Red Hat Day Events

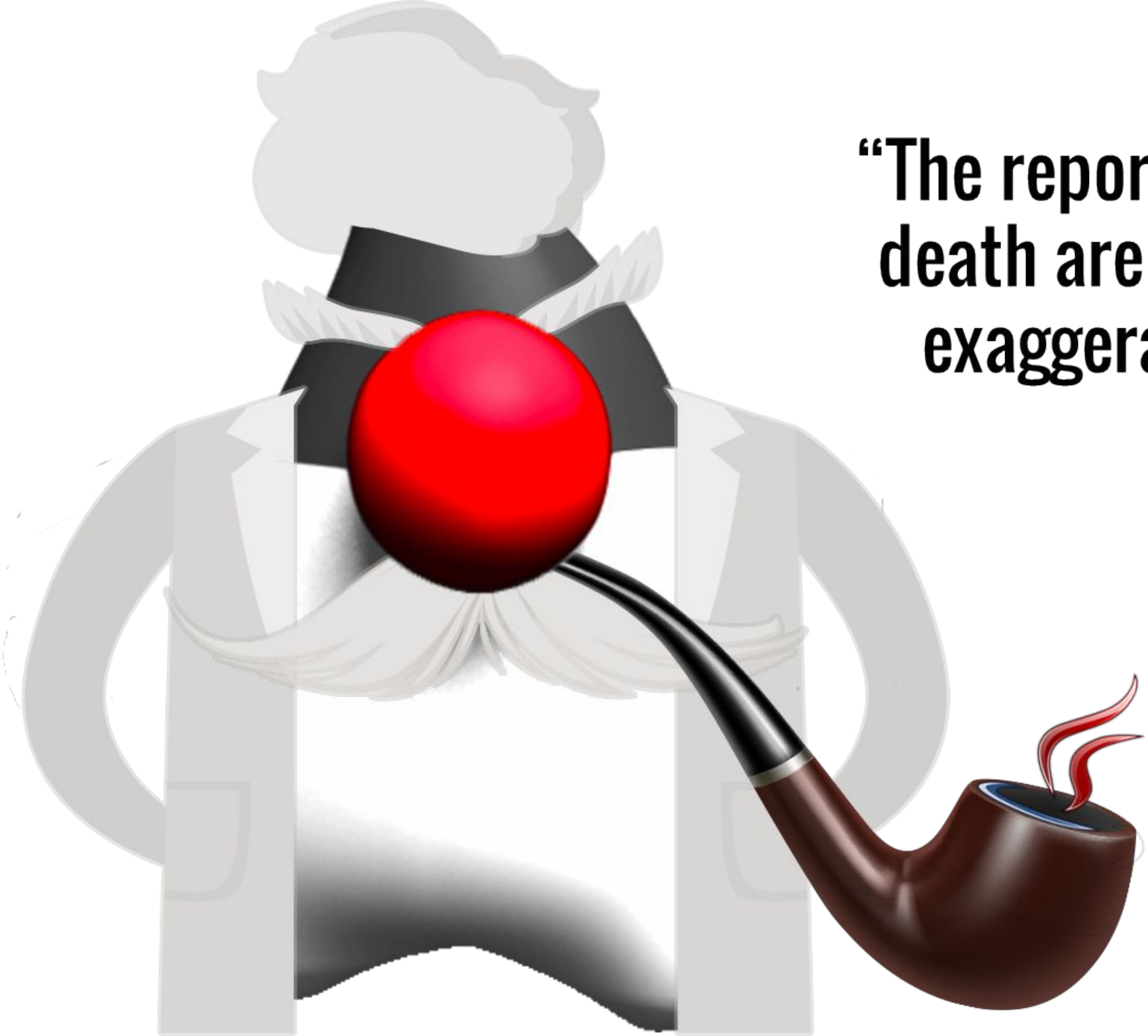
January 30, Vancouver

OpenShift
The Platform for Big Ideas

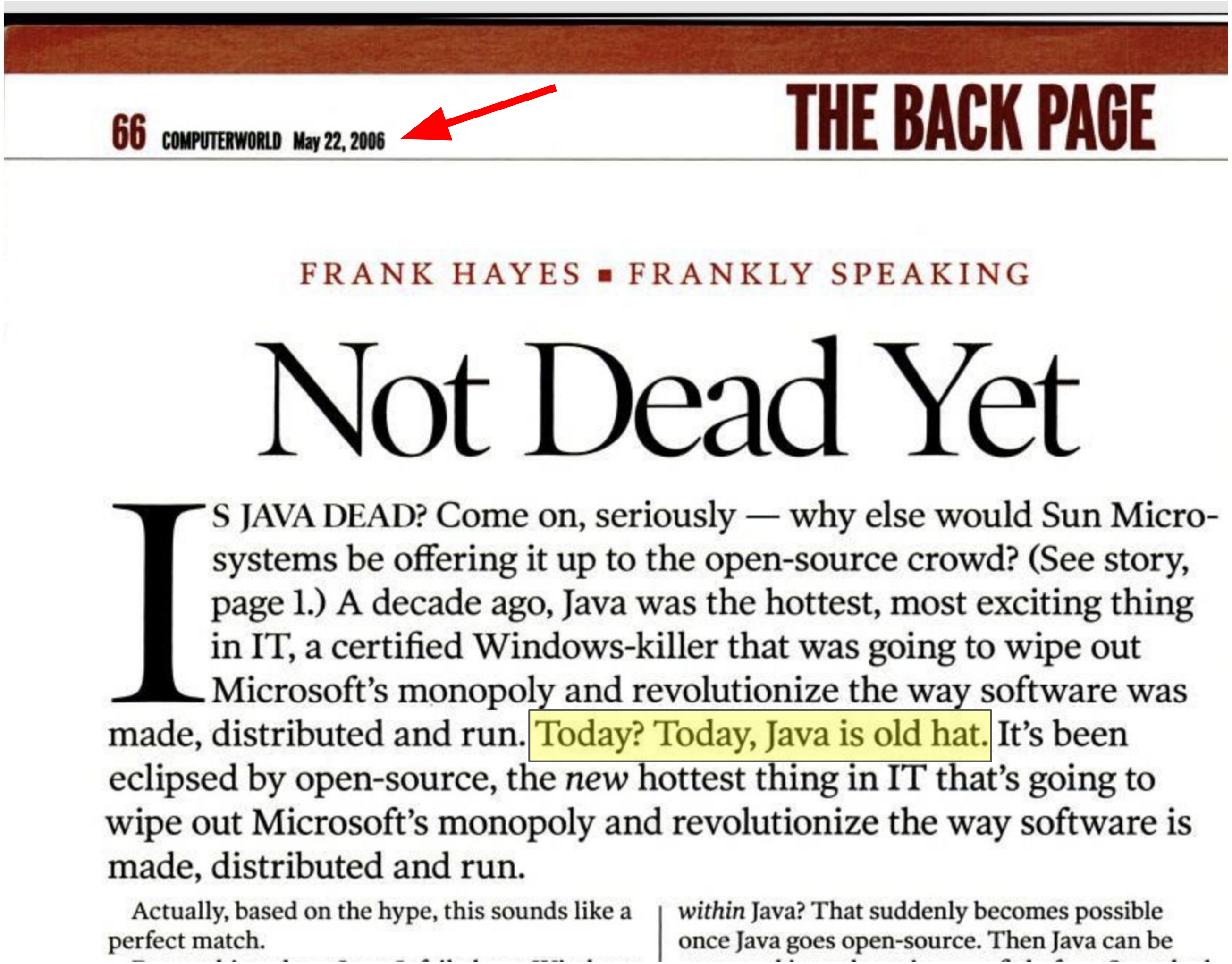
Red Hat

KUBERNETES-NATIVE JAVA

James Falkner / Red Hat / @schtool



“The reports of my death are greatly exaggerated.”



66 COMPUTERWORLD May 22, 2006

THE BACK PAGE

FRANK HAYES ▪ FRANKLY SPEAKING

Not Dead Yet

IS JAVA DEAD? Come on, seriously — why else would Sun Microsystems be offering it up to the open-source crowd? (See story, page 1.) A decade ago, Java was the hottest, most exciting thing in IT, a certified Windows-killer that was going to wipe out Microsoft's monopoly and revolutionize the way software was made, distributed and run. Today? Today, Java is old hat. It's been eclipsed by open-source, the *new* hottest thing in IT that's going to wipe out Microsoft's monopoly and revolutionize the way software is made, distributed and run.

Actually, based on the hype, this sounds like a perfect match.

