

**CLOUD NATIVE  
COMPUTING FOUNDATION**

# Hybrid Serverless Development using Quarkus

---

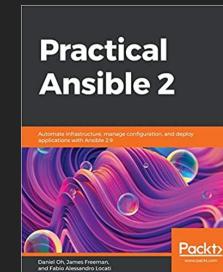
Daniel Oh  
Principal Technical Marketing Manager  
Red Hat

# About Me



## Daniel Oh

- Principal Technical Marketing Manager at Red Hat 
- Cloud Native App Development
- Agile & DevOps practices
- Ambassador for CNCF  and DevOps Institute 
- Opensource.com Correspondents
- Public Speaker & Developer
- Author of [Practical Ansible 2](#)



 @danieloh30

 [bit.ly/danielohtv](https://bit.ly/danielohtv)

 danieloh30



IN THE BEGINNING...

1999

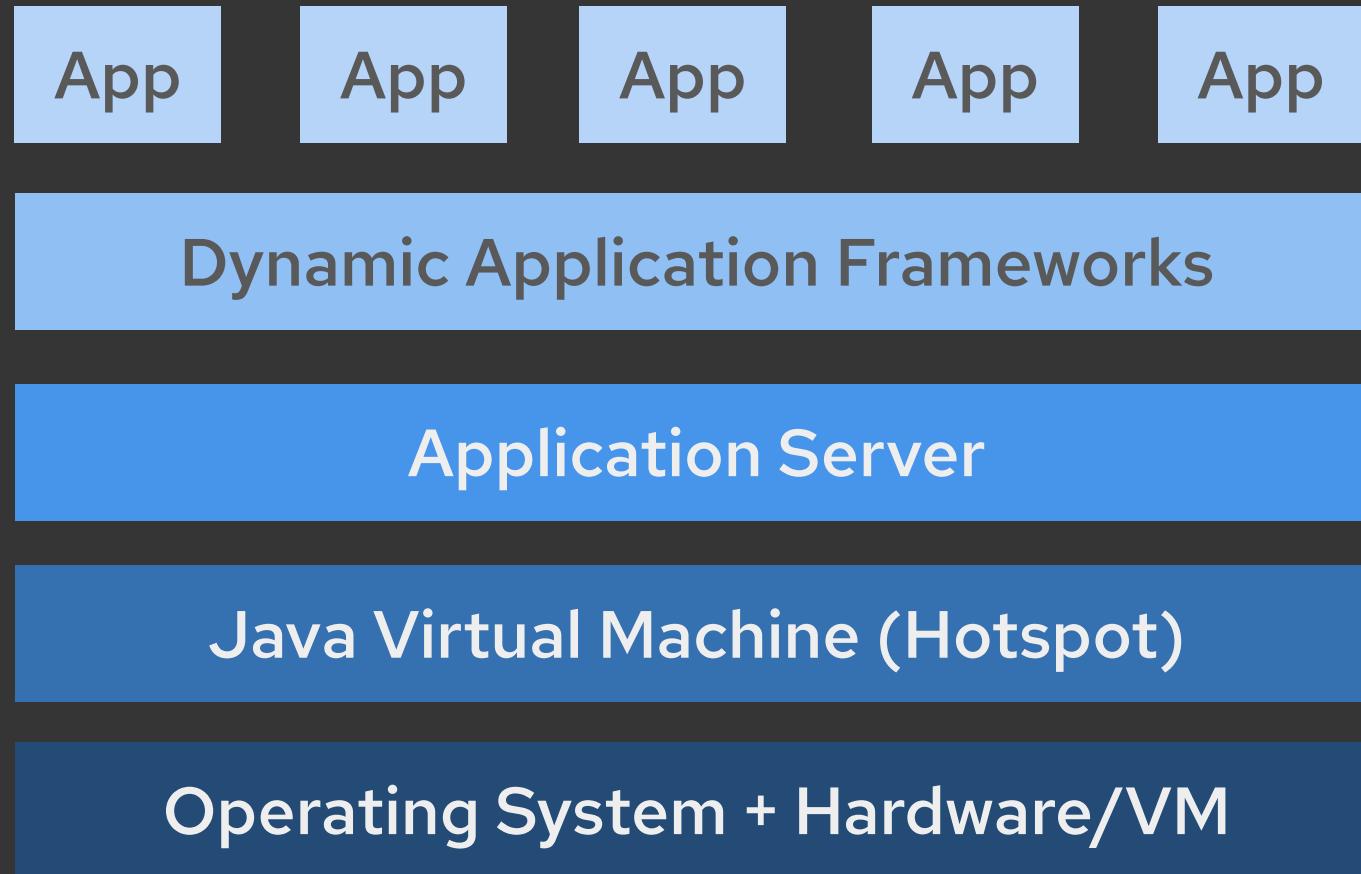
# Cost of a Java-based Web App circa 1999

