Eclipse MicroProfile Metrics: Practical use cases

Jorge Cajas - José Diaz - Víctor Orozco

October 22, 2018

GuateJUG - PeruJUG

NABENIK

Overview

Why Metrics?

Monoliths

Reactive applications

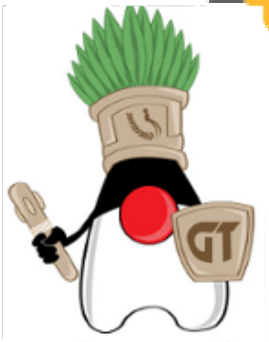
Eclipse MicroProfile

Practical Use Cases

NABENIK

Víctor Orozco

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



NABENIK

Jorge Cajas

- JUG Leader
- @cajasmota
- CTO@inin.global
- I write Java EE code everyday

NABENIK

José Díaz

- Peru JUG Leader
- Java Champion
- @jamdiazdiaz
- http://blog.joedayz.pe

NABENIK

Why Metrics?

NABENIK

Why Metrics?

Do I need metrics?

NABENIK

Why Metrics?

aws Products Solutions Pricing Learn Partner Network AWS Marketplace Explore More

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

Why Metrics?

What about non-DevOps?

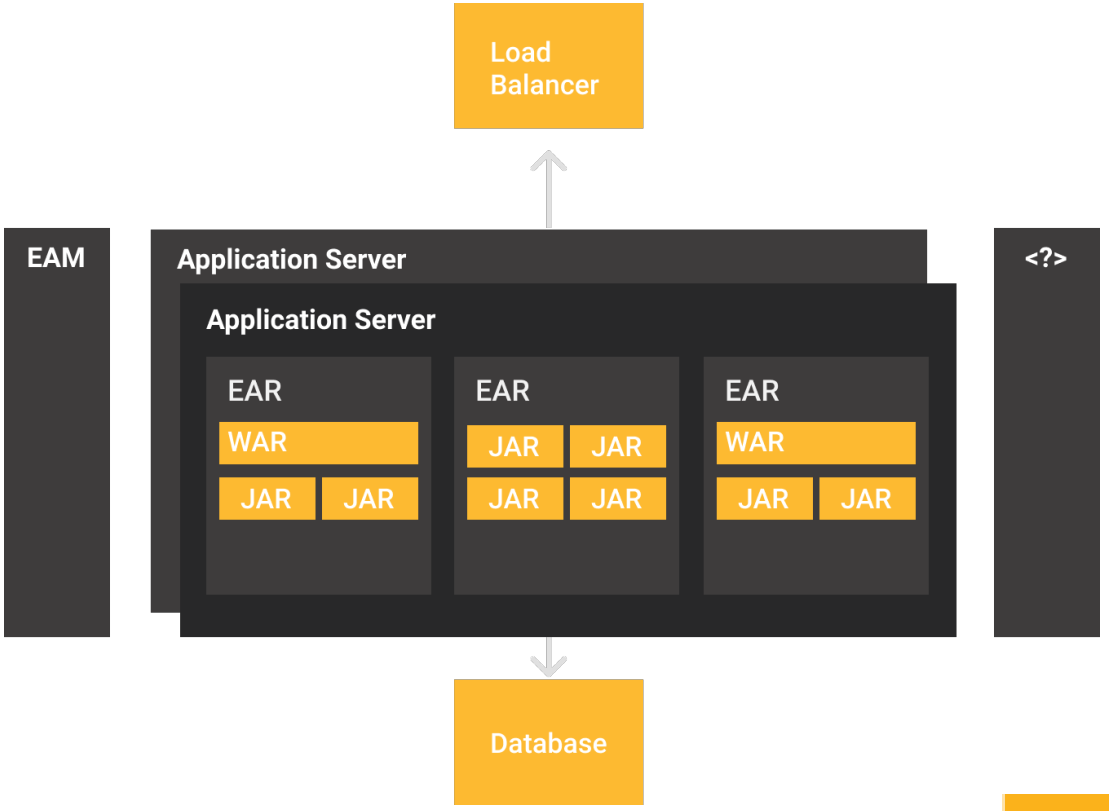
NABENIK

Monoliths

NABENIK

Metrics in Java Monoliths

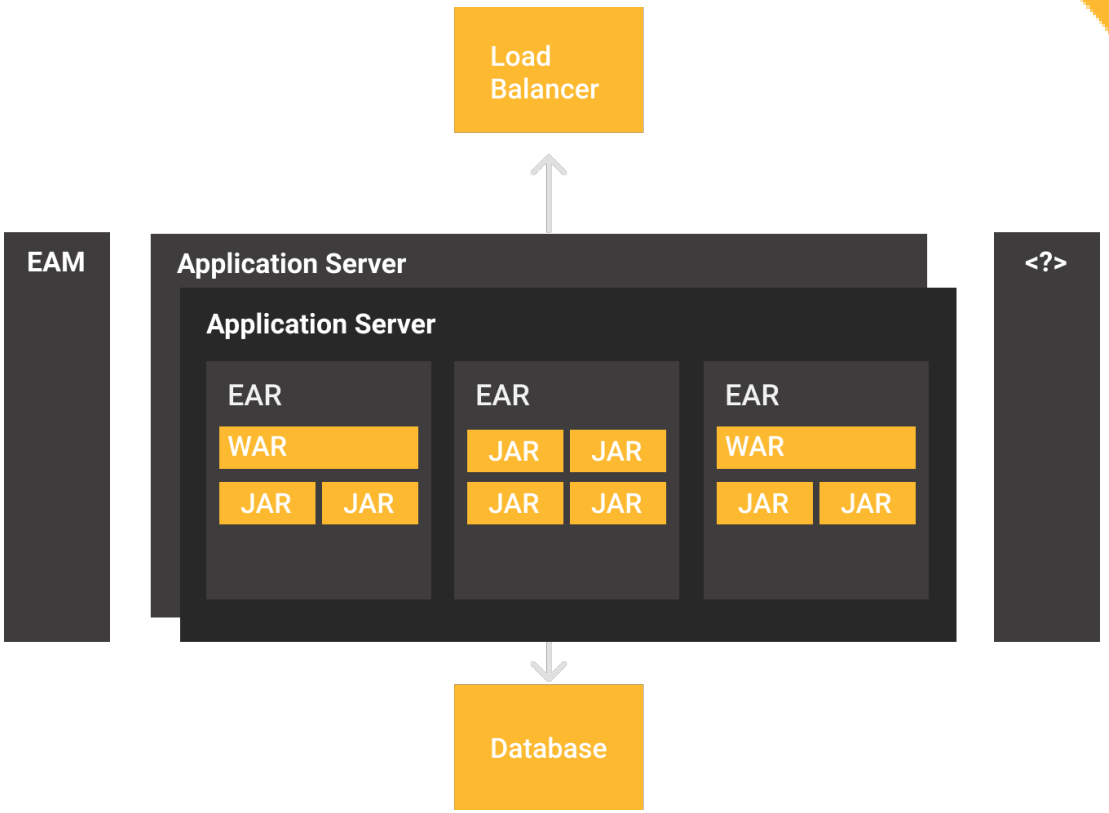
- Long running JVM
- Scale as . . . more long running JVMs
- Ideally never rebooted



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?