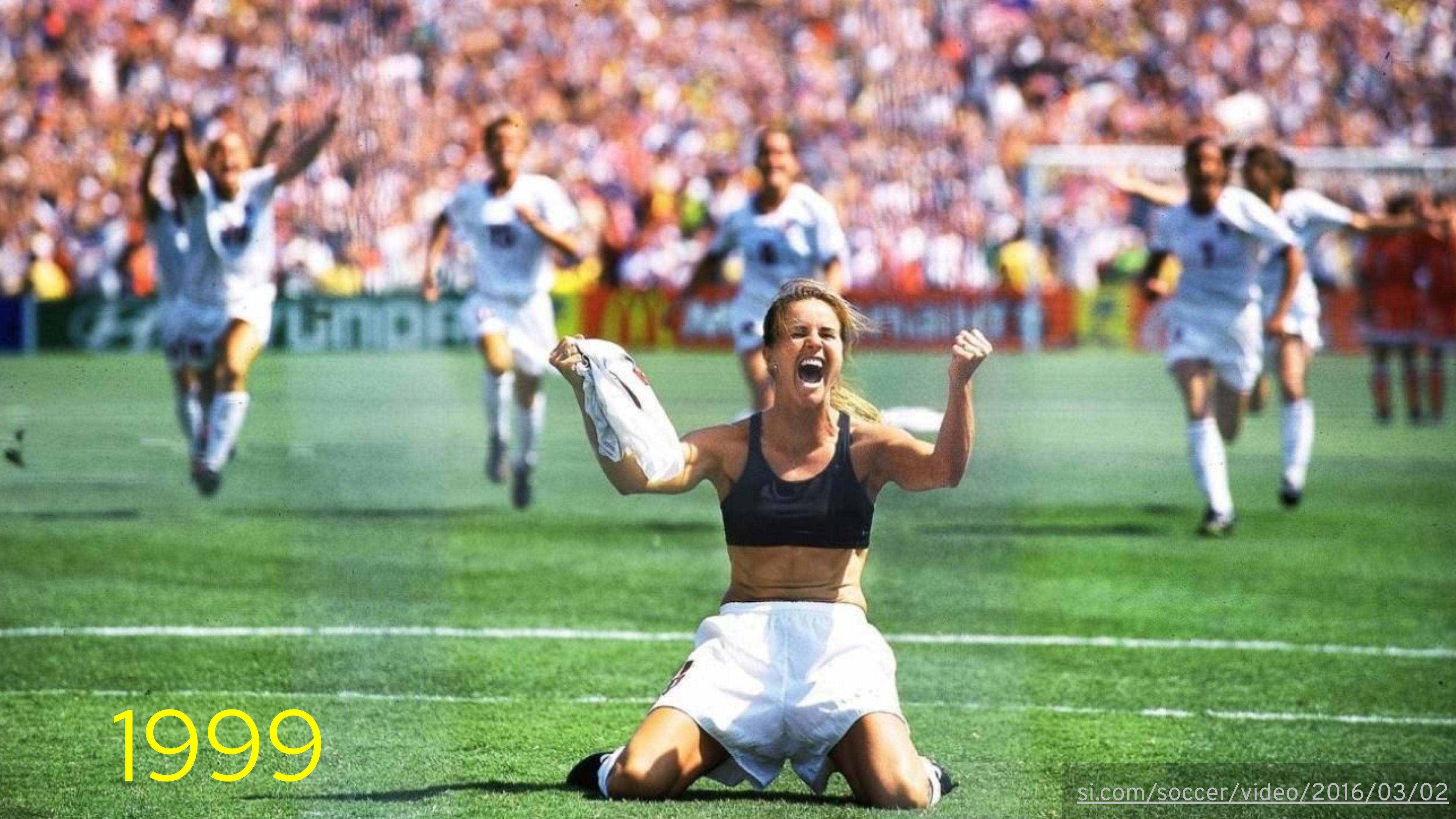
within Java? That suddenly becomes possible once Java goes open-source. Then Java can be

[source]

