Buzzwords from Oracle Code One - 2019

GraalVM

Quarkus/Micronaut/Helidon

Evolution of Server Side Java – Java EE to Jakarta EE

Microprofile

Cloud Native Architectures with Service Mesh

JDK Features overview – Versions - 9,10,11,12,13

Contributing to Opensource

Graal VM – What is it ?

Universal
Virtual
Machine

Start your programs
Blazing fast

Written in
Java and is
Opensource

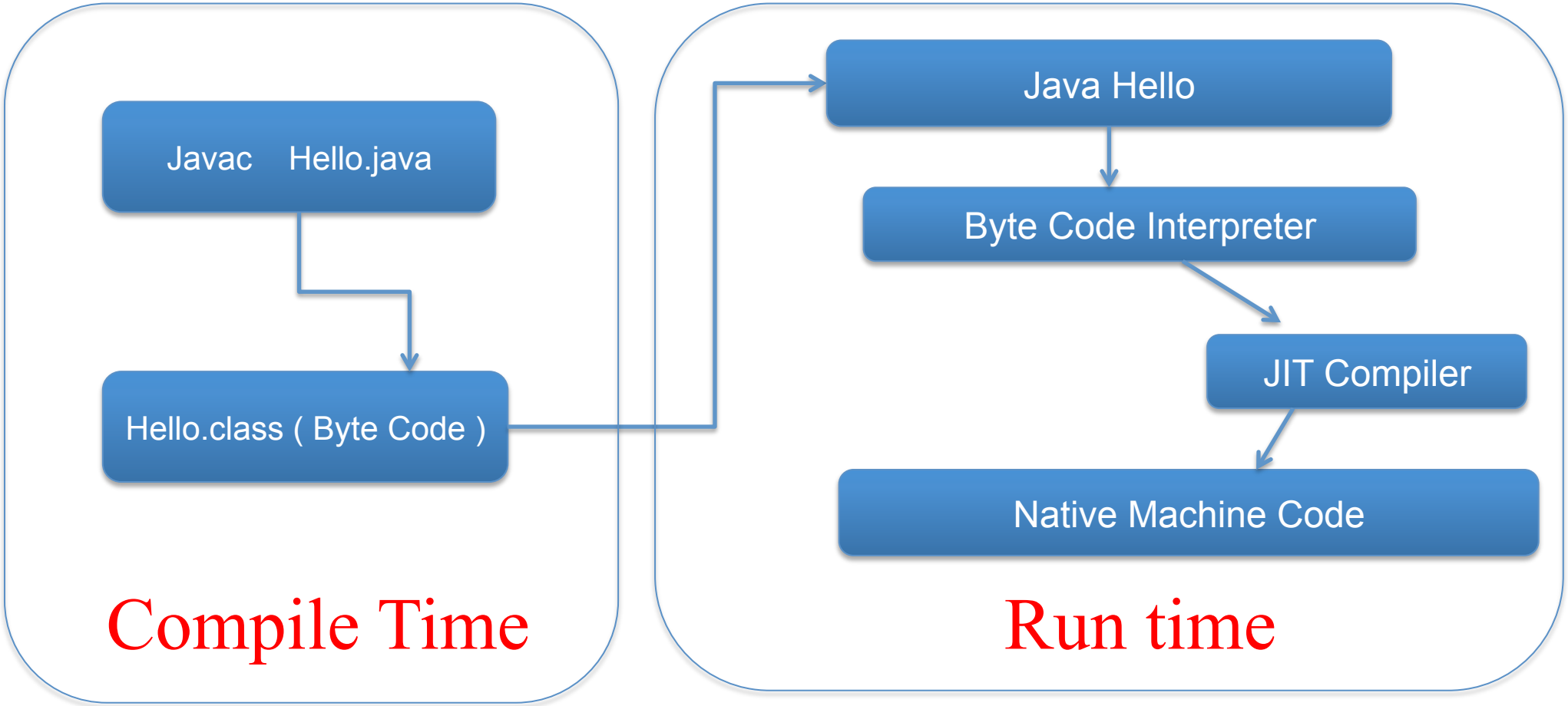
Very low
memory foot
print

Polyglot VM that can run
Java, Javascript, Python,
Ruby, R , Scala, Kotlin etc

Native Image builder

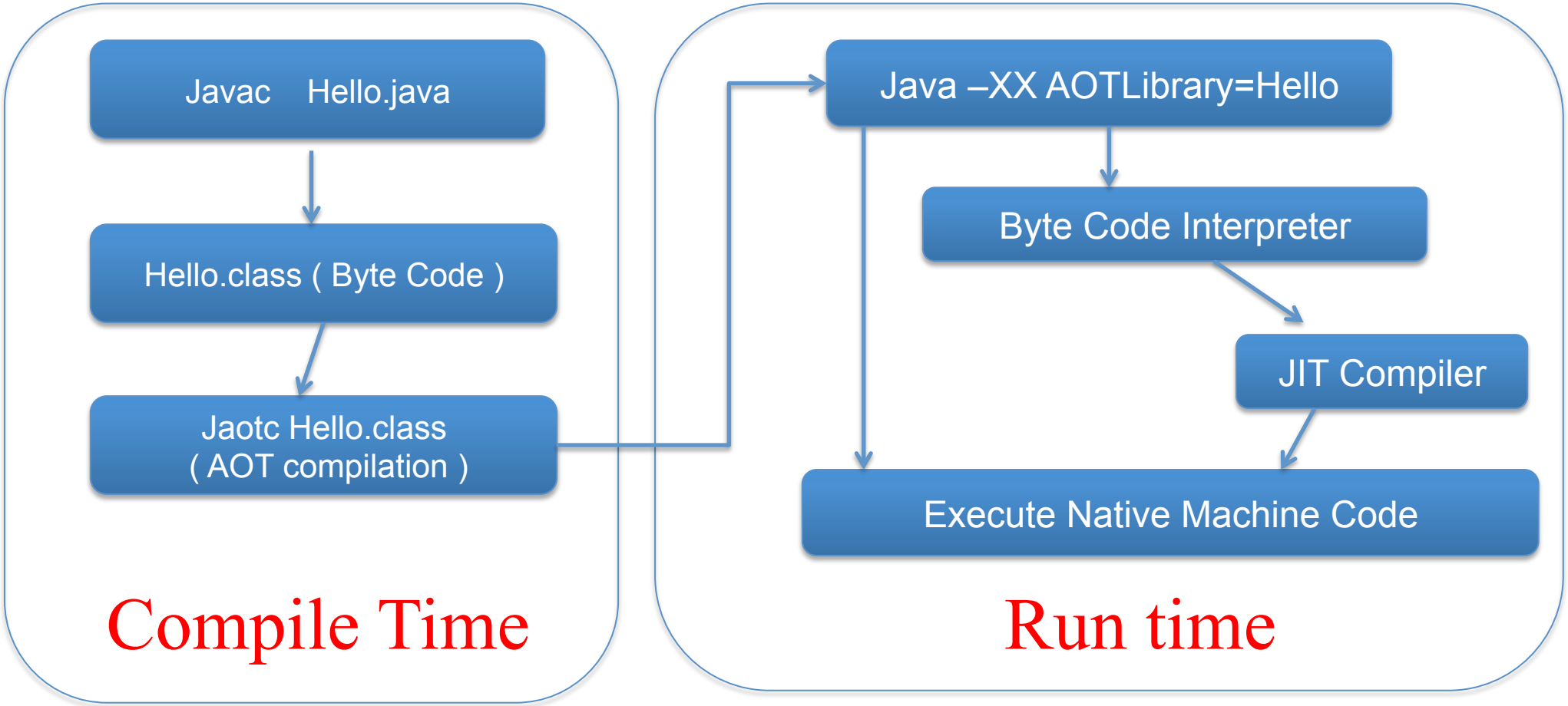
Graal VM – How does it work ?