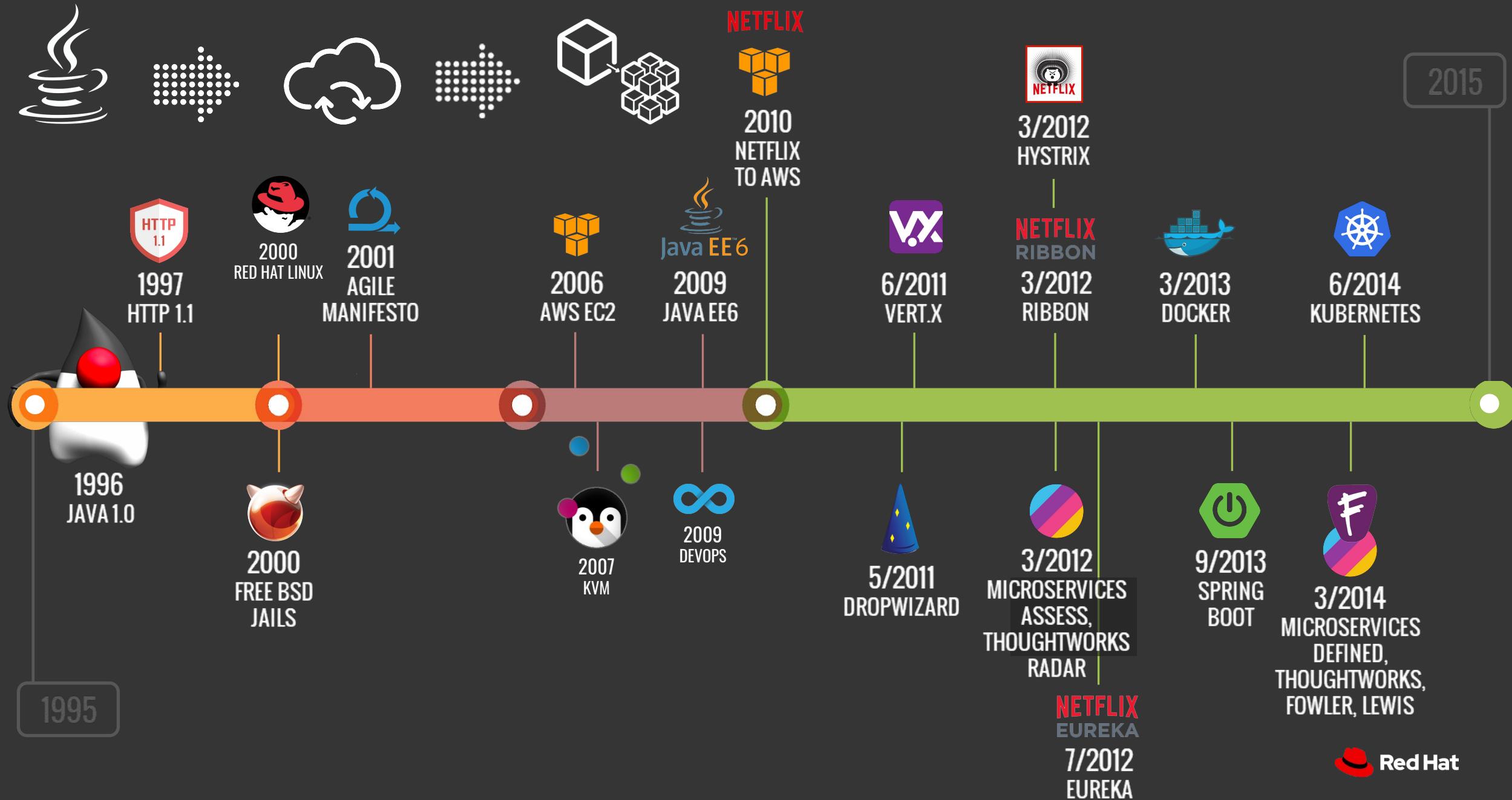
- \$18,000 Sun Sparc App Server Box (4 CPUs, 2GB of RAM)
  - + \$60,000 BEA Weblogic
  - + \$92,000 Sun Sparc DB Server Box (8 CPUs)
  - + \$243,000 Oracle RDBMS
  - + \$50,000 Symantec Visual Café for 10 developers
- 

\$463,000 (capex) + ~\$80,000 annual maint (opex)

# 1999 Enterprise Java Stack

Architecture: **Monolith**  
Deployment: **Multi-app,  
App server**  
Lifecycle: **Months**  
Memory: **1GB+ RAM**  
Startup: **10s of secs**





m5ad.4xlarge	16	N/A	64 GiB	2 x 300 NVMe SSD	\$0.824 per Hour
m5ad.12xlarge	48	N/A	192 GiB	2 x 900 NVMe SSD	\$2.472 per Hour
m5ad.24xlarge	96	N/A	384 GiB	4 x 900 NVMe SSD	\$4.944 per Hour
m5d.large	2	8	8 GiB	1 x 75 NVMe SSD	\$0.113 per Hour
m5d.xlarge	4	16	16 GiB	1 x 150 NVMe SSD	\$0.226 per Hour
m5d.2xlarge	8	31	32 GiB	1 x 300 NVMe SSD	\$0.452 per Hour



