Developing Microservices With Quarkus and MicroProfile

March 25, 2020

Poll
What is your level of microservice experience?

(Select only One)

- No understanding / experience
- Basic understanding / experience
- Strong understanding / experience
- Have developed microservices
- Have decomposed a monolith into microservices

Poll
Experience developing with:

(Select all that apply)

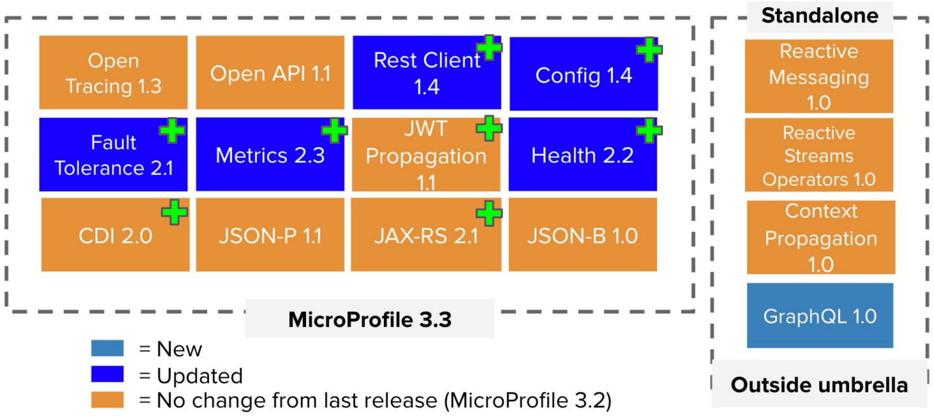
- Java EE / Jakarta EE
- Spring / Spring Boot
- Eclipse MicroProfile
- Micronaut
- Helidon
- Quarkus
- Other (Dropwizard, home grown Java stack, etc)

