Real World Lean Java Practices, Patterns, Hacks and Workarounds

🚀productive && ⚙️pragmatic

…actually: just common sense

airhacks.industries

airhacks.live

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

airhacks.fm

#232 Kubernetes Was Never Supposed To Leak

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

An airhacks.fm conversation with Kelsey Hightower (@kelseyhightower) about:

HP lapt
TI-BASI
at Goog
working
rewritir
to Pyth
for con
Java, J
Cost Dr
starting
and clu
...

airhack

#103 Unit Testing Considered Harmful

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

A

#259 How Boundary Control Entity, UML and Components Happened

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

An airhacks.fm conversation with Ivar Jacobson (@ivarjacobson) about:

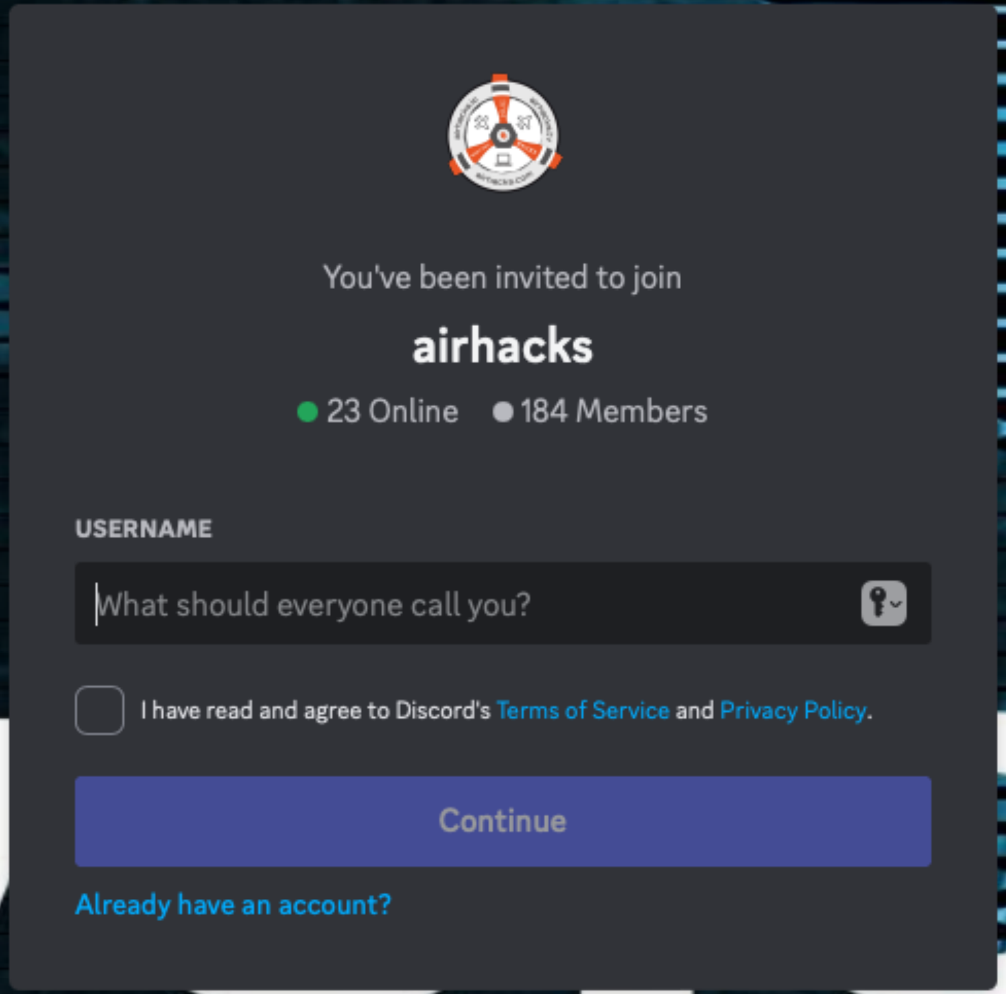
Apple 2c at ericsson.com, building software with components, writing about science of component based development, devops in 1976, function and logic programming in 1983, imperative, logic and functional programming, leaving Ericsson, the Rational Objectory airhacks.TV

with the time machine, “100 episodes ago segment”

…any questions left?

youtube.com/
@bienadam

youtube.com/bienadam/shorts


You've been invited to join

airhacks

● 23 Online ● 184 Members

USERNAME

What should everyone call you?

I have read and agree to Discord's Terms of Service and Privacy Policy.