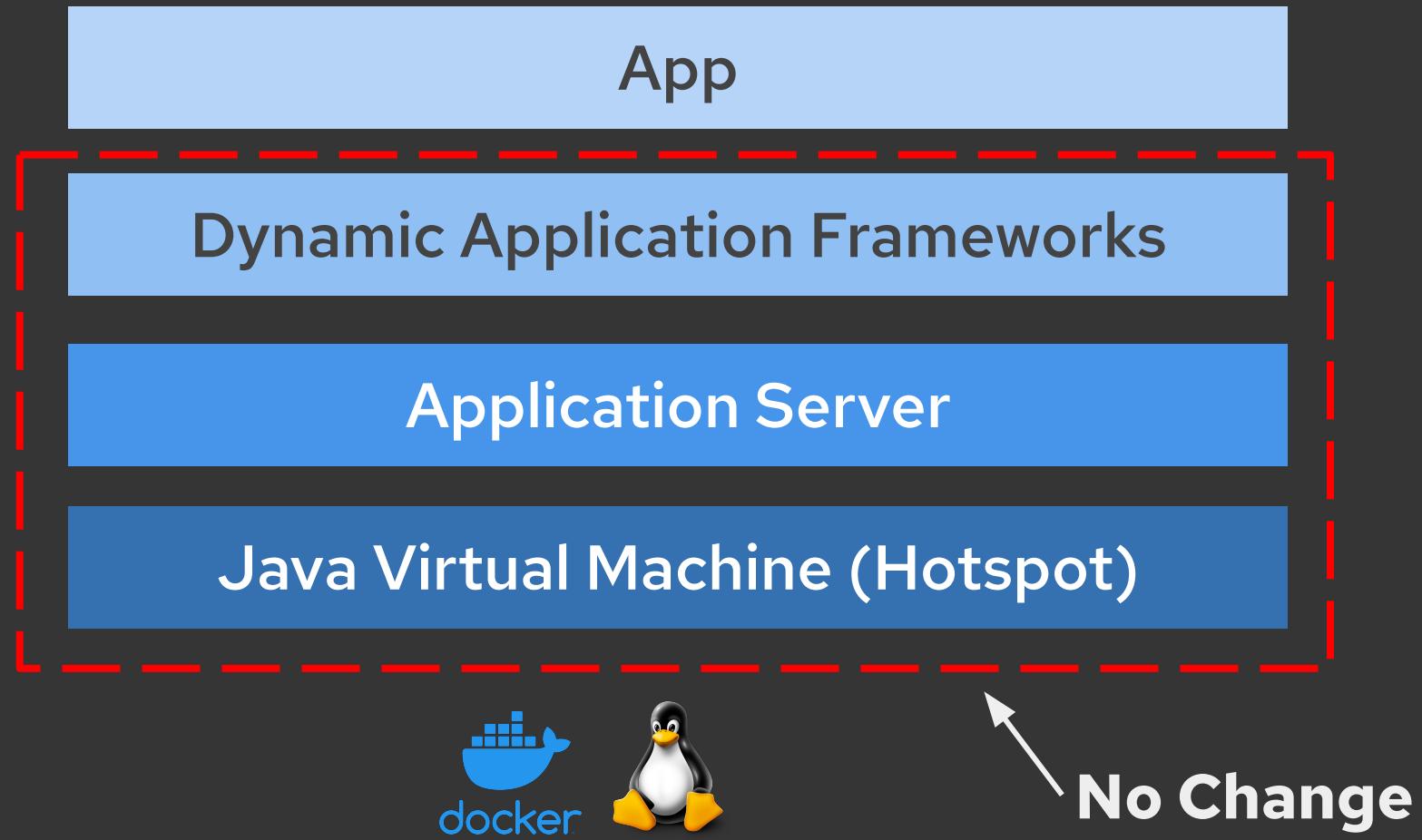
MEMORY	VCPUS	SSD DISK	TRANSFER	PRICE
1GB	1 vCPU	25 GB	1TB	\$5/mo \$0.007/hr
2 GB	1 vCPU	50 GB	2 TB	\$10/mo \$0.015/hr
3 GB	1 vCPU	60 GB	3 TB	\$15/mo \$0.022/hr
2 GB	2 vCPUs	60 GB	3 TB	\$15/mo \$0.022/hr
1GB	3 vCPUs	60 GB	3 TB	\$15/mo \$0.022/hr
4 GB	2 vCPUs	80 GB	4 TB	\$20/mo \$0.030/hr
8 GB	4 vCPUs	160 GB	5 TB	\$40/mo \$0.060/hr
16 GB	6 vCPUs	320 GB	6 TB	\$80/mo \$0.119/hr

INSTANCE	VCPU	RAM	TEMPORARY STORAGE	PAY AS YOU GO
D2 v3	2	8 GiB	50 GiB	\$0.096/hour
D4 v3	4	16 GiB	100 GiB	\$0.192/hour
D8 v3	8	32 GiB	200 GiB	\$0.384/hour



# “Cloud Native” Java Stack

Architecture: **Microservices**  
Deployment: **Single App,  
Container**  
Lifecycle: **Days**  
Memory: **100MBs+ RAM**  
Startup: **Seconds**



A wide-angle photograph of a massive concrete dam. A powerful stream of white water is cascading down the right side of the dam, creating a large, turbulent mist. The dam itself is a dark grey concrete structure with multiple levels and a textured surface. It is situated on a rocky, brown hillside with sparse green vegetation. The sky is clear and blue.

