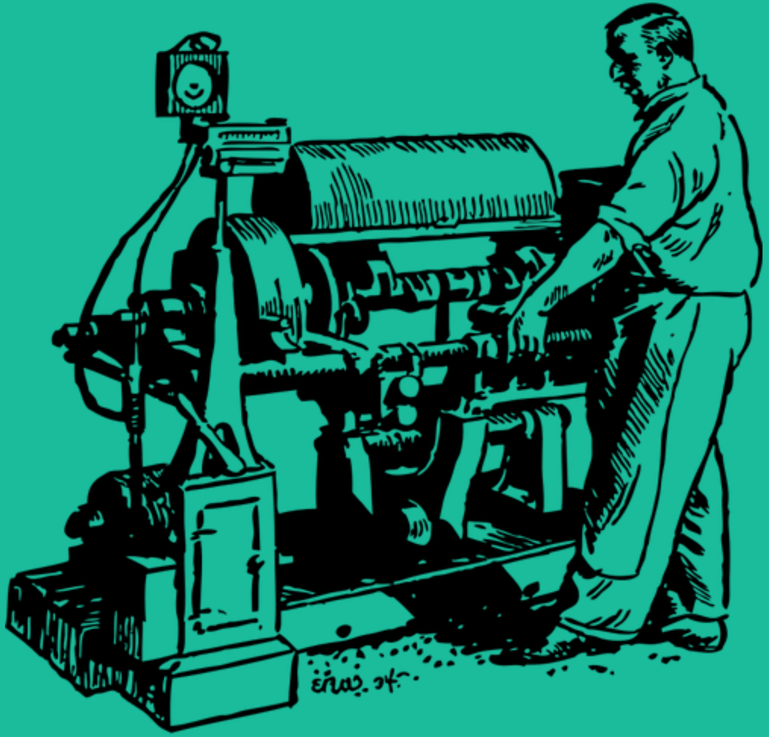


Chapter 10. Cloud-native Application Development

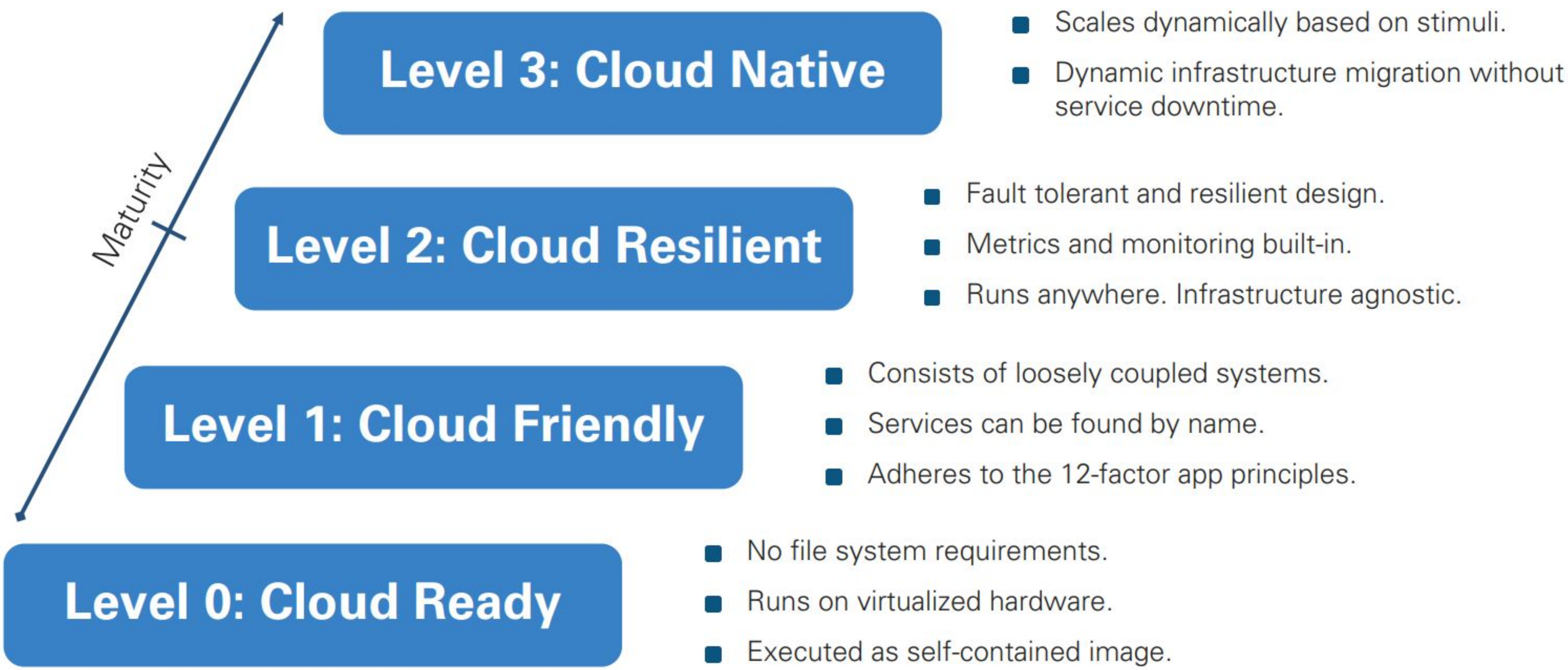
Bilkent University | CS443 | 2021, Spring | Dr. Orçun Dayıbaş

Cloud-native Applications

- **What is “Cloud-nativeness”?**
  - CNCF Definition (v1.0)
    - Cloud native technologies empower organizations to build and **run scalable applications in modern, dynamic environments** such as public, private, and hybrid clouds.
    - Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.
    - These techniques enable loosely coupled systems that are **resilient, manageable, and observable**. Combined with robust **automation**, they allow engineers to make high-impact changes frequently and predictably with minimal toil.
  - Older version: “container packaged, dynamically scheduled, microservices-based application development & operations”
  - Landscape: https://landscape.cncf.io/

Cloud-native Applications

- **Maturity of Cloud-native Applications**



Level 3: Cloud Native

- Scales dynamically based on stimuli.
- Dynamic infrastructure migration without service downtime.

Level 2: Cloud Resilient

- Fault tolerant and resilient design.
- Metrics and monitoring built-in.
- Runs anywhere. Infrastructure agnostic.