Continue

Already have an account?

airhacks.live

NEW https://discord.gg/airhacks

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

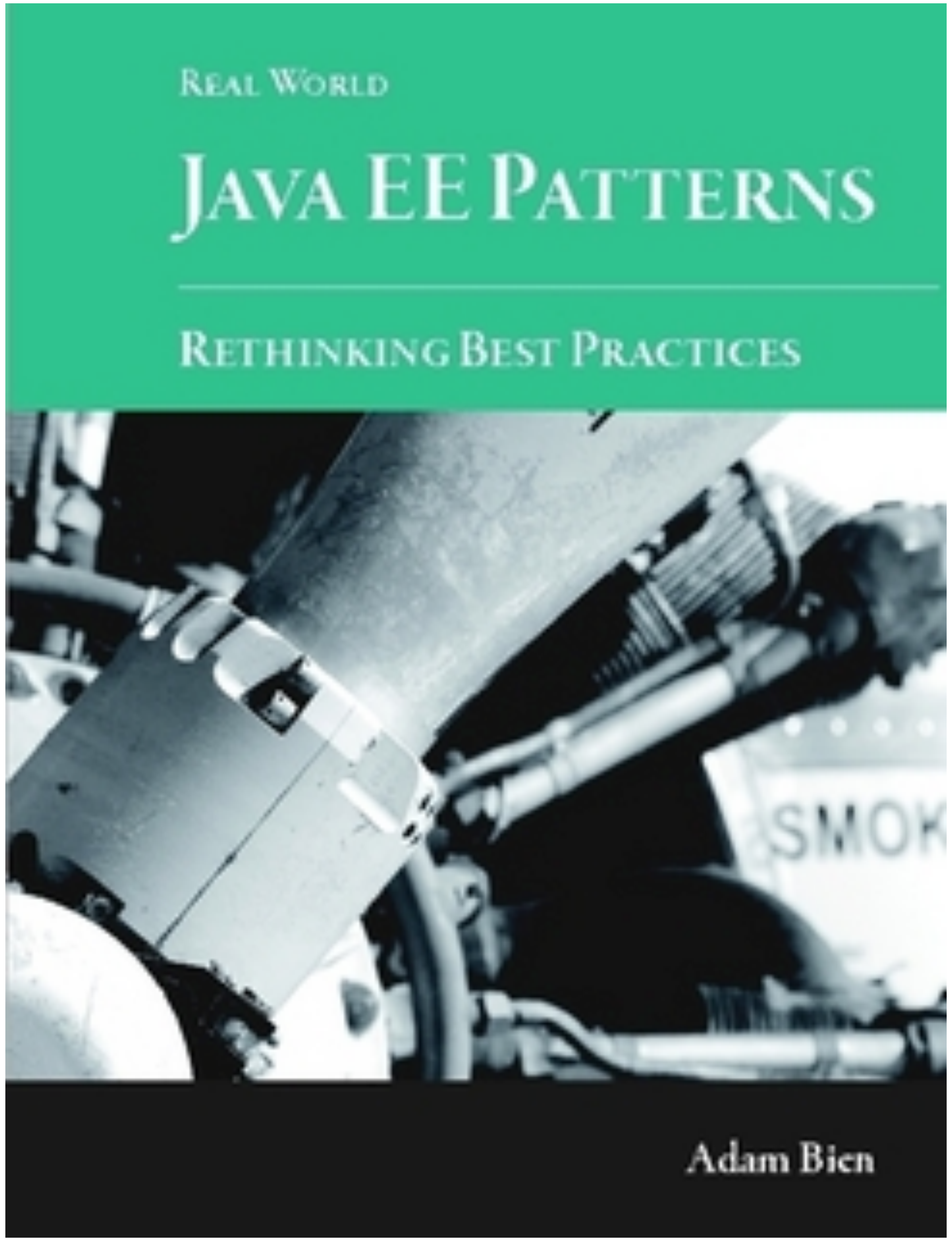
Live, Virtual Online Workshops, Summer 2025:

- LLM / GenAI Patterns, Architectures and Use Cases with Java, 10 July 2025
- Hardcore Serverless Java on AWS, 17 July 2025