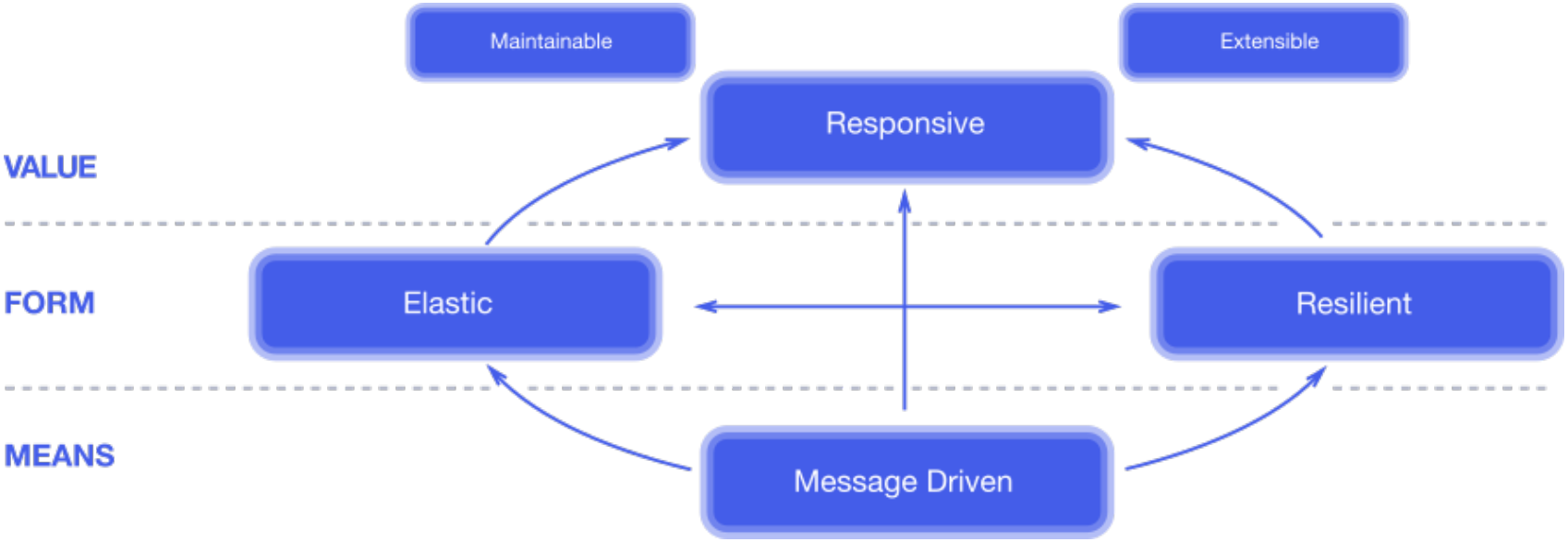
NABENIK

Reactive applications

NABENIK

Reactive applications

Reactive often means Microservices

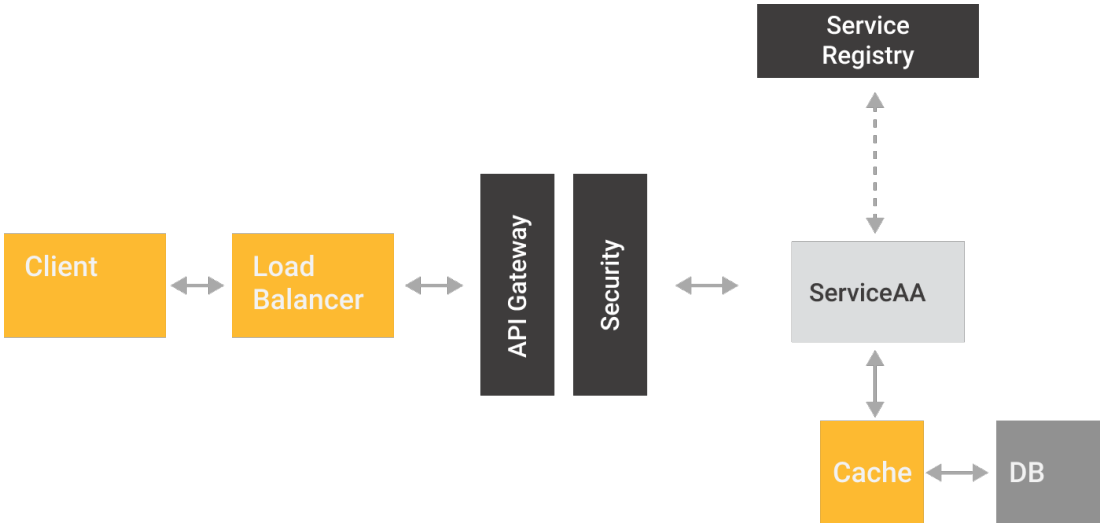


Key concept: Non-long running JVM

NABENIK

Metrics in Microservices

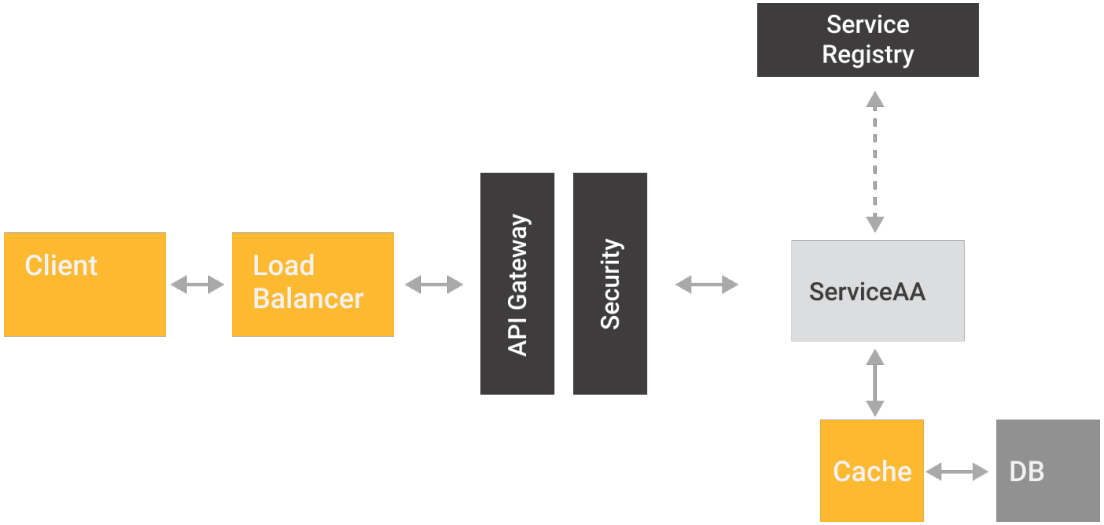
- Short lived JVMs
- Orchestrated through Swarm/Kubernetes
- Provisioned as needed



