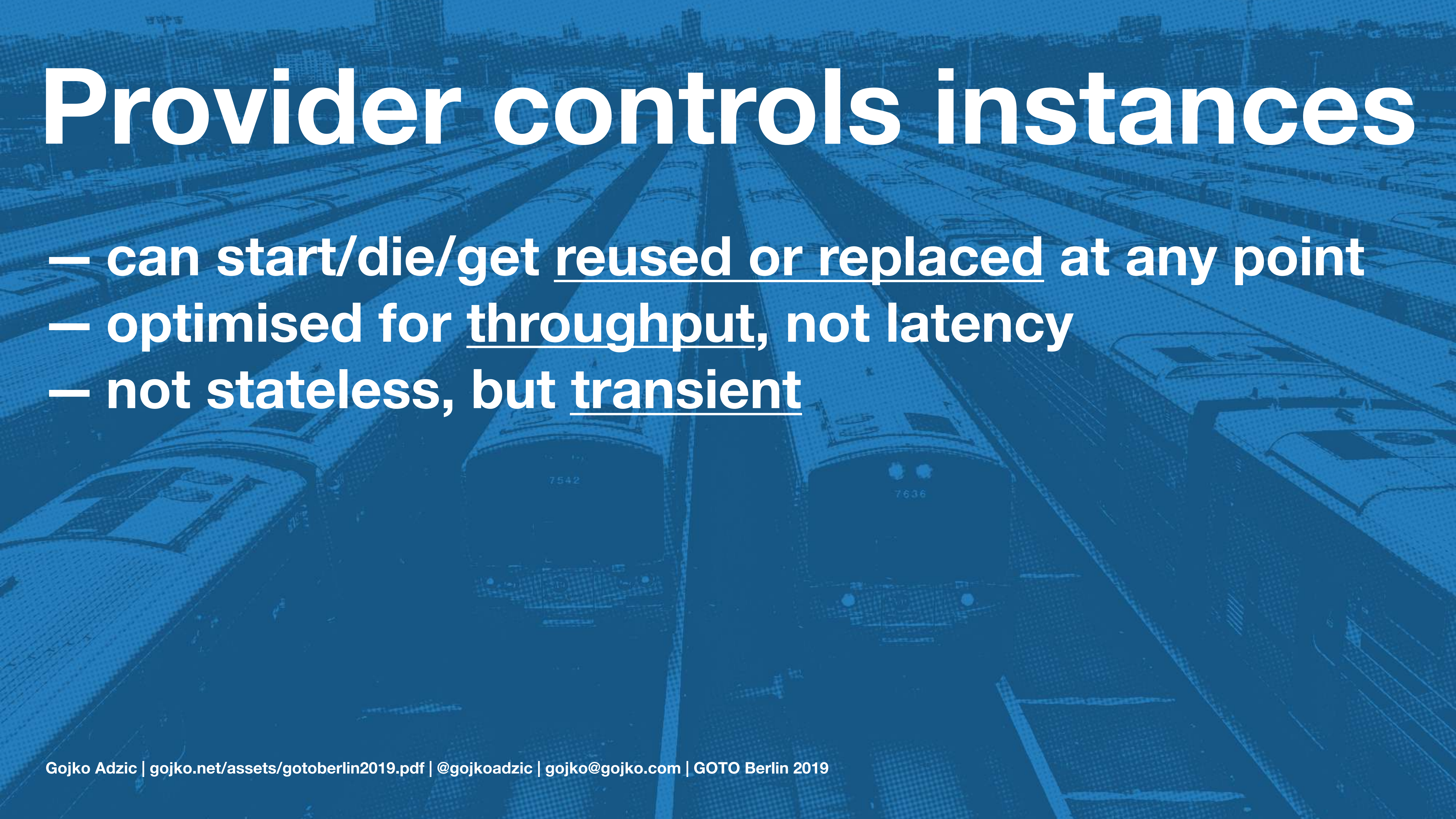
`https://www.zdnet.com/article/serverless-computing-containers-see-triple-digit-quarterly-growth-among-cloud-users/`

