SERVERLESS

apps without infrastructure

Alexander Reelsen

@spinscale

alr@spinscale.de

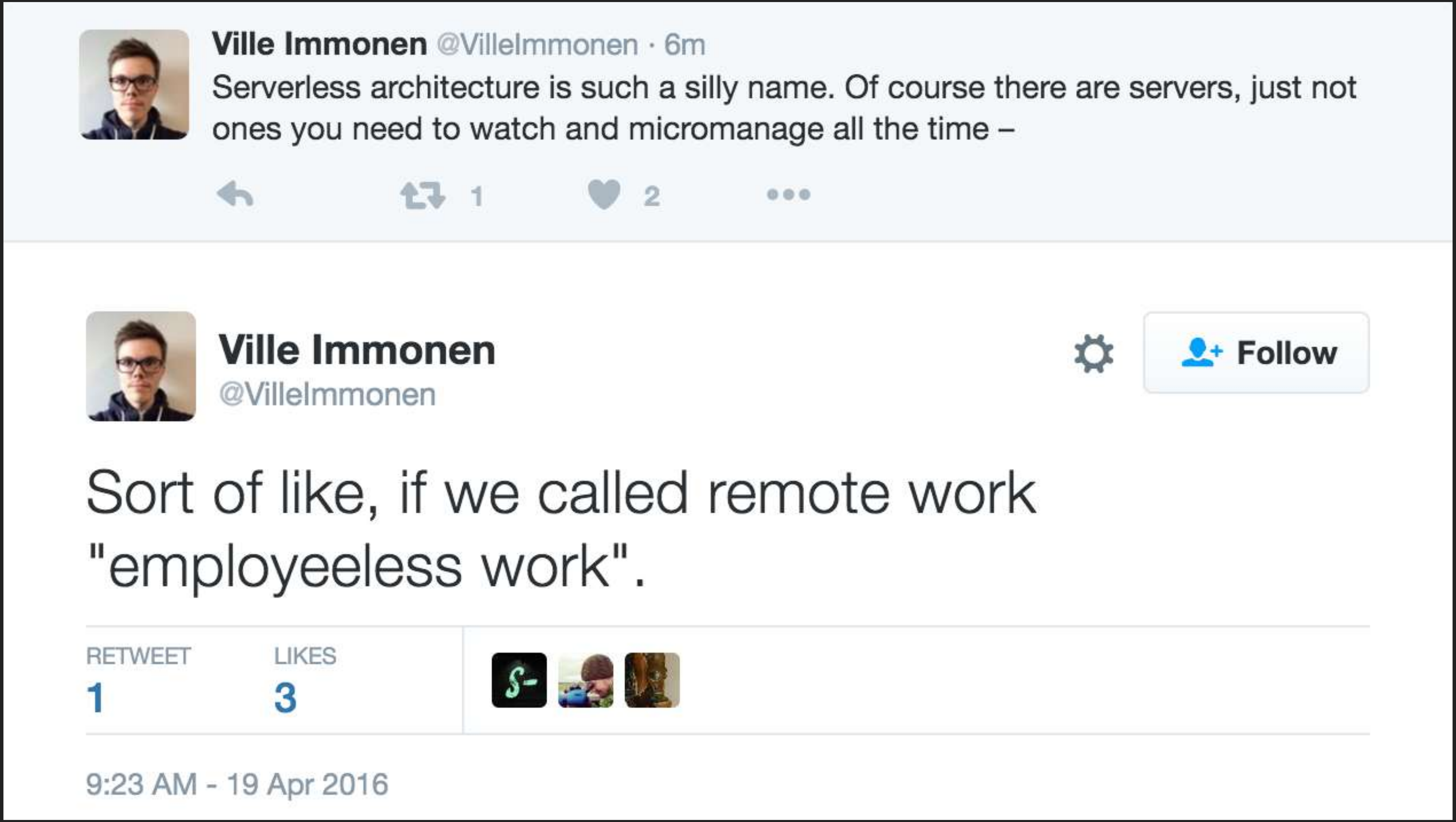
<<MARKETING>>

<<lie, damned lies and it terminology>>

Alexander Reelsen

@spinscale

alr@spinscale.de



Ville Immonen @VilleImmonen · 6m

Serverless architecture is such a silly name. Of course there are servers, just not ones you need to watch and micromanage all the time –

1 2

Ville Immonen
@VilleImmonen

Follow

Sort of like, if we called remote work "employeeless work".

RETWEET 1 LIKES 3

9:23 AM - 19 Apr 2016

ABOUT ME

- Java developer by day (Elasticsearch + commercial extensions)
- 'Whatever looks interesting' developer by night
- Interested in Basketball, Linux, JVM, scalability, node, command line apps, keyboard shortcuts and productivity
- Likes tech meetups, organizing Search Meetup Munich, did devcampmuc unconference

SERVERLESS

apps without infrastructure

Alexander Reelsen

@spinscale

alr@spinscale.de

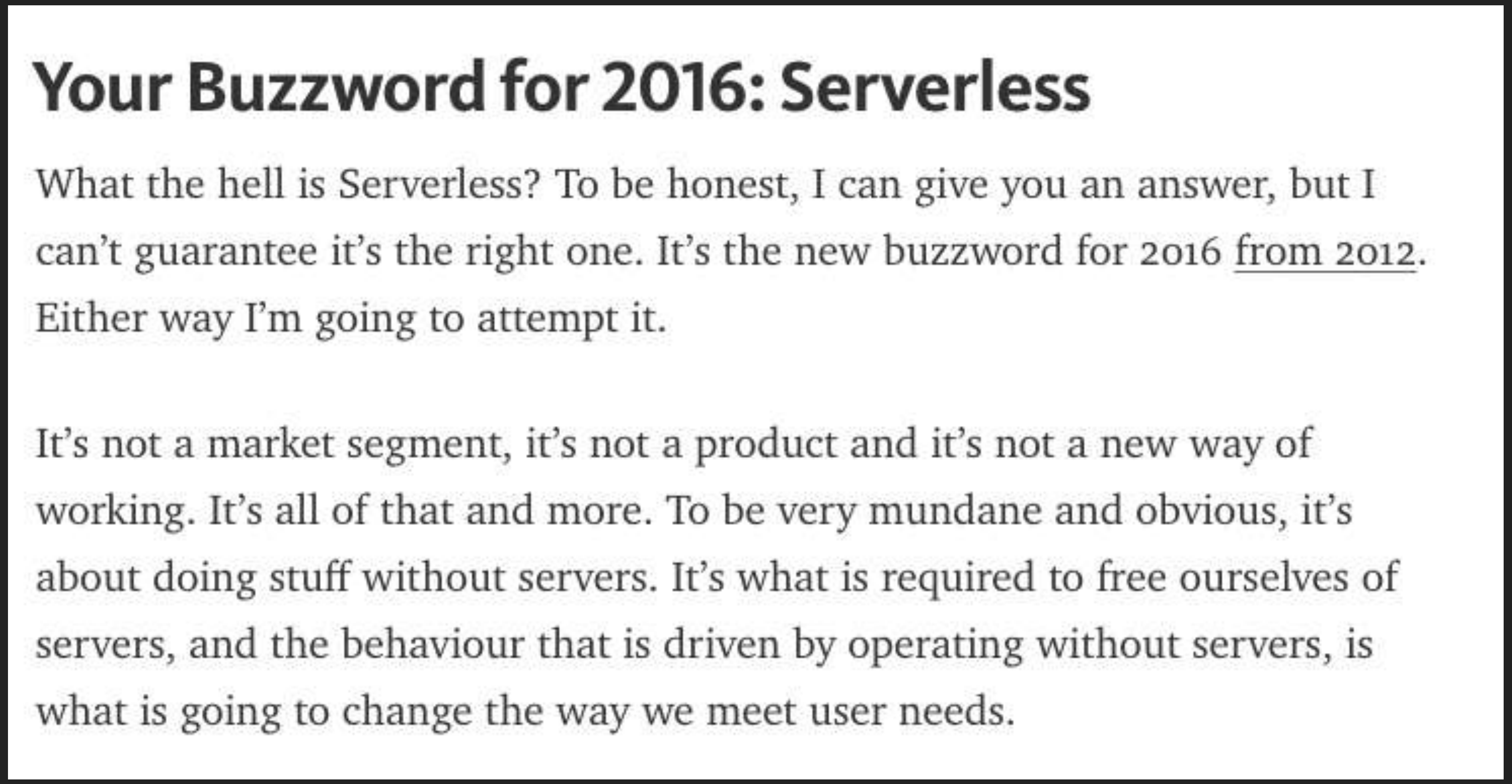
THE TRUTH[TM]

THERE IS NO SERVERLESS ARCHITECTURE

IT'S JUST SOMEONE ELSE'S EXECUTION ENVIRONMENT

Alex R.

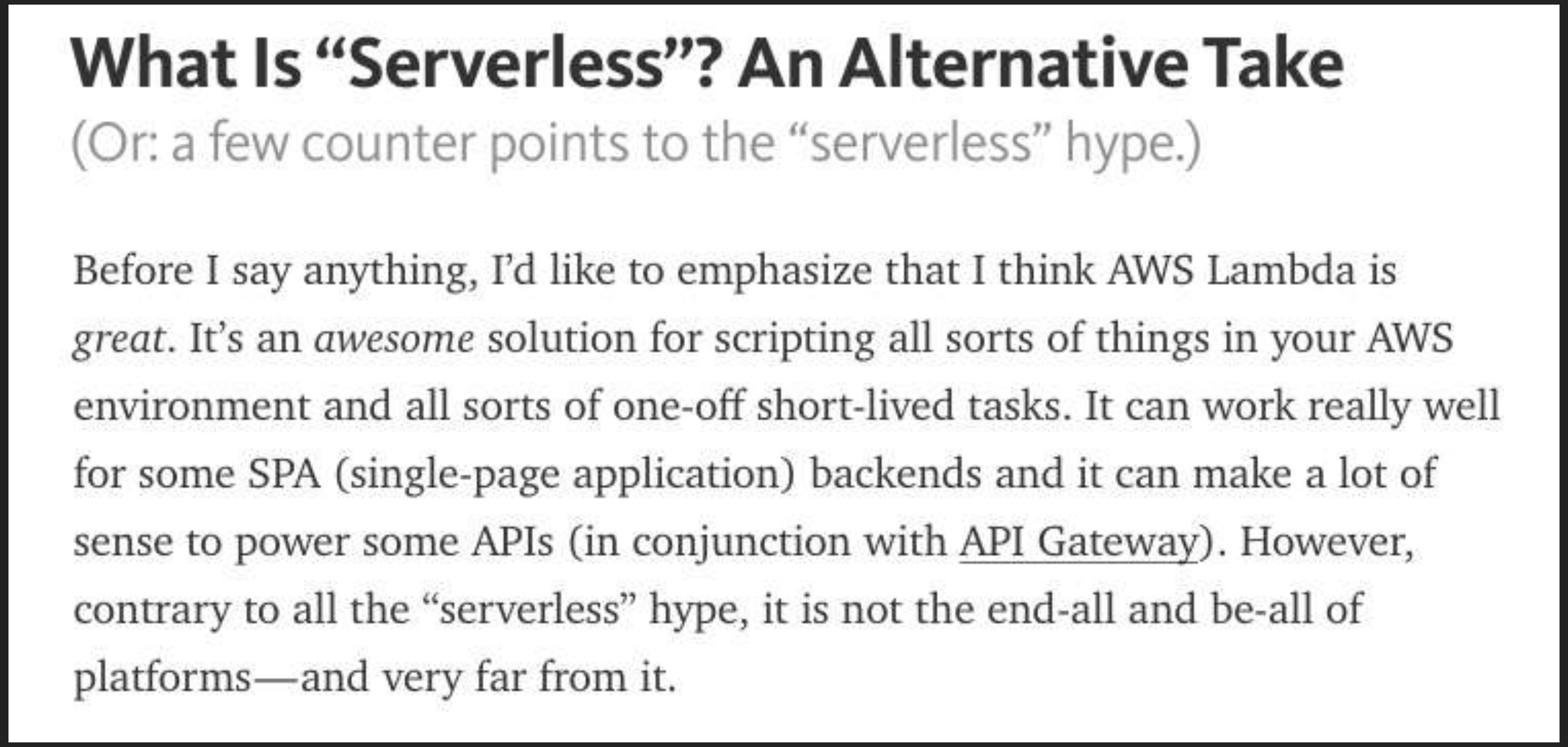
WAT?