NABENIK

Metrics in Microservices

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



NABENIK

Eclipse MicroProfile

NABENIK

Eclipse MicroProfile

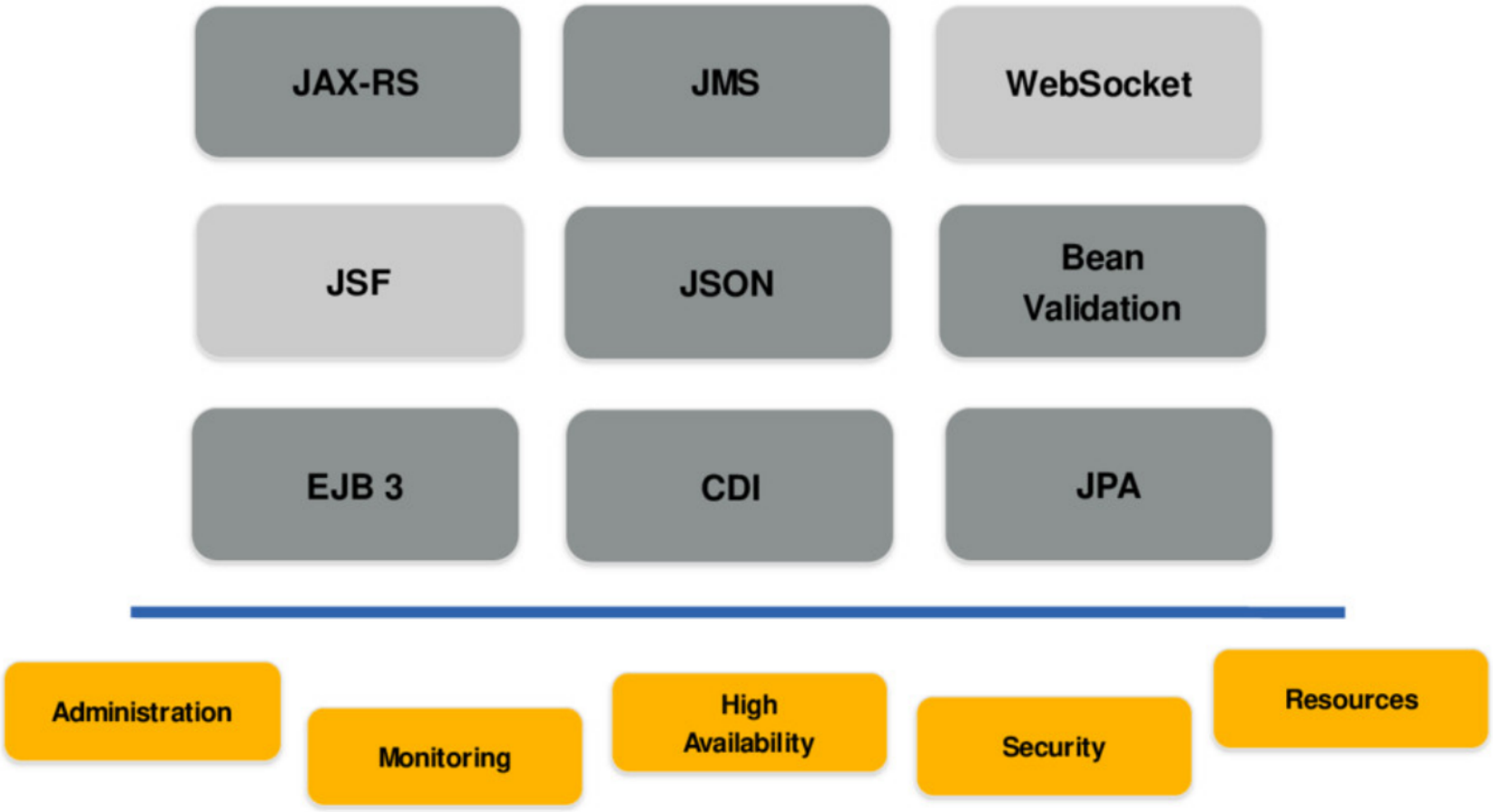
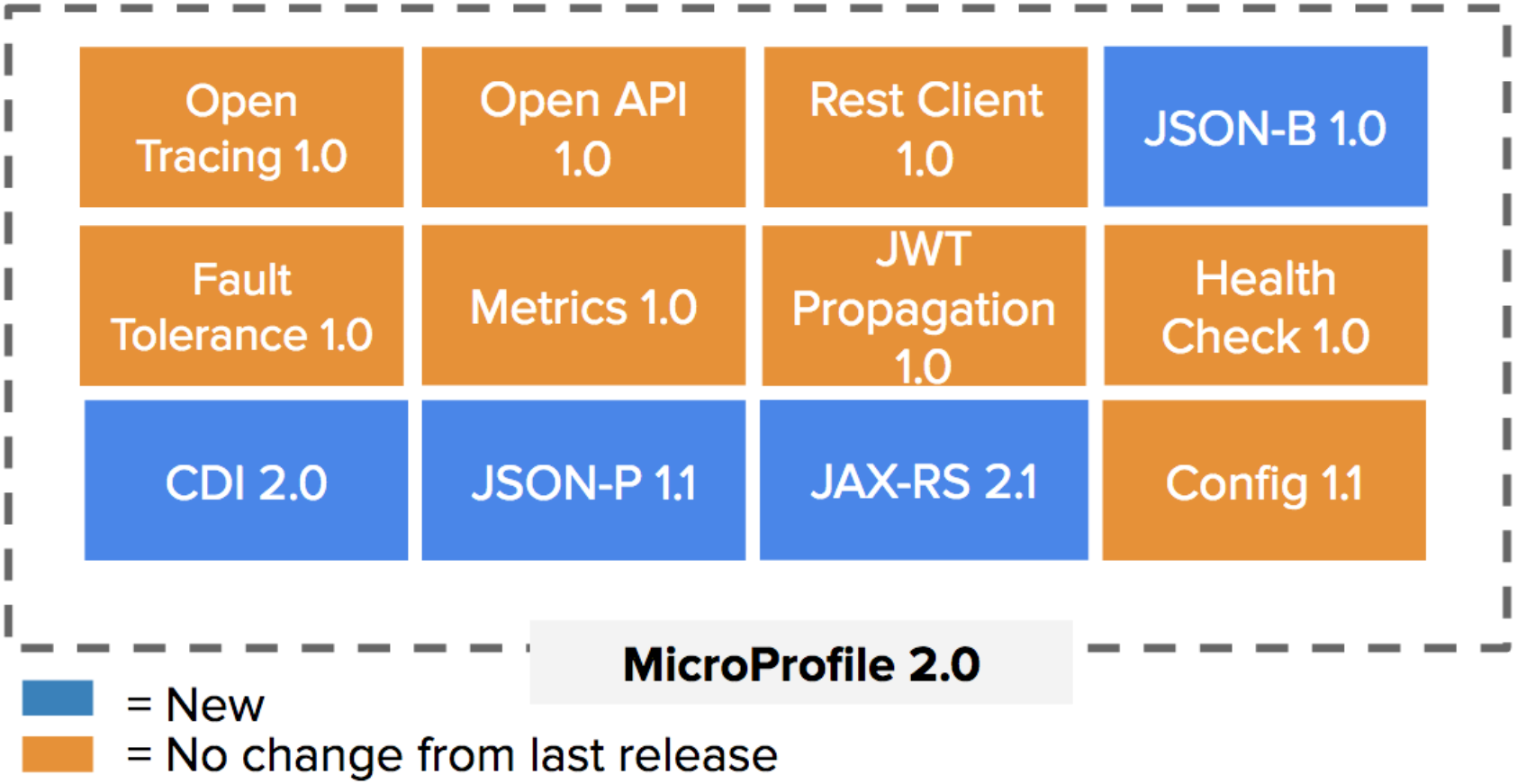


