Reasonable Cloud Practices

with Java on AWS

airhacks.live

airhacks.industries

airhacks.live

"It's not work if you like it"
…so I never worked. #java

airhacks.fm

#185 A Cloud Migration Story: From J2EE to Serverless Java

[episode link] Listen on Apple Podcasts LISTEN ON Spotify Listen on Google Podcasts [RSS]

An airhacks.fm

ZX Spectru
CPC 64, De
in 1993, usi
Lambda, Cl
clouds the
services, n
quarkus in
the cloud h

#219 Java, CraC and Reducing Cold Start Duration with AWS Lambda SnapStart

[episode li

An airha

CRaC
snap
with
befo
Zone
Snap
Lam
meta

#263 How JAX-RS Happened

[episode link] Listen on Apple Podcasts LISTEN ON Spotify Listen on Google Podcasts [RSS]

An airhacks.fm conversation with Paul Sandoz (@paulsandoz) about:

bbc micro, Archimedes was the start of ARM, the Elite game, writing a painting appl
protocol emulation, studying cybernetics, remote control of production factories, d
headset, using Silicon Graphics machines, building a 3D engine, working on Sun Mic
the CDE environment, switching to XML technology group at Sun Microsystems, Ap
was the past of least resistance, the WS-* specifications, Roy Fielding and Represer

airhacks.live

airhacks.TV

with the time machine, “100 episodes ago segment”

…any questions left?

youtube.com/ @bienadam/shorts

(1st anniversary 👉 > 390 shorts)

airhacks.live

NEW https://discord.gg/airhacks

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Winter 2023:

- AWS Security, Authentication and Authorization for Java, December 7th, 2023
- Serverless Java Patterns and Best Practices on AWS, December 14th, 2023

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io

…I started with DevOps in 1995

then continued with serverless computing in 2001…

…before we start

Cloudy Java / Green IT

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

GraalVM™

reduces RAM footprint

https://sites.google.com/view/energy-efficiency-languages/results?authuser=0

motivation / facts

Configure AWS Lambda Info

Architecture

Arm

Number of requests

2

Unit

per second

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

100

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

Value

2

Unit

GB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

Value

512

Unit

MB

Show calculations

Provisioned Concurrency Info

Total Upfront cost: 0.00 USD
Total Monthly cost: 15.07 USD

Show Details

Save and view summary

Save and add service

Configure AWS Fargate Info

Operating system

Linux

CPU Architecture

ARM

Number of tasks or pods

Enter the number of tasks or pods running for your application

Value: 1

Unit: per month

Average duration

Enter the time period for which your tasks or pods are running. Pricing is per second with a 1-minute minimum (Linux) and a 15-minute minimum (Windows). Duration is calculated from the time you start to download your container image (docker pull) until the Task or Pod terminates, rounded up to the nearest second.

Value: 30

Unit: days

Amount of vCPU allocated

Enter the amount between 0.25 vCPU and 16 vCPU

1

vCPU selected supports memory values between 2 GB and 8 GB, in 1 GB increments

Amount of memory allocated

8

Unit: GB

Total Upfront cost: 0.00 USD
Total Monthly cost: 50.38 USD

Show Details

Save and view summary

Save and add service

○	t4g.nano	2	0.5 GiB	Up to 5 Gigabit	EBS only	0.0048	Yes	0.0018 (63%)
○	t3a.nano	2	0.5 GiB	Up to 5 Gigabit	EBS only	0.0054	Yes	0.0020 (63%)
○	t3.nano	2	0.5 GiB	Up to 5 Gigabit	EBS only	0.006	Yes	0.0022 (63%)
○	t2.nano	1	0.5 GiB	Low	EBS only	0.0067	Yes	0.0030 (55%)
○	t4g.micro	2	1 GiB	Up to 5 Gigabit	EBS only	0.0096	Yes	0.0036 (62%)
○	t3a.micro	2	1 GiB	Up to 5 Gigabit	EBS only	0.0108	Yes	0.0041 (62%)
○	t3.micro	2	1 GiB	Up to 5 Gigabit	EBS only	0.012	Yes	0.0045 (62%)
○	t2.micro	1	1 GiB	Low to Moderate	EBS only	0.0134	Yes	0.0061 (55%)
◉	t4g.small	2	2 GiB	Up to 5 Gigabit	EBS only	0.0192	Yes	0.0072 (62%)
○	t3a.small	2	2 GiB	Up to 5 Gigabit	EBS only	0.0216	Yes	0.0081 (62%)

