# Eclipse MicroProfile Metrics: Practical use cases

Jorge Cajas - José Diaz - Víctor Orozco

October 20, 2018

GuateJUG - PeruJUG

# Overview

Why Metrics?

Monoliths

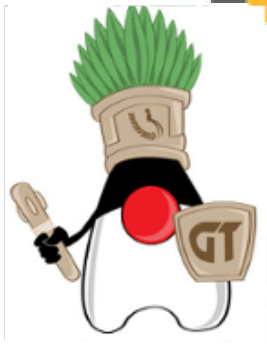
Reactive applications

Eclipse MicroProfile

Practical Use Cases

# Víctor Orozco

- I like Java EE
- CTO@Nabenik
- @tuxtor
- http://vorozco.com
- http://tuxtor.shekalug.org

Oracle Groundbreakers


ORACLE Certified Professional Java SE 8 Programmer

ORACLE Certified Associate Java SE 8 Programmer

NABENIK

# Jorge Cajas

- JUG Leader





NABENIK

# José Díaz

- JUG Leader



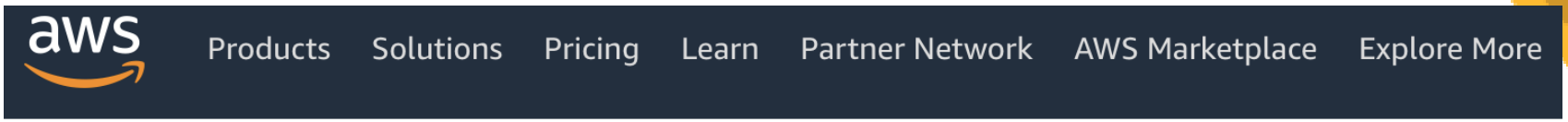



NABENIK

# Why Metrics?

Do I need metrics?

# Why Metrics?




## DevOps Practices

The following are DevOps best practices:

- Continuous Integration
- Continuous Delivery
- Microservices
- Infrastructure as Code
- Monitoring and Logging
- Communication and Collaboration

Below you can learn more about each particular practice.

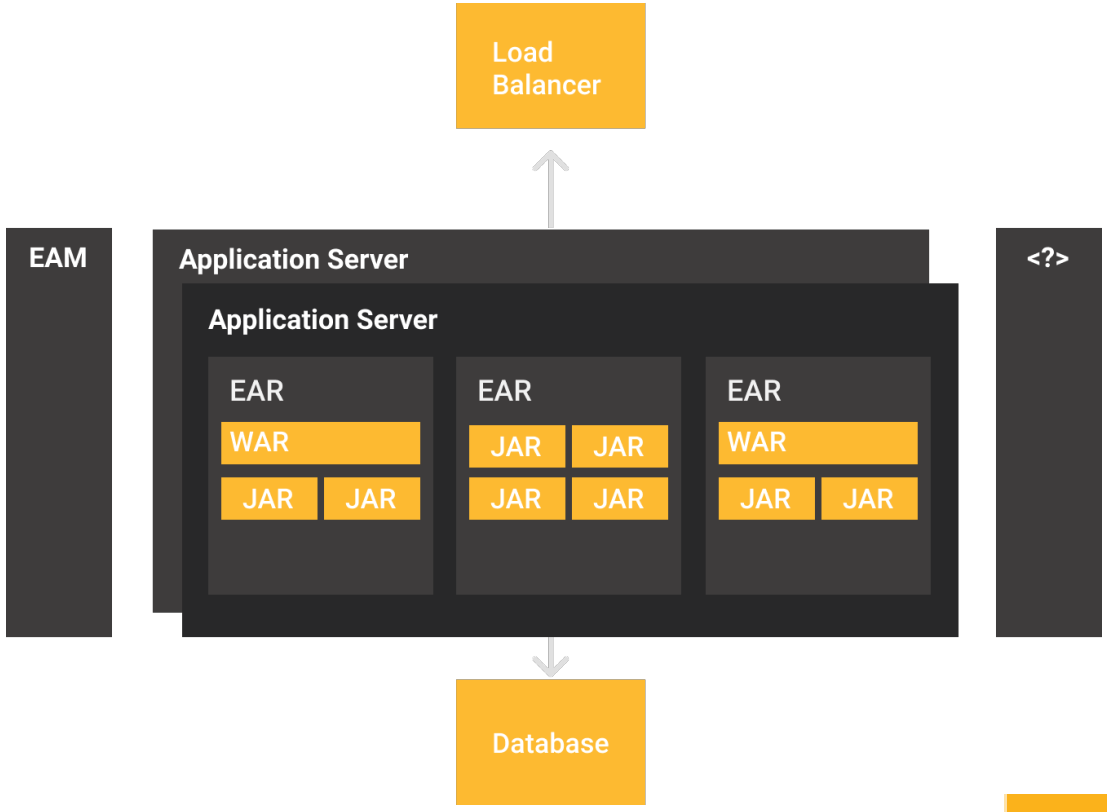
NABENIK

What about non-DevOps?

# Monoliths

# Metrics in Java Monoliths

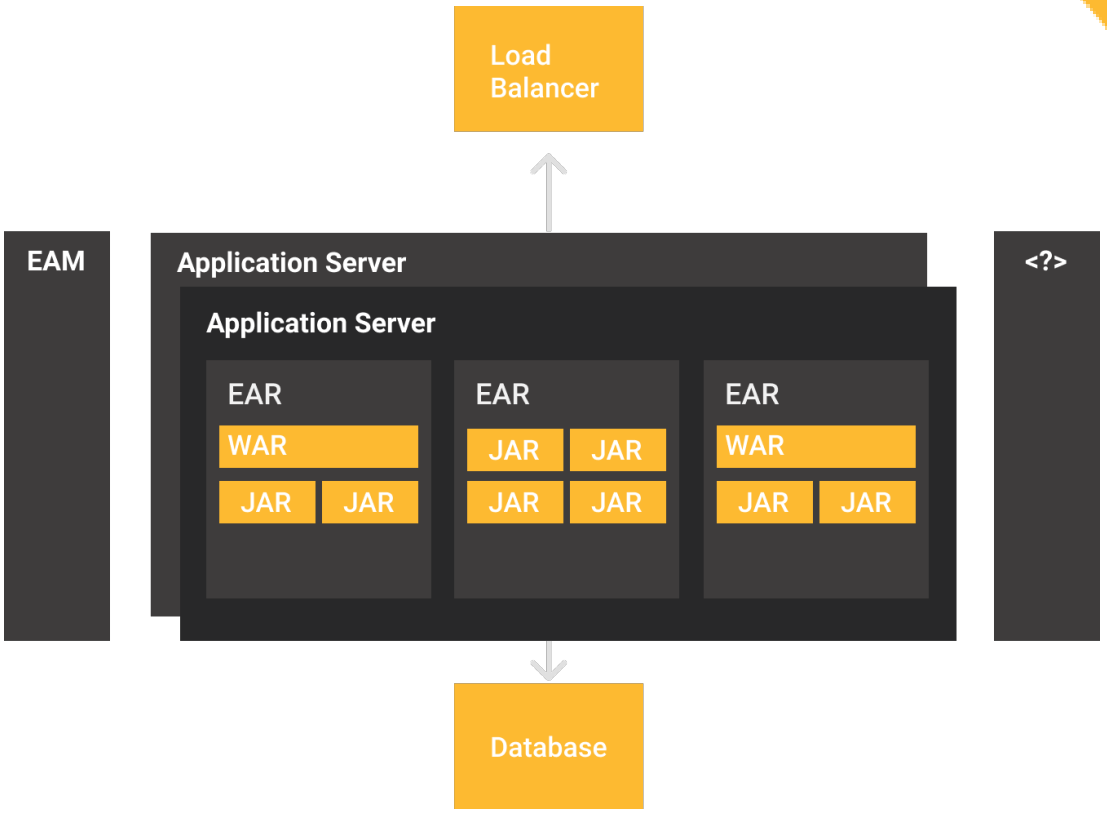
- Long running JVM
- Scale as . . . more long running JVMs
- Ideally never rebooted