Your Buzzword for 2016: Serverless

What the hell is Serverless? To be honest, I can give you an answer, but I can't guarantee it's the right one. It's the new buzzword for 2016 from 2012. Either way I'm going to attempt it.

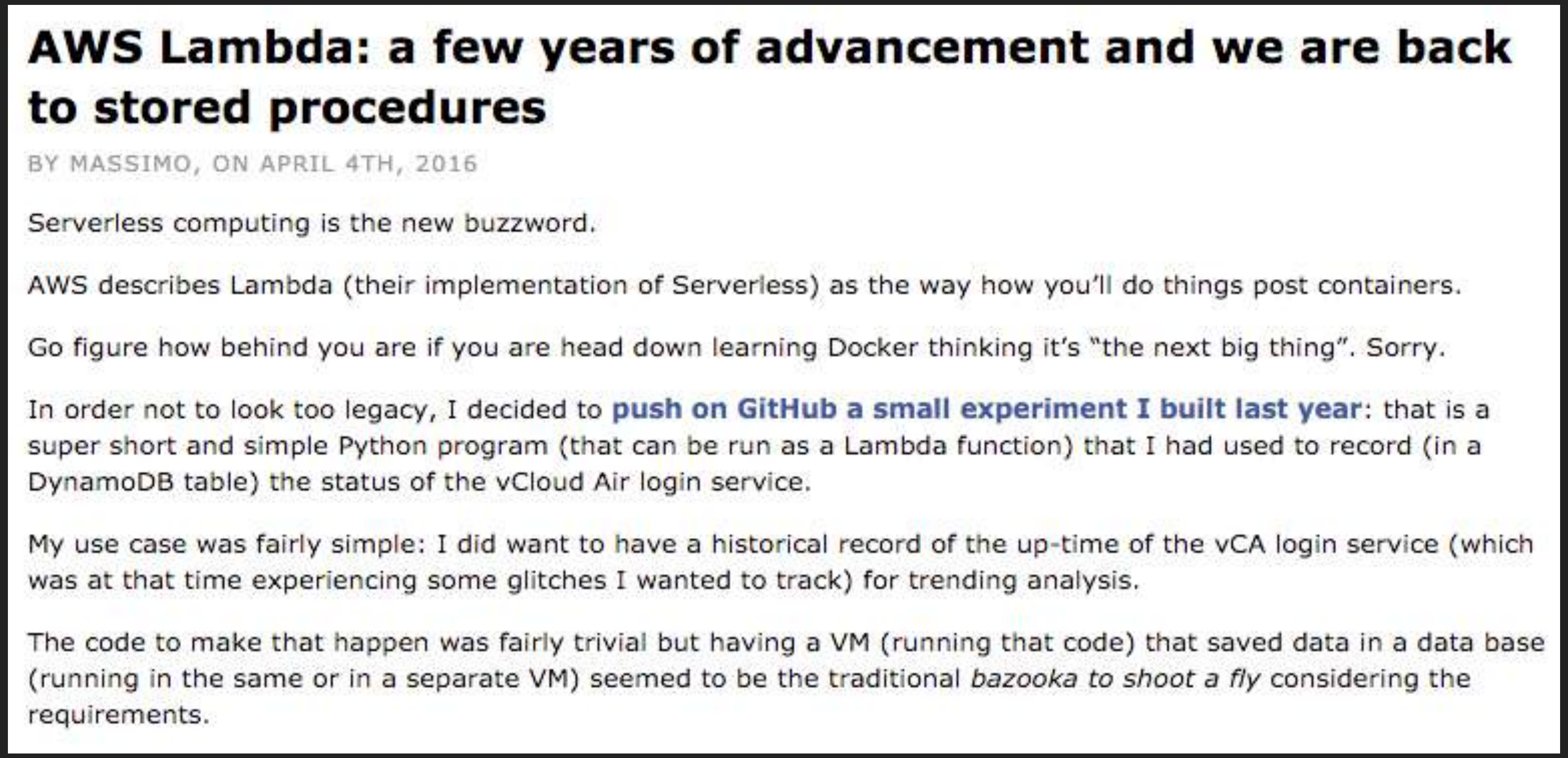
It's not a market segment, it's not a product and it's not a new way of working. It's all of that and more. To be very mundane and obvious, it's about doing stuff without servers. It's what is required to free ourselves of servers, and the behaviour that is driven by operating without servers, is what is going to change the way we meet user needs.



What Is "Serverless"? An Alternative Take

(Or: a few counter points to the "serverless" hype.)

Before I say anything, I'd like to emphasize that I think AWS Lambda is *great*. It's an *awesome* solution for scripting all sorts of things in your AWS environment and all sorts of one-off short-lived tasks. It can work really well for some SPA (single-page application) backends and it can make a lot of sense to power some APIs (in conjunction with API Gateway). However, contrary to all the "serverless" hype, it is not the end-all and be-all of platforms—and very far from it.



AWS Lambda: a few years of advancement and we are back to stored procedures

BY MASSIMO, ON APRIL 4TH, 2016

Serverless computing is the new buzzword.

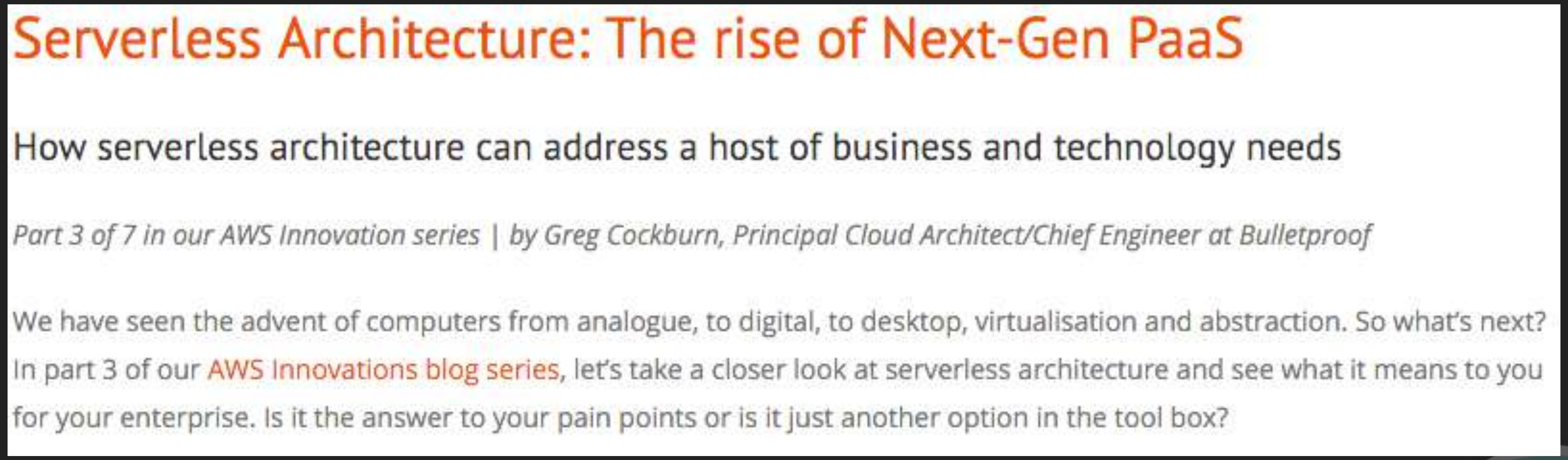
AWS describes Lambda (their implementation of Serverless) as the way how you'll do things post containers.

Go figure how behind you are if you are head down learning Docker thinking it's "the next big thing". Sorry.

In order not to look too legacy, I decided to **push on GitHub a small experiment I built last year**: that is a super short and simple Python program (that can be run as a Lambda function) that I had used to record (in a DynamoDB table) the status of the vCloud Air login service.

My use case was fairly simple: I did want to have a historical record of the up-time of the vCA login service (which was at that time experiencing some glitches I wanted to track) for trending analysis.

The code to make that happen was fairly trivial but having a VM (running that code) that saved data in a data base (running in the same or in a separate VM) seemed to be the traditional *bazooka to shoot a fly* considering the requirements.



Serverless Architecture: The rise of Next-Gen PaaS

How serverless architecture can address a host of business and technology needs

Part 3 of 7 in our AWS Innovation series | by Greg Cockburn, Principal Cloud Architect/Chief Engineer at Bulletproof

We have seen the advent of computers from analogue, to digital, to desktop, virtualisation and abstraction. So what's next? In part 3 of our AWS Innovations blog series, let's take a closer look at serverless architecture and see what it means to you for your enterprise. Is it the answer to your pain points or is it just another option in the tool box?

There is no cloud
it's just someone else's computer

WHAT IS SERVERLESS ARCHITECTURE? RUNNING WITH...

- No bare metal server
- No virtual machine
- No operating system
- No state

- Event-Triggered Language Specific Single Process Execution Environment

SHOW ME THE CODE!