Payment options

Total Upfront cost: 0.00 USD
Total Monthly cost: 8.25 USD

Show Details ▼

Cancel

Update

EC2 Instances (589)

Based on your inputs, this is the lowest-cost EC2 instance: **t4g.nano**

Chosen instance: **t3.large | Family: t3 | 2vCPU | 8 GiB Memory**

Search instance type

Search by instance name or filter by keyword

Instance family Info

Any Instance family

vCPUs

Any vCP

Amazon EC2 On-Demand instances cost (Monthly): 70.08 USD

Total Upfront cost: 0.00 USD

Total Monthly cost: 70.08 USD

Show Details

Configure Amazon RDS for PostgreSQL Info

Selected Instance:

db.m4.large

vCPU: 2

Memory: 8 GiB

Utilization (On-Demand only)

With utilization, you still have to stop the instance to get the cost benefit. Utilization only affects OnDemand pricing for instances and not the storage, backups, etc.

Value	Unit
100	%Utilized/Month

Deployment Option

Single-AZ

Pricing Model

OnDemand

▼ Show calculations

1 instance(s) x 0.217 USD hourly x (100 / 100 Utilized/Month) x 730 hours in a month = 158.4100 USD

Amazon RDS PostgreSQL instances cost (monthly): 158.41 USD

Amazon RDS PostgreSQL instances cost (upfront): 0.00 USD

Total Upfront cost: 0.00 USD

Total Monthly cost: 188.80 USD

Show Details ▼

Specifications

Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
Gigabit Ethernet
2 USB 3.0 ports; 2 USB 2.0 ports.
Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
2 × micro-HDMI® ports (up to 4kp60 supported)
2-lane MIPI DSI display port
2-lane MIPI CSI camera port
4-pole stereo audio and composite video port
H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
OpenGL ES 3.1, Vulkan 1.0
Micro-SD card slot for loading operating system and data storage
5V DC via USB-C connector (minimum 3A*)
5V DC via GPIO header (minimum 3A*)
Power over Ethernet (PoE) enabled (requires separate PoE HAT)
Operating temperature: 0 – 50 degrees C ambient