Figure 1: Credits: Reza Rahman

Eclipse MicroProfile



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>
```

NABENIK

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

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

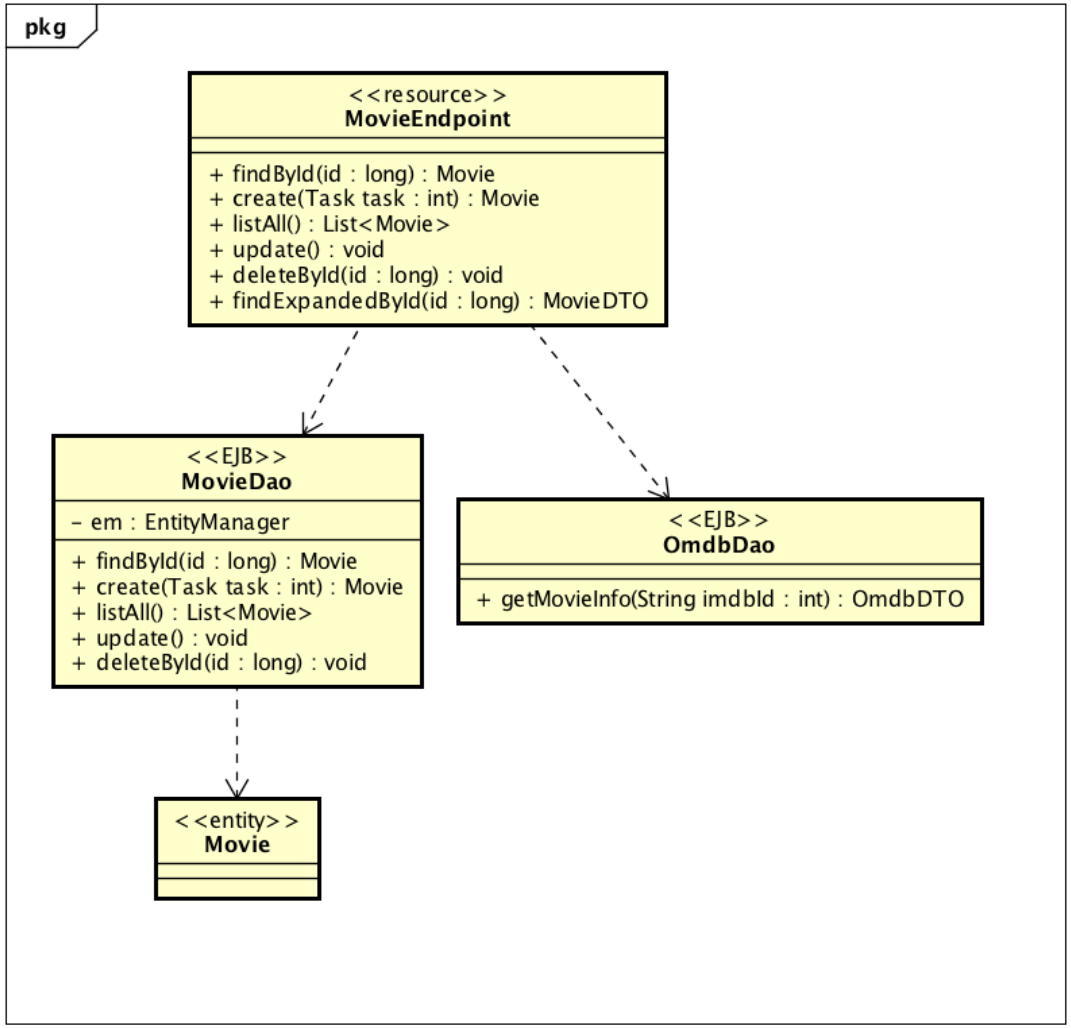
https://github.com/tuxtor/omdb-demo

NABENIK

Payara Micro - Traditional Java EE

Take for granted

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

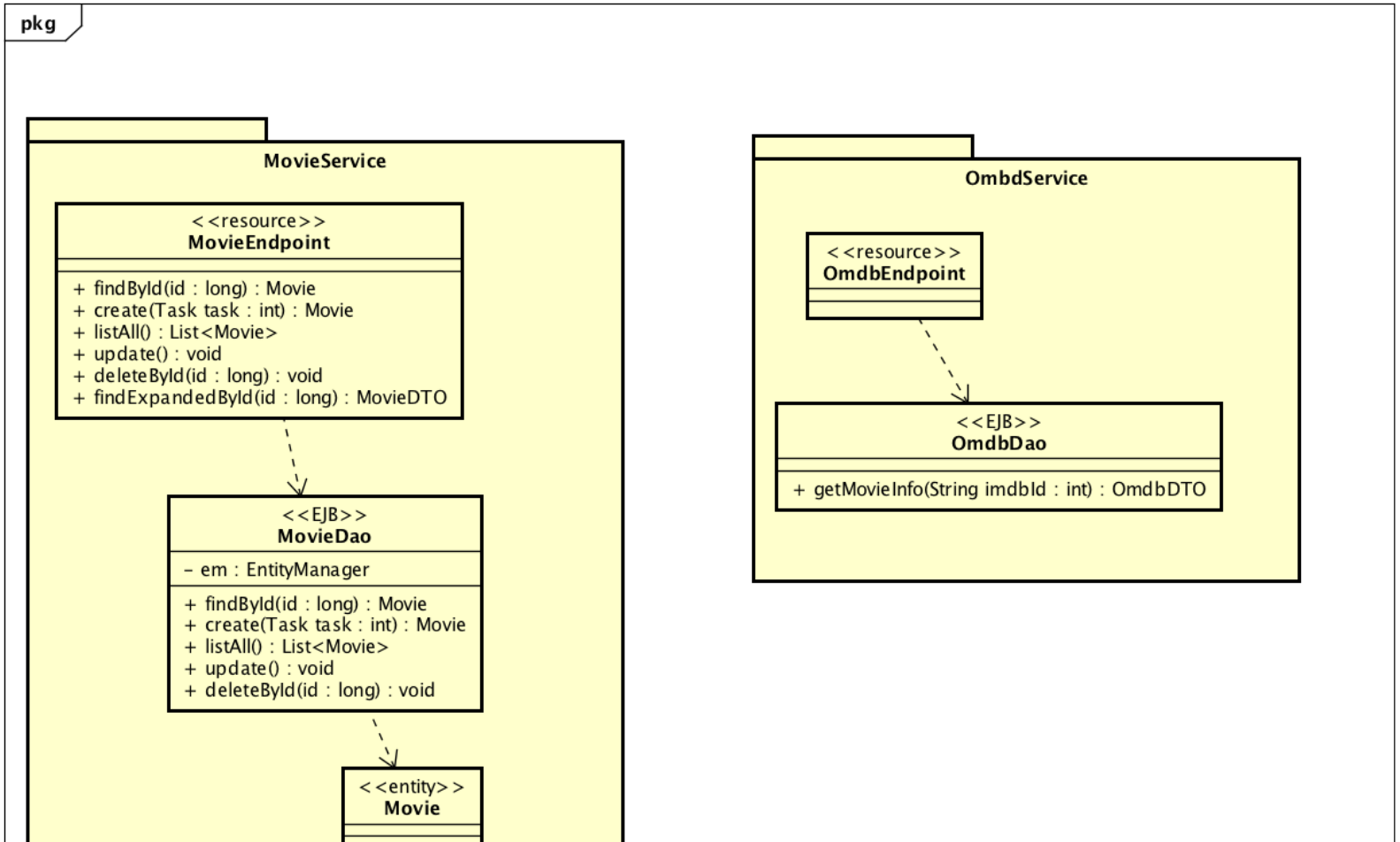


EE + MicroProfile - Demo

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;
```