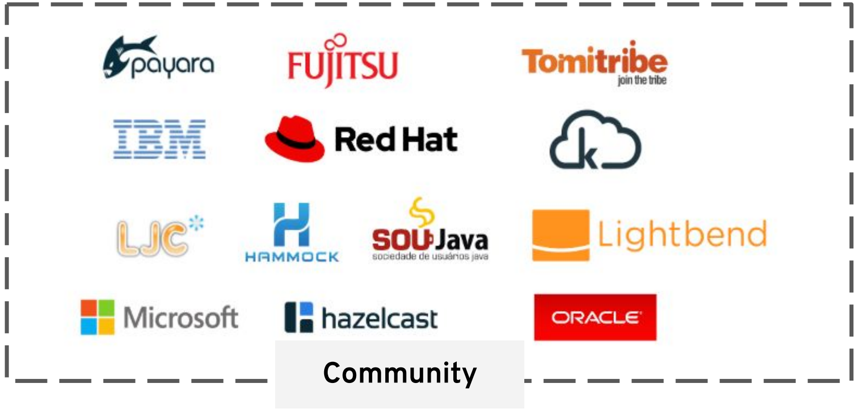
Eclipse MicroProfile

Optimizing Enterprise Java
for a Microservices Architecture

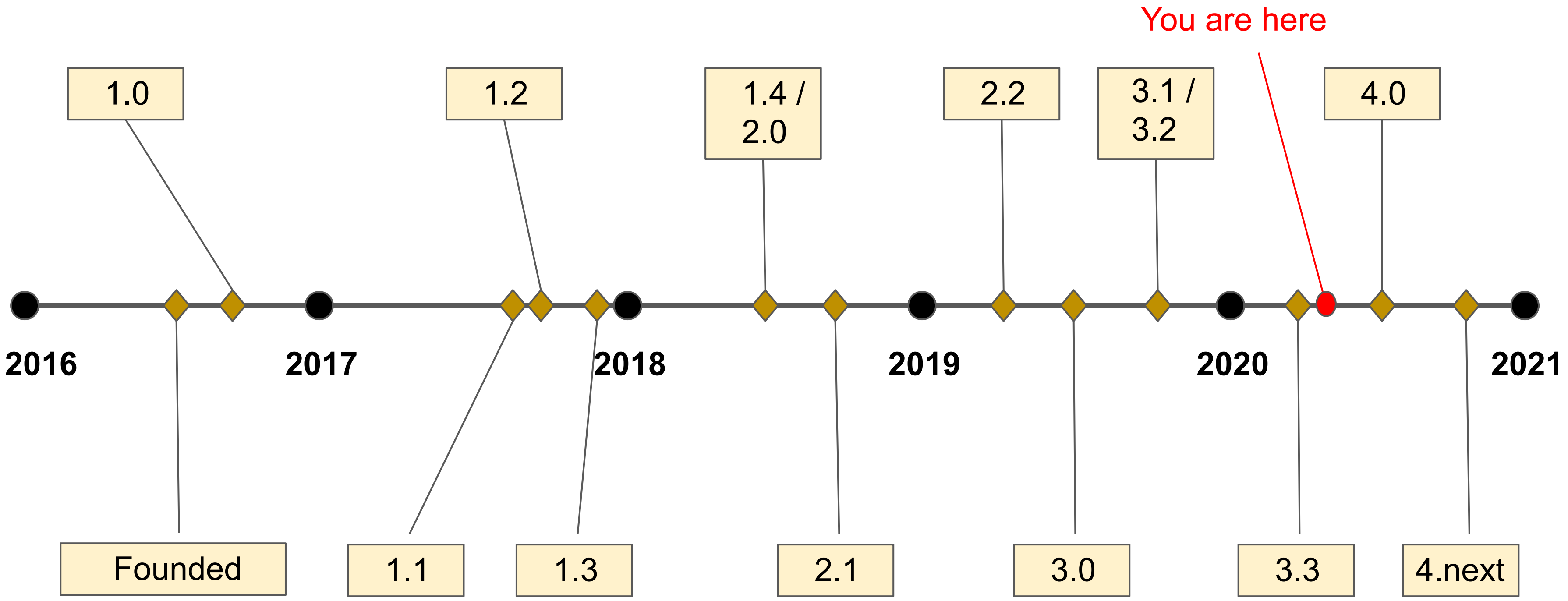
Eclipse MicroProfile

- Open Source community specifications for Enterprise Java microservices
- bit.ly/MicroProfileForum
- MicroProfile.io, @MicroProfileIO, http://bit.ly/MicroProfileYouTube





