

Eclipse MicroProfile Metrics: Practical use cases

Jorge Cajas - José Diaz - Víctor Orozco

October 21, 2018

GuateJUG - PeruJUG

Overview

Why Metrics?

Monoliths

Reactive applications

Eclipse MicroProfile

Practical Use Cases

- 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

- JUG Leader



- JUG Leader



Why Metrics?

Why Metrics?

Do I need metrics?



[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.

Why Metrics?

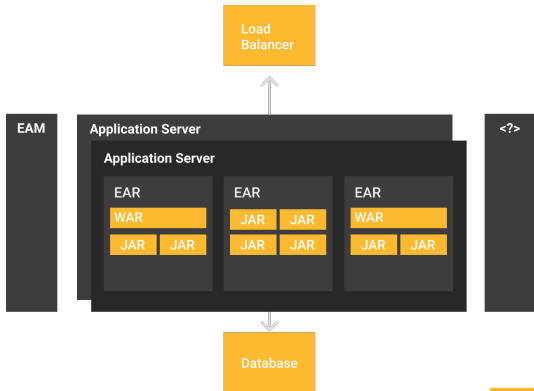
What about non-DevOps?



Monoliths

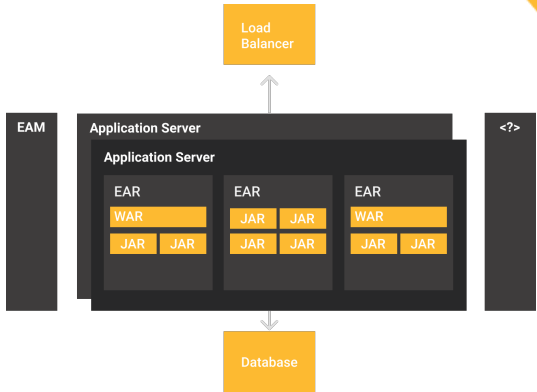
Metrics in Java Monoliths

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



Metrics in Java Monoliths

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



How do I choose between JMX or a telemetry API?

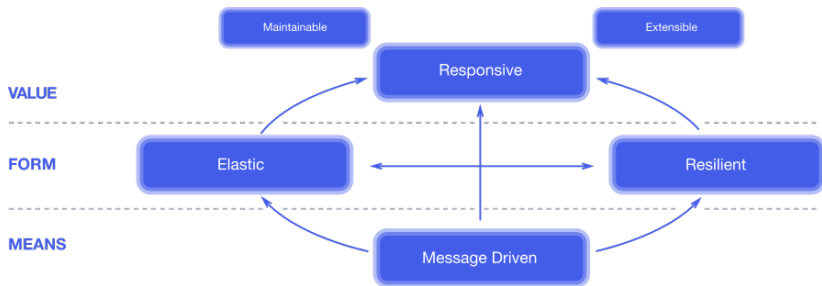
How do I get access to JMX if I'm using PaaS?