Reserved ⇨ Utilised capacity

Reserved ⇨ Utilised capacity

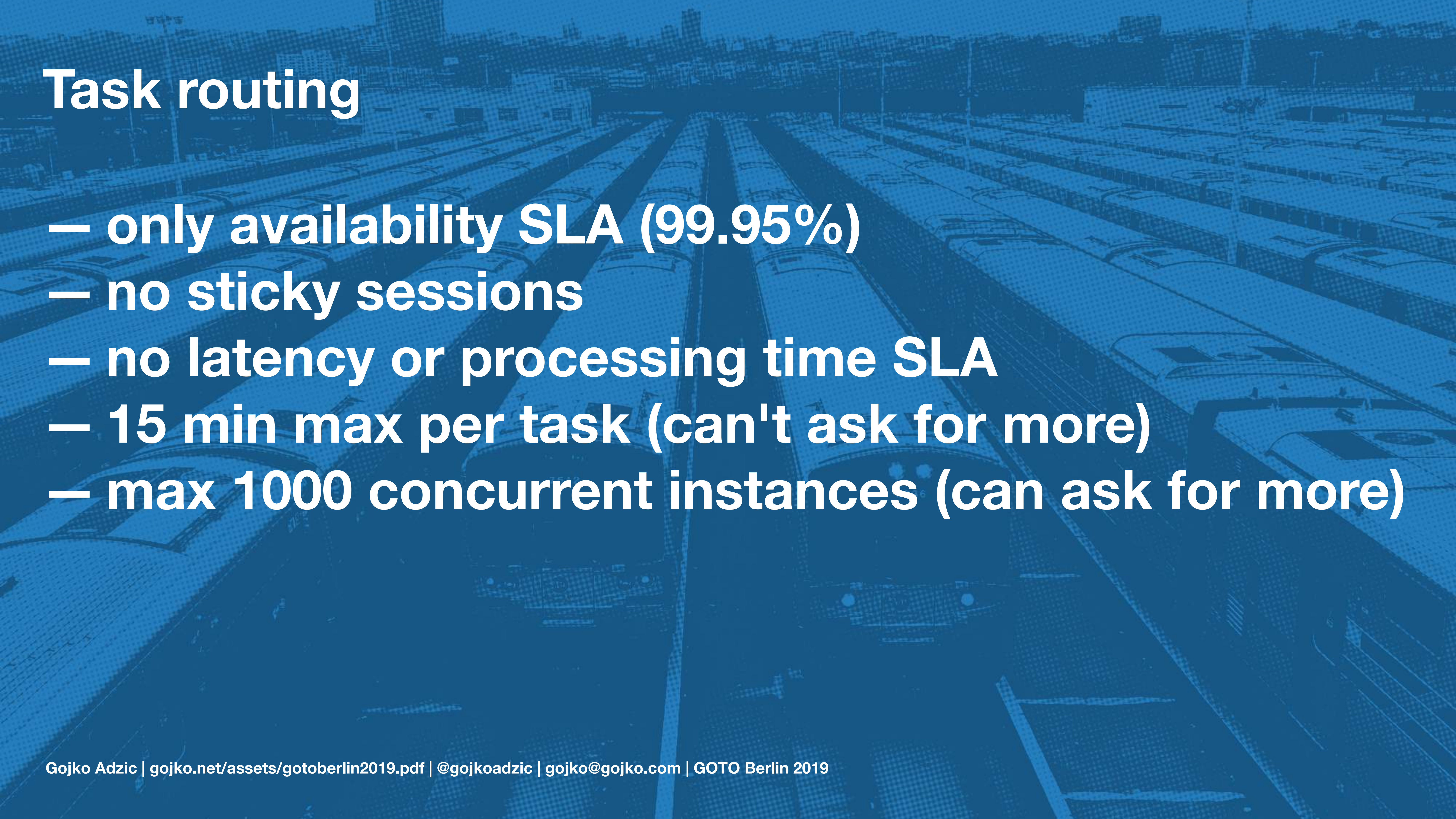
Gegenteilvonflughafenberlinbrandenburg





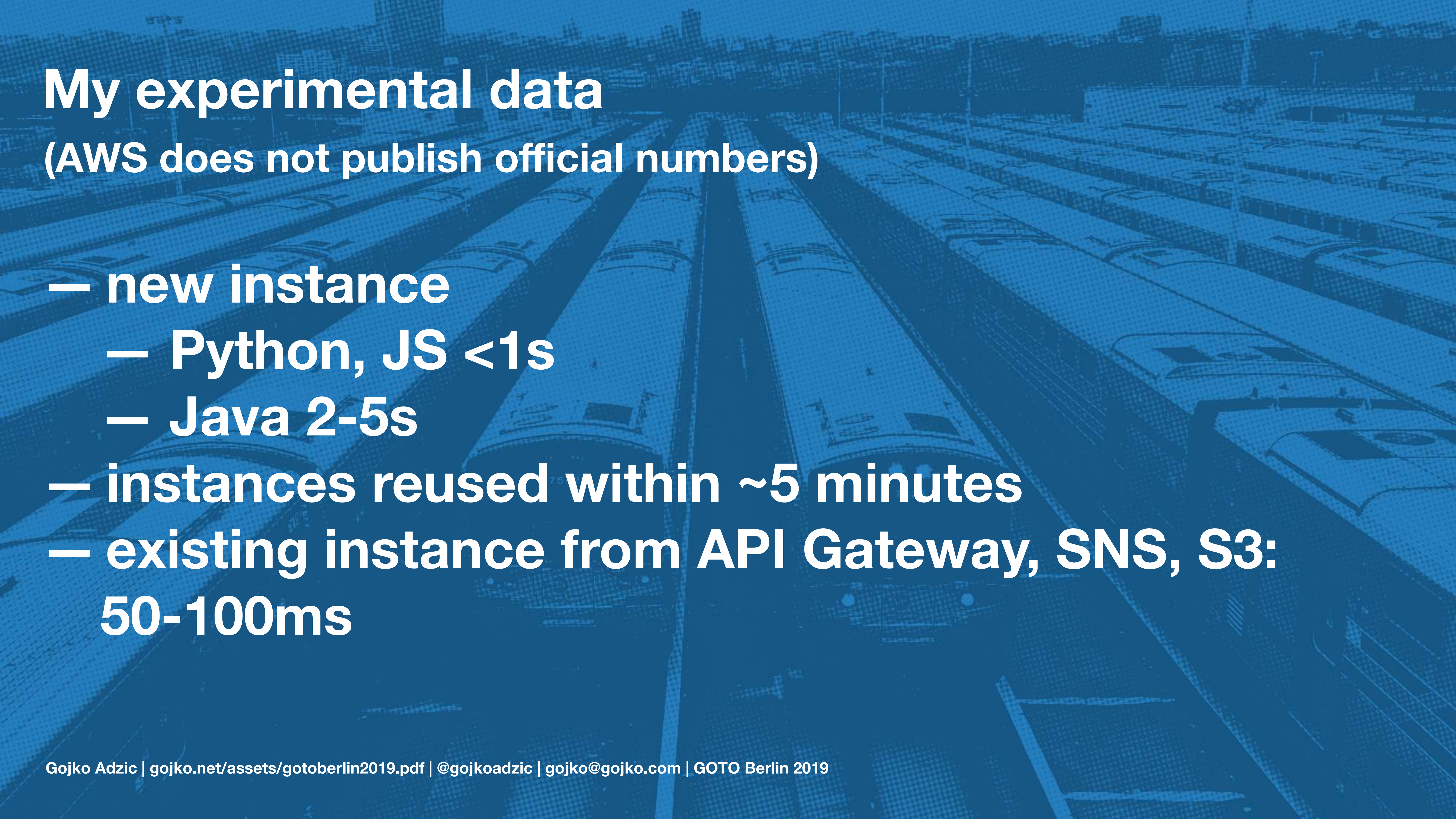
Provider controls instances

— can start/die/get reused or replaced at any point
— optimised for throughput, not latency
— not stateless, but transient



Task routing

- — only availability SLA (99.95%)
- — no sticky sessions
- — no latency or processing time SLA
- — 15 min max per task (can't ask for more)
- — max 1000 concurrent instances (can ask for more)



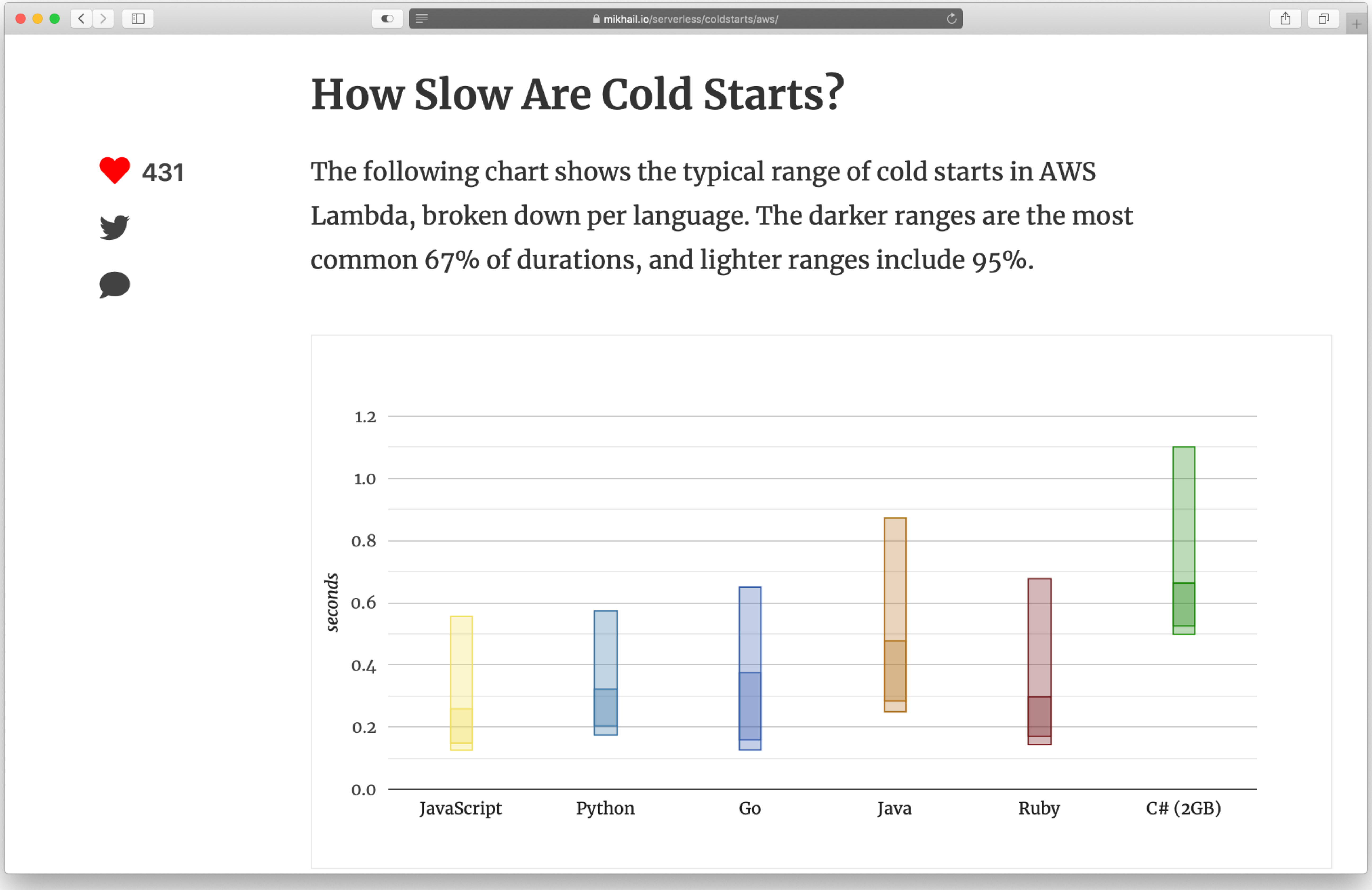
My experimental data

(AWS does not publish official numbers)