NABENIK

# Metrics in Java Monoliths

1. Telemetry Vendor APIs (Glassfish Metrics)
2. JMX (VisualVM, Mission Control)
3. Shell wranglers + Logs




NABENIK

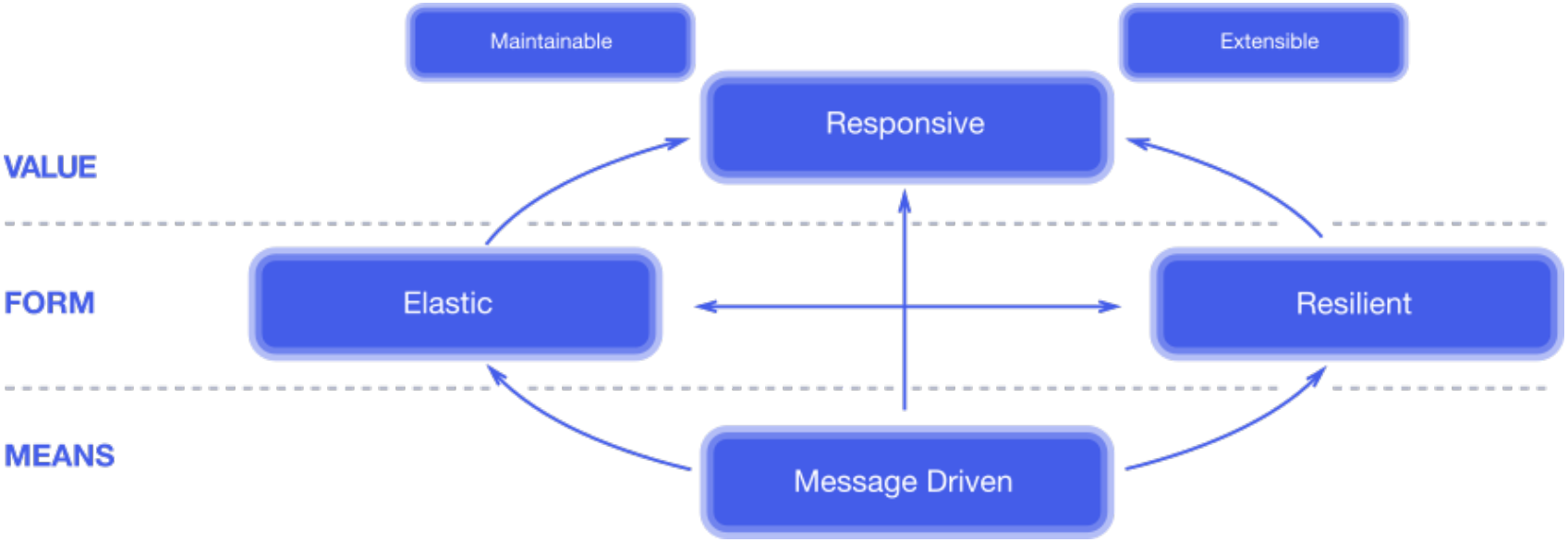
How do I choose between JMX or a telemetry API?

How do I get access to JMX if I´m using PaaS?

# Reactive applications

# Reactive applications

Reactive often means Microservices

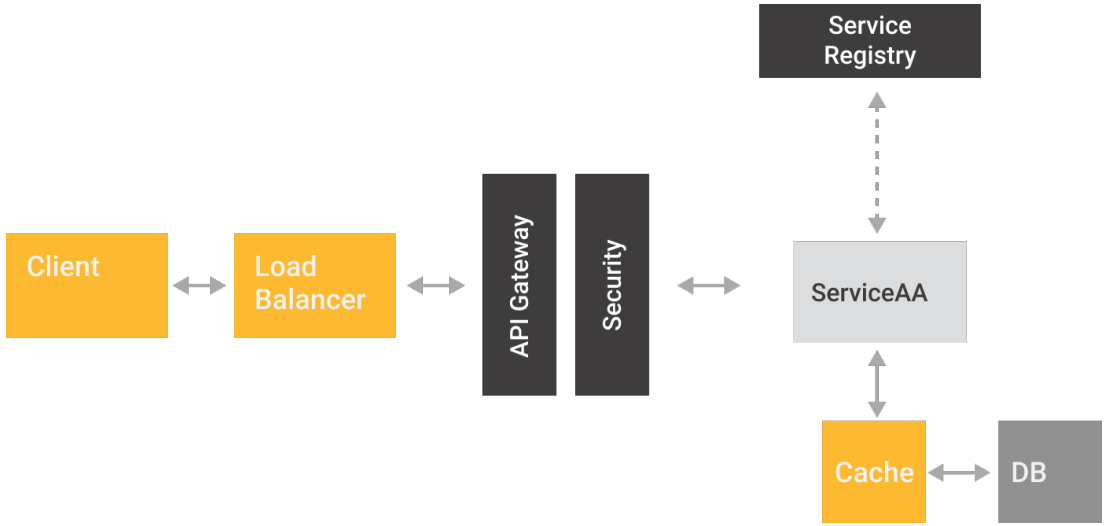



Key concept: Non-long running JVM

NABENIK

# Metrics in Microservices

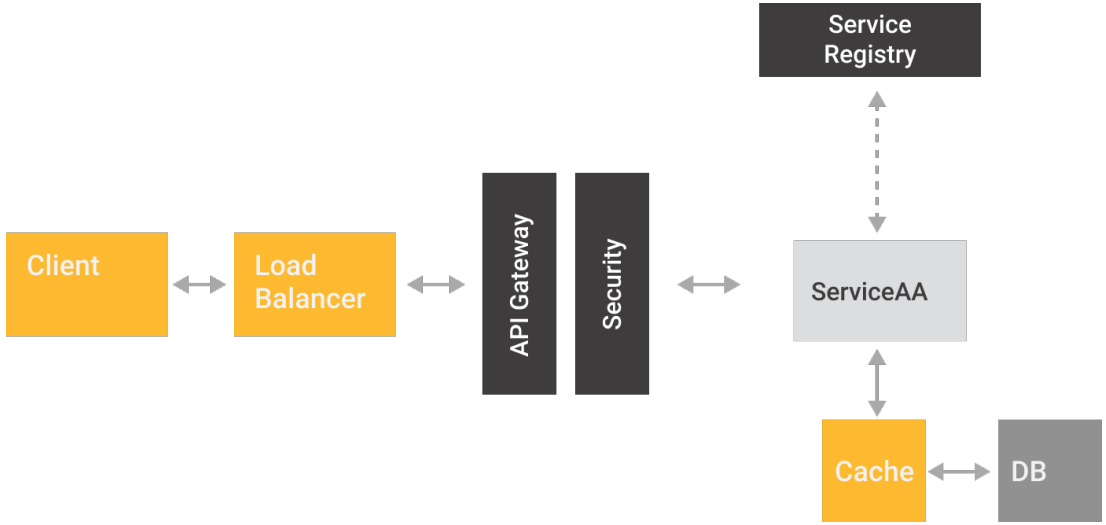
- Short lived JVMs
- Orchestrated through Swarm/Kubernetes
- Provisioned as needed