Designed for  
Throughput

At the expense  
of **footprint**



Designed to be long-running

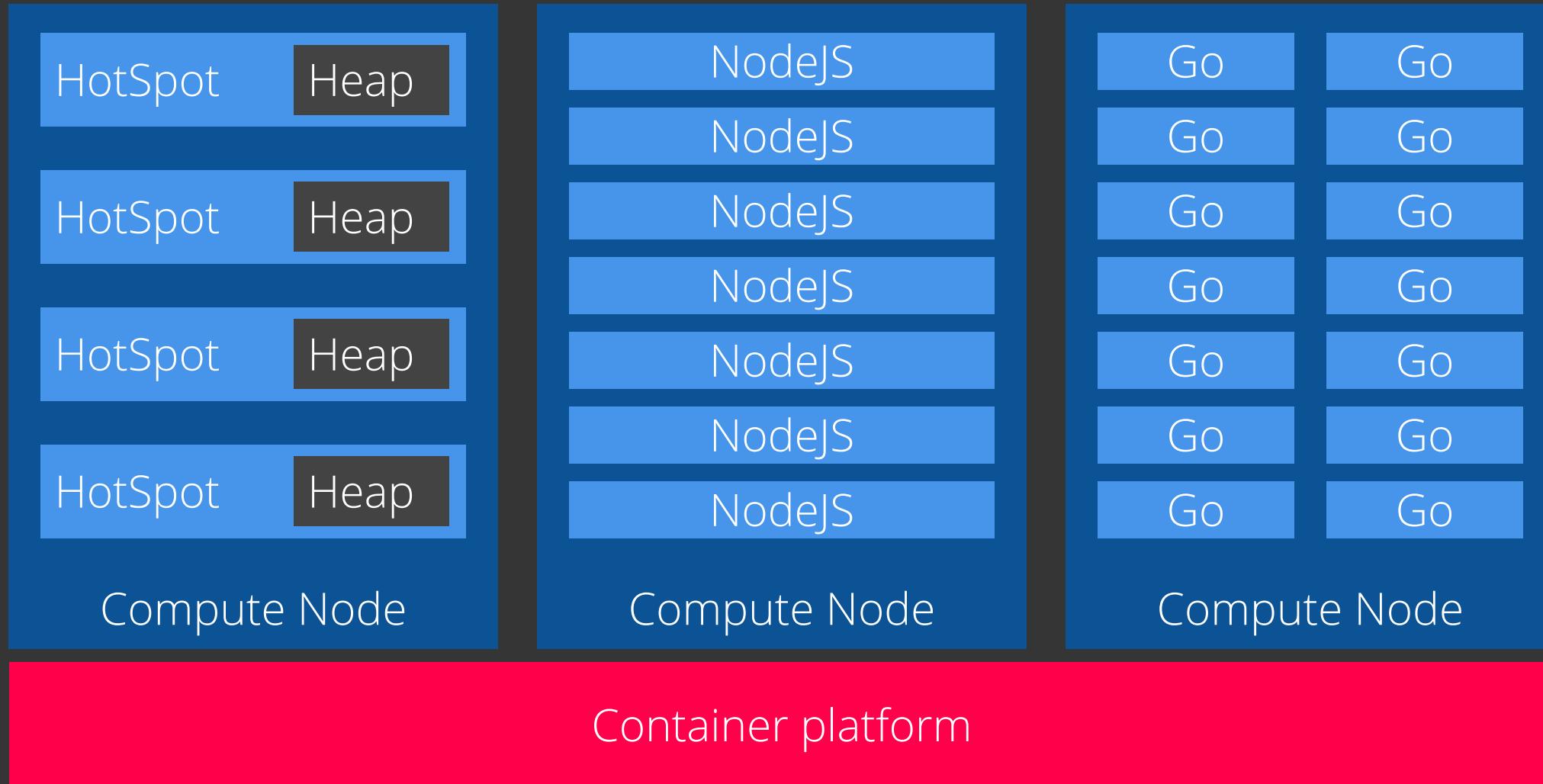
At the expense of startup speed



Rich dynamic behavior  
built for **mutable**  
systems

Yet containers  
are primarily  
**immutable**

# Java + Containers: The Hidden Truth



# Languages used on AWS Lambda

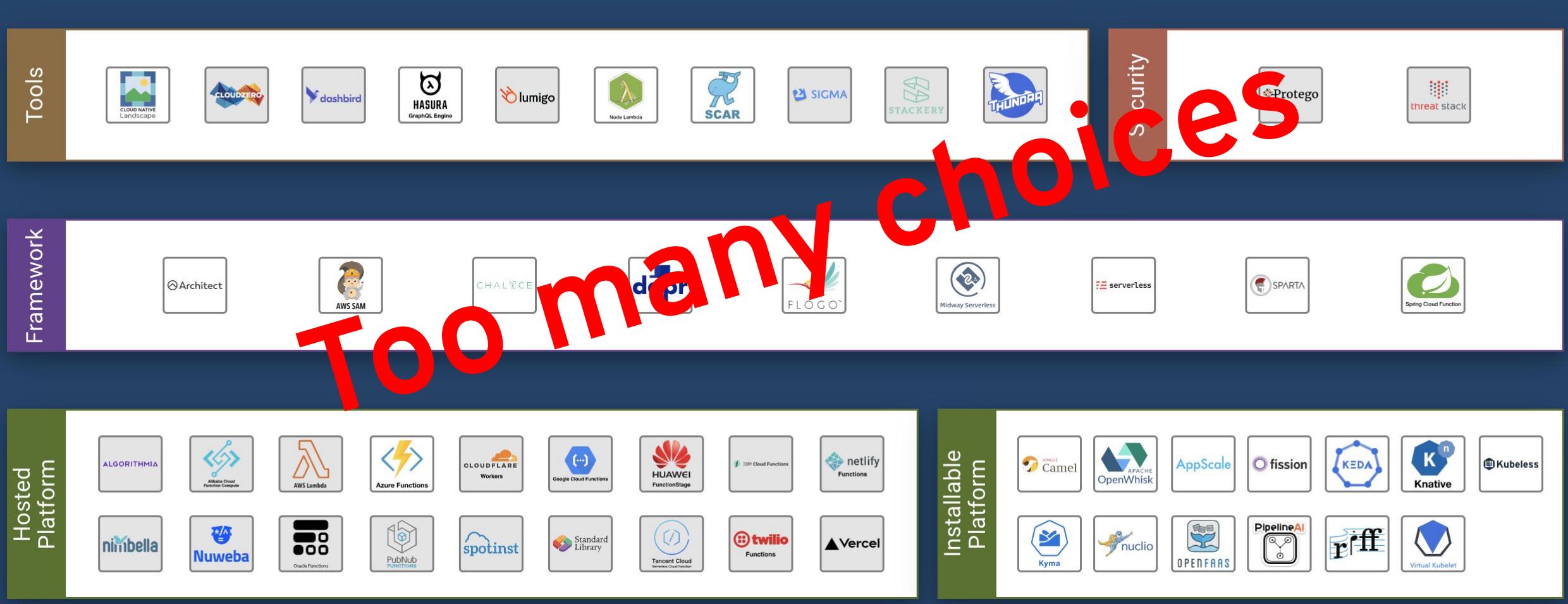


Source: <https://newrelic.com/resources/ebooks/serverless-benchmark-report-aws-lambda-2020>