Level 1: Cloud Friendly

- Consists of loosely coupled systems.
- Services can be found by name.
- Adheres to the 12-factor app principles.

Level 0: Cloud Ready

- No file system requirements.
- Runs on virtualized hardware.
- Executed as self-contained image.

https://www.slideshare.net/MarioLeanderReimer/a-hitchhikers-guide-to-the-cloud-native-stack-cds17

Cloud-native Applications

- **Self assessment**

1. Can you redeploy your entire application in minutes?
2. Is your application independent from specific IP addresses, ports, file systems that are not part of the automated installation?
3. Can your application survive, and auto-recover from, infrastructure (compute, network, storage) failures?
4. Can you upgrade and downgrade, your application (or parts of the application) without any impact to users?
5. Can you run multiple versions of your application services, in the same environment at the same time?
6. Can you safely test in production?
7. If a part of an application fails, will other parts continue to operate?
8. Can parts of your application scale-up and scale-down automatically, based on user load or other factors (stimuli)?
9. Can you deploy application components across cloud providers?
10. Can you deploy an application component on a different cloud provider?

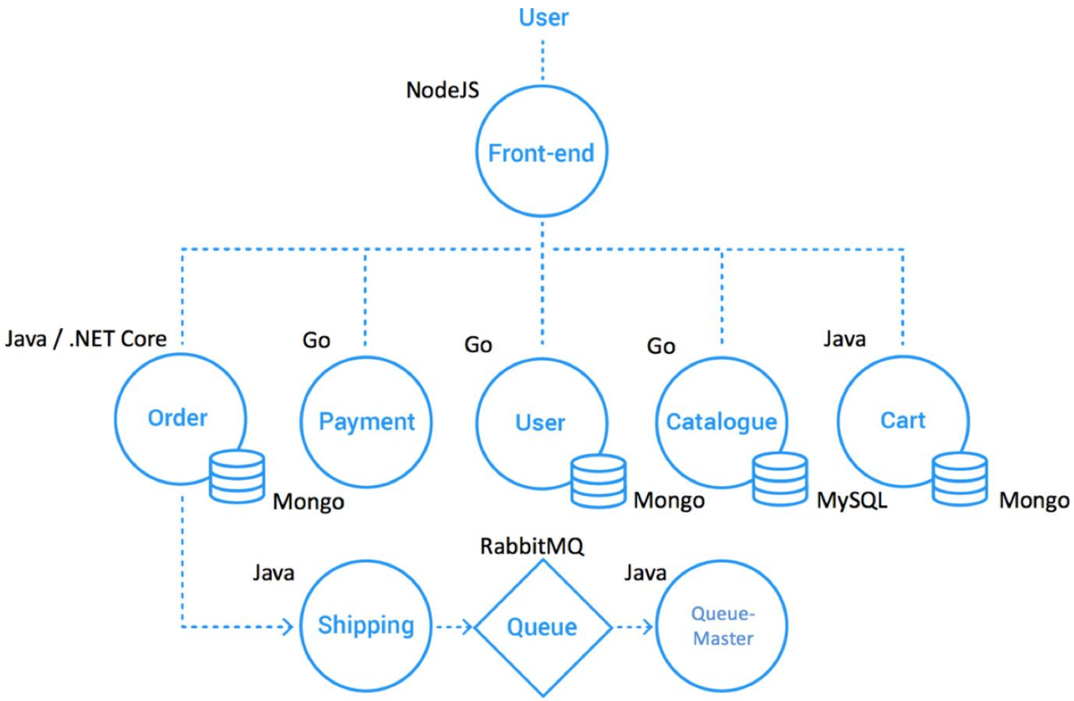
source: https://dzone.com/articles/cloud-native-application

Cloud-native Applications

- **Design principles**
  - Distribution
    - Containers, microservices, API-driven dev.
  - Performance
    - Resource efficient, concurrent, responsive
  - Automation
    - Automated DevOps tasks
  - Resiliency
    - Fault-tolerant, self-healing
  - Elasticity
    - Scales dynamically and react to stimuli
  - Observability
    - Logs, metrics and traces

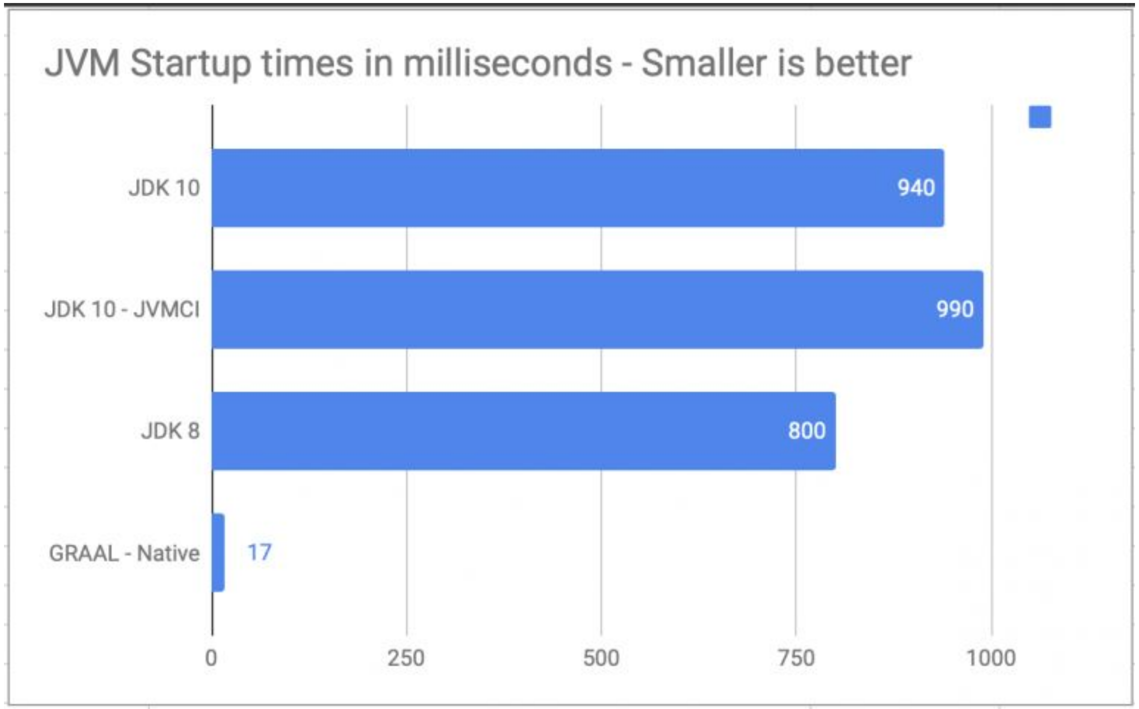
Cloud-native Applications

- **Distribution**
  - Containerized applications are de facto
  - Microservices makes development more efficient
  - API-driven development mitigates integration risks (remember Postel’s law)



