DevDays 19

Lightweight Enterprise Java

With MicroProfile

Jean-Louis Monteiro
Tomitribe

DevDays 2019

Agenda

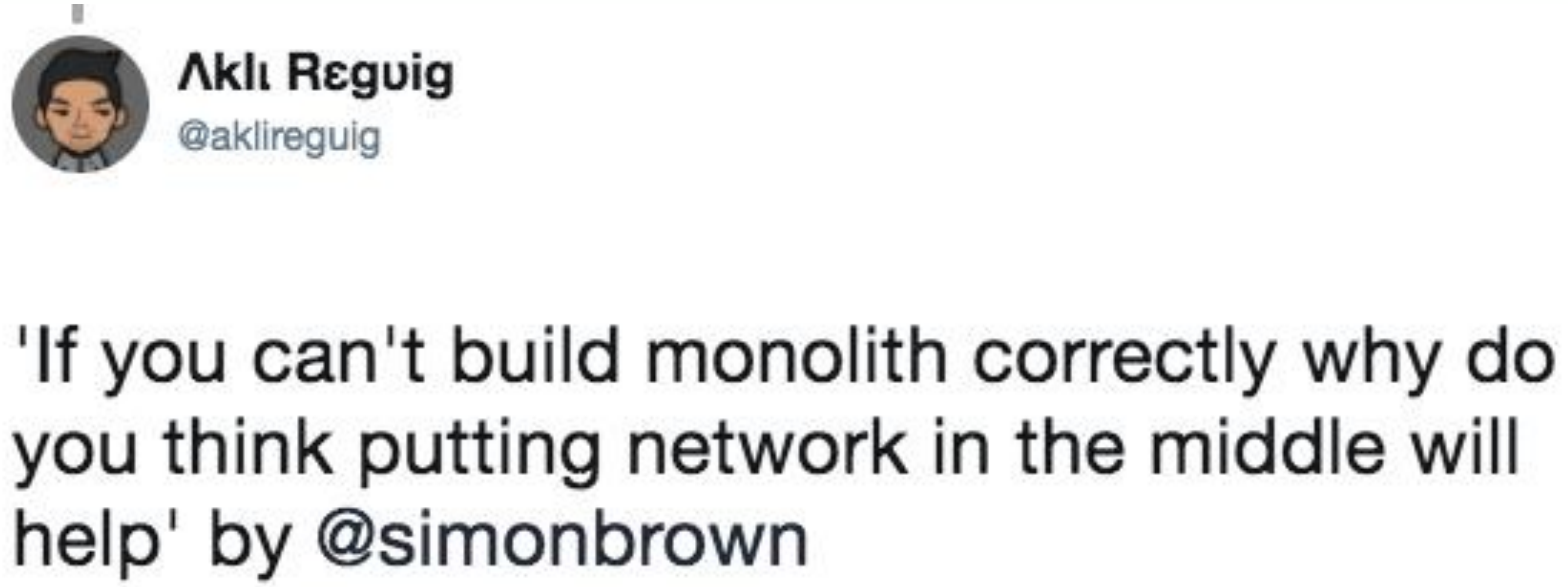
- Microservices!
- What is MicroProfile?
- Specs Overview
- Demos
- MicroProfile Future

Microservices Anyone?

What do people mean by Microservices?

- “Small autonomous services working together”
- “Small enough but not too small”
- “Can be written in 2 weeks”
- “Single responsibility principle”
- “Domain-driven design”