— new instance
  - — Python, JS <1s
  - — Java 2-5s

— instances reused within ~5 minutes

— existing instance from API Gateway, SNS, S3: 50-100ms



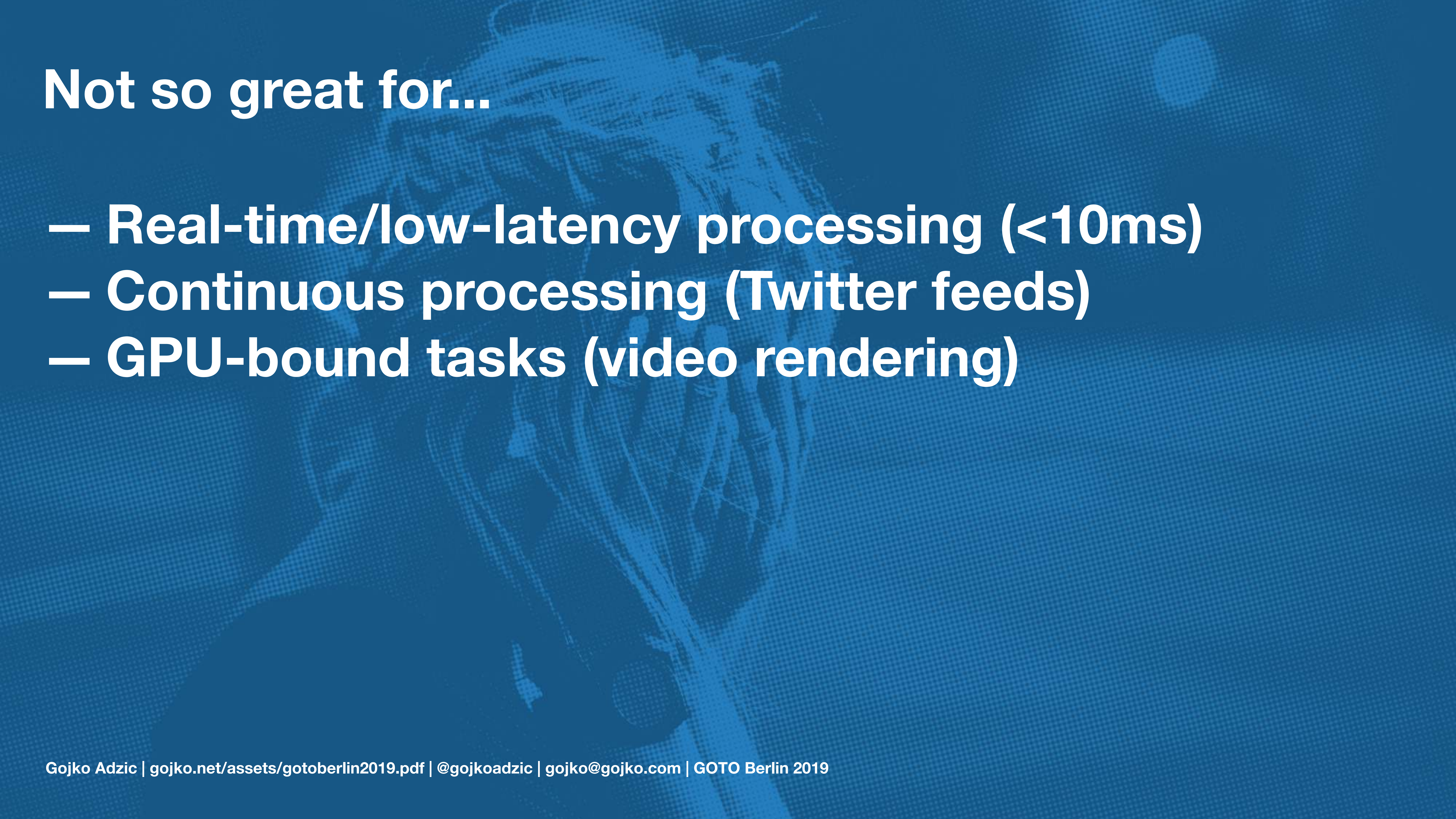
How Slow Are Cold Starts?

The following chart shows the typical range of cold starts in AWS Lambda, broken down per language. The darker ranges are the most common 67% of durations, and lighter ranges include 95%.

Mikhail Shilkov, September 2019, https://mikhail.io/serverless/coldstarts/aws/

Great for...

- HTTP API
- Image conversions
- Payment processing
- Reporting



Not so great for...

— Real-time/low-latency processing (<10ms)
— Continuous processing (Twitter feeds)
— GPU-bound tasks (video rendering)

Optimise for Recovery

⇨

Optimise for Start

Design for:

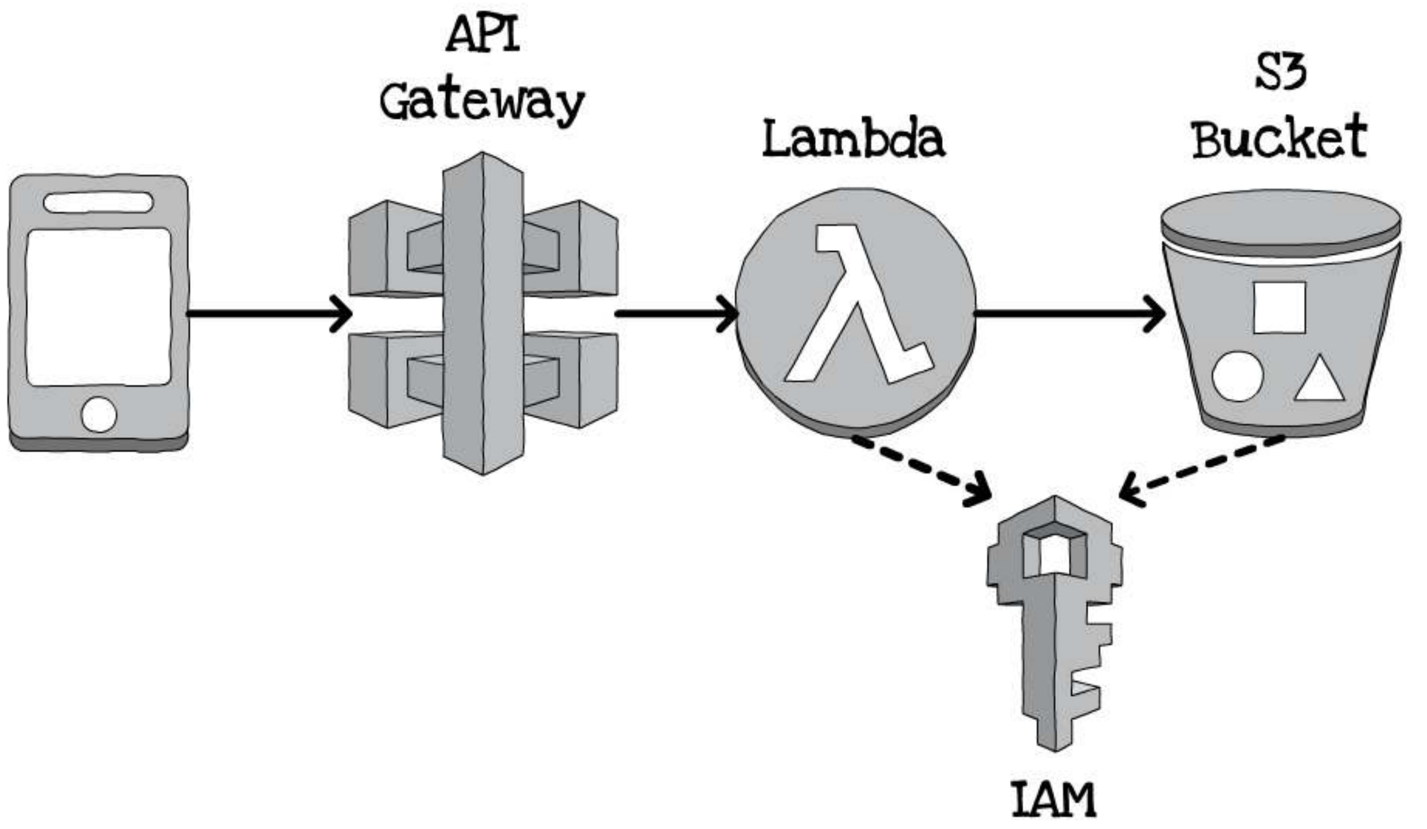
parallelisation

quick data access

processing data aggregates

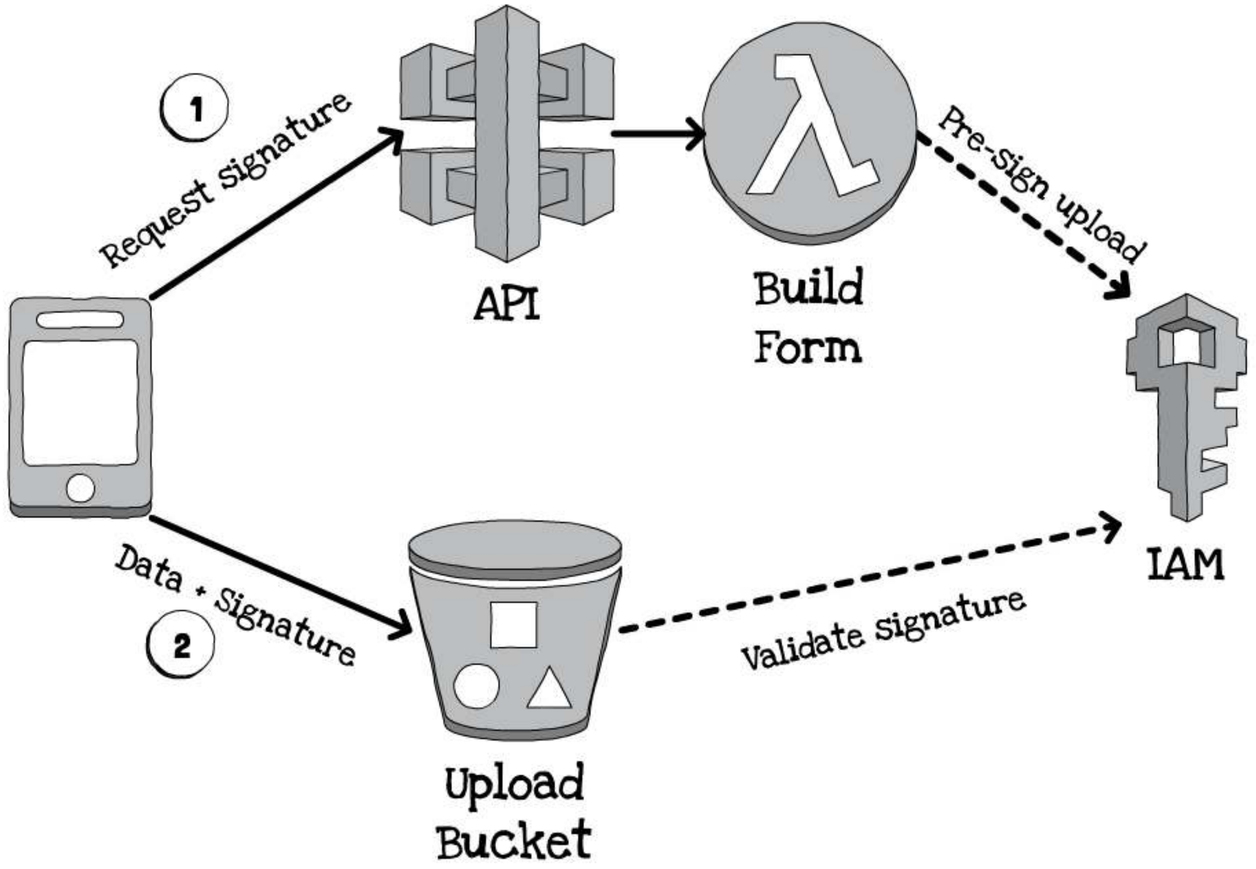
Use the platform for typical server responsibilities

What's it doing here?