1999

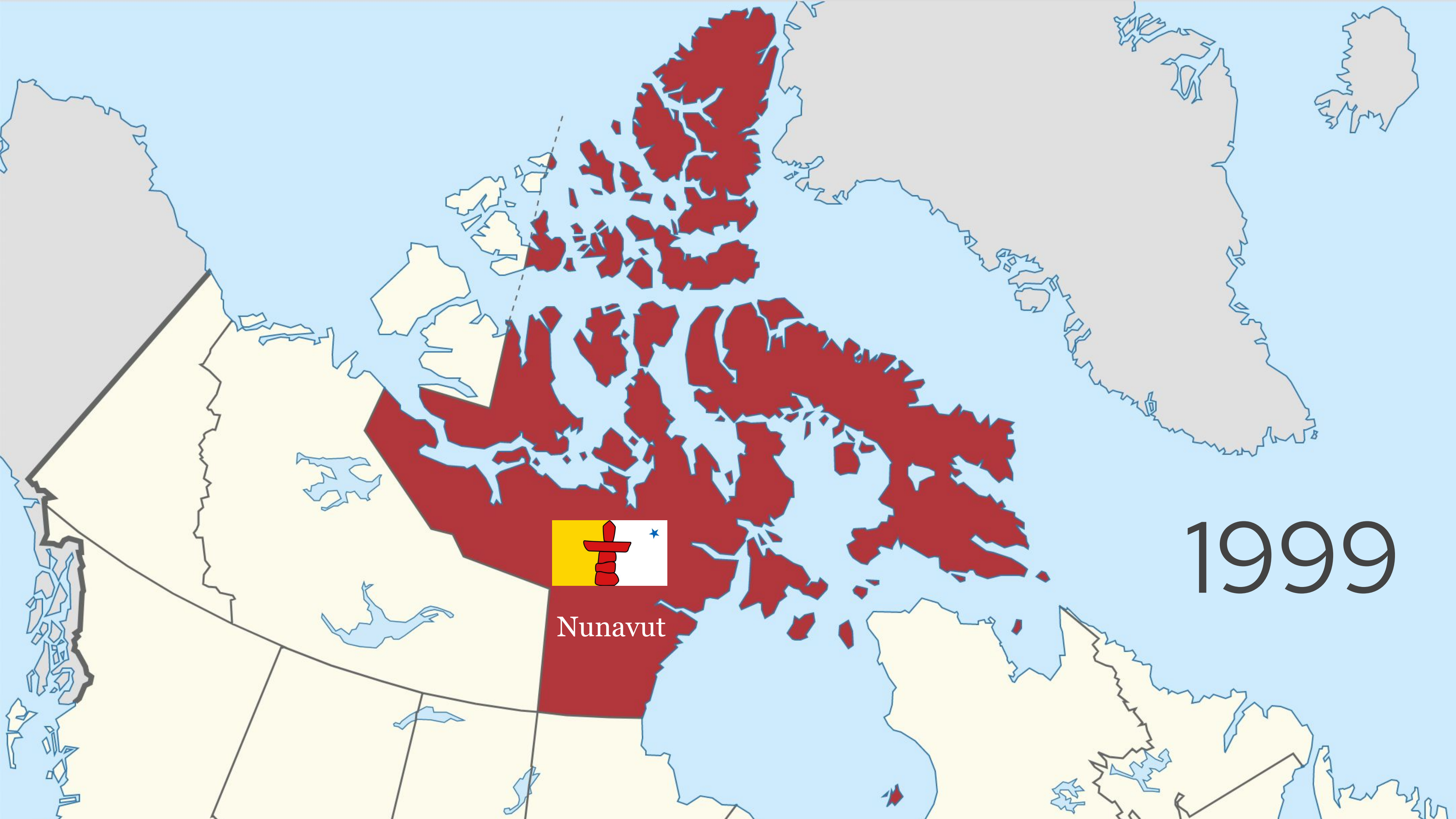
1999

USA Wins FIFA World Cup



1999

si.com/soccer/video/2016/03/02



1999

1999

Julie Payette
First Canadian
aboard the ISS

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

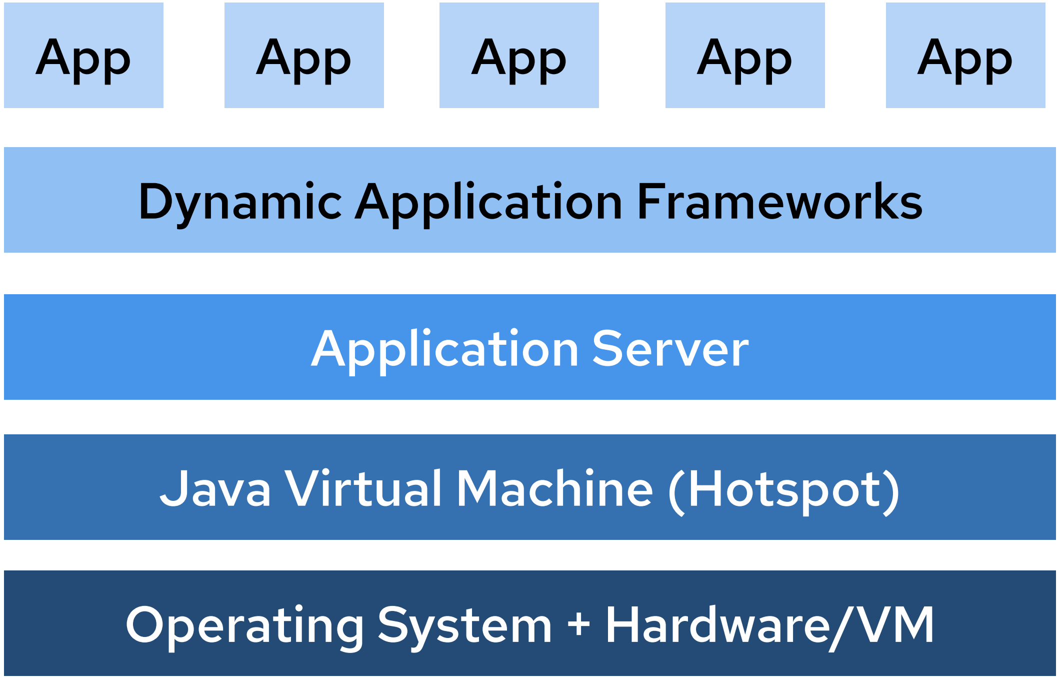
Architecture: Monoliths

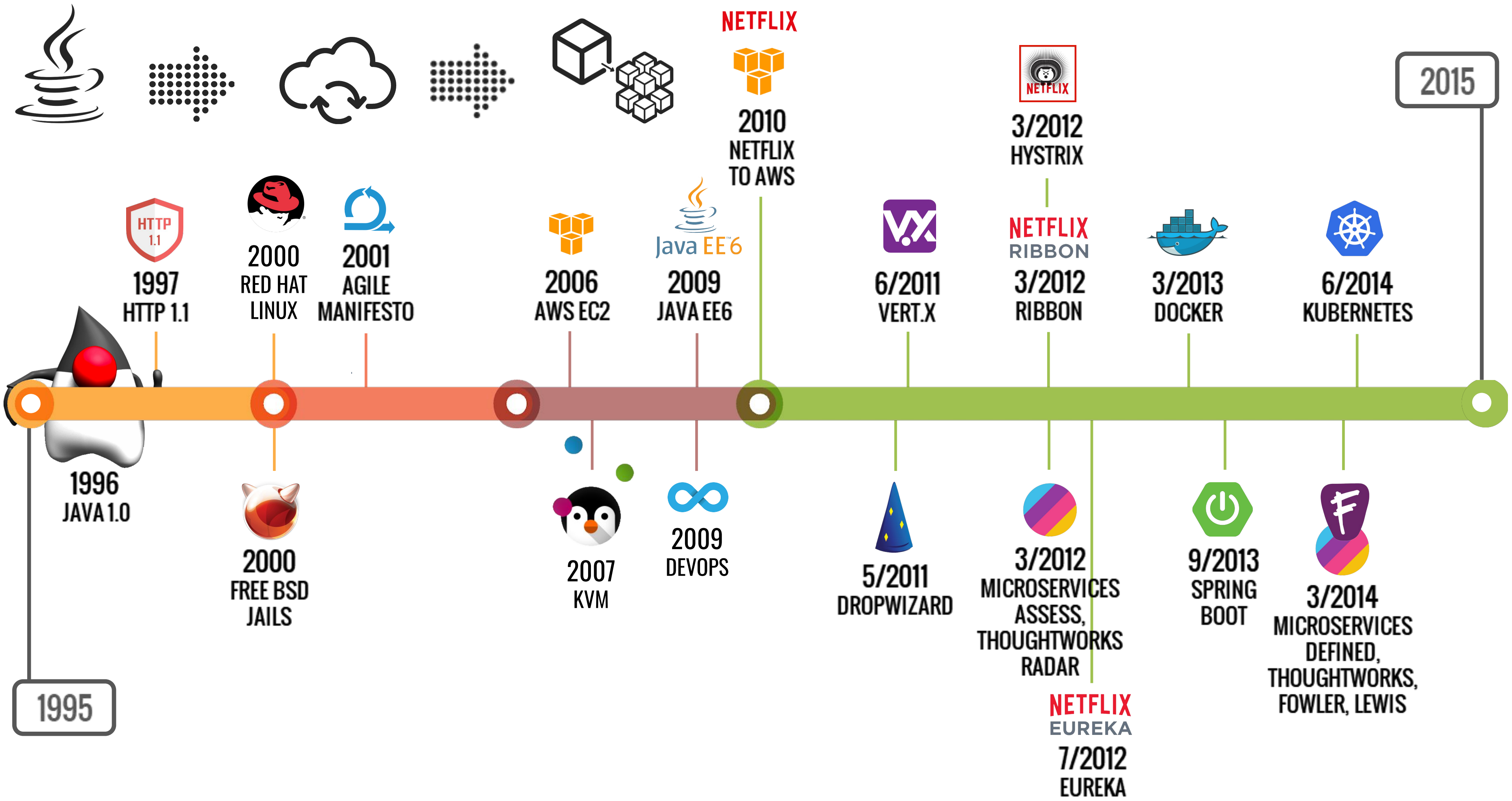
Deployment: multi-app, appserver

App Lifecycle: Months

Memory: 1GB+ RAM

Startup Time: 10s of sec


App
App
App
App
App
Dynamic Application Frameworks
Application Server
Java Virtual Machine (Hotspot)
Operating System + Hardware/VM



1995

1996 JAVA 1.0

1997 HTTP 1.1

2000 RED HAT LINUX

2000 FREE BSD JAILS

2001 AGILE MANIFESTO

2006 AWS EC2

2007 KVM

2009 JAVA EE6

2009 DEVOPS

2010 NETFLIX TO AWS

5/2011 DROPWIZARD

6/2011 VERT.X

3/2012 HYSTRIX

3/2012 RIBBON

3/2012 MICROSERVICES ASSESS, THOUGHTWORKS RADAR

7/2012 EUREKA

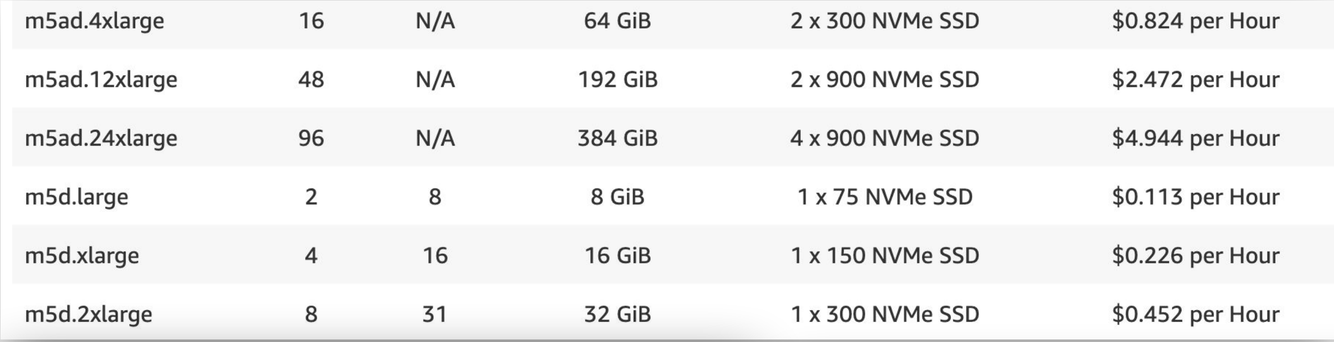
3/2013 DOCKER

9/2013 SPRING BOOT

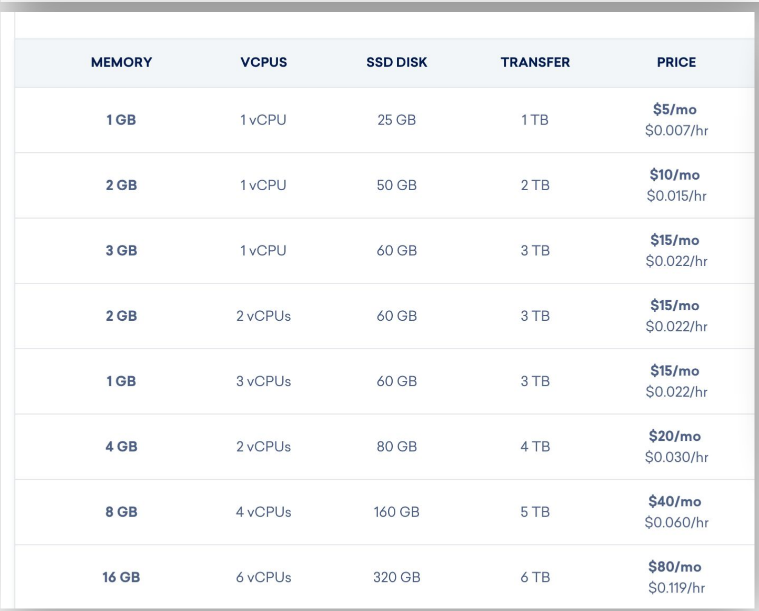
3/2014 MICROSERVICES DEFINED, THOUGHTWORKS, FOWLER, LEWIS

6/2014 KUBERNETES

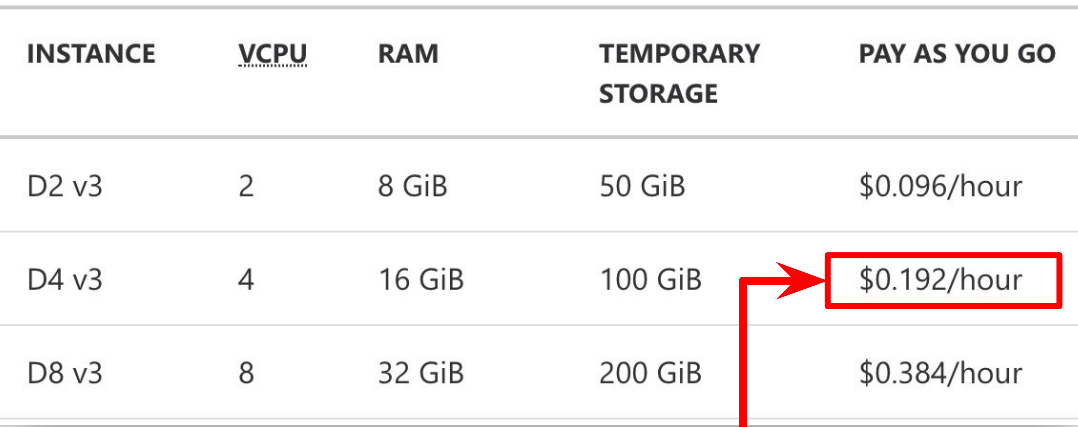
2015



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
1 GB	1 vCPU	25 GB	1 TB	$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
1 GB	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