NABENIK

Fault tolerance

- Circuit Breaker
- Bulkhead
- Fallback
- Retry
- Timeout

NABENIK

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) {
...
}
```

NABENIK

Metrics

Where

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

How

- Counted
- Gauge
- Metered
- Timed
- Histogram

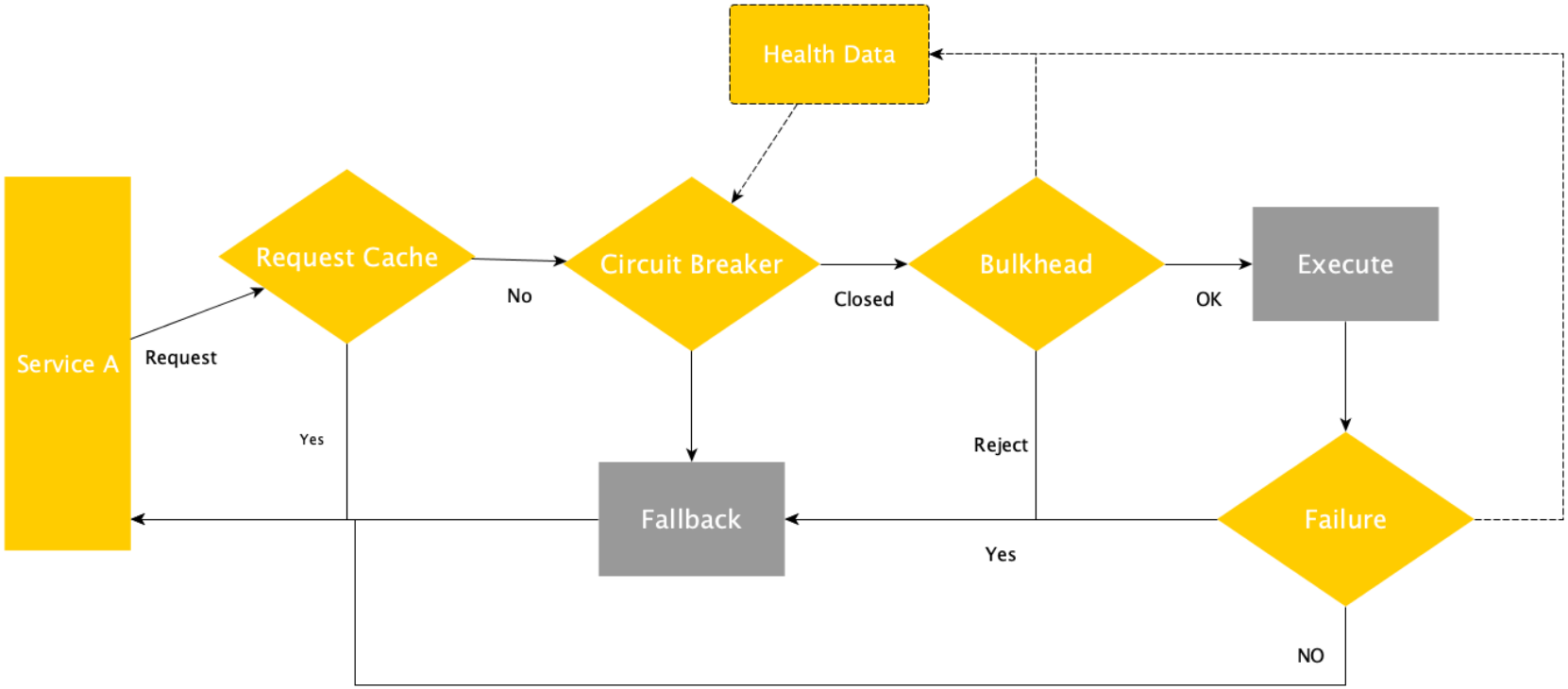
NABENIK

Practical Use Cases

NABENIK

Case 0 - Metrics for microservices

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



NABENIK

Case 0.1 - Counted

```
@Inject
@Metric
Counter failedQueries;

@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) {
        ...
        failedQueries.inc();
}
```

NABENIK

Case 0.2 - Gauge

Ideal for inc-dec counter in real time