Reactive applications

Reactive applications

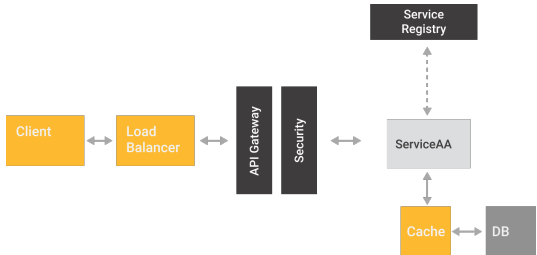
Reactive often means Microservices



Key concept: Non-long running JVM

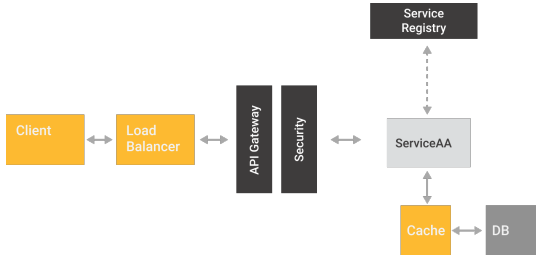
Metrics in Microservices

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



Metrics in Microservices

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



Eclipse MicroProfile

Eclipse MicroProfile

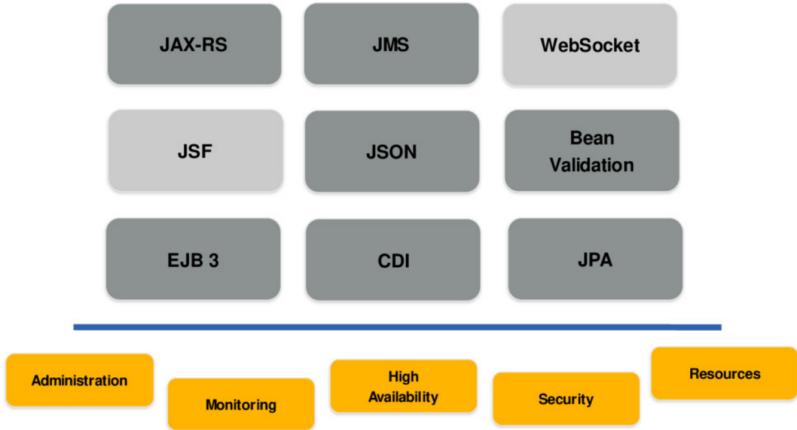
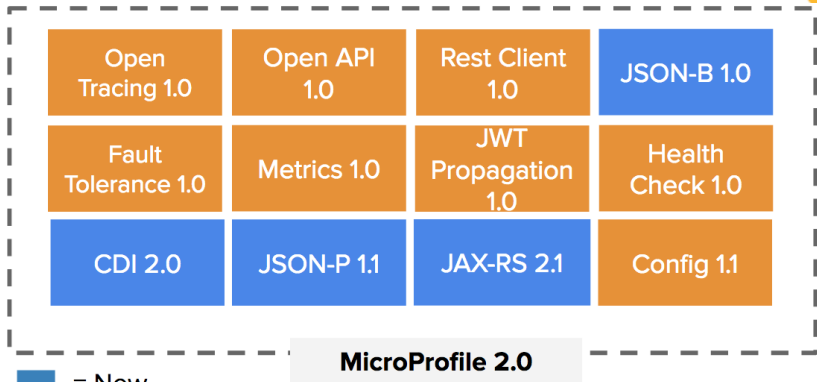




Figure 1: Credits: Reza Rahman

Eclipse MicroProfile



-  = New
-  = No change from last release

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>
```

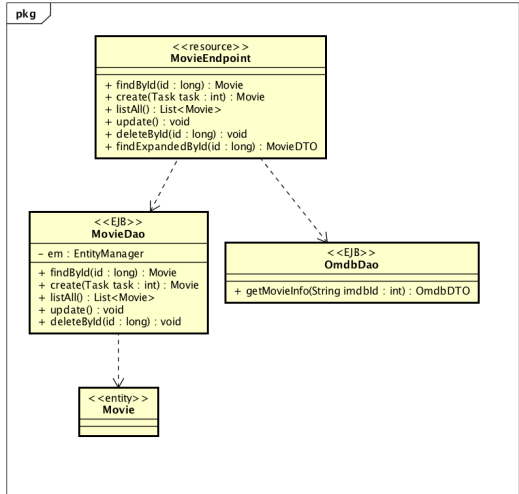
Java 8, JAX-RS, CDI, EJB, Microprofile

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

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

Take for granted

- EJB
- JTA
- JAX-RS
- CDI

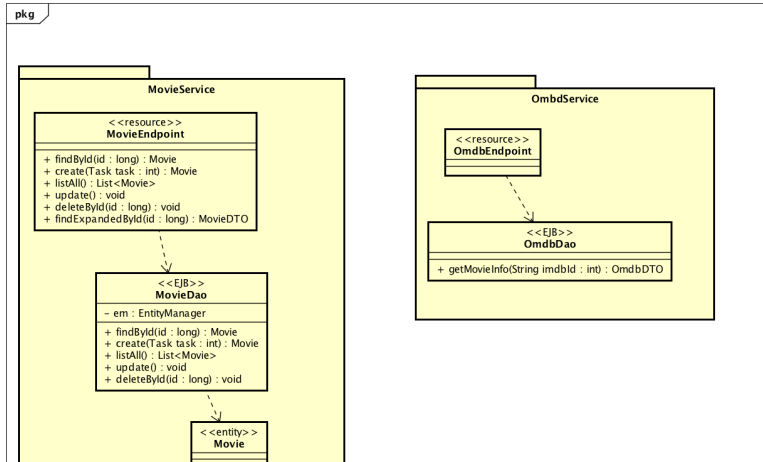


EE + MicroProfile - Demo

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

Payara Micro: EJB, JTA (Per service)

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




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

- 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) {
    ...
}
```

Where

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

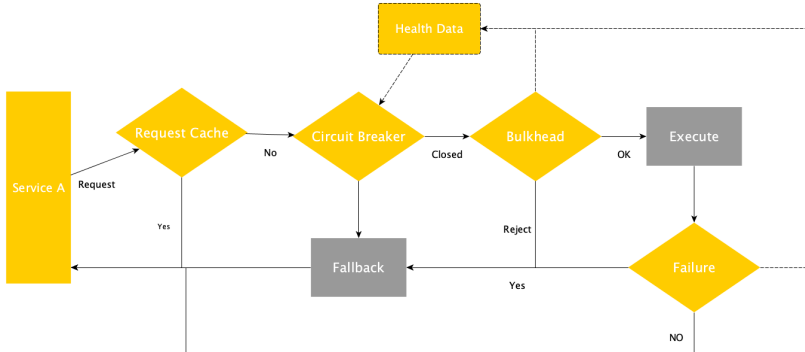
How

- Counted
- Gauge
- Metered
- Timed
- Histogram

Practical Use Cases

Case 0 - Metrics for microservices

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



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) {
    ...
}
```

Flat metric