NABENIK

# Metrics in Microservices

- JVM over CaaS over PaaS
- Dynamic and ever-changing addresses and ports
- Logs inside the container




NABENIK

# Eclipse MicroProfile

# Eclipse MicroProfile

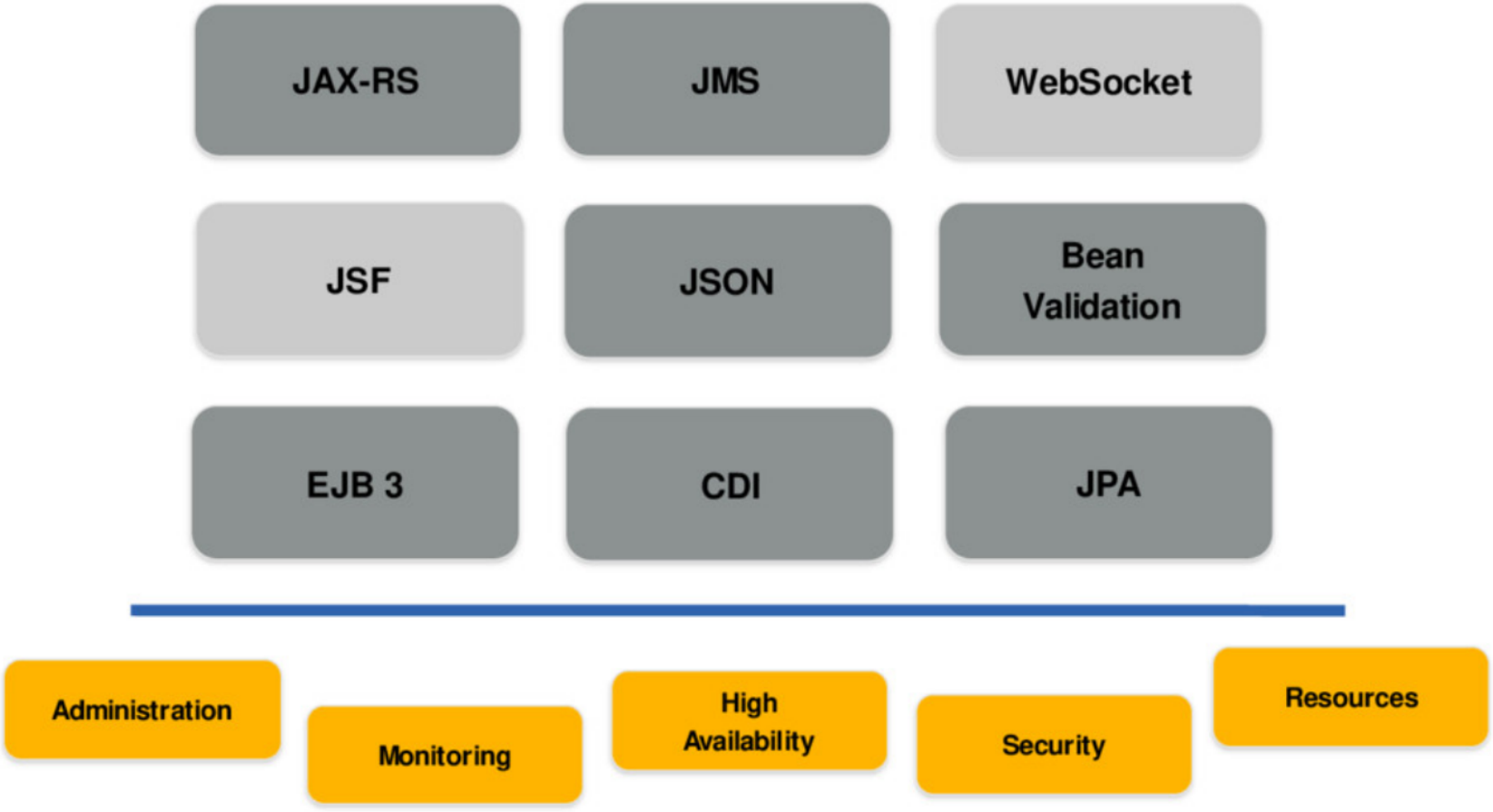



Figure 1: Credits: Reza Rahman

NABENIK

# Eclipse MicroProfile

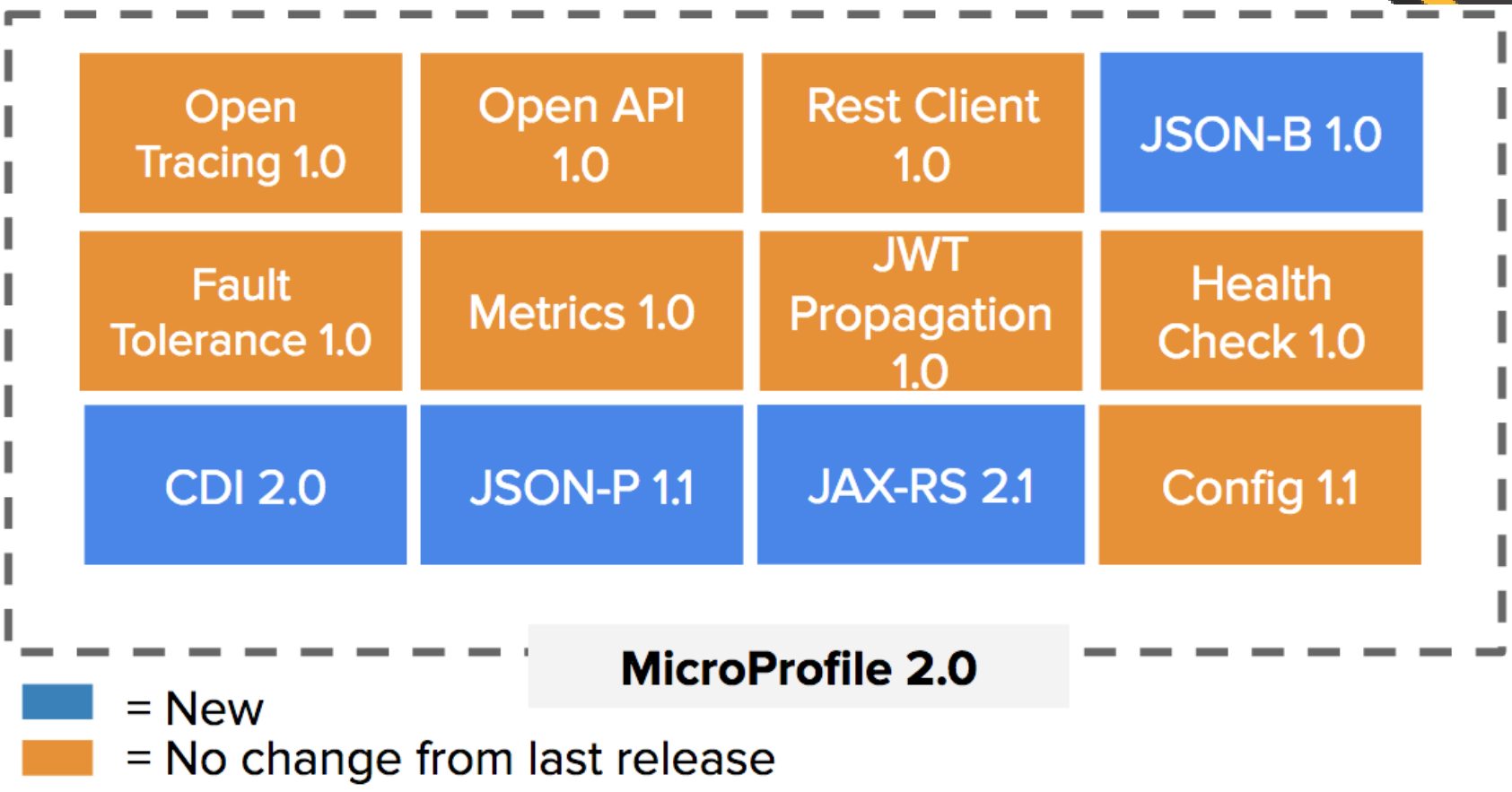