```
@Gauge(unit = "ExternalDatabases", name = "movieDatabases",
public long getDatabases() {
        int number = (int)(Math.random() * 100);
        int criteria = number % 2;

        if(criteria == 0) {
                return 100;
        }else {
                return 50;
        }
}
```

/metrics/application/movieDatabases

NABENIK

Case 0.3 - Metered

Measure of events rate

```
@Metered(name = "moviesRetrieved",
         unit = MetricUnits.MINUTES,
         description = "Metrics␣to␣monitor␣movies",
         absolute = true)
public Response findExpandedById(@PathParam("id") final Long
```

/metrics/application/movieDatabases

NABENIK

Case 0.4 - Timed

Measure of events delay and performance

```
@Timed(name = "moviesDelay",
        description = "Metrics␣to␣monitor␣the␣times␣of␣movi
        unit = MetricUnits.MINUTES,
        absolute = true)
public Response findExpandedById(@PathParam("id") final Lon
```

```
/metrics/application/moviesDelay
```

NABENIK

Case 0.5 - Histogram

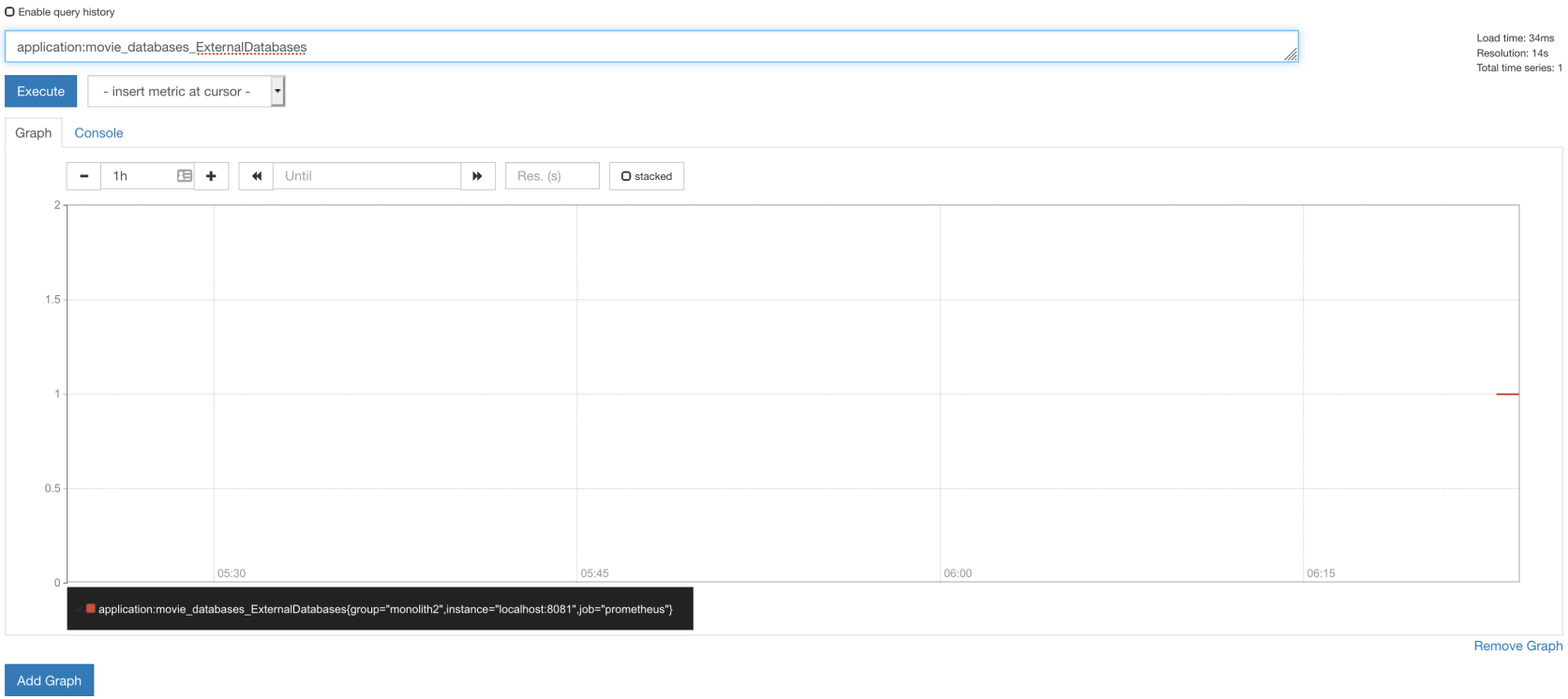
Distribution of a value

```
@Inject
MetricRegistry registry;