Input

Environment

```
exports.handler = function(event, context) {
   console.log("value1 = " + event.key1)
   console.log("value2 = " + event.key2)

  if (event.key1 === 'undefined') {
     context.fail('missing event key')
   } else {
     context.succeed('some message');
  }
}
```

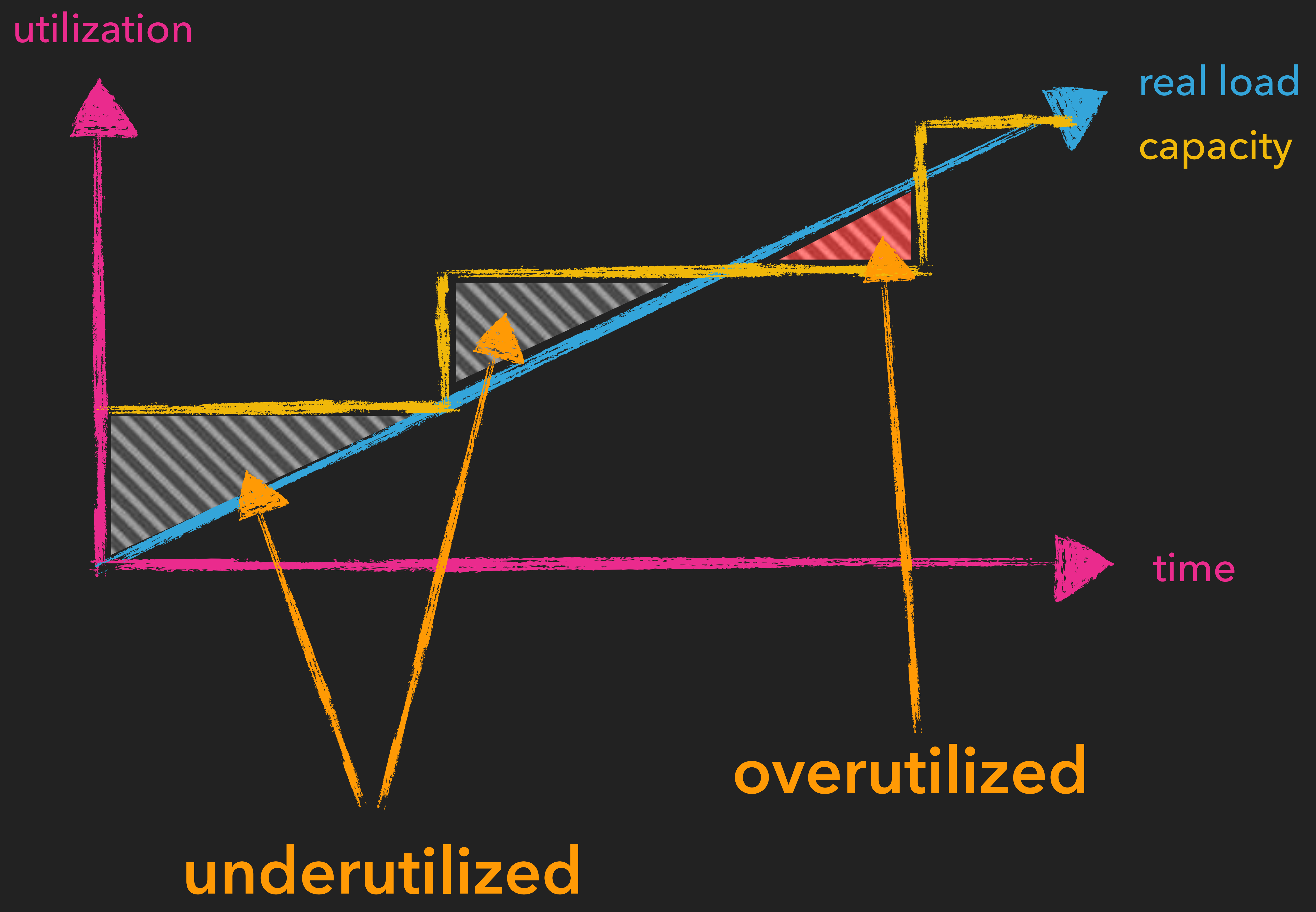
WORKFLOW

- Write code
- Unit test code
- Publish code
- Integration test code
- Go live

WHY?

BUT WHY?

- Scalability
- Setup
- Cost
  - Initial setup
  - Ongoing fixed fee
  - Pay per use

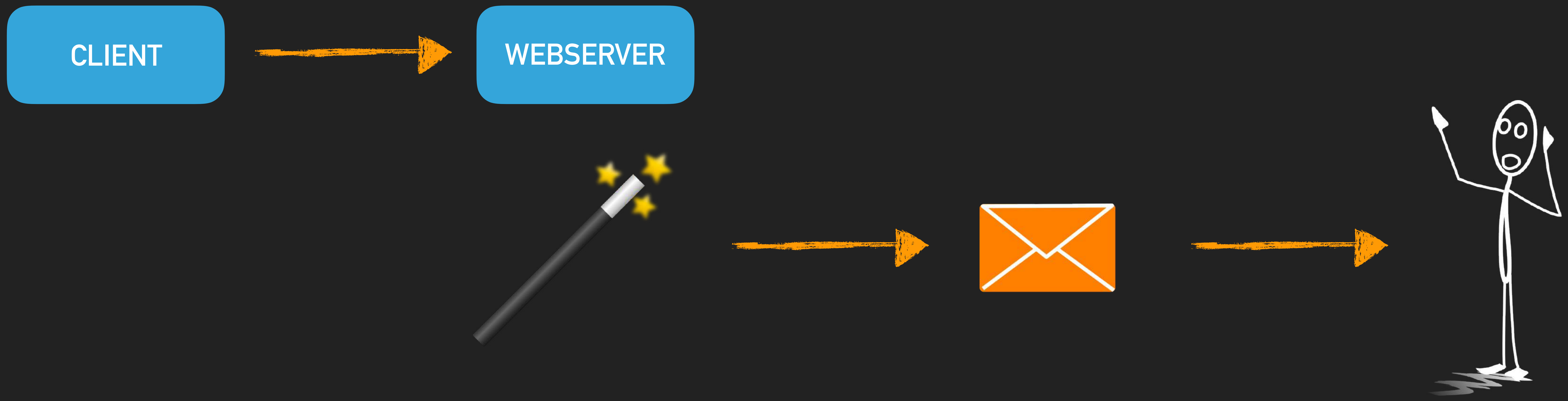


WHAT?

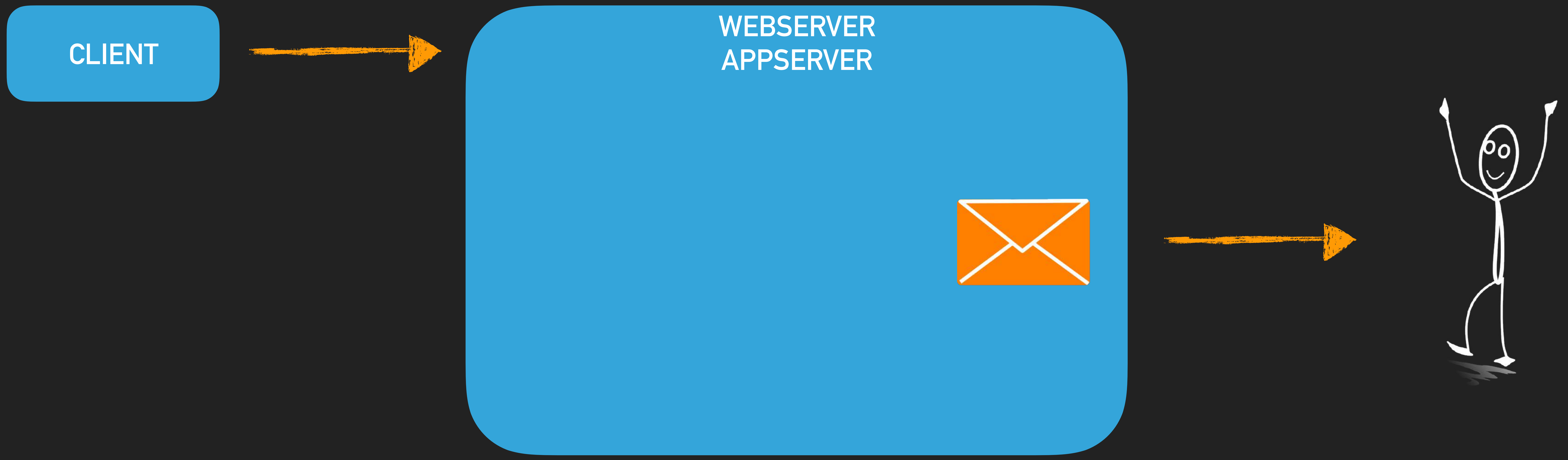
USE-CASES

- Webhooks
- Scheduled Tasks
- Triggers inside of the infrastructure provider
- CI build notifications
- Load testing

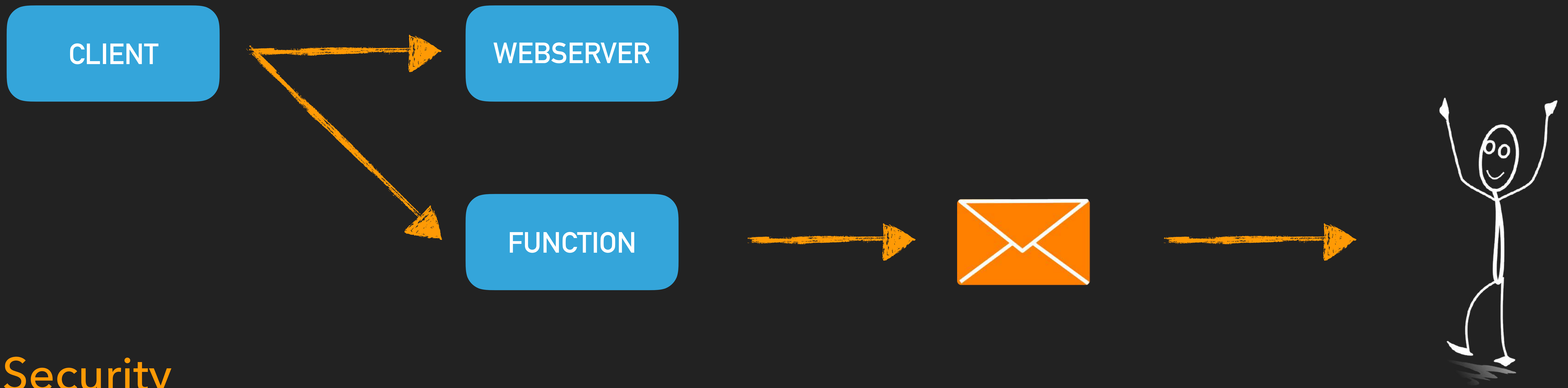
STATIC WEBSITE WITH DYNAMIC ELEMENTS



STATIC WEBSITE WITH DYNAMIC ELEMENTS



STATIC WEBSITE WITH DYNAMIC ELEMENTS



- Security
- Resources
- Limitations

IMPLEMENTATIONS

- AWS Lambda
- Google Cloud Functions
- Azure Functions
- openwhisk
- webtask.io
- iron.io
- Firebase

AWS LAMBDA

- Execution time limit: 5min
- 100 parallel executions
- Languages: Javascript, Java, python
- HTTP invocation: API Gateway
- Logs: Cloudwatch
- Security: IAM

- Versions & Aliases
- Max body payload: 6 MB
- Max size zip: 50 MB
- Max size package: 250 MB
- Max size all lambdas: 75 GB
- Node v0.10.36 + 4.3

AWS LAMBDA + AWS API GATEWAY

- Map lambdas to HTTP endpoints
- Authorization
- SDK Generation
- Monitoring
- Third party API keys

HOW?