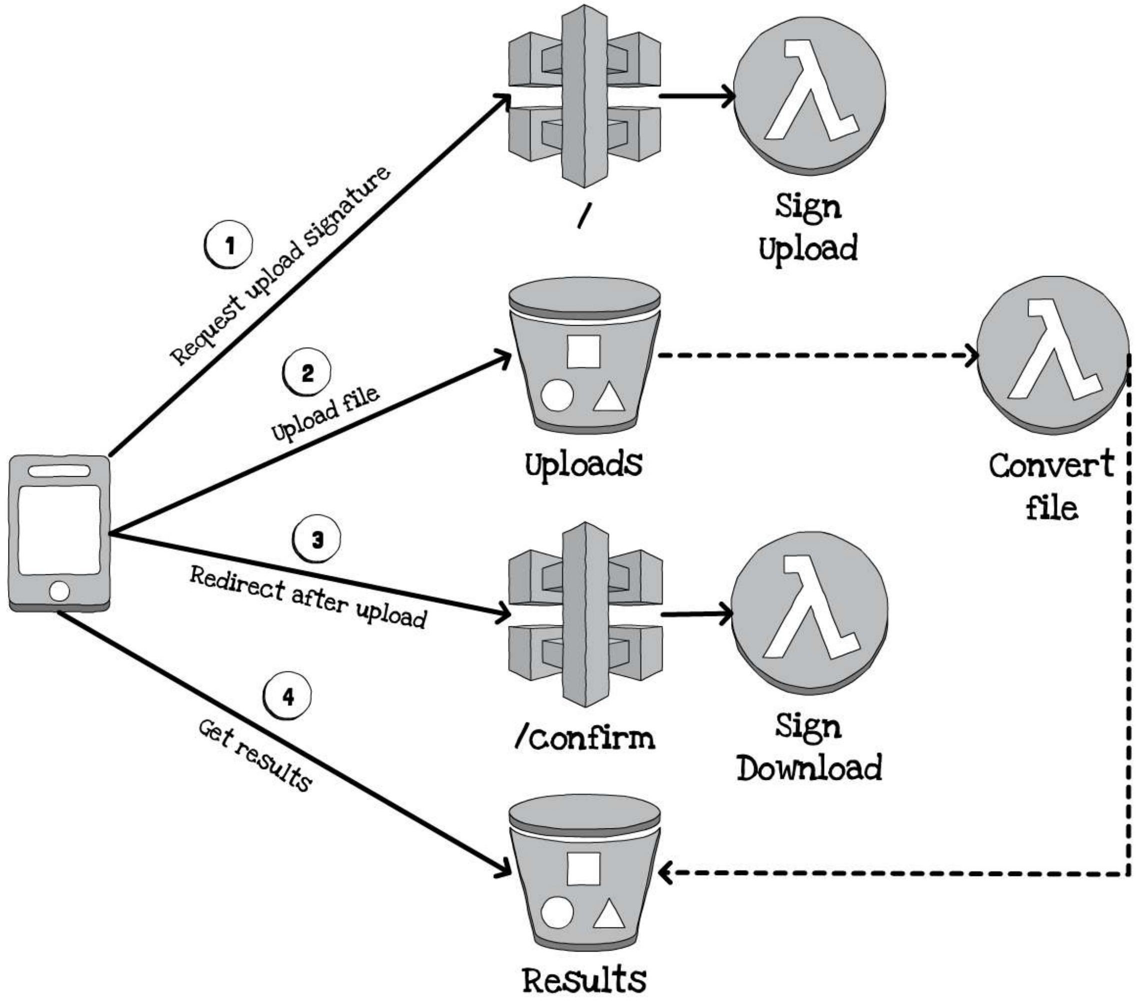
Don't use Lambda as a gatekeeper

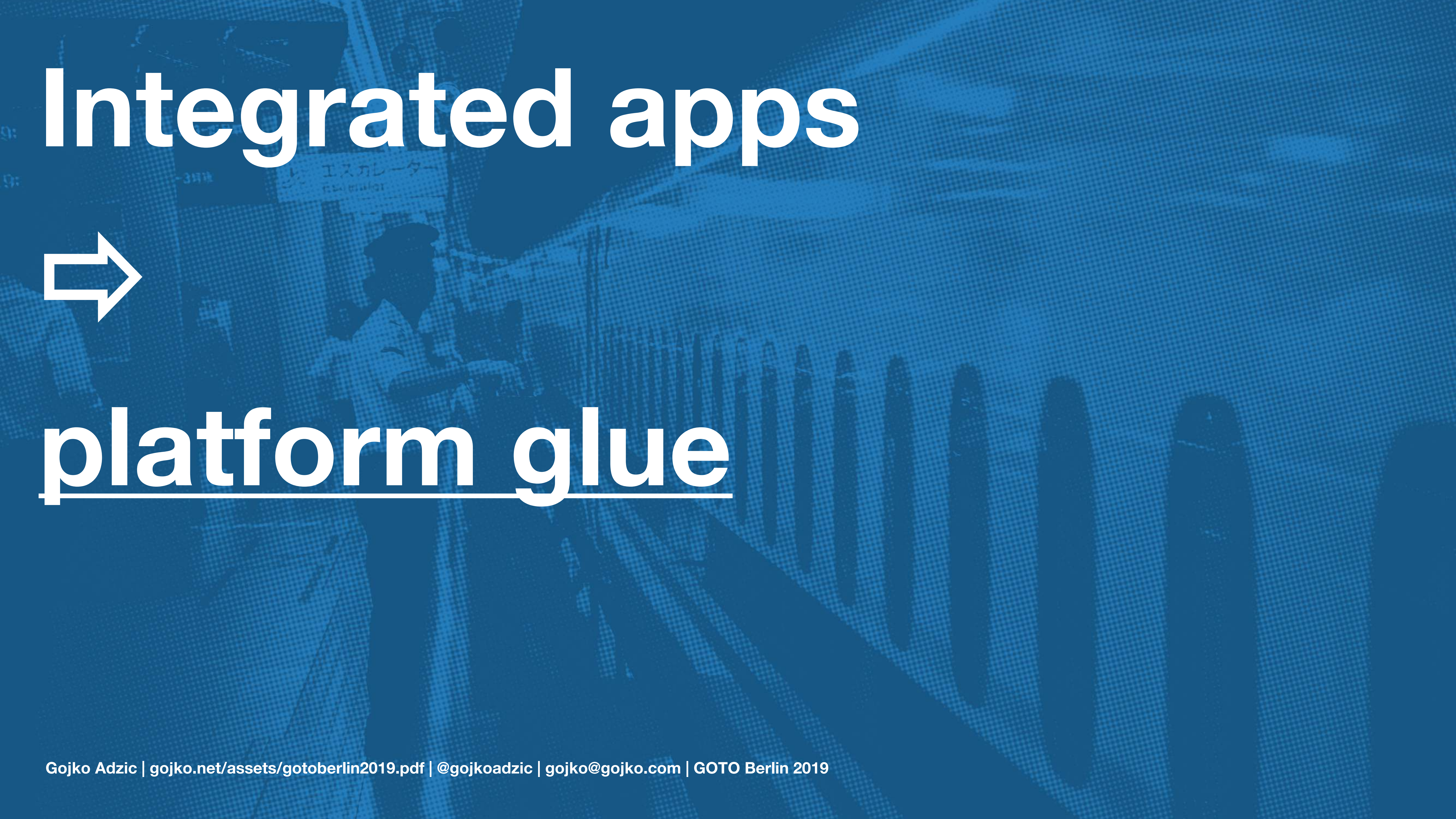
Let client devices talk directly to resources



Don't use Lambda for orchestration

Use platform events + client-side workflows

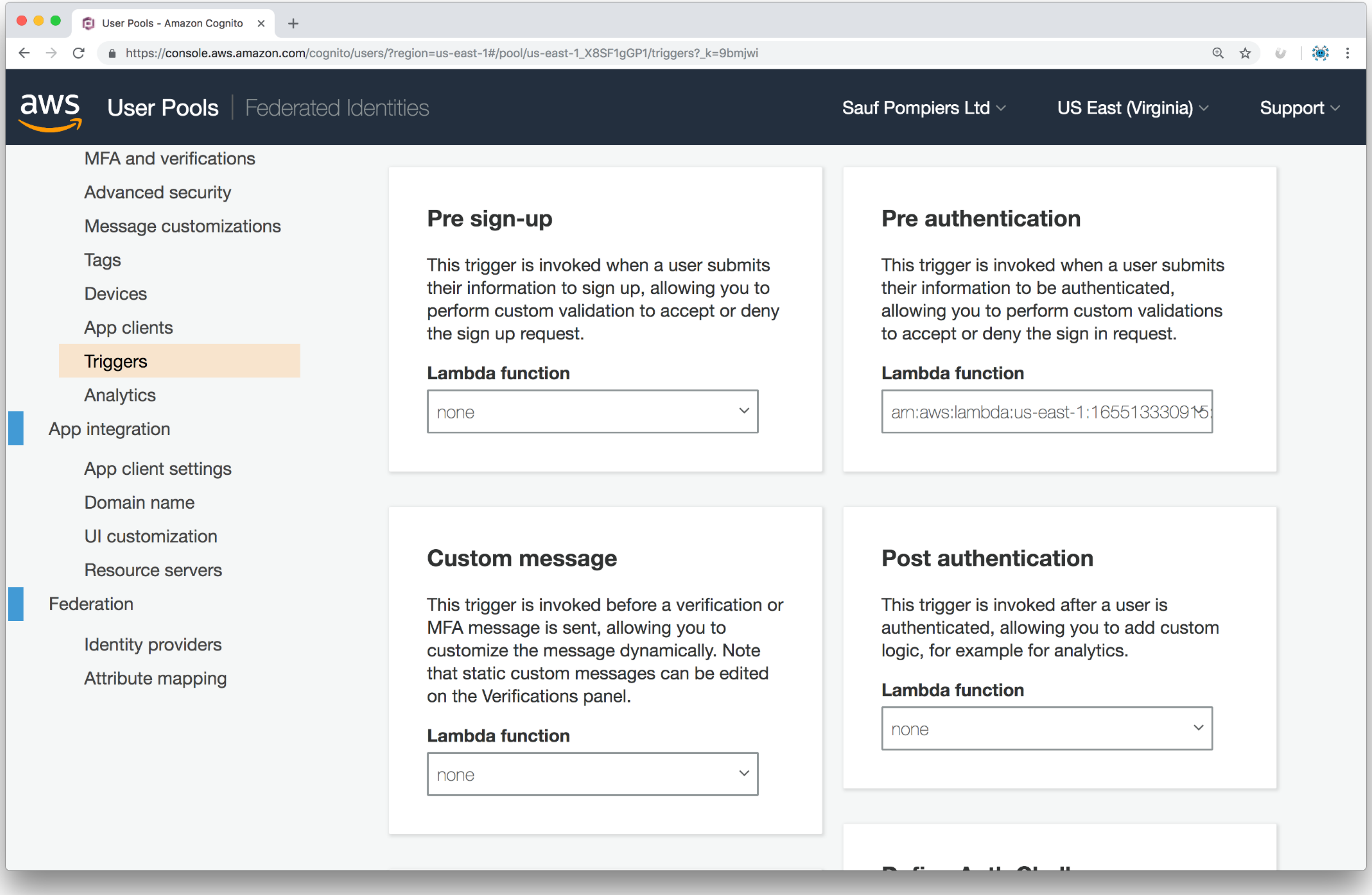




Integrated apps

⇨

platform glue



Pre sign-up

This trigger is invoked when a user submits their information to sign up, allowing you to perform custom validation to accept or deny the sign up request.