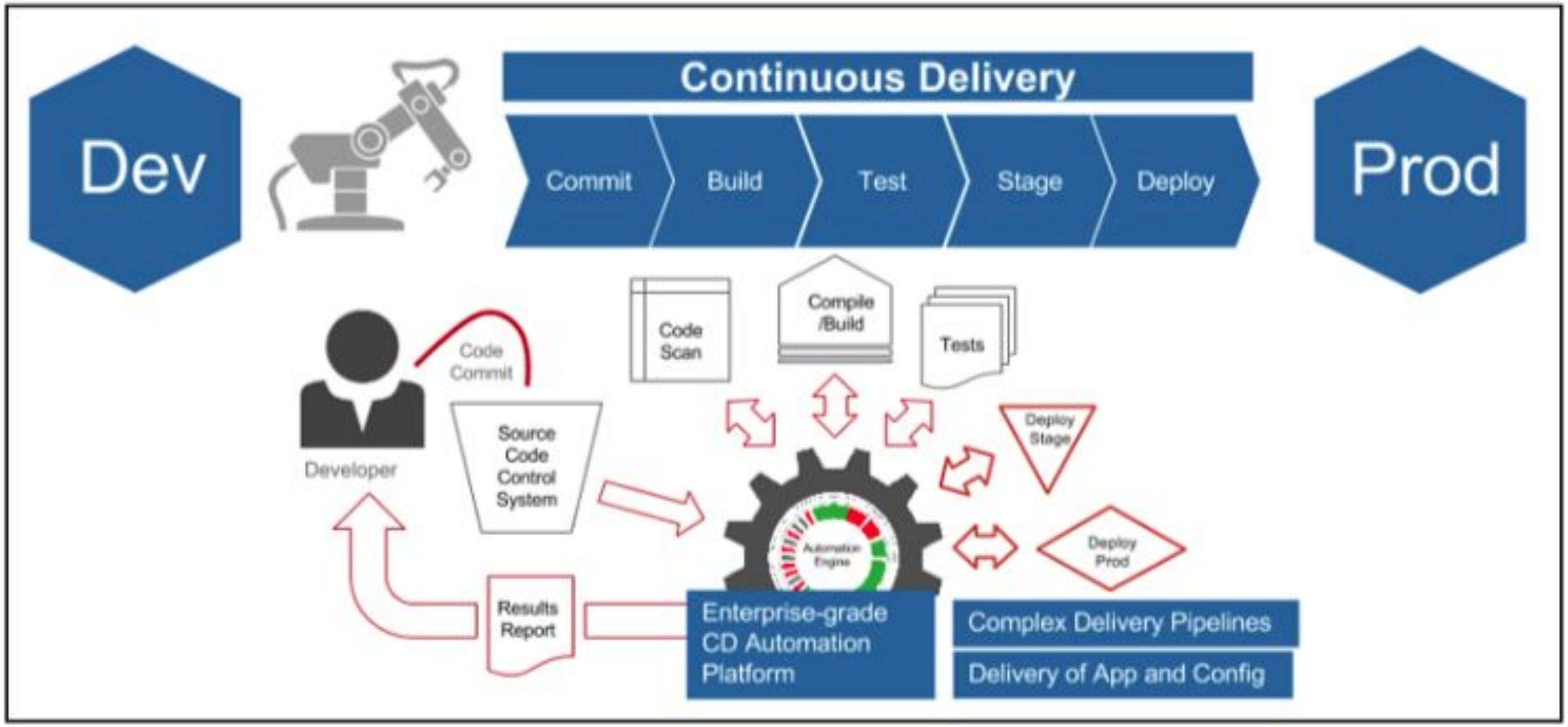
Cloud-native Applications

- **Performance**
  - Optimized for speed & performance
  - Originally platforms like JVM was optimized for high throughput (e.g. slow boot up to create low latency for later requests)



Cloud-native Applications

- **Automation**
  - Automated DevOps tasks
  - Event-driven workflows
  - No human interaction