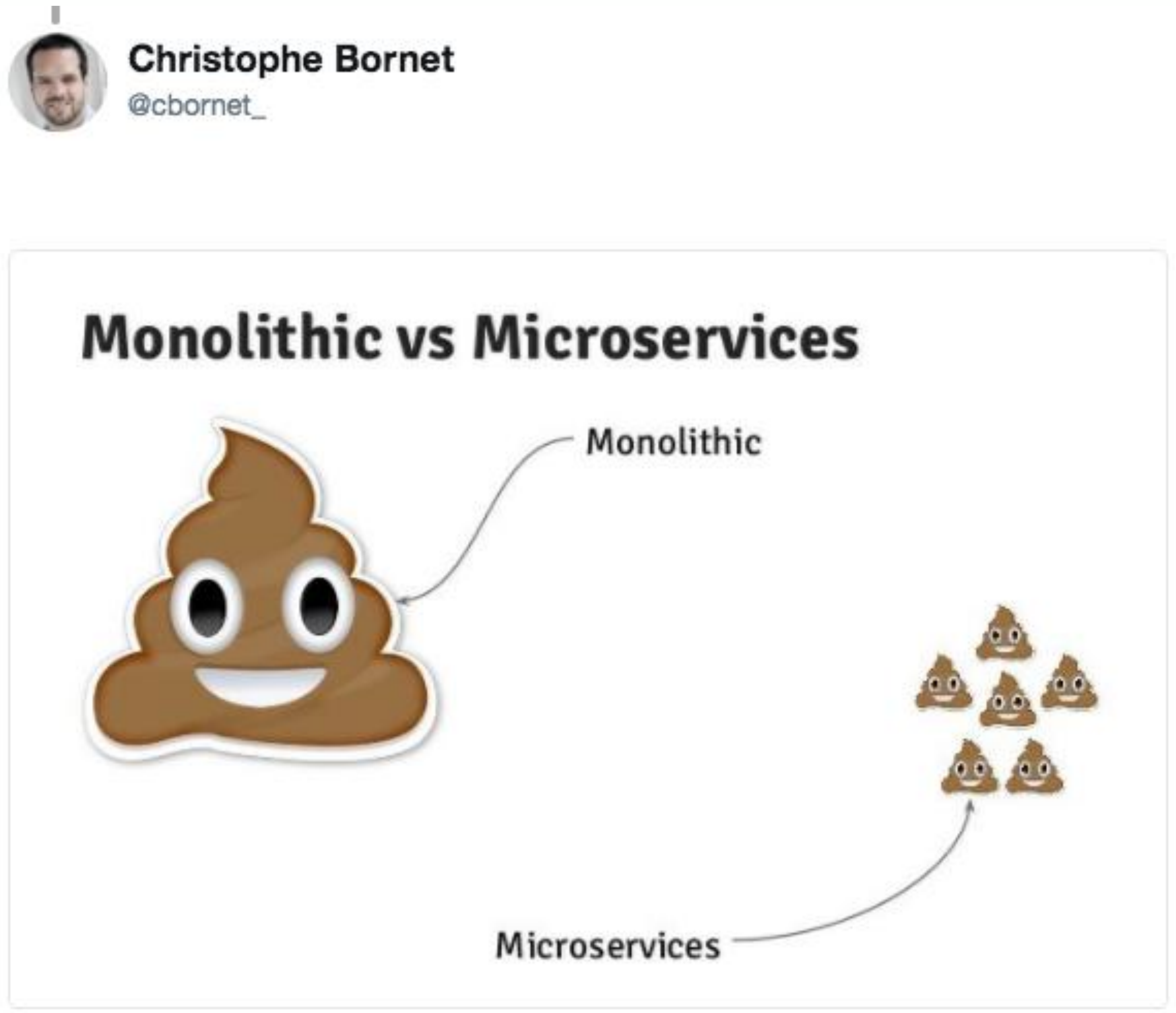
What do others think?