Traditional JIT compilation at a high-level

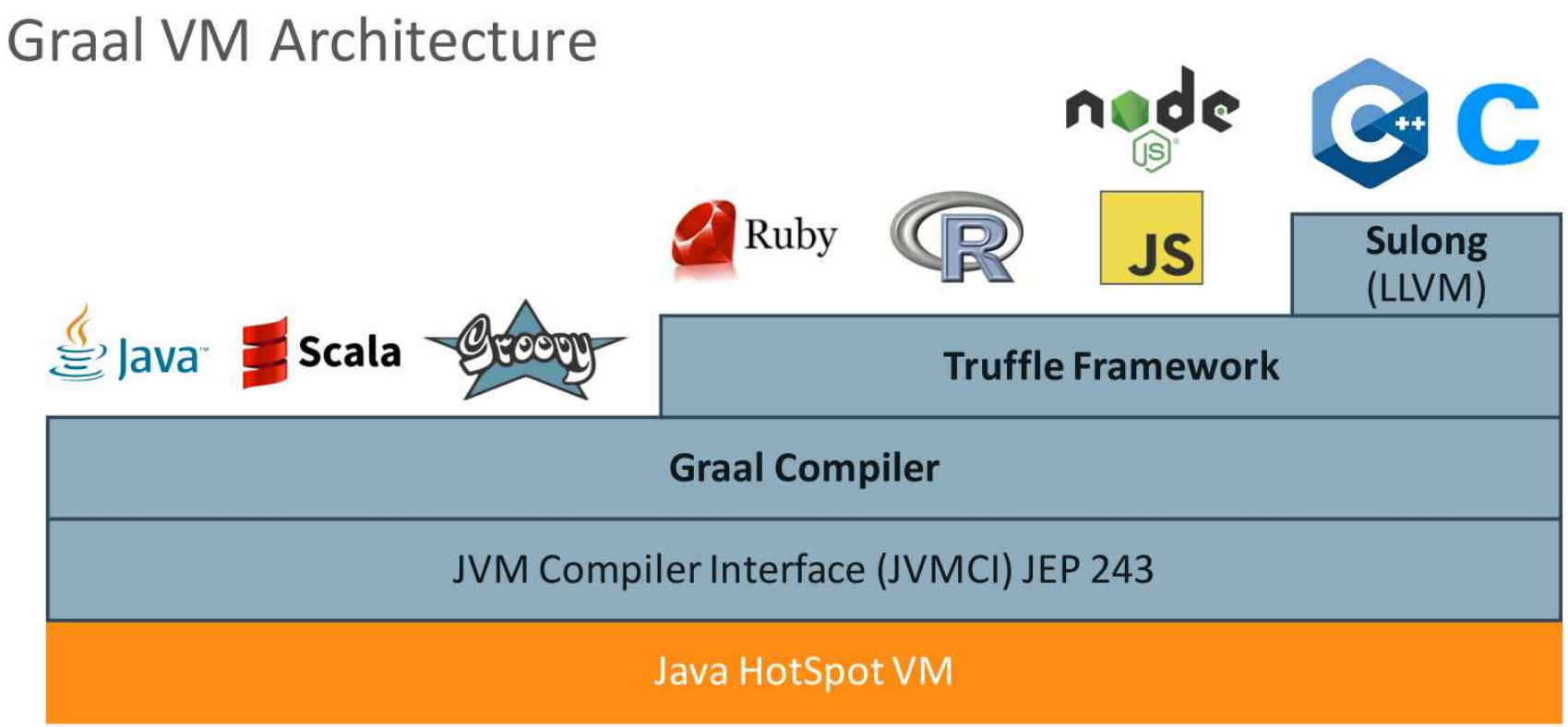


Graal VM – How does it work ?

Graal Compilation at a high-level



Graal VM – Architecture



ORACLE

Graal VM – Demo Hello World

Quarkus/Micronaut/Helidon – What are they ?

- Light weight Frame works
- Built to apply the benefits of GraalVM
- Helidon is Oracle's and Quarkus is Redhat's. Micronaut is kind of independent
- All these projects are intended for micro service development using light weight specification from microprofile and EE8 .
- Anything extra can be added or configured to the project using extensions

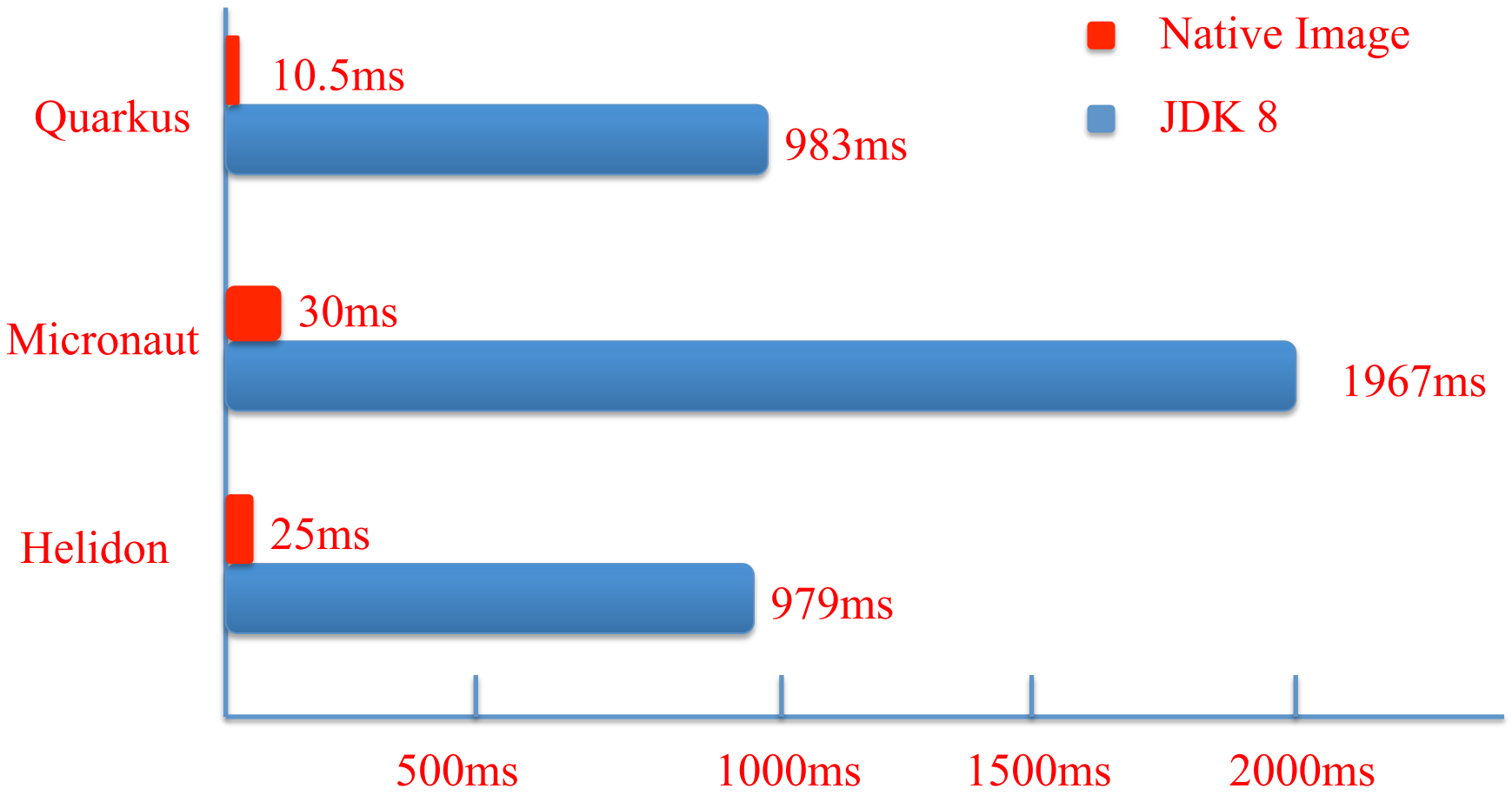
How do they Work ??