Approx. $140/month

“Cloud Native” Java Stack

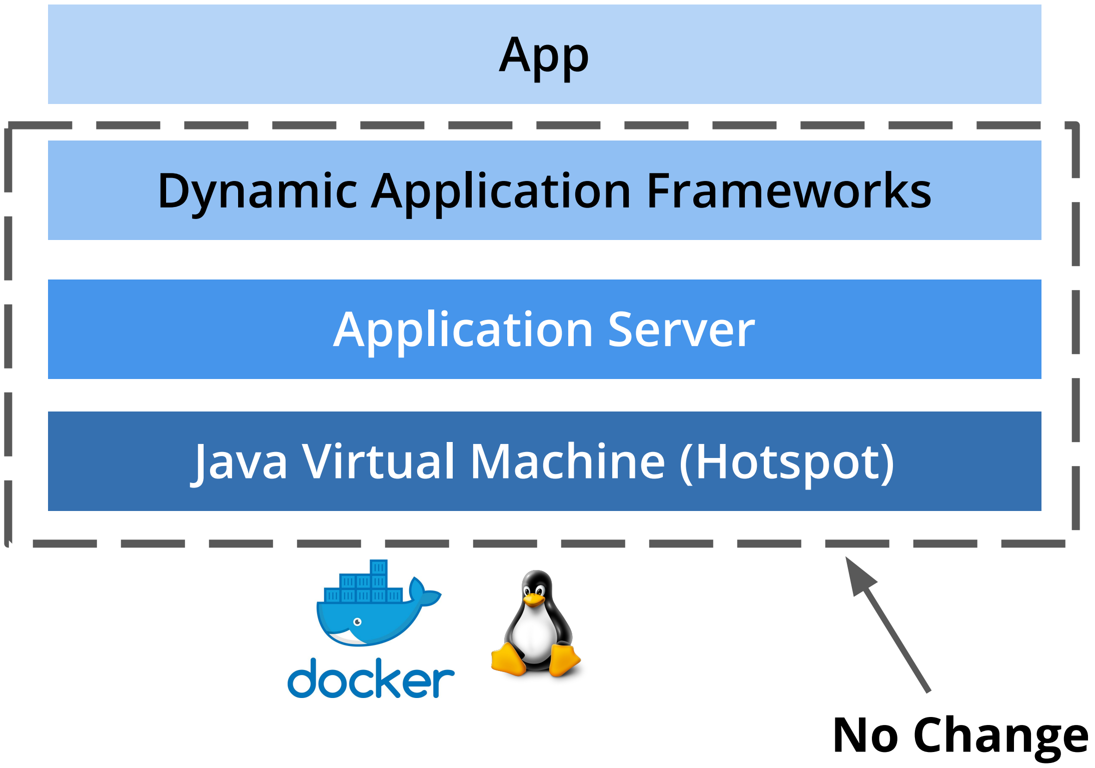
Architecture: Microservices

Deployment: Single App, Container

App Lifecycle: Days

Memory: 100MBs+ RAM

Startup Time: Seconds



App

Dynamic Application Frameworks

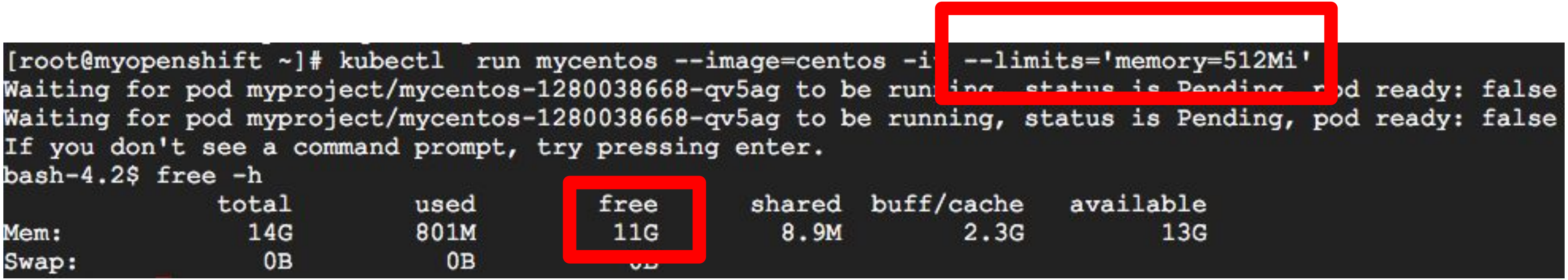
Application Server

Java Virtual Machine (Hotspot)

No Change

Hey, is it getting a little tight in here?

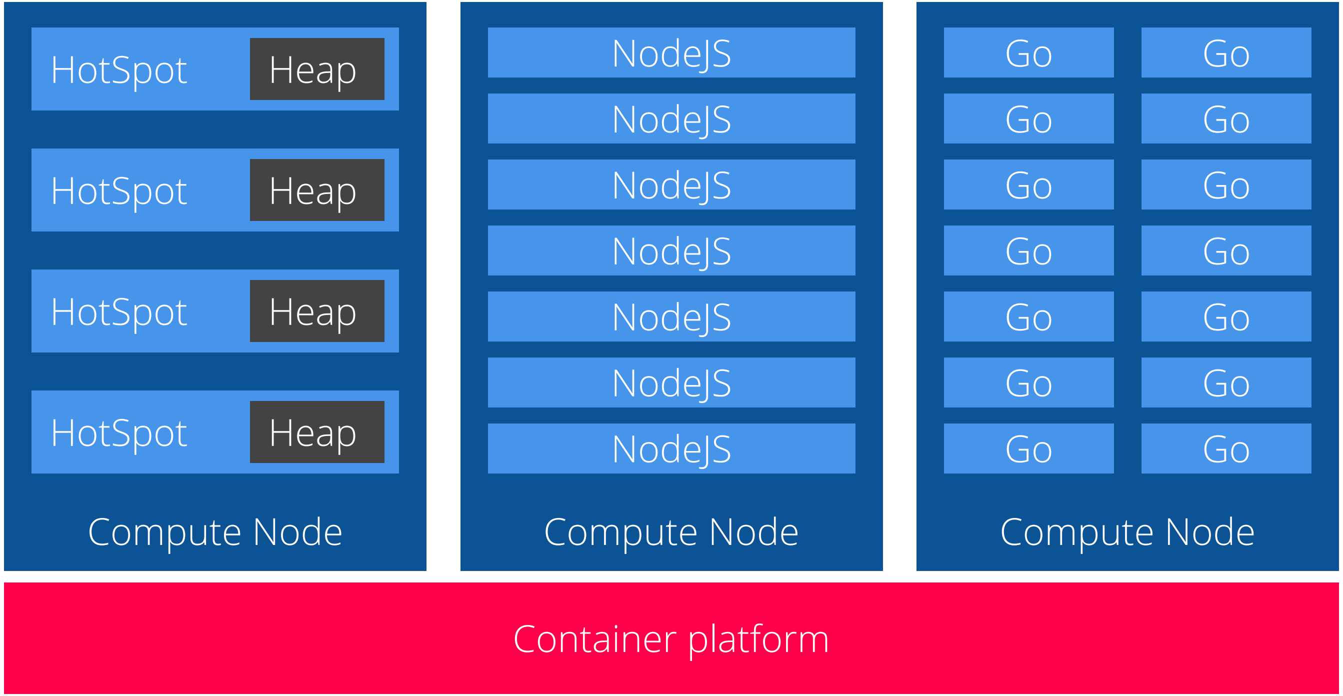
BOOM!



```
[root@myopenshift ~]# kubectl  run mycentos --image=centos -it --limits='memory=512Mi'
Waiting for pod myproject/mycentos-1280038668-qv5ag to be running, status is Pending, pod ready: false
Waiting for pod myproject/mycentos-1280038668-qv5ag to be running, status is Pending, pod ready: false
If you don't see a command prompt, try pressing enter.
bash-4.2$ free -h
              total        used        free      shared  buff/cache   available
Mem:            14G        801M         11G        8.9M        2.3G         13G
Swap:            0B          0B          0B
```

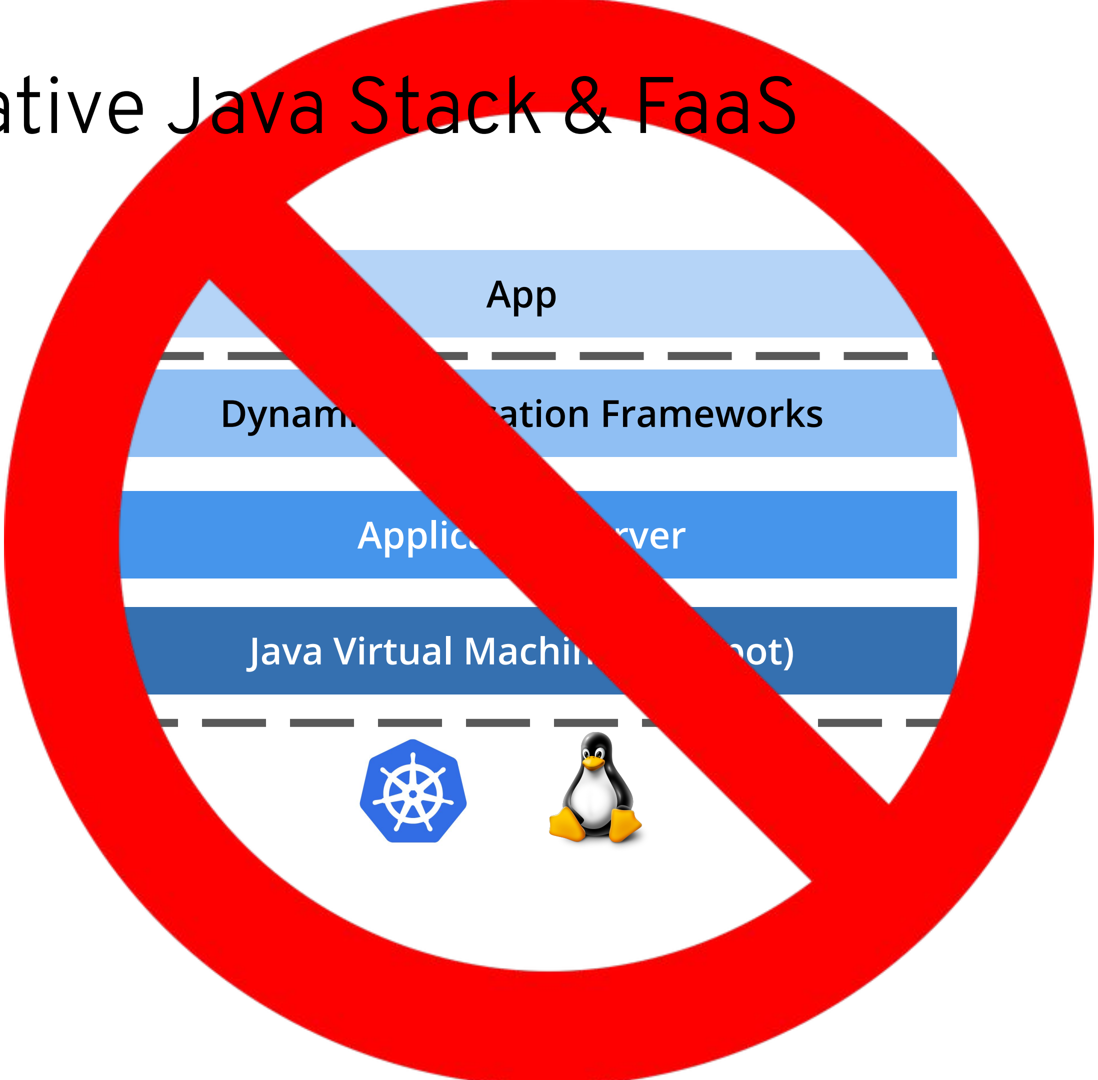
?