“web-scale” applications

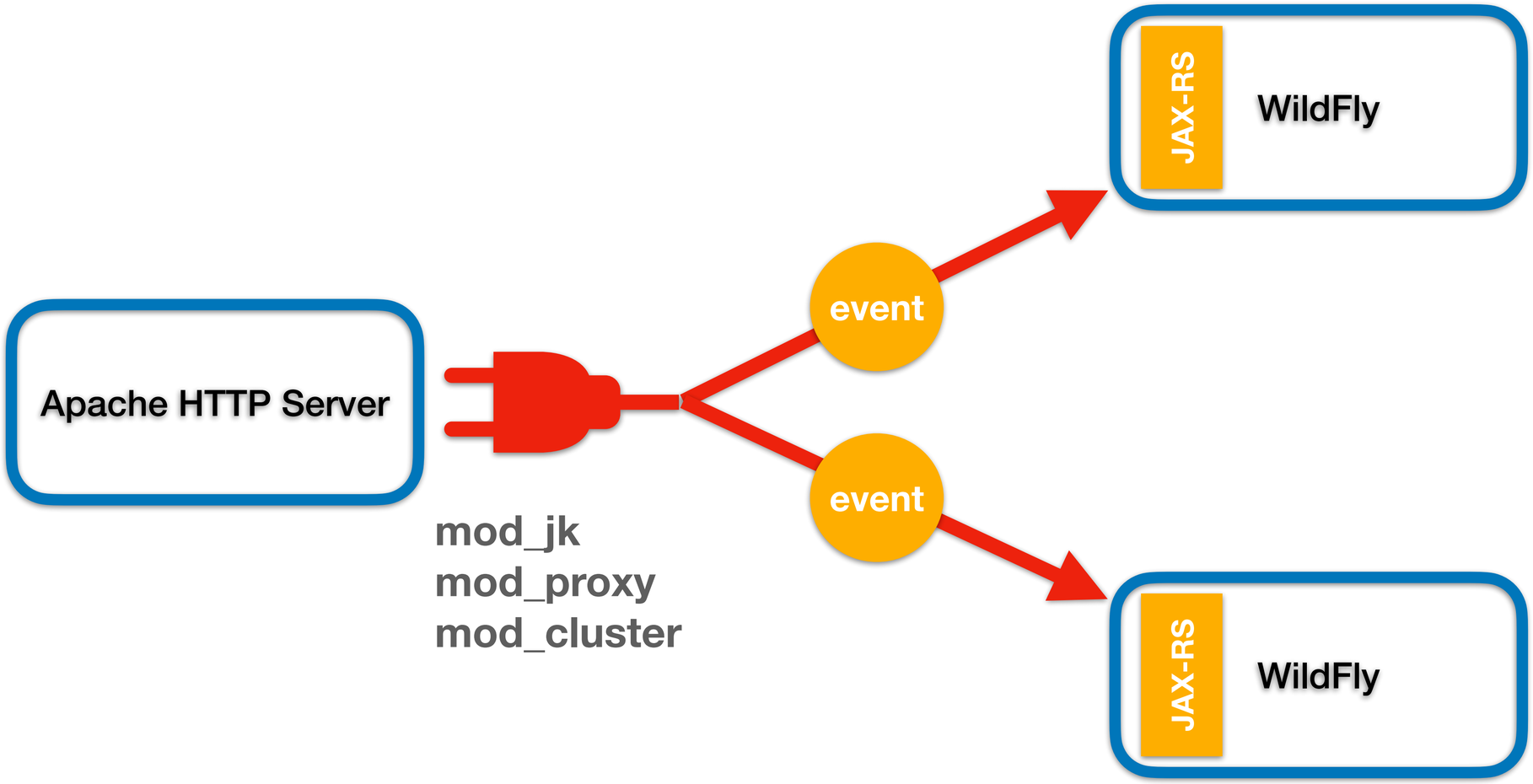
web-scale applications

- no vacations, weekends or 9am-5pm 😀
- unpredictable loads
- fast scaling necessary
- cold starts are problematic
- latency hurts
- multi-region deployments may be necessary