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

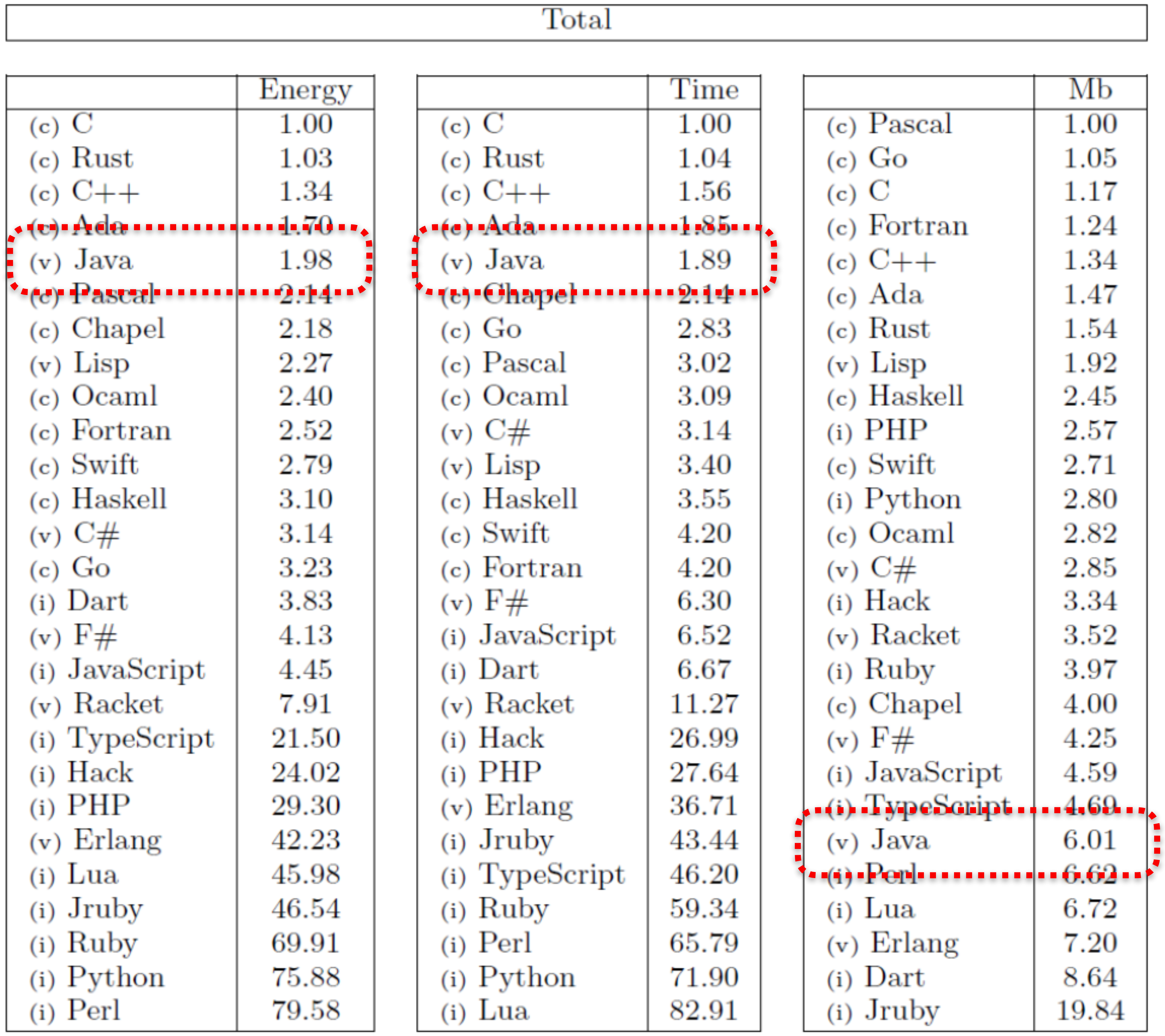
by Adam Bien

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


Real World
Java EE Patterns
Rethinking Best Practices
Adam Bien

(2012)

Java + Clouds = ?



Total

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

GraalVM™

reduces RAM footprint

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

In practice, across its many manifestations, postmodernism shares an attitude of skepticism towards grand explanations and established ways of doing things. In art, literature, and architecture, it blurs boundaries between styles and genres, and encourages freely mixing elements, challenging traditional distinctions like high art versus popular art. In science, it emphasizes multiple ways of seeing things, and how our cultural and personal backgrounds shape how we see the world, making it impossible to be completely objective. In philosophy, education, history, politics, and many other fields, it encourages critical re-examination of established institutions and social norms, embracing diversity and breaking down disciplinary boundaries. Though these ideas weren't strictly new, postmodernism amplified them, using an often playful, at times deeply critical, attitude of pervasive skepticism to turn them into defining features.[15][16][17][18][19][20]

how to start

first steps

- pick boring technology
- minimize dependencies
- automate build
- automate deployment
- setup CI
- e2e test after every commit
- use standards (no / easier migrations)

common sense

- “main” only, no branches
- system / API tests first with coverage
- stress- over system- over unit-tests
- no problems, no patterns
- decoupling is not a best practice
- write simple code
- use standards 👉 more vacations

“the enterprise startup”

no mocks, high fidelity

interesting / productive
discussions only

micro shift left

errorhandling and logging

copy and paste vs. reuse

the LLM impact and MDA

system tests as application

vision is not enough

we are doing this because…

there should be a reason for everything

mission statement is useful, but:

Strategy

Diagnosis (long development cycles and slow time-to-market)

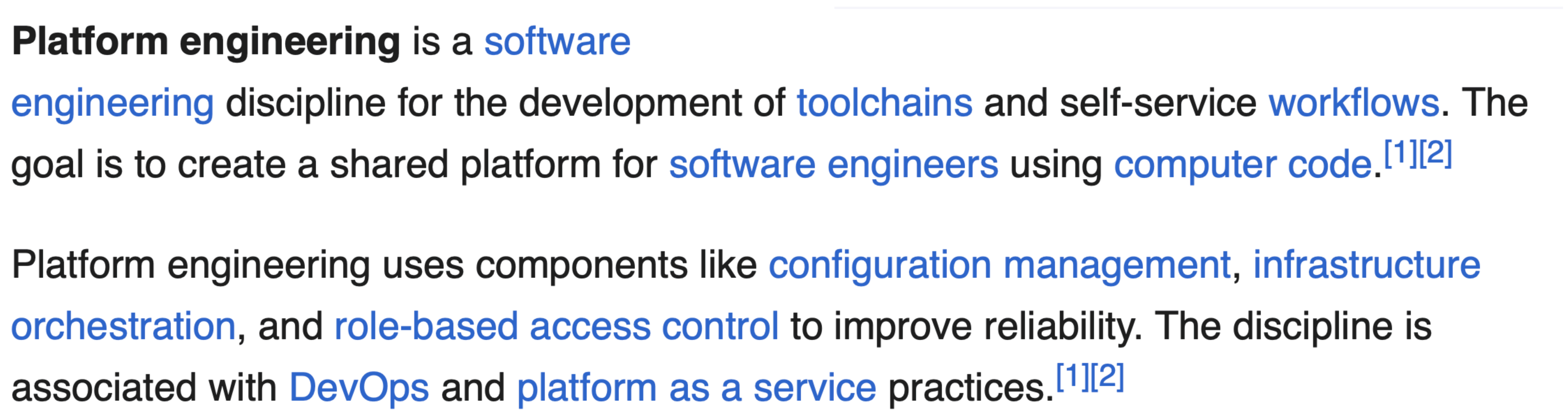
Guiding Policy (focus on business logic, eliminate superfluous code)