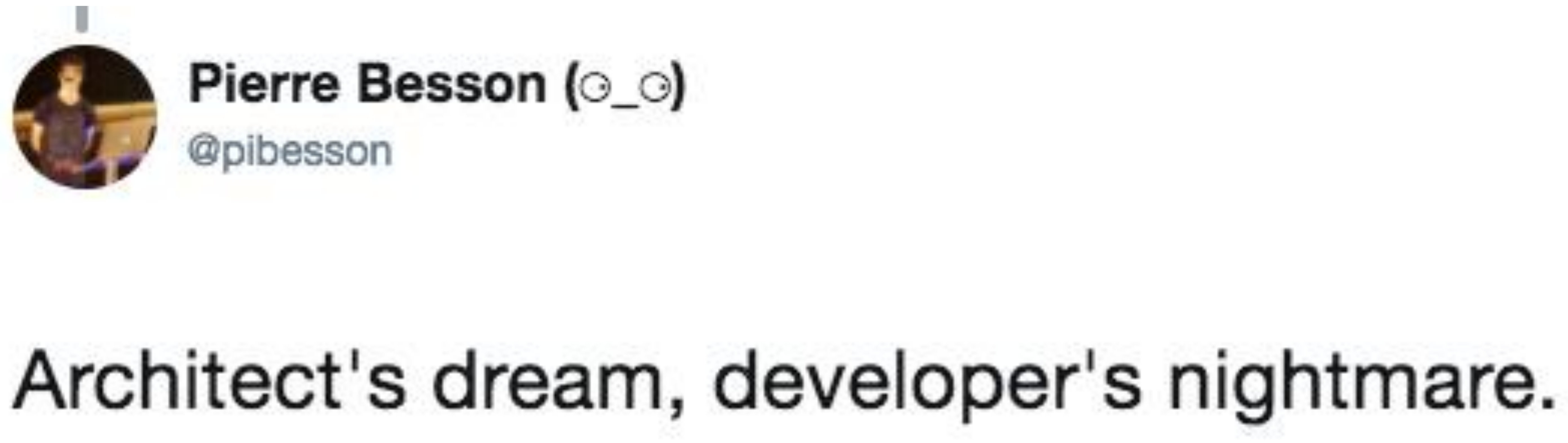
Λklı Rεgʋig
@aklireguig

'If you can't build monolith correctly why do you think putting network in the middle will help' by @simonbrown

What do others think?


Christophe Bornet
@cbornet_

What do others think?



Pierre Besson (◉_◉)
@pibesson

Architect's dream, developer's nightmare.

DevDays 2019

Why Microservices?

- Deliver new features quicker
- Smaller, agile teams
- Scale services independently
- Cloud

Microservices Challenges

- Scalability
- Cost Reduction
- Resilience
- Monitoring
- Security

What is MicroProfile?

