Micronaut in practice

Agenda

- Introduction
- Spring(Boot) basics
- Micronaut basics
- Spring(Boot) vs Micronaut
- Micronaut in practice

Me

- Software Engineer, experienced in Java
- Java Trainer
- Tesco Technology
- CzirjakTech
- CodeJar

SpringBoot basics

- Dependency Injection and Inversion of Control (IoC)
  - C.W.C.M.
- Aspect Oriented Programming (AOP)
- Sensible Defaults and Auto-Configuration

- Proxies
- Heavy start-up time
- Reflection API

Micronaut basics

- An efficient compile-time dependency-injection container
  - Minimal start-up time and memory usage
- A reactive HTTP server & client
  - Netty
- Cloud-native, microservice systems
- Annotation processors
- Ahead of Time compilation

REST layer

- `@RestController`
- `@FeignClient`
- CRUD (Spring-Data-REST)
- MVC or WebFlux

- `@Controller`
- `@Client`
- ?
- “Reactive by nature”

Service layer

- **Default scope:** `Singleton`
- `@Autowire`
- `@Configuration`
- `...`

- **Default scope:** `Prototype`
- `@Inject`
- `@Factory`
- `...`

Persistence layer

- JPA, Hibernate
- JDBC
- Query: at runtime

- JPA, Hibernate
- JDBC

Query: at compile time

`Note: @Where("@.enabled = true")`

Testing & Monitoring

- SpringBootTest
- Actuator
- More configurations

- MicronautTest
- Built-In-Endpoints
- Micrometer

Security

- Spring Security
- OAth2
- JWT