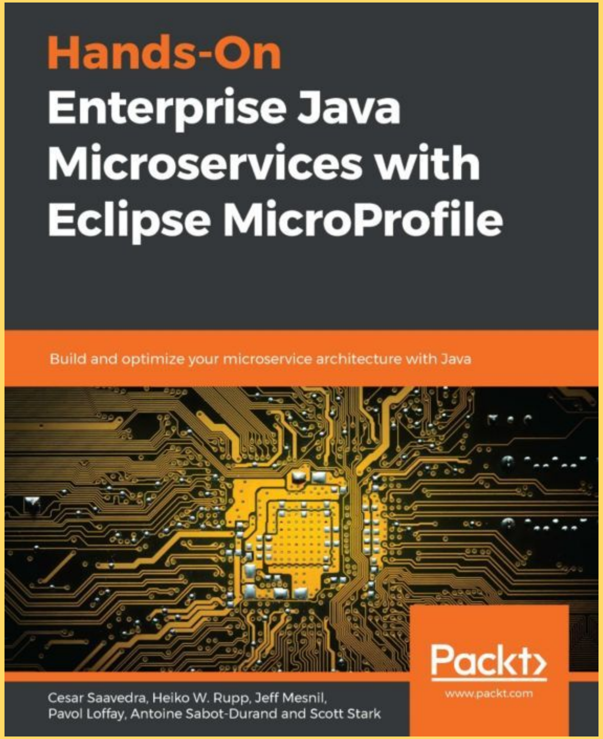
MicroProfile Timeline

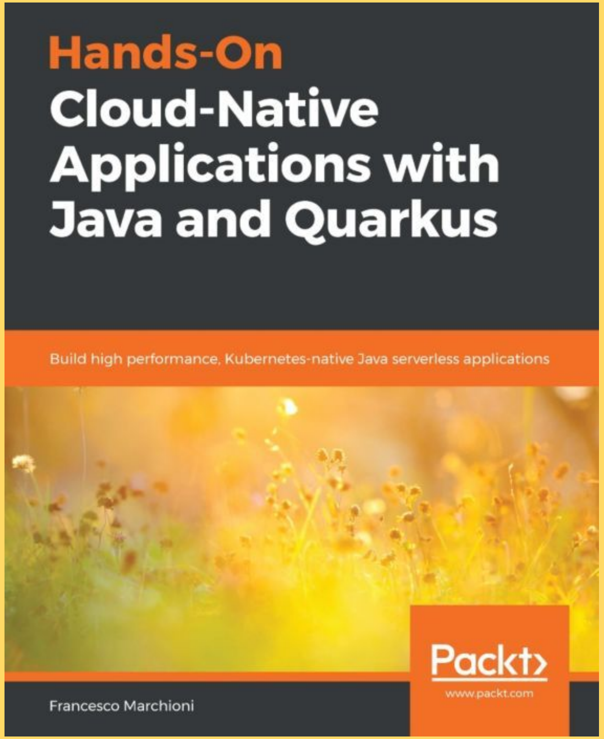


MicroProfile Relationship to Java EE / Jakarta EE

- MicroProfile and Java EE / Jakarta EE are different projects
  - Many participants participate in both projects
- MicroProfile utilizes 5 Java EE / Jakarta EE specifications
  - JAX-RS, CDI, JSON-P, JSON-B, Common Annotations
- Implementations
  - Most Java EE / Jakarta EE application servers support MicroProfile specifications
  - New generation of Java runtimes like Quarkus, Helidon, Piranha

Community


Hands-On
Enterprise Java Microservices with Eclipse MicroProfile
Build and optimize your microservice architecture with Java
Packt
www.packt.com
Cesar Saavedra, Heiko W. Rupp, Jeff Mesnil, Pavol Loffay, Antoine Sabot-Durand and Scott Stark


Hands-On
Cloud-Native Applications with Java and Quarkus
Build high performance, Kubernetes-native Java serverless applications
Packt
www.packt.com
Francesco Marchioni

bit.ly/HandsOnMicroProfile

bit.ly/HandsOnCloudNativeQuarkus

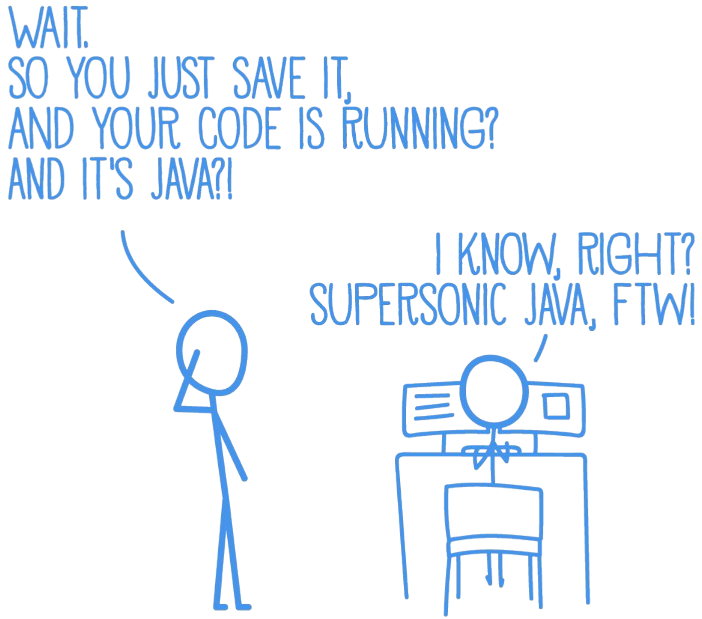
Introduction to Quarkus

Kubernetes-Native Java

Benefit No. 1: Developer Joy

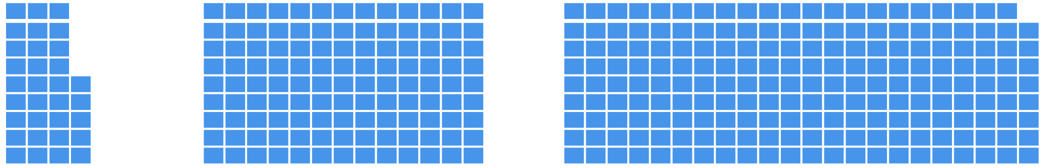
A cohesive platform for optimized developer joy:

- Zero config, live reload in the blink of an eye
- Supports standards, but not limited to them
- Lower barrier to entry w/Spring API compatibility, Vert.x, and Java EE / MicroProfile
- Unified configuration
- Streamlined code for the 80% common usages, flexible for the 20%
- No hassle native executable generation



Benefit No. 2: Supersonic Subatomic Java

REST + CRUD

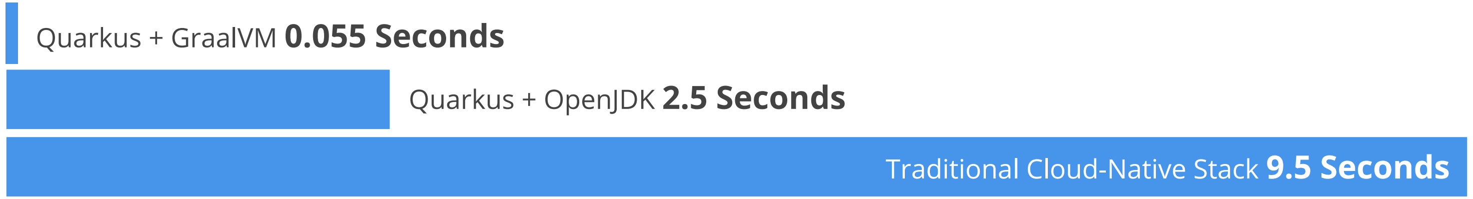


Quarkus + GraalVM **35 MB**

Quarkus + OpenJDK **130 MB**

Traditional Cloud-Native Stack **218 MB**

REST + CRUD



Quarkus + GraalVM **0.055 Seconds**

Quarkus + OpenJDK **2.5 Seconds**

Traditional Cloud-Native Stack **9.5 Seconds**

Time to first response

Benefit No. 3: Unifies Imperative and Reactive

```
@Inject
SayService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hello() {
      return say.hello();
}
```

```
@Inject @Channel(”kafka”)
Publisher<String> reactiveSay;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public Publisher<String> stream() {
      return reactiveSay;
}
```

- Combine both Reactive and imperative development in the same application
- Use the technology that fits your use-case
- Key for reactive systems based on event driven apps

Benefit No. 4: Best of Breed Frameworks & Standards

Quarkus provides a cohesive, fun to use, full-stack framework by leveraging a growing list of over fifty best-of-breed libraries that you love and use. All wired on a standard backbone.

Quarkus
HANDS ON

Mixing Spring APIs and MicroProfile APIs

- Quarkus Spring support
  - Spring DI
  - Spring Web
  - Spring Data JPA
  - Spring Security
  - Spring Cloud Config Server

- Mix and match Spring, MicroProfile, Vert.x, native Quarkus APIs
  - In same application
  - In same Java class
  - Sample repository (Spring + MicroProfile)

Q & A

MicroProfile Config

Externalizing Application Configuration

Why Externalize Configuration?

- Separate configuration values from application logic
- Configuration differs depending on environment
- Separate roles and responsibilities
- Abstracts configuration source

Introduction to MicroProfile Config

- Programmatic APIs & annotation (DI) based APIs
- 3 "out-of-the-box" Configuration sources
- Custom config sources
- Converters, built-in and custom converter API

Configuration Source	Ordinal
System properties	400
Environmental Variables	300
microprofile-config.properties	100

Highest Priority

$\downarrow$

Lowest Priority

MicroProfile Runtime Configuration

- A MicroProfile runtime *may* use MicroProfile Config to configure itself
- Quarkus (All configuration properties)
    - Uses MicroProfile Config for configuration