AWSCLI

AWSCLI

```
zip -r $function.zip $function.js node_modules
```

AWSCLI

```
aws iam create-role \
  --role-name "$lambda_execution_role_name" \
  --assume-role-policy-document '{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "",
          "Effect": "Allow",
          "Principal": {
            "Service": "lambda.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
```

zip

AWSCLI

```
aws iam put-role-policy \
  --role-name "$lambda_execution_role_name" \
  --policy-name "$lambda_execution_access_policy_name" \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [ "logs:*" ],
        "Resource": "arn:aws:logs:*:*:*"
      },
…
}'
```

```
zip
```

```
aws iam create-role
```

AWSCLI

```
aws lambda upload-function \
  --function-name "$function" \
  --function-zip "$function.zip" \
  --role "$lambda_execution_role_arn" \
  --mode event \
  --handler "$function.handler" \
  --timeout 30 \
  --runtime nodejs
```

```
zip
```

```
aws iam create-role
```

```
aws iam put-role-policy
```

AWSCLI

```
aws lambda invoke-async \
  --function-name "$function" \
  --invoke-args "$function-data.json"
```

```
zip
```

```
aws iam create-role
```

```
aws iam put-role-policy
```

```
aws upload-function
```

AWSCLI

```
# aws lambda list-functions  --output text --query 'Functions[*].[FunctionName]'
# aws lambda get-function --function-name "$function"

# aws iam list-roles --output text --query 'Roles[*].[RoleName]'

# aws iam get-role --role-name "$lambda_execution_role_name"
  --output json --query 'Role.AssumeRolePolicyDocument.Statement'
# aws iam list-role-policies --role-name "$lambda_execution_role_name"
  --output text --query 'PolicyNames[*]'
# aws iam get-role-policy --role-name "$lambda_execution_role_name" \
  --policy-name "$lambda_execution_access_policy_name" \
  --output json --query 'PolicyDocument'

# aws iam get-role --role-name "$lambda_invocation_role_name" \
  --output json --query 'Role.AssumeRolePolicyDocument.Statement'
# aws iam list-role-policies --role-name "$lambda_invocation_role_name" \
  --output text --query 'PolicyNames[*]'
# aws iam get-role-policy --role-name "$lambda_invocation_role_name" \
  --policy-name "$lambda_invocation_access_policy_name" --output json \
  --query 'PolicyDocument'
```

```
zip
```

```
aws iam create-role
```

```
aws iam put-role-policy
```

```
aws upload-function
```

```
aws invoke async
```

AWSCLI

- Check logs via cloudwatch
- Further privileges for AWS services
- No mapping with API Gateway

```
zip
```

```
aws iam create-role
```

```
aws iam put-role-policy
```

```
aws upload-function
```

```
aws invoke async
```

https://alestic.com/2014/11/aws-lambda-cli/

SERVERLESS

ONE FRAMEWORK TO RULE THEM ALL

- Provides structure, automation and organization
- CLI to control Lambdas, API Gateway Endpoints plus AWS resources via CloudFormation
- The does-it-all framework
- Pluggable

- https://github.com/serverless
- http://docs.serverless.com/v0.5.0/docs

GETTING UP AND RUNNING

```
npm -g install serverless

serverless project create

serverless function create functions/ses-mailer
```

DIRECTORY STRUCTURE

```
s-project.json (project and author data)
s-resources-cf.json (CloudFormation template for all stages/regions)
admin.env (AWS Profiles - gitignored)
_meta (meta data that holds stage/regions config and variables - gitignored)
    |__resources (final CF templates for each stage/region)
         |__s-resources-cf-dev-useast1.json
    |__variables (variables specific to stages and regions)
         |__s-variables-common.json
         |__s-variables-dev.json
         |__s-variables-dev-useast1.json
functions (folder to group your project functions)
    |__ses-mailer (your first function)
         |__event.json (sample event for testing function locally)
         |__handler.js (your function handler file)
         |__s-function.json (data for your lambda function, endpoints and event sources)
```

functions/ses-mailer/s-function.json

```
{
  "name": "ses-mailer",
  "runtime": "nodejs",
  "description": "Serverless Lambda function for project: serverless-starter",
  "customName": false,
  "customRole": false,
  "handler": "handler.handler",
  "timeout": 6,
  "memorySize": 128,
  "authorizer": {},
  "custom": {
    "excludePatterns": []
  },

  …
```

functions/ses-mailer/s-function.json

```
 …
"endpoints": [
   {
     "path": "ses-mailer",
     "method": "POST",
     "type": "AWS",
     "authorizationType": "none",
     "authorizerFunction": false,
     "apiKeyRequired": false,
     "requestParameters": {},
     "requestTemplates": "$${apiGatewayRequestTemplate}",

     "responses": {
        …
```

functions/ses-mailer/s-function.json

```
  …
     "responses": {
        "400": {
          "statusCode": "400"
        },
        "default": {
          "statusCode": "200",
          "responseParameters": {},
          "responseModels": {},
          "responseTemplates": {
            "application/json": ""
          }
        }
      }
    }
  ],
…
```

functions/ses-mailer/s-templates.json

```
{
  "apiGatewayRequestTemplate": {
    "application/json": {
      "body": "$input.json('$')",
      "queryParams": "$input.params().querystring"
    },
    "application/x-www-form-urlencoded": "{\n  \"postBody\" : $input.json(\"$\")\n}"
  }
}
```

functions/ses-mailer/handler.js

```javascript
'use strict';
var AWS = require('aws-sdk')
var SES = new AWS.SES()
var querystring = require('querystring')

// requires cmd: aws ses verify-email-identity --email-address alr@spinscale.de
var sender = 'Spinscale Form Mailer <alr@spinscale.de>'
var recipient = 'Alexander Reelsen <alr@spinscale.de>'
var subject = 'Form mailer for spinscale.de: New enquiry'

module.exports.handler = function(event, context) {
  var data = querystring.parse(event.postBody)

  var msg = ''
  for (var key in data) {
    msg += key + ': ' + data[key] + '\n'
  }

  var email = { Source: sender, Destination: { ToAddresses: [ recipient ] }, Message: { Subject: { Data: subject }, Body: { Text: { Data: msg } } } }
  SES.sendEmail(email, function (err, data) {
    if (err) {
      console.log('Error sending mail: ', err)
      context.fail(new Error('Could not sent mail'))
    } else {
      context.succeed({ status: 'OK' })
    }
  })
};
```

GETTING UP AND RUNNING

```
serverless function run ses-mailer


serverless resources deploy
serverless function deploy
serverless endpoint deploy


serverless function logs ses-mailer -t true
```

PLUGINS...

- serverless-serve/serverless-offline
- jshint
- cronjob
- alerting (cloudwatch w/ SNS)
- swagger
- CORS
- optimizer
- client S3
- email

SUMMARY

- powerful
- highly configurable
- staging built-in
- No abstraction

- No abstraction
- Lots of boilerplate
- Cant be googled

Claudia.js

CLAUDIA

There is no cloud
it's just someone else's computer

CLAUDIA.JS - OVERVIEW

Claudia.js

- Lambda + API Gateway is abstracted away
- Support for scheduled, S3 and SNS events

- HTTP endpoints: claudia-api-builder
- No need to edit any API gateway configuration
- Cors support

GETTING UP AND RUNNING

web.js

```
var ApiBuilder = require('claudia-api-builder'),
    api = new ApiBuilder(),
    superb = require('superb');

module.exports = api;

api.get('/greet', function (request) {
    return request.queryString.name + ' is ' + superb();
});
```

```
claudia create --region us-east-1 --api-module web
```

There is no cloud
it's just someone else's computer

GETTING UP AND RUNNING

claudia.json

```
{
  "lambda": {
    "role": "web-api–executor",
    "name": "web-api",
    "region": "us-east-1"
  },
  "api": {
    "id": "iqy9sml11c",
    "module": "web-api"
  }
}
```

```
curl https://iqy9sml11c.execute-api.us-east-1.amazonaws.com/latest/greet?name=Alex
```

THE SES MAILER

Claudia.js

ses-mailer.js

```
var ApiBuilder = require('claudia-api-builder')
var api = new ApiBuilder()
var fs = require('fs')

var AWS = require('aws-sdk-promise')
var SES = new AWS.SES()

module.exports = api

var sender = 'Spinscale Form Mailer <alr@spinscale.de>'
var recipient = 'Alexander Reelsen <alr@spinscale.de>'
var subject = 'Form mailer for spinscale.de: New enquiry'
```

THE SES MAILER

ses-mailer.js

```
api.get('/version', function (req) {
  var packageJson = JSON.parse(fs.readFileSync('package.json'))
  return { 'version' : packageJson.version }
})
```

Claudia.js

There is no cloud
it's just someone else's computer

THE SES MAILER

ses-mailer.js

```
api.post('/mail', function (req) {
  var msg = ''
  for (var key in req.post) {
    msg += key + ': ' + req.post[key] + '\n'
  }

  var email = { Source: sender, Destination: { ToAddresses: [ recipient ] }, Message: {
Subject: { Data: subject }, Body: { Text: { Data: msg } } } }

  return SES.sendEmail(email).promise()
    .then(function (data) {
      return { 'status': 'OK' }
    })
    .catch(function (err) {
      console.log('Error sending mail: ' + err)
      return { 'status': 'ERROR' }
    })
})
```

There is no cloud
it's just someone else's computer

THE SES MAILER

Claudia.js

package.json

```
{
  "name": "mailer", "version": "0.0.2",
  "private": true,
  "files": [ "*.js", "package.json" ],
  "scripts": {
    "lambda-tail": "node_modules/.bin/smoketail -f /aws/lambda/ses-mailer",
    "lambda-create": "node_modules/.bin/claudia create --name ses-mailer --region us-east-1 --api-module
ses-mailer --policies policies",
    "lambda-update": "node_modules/.bin/claudia update",
    "lambda-destroy": "node_modules/.bin/claudia destroy"
  },
 "devDependencies": {
    "claudia": "1.1.2",
    "smoketail": "0.1.0",
    "standard": "6.0.8"
  },
  "dependencies": {
    "aws-sdk": "2.2.41",
    "aws-sdk-promise": "0.0.2",
    "claudia-api-builder": "1.1.0"
  }
}
```

Policy!