Dev

Continuous Delivery

Commit

Build

Test

Stage

Deploy

Prod

Developer

Code Commit

Source Code Control System

Code Scan

Compile /Build

Tests

Deploy Stage

Deploy Prod

Automation Engine

Results Report

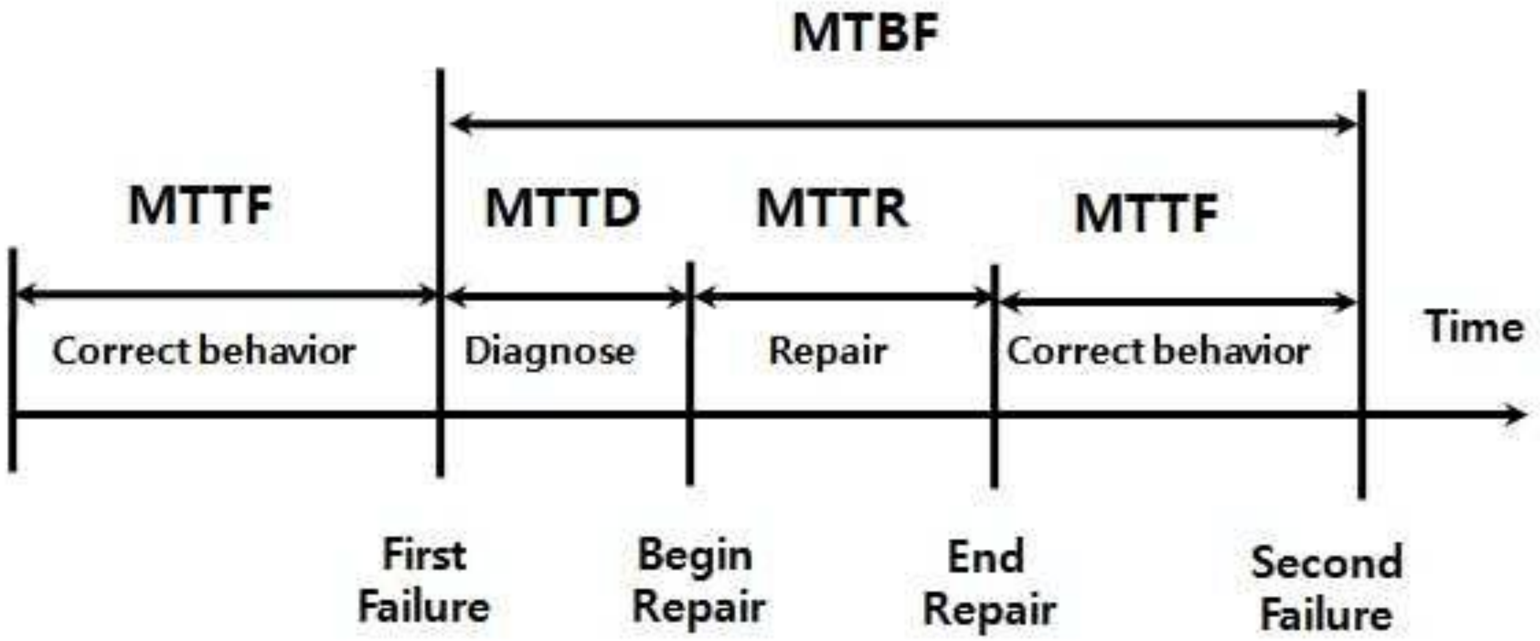
Enterprise-grade CD Automation Platform

Complex Delivery Pipelines

Delivery of App and Config

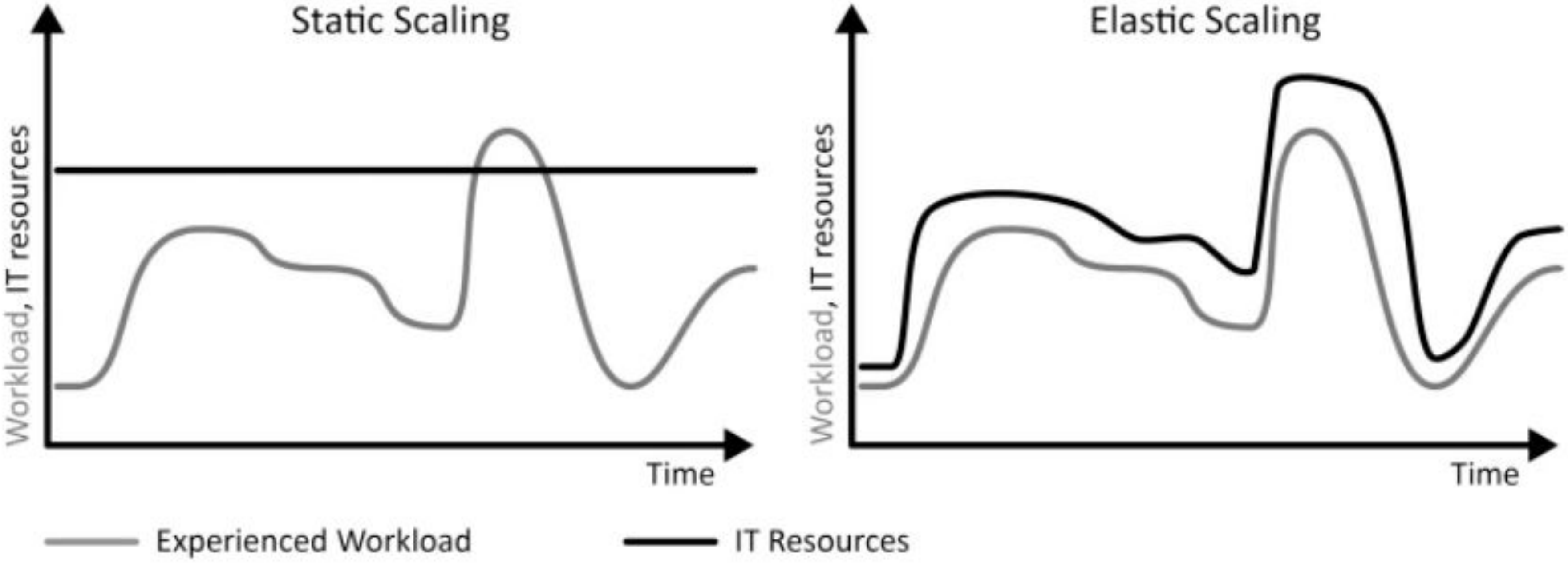
Cloud-native Applications

- **Resiliency**
  - Cloud-native applications are resilient by design
  - You must embrace the partial failures that will certainly occur eventually