# Serverless Landscape

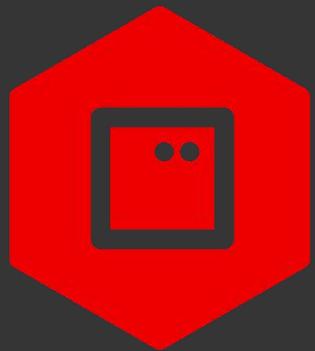
Too many choices



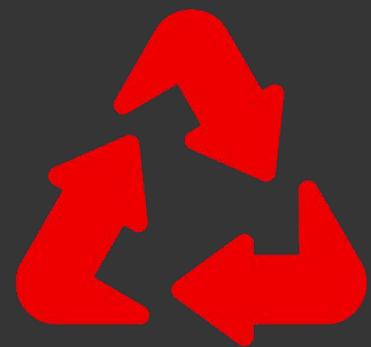
Source: <https://landscape.cnf.io/format=serverless&zoom=150>



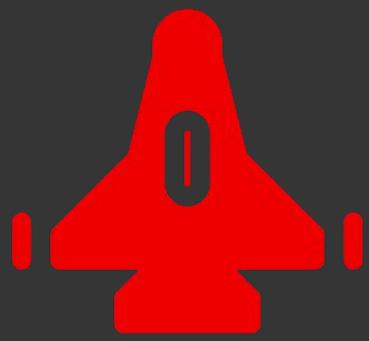
# Java has tried to pivot before



LINUX CONTAINERS



SHENANDOAH GC



ECLIPSE OPENJ9



GCJ

---

MEMORY UTILIZATION

---

CPU UTILIZATION

---

JIT AS A SERVICE

---

DALVIK

---

AVIAN

WE  
NEED  
SOMETHING  
DIFFERENT



# QUARKUS

An Open Source  
stack to write Java apps



Cloud Native



Microservices

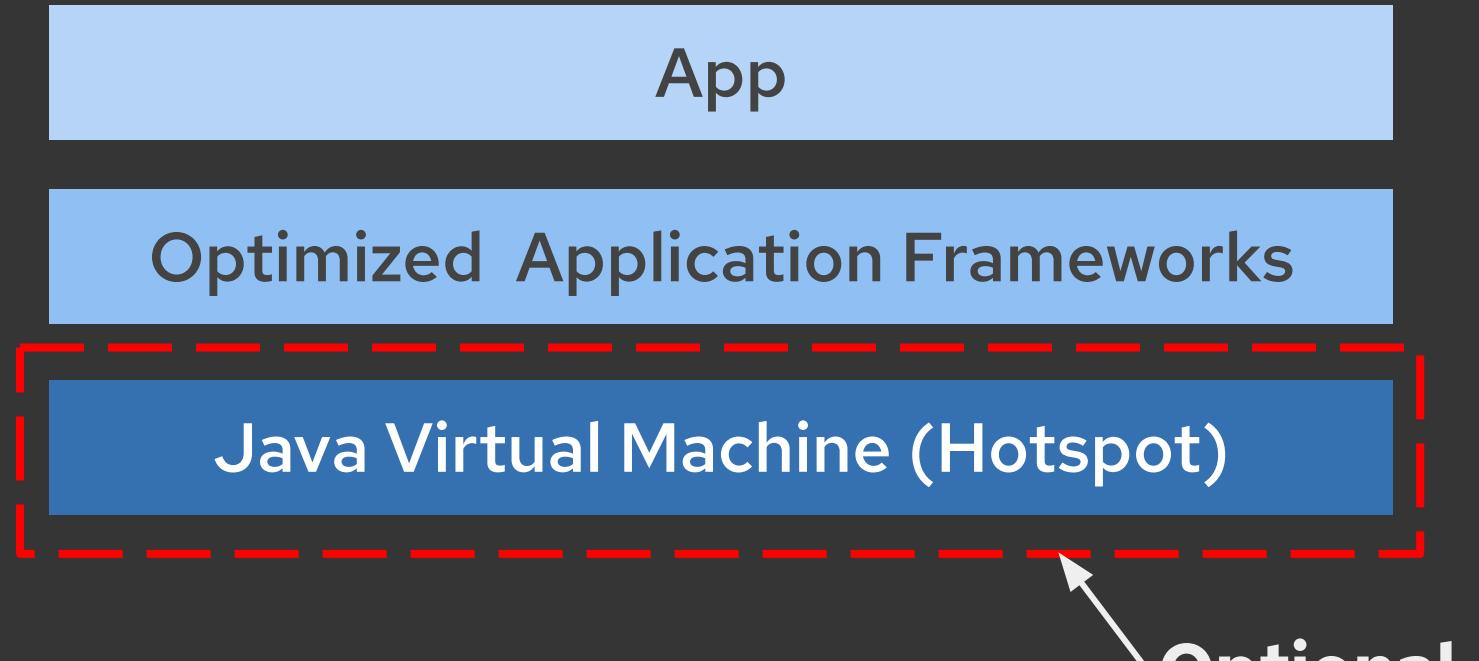


Serverless

# Quarkus - Optimizing the Stack

Architecture: **Microservices**,  
Deployment: **Serverless**

Single App  
Lifecycle: **ms to Days**  
Memory: **10MBs+ RAM**  
Startup: **Milliseconds**