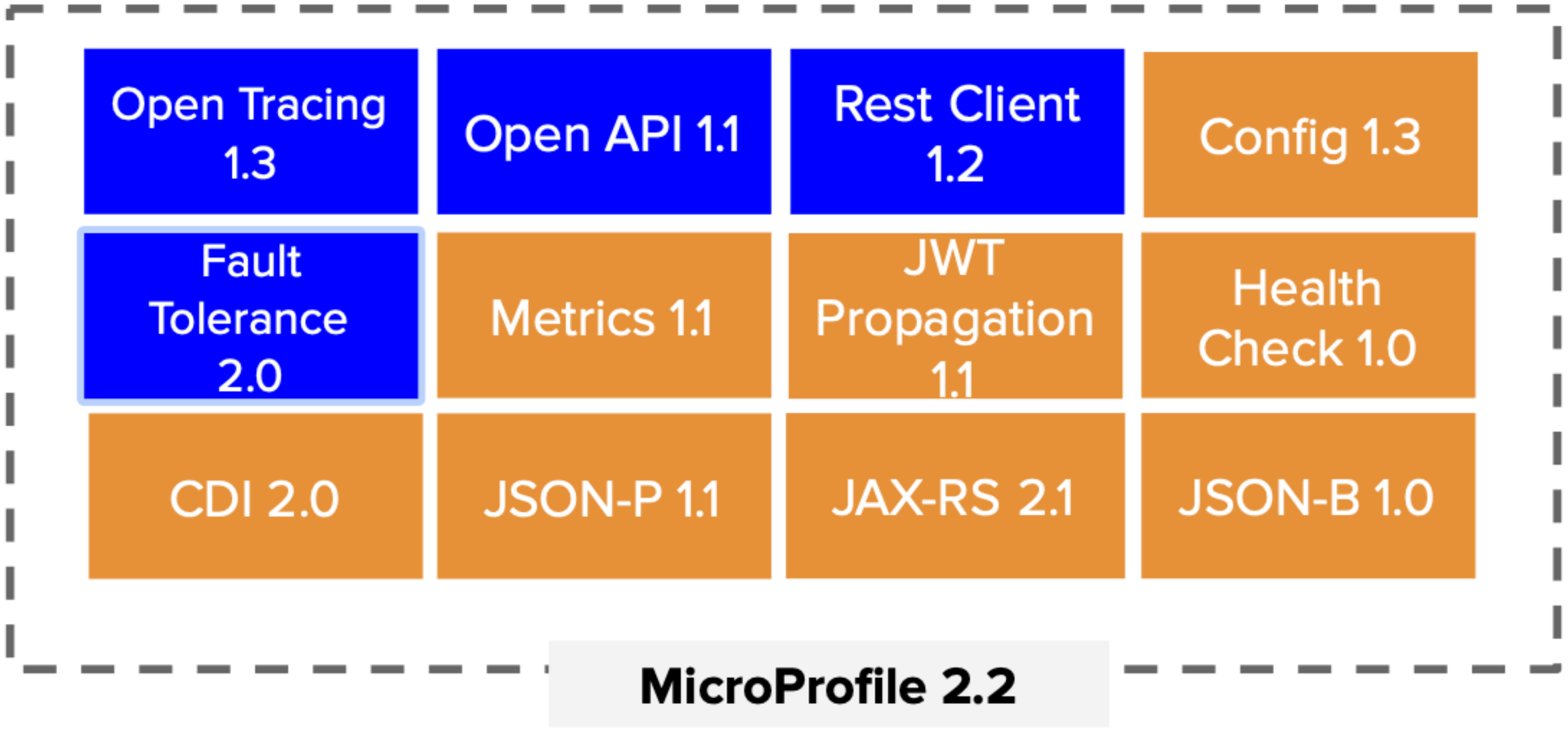
- http://microprofile.io/
- Enterprise Java for Microservices
- Open Source (Eclipse)
- 3 years old platform

What is MicroProfile?

- Initial version 1.0 with CDI, JAX-RS, JSON-P in Sep 2016
- Application portability across runtimes
- Currently at version 2.2 since Feb 2019
- Configuration, Fault Tolerance, JWT Propagation, Health Check, Metrics, Open Tracing, Open API and REST Client
- Reactive Streams released standalone

DevDays 2019

What is MicroProfile?