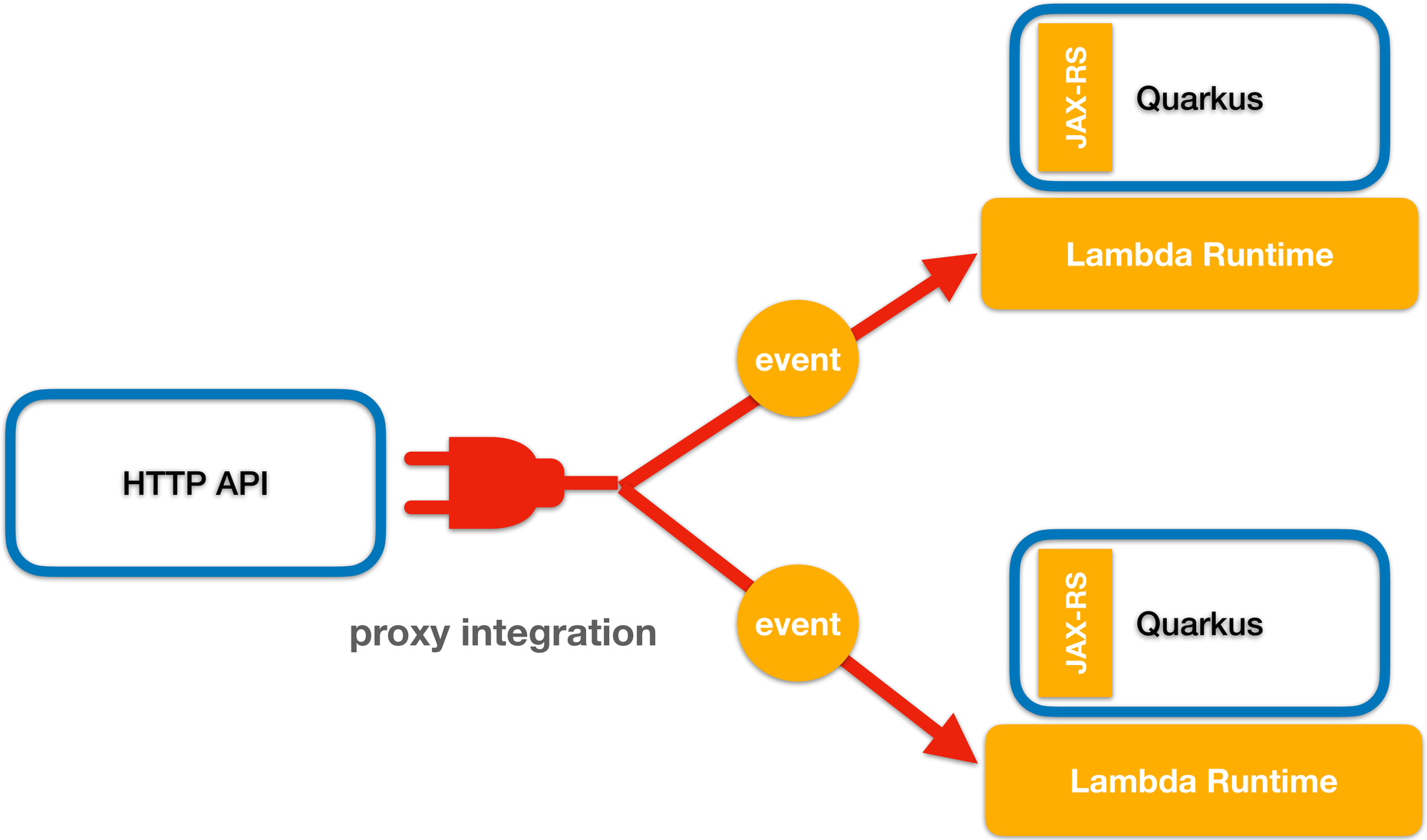
enterprise applications

enterprise applications

- most traffic 8am-5pm
- on vacations, weekends and nights - low traffic
- spikes are possible, low concurrency
- users are used to performance problems :-)
- mostly boring CRUD use cases
- often “private cloud”
- usually one-region deployment


Apache HTTP Server
mod_jk
mod_proxy
mod_cluster
event
event
JAX-RS
WildFly
JAX-RS
WildFly

AWS Lambda


JAX-RS
Quarkus
Lambda Runtime
event
HTTP API
proxy integration
event
JAX-RS
Quarkus
Lambda Runtime

serverless for enterprise

- better target tracking autoscaling
- almost infinite autoscaling (1000 lambdas are the default)
- scale to zero - no traffic, no costs
- free staging
- AB deployment
- built-in rolling updates

shifting left + right
(at the same time 😀)

Shift Right

What does shift right mean in DevOps?