Do all the hard work at build time

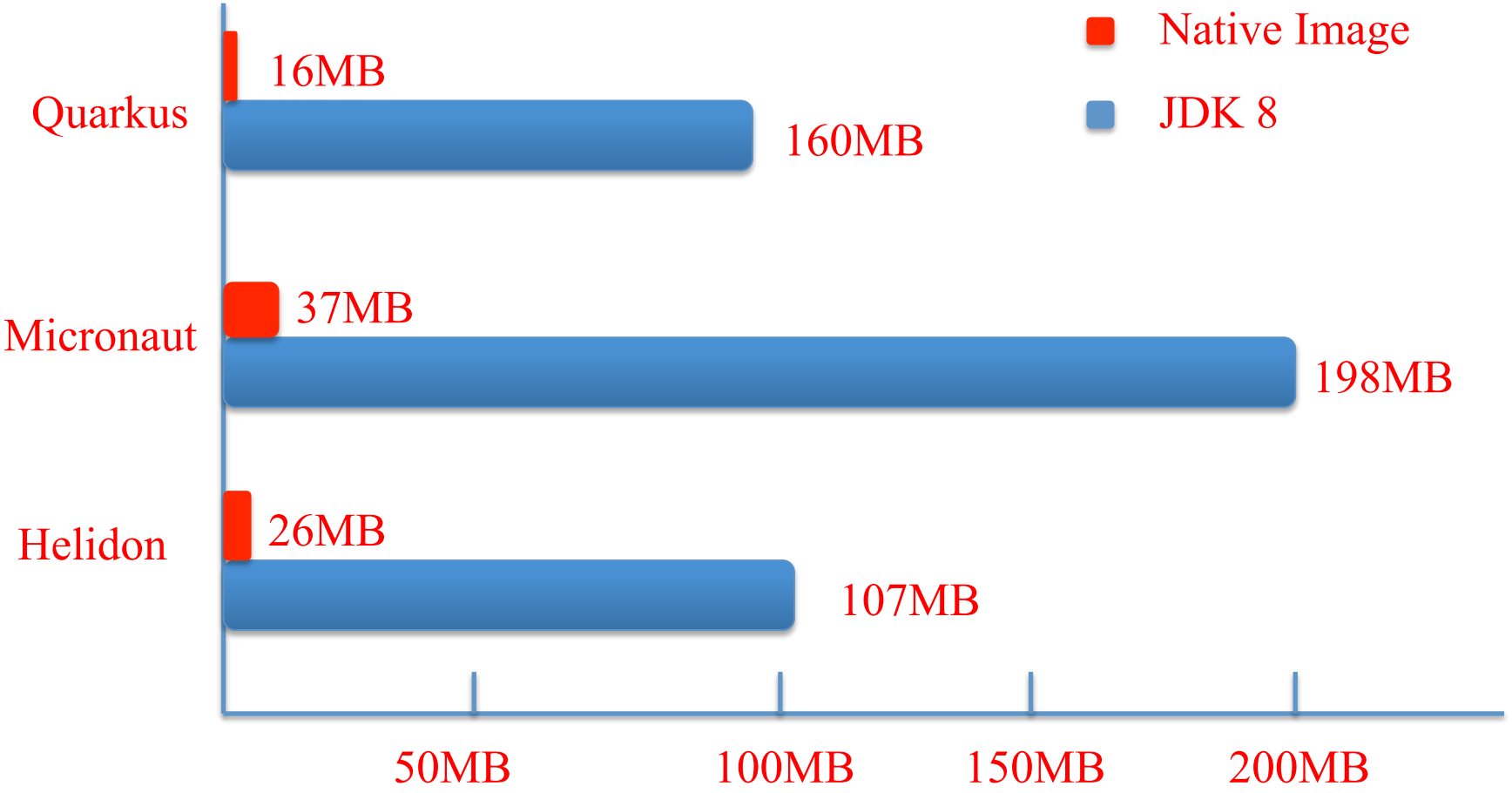
Minimize the amount of Metaspace needed for JIT compilation

Use Native-Image from Graal-VM to build on top of what is already pre-compiled and optimized

Quarkus/Micronaut/Helidon – Startup Time graph



Quarkus/Micronaut/Helidon – Memory Footprint graph



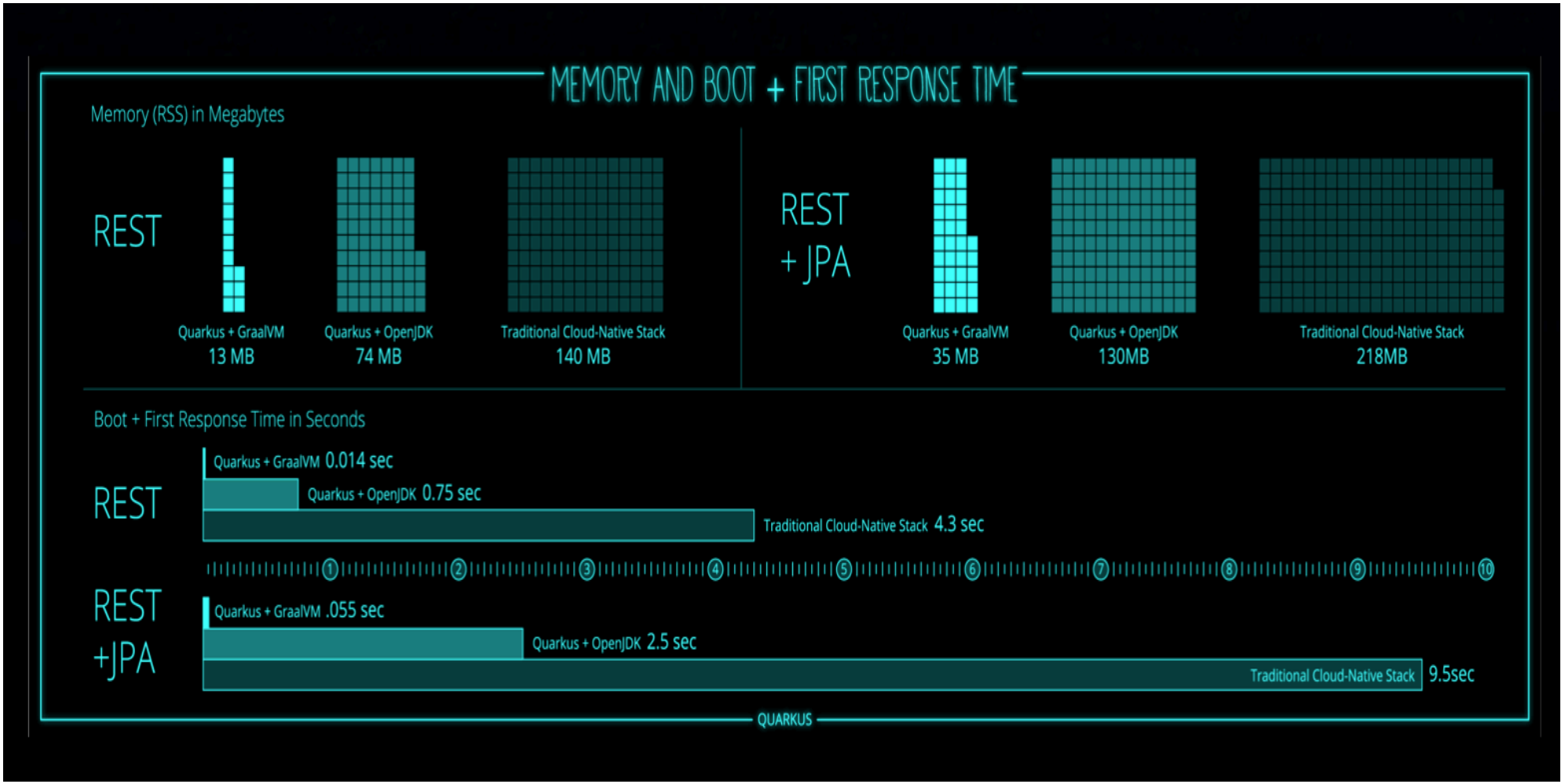
Quarkus/Micronaut/Helidon – Why care about startup and memory footprint