@POST
@Path("/add/{attendees}")
public Response addAttendees(@PathParam("attendees") Long a
        Metadata metadata =
        new Metadata("matrix␣attendees",
        MetricType.HISTOGRAM);
        Histogram histogram =
        registry.histogram(metadata);
        histogram.update(attendees);
        return Response.ok().build();
}
```

NABENIK

Case 0.99 - Prometheus



NABENIK

Case 0.999 - Application metrics

Prometheus Alerts Graph Status Help

Enable query history

Expression (press Shift+Enter for newlines)

Execute

Graph

Element	Value
no data	

Add Graph

- insert metric at cursor -
application:com_nabenik_demo_rest_movie_endpoint_find_expanded_by_id_fifteen_min_rate_per_second
application:com_nabenik_demo_rest_movie_endpoint_find_expanded_by_id_five_min_rate_per_second
application:com_nabenik_demo_rest_movie_endpoint_find_expanded_by_id_one_min_rate_per_second
application:com_nabenik_demo_rest_movie_endpoint_find_expanded_by_id_rate_per_second
application:com_nabenik_demo_rest_movie_endpoint_find_expanded_by_id_total
application:com_nabenik_omdb_rest_omdb_endpoint_failed_queries
application:com_nabenik_omdb_rest_omdb_endpoint_find_by_id_fifteen_min_rate_per_second
application:com_nabenik_omdb_rest_omdb_endpoint_find_by_id_five_min_rate_per_second
application:com_nabenik_omdb_rest_omdb_endpoint_find_by_id_one_min_rate_per_second
application:com_nabenik_omdb_rest_omdb_endpoint_find_by_id_rate_per_second
application:com_nabenik_omdb_rest_omdb_endpoint_find_by_id_total
application:ft_com_nabenik_omdb_rest_omdb_endpoint_find_by_id_fallback_calls_total
application:ft_com_nabenik_omdb_rest_omdb_endpoint_find_by_id_invocations_total
application:ft_com_nabenik_omdb_rest_omdb_endpoint_find_by_id_timeout_calls_not_timed_out_total
application:ft_com_nabenik_omdb_rest_omdb_endpoint_find_by_id_timeout_calls_timed_out_total
application:ft_com_nabenik_omdb_rest_omdb_endpoint_find_by_id_timeout_execution_duration
application:ft_com_nabenik_omdb_rest_omdb_endpoint_find_by_id_timeout_execution_duration_count
application:ft_com_nabenik_omdb_rest_omdb_endpoint_find_by_id_timeout_execution_duration_max
application:ft_com_nabenik_omdb_rest_omdb_endpoint_find_by_id_timeout_execution_duration_mean

NABENIK

Case 1 - Telemetry for monoliths

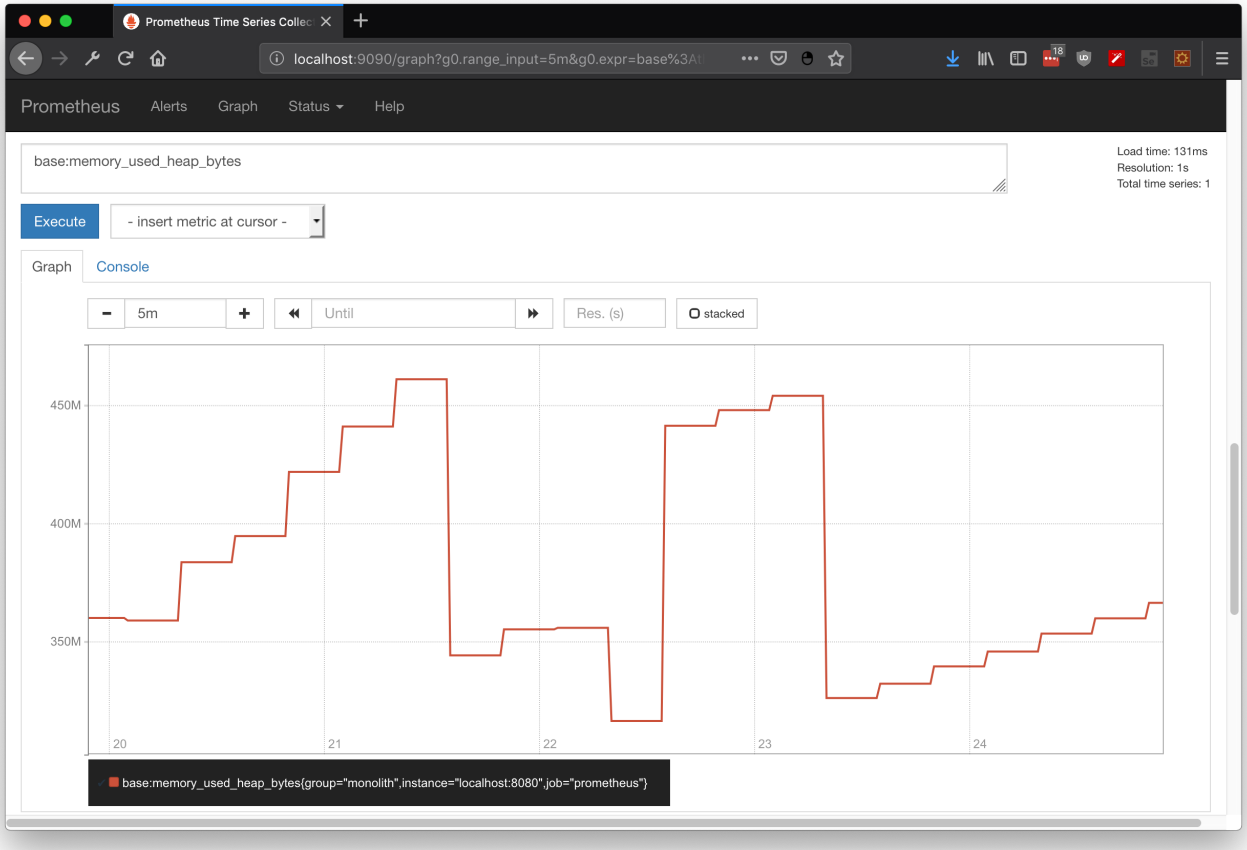
1. Base metrics are almost JVM metrics
2. JMX is also a pull-based monitoring technology
3. Ideal for PaaS

localhost:8080/metrics/base