Shift right is the practice of performing testing, quality, and performance evaluation in production under real-world conditions. Shift right methods ensure that applications running in production can withstand real user load while ensuring the same high levels of quality.

With shift right, DevOps teams test a built application to ensure performance, resilience, and software reliability. The goal is to detect and remediate issues that would be difficult to anticipate in development environments.

Shift Left

Shift-left testing[1] is an approach to software testing and system testing in which testing is performed earlier in the lifecycle (i.e. moved left on the project timeline). It is the first half of the maxim "test early and often".[2] It was coined by Larry Smith in 2001.[3][4]

serverless first

(…or bare metal)

cloud native services 👉 CDK + SDK

Fat, Monolithic Function

“Nano”, Asynchronous / Event Driven Function

Serverless Archive: SAR (~~EAR~~😊)

App as a Function

on-premise / kubernetes “edition”

Jakarta EE 11 Platform

JAKARTA EE

	Authentication 3.1	Data 1.0	
Authorization 3.0	Concurrency 3.1	Persistence 3.2	RESTful Web Services 4.0
Activation 2.1	CDI 4.1	Pages 4.0	JSON Processing 2.1
Batch 2.1	Expression Language 6.0	WebSocket 2.2	JSON Binding 3.0
Connectors 2.1	Faces 5.0	Validation 3.1	Annotations 3.0
Mail 2.1	Security 4.0	Debugging Support 2.0	Interceptors 2.2
Messaging 3.1	Servlet 6.1	Enterprise Beans Lite 4.0	Dependency Injection 2.0
Enterprise Beans 4.0	Standard Tag Libraries 3.0	Transactions 2.0	CDI Lite 4.1

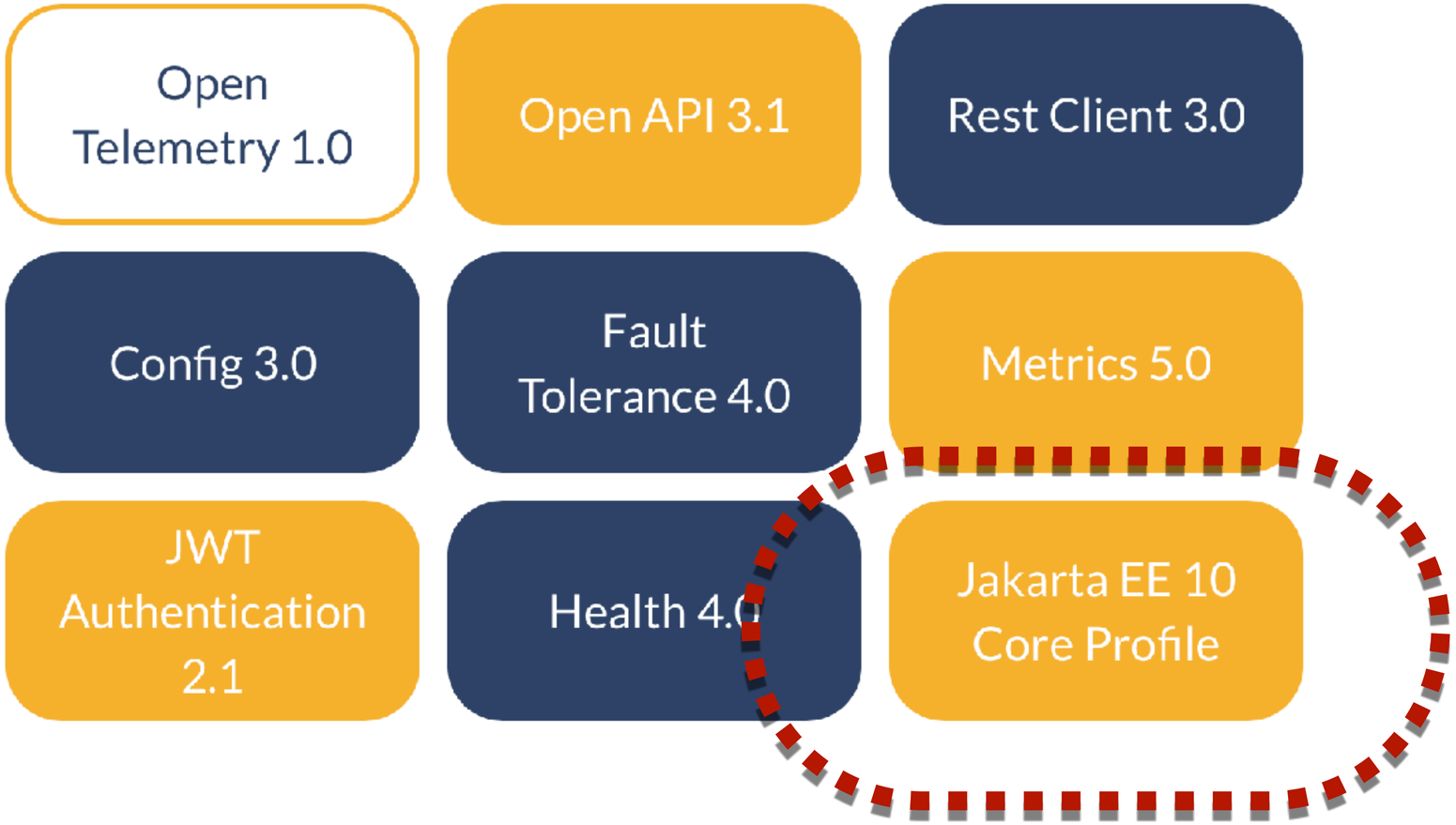
“serverless”