The “hidden” truth about Java + containers



Cloud Native Java Stack & FaaS

Architecture:	FaaS
Deployment:	Functions
Lifecycle:	Seconds
Memory:	MBs of Ram
Startup Time:	Milliseconds



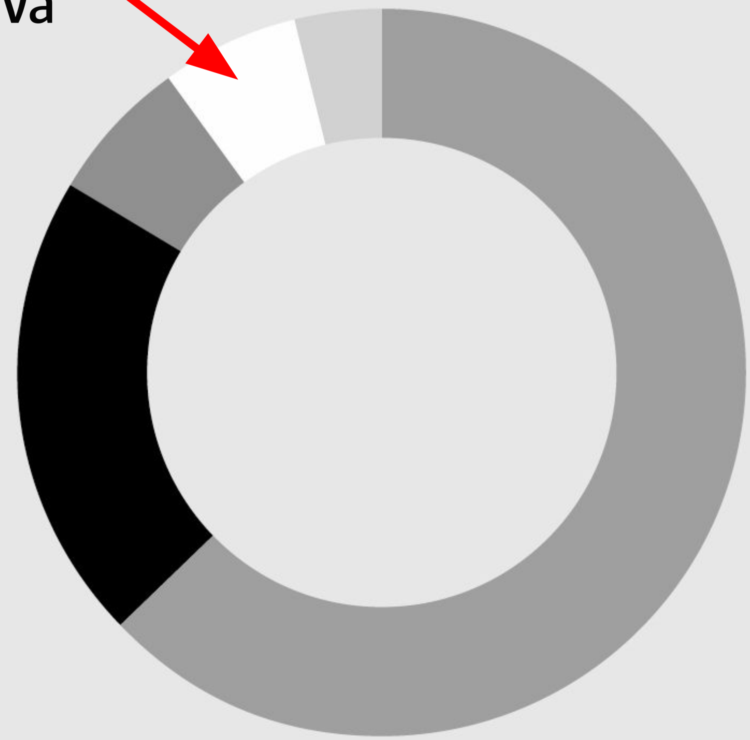
App

Dynam ation Frameworks

Applic rver

Java Virtual Machi ot)

Languages used on AWS Lambda



●	**62.9%**	Node.js
●	**20.8%**	Python
●	**6.4%**	Go
○	**6.1%**	Java
●	**3.8%**	C#

https://serverless.com/blog/2018-serverless-community-survey-huge-growth-usage/

QUARKUS

Supersonic. Subatomic. Java.

Supersonic, Subatomic Java

Quarkus powers the next-generation Java stack for hybrid-cloud applications

Cloud Efficiency

(low memory, fast startup: supersonic, subatomic; efficient, cost effective)

Developer Joy

(live coding, IDE extensions, familiar APIs, reuse Java skills)

Hybrid Cloud

(Kubernetes-native, hybrid-cloud application development)

Quarkus powers Red Hat and third-party commercial apps

Quarkus - Optimizing the Stack

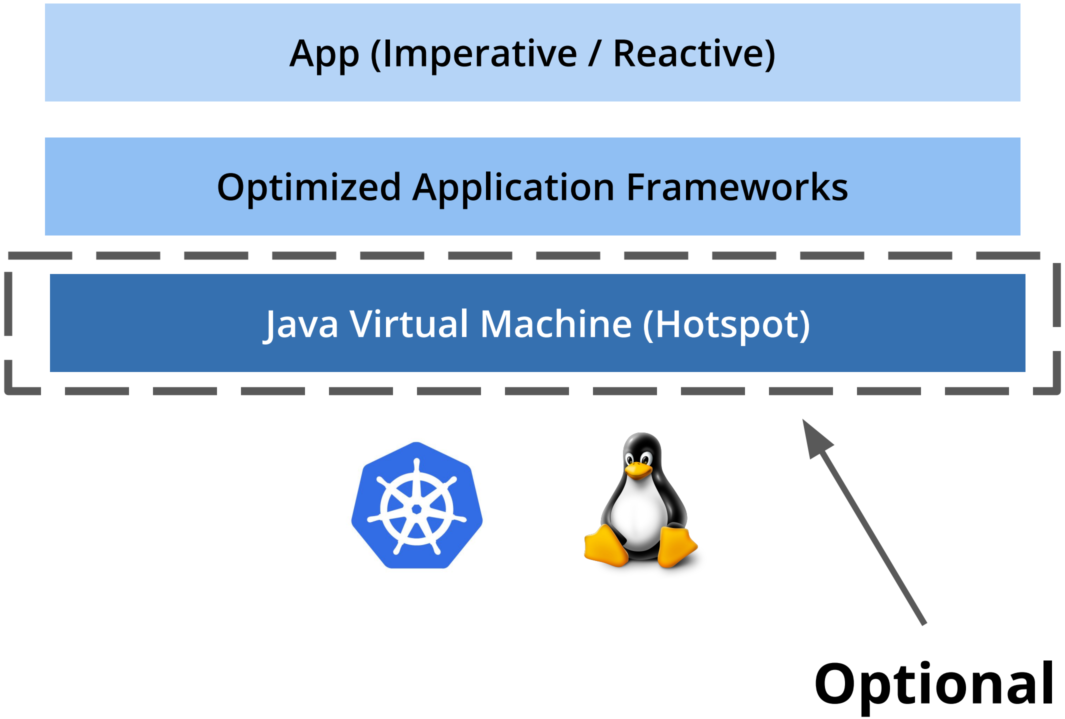
Architecture: Microservices, Serverless

Deployment: Single App

App Lifecycle: Milliseconds to Days

Memory: 10MBs+ RAM

Startup Time: Milliseconds



App (Imperative / Reactive)

Optimized Application Frameworks

Java Virtual Machine (Hotspot)

Optional

Moving to Compile-Time Boot

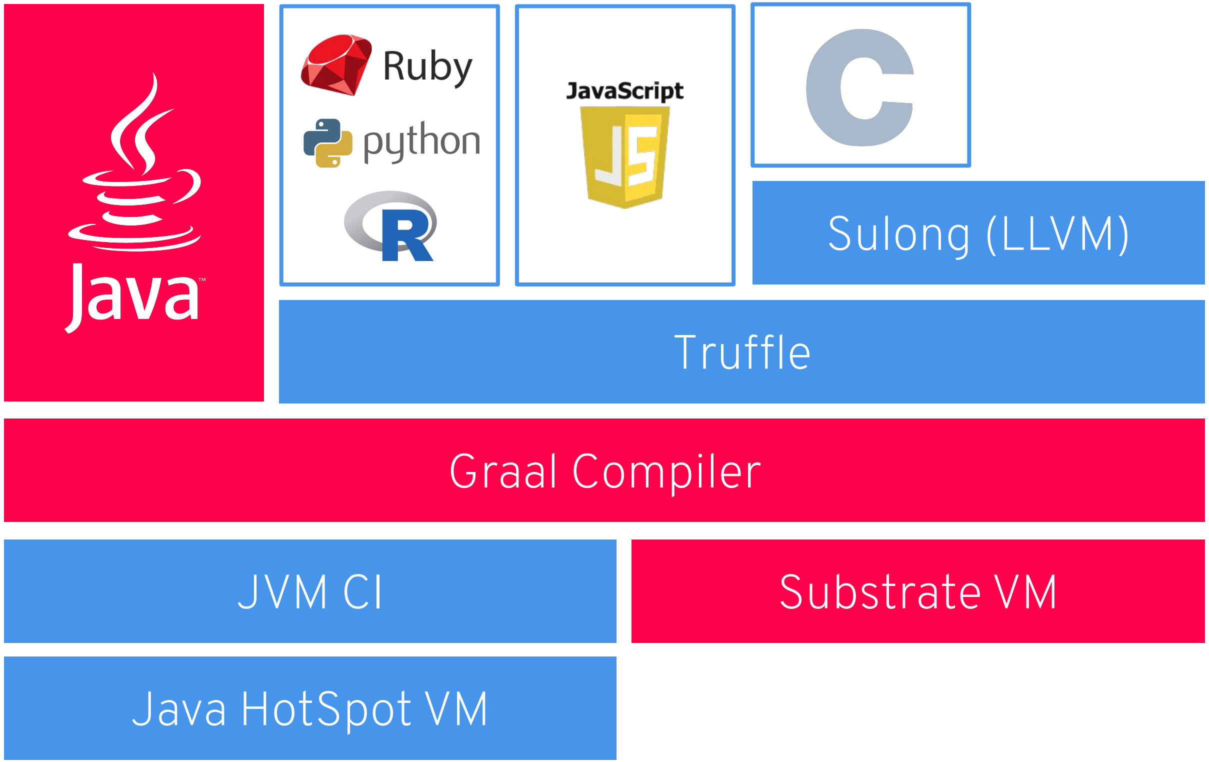
What does a framework do at startup time?

- Parse config files
- Classpath & classes scanning
  - for annotations, getters or other metadata
- Build framework metamodel objects
- Prepare reflection and build proxies
- *Start and open IO, threads etc*

Quarkus Optimizations

- Move as much as possible to build phase
- Minimize runtime dependencies
- Maximize dead code elimination
- Introduce clear metadata contracts
- Spectrum of optimization levels (all → some → no runtime reflection)

GraalVM


Java
Ruby
python
R
JavaScript
JS
C
Sulong (LLVM)
Truffle
Graal Compiler
JVM CI
Substrate VM
Java HotSpot VM