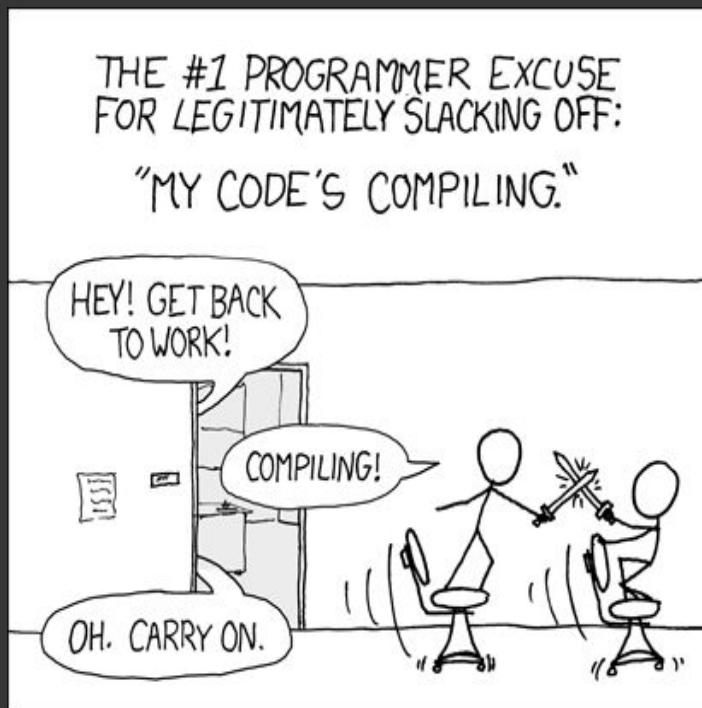


# How does a Framework start?

## Build the app

---

1. Bytecode compilation
2. Packaging (Maven, Gradle)



## Run the app

---

1. Load config files and parse them
2. Load classes, enable/disable features
3. Verify bytecode, warm up the JIT
4. Scan classes to process annotations
5. Build framework meta-models, proxies
6. Start the management (threads, pools)

# How does Quarkus start?

## Build the app

---

1. Bytecode compilation
2. Load config files and parse them
3. Process annotations
4. Build framework meta-models
5. Packaging (Maven, Gradle)
6. [Optional] Native compilation / dead code elimination

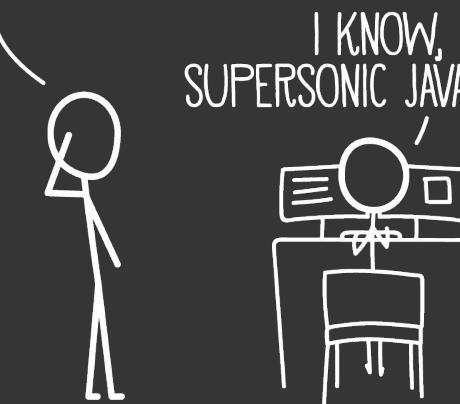
## Run the app

---

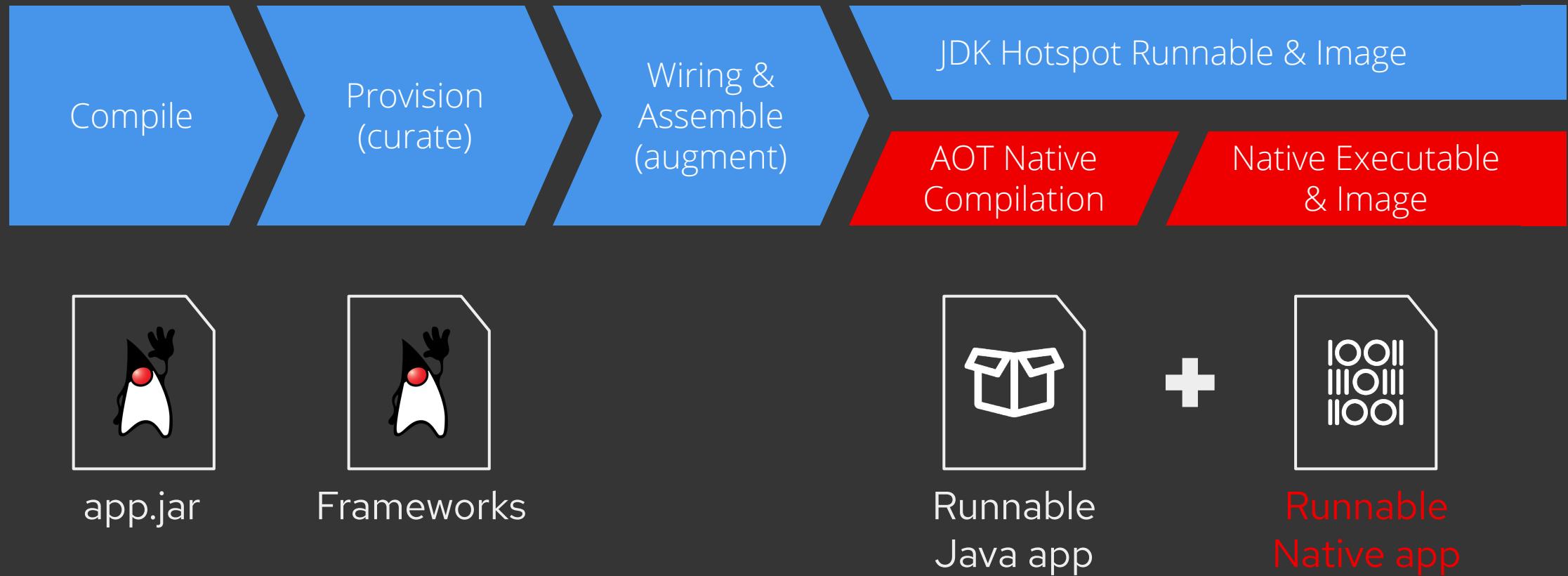
1. Start the management (threads, pools)