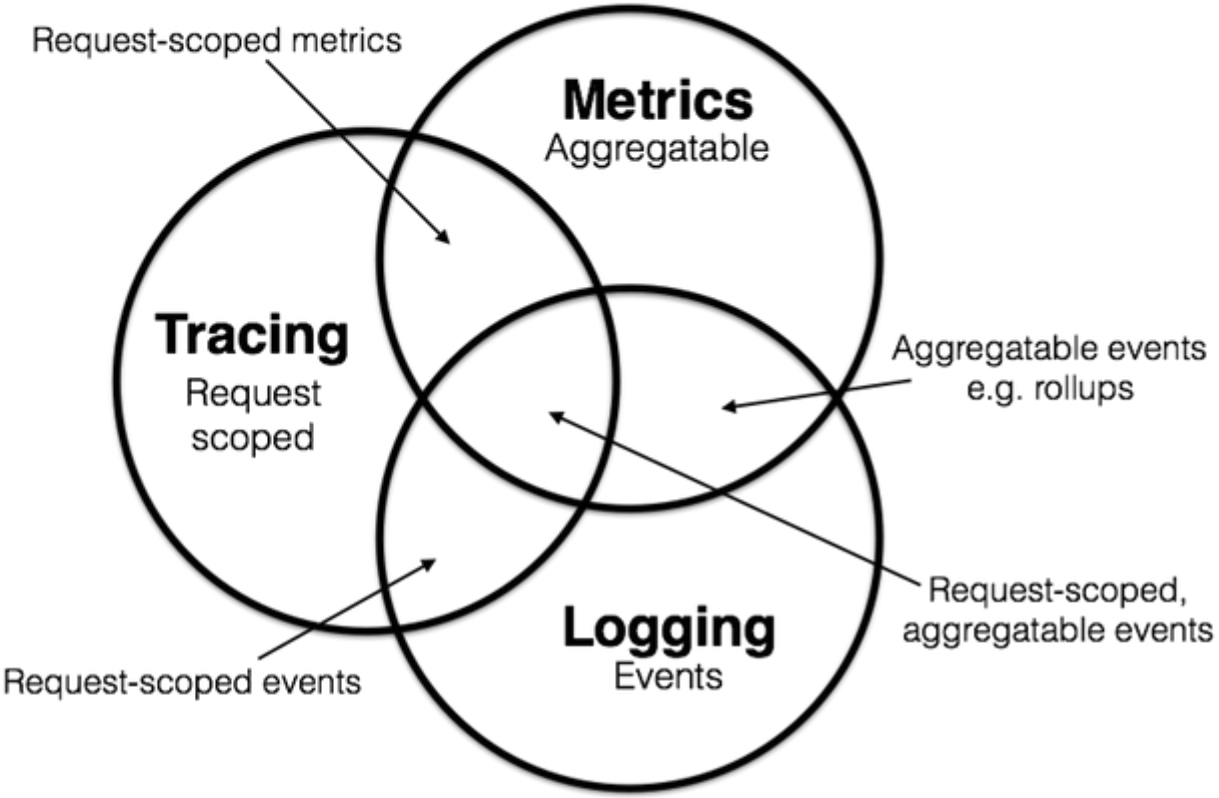
Cloud-native Applications

- **Elasticity**
  - Cloud-native applications reacts to stimuli



Cloud-native Applications

- **Observability**
  - Cloud-native applications are highly observable
  - The three pillars: logs, metrics, traces



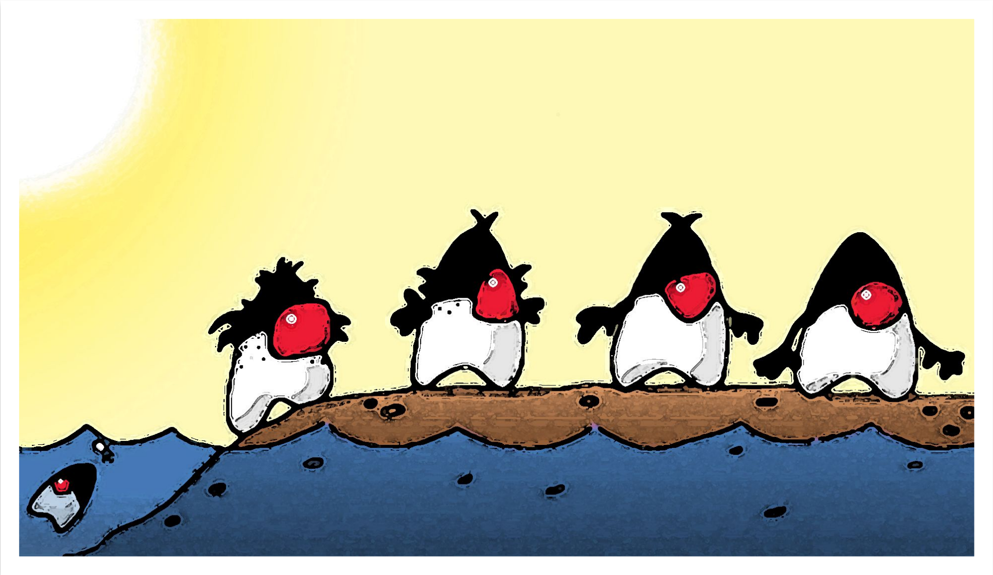
https://peter.bourgon.org/blog/2017/02/21/metrics-tracing-and-logging.html

(Local) Development Env.

- **Minikube / Microk8s / K3s**
  - Stand-alone/simple/single-node K8s
- **Localstack**
  - A fully functional local AWS cloud stack (offline)
- **Testcontainers**
  - Java library that supports JUnit tests, providing lightweight, throwaway instances of common databases, tools, or anything else that can run in a Docker container.
- **Eclipse JKube**
  - Generates and deploys Kubernetes/OpenShift manifests at compile time.
- **Locust**
  - Load testing tool