Best of Breed Frameworks & Standards

Eclipse Vert.x

Hibernate

RESTEasy

Apache Camel

Eclipse MicroProfile

Netty

Kubernetes

OpenShift

Jaeger

Prometheus

Apache Kafka

Infinispan

Flyway

Neo4j

MongoDB

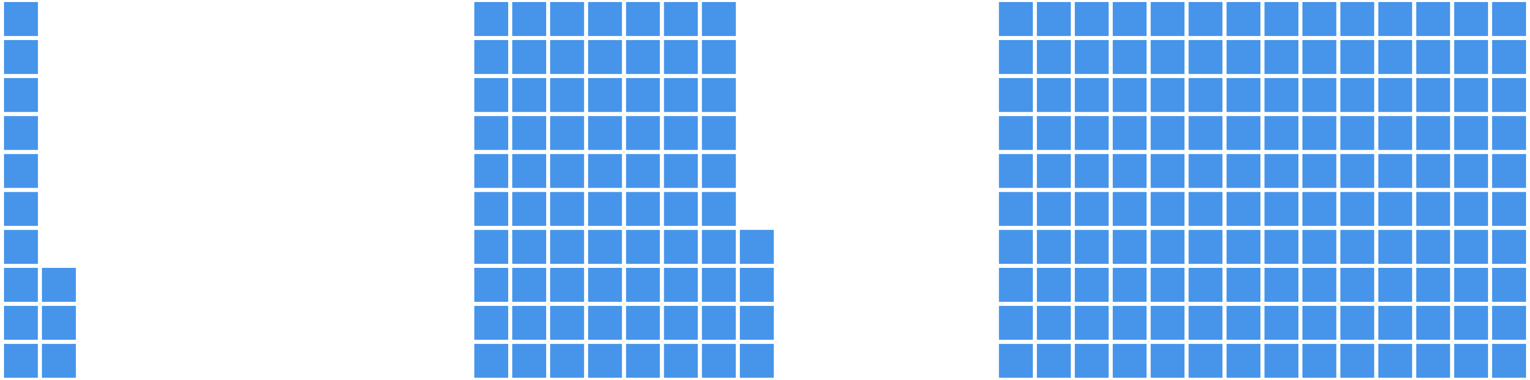
MQTT

KeyCloak

Apache Tika

Memory Utilization

REST



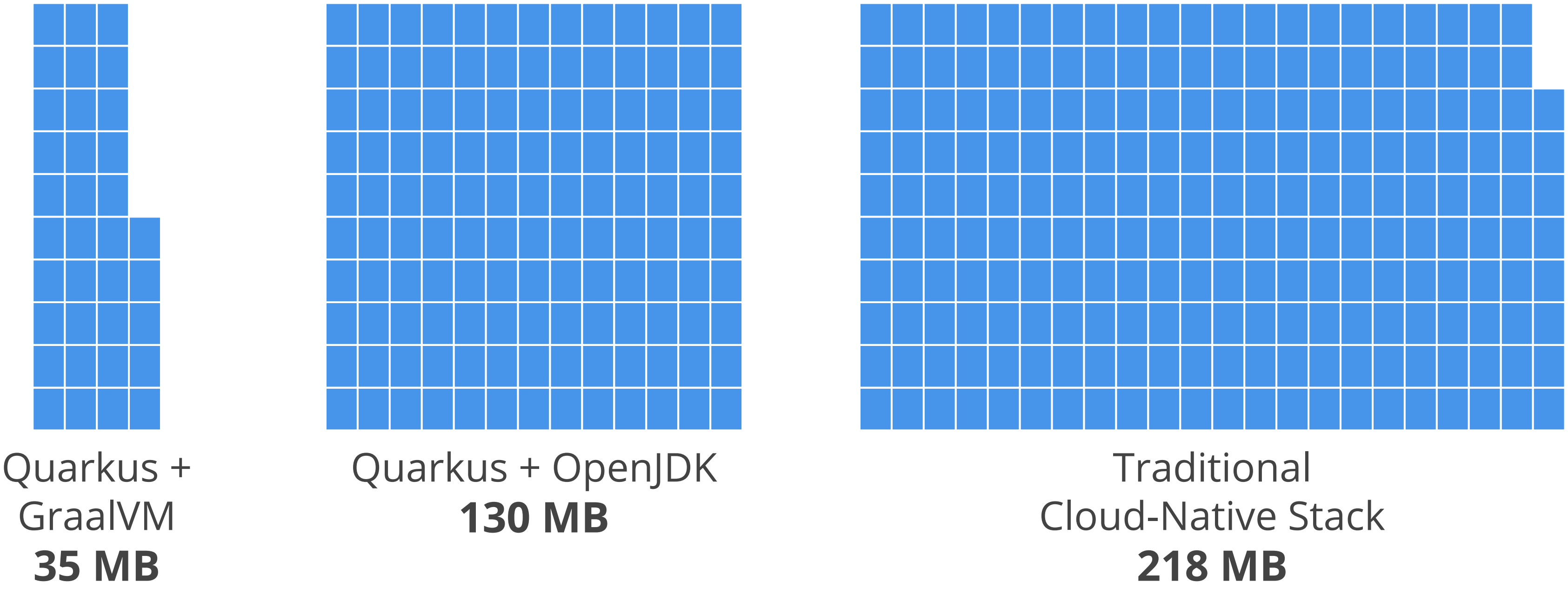
Quarkus + GraalVM
13 MB

Quarkus + OpenJDK
74 MB

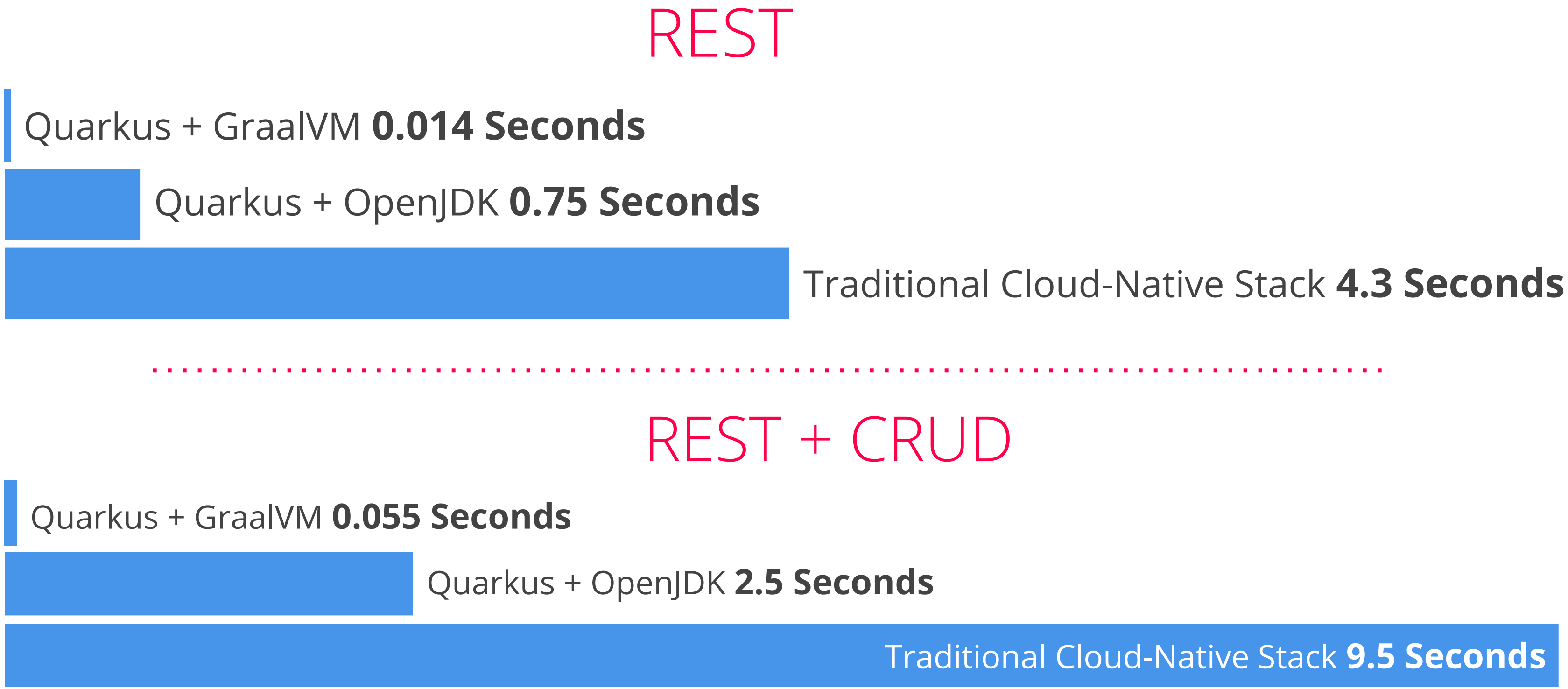
Traditional Cloud-Native Stack
140 MB

Memory Utilization

REST + CRUD



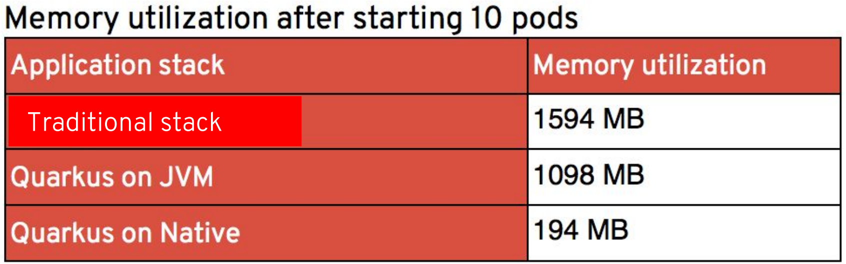
Quarkus Improves Startup Time



REST

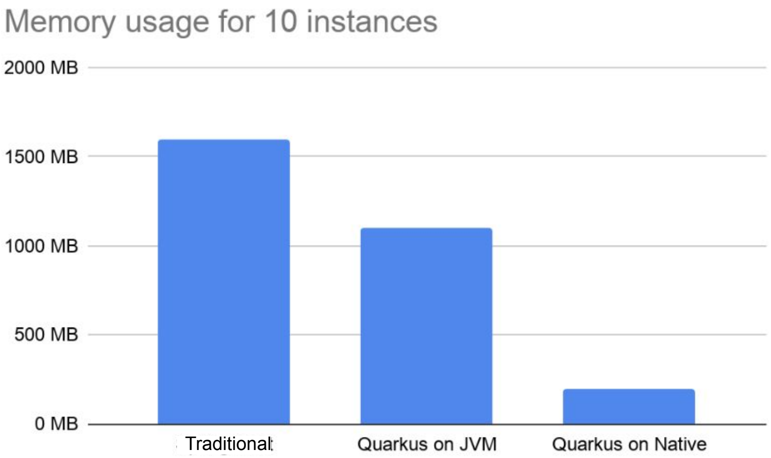
REST + CRUD

Deployment density - OCP cluster on AWS



Memory utilization after starting 10 pods

Application stack	Memory utilization
Traditional stack	1594 MB
Quarkus on JVM	1098 MB
Quarkus on Native	194 MB