Packaged dependencies

There is no cloud
it's just someone else's computer

THE SES MAILER

policies/send-mail.json

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ses:SendEmail"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:*:*"
            ]
        }
    ]
}
```

SUMMARY

Claudia.js

- Abstracts away AWS
- Many things implicit
- Staging (can use `/latest` for that!)
- No profiles

- Getting up and running is crazy easy!

APPROVED

There is no cloud
it's just someone else's computer

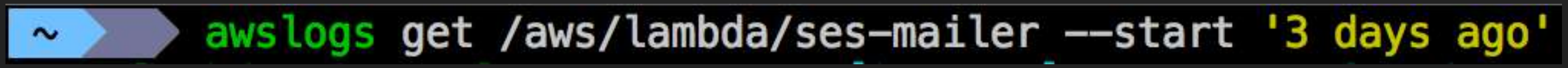
OTHERS

ALTERNATIVES

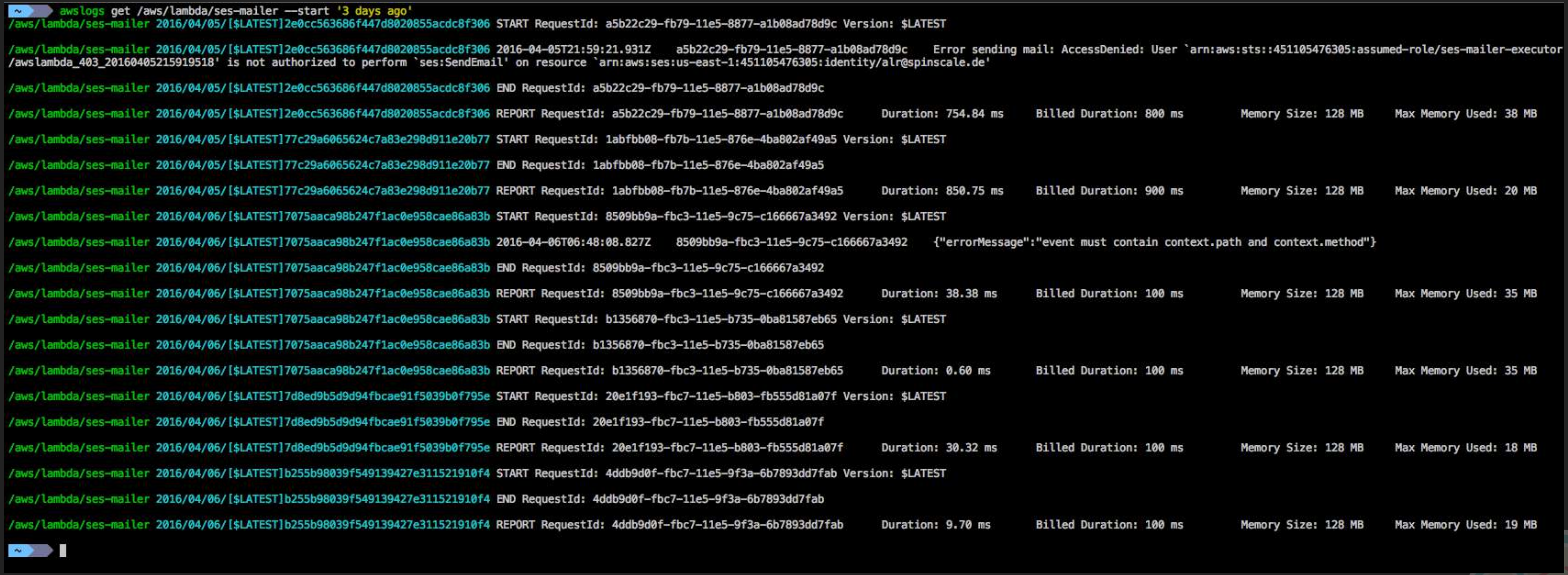
- node: deep framework
- python: kappa, zappa
- go: sparta
- java: lambada

TOOLS

AWSLOGS

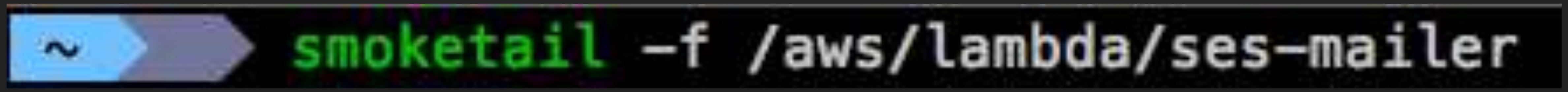


```
~ awslogs get /aws/lambda/ses-mailer --start '3 days ago'
```

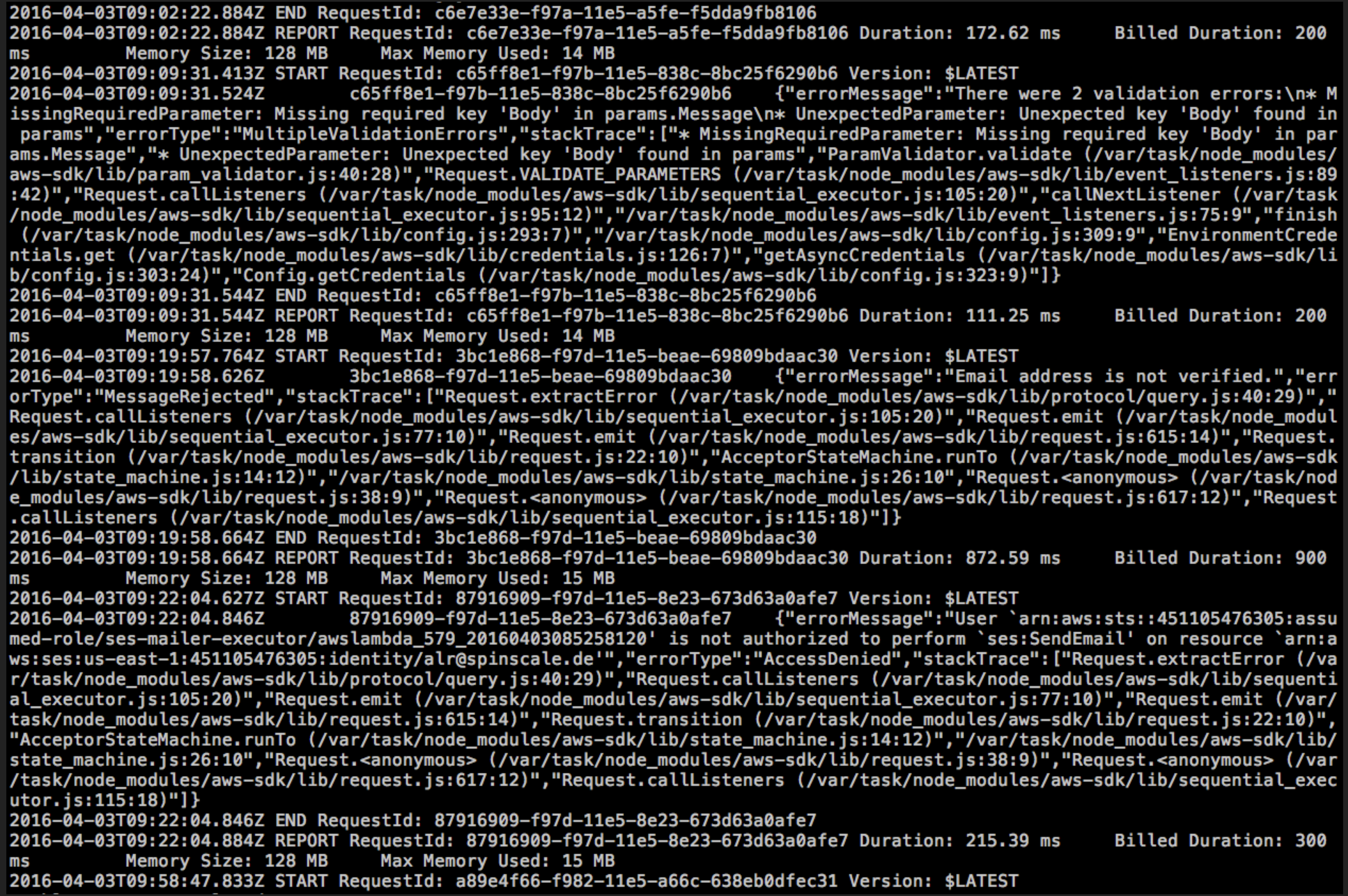


```
~ awslogs get /aws/lambda/ses-mailer --start '3 days ago'
/aws/lambda/ses-mailer 2016/04/05/[$LATEST]2e0cc563686f447d8020855acdc8f306 START RequestId: a5b22c29-fb79-11e5-8877-a1b08ad78d9c Version: $LATEST

/aws/lambda/ses-mailer 2016/04/05/[$LATEST]2e0cc563686f447d8020855acdc8f306 2016-04-05T21:59:21.931Z    a5b22c29-fb79-11e5-8877-a1b08ad78d9c    Error sending mail: AccessDenied: User `arn:aws:sts::451105476305:assumed-role/ses-mailer-executor
/awslambda_403_20160405215919518' is not authorized to perform `ses:SendEmail' on resource `arn:aws:ses:us-east-1:451105476305:identity/alr@spinscale.de'

/aws/lambda/ses-mailer 2016/04/05/[$LATEST]2e0cc563686f447d8020855acdc8f306 END RequestId: a5b22c29-fb79-11e5-8877-a1b08ad78d9c

/aws/lambda/ses-mailer 2016/04/05/[$LATEST]2e0cc563686f447d8020855acdc8f306 REPORT RequestId: a5b22c29-fb79-11e5-8877-a1b08ad78d9c      Duration: 754.84 ms     Billed Duration: 800 ms         Memory Size: 128 MB     Max Memory Used: 38 MB

/aws/lambda/ses-mailer 2016/04/05/[$LATEST]77c29a6065624c7a83e298d911e20b77 START RequestId: 1abfbb08-fb7b-11e5-876e-4ba802af49a5 Version: $LATEST

/aws/lambda/ses-mailer 2016/04/05/[$LATEST]77c29a6065624c7a83e298d911e20b77 END RequestId: 1abfbb08-fb7b-11e5-876e-4ba802af49a5

/aws/lambda/ses-mailer 2016/04/05/[$LATEST]77c29a6065624c7a83e298d911e20b77 REPORT RequestId: 1abfbb08-fb7b-11e5-876e-4ba802af49a5      Duration: 850.75 ms     Billed Duration: 900 ms         Memory Size: 128 MB     Max Memory Used: 20 MB

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7075aaca98b247f1ac0e958cae86a83b START RequestId: 8509bb9a-fbc3-11e5-9c75-c166667a3492 Version: $LATEST

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7075aaca98b247f1ac0e958cae86a83b 2016-04-06T06:48:08.827Z    8509bb9a-fbc3-11e5-9c75-c166667a3492    {"errorMessage":"event must contain context.path and context.method"}

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7075aaca98b247f1ac0e958cae86a83b END RequestId: 8509bb9a-fbc3-11e5-9c75-c166667a3492

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7075aaca98b247f1ac0e958cae86a83b REPORT RequestId: 8509bb9a-fbc3-11e5-9c75-c166667a3492      Duration: 38.38 ms      Billed Duration: 100 ms         Memory Size: 128 MB     Max Memory Used: 35 MB

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7075aaca98b247f1ac0e958cae86a83b START RequestId: b1356870-fbc3-11e5-b735-0ba81587eb65 Version: $LATEST

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7075aaca98b247f1ac0e958cae86a83b END RequestId: b1356870-fbc3-11e5-b735-0ba81587eb65

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7075aaca98b247f1ac0e958cae86a83b REPORT RequestId: b1356870-fbc3-11e5-b735-0ba81587eb65      Duration: 0.60 ms       Billed Duration: 100 ms         Memory Size: 128 MB     Max Memory Used: 35 MB