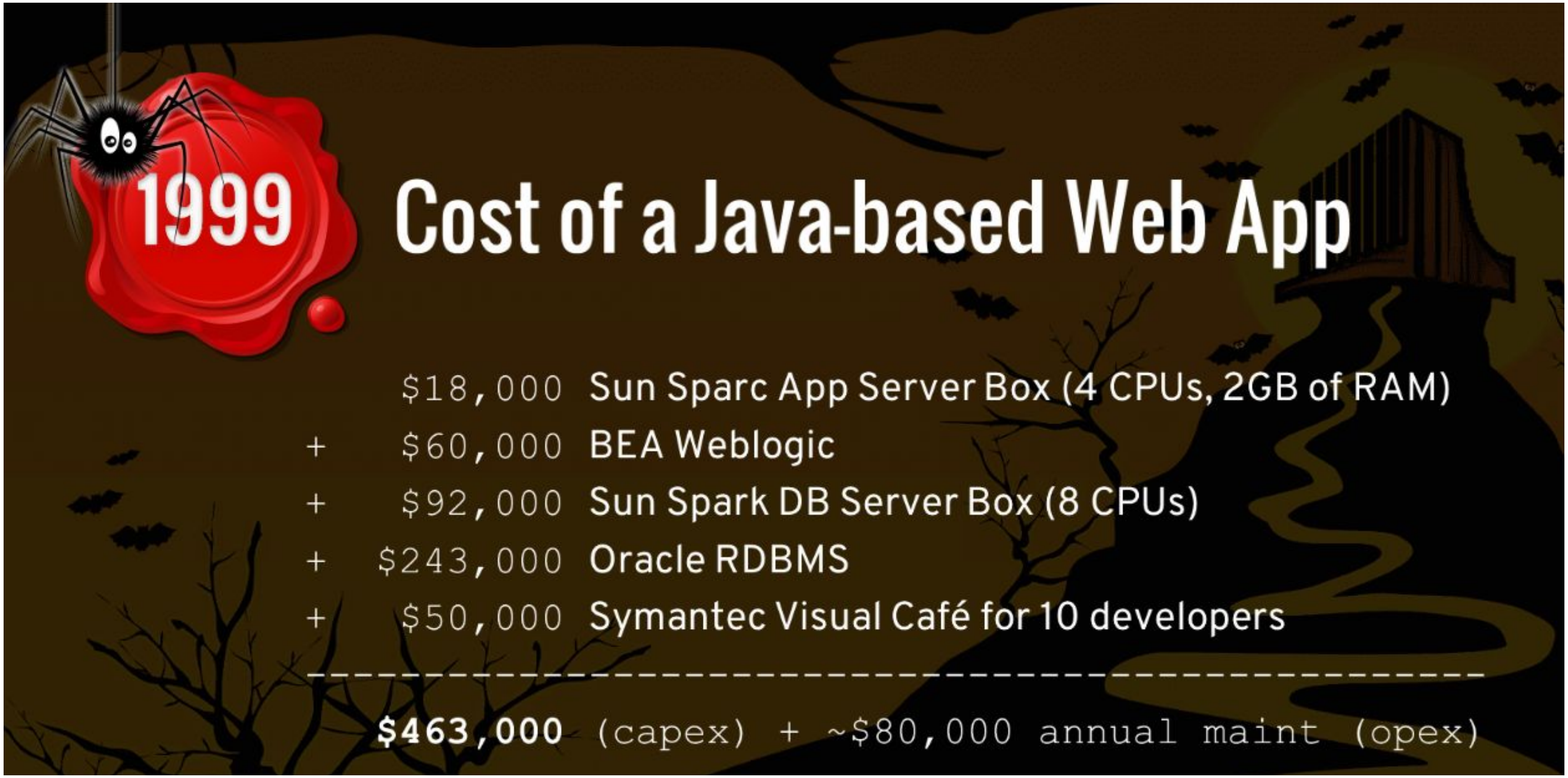
Java Ecosystem

- **Why Java?**
  - Most Popular Programming Languages 1965 - 2019
  - https://www.tiobe.com/tiobe-index/
  - Very well established ecosystem (tooling, community, etc.)
  - Still evolving to cope with emerging languages/platforms/ecosystems...



Java Ecosystem

- **~20 years ago**



1999

Cost of a Java-based Web App

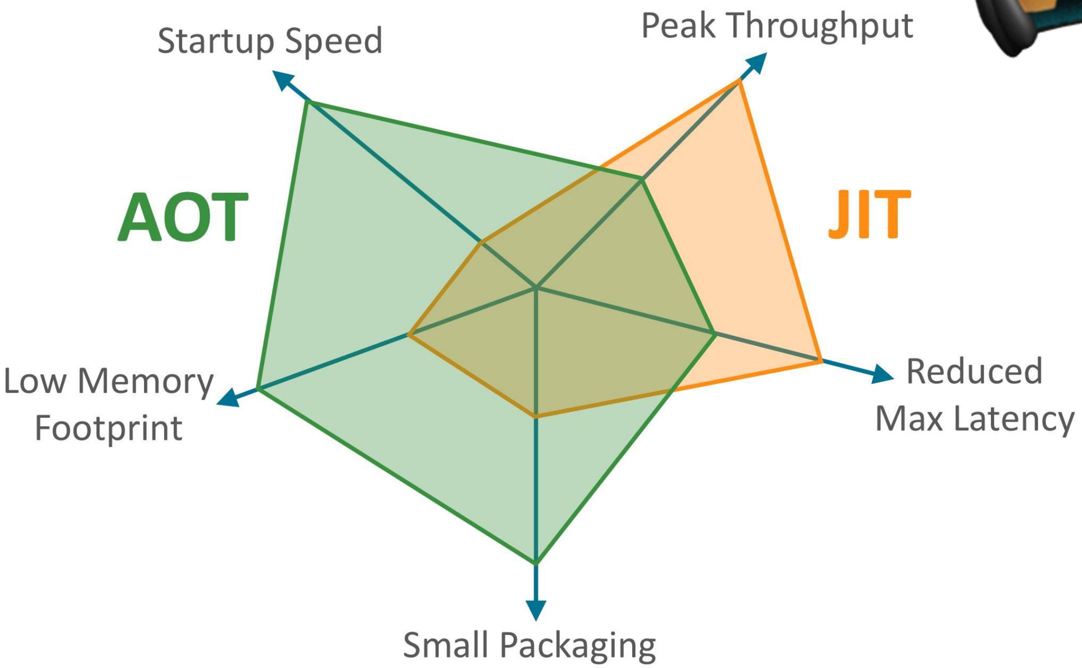
	$18,000	Sun Sparc App Server Box (4 CPUs, 2GB of RAM)
+	$60,000	BEA Weblogic
+	$92,000	Sun Spark DB Server Box (8 CPUs)
+	$243,000	Oracle RDBMS
+	$50,000	Symantec Visual Café for 10 developers

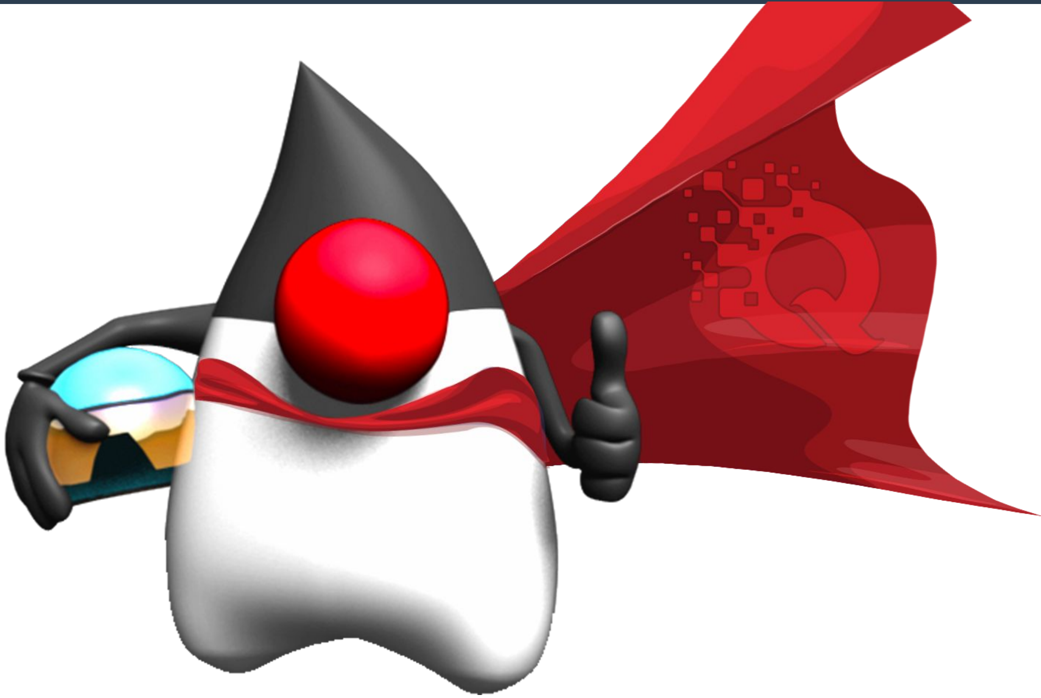
$463,000 (capex) + ~$80,000 annual maint (opex)