WAIT.  
SO YOU JUST SAVE IT,  
AND YOUR CODE IS RUNNING?  
AND IT'S JAVA?!

I KNOW, RIGHT?  
SUPERSONIC JAVA, FTW!



# Quarkus native compilation



# Quarkus Funqy for Serverless

## Cloud

### Quarkus Funqy

This guide explains basics of the Funqy framework, a simple portable cross-provider cloud function API.

### Quarkus Funqy HTTP

This guide explains Funqy's HTTP binding

### Quarkus Funqy Amazon Lambdas

This guide explains Funqy's Amazon Lambda binding

### Quarkus Funqy Amazon Lambdas HTTP

This guide explains Funqy's Amazon Lambda HTTP binding

### Quarkus Funqy Knative Events

This guide explains Funqy's Knative Events binding

### Quarkus Funqy Azure Functions HTTP

This guide explains Funqy's Azure Functions HTTP binding

### Quarkus Funqy Google Cloud Platform

This guide explains Funqy's Google Cloud Platform binding

#LiveCoding





# Red Hat Runtimes

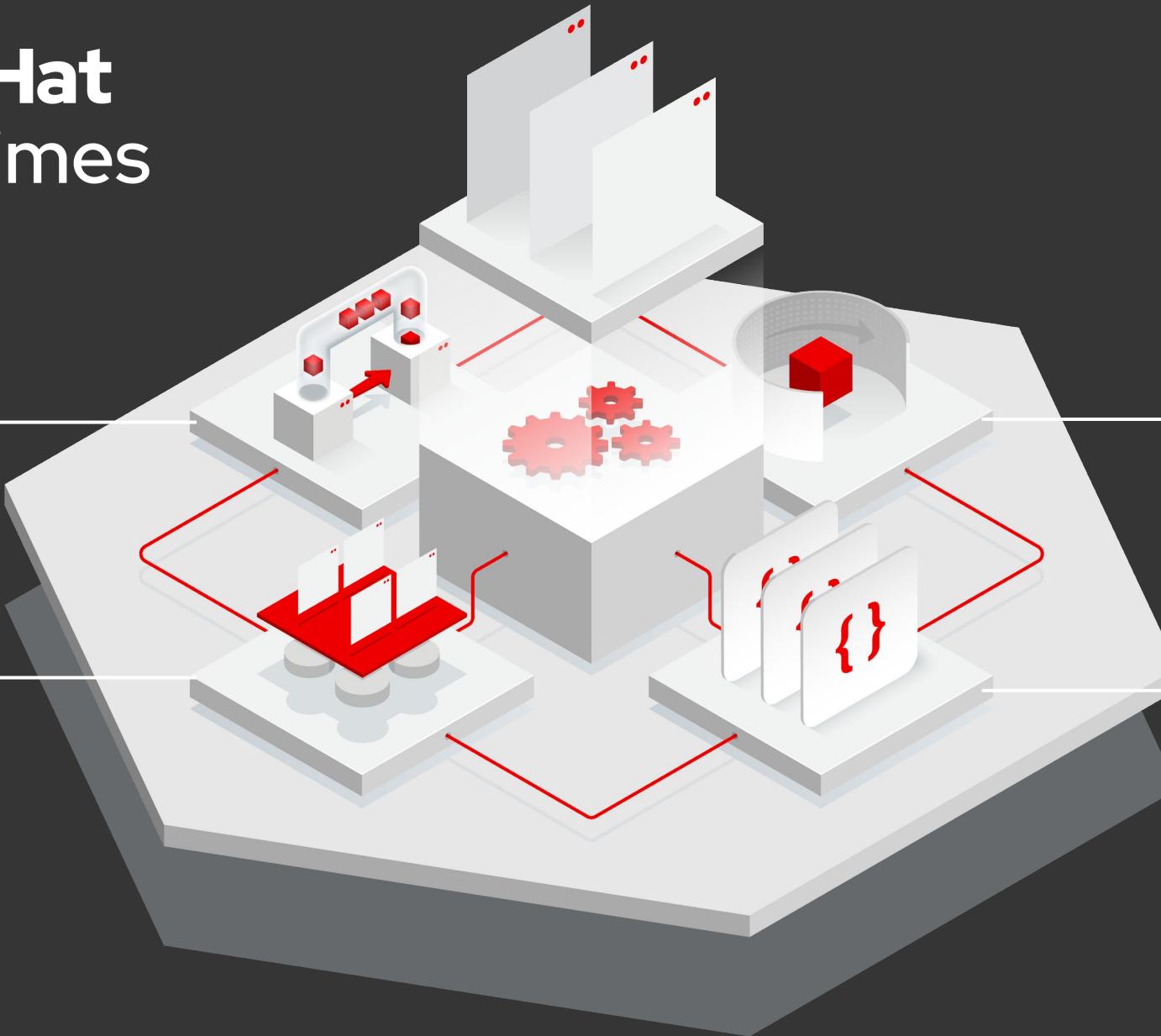
Red Hat  
AMQ Broker

Red Hat  
Data Grid

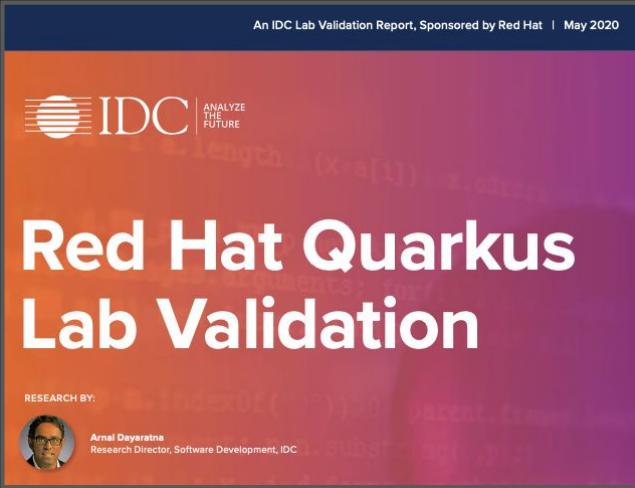
Red Hat  
Migration Toolkit  
for Applications

Red Hat  
Cloud-Native  
Runtimes  
including **Quarkus**

Red Hat  
Single Sign-On



# How to get started



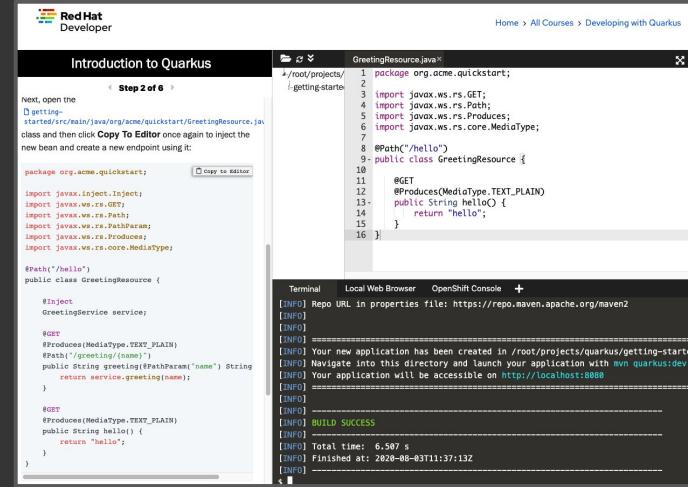
An IDC Lab Validation Report, Sponsored by Red Hat | May 2020

**IDC** ANALYZE THE FUTURE

## Red Hat Quarkus Lab Validation

RESEARCH BY:  
 Arnal Dayaratna  
Research Director, Software Development, IDC

[red.ht/idc-quarkus-study](http://red.ht/idc-quarkus-study)



Introduction to Quarkus

Step 2 of 6

Next, open the `getting-started/src/main/java/org/acme/quicksstart/GreetingResource.java` class and then click **Copy To Editor** once again to inject the new bean and create a new endpoint using it:

```
package org.acme.quickstart;

import javax.inject.Inject;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
```

```
#Path("hello")
public class GreetingResource {

    @Inject
    GreetingService service;

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    @Path("/{greet}/{name}")
    public String greeting(@PathParam("name") String name) {