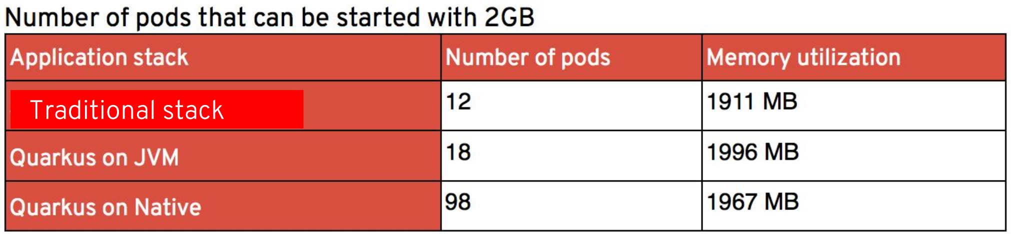


Quarkus JVM ~ 30% less than Traditional

Quarkus native ~ 1/8 RAM of Traditional

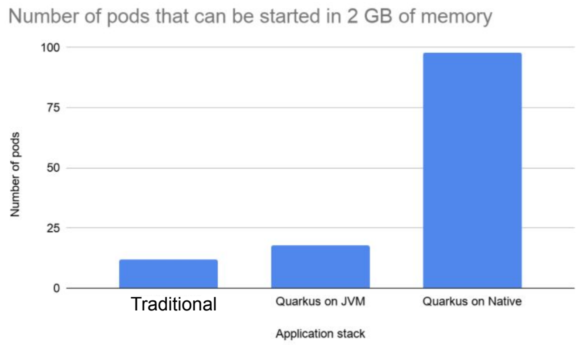
Source: Study to be published April 2020

Deployment density - OCP cluster on AWS



Number of pods that can be started with 2GB

Application stack	Number of pods	Memory utilization
Traditional stack	12	1911 MB
Quarkus on JVM	18	1996 MB
Quarkus on Native	98	1967 MB



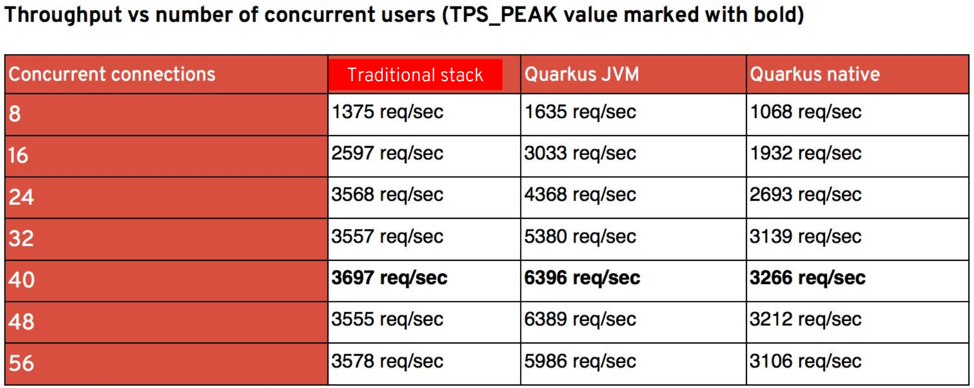
Quarkus JVM ~ 50% more than Traditional

Quarkus native ~ 800% more than Traditional

higher is better

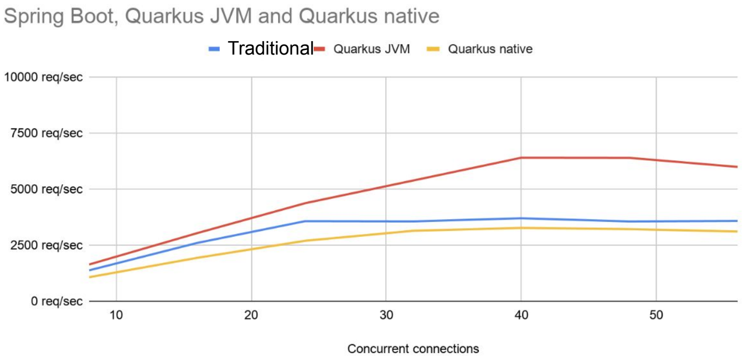
Source: Study to be published April 2020

TPS under load - containers on bare metal



Throughput vs number of concurrent users (TPS_PEAK value marked with bold)

Concurrent connections	Traditional stack	Quarkus JVM	Quarkus native
8	1375 req/sec	1635 req/sec	1068 req/sec
16	2597 req/sec	3033 req/sec	1932 req/sec
24	3568 req/sec	4368 req/sec	2693 req/sec
32	3557 req/sec	5380 req/sec	3139 req/sec
40	**3697 req/sec**	**6396 req/sec**	**3266 req/sec**
48	3555 req/sec	6389 req/sec	3212 req/sec
56	3578 req/sec	5986 req/sec	3106 req/sec

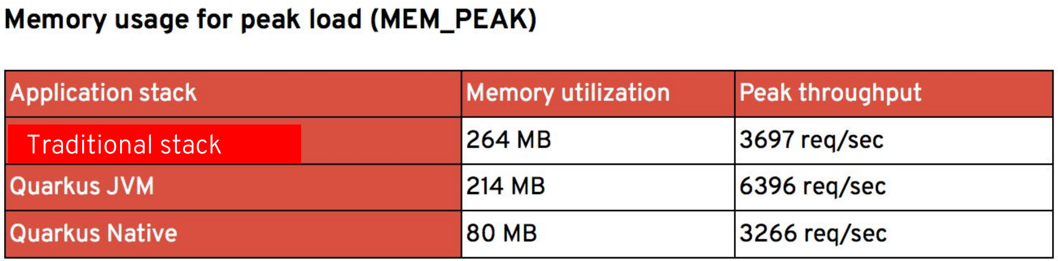


Quarkus JVM ~ 70% higher than Traditional

Quarkus native ~ 10% lower than Traditional, **BUT at what cost?**

Source: Study to be published April 2020

Mem under load - containers on bare metal

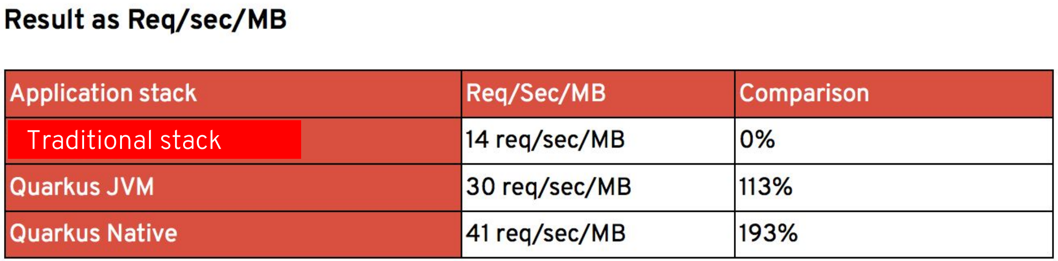


Memory usage for peak load (MEM_PEAK)

Application stack	Memory utilization	Peak throughput
Traditional stack	264 MB	3697 req/sec
Quarkus JVM	214 MB	6396 req/sec
Quarkus Native	80 MB	3266 req/sec

Quarkus JVM ~ 20% less than Traditional

Quarkus native ~ 1/3 RAM of Traditional



Result as Req/sec/MB

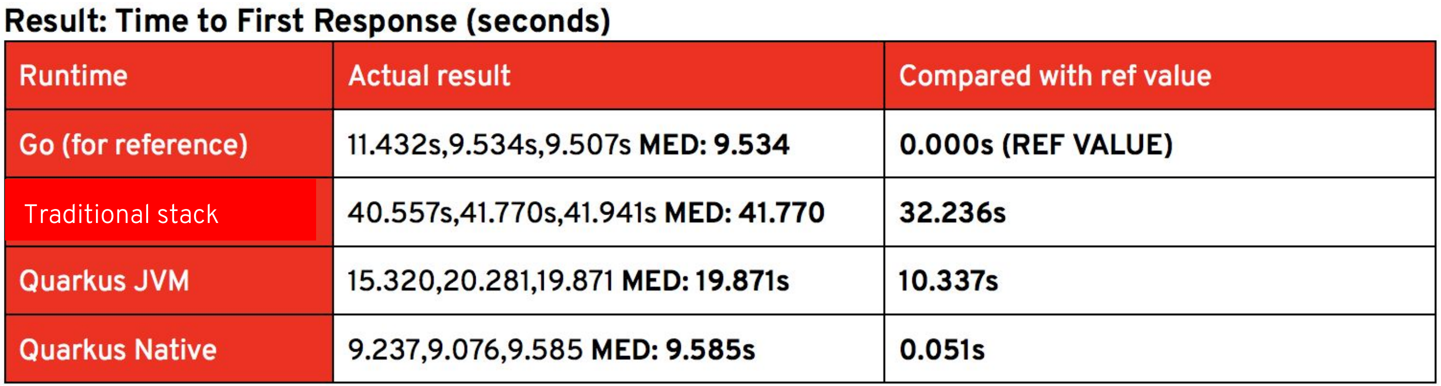
Application stack	Req/Sec/MB	Comparison
Traditional stack	14 req/sec/MB	0%
Quarkus JVM	30 req/sec/MB	113%
Quarkus Native	41 req/sec/MB	193%

} **higher is better**

You get higher TPS for each consumed MB of RAM

Source: Study to be published April 2020

Serverless - OCP Knative on AWS



Result: Time to First Response (seconds)

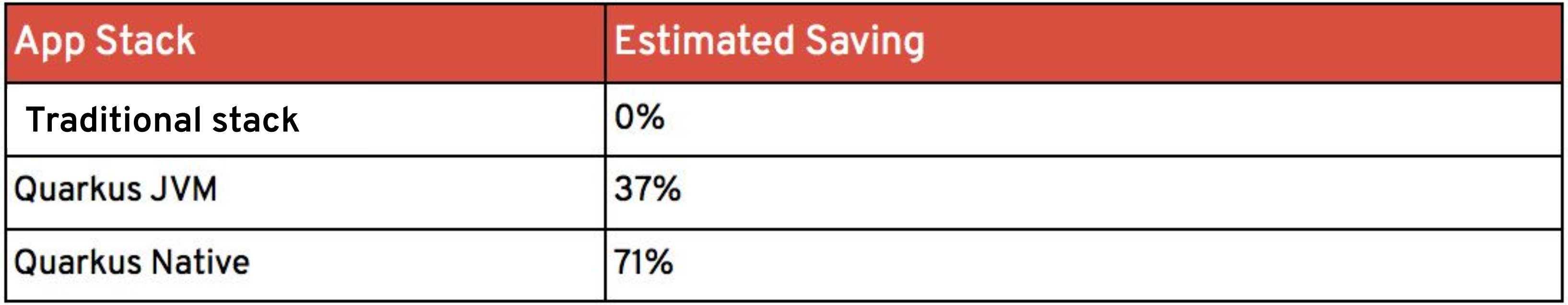
Runtime	Actual result	Compared with ref value
Go (for reference)	11.432s,9.534s,9.507s **MED: 9.534**	**0.000s (REF VALUE)**
Traditional stack	40.557s,41.770s,41.941s **MED: 41.770**	**32.236s**
Quarkus JVM	15.320,20.281,19.871 **MED: 19.871s**	**10.337s**
Quarkus Native	9.237,9.076,9.585 **MED: 9.585s**	**0.051s**

Quarkus JVM ~ 2x faster than Traditional

Quarkus native ~ 4.5x faster than Traditional **(comparable to Golang)**

Source: Study to be published April 2020

Cost savings - containers on OCP on AWS



App Stack	Estimated Saving
Traditional stack	0%
Quarkus JVM	37%
Quarkus Native	71%

Quarkus JVM would need 37% less memory than Traditional

Quarkus native would need 71% less memory than Traditional

Assumption: A customer has about 300 services deployed and 40% (120 pods) are in the category of **high**, 40% (120 pods) are in the category of **medium**, and 20% (60 pods) are in the category of **low**.