Figure 2: Credits: Reza Rahman

NABENIK

# Eclipse MicroProfile on Payara 5

```
<dependency>
        <groupId>org.eclipse.microprofile</groupId>
        <artifactId>microprofile</artifactId>
        <type>pom</type>
        <version>2.0.1</version>
        <scope>provided</scope>
</dependency>
```

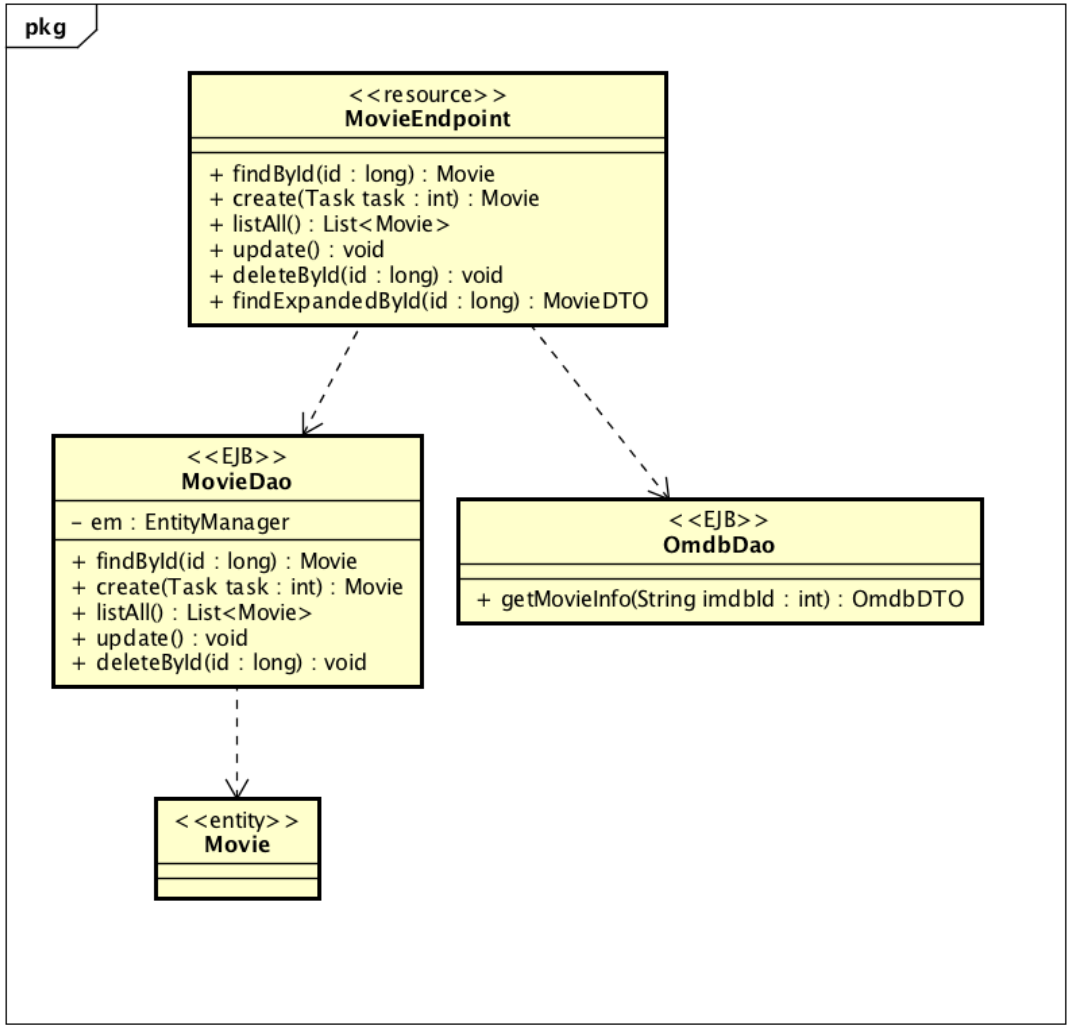
# Java 8, JAX-RS, CDI, EJB, Microprofile