Lambda function

none

Pre authentication

This trigger is invoked when a user submits their information to be authenticated, allowing you to perform custom validations to accept or deny the sign in request.

Lambda function

arn:aws:lambda:us-east-1:165513330915:

Custom message

This trigger is invoked before a verification or MFA message is sent, allowing you to customize the message dynamically. Note that static custom messages can be edited on the Verifications panel.

Lambda function

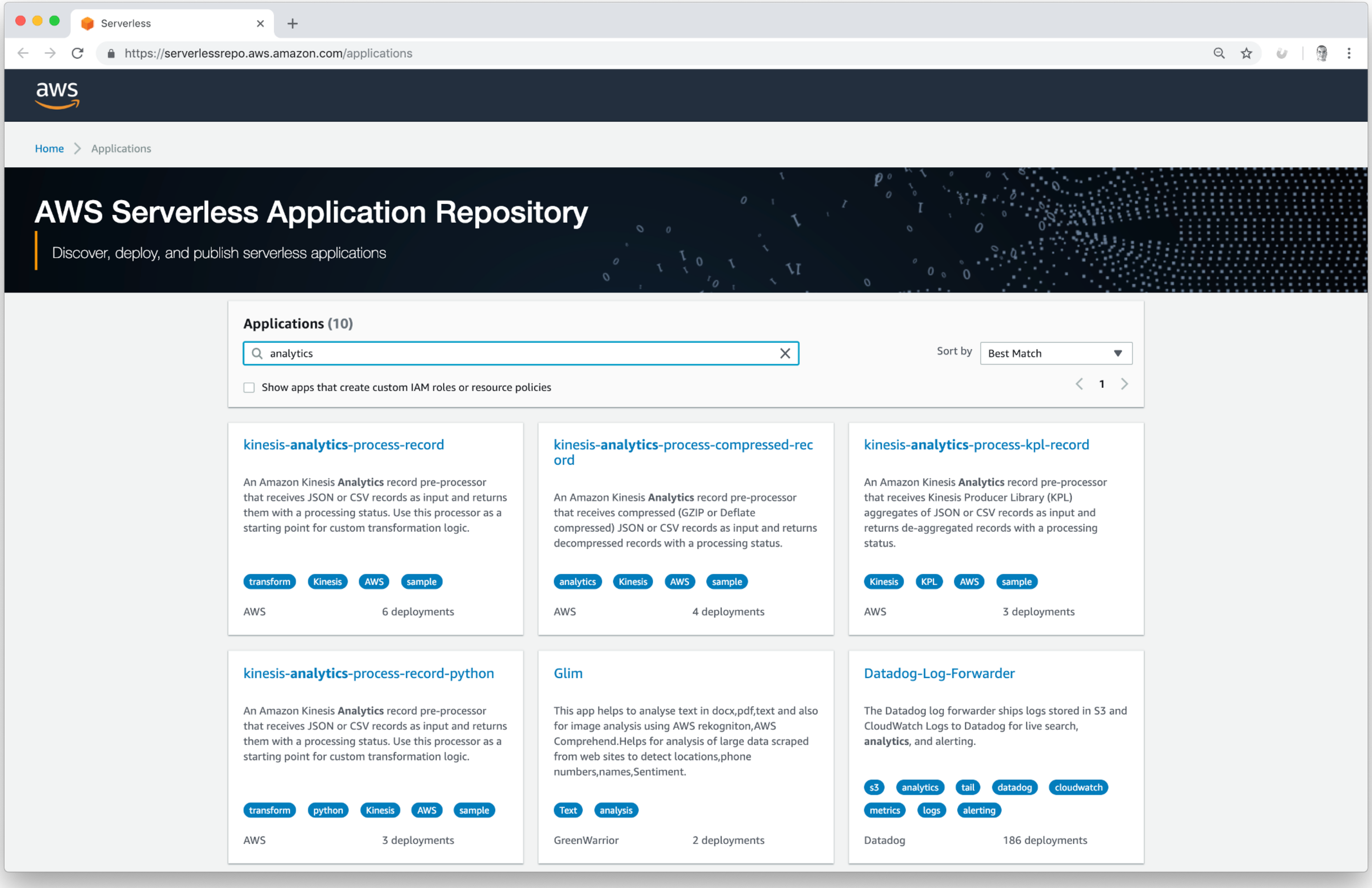
none

Post authentication

This trigger is invoked after a user is authenticated, allowing you to add custom logic, for example for analytics.

Lambda function

none



Home > Applications

AWS Serverless Application Repository

Discover, deploy, and publish serverless applications

Applications (10)

analytics

Sort by Best Match

□ Show apps that create custom IAM roles or resource policies

< 1 >

kinesis-analytics-process-record

An Amazon Kinesis **Analytics** record pre-processor that receives JSON or CSV records as input and returns them with a processing status. Use this processor as a starting point for custom transformation logic.

transform Kinesis AWS sample

AWS 6 deployments

kinesis-analytics-process-compressed-record

An Amazon Kinesis **Analytics** record pre-processor that receives compressed (GZIP or Deflate compressed) JSON or CSV records as input and returns decompressed records with a processing status.

analytics Kinesis AWS sample

AWS 4 deployments

kinesis-analytics-process-kpl-record

An Amazon Kinesis **Analytics** record pre-processor that receives Kinesis Producer Library (KPL) aggregates of JSON or CSV records as input and returns de-aggregated records with a processing status.

Kinesis KPL AWS sample

AWS 3 deployments

kinesis-analytics-process-record-python

An Amazon Kinesis **Analytics** record pre-processor that receives JSON or CSV records as input and returns them with a processing status. Use this processor as a starting point for custom transformation logic.

transform python Kinesis AWS sample

AWS 3 deployments

Glim

This app helps to analyse text in docx,pdf,text and also for image analysis using AWS rekogniton,AWS Comprehend.Helps for analysis of large data scraped from web sites to detect locations,phone numbers,names,Sentiment.

Text analysis

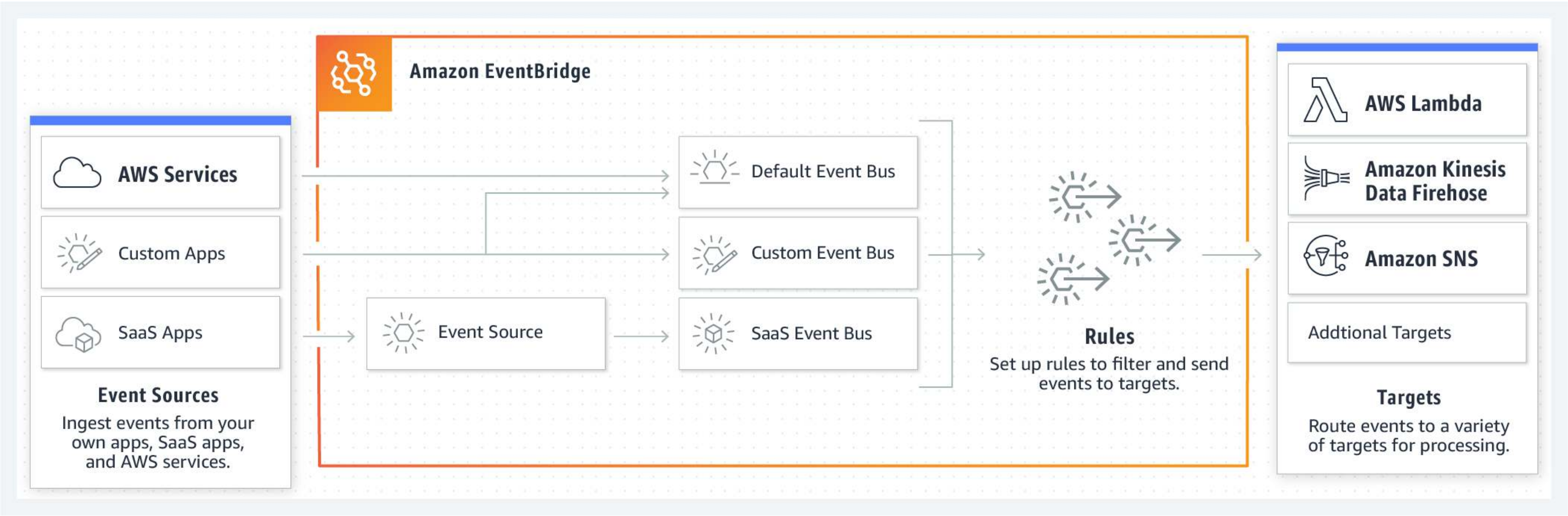
GreenWarrior 2 deployments

Datadog-Log-Forwarder

The Datadog log forwarder ships logs stored in S3 and CloudWatch Logs to Datadog for live search, **analytics**, and alerting.