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7d8ed9b5d9d94fbcae91f5039b0f795e START RequestId: 20e1f193-fbc7-11e5-b803-fb555d81a07f Version: $LATEST

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7d8ed9b5d9d94fbcae91f5039b0f795e END RequestId: 20e1f193-fbc7-11e5-b803-fb555d81a07f

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]7d8ed9b5d9d94fbcae91f5039b0f795e REPORT RequestId: 20e1f193-fbc7-11e5-b803-fb555d81a07f      Duration: 30.32 ms      Billed Duration: 100 ms         Memory Size: 128 MB     Max Memory Used: 18 MB

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]b255b98039f549139427e311521910f4 START RequestId: 4ddb9d0f-fbc7-11e5-9f3a-6b7893dd7fab Version: $LATEST

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]b255b98039f549139427e311521910f4 END RequestId: 4ddb9d0f-fbc7-11e5-9f3a-6b7893dd7fab

/aws/lambda/ses-mailer 2016/04/06/[$LATEST]b255b98039f549139427e311521910f4 REPORT RequestId: 4ddb9d0f-fbc7-11e5-9f3a-6b7893dd7fab      Duration: 9.70 ms       Billed Duration: 100 ms         Memory Size: 128 MB     Max Memory Used: 19 MB

~ ▌
```

SMOKETAIL



```
~ smoketail -f /aws/lambda/ses-mailer
```

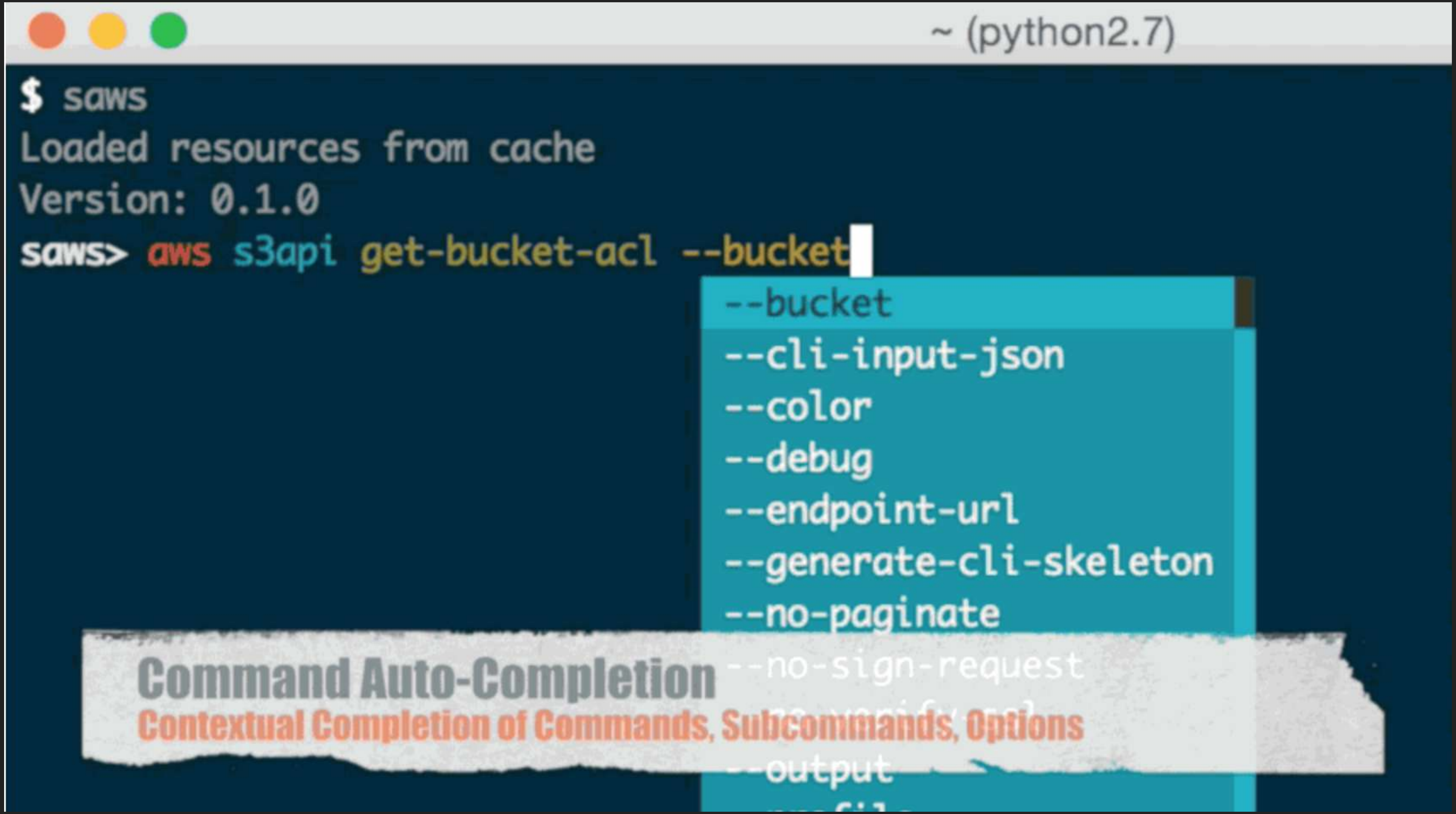


```
2016-04-03T09:02:22.884Z END RequestId: c6e7e33e-f97a-11e5-a5fe-f5dda9fb8106
2016-04-03T09:02:22.884Z REPORT RequestId: c6e7e33e-f97a-11e5-a5fe-f5dda9fb8106 Duration: 172.62 ms     Billed Duration: 200
ms          Memory Size: 128 MB     Max Memory Used: 14 MB
2016-04-03T09:09:31.413Z START RequestId: c65ff8e1-f97b-11e5-838c-8bc25f6290b6 Version: $LATEST
2016-04-03T09:09:31.524Z        c65ff8e1-f97b-11e5-838c-8bc25f6290b6    {"errorMessage":"There were 2 validation errors:\n* M
issingRequiredParameter: Missing required key 'Body' in params.Message\n* UnexpectedParameter: Unexpected key 'Body' found in
 params","errorType":"MultipleValidationErrors","stackTrace":["* MissingRequiredParameter: Missing required key 'Body' in par
ams.Message","* UnexpectedParameter: Unexpected key 'Body' found in params","ParamValidator.validate (/var/task/node_modules/
aws-sdk/lib/param_validator.js:40:28)","Request.VALIDATE_PARAMETERS (/var/task/node_modules/aws-sdk/lib/event_listeners.js:89
:42)","Request.callListeners (/var/task/node_modules/aws-sdk/lib/sequential_executor.js:105:20)","callNextListener (/var/task
/node_modules/aws-sdk/lib/sequential_executor.js:95:12)","/var/task/node_modules/aws-sdk/lib/event_listeners.js:75:9","finish
 (/var/task/node_modules/aws-sdk/lib/config.js:293:7)","/var/task/node_modules/aws-sdk/lib/config.js:309:9","EnvironmentCrede
ntials.get (/var/task/node_modules/aws-sdk/lib/credentials.js:126:7)","getAsyncCredentials (/var/task/node_modules/aws-sdk/li
b/config.js:303:24)","Config.getCredentials (/var/task/node_modules/aws-sdk/lib/config.js:323:9)"]}
2016-04-03T09:09:31.544Z END RequestId: c65ff8e1-f97b-11e5-838c-8bc25f6290b6
2016-04-03T09:09:31.544Z REPORT RequestId: c65ff8e1-f97b-11e5-838c-8bc25f6290b6 Duration: 111.25 ms     Billed Duration: 200
ms          Memory Size: 128 MB     Max Memory Used: 14 MB
2016-04-03T09:19:57.764Z START RequestId: 3bc1e868-f97d-11e5-beae-69809bdaac30 Version: $LATEST
2016-04-03T09:19:58.626Z        3bc1e868-f97d-11e5-beae-69809bdaac30    {"errorMessage":"Email address is not verified.","err
orType":"MessageRejected","stackTrace":["Request.extractError (/var/task/node_modules/aws-sdk/lib/protocol/query.js:40:29)","
Request.callListeners (/var/task/node_modules/aws-sdk/lib/sequential_executor.js:105:20)","Request.emit (/var/task/node_modul
es/aws-sdk/lib/sequential_executor.js:77:10)","Request.emit (/var/task/node_modules/aws-sdk/lib/request.js:615:14)","Request.
transition (/var/task/node_modules/aws-sdk/lib/request.js:22:10)","AcceptorStateMachine.runTo (/var/task/node_modules/aws-sdk
/lib/state_machine.js:14:12)","/var/task/node_modules/aws-sdk/lib/state_machine.js:26:10","Request.<anonymous> (/var/task/nod
e_modules/aws-sdk/lib/request.js:38:9)","Request.<anonymous> (/var/task/node_modules/aws-sdk/lib/request.js:617:12)","Request
.callListeners (/var/task/node_modules/aws-sdk/lib/sequential_executor.js:115:18)"]}
2016-04-03T09:19:58.664Z END RequestId: 3bc1e868-f97d-11e5-beae-69809bdaac30
2016-04-03T09:19:58.664Z REPORT RequestId: 3bc1e868-f97d-11e5-beae-69809bdaac30 Duration: 872.59 ms     Billed Duration: 900
ms          Memory Size: 128 MB     Max Memory Used: 15 MB
2016-04-03T09:22:04.627Z START RequestId: 87916909-f97d-11e5-8e23-673d63a0afe7 Version: $LATEST
2016-04-03T09:22:04.846Z        87916909-f97d-11e5-8e23-673d63a0afe7    {"errorMessage":"User `arn:aws:sts::451105476305:assu
med-role/ses-mailer-executor/awslambda_579_20160403085258120' is not authorized to perform `ses:SendEmail' on resource `arn:a
ws:ses:us-east-1:451105476305:identity/alr@spinscale.de'","errorType":"AccessDenied","stackTrace":["Request.extractError (/va
r/task/node_modules/aws-sdk/lib/protocol/query.js:40:29)","Request.callListeners (/var/task/node_modules/aws-sdk/lib/sequenti
al_executor.js:105:20)","Request.emit (/var/task/node_modules/aws-sdk/lib/sequential_executor.js:77:10)","Request.emit (/var/
task/node_modules/aws-sdk/lib/request.js:615:14)","Request.transition (/var/task/node_modules/aws-sdk/lib/request.js:22:10)",
"AcceptorStateMachine.runTo (/var/task/node_modules/aws-sdk/lib/state_machine.js:14:12)","/var/task/node_modules/aws-sdk/lib/
state_machine.js:26:10","Request.<anonymous> (/var/task/node_modules/aws-sdk/lib/request.js:38:9)","Request.<anonymous> (/var
/task/node_modules/aws-sdk/lib/request.js:617:12)","Request.callListeners (/var/task/node_modules/aws-sdk/lib/sequential_exec
utor.js:115:18)"]}
2016-04-03T09:22:04.846Z END RequestId: 87916909-f97d-11e5-8e23-673d63a0afe7
2016-04-03T09:22:04.884Z REPORT RequestId: 87916909-f97d-11e5-8e23-673d63a0afe7 Duration: 215.39 ms     Billed Duration: 300
ms          Memory Size: 128 MB     Max Memory Used: 15 MB
2016-04-03T09:58:47.833Z START RequestId: a89e4f66-f982-11e5-a66c-638eb0dfec31 Version: $LATEST
```

APEX

- Support for GO
- Binary install (install apex quickly for continuous deployment in CI etc)
- Hook support for running commands (transpile code, lint, etc)
- Batteries included but optional (opt-in to higher level abstractions)
- Transparently generates a zip for your deploy
- Project bootstrapping with optional Terraform support
- Function rollback support
- Tail function logs
- Concurrency for quick deploys
- Dry-run to preview changes
- VPC support

SAWS