Source: Study to be published April 2020

DEMO

Quarkus in production

Read more at quarkus.io

COMING SOON

Flo Urban Solutions

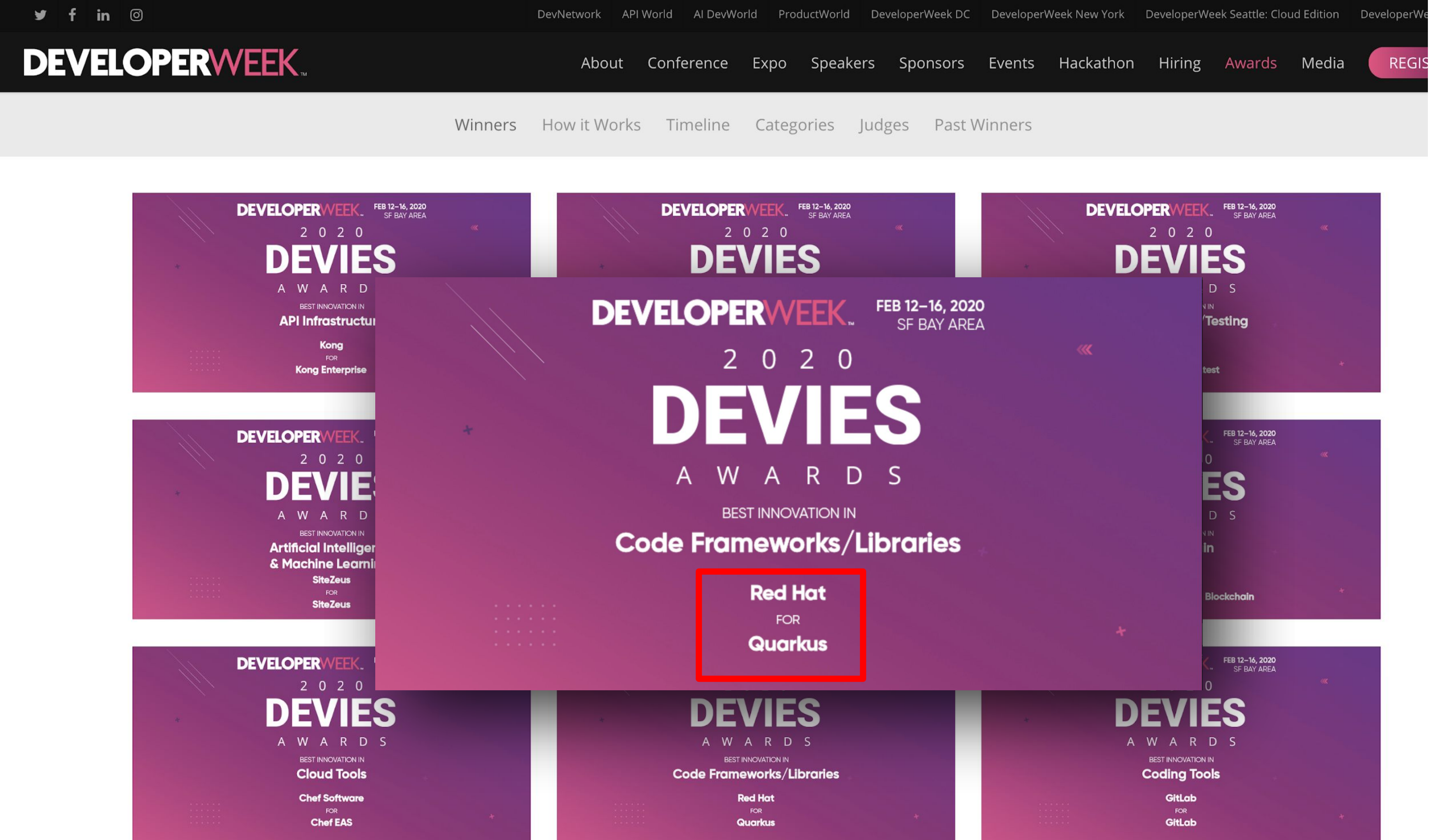
talkdesk®

COMING SOON

vodafone

COMING SOON

COMING SOON



DEVELOPERWEEK FEB 12–16, 2020 SF BAY AREA

2020 DEVIES AWARDS

BEST INNOVATION IN

Code Frameworks/Libraries

Red Hat

FOR

Quarkus

DEVELOPERWEEK FEB 12–16, 2020 SF BAY AREA 2020 DEVIES AWARD BEST INNOVATION IN API Infrastructur Kong FOR Kong Enterprise

DEVELOPERWEEK FEB 12–16, 2020 SF BAY AREA 2020 DEVIES

DEVELOPERWEEK FEB 12–16, 2020 SF BAY AREA 2020 DEVIES D S IN /Testing test

DEVELOPERWEEK 2020 DEVIE AWARD BEST INNOVATION IN Artificial Intelliger & Machine Learni SiteZeus FOR SiteZeus

FEB 12–16, 2020 SF BAY AREA 0 ES D S IN in Blockchain

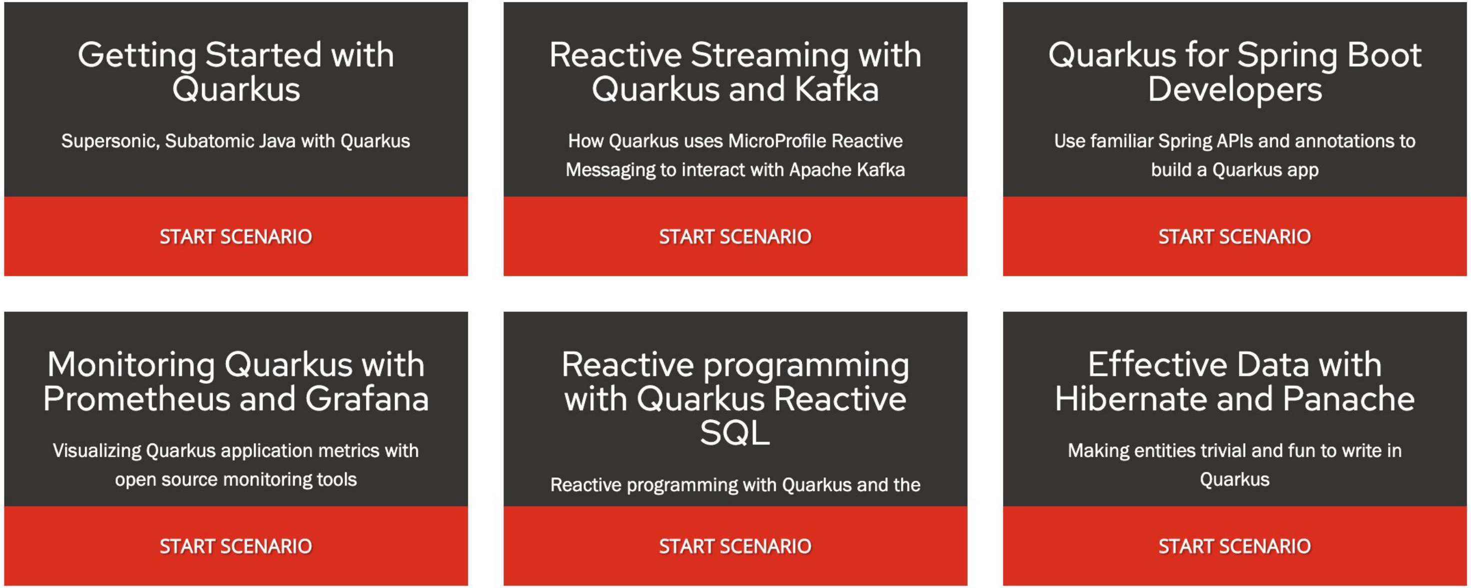
DEVELOPERWEEK 2020 DEVIES AWARDS BEST INNOVATION IN Cloud Tools Chef Software FOR Chef EAS

DEVIES AWARDS BEST INNOVATION IN Code Frameworks/Libraries Red Hat FOR Quarkus

FEB 12–16, 2020 SF BAY AREA 0 DEVIES AWARDS BEST INNOVATION IN Coding Tools GitLab FOR GitLab

Try it yourself

bit.ly/try-quarkus



Getting Started with Quarkus

Supersonic, Subatomic Java with Quarkus

START SCENARIO

Reactive Streaming with Quarkus and Kafka

How Quarkus uses MicroProfile Reactive Messaging to interact with Apache Kafka

START SCENARIO

Quarkus for Spring Boot Developers

Use familiar Spring APIs and annotations to build a Quarkus app

START SCENARIO

Monitoring Quarkus with Prometheus and Grafana

Visualizing Quarkus application metrics with open source monitoring tools

START SCENARIO

Reactive programming with Quarkus Reactive SQL

Reactive programming with Quarkus and the

START SCENARIO

Effective Data with Hibernate and Panache

Making entities trivial and fun to write in Quarkus

START SCENARIO

Supersonic, Subatomic Java

Quarkus powers the next-generation Java stack for hybrid-cloud applications

Cloud Efficiency

(low memory, fast startup: supersonic, subatomic; efficient, cost effective)

Developer Joy

(live coding, IDE extensions, familiar APIs, reuse Java skills)

Hybrid Cloud

(Kubernetes-native, hybrid-cloud application development)

Red Hat Day Events

January 30, Vancouver

OpenShift
The Platform for Big Ideas

Red Hat