```
# TYPE base:classloader_total_loaded_class_count counter
# HELP base:classloader_total_loaded_class_count Displays the total number of classes that have been loaded since the JVM has started execution.
base:classloader_total_loaded_class_count 19509
# TYPE base:cpu_system_load_average gauge
# HELP base:cpu_system_load_average Displays the system load average for the last minute. The system load average is the sum of the number of runnable
entities queued to the available processors and the number of runnable entities running on the available processors averaged over a period of time. The
way in which the load average is calculated is operating system specific but is typically a damped time-dependent average. If the load average is not
available, a negative value is displayed. This attribute is designed to provide a hint about the system load and may be queried frequently. The load
average may be unavailable on some platform where it is expensive to implement this method.
base:cpu_system_load_average 2.8955078125
# TYPE base:thread_count counter
# HELP base:thread_count Displays the current number of live threads including both daemon and non-daemon threads.
base:thread_count 117
# TYPE base:classloader_current_loaded_class_count counter
# HELP base:classloader_current_loaded_class_count Displays the number of classes that are currently loaded in the JVM.
base:classloader_current_loaded_class_count 19379
# TYPE base:jvm_uptime_seconds gauge
# HELP base:jvm_uptime_seconds Displays the uptime of the JVM in milliseconds.
base:jvm_uptime_seconds 251.985
# TYPE base:memory_committed_non_heap_bytes gauge
# HELP base:memory_committed_non_heap_bytes Displays the amount of memory in bytes that is committed for the JVM to use.
base:memory_committed_non_heap_bytes 1.71180032E8
# TYPE base:gc_ps_mark_sweep_count counter
# HELP base:gc_ps_mark_sweep_count Displays the total number of collections that have occurred. This attribute lists -1 if the collection count is
undefined for this collector.
base:gc_ps_mark_sweep_count 10
# TYPE base:memory_committed_heap_bytes gauge
# HELP base:memory_committed_heap_bytes Displays the amount of memory in bytes that is committed for the JVM to use.
base:memory_committed_heap_bytes 4.63470592E8
# TYPE base:thread_max_count counter
# HELP base:thread_max_count Displays the peak live thread count since the Java virtual machine started or peak was reset. This includes daemon and non-
daemon threads.
base:thread_max_count 159
# TYPE base:gc_ps_scavenge_count counter
# HELP base:gc_ps_scavenge_count Displays the total number of collections that have occurred. This attribute lists -1 if the collection count is undefined
```

NABENIK

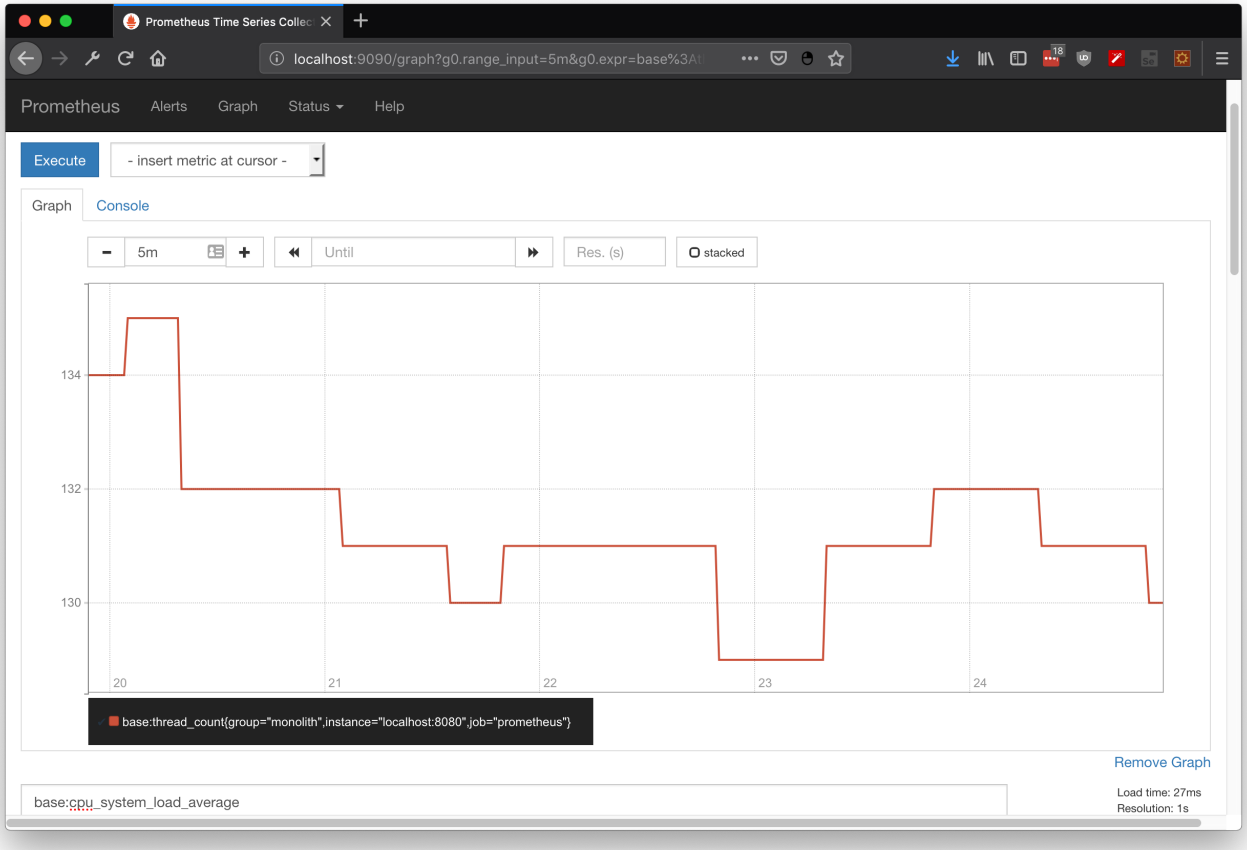
Case 1 - Heap performance



`base:memory_used_heap_bytes`

NABENIK

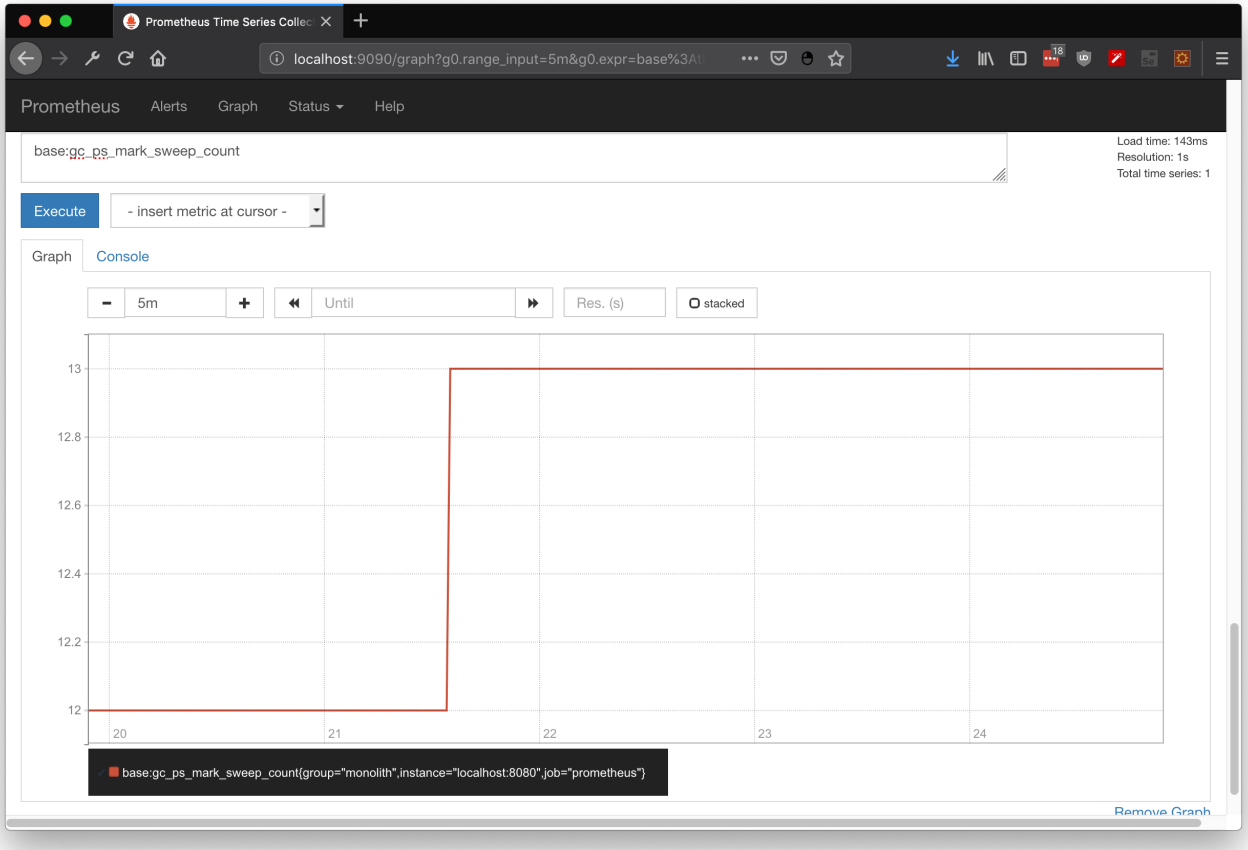
Case 1 - CPU Utilization



`base:cpu_system_load_average`

NABENIK

Case 1 - GC Executions



`base:gc_ps_mark_sweep_count`

NABENIK

Thank you

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

NABENIK