Action Plan (remove interfaces, meaningless UTs, DTOs, Impls,…)

https://en.wikipedia.org/wiki/Strategy

define a clear strategy
…or you get 3 devs and > 100 µservices

platform engineering?



Platform engineering is a software engineering discipline for the development of toolchains and self-service workflows. The goal is to create a shared platform for software engineers using computer code.[1][2]

Platform engineering uses components like configuration management, infrastructure orchestration, and role-based access control to improve reliability. The discipline is associated with DevOps and platform as a service practices.[1][2]

https://en.wikipedia.org/wiki/Platform_engineering

rethinking design

design

- domain first
- synchronous first (http)
- no 3rd-party dependencies
- “use the platform”
- “know your enemy”— (goal, problem)
- measure first then optimise

design

- Java SE features first
- write simple code first
- prefer Java SE libraries (Base64, String.isBlank, logging)
- Java 21, virtual threads, record classes, deconstruction
- Data Oriented Programming

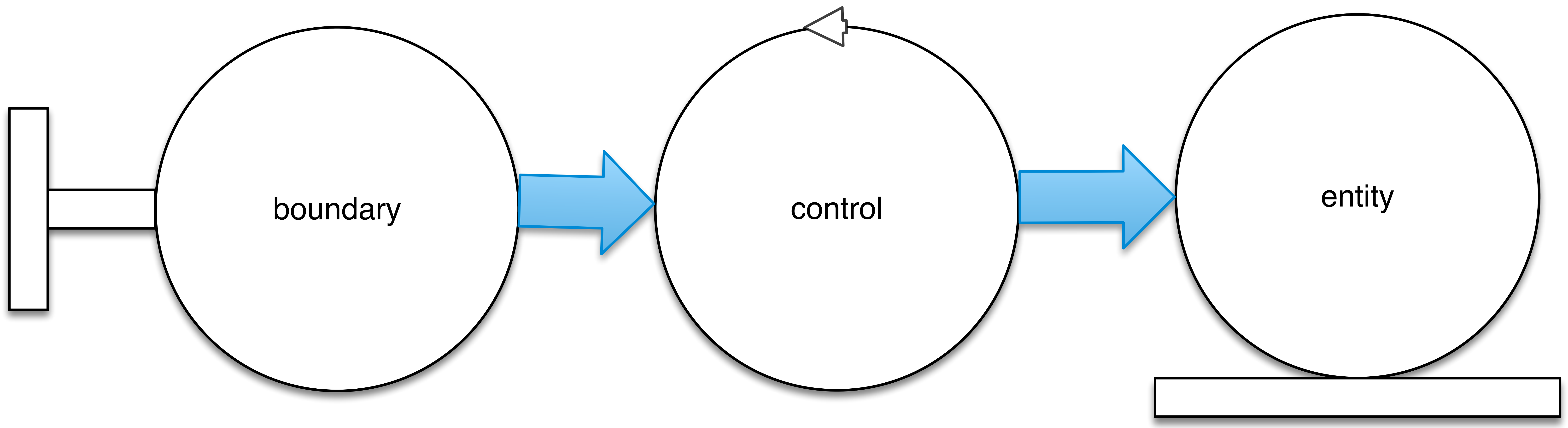
rethinking architecture

rethinking architecture

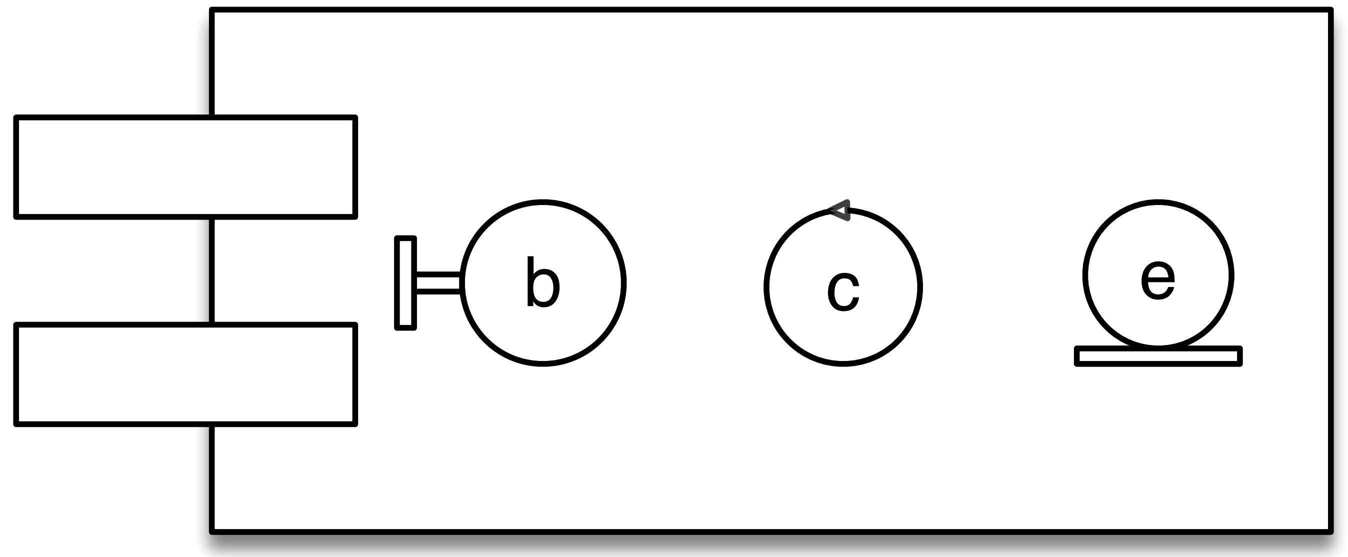
- KISS and YAGNI
- don’t distribute
- no distributed transactions
- asynchronous invocations for asynchronous use cases
- reactive programming for reactive use cases
- start with nice monoliths