    - Configuration sources
        - Environment variables, system properties
        - Microprofile-config.properties, application.properties, application.yaml
        - Eureka (community), Vault, Spring Cloud Config Client [Experimental]

    - Native compilation - Subset of configuration properties overridable at runtime

MicroProfile Config
HANDS ON

Q & A

MicroProfile REST Client

Type-safe Invocation of RESTful Endpoints

MicroProfile REST Client

- Leverages JAX-RS annotations on an interface to describe contract with remote service (@POST, @GET, etc).
- New annotations to augment behavior, like header propagation (@ClientHeaderParam)
- Integration with other MicroProfile specifications, like MicroProfile Config and Fault Tolerance
- Asynchronous support (client interface returns CompletionStage)
- CDI and programmatic builder API

Sample Rest Client Configuration Parameters

- **org.acme.myClient/mp-rest/url.** Service base URL
- **org.acme.myClient/mp-rest/scope**. FQCN to a CDI scope.
- **org.acme.myClient/mp-rest/providers**. List of FQCN to include in client.
- **org.acme.myClient/mp-rest/connectTimeout**. Time (in ms) to wait for remote connection.
- **org.acme.myClient/mp-rest/readTimeout**. Time (in ms) to wait for response
- **org.acme.myClient/mp-rest/trustStore**. Path to Java key store.
- **org.acme.myClient/mp-rest/trustStorePassword**. Trust store password.
- **org.acme.myClient/mp-rest/trustStoreType**. Defaults to "JKS"

MicroProfile Rest Client HANDS ON

Q & A

Fault Tolerance

Improving Application Robustness

Introduction to MicroProfile Fault Tolerance

- Multiple strategies for handling undesirable conditions

Fault Tolerance Annotations

Annotation	Description
@Asynchronous	Executes logic in a separate thread
@Bulkhead	Limits number of concurrent requests
@CircuitBreaker *	Prevents repeated failures.
@Fallback *	Logic called when a method completes "exceptionally"
@Retry *	Retries an operation
@Timeout *	Prevents execution from waiting longer than desired

* Used in this tutorial

Introduction to MicroProfile Fault Tolerance

- Multiple strategies for handling undesirable conditions
- Implemented as Interceptors
- Fault tolerance annotations can be combined
- All interceptor parameters are configurable via MicroProfile Config
- When used with MicroProfile Metrics, metrics are automatically added for fault tolerance annotations

MicroProfile Fault Tolerance HANDS ON

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
1	Good	Closed	0	1
2	Good	Closed	0	2
3	Good	Closed	0	3
4	Good	Closed	0	4
5	Good	Closed	0	5
6	Good	Closed	0	6
7	Good	Closed	0	7
8	Good	Closed	0	8
9	Good	Closed	0	9
10	Good	Closed	0	10

Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (4 * .5 = 2)
- Delay: 10 seconds
- successThreshold: 3

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
2	Good	Closed	0	2
3	Good	Closed	0	3
4	Good	Closed	0	4
5	Good	Closed	0	5
6	Good	Closed	0	6
7	Good	Closed	0	7
8	Good	Closed	0	8
9	Good	Closed	0	9
10	Good	Closed	0	10
11	Bad	Closed	1	10

$\left.\begin{array}{l} 8 \\ 9 \\ 10 \\ 11 \end{array}\right\}$ Request Window

<u>Circuit Breaker Settings</u>

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
3	Good	Closed	0	3
4	Good	Closed	0	4
5	Good	Closed	0	5
6	Good	Closed	0	6
7	Good	Closed	0	7
8	Good	Closed	0	8
9	Good	Closed	0	9
10	Good	Closed	0	10
11	Bad	Closed	1	10
12	Good	Closed	1	11

Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
4	Good	Closed	0	4
5	Good	Closed	0	5
6	Good	Closed	0	6
7	Good	Closed	0	7
8	Good	Closed	0	8
9	Good	Closed	0	9
10	Good	Closed	0	10
11	Bad	Closed	1	10
12	Good	Closed	1	11
13	Good	Closed	1	12

Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
5	Good	Closed	0	5
6	Good	Closed	0	6
7	Good	Closed	0	7
8	Good	Closed	0	8
9	Good	Closed	0	9
10	Good	Closed	0	10
11	Bad	Closed	1	10
12	Good	Closed	1	11
13	Good	Closed	1	12
14	Bad	Open	2	12

Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
6	Good	Closed	0	6
7	Good	Closed	0	7
8	Good	Closed	0	8
9	Good	Closed	0	9
10	Good	Closed	0	10
11	Bad	Closed	1	10
12	Good	Closed	1	11
13	Good	Closed	1	12
14	Bad	Open	2	12
15	Bad	Open	3	12

Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
7	Good	Closed	0	7
8	Good	Closed	0	8
9	Good	Closed	0	9
10	Good	Closed	0	10
11	Bad	Closed	1	10
12	Good	Closed	1	11
13	Good	Closed	1	12
14	Bad	Open	2	12
15	Bad	Open	3	12
16*	Good	Open	4	12

Request Window

<u>Circuit Breaker Settings</u>

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3

* Within delay window

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
8	Good	Closed	0	8
9	Good	Closed	0	9
10	Good	Closed	0	10
11	Bad	Closed	1	10
12	Good	Closed	1	11
13	Good	Closed	1	12
14	Bad	Open	2	12
15	Bad	Open	3	12
16*	Good	Open	4	12
17	Good	Half Open	4	13[1]

Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3[1,2,3]

* Within delay window

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
9	Good	Closed	0	9
10	Good	Closed	0	10
11	Bad	Closed	1	10
12	Good	Closed	1	11
13	Good	Closed	1	12
14	Bad	Open	2	12
15	Bad	Open	3	12
16*	Good	Open	4	12
17	Good	Half Open	4	13[1]
18	Good	Half Open	4	14[2]

$\}$ Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3[1,2,3]

* Within delay window

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
10	Good	Closed	0	10
11	Bad	Closed	1	10
12	Good	Closed	1	11
13	Good	Closed	1	12
14	Bad	Open	2	12
15	Bad	Open	3	12
16*	Good	Open	4	12
17	Good	Half Open	4	13[1]
18	Good	Half Open	4	14[2]
19	Good	Closed	4	15[3]

Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3[1,2,3]

* Within delay window

Circuit Breaker

Request ID	Service State	CircuitBreaker State	CB Failed Requests	CB Successful Requests
11	Bad	Closed	1	10
12	Good	Closed	1	11
13	Good	Closed	1	12
14	Bad	Open	2	12
15	Bad	Open	3	12
16*	Good	Open	4	12
17	Good	Half Open	4	13[1]
18	Good	Half Open	4	14[2]
19	Good	Closed	4	15[3]
20	Good	Closed	4	16

Request Window

Circuit Breaker Settings

- RequestVolumeThreshold: 4
- failureRatio: .5 (2 failures)
- Delay: 10 seconds
- successThreshold: 3[1,2,3]

* Within delay window

MicroProfile Fault Tolerance HANDS ON

Q & A

Metrics

Expose Telemetry of a Running Server

Introduction to MicroProfile Metrics

- Why MicroProfile Metrics
  - Easy-to-use API
  - Cloud-friendly - supports OpenMetrics (aka Prometheus) and JSON formats
  - Includes metadata - description, units of measure
- Scopes
  - **Base**: Required by every implementation (memory, CPU, JVM)
  - **Vendor**: Provided by and specific to a (runtime) implementation
  - **Application**: Custom metrics
- Inspired by Dropwizard Metrics

Quarkus Metrics

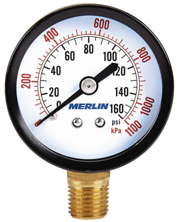
Extension	Description	Property (set to true)
quarkus-resteasy	JAX-RS. Included by default. Supports *optional* Metrics 2.3 metrics	quarkus.resteasy.metrics.enabled
quarkus-agroal	Connection pool (included with Hibernate). Named datasource metrics	quarkus.datasource.metrics.enabled quarkus.datasource."datasource-name".jdbc.enable-metrics
quarkus-hibernate-orm	Hibernate statistics and metrics	quarkus.hibernate-orm.statistics quarkus.hibernate-orm.metrics.enabled
quarkus-mongodb-client	Mongo client metrics	quarkus.mongodb.metrics.enabled
quarkus-neo4j	Neo4j client metrics	quarkus.neo4j.pool.metrics-enabled
quarkus-smallrye-reactive-messaging	Reactive messaging metrics, connectors: kafka, AMQP, MQTT	quarkus.reactive-messaging.metrics.enabled