`https://github.com/tuxtor/payara-demo`

`https://github.com/tuxtor/omdb-demo`

Take for granted

- EJB
- **JTA**
- JAX-RS
- CDI

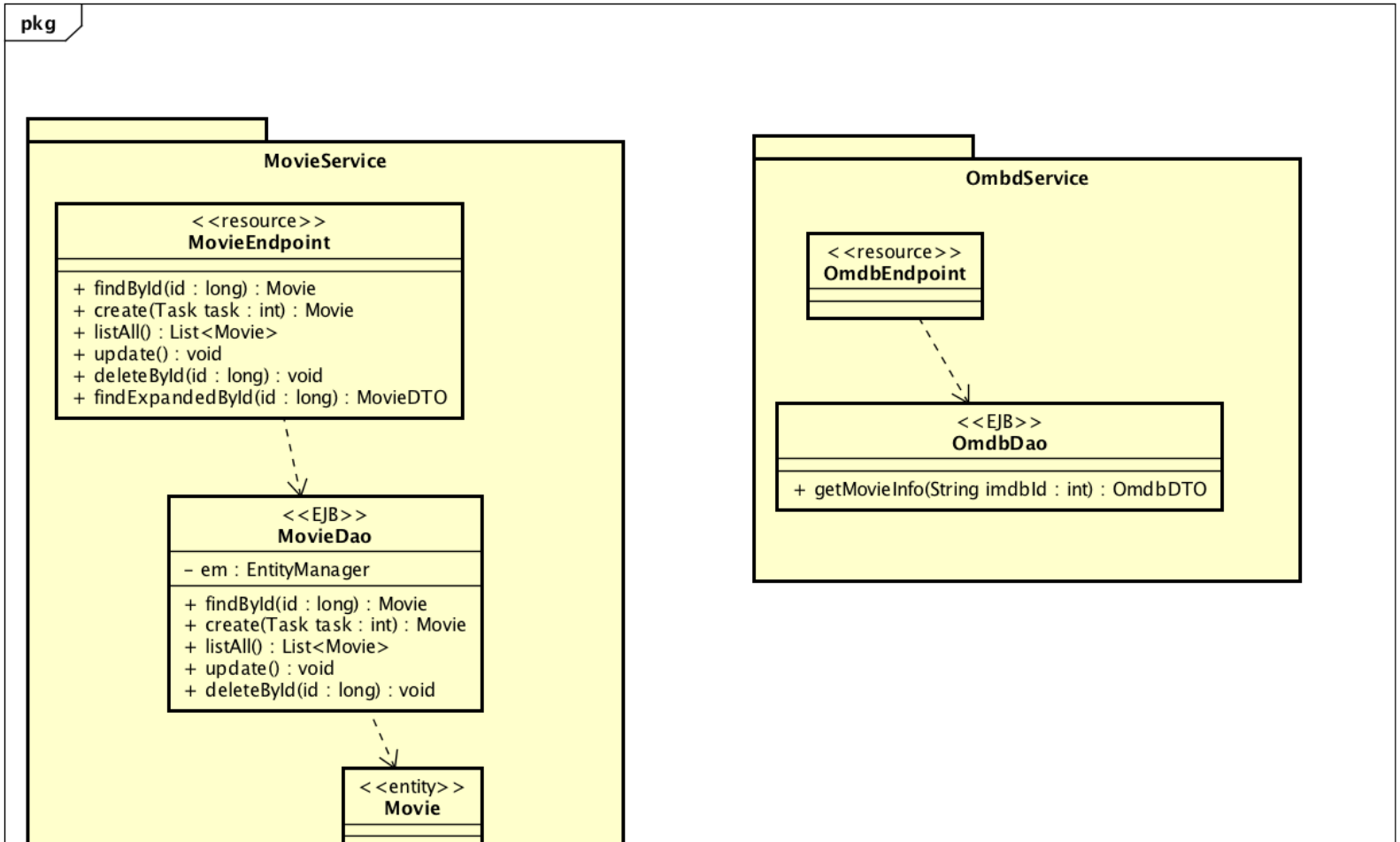


# EE + MicroProfil - Demo

MicroProfile: JAX-RS, CDI (Per service), Config, Fault Tolerance, Metrics

Payara Micro: EJB, JTA (Per service)

External: Location, Deployment, Orchestation, Balancing, Consistency, Patterns




BENIK

# Config

```
@Inject
@ConfigProperty(name = "omdbservice.url")
String omdbDaemonServiceUrl;
```

# Fault tolerance

- Circuit Breaker
- Bulkhead
- Fallback
- Retry
- Timeout

# Fault tolerance - Fallback, Timeout

```
@GET
@Path("/{id:[a-z]*[0-9][0-9]*}")
@Fallback(fallbackMethod = "findByIdFallBack")
@Timeout(TIMEOUT)
public Response findById(@PathParam("id")
final String imdbId) {
...
}

public Response findByIdFallBack(@PathParam("id")
final String imdbId) {
...
}
```