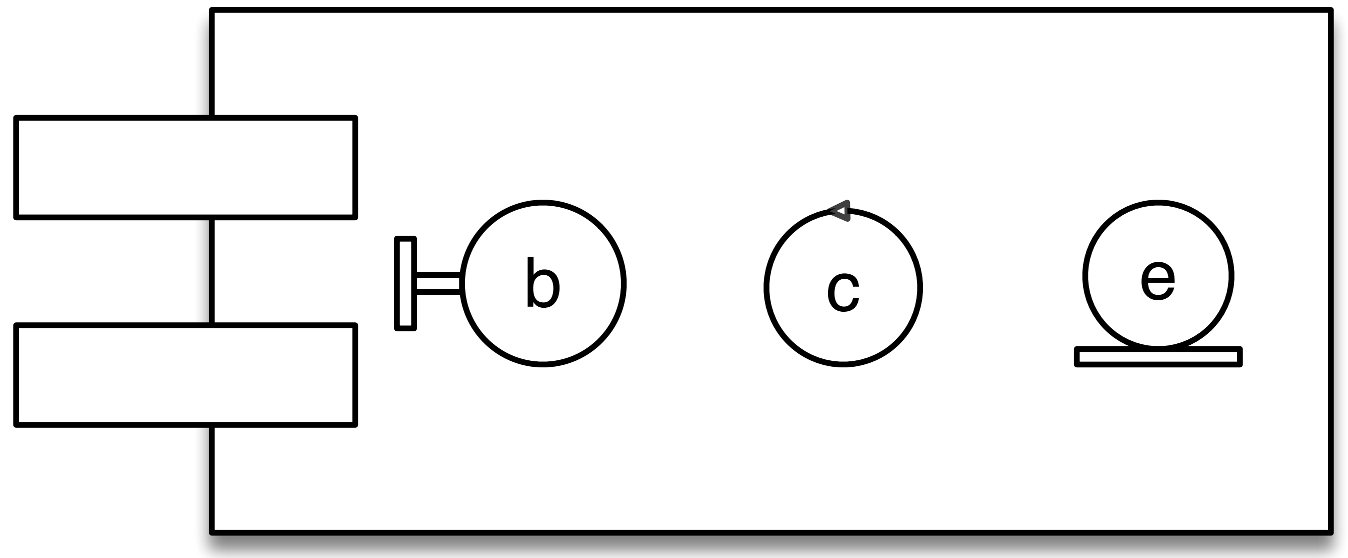
the ultimate internal structure

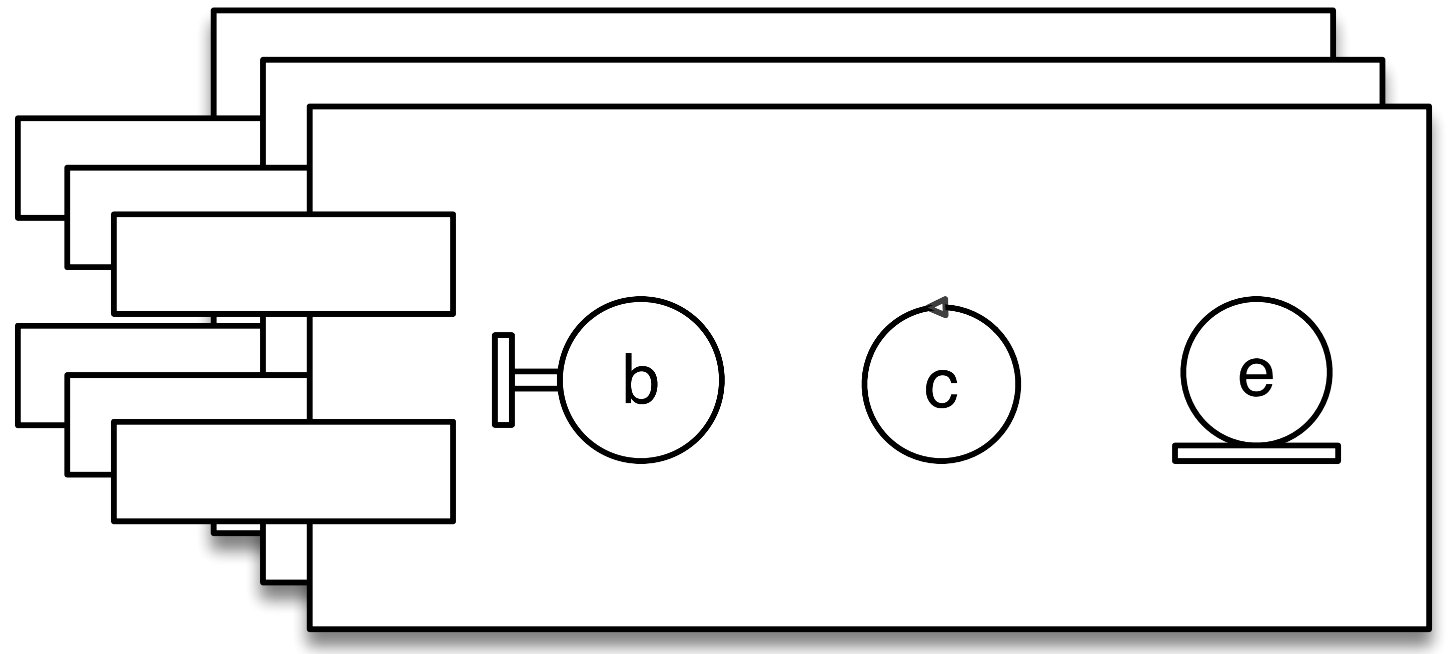
Boundary Control Entity



Boundary Control Entity







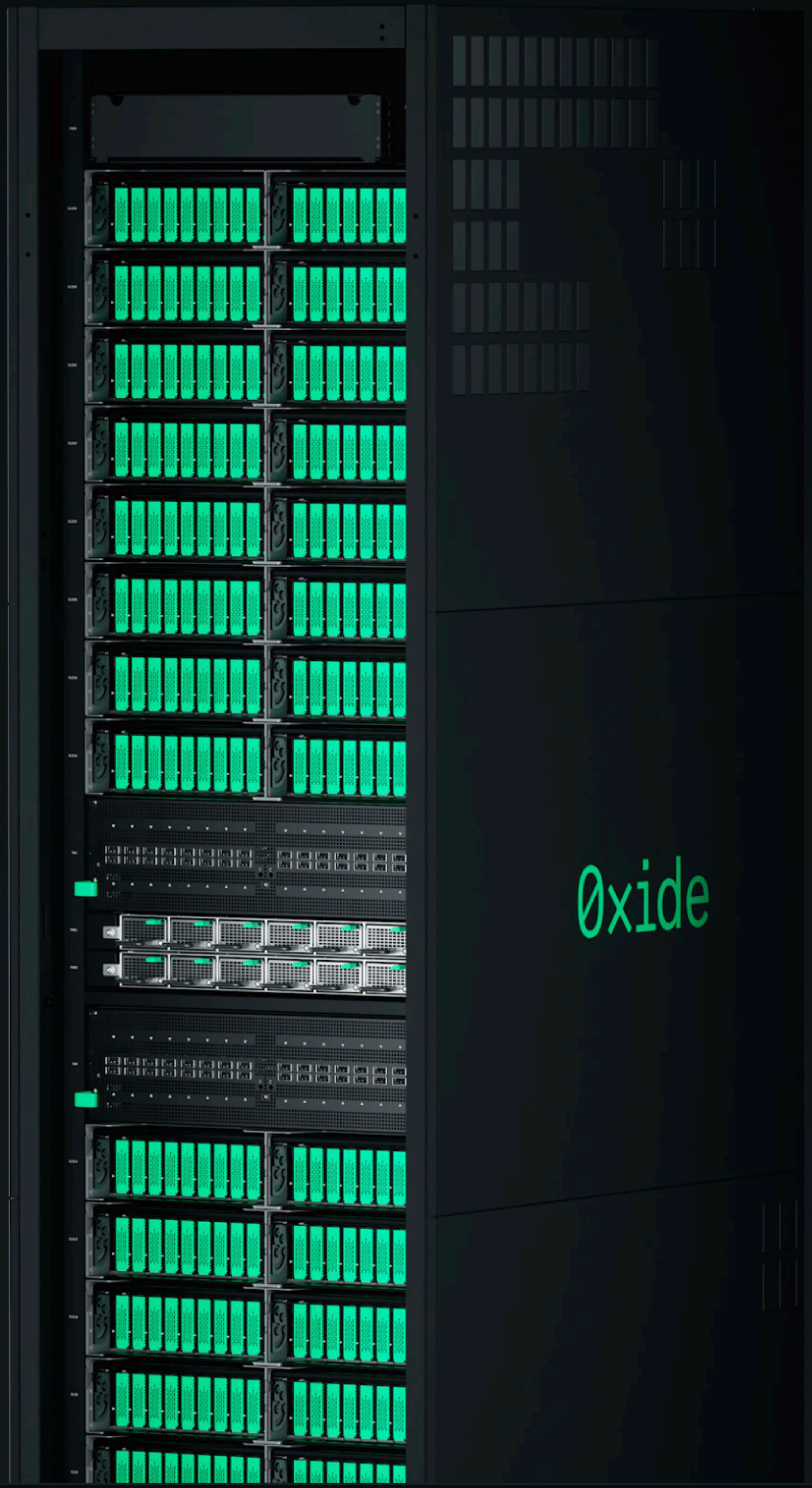
on-premise architectures

Sun Starfire 10000

Oxide Cloud Computer

No Cables. No Assembly.
Just Cloud.

CONTACT SALES GET OUR NEWSLETTER



java on premises

- Intel / AMD dominance
- high-end hardware, multi-CPU with a lot RAM
- project loom / virtual threads
- docker + layering
- docker compose
- GraalVM
- CraC
- Java Flight Recorder custom events

java on premises

- real time monitoring
- extensive logging
- supported openJDK distributions
- kubernetes operator automation

java on premises

- stateful architectures