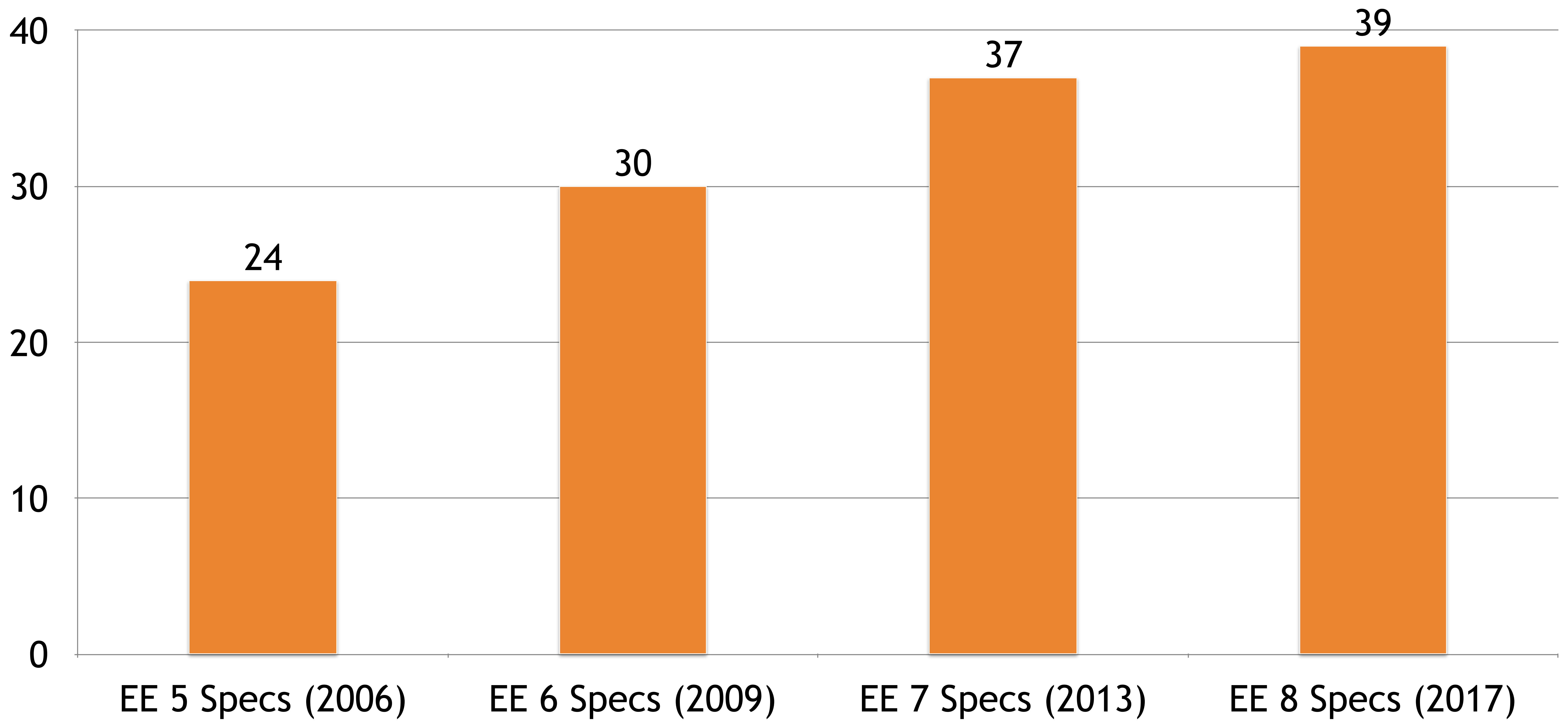


🟦 = New
🟦 = Updated
🟧 = No change from last release (MicroProfile 2.1)

Why MicroProfile?

- Slowdown in Java EE innovation
- Negative perception towards the technology
- Not prepared for Microservices development

Why MicroProfile?



Why MicroProfile?

- A set of “standards” were required to offer guidance in building Microservices in Java
- These need to evolve very quickly to adjust to the fast moving Microservices world
- A community of individuals, organisations and vendors collaborating to make this a reality

MicroProfile != EE (or Jakarta EE)

Who is involved in MicroProfile?

payara FUJITSU Tomitribe join the tribe

IBM redhat

LJC HAMMOCK SOUJava sociedade de usuários java Lightbend

Microsoft hazelcast ORACLE

DevDays 2019

MicroProfile Implementations

payara

redhat.

Tomitribe
join the tribe

FUJITSU

IBM

HAMMOCK

ORACLE

My first MicroProfile App

CDI

- Contexts and Dependency Injection
- Bean Lifecycle and Typesafe Injection
- Producers
- Interceptors
- Observers

JAX-RS

- RESTful Web Services
- Annotation based
- HTTP Centric

DevDays 2019

JSON-P