# Metrics

Where

- JSON or OpenMetrics (Prometheus)
- Vendor
- Base
- Application

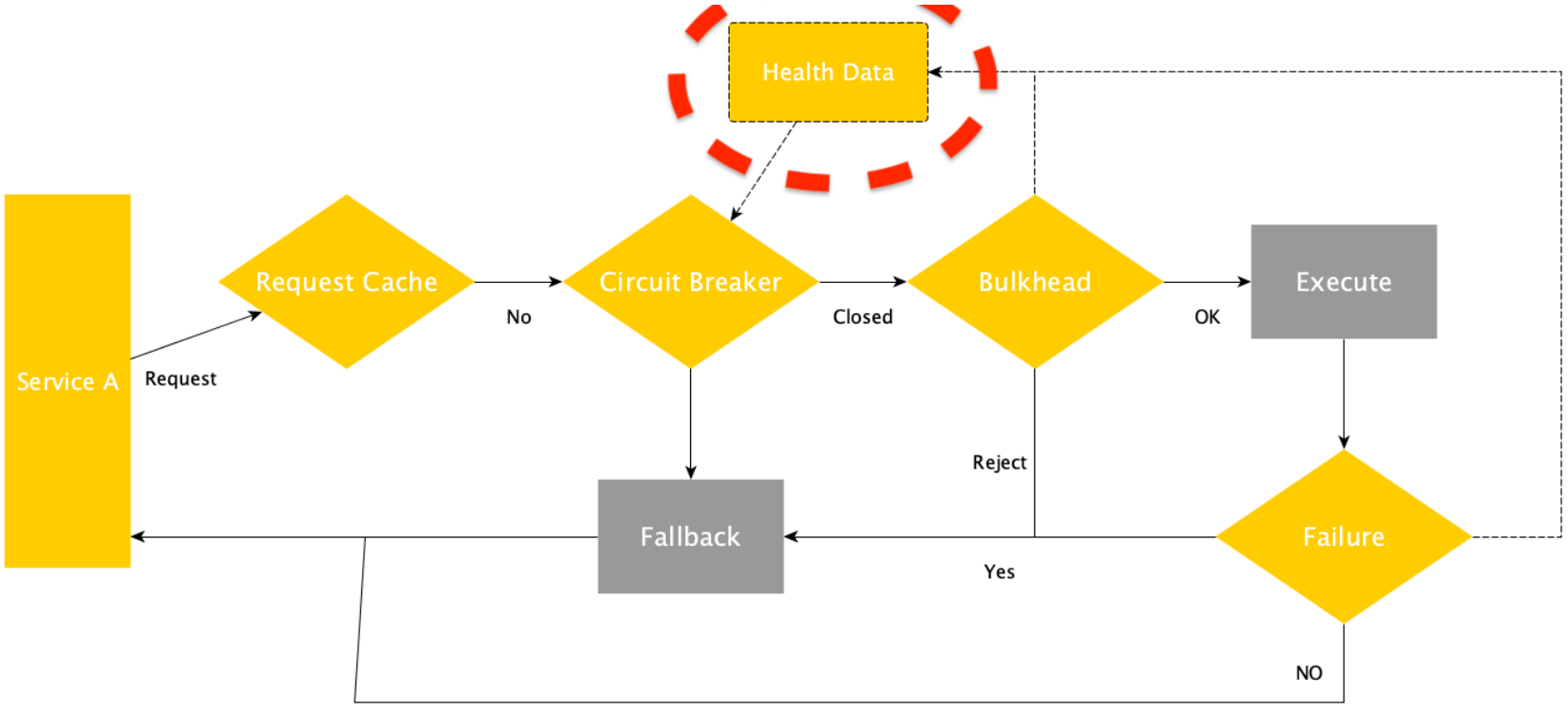
How

- Counter
- Gauge
- Timed
- Histogram

# Practical Use Cases

# Case 0 - Metrics for microservices

1. Metrics are being generated anyway
2. Base for improvements, diagnosis if exposed properly




NABENIK

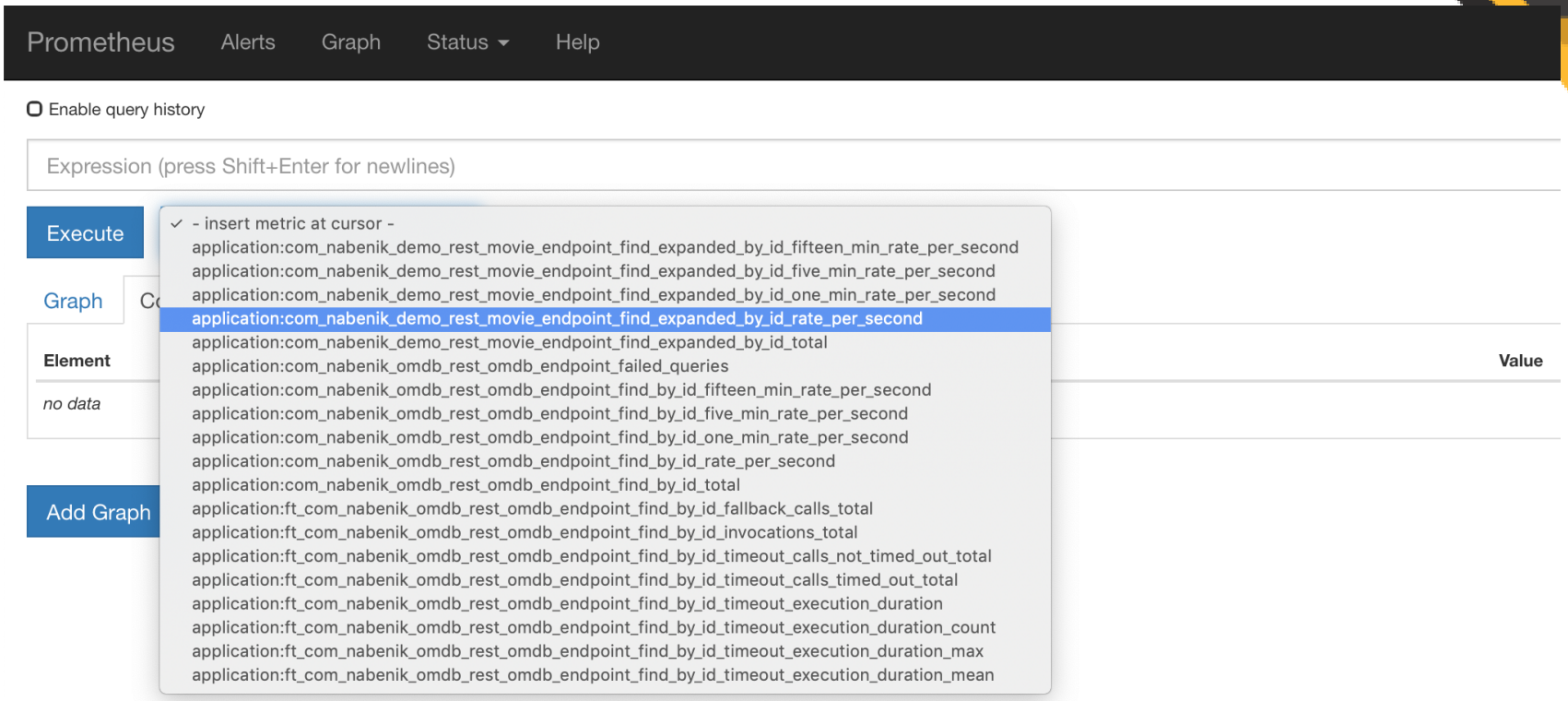
# Case 0 - Counter

```
@Inject
@Metric
Counter failedQueries;

@Metered
@GET
@Path("/{id:[a-z]*[0-9][0-9]*}")
@Fallback(fallbackMethod = "findByIdFallBack")
@Timeout(TIMEOUT)
public Response findById(@PathParam("id")
final String imdbId) {
...
}

public Response findByIdFallBack(@PathParam("id")
final String imdbId) {
...
}
```

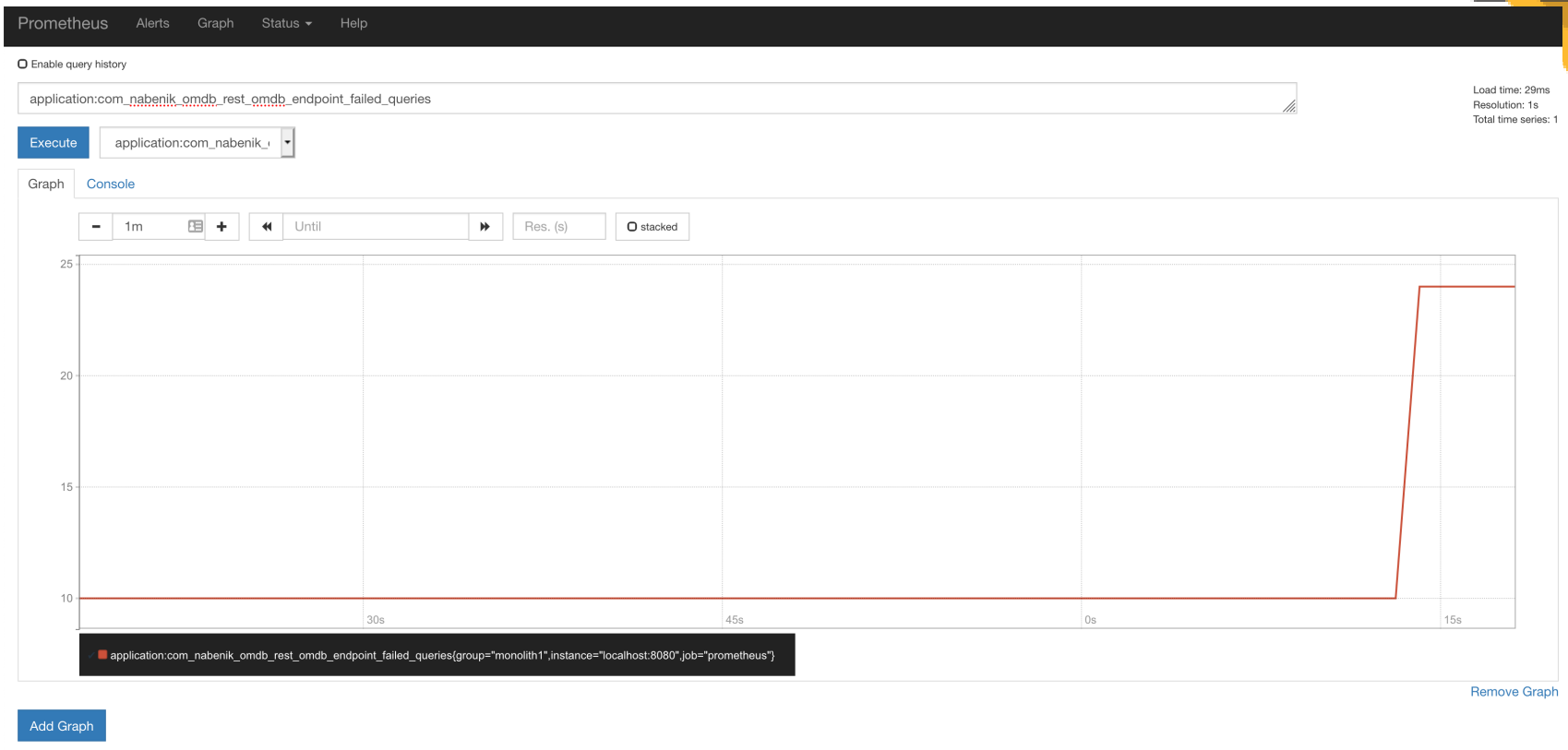
# Case 0 - Prometheus

```
@Inject
@Metric
Counter failedQueries;
```