Low compute costs that can be saved on Memory and CPU

Low Down time During the scaling process

Startup time is very important when applications autoscale in large numbers

Quarkus – Supersonic Subatomic Java from Redhat



Quarkus/Micronaut/Helidon - Challenges for adoption

Not Supported

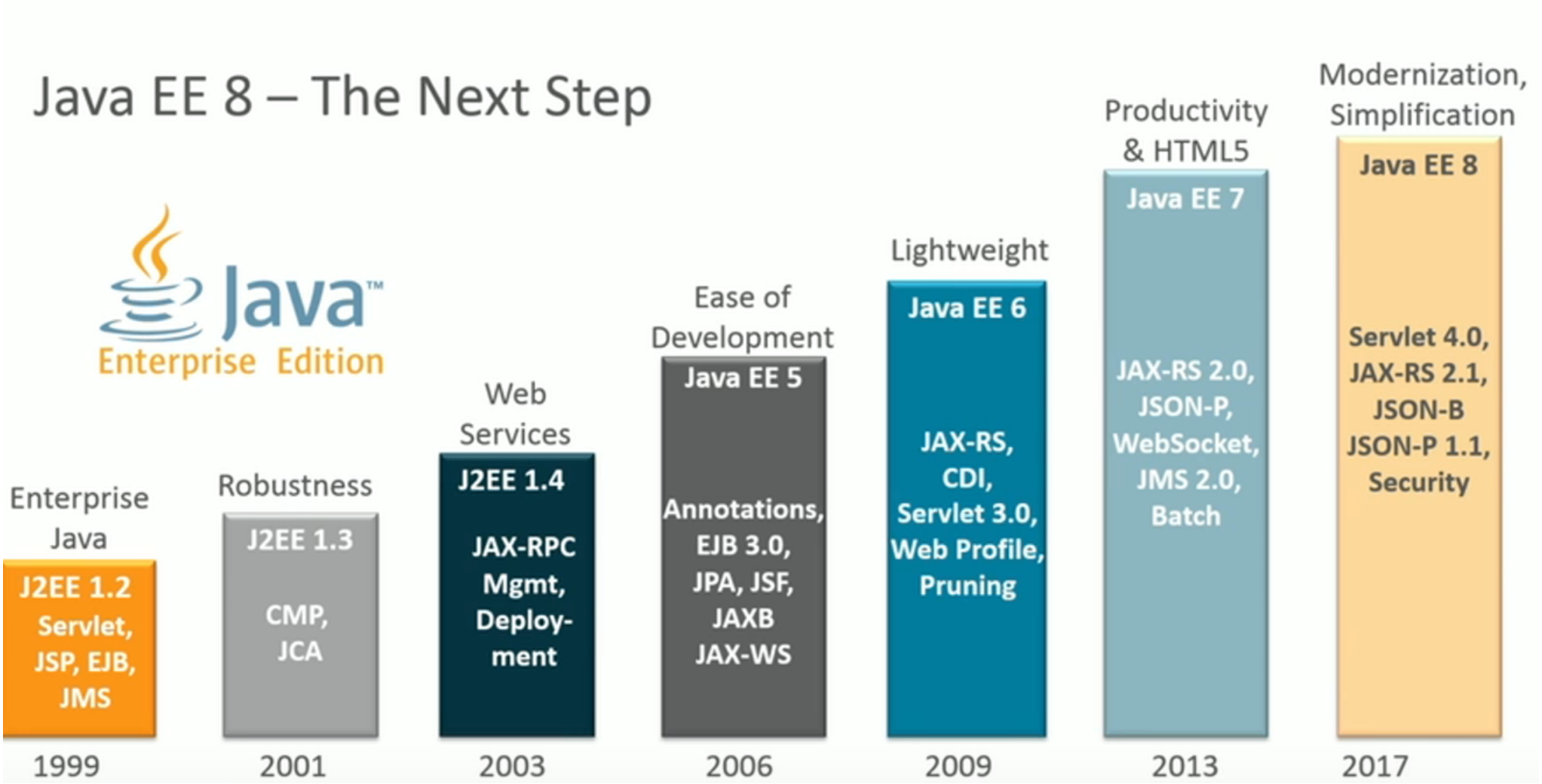
- Dynamic Class loading
- InvokeDynamic & method handles
- Reflection Implementations

OK with Caveats

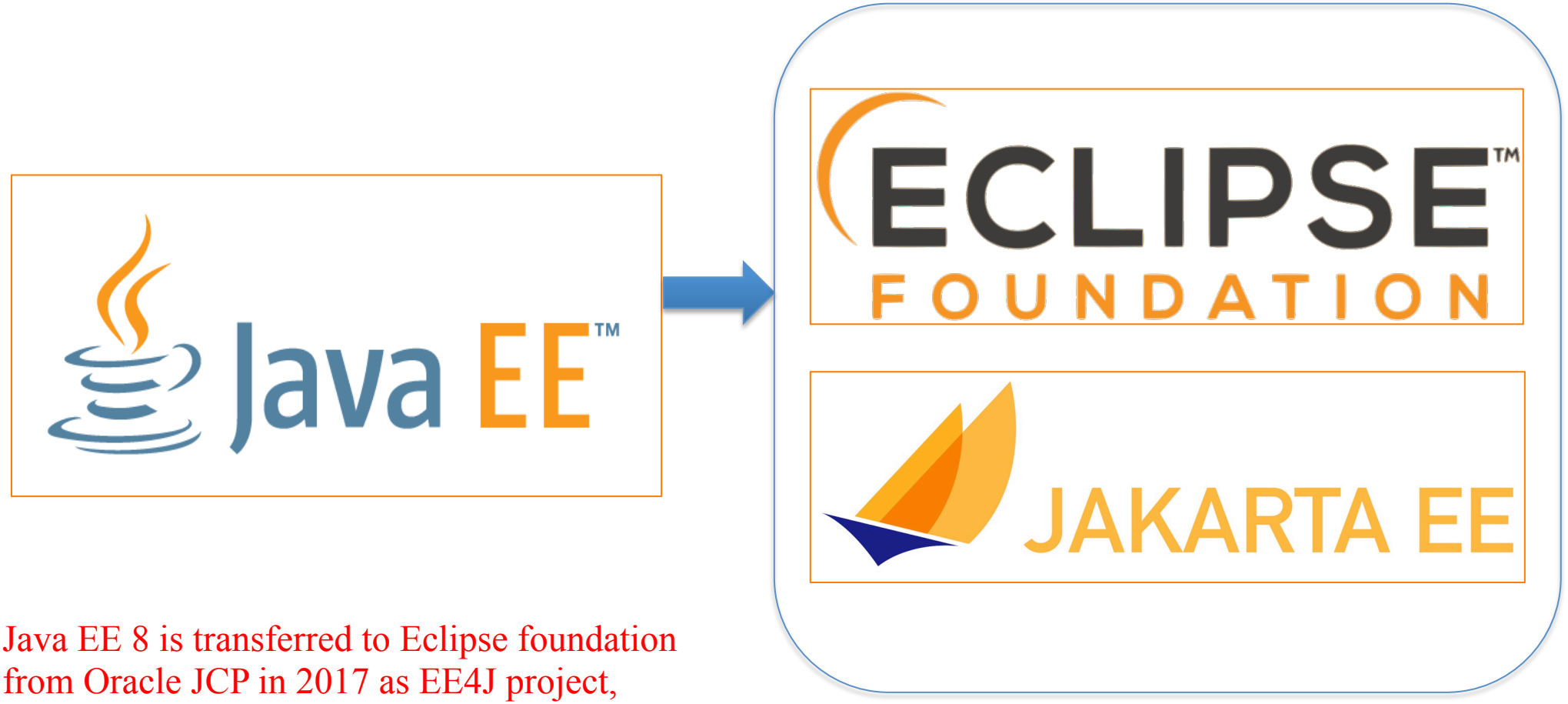
- Reflection(Manual List)
- Dynamic Proxy (Manual list)
- JNI (manual list)
- Static initializers (eager)

Quarkus – Demo

Evolution of Serverside Java – Java EE to JakartaEE



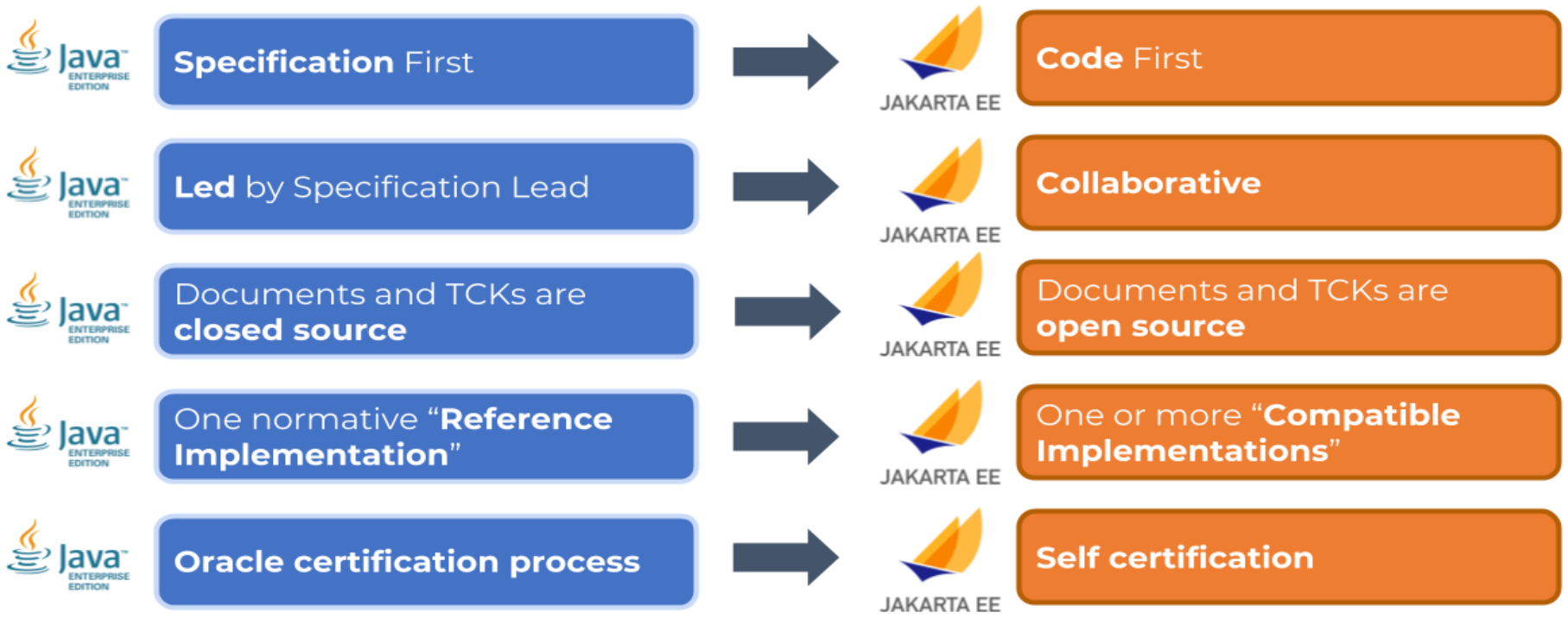
Evolution of Serverside Java – Java EE to JakartaEE



Java EE 8 is transferred to Eclipse foundation
from Oracle JCP in 2017 as EE4J project,
which now is called Jakarta EE 8 (Released Sep 2018).