s3 analytics tail datadog cloudwatch metrics logs alerting

Datadog 186 deployments



Amazon EventBridge

AWS Services

Custom Apps

SaaS Apps

Event Sources

Ingest events from your own apps, SaaS apps, and AWS services.

Default Event Bus

Custom Event Bus

Event Source

SaaS Event Bus

Rules

Set up rules to filter and send events to targets.

AWS Lambda

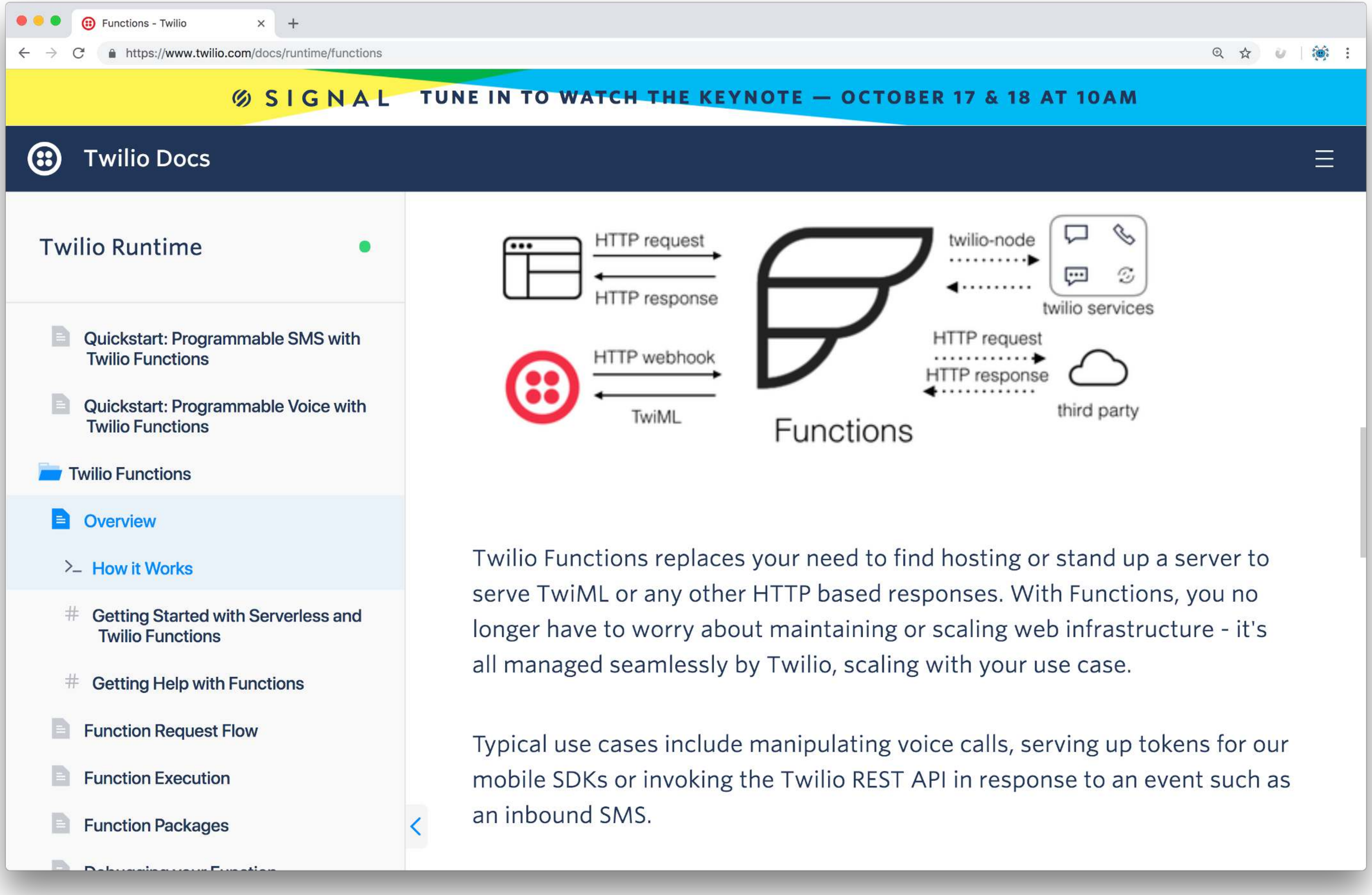
Amazon Kinesis Data Firehose

Amazon SNS

Addtional Targets

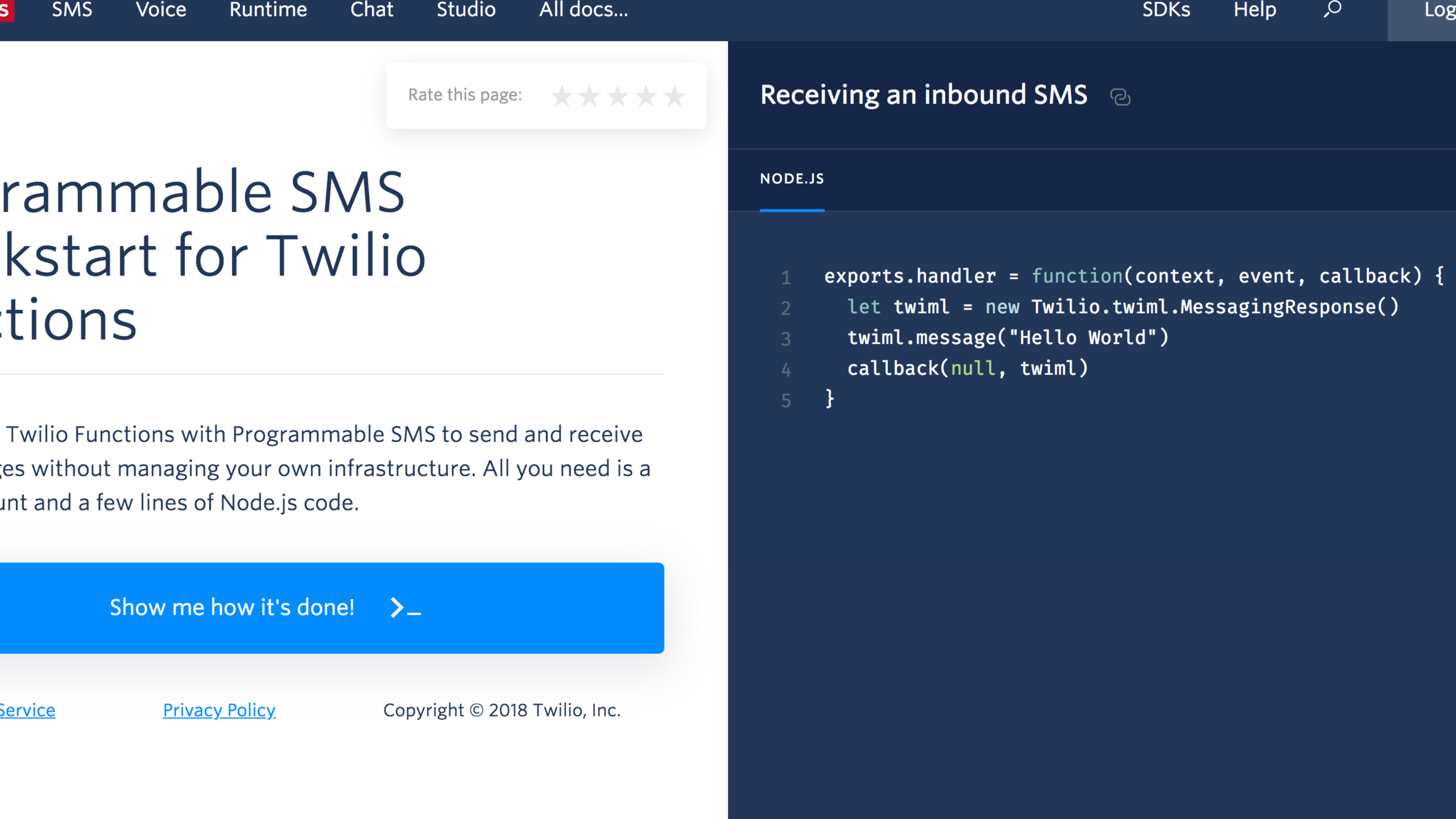
Targets

Route events to a variety of targets for processing.


Functions - Twilio
https://www.twilio.com/docs/runtime/functions
SIGNAL TUNE IN TO WATCH THE KEYNOTE — OCTOBER 17 & 18 AT 10AM
Twilio Docs
Twilio Runtime
Quickstart: Programmable SMS with Twilio Functions
Quickstart: Programmable Voice with Twilio Functions
Twilio Functions
Overview
How it Works
Getting Started with Serverless and Twilio Functions
Getting Help with Functions
Function Request Flow
Function Execution
Function Packages
Debugging your Function
HTTP request
HTTP response
HTTP webhook
TwiML
Functions
twilio-node
twilio services
HTTP request
HTTP response
third party

Twilio Functions replaces your need to find hosting or stand up a server to serve TwiML or any other HTTP based responses. With Functions, you no longer have to worry about maintaining or scaling web infrastructure - it's all managed seamlessly by Twilio, scaling with your use case.

Typical use cases include manipulating voice calls, serving up tokens for our mobile SDKs or invoking the Twilio REST API in response to an event such as an inbound SMS.