Metrics Types

@Counted

Counts object invocations

@Gauge,
@ConcurrentGauge



Samples value, Parallel invocations

@Metered

Tracks frequency of invocations

@Timed
@SimplyTimed

Tracks duration

MicroProfile Metrics
HANDS ON

Q & A

MicroProfile Health

MicroProfile Health

- Goal: Enable zero-downtime deployments
- "Liveness"
    - Is a service in a state it can recover from? If not, restart the container (or Pod)
    - MicroProfile @Liveness - Custom logic checks liveness.
    - Example: Misconfigured service. Liveness probe can stop rolling upgrade
- "Readiness"
    - Platform (ex: load balancer) will not send requests until a service is "ready".
    - MicroProfile @Readiness - Custom logic checks readiness
    - Example: Pre-populate a cache, wait for a database connection
- UP: HTTP Response Code 200. Healthy.
- DOWN: HTTP Response Code 500. Unhealthy.
- DOWN: HTTP Response code 503. Not ready to respond to requests

Quarkus Built-In Health Readiness Checks

- Datasources, MongoDB, Neo4j, Artemis (JMS), Kafka client
- Properties to enable/disable readiness checks

Data Source Health Readiness Example

```
{
  "status": "DOWN",
  "checks": [
    {
     "name": "Database connections health check",
     "status": "DOWN",
     "data": {
        "default": "Unable to execute the validation check for the default DataSource: Connection to
:5432 refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP
connections."
     } } ] }
```

Probe Configuration

docker-compose.yaml

```
healthcheck:
  timeout: 5s
  interval: 1s
  retries: 0
  test: curl --fail -s \
         http://localhost:8081/health/live || exit 1
```

Dockerfile

```
HEALTHCHECK CMD \
     --retries=2 \
     --internval=1m \
     curl --fail http://localhost:8080/health || exit 1
```

docker command

```
docker run -rm \
--health-cmd="curl … " \
--health-interval=5s \
acme/student:1.0
```

Kubernetes

```
livenessProbe:
  httpGet:
    path: /health/live
    port: 8080
  failureThreshold: 1
  periodSeconds: 10
```

- Supports tcpSocket
- Supports exec command

MicroProfile Health
HANDS ON

Q & A

MicroProfile Interoperable JWT RBAC

Securing Microservices

What is it?

OpenID Connect (OIDC) based

JWT (JSON Web Tokens)

for RBAC (Role-based Access Control)

JWT Token

Header
Body
Signature

eyJraWQiOiJqd3Qua2V5IiwidHlwIjoiSldUIiwiYWxnIjoiUlMyNTYifQ.**eyJzdWIiOiJ1
c2VyXC80Mzk3MSIsInVwbiI6ImRlbW9AYWNtZS5vcmciLCJteWMiOiJNeSBDd
XN0b20gQ2xhaW0iLCJhdXRoX3RpbWUiOjE1Nzg2NTEyODMsImlzcyI6ImFpc
mhhY2tzIiwiZ3JvdXBzIjpbInVzZXIiLCJhZG1pbiJdLCJleHAiOjMxNTU4ODI4O
TgsImlhdCI6MTU3ODY1MTI4MywianRpIjoiYWlyaGFja3Mtand0LXVuaXF1ZS1
pZC0xMjM0MjE0MiJ9**.*Eaqe3sTH64doIVW3on25EA_uD9XrfppndiweUNLVbFK3K
xaIfXaAdQ4N9IkQG6Iw0A7I7kngjeSHwb2DzH8rQE8yp7sCtey6kmC689eQC0j2k-
YbyGZ68xnsMj5taOBVGH_ZSWC6E1L-Gk-GgcTvX6I3SaBC8pwZ267q6psknqIAt
fD2JoE7ezEb7LrLVwP1vaGqKzC2X6pv5J-07DNBqe75uBWQyqX_WE856ug3uq
WcHtNck8nqU6VhwXqxHZ6vkRIx9VoMgFUF851D-WuKMCUdfXJHekDyKmjYuyL
iw7jtQSdIiY3ONOXgFm_uzjKGuZ1VKPdQXyx7GQ9NsNTYfw*