  - Cookie vs Bearer

- Micronaut Security
- OAth2
- JWT
  - Cookie vs Bearer

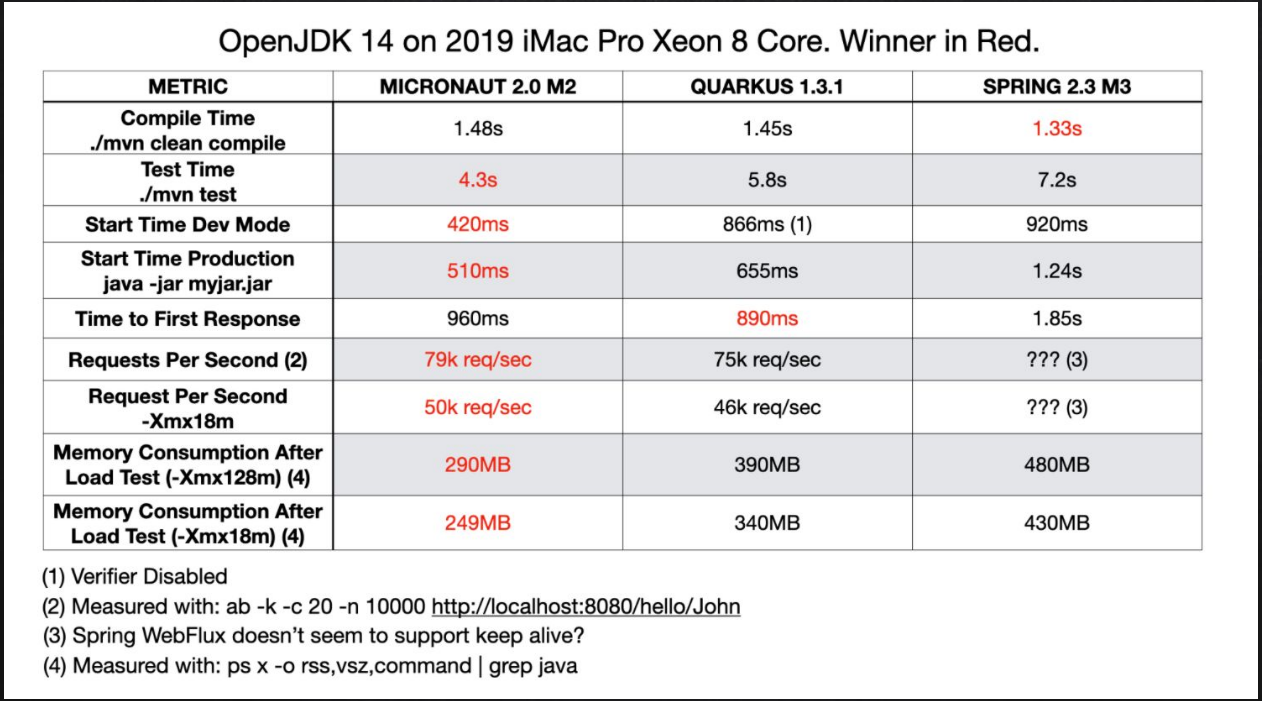
Micronaut - Cloud Native Enabled

- Service discovery
  - eg: Eureka, Consul, Kubernetes
- Client side load-balancing
  - Netflix Ribbon
  - Distributed tracing / configurations
- `@Retryable` on `@Client`

Micronaut - Serverless Functions

- Usage of resources
  - Especially memory
- eg, AWS lambdas

Performance review



OpenJDK 14 on 2019 iMac Pro Xeon 8 Core. Winner in Red.

METRIC	MICRONAUT 2.0 M2	QUARKUS 1.3.1	SPRING 2.3 M3
Compile Time ./mvn clean compile	1.48s	1.45s	1.33s
Test Time ./mvn test	4.3s	5.8s	7.2s
Start Time Dev Mode	420ms	866ms (1)	920ms
Start Time Production java -jar myjar.jar	510ms	655ms	1.24s
Time to First Response	960ms	890ms	1.85s
Requests Per Second (2)	79k req/sec	75k req/sec	??? (3)
Request Per Second -Xmx18m	50k req/sec	46k req/sec	??? (3)
Memory Consumption After Load Test (-Xmx128m) (4)	290MB	390MB	480MB
Memory Consumption After Load Test (-Xmx18m) (4)	249MB	340MB	430MB

(1) Verifier Disabled

(2) Measured with: ab -k -c 20 -n 10000 http://localhost:8080/hello/John

(3) Spring WebFlux doesn't seem to support keep alive?

(4) Measured with: ps x -o rss,vsz,command | grep java

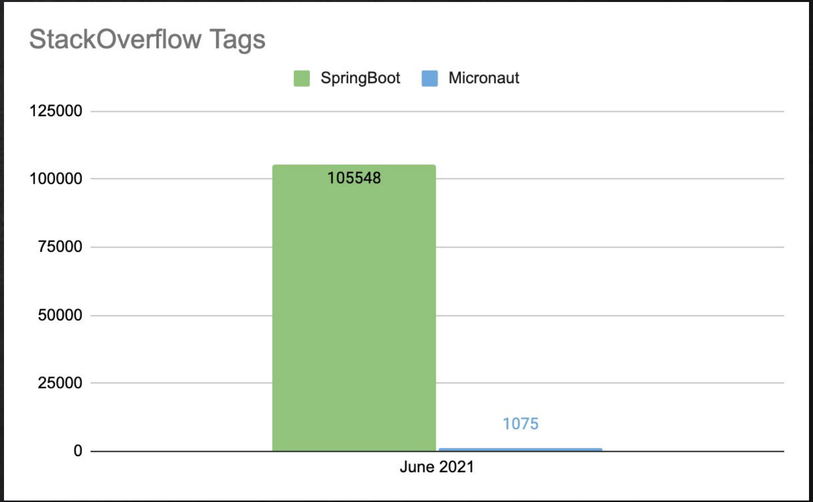
- Source: https://jaxenter.com/micronaut-speed-test-170870.html