```
@Gauge(unit = "ExternalDatabases",  
        name = "movieDatabases",  
        absolute = true)  
public long getDatabases() {  
    return 1;  
}
```

/metrics/application/movieDatabases

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
```

Measure of events delay and performance

```
@Timed(name = "moviesDelay",  
        description = "Metrics to monitor the times of movie",  
        unit = MetricUnits.MINUTES,  
        absolute = true)  
public Response findExpandedById(@PathParam("id") final Long  
  
/metrics/application/moviesDelay
```

Case 0.3 - Prometheus

☐ Enable query history

application:movie_databases_ExternalDatabases

Execute

- insert metric at cursor -

Graph Console

1h + Until Res. (s) stacked



application:movie_databases_ExternalDatabases(group="monolith",instance="localhost:9091",job="prometheus")

Remove Graph

Add Graph

Load time: 24ms
Resolution: 14s
Total time series: 1

Case 1 - Application metrics

Prometheus Alerts Graph Status ▾ Help

☐ Enable query history

Expression (press Shift+Enter for newlines)

Execute

Graph

Element

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

Value

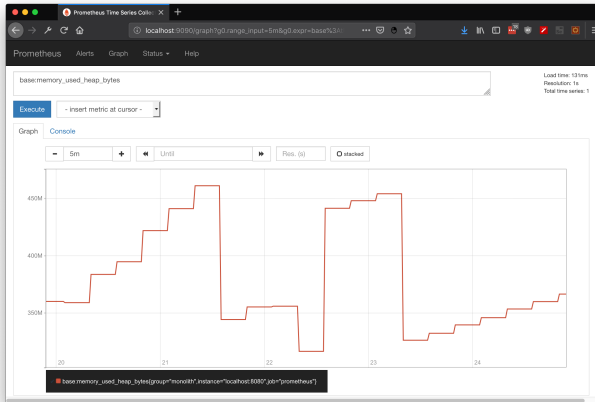


Case 1 - Telemetry for monoliths

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

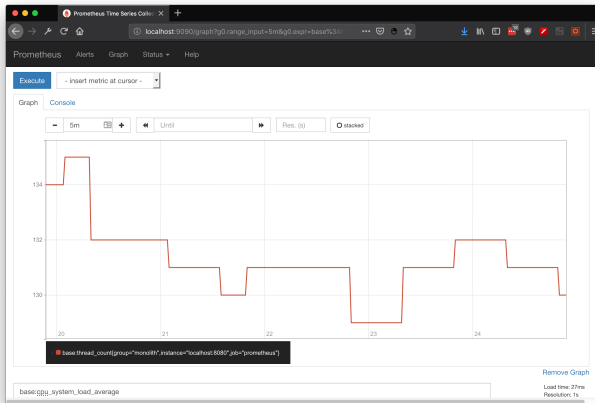
```
localhost:8080(metrics/base)
# TYPE base:classloader_total_loaded_class_count counter
# HELP base:classloader_total_loaded_class_count Displays the total number of classes that have been loaded since the JVM has started execution.
base:classloader_total_loaded_class_count 19509
# TYPE base:cpu_system_load_average 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 available 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
# TS 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
```

Case 1 - Heap performance



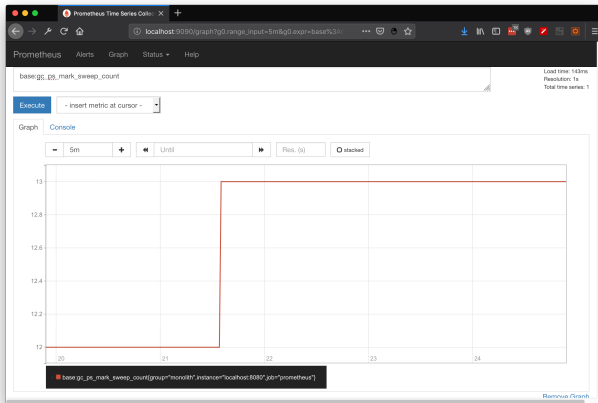
`base:memory_used_heap_bytes`

Case 1 - CPU Utilization



`base:cpu_system_load_average`

Case 1 - GC Executions



`base:gc_ps_mark_sweep_count`

Thank you

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



This work is licensed under a Creative Commons
Attribution-ShareAlike 3.0.