rammable SMS kstart for Twilio tions

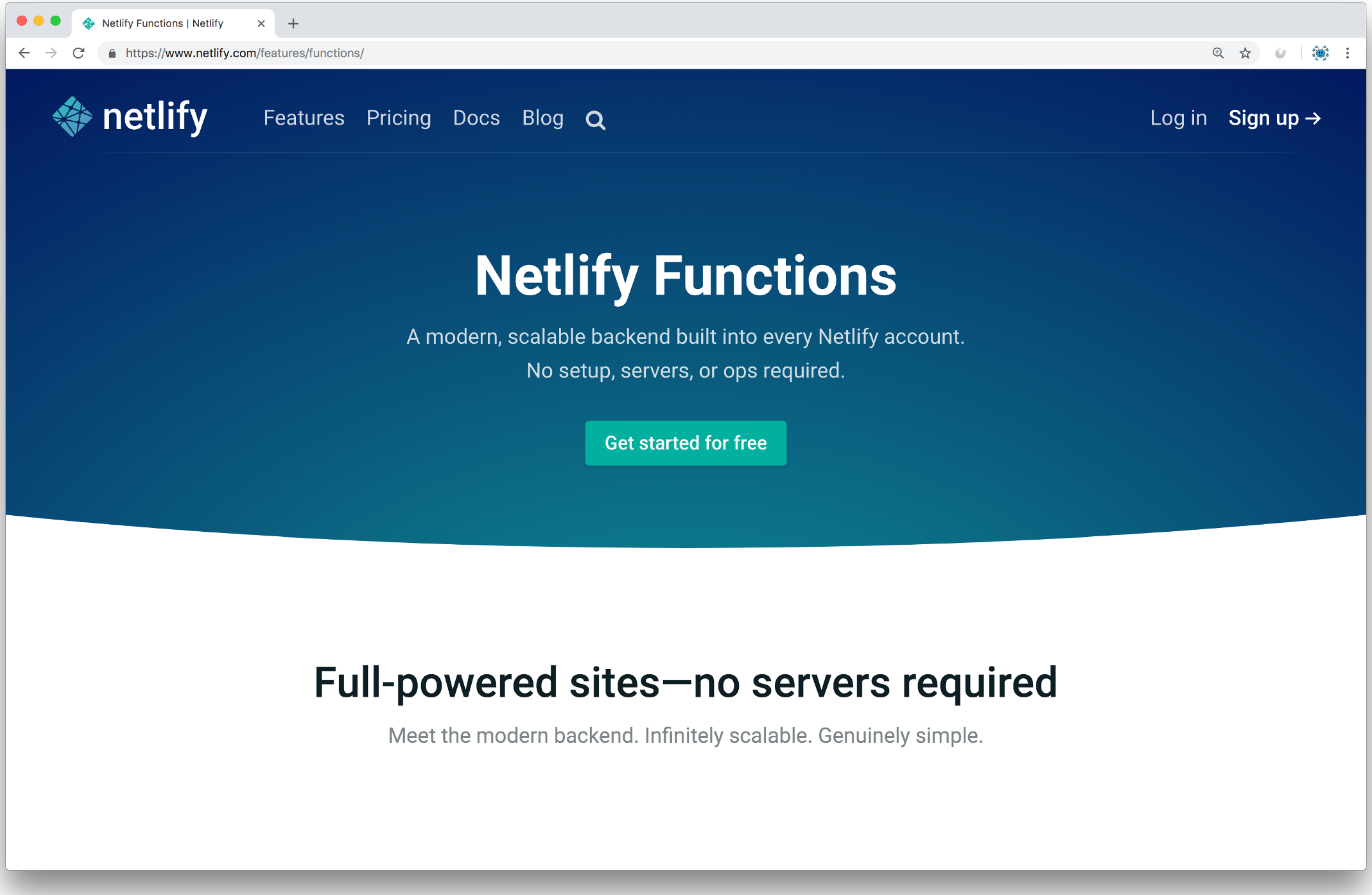
Twilio Functions with Programmable SMS to send and receive ges without managing your own infrastructure. All you need is a unt and a few lines of Node.js code.

Service Privacy Policy

Receiving an inbound SMS

NODE.JS

```
exports.handler = function(context, event, callback) {
  let twiml = new Twilio.twiml.MessagingResponse()
  twiml.message("Hello World")
  callback(null, twiml)
}
```



Netlify Functions | Netlify
https://www.netlify.com/features/functions/
netlify
Features
Pricing
Docs
Blog
Log in
Sign up →
Get started for free

Netlify Functions

A modern, scalable backend built into every Netlify account.
No setup, servers, or ops required.

Full-powered sites—no servers required

Meet the modern backend. Infinitely scalable. Genuinely simple.

called from an address relative to the deployed site root: `/.netlify/functions/{function_name}`. You can also set a func

be triggered by certain Netlify events.

THE HANDLER METHOD

Each JavaScript file to be deployed as a Lambda function must ex

`handler` method with the following general syntax:

```
exports.handler = function(event, context, callback) {
    // your server-side functionality
}
```

Netlify provides the `event` and `context` parameters when the fu

Serverless functions will replace webhooks

Webhakenzerstörer

RUNNING
SERVERLESS
Gojko Adzic
introduction to aws lambda and the serverless application model

RUNNING SERVERLESS

Gojko Adzic

https://leanpub.com/running-serverless/c/gotober

50% off for the next 24 hours