# Case 1 - Application metrics



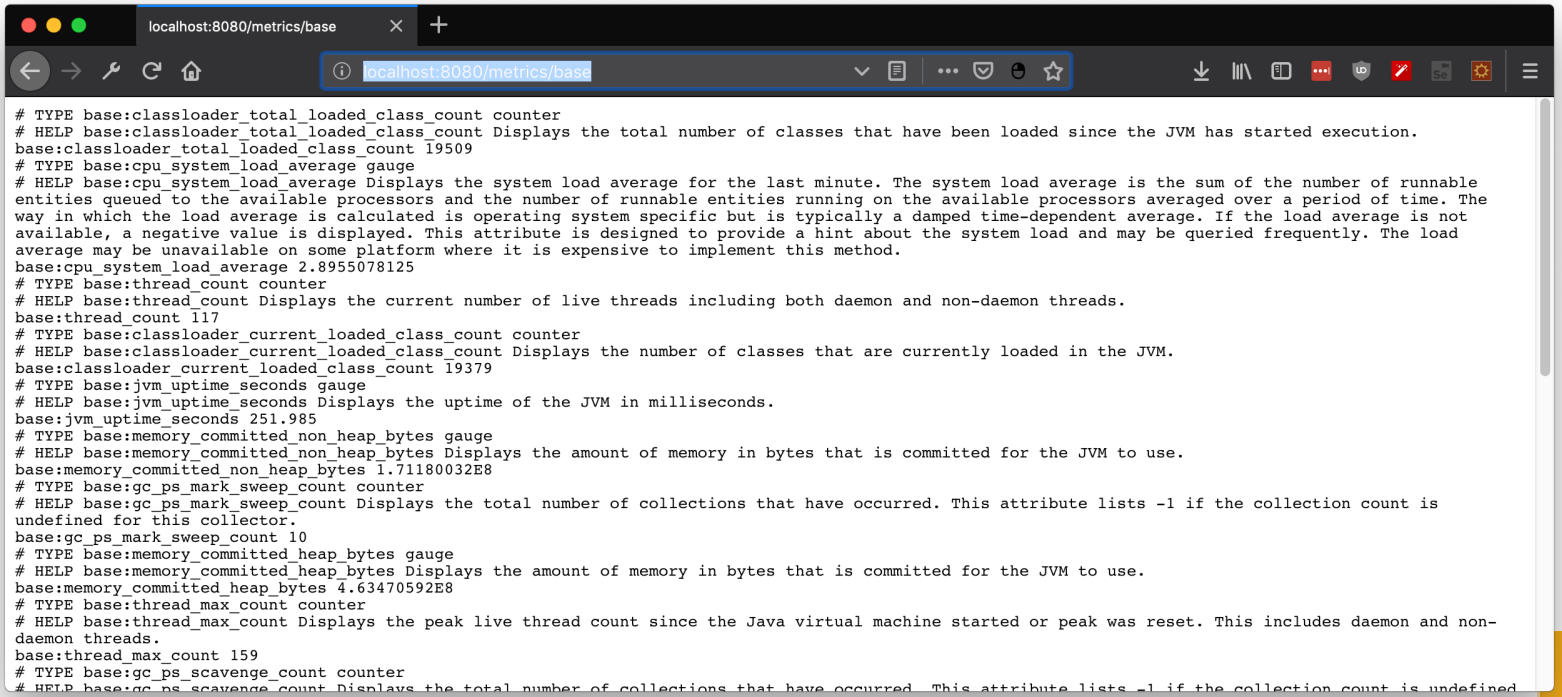
# Case 1 - Counter over time




```
application:com_nabenik_omdb_rest_omdb_endpoint_failed_queries
```