Evolution of Serverside Java – Java EE to JakartaEE

JCP compared to the EFSP — key points



ECLIPSE FOUNDATION

Microprofile

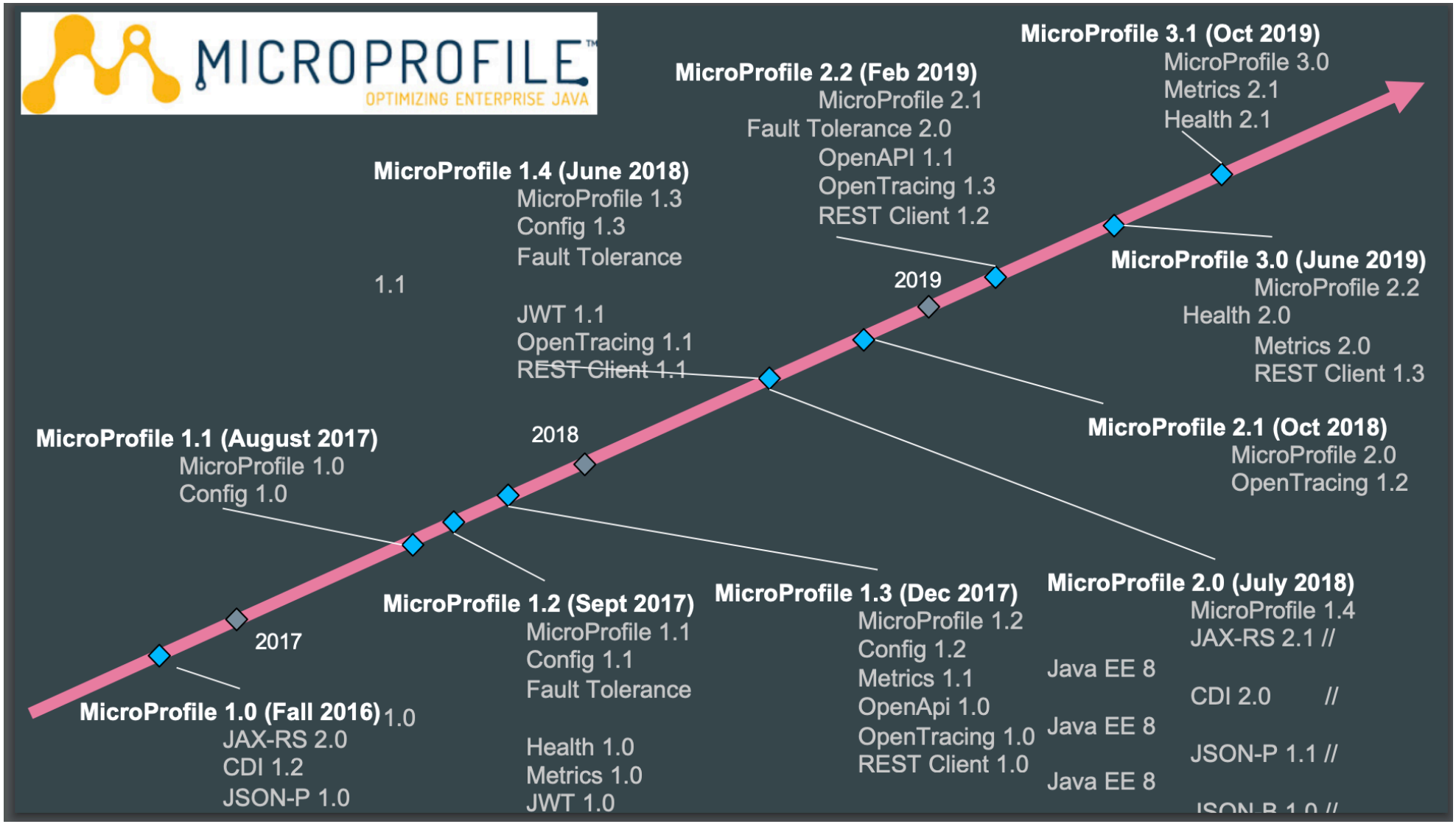
Implements light weight specs which are needed for Modern cloud native apps.

Major vendors contribute toward microprofile and have their own implementations

It includes a subset of Java EE specs and also some new specs needed for cloud native apps.

IBM Open Liberty
Payara Micro
Apache TomEE
Redhat Thorntail (will now be transferring to Quarkus)

Microprofile – Roadmap



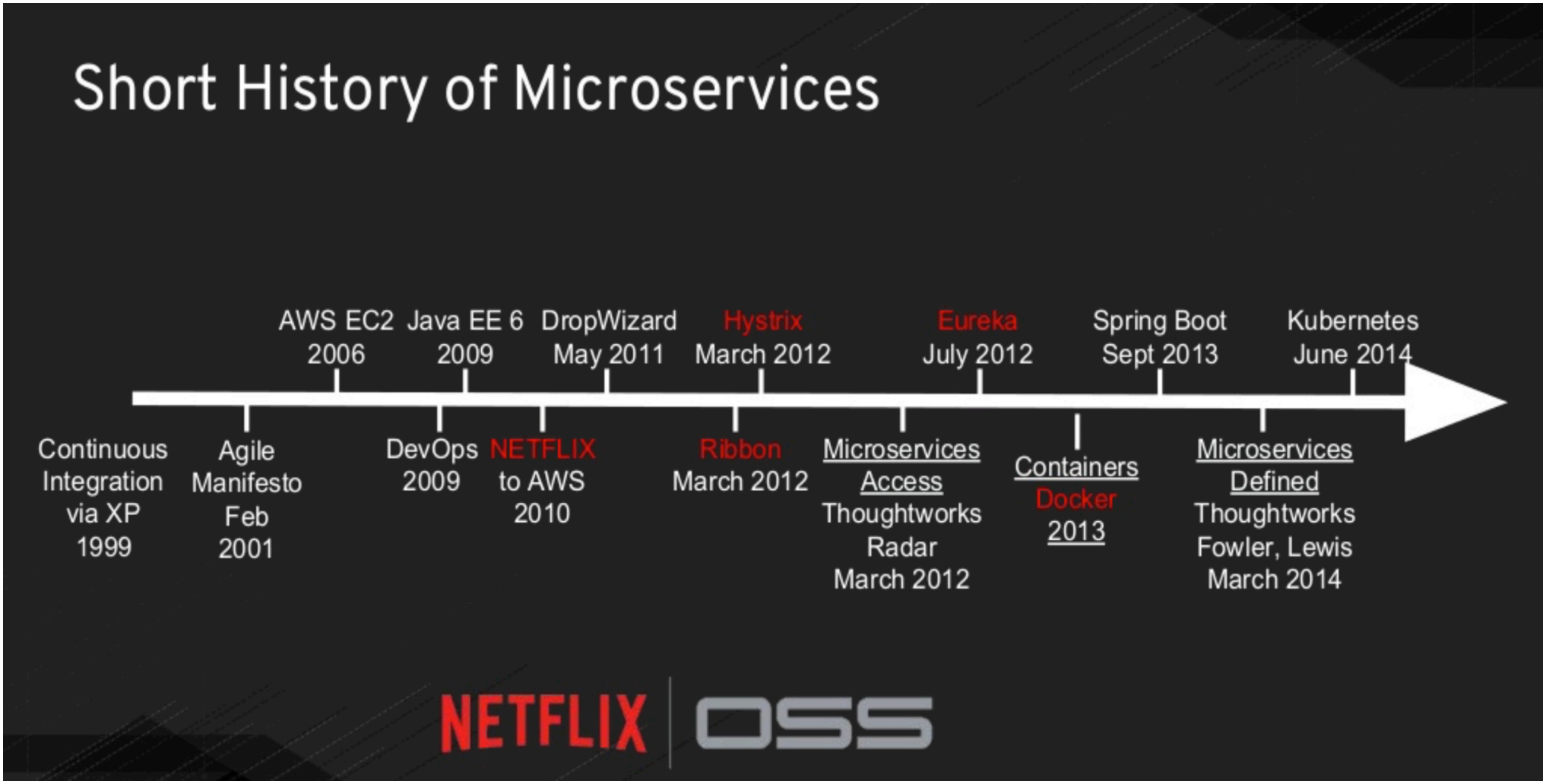
Microprofile and JakartaEE support Graph