source: http://bit.ly/clujnapoca2019

Java Ecosystem

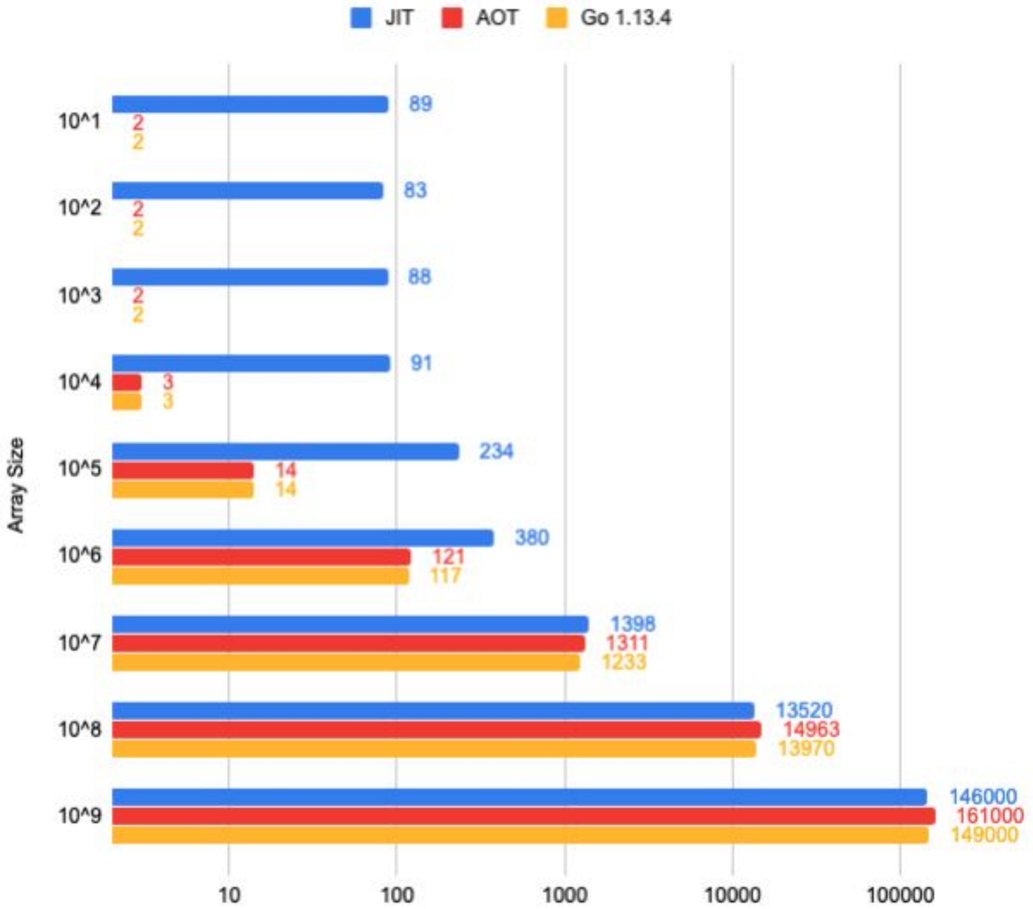
- **Quarkus**
  - Supersonic, subatomic Java
  - https://quarkus.io/
  - GraalVM images
    - AOT vs JIT ([src]())