```
$ saws
Loaded resources from cache
Version: 0.1.0
saws> aws s3api get-bucket-acl --bucket
                                    --bucket
                                    --cli-input-json
                                    --color
                                    --debug
                                    --endpoint-url
                                    --generate-cli-skeleton
                                    --no-paginate
                                    --no-sign-request
                                    --no-verify-ssl
                                    --output
```

CONCERNS

THINK ABOUT

- Staging envs
- Compliance
- Security
- Debugging
- Logging
- Pay as you go…
- Latency
- Rate limiting
- Platform complexity
- Decoupling
- Vendor lock-in
- Tooling
- Frameworks

RESOURCES

OVERVIEW SPEAKERS EVENT TICKETS TEAM CONTACT

the future will be

SERVERLESS

MAKE HISTORY IN NYC

May 26 & 27 2016

RESOURCES

- Serverless Conference, NY, 26th-27th May: http://serverlessconf.io/

- Serverless: http://serverless.com
- Claudia.js: https://github.com/claudiajs/

- Smoketail: https://github.com/cinema6/smoketail
- awslogs: https://github.com/jorgebastida/awslogs
- Apex: https://github.com/apex/apex
- Saws: https://pythonhosted.org/saws/

There is no cloud
it's just someone else's computer

RESOURCES

- Lambda: https://docs.aws.amazon.com/lambda/latest/
- Google Cloud Functions: https://cloud.google.com/functions/docs
- Azure Functions: https://azure.microsoft.com/en-us/services/functions/
- openwhisk: https://developer.ibm.com/openwhisk/
- iron.io: https://www.iron.io/
- webtask.io: https://webtask.io/
- firebase: https://www.firebase.com/

RESOURCES

- python
  - Kappa: https://github.com/garnaat/kappa
  - Zappa: https://github.com/Miserlou/Zappa
- Java
  - Lambada: https://github.com/lambadaframework/lambadaframework
- Go
  - Sparta: http://gosparta.io/
- Node
  - deep framework: https://github.com/MitocGroup/deep-framework

BLOGPOSTS

- https://gojko.net/2016/02/22/introducing-claudia/
- https://blog.codeship.com/a-serverless-rest-api-in-minutes/
- https://medium.com/@tjholowaychuk/introducing-apex-800824ffaa70
- http://www.rylerhockenbury.com/blog/making-serverless-architectures-manageable
- http://julienblanchard.com/2015/rust-on-aws-lambda/
- http://veldstra.org/2016/02/18/project-dino-load-testing-on-lambda-with-artillery.html
- https://medium.com/precipitation-io/your-buzzword-for-2016-serverless-fd7620eb35f2
- http://www.it20.info/2016/04/aws-lambda-a-few-years-of-advancement-and-we-are-back-to-stored-procedures/

BLOGPOSTS

- https://medium.com/teletext-io-blog/the-serverless-start-up-228370932cb8
- https://spinscale.de/posts/2016-03-21-using-webtasks-to-send-emails-with-harp.html
- https://spinscale.de/posts/2016-04-06-using-claudia-js-to-send-emails-using-aws-lambda.html