- pooling / caching
- custom load balancing
- managed kubernetes or docker compose
- opensource services (PostgreSQL, Neon, Prometheus, Kafka)
- connections and resource pooling
- jlink

java on premises

- choose your high performance openSource JDKs: Azul Systems, BellSoft, GraalVM, SAP (…)
- use proprietary features: Azul Platform Prime, Azul Code Inventory, IBM JDK CRIU, GraalVM, Mandrel polyglot
- Panama, Babylon, Vector API
- Project Leyden, GC optimizations

public cloud architectures

“no plumbing,
business logic only”

simplicity over portability

time to migration vs. portability

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

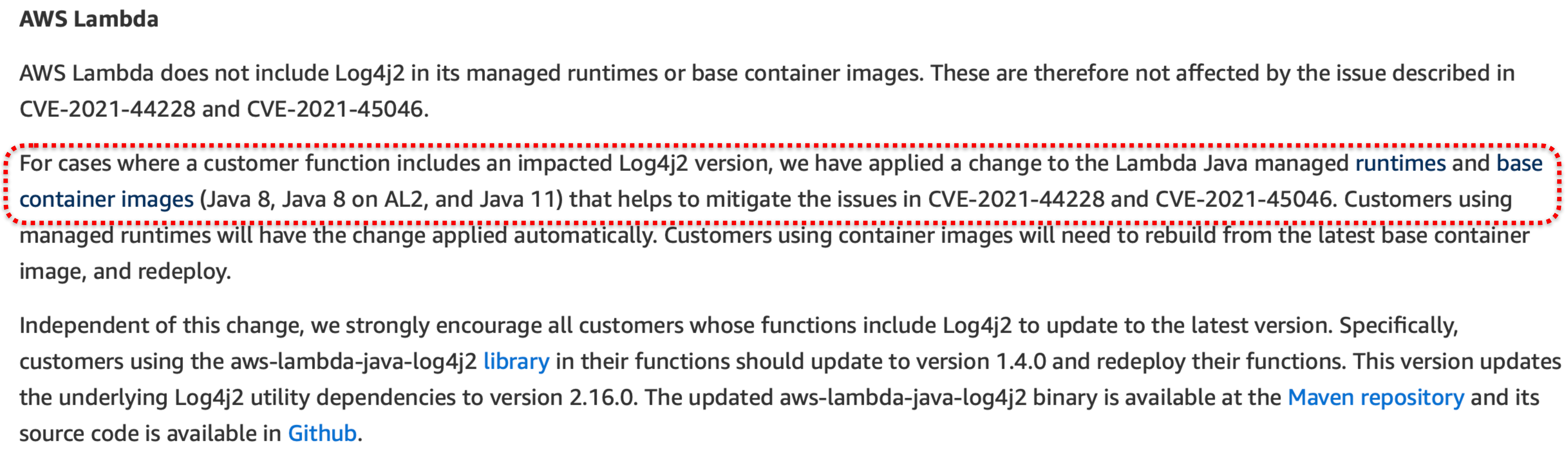
Amount of memory allocated
Enter the amount between 128 MB and 10 GB
Value: 2
Unit: GB

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



AWS Lambda

AWS Lambda does not include Log4j2 in its managed runtimes or base container images. These are therefore not affected by the issue described in CVE-2021-44228 and CVE-2021-45046.

For cases where a customer function includes an impacted Log4j2 version, we have applied a change to the Lambda Java managed runtimes and base container images (Java 8, Java 8 on AL2, and Java 11) that helps to mitigate the issues in CVE-2021-44228 and CVE-2021-45046. Customers using managed runtimes will have the change applied automatically. Customers using container images will need to rebuild from the latest base container image, and redeploy.