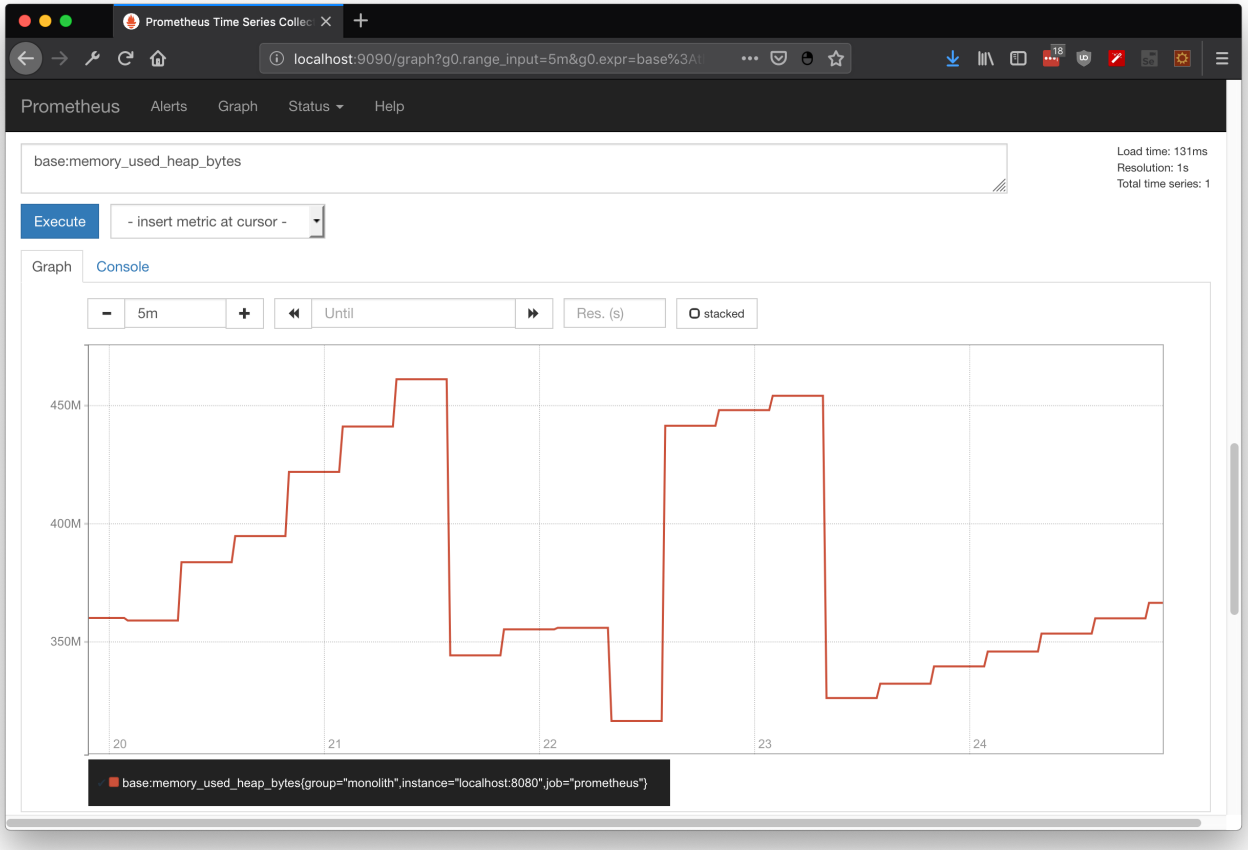
# Case 1 - Telemetry for monoliths

1. Base metrics are almost JVM metrics
2. JMX is also a pull-based monitoring technology
3. Ideal for PaaS




NABENIK

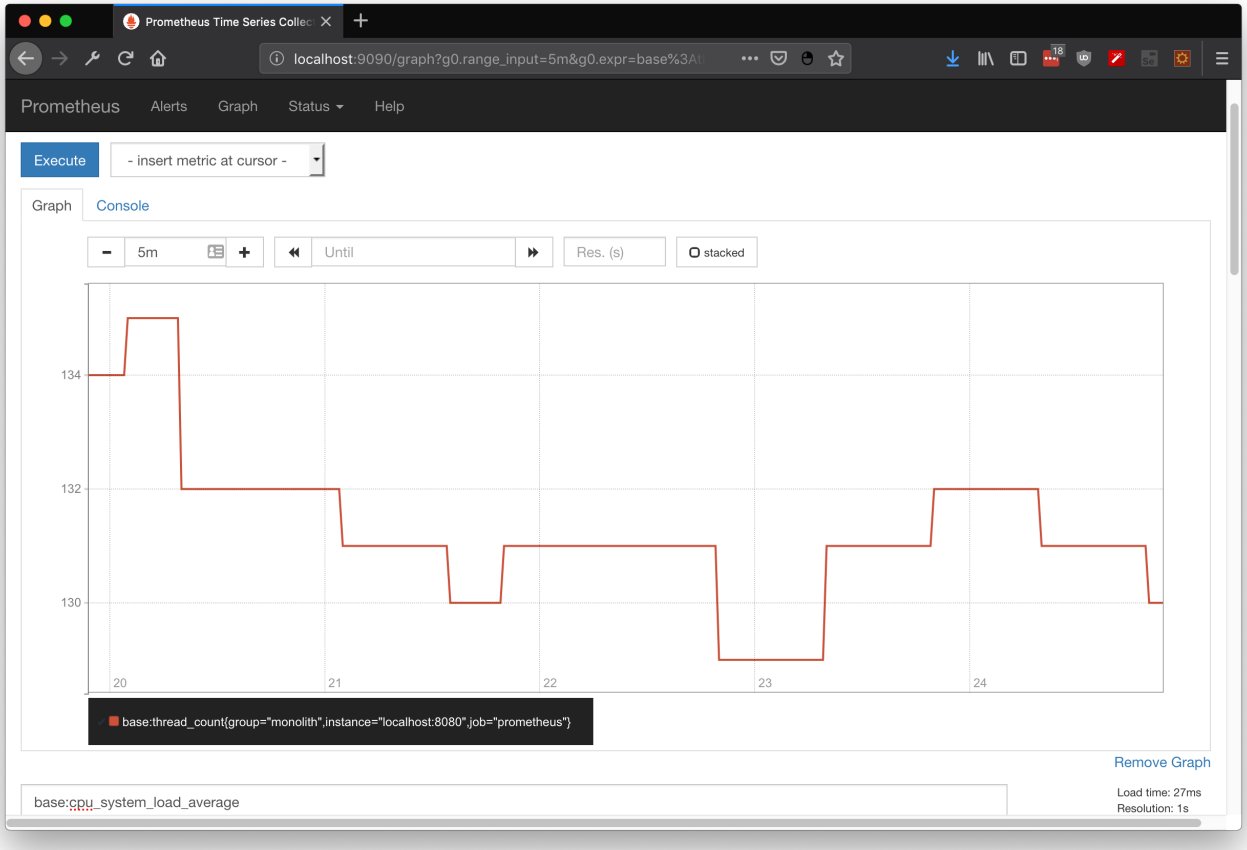
# Case 1 - Heap performance




`base:memory_used_heap_bytes`

NABENIK

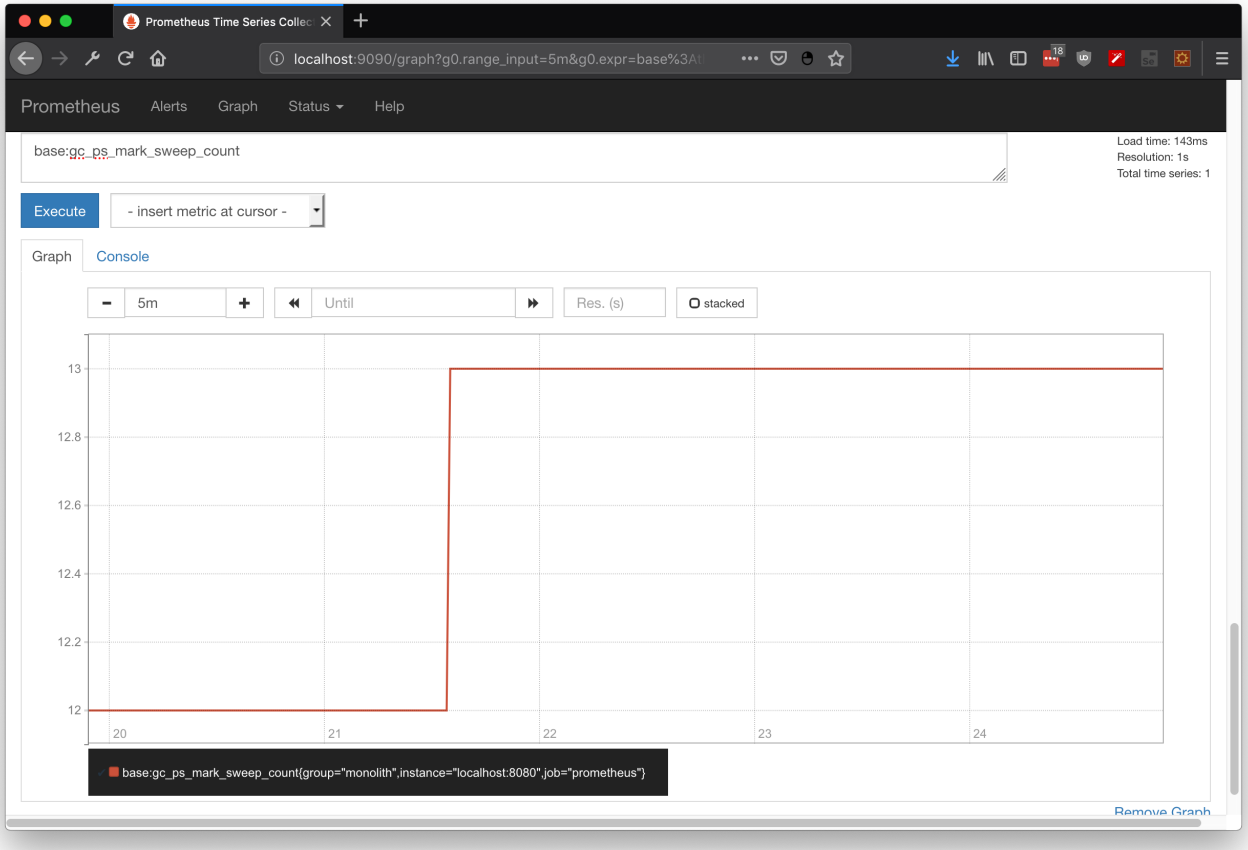
# Case 1 - CPU Utilization




`base:cpu_system_load_average`

NABENIK

# Case 1 - GC Executions




`base:gc_ps_mark_sweep_count`

NABENIK

# Thank you

- me@vorozco.com
- http://vorozco.com
- http://github.com/tuxtor/slides



NABENIK