Microprofile – Quarkus Cheat Sheet

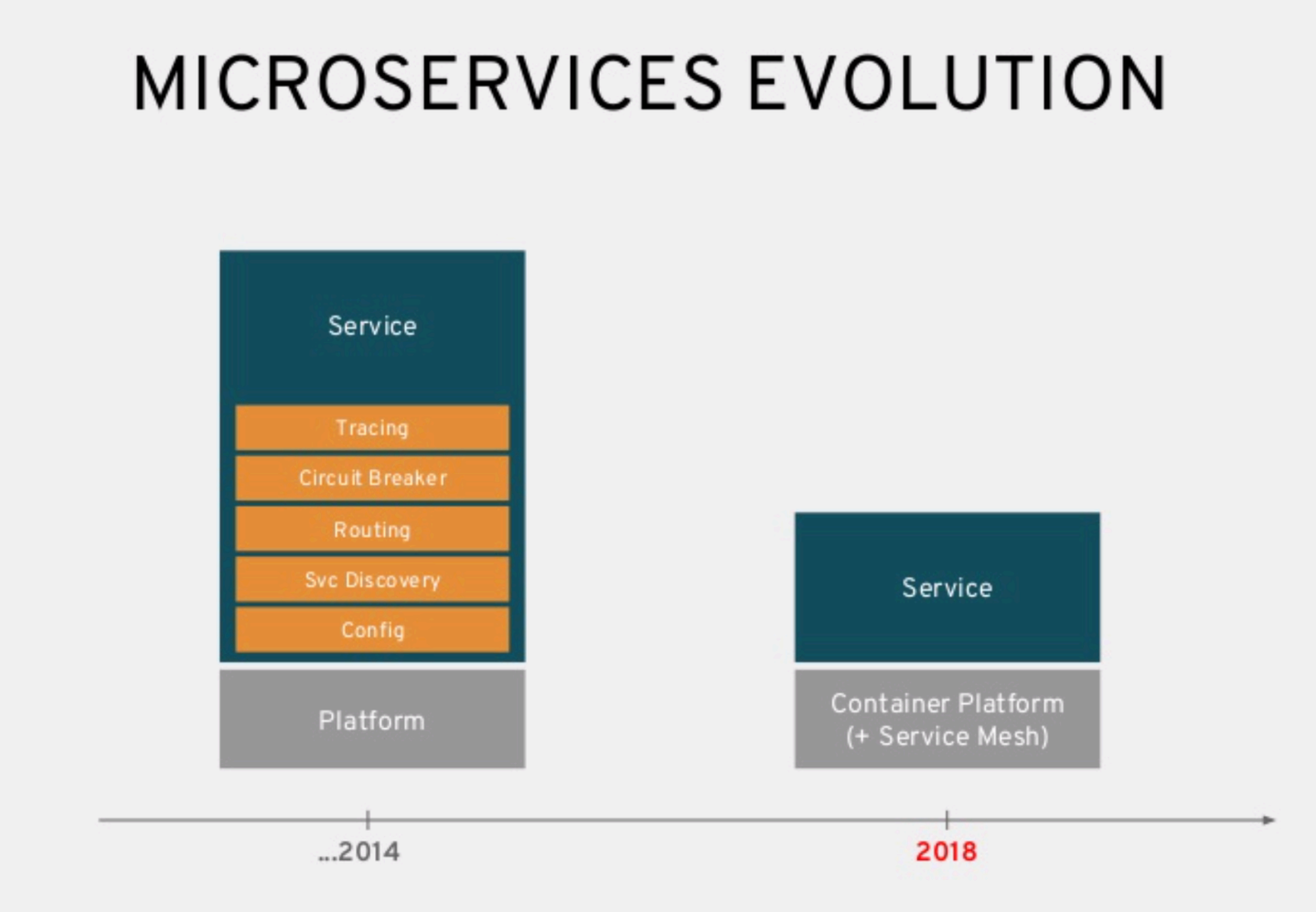
MicroProfile Specification	Quarkus Dependency (Maven group:artifact)
CDI	io.quarkus:quarkus-arc
JAX-RS	io.quarkus:quarkus-resteasy
JSON-P	io.quarkus:quarkus-jsonp
JSON-B	io.quarkus:quarkus-jsonb
Config	N/A. Included by default
Rest Client	io.quarkus:quarkus-smallrye-rest-client
Fault Tolerance	io.quarkus:quarkus-smallrye-fault-tolerance
Health Check	io.quarkus:quarkus-smallrye-health
Metrics	io.quarkus:quarkus-smallrye-metrics
JWT Security	io.quarkus:quarkus-smallrye-jwt
OpenAPI	io.quarkus:quarkus-smallrye-openapi
OpenTracing	io.quarkus:quarkus-smallrye-opentracing
Reactive Streams Operators	io.quarkus:quarkus-smallrye-reactive-streams-operators
Reactive Messaging (Draft)	io.quarkus:quarkus-smallrye-reactive-messaging

Miproprofile - Demo

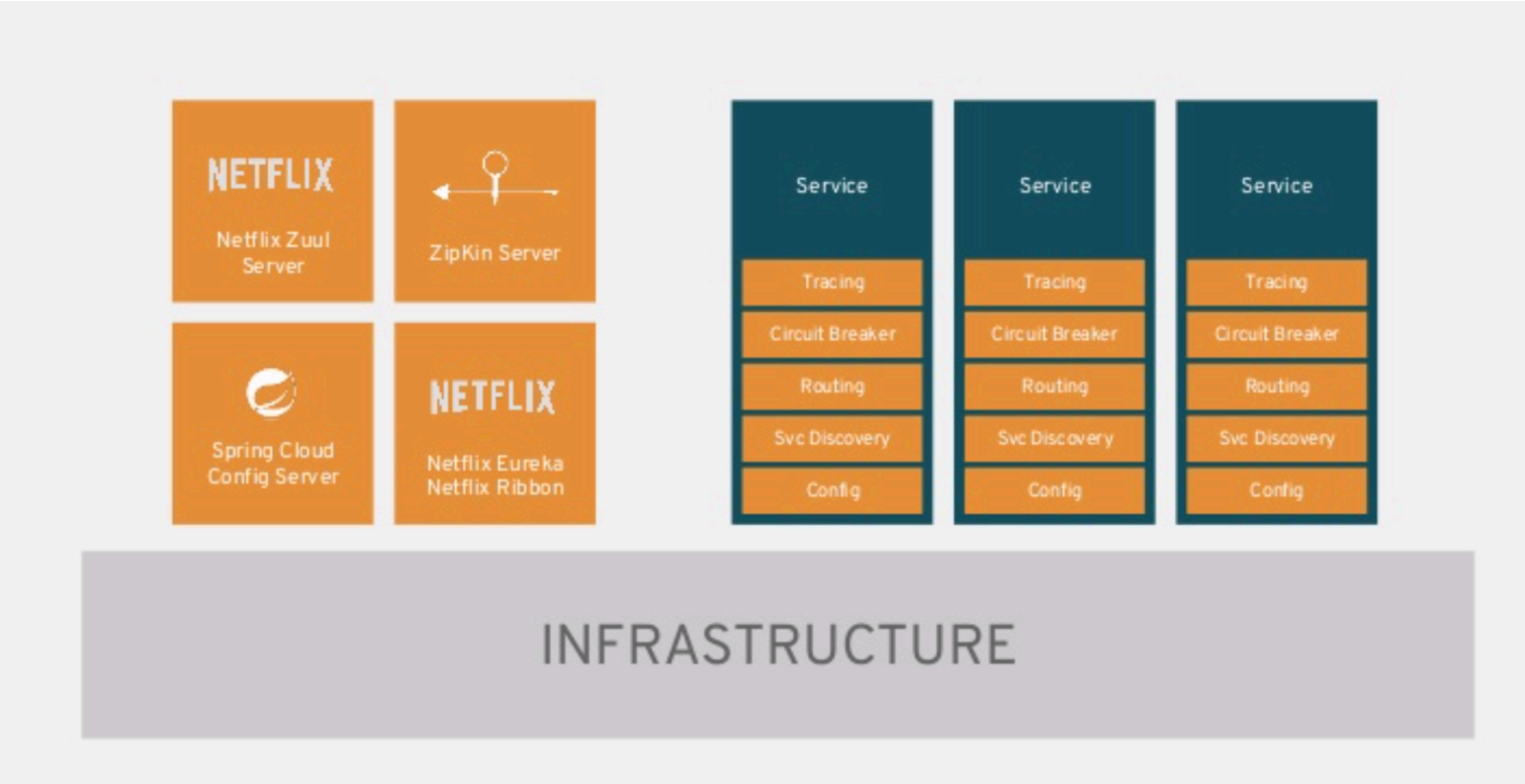
Cloud Native Architectures with Service Mesh- Brief History of Microservices