Jakarta EE 11 Platform

JAKARTA EE

	Authentication 3.1	Data 1.0	
Authorization 3.0	Concurrency 3.1	Persistence 3.2	RESTful Web Services 4.0
Activation 2.1	CDI 4.1	Pages 4.0	JSON Processing 2.1
Batch 2.1	Expression Language 6.0	WebSocket 2.2	JSON Binding 3.0
Connectors 2.1	Faces 5.0	Validation 3.1	Annotations 3.0
Mail 2.1	Security 4.0	Debugging Support 2.0	Interceptors 2.2
Messaging 3.1	Servlet 6.1	Enterprise Beans Lite 4.0	Dependency Injection 2.0
Enterprise Beans 4.0	Standard Tag Libraries 3.0	Transactions 2.0	CDI Lite 4.1

microprofile on-premise / kubernetes edition



“serverless“ microprofile

if you go to the cloud
…use the cloud!

references

- “Ten Years After: From Java EE 6 to Quarkus and AWS Lambda”: http://youtube.com/@bienadam/
- “Hey Enterprise EJB Developers Now Is The Time To Go Serverless”: http://youtube.com/@bienadam/
- “Past, Present and Future: Serverside Java on Premise and in the Clouds”: http://youtube.com/@bienadam/
- “Saving Costs with Pragmatic Serverless Java Architectures”: http://youtube.com/@bienadam/
- https://github.com/AdamBien/x-ray (~15 years old code)
- https://github.com/AdamBien/lightfish (~15 years old code)

references

- https://github.com/AdamBien/aws-java-functionurl-cdk-plain
- https://github.com/AdamBien/aws-quarkus-lambda-cdk-plain
- https://github.com/AdamBien/aws-cdk-plain
- https://github.com/AdamBien/eras3r
- “Using Amazon Corretto (OpenJDK) for lean, fast, and efficient AWS Lambda Applications” https://aws.amazon.com/blogs/developer/lean-fast-and-oversized-aws-lambdas-on-jvm-amazon-corretto-java/

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Winter 2023:

- AWS Security, Authentication and Authorization for Java, December 7th, 2023
- Serverless Java Patterns and Best Practices on AWS, December 14th, 2023

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io

Thank YOU!

airhacks.industries