- https://blog.ouseful.info/2016/03/16/implementing-slash-commands-using-amazon-lambda-functions-encrypting-the-slack-token/
- http://go.iron.io/project-kratos
- https://aws.amazon.com/blogs/compute/the-squirrelbin-architecture-a-serverless-microservice-using-aws-lambda/

PRESENTATIONS

- https://speakerdeck.com/stevenringo/going-serverless-noops-is-the-best-ops
- https://speakerdeck.com/martinb3/going-serverless-with-aws-lambda
- https://slidr.io/s0enke/aws-infrastructure-plumbing

- https://github.com/anaibol/awesome-serverless

BOOKS

- Serverless by Obie Fernandez
- https://leanpub.com/serverless

BETA
SERVERLESS
Patterns of Modern Application Design Using Microservices
Obie Fernandez
AMAZON WEB SERVICES EDITION

BOOKS

- AWS Lambda - A guide to serverless microservices, by Matthew Fuller
- http://www.amazon.de/AWS-Lambda-Serverless-Microservices-English-ebook/dp/B016JOMAEE

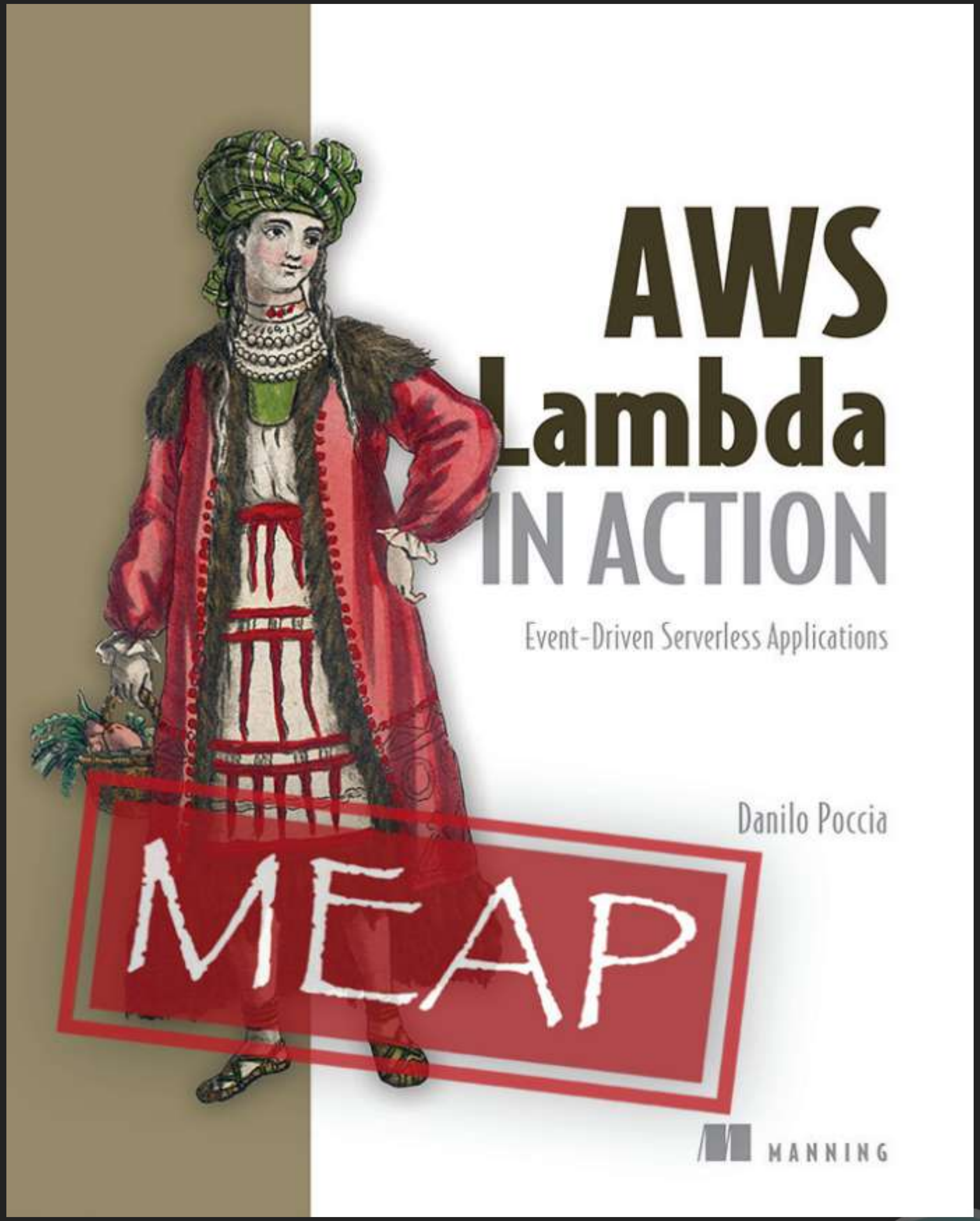
BOOKS

- Serverless Single Page Apps, by Ben Rady
- https://pragprog.com/book/brapps/serverless-single-page-apps

The Pragmatic Programmers
Serverless Single Page Apps
Fast, Scalable, and Available
Ben Rady
edited by Jacquelyn Carter

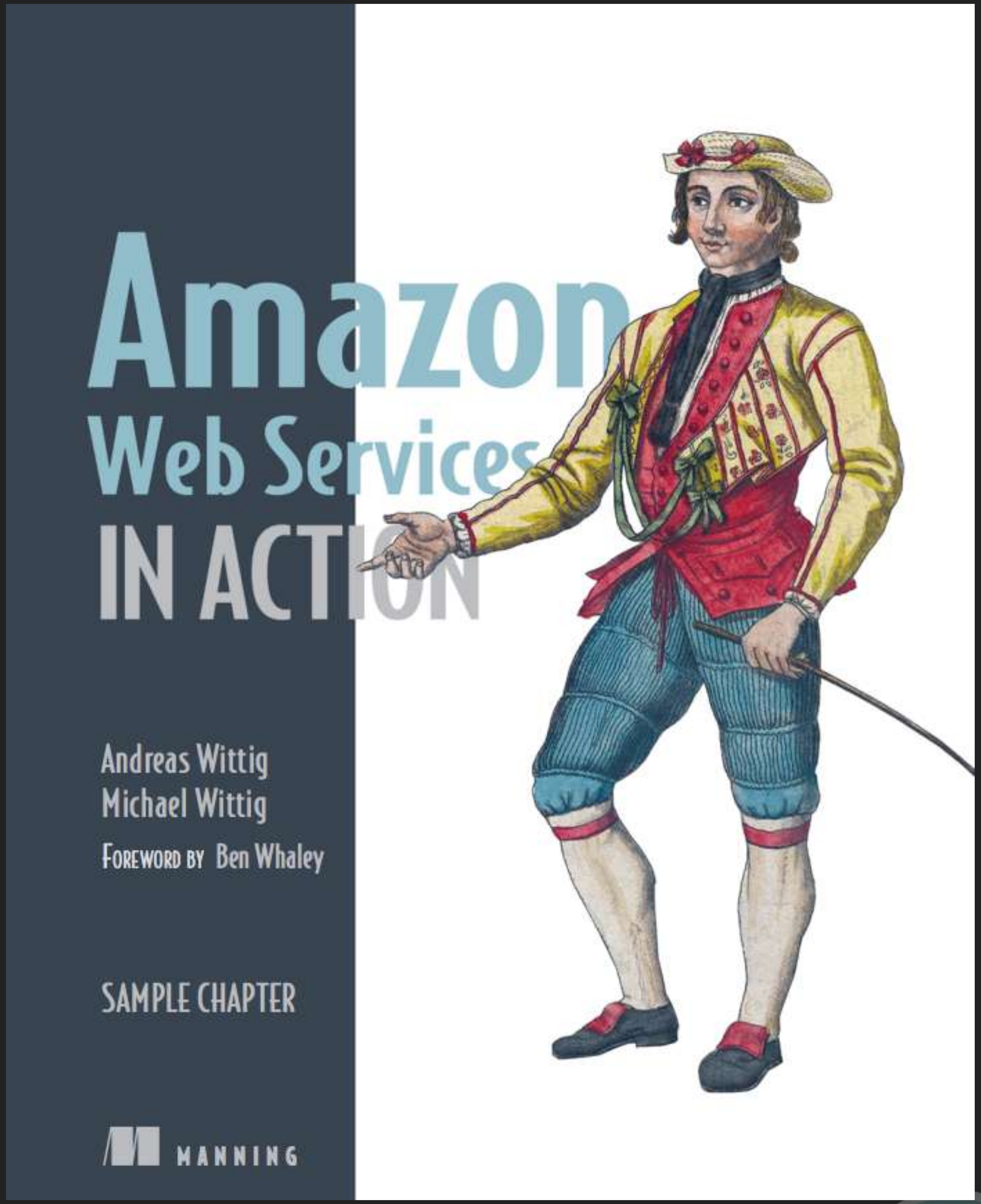
BOOKS

- AWS Lambda in Action, by Danilo Poccia
- https://www.manning.com/books/aws-lambda-in-action



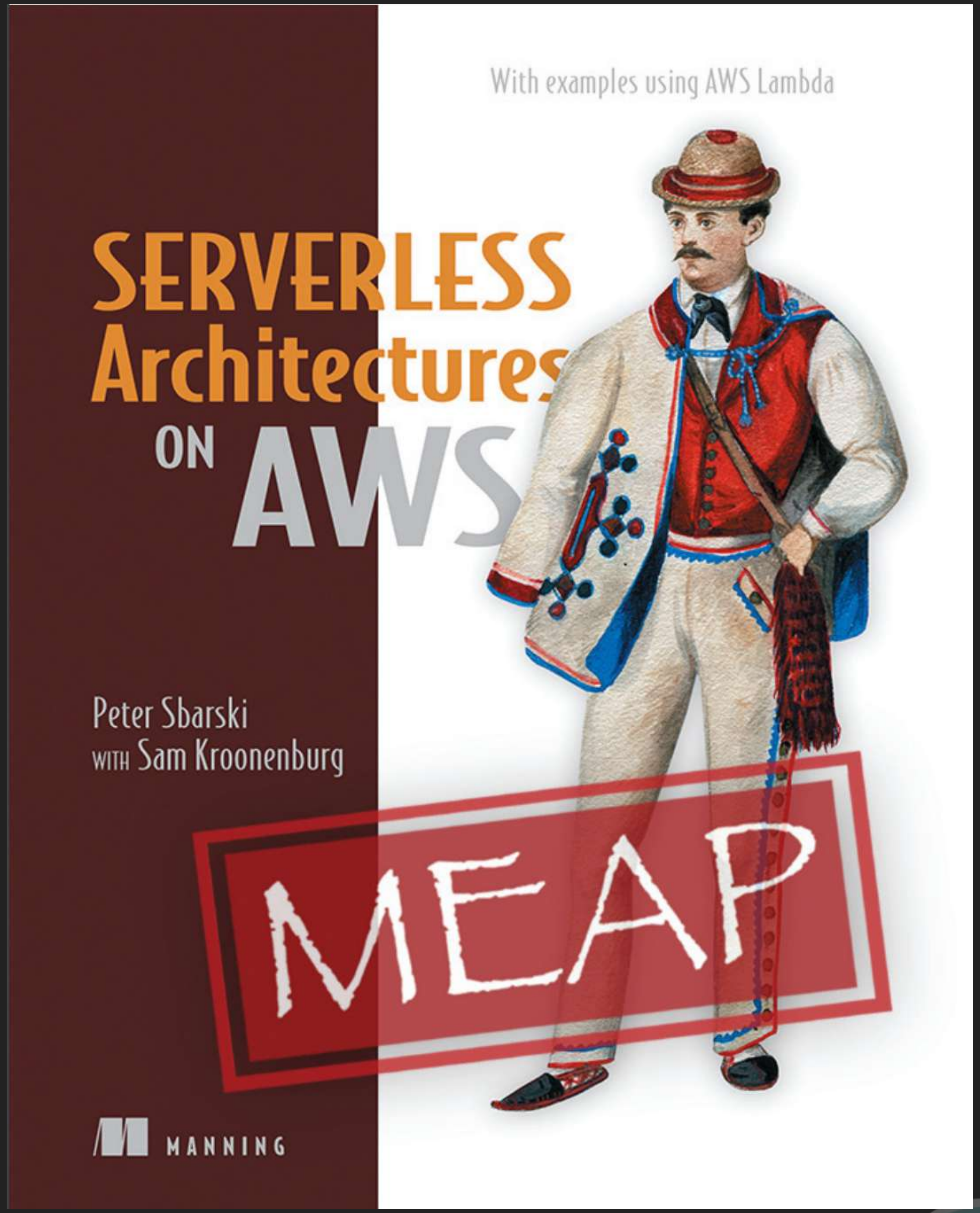
BOOKS

- Amazon Web Services in Action, by Andreas Wittig & Michael Wittig
- https://www.manning.com/books/amazon-web-services-in-action



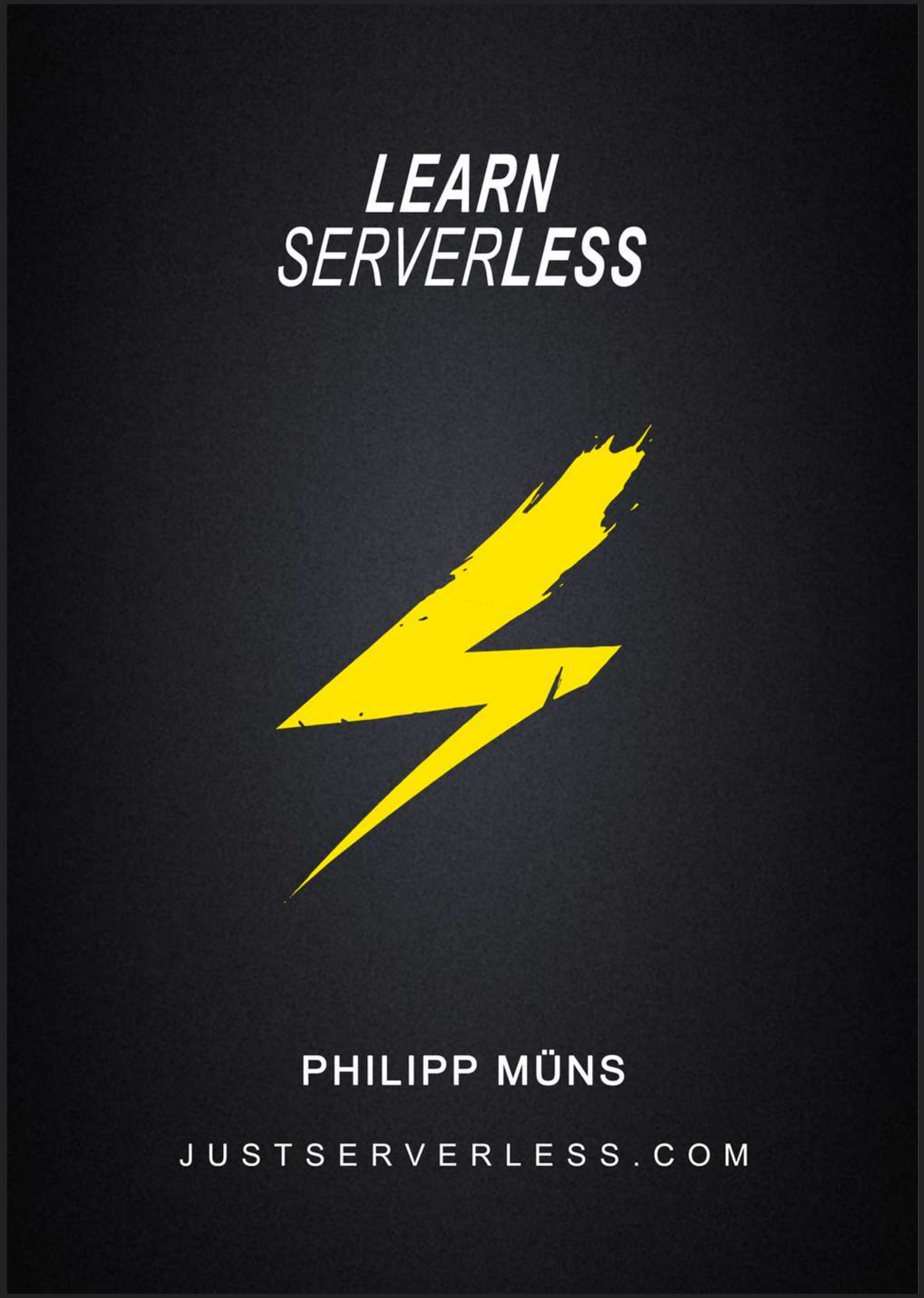
BOOKS

- Serverless Architectures on AWS, by Peter Sbarski & Sam Kroonenburg
- https://www.manning.com/books/serverless-architectures-on-aws


With examples using AWS Lambda
SERVERLESS Architectures ON AWS
Peter Sbarski WITH Sam Kroonenburg
MEAP
MANNING

BOOKS

- Learn Serverless, by Philip Muens
- http://gum.co/learn-serverless-book



RESOURCES

- Pictures for illustration: https://pixabay.com

QUESTIONS?

https://speakerdeck.com/spinscale/serverless-apps-without-infrastructure

https://slidr.io/spinscale/serverless-apps-without-infrastructure

Alexander Reelsen

@spinscale

alr@spinscale.de

CLAUDIA QUICKSTART / DEMO

```
mkdir claudia-hello-world
cd claudia-hello-world
npm init -y
npm i --save-dev claudia
npm i --save claudia-api-builder superb
vi api.js
claudia create --region us-east-1 --api-module api
curl -v 'https://<<ID>>.execute-api.us-east-1.amazonaws.com/latest/hello?name=Alex'
vi api.js
claudia update
curl -v 'https://<<ID>>.execute-api.us-east-1.amazonaws.com/latest/hello?name=Alex'
```

api.js

```
var ApiBuilder = require('claudia-api-builder'),
    api = new ApiBuilder(),
    superb = require('superb');

module.exports = api;
// TODO make me return JSON in the next step
api.get('/greet', function (request) {
    return request.queryString.name + ' is ' + superb()
});
```