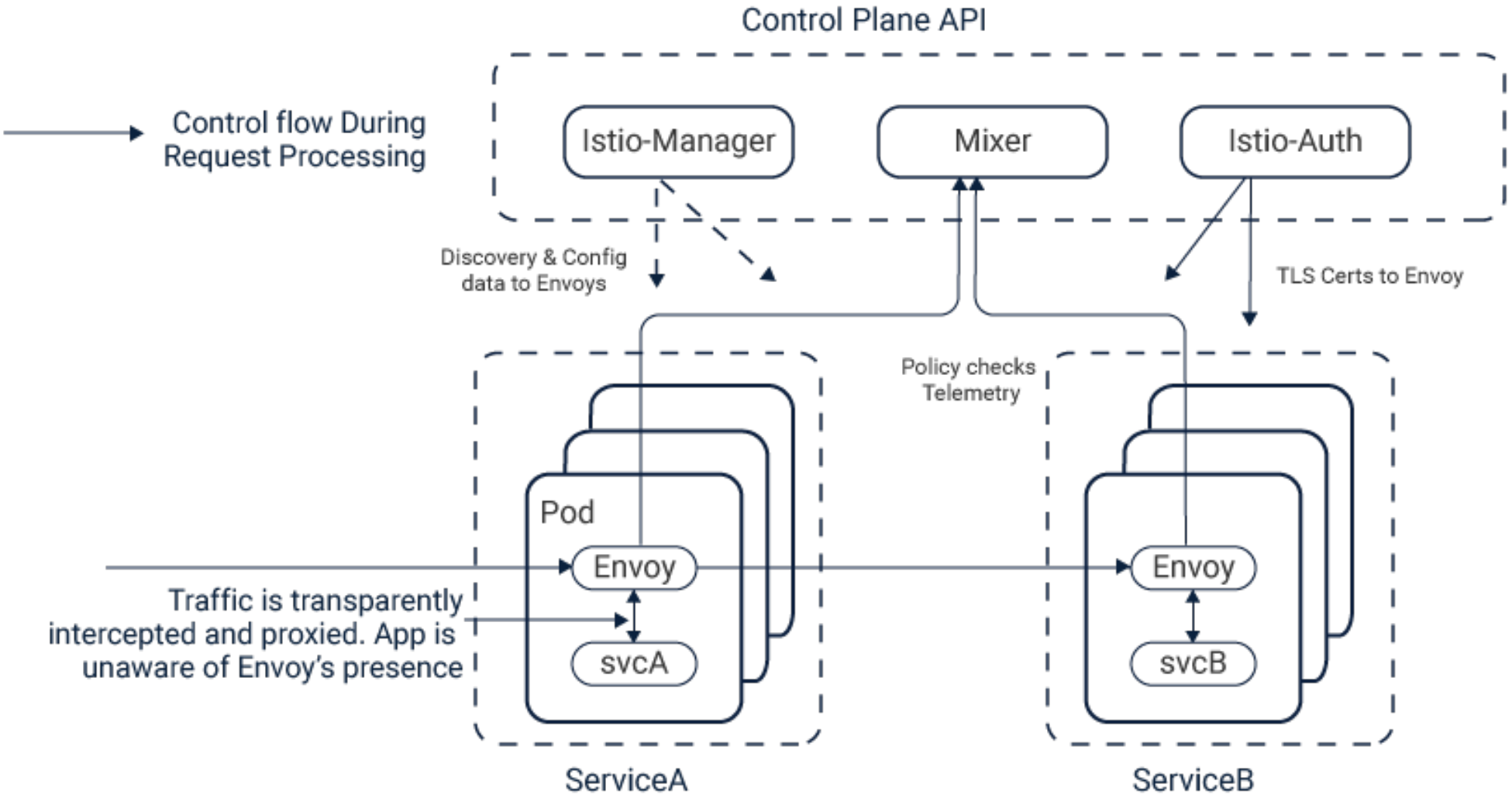
Cloud Native Architectures with Service Mesh- Microservices Evolution from 2014 to 2018



Cloud Native Architectures with Service Mesh - Netflix OSS Architecture



Cloud Native Architectures with Service Mesh - ISTIO Architecture



Cloud Native Architectures with Service Mesh - ISTIO

Traffic Management (Routing/loadbalancing)

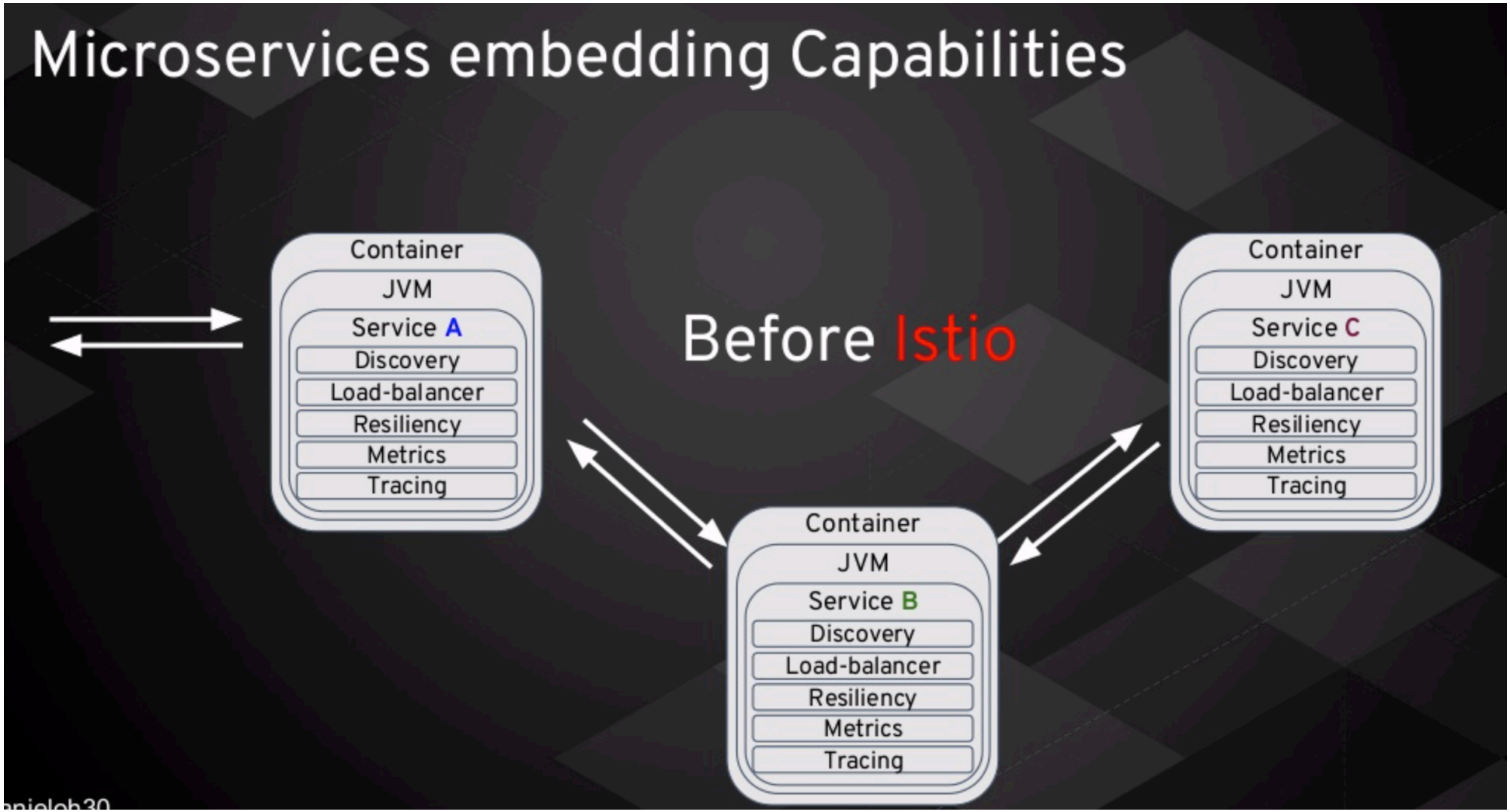
Policies and Security (Authorization/ Authentication for Service to Service communication)

Observability (Service Discovery and Circuit breaking features)