Micronaut in practice I. - Example

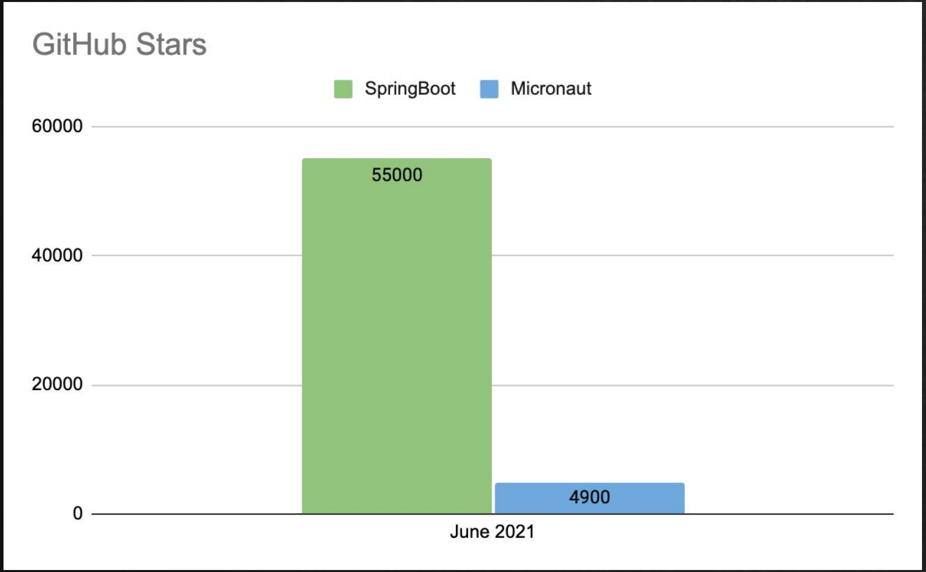
- Microservices
  - Self Service Checkout
    - Limited resources (CPU, Memory, HDD)
  - Cloud components in AKS

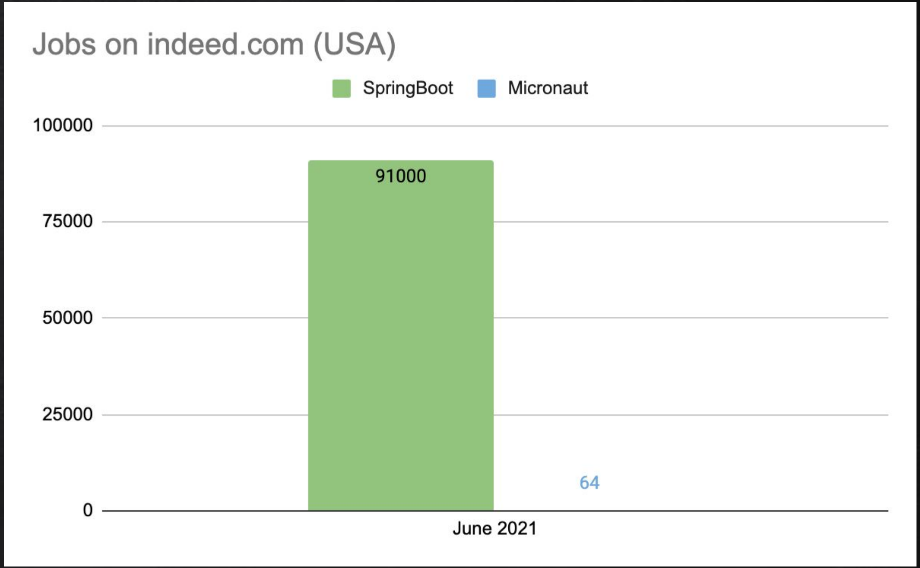
Micronaut in practice II. - Hard times

- Small community
- Lack of documentations
- Missing solutions
  - DB connectivity
- Spring initializer



Community


GitHub Stars
SpringBoot
Micronaut
60000
40000
20000
0
55000
4900
June 2021


Jobs on indeed.com (USA)
SpringBoot
Micronaut
100000
75000
50000
25000
0
91000
64
June 2021

Summary - Micronaut ++

- Fast application startup time
- Low runtime memory footprint
- Minimal use of reflection and proxies
- Few external dependencies
- Simple and fast application tests