Quarkus

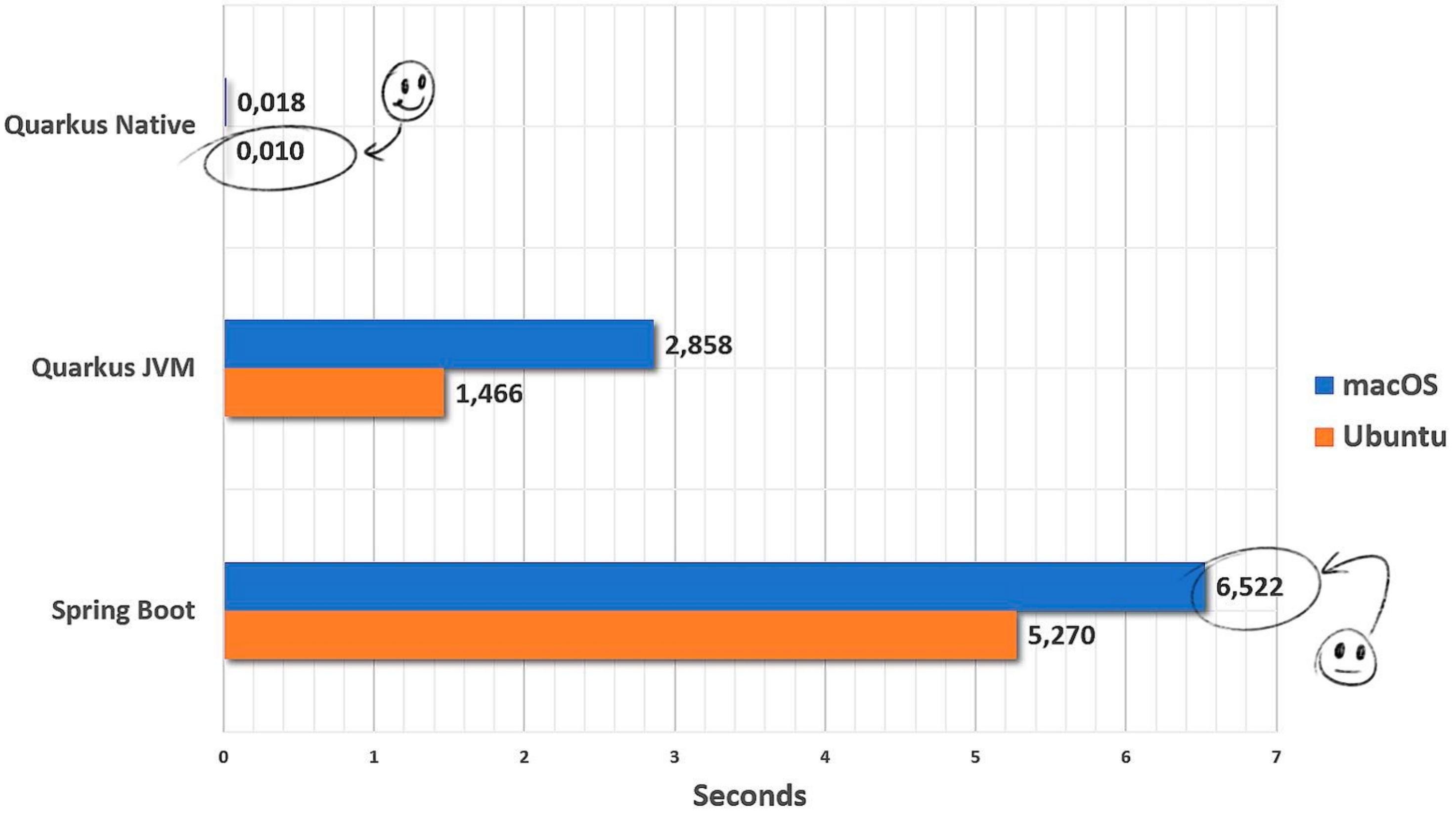
- **Benchmarks**
  - Execution time (Quicksort / ms / logarithmic scale)



http://macias.info/entry/201912201300_graal_aot.md

Quarkus

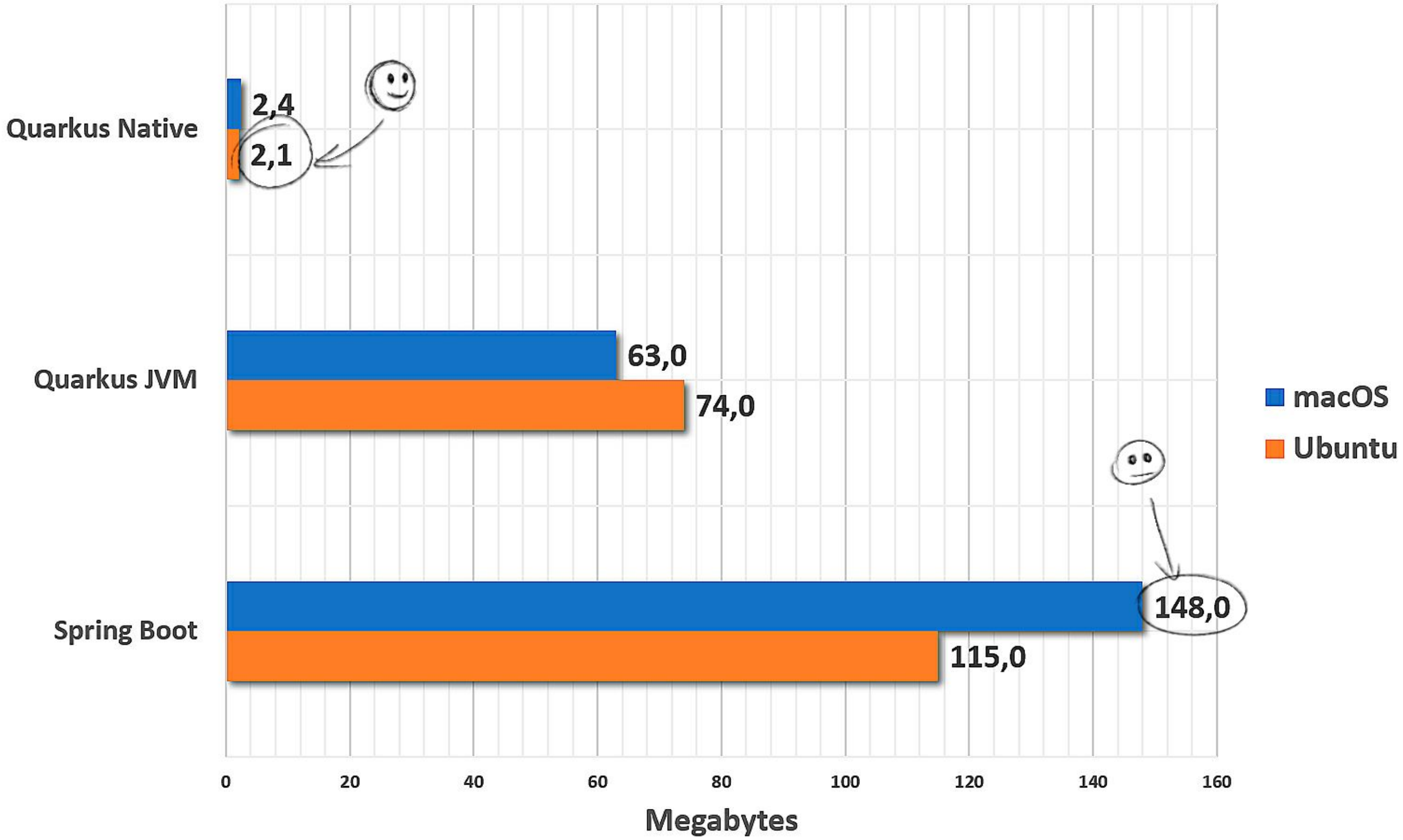
- **Benchmarks**
  - Startup time



https://dzone.com/articles/microservices-quarkus-vs-spring-boot

Quarkus

- **Benchmarks**
  - Memory footprint



https://dzone.com/articles/microservices-quarkus-vs-spring-boot

Q/A