Independent of this change, we strongly encourage all customers whose functions include Log4j2 to update to the latest version. Specifically, customers using the aws-lambda-java-log4j2 library in their functions should update to version 1.4.0 and redeploy their functions. This version updates the underlying Log4j2 utility dependencies to version 2.16.0. The updated aws-lambda-java-log4j2 binary is available at the Maven repository and its source code is available in Github.

☁️ 👉cost driven architectures

serverless for enterprise

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

DEFINITION

NoOps (no operations)

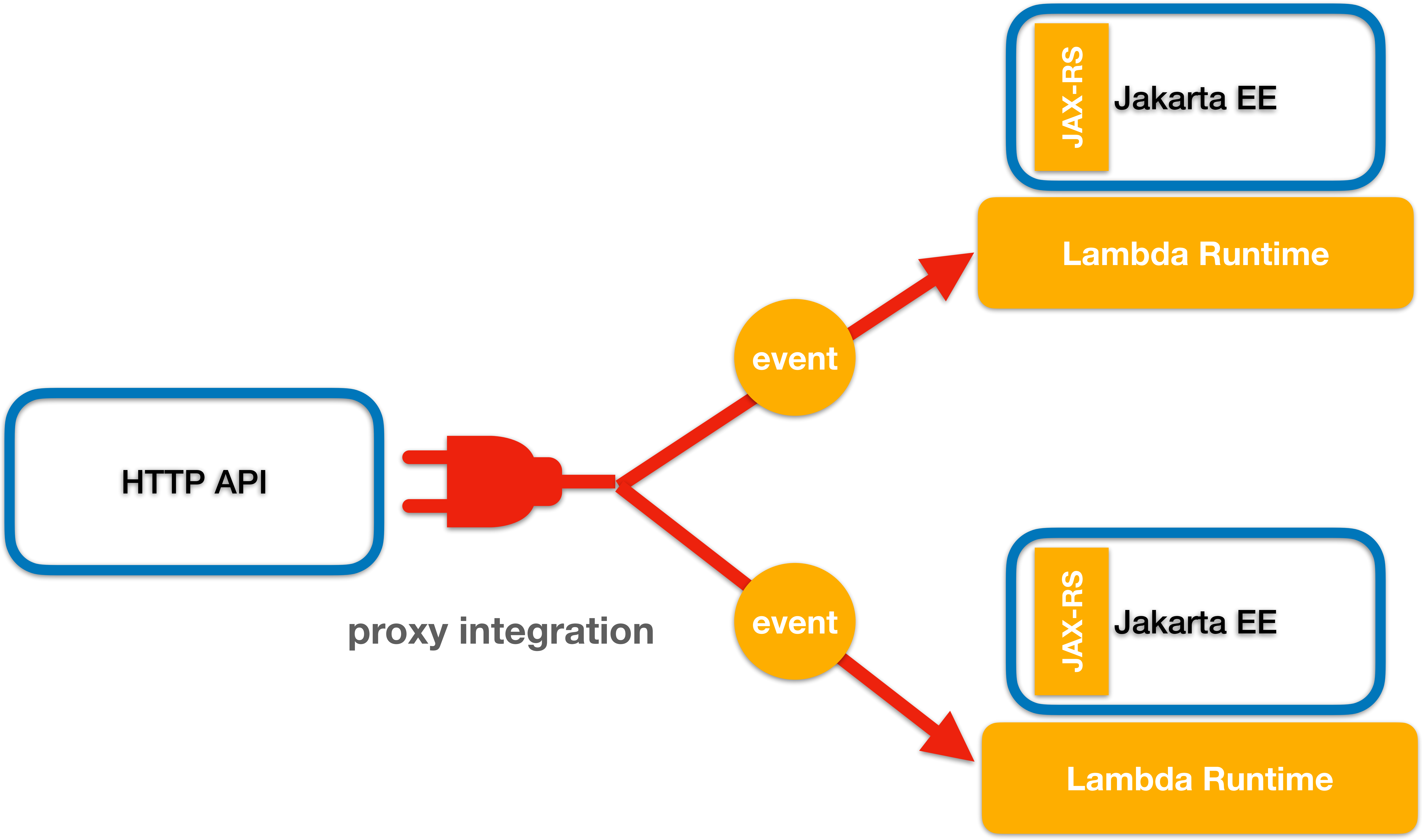
By Rahul Awati

What is NoOps (no operations)?

NoOps (no operations) is a concept that an IT environment can become so automated and abstracted from the underlying infrastructure that there's no need for a dedicated team to manage software in-house.

First coined by research and advisory company Forrester, the term describes the goal of NoOps as to "improve the process of deploying applications" so that "application developers will never have to speak with an operations professional again." Forrester's Mike Gualtieri called DevOps "a step backward" even though, in his short 2011 article, he wrote that he admires its goal to improve application release deployment processes.

Fat, Monolithic Function


JAX-RS
Jakarta EE
Lambda Runtime
event
HTTP API
proxy integration
event
JAX-RS
Jakarta EE
Lambda Runtime

Serverless Archive: SAR (~~EAR~~😊)

lift and shift?

now …some code

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Summer 2025:

- LLM / GenAI Patterns, Architectures and Use Cases with Java, 10 July 2025
- Hardcore Serverless Java on AWS, 17 July 2025

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