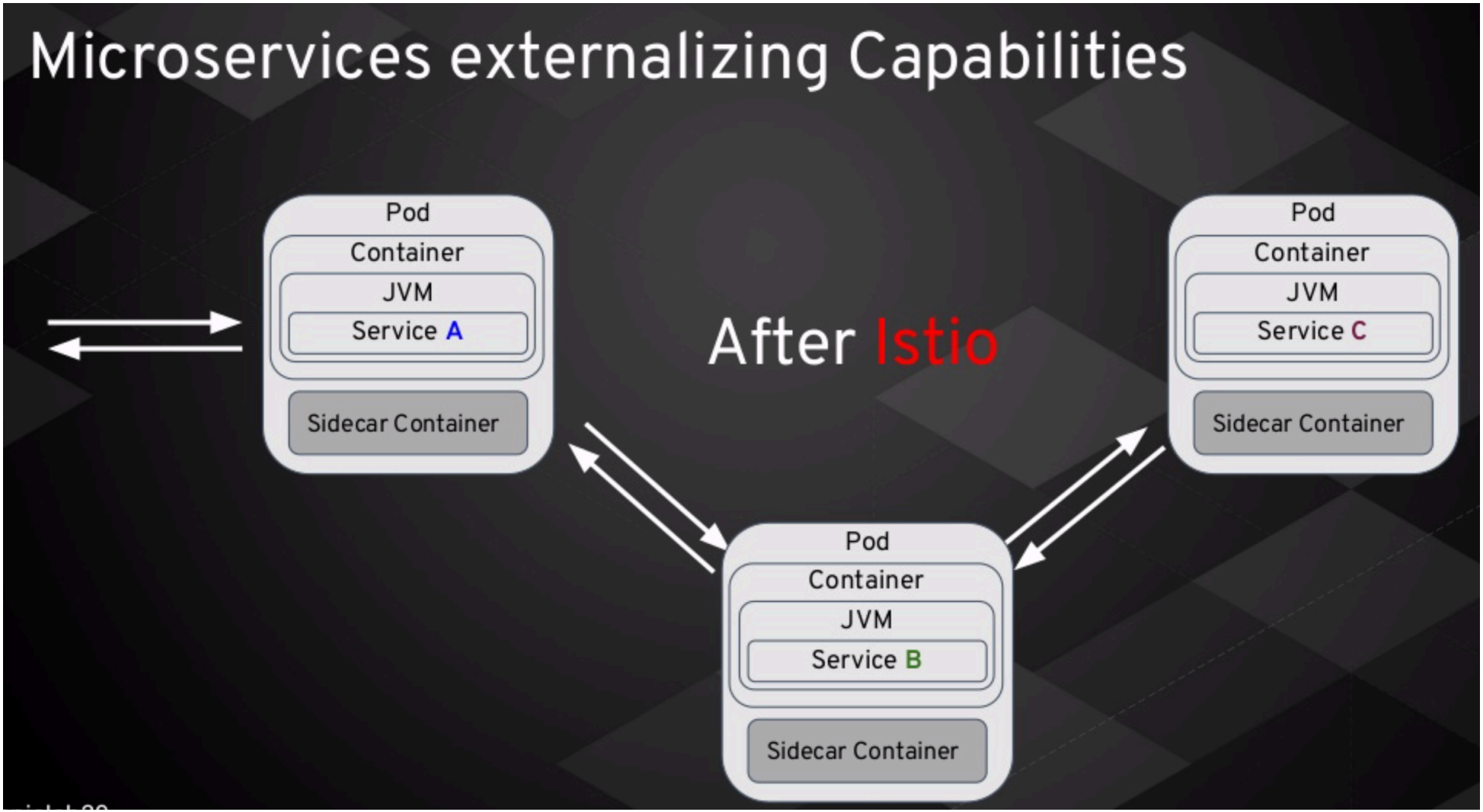
Performance and Scalability (monitoring and sending telemetry and performance details)

Deployment Models

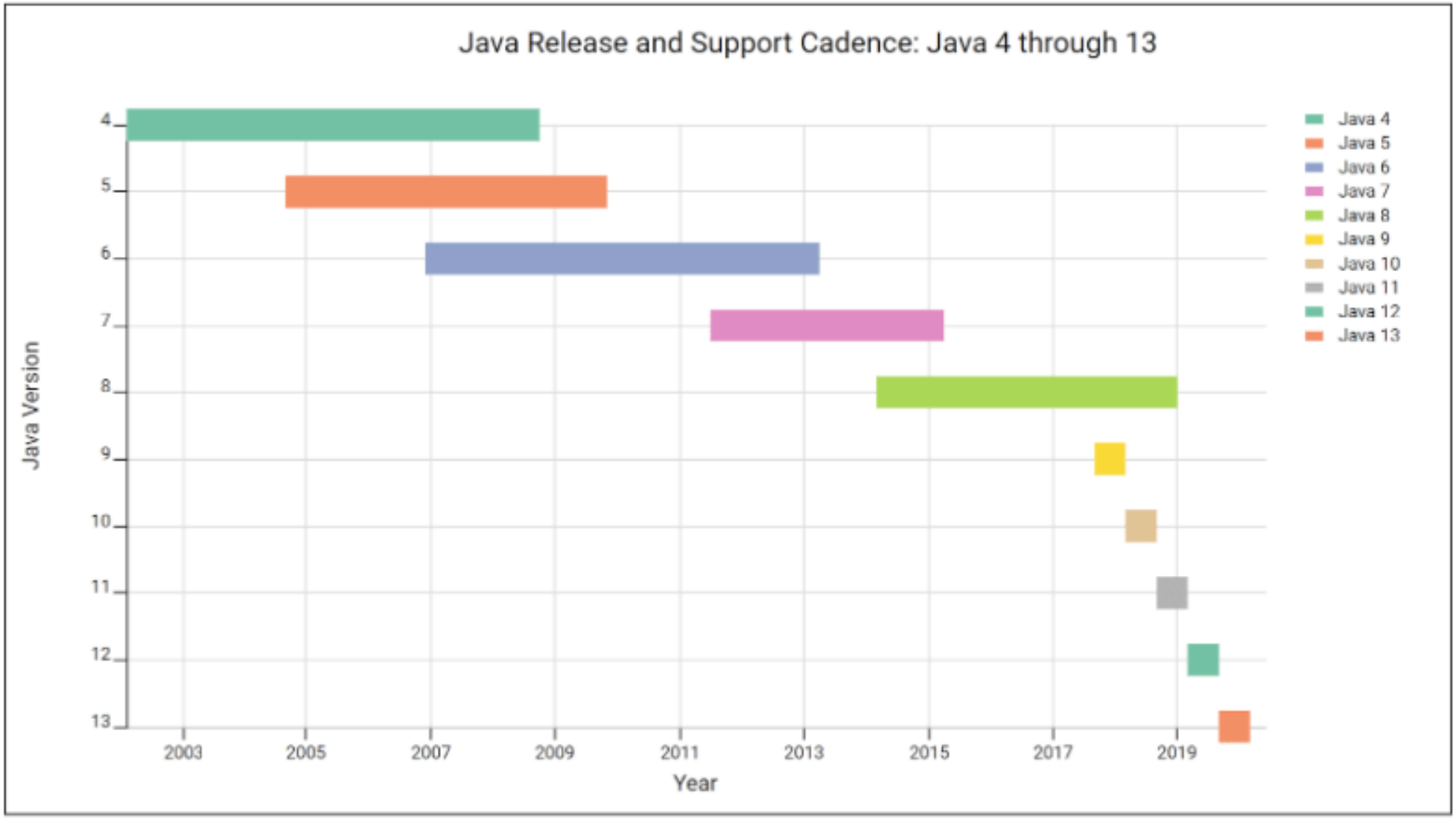
Cloud Native Architectures with Service Mesh - Before ISTIO



Cloud Native Architectures with Service Mesh - After ISTIO



JDK Features overview – Versions - 9,10,11,12,13



Contributing to Open source - Why Contribute ??

Opportunity to get recognized in global forums for any great work you do for community

Opportunity to get Involved in the Community

Improve you software skills and engineering processes from different oragnizations

Experience working with distributed teams spread across globe.

meet new people from different industries and different skill sets

Business Opportunity for your work.

Contributing to Open source – What to contribute ??

- Documentation
- Website fixes
- Code tests
- Code examples
- Coverage increase for unit, integration, performance and static tests
- New features
- Dependencies updates
- Workshops

Contributing to Open source – How to contribute ??

- Source Code Repositories – GitHub Forking work flow
- JIRA and Bugzilla
- Project Mailing lists
- Dev discussion Groups (Google groups/Yahoo groups)
- Instant Messaging/Chat (Gitter , Slack, etc)
- Video Communications (Google hangouts, Zoom, skype etc)


Questions