Security Tokens

- Lightweight and interoperable way to propagate identities
- Well known format, so services can validate token
- JWT token is self-contained and does not require a 3rd party service to validate
- MicroProfile JWT-specific requirements
  - Usable as a authentication token
  - Usable as an authorization token
  - Can be mapped to Java EE Security JSR IdentityStore
  - Support registered claims (IANA JWT Assignments)
  - Two new (customer) claims
    - upn: human-readable claim that uniquely identifies a principal/subject
    - group: Subject's group membership that can be mapped to Java EE-style roles

MicroProfile JWT Header

Claim	Description	Example
typ	Must be JWT	Must be JWT
alg	Must be RS256	Must be RS256
kid	Key ID - Hint indicating which key was used to secure JWS; signals a change of key to recipients	jwt.key

MicroProfile JWT Body

Claim	Description	Example
iss	Token issuer	airhacks
alg	Must be RS256	Must be RS256
sub	Principal that is subject of the token	user/43971
iat	Epoch time token issued (seconds since Jan 1, 1970)	1579415241
exp	Epoch time token expires (seconds since Jan 1, 1970)	1579415260
jti	JWT "reasonably unique" identifier	airhacks-jwt-unique-id-12342142

MicroProfile JWT Body (Continued)

Claim	Description	Example
upn	MP-JWT custom claim. This MP-JWT custom claim is the user principal name in the java.security.Principal interface, and is the caller principal name in javax.security.enterprise.identitystore.IdentityStore.	demo@acme.org
groups	This MP-JWT custom claim is the list of group names that have been assigned to the principal of the MP-JWT. This typically will required a mapping at the application container level to application deployment roles	user, admin

MicroProfile JWT Java APIs & Behavior

- **Claim**: enum of all (IANA) registered claims
  - Injecting a claim must be @RequestScoped
- **JsonWebToken** (interface extends Principal): Access Claims
  - Injecting a Principal must inject a JsonWebToken
- Required JAX-RS containers support
  - javax.ws.rs.core.SecurityContext.getUserPrincipal() must return a JsonWebToken
  - javax.ws.rs.core.SecurityContext#isUserInRole(String) must include JWT group claims
- Common Security Annotations (JSR 250) must work as expected in MP-JWT containers
  - PermitAll, DenyAll, RolesAllowed
  - RolesAllowed() must include JWT group claims

MicroProfile JWT Recommended Integrations

- EJB Container
  - javax.ejb.SessionContext.getCallerPrincipal() must return a JsonWebToken
  - javax.ejb.SessionContext#isCallerInRole(String) must include JWT group claims

- ServletContainer
  - javax.servlet.http.HttpServletRequest.getUserPrincipal() must return a JsonWebToken
  - javax.servlet.http.HttpServletRequest#isUserInRole(String) must include JWT group claims

JWT RBAC Lab

Q & A

Packaging, Deploying, Monitoring

Packaging, Deploying, Monitoring

- Packaging
  - Thin Jar
  - Docker image
    - Layer with jar libraries created once
    - Layer with small application jar created each time
  - (Optional) Instructions for using native binaries
- Deploying - docker-compose
- Monitoring - Prometheus & Grafana

Packaging, Deploying, Monitoring

HANDS ON

Q & A

Eclipse MicroProfile

The End

MicroProfile Metrics REST Response Codes

Response Code	**Description**
200	Successful retrieval of object
204	Subtree has no content
401	Unauthorized (if security enabled)
404	Directly addressed item does not exist
406	Accept header cannot be handled
500	Request failed due to bad health, body should contain error