        return service.greeting(name);
    }

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String hello() {
        return "Hello";
    }
}
```

GreetingResource.java

Terminal Local Web Browser OpenShift Console +

```
[INFO] Repo URL in properties file: https://repo.maven.apache.org/maven2
[INFO]
[INFO]
[INFO] =====
[INFO] Your new application has been created in /root/projects/quarkus/getting-started
[INFO] Navigate into this directory and launch your application with mvn quarkus:dev
[INFO] Your application will be accessible on http://localhost:8080
[INFO]
[INFO] =====
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 6.587 s
[INFO] Finished at: 2020-08-03T11:37:13Z
[INFO]
```

[bit.ly/try-quarkus](http://bit.ly/try-quarkus)



QUARKUS

Configure your application details

Group: org.acme  
Artifact: code-with-quarkus  
Build Tool: Maven

Generate your application (C + ⌘)

Pick your extensions

RESTEasy, Hibernate ORM, Web...

Selected Extensions

This page will help you bootstrap your Quarkus application and discover its extension ecosystem.

Explore the wide breadth of technologies Quarkus applications can be made with. Generate your application!

[Missing a feature? Found a bug? We are listening for feedback]

Web

RESTEasy JAX-RS INCLUDED

[code.quarkus.io](http://code.quarkus.io)



# Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

 [linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)

 [facebook.com/redhatinc](https://www.facebook.com/redhatinc)

 [youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)

 [twitter.com/RedHat](https://twitter.com/RedHat)

#### Image Credits

Waterfall: [https://unsplash.com/photos/-5mHMfz\\_CMY](https://unsplash.com/photos/-5mHMfz_CMY)

Marathon: Märtinš Zemlickis [https://unsplash.com/photos/-5mHMfz\\_CMY](https://unsplash.com/photos/-5mHMfz_CMY)

Chameleon: Ante Hamersmit [https://unsplash.com/photos/-5mHMfz\\_CMY](https://unsplash.com/photos/-5mHMfz_CMY)

Programmer Cartoon: <https://xkcd.com/303/>

Coding: <https://unsplash.com/photos/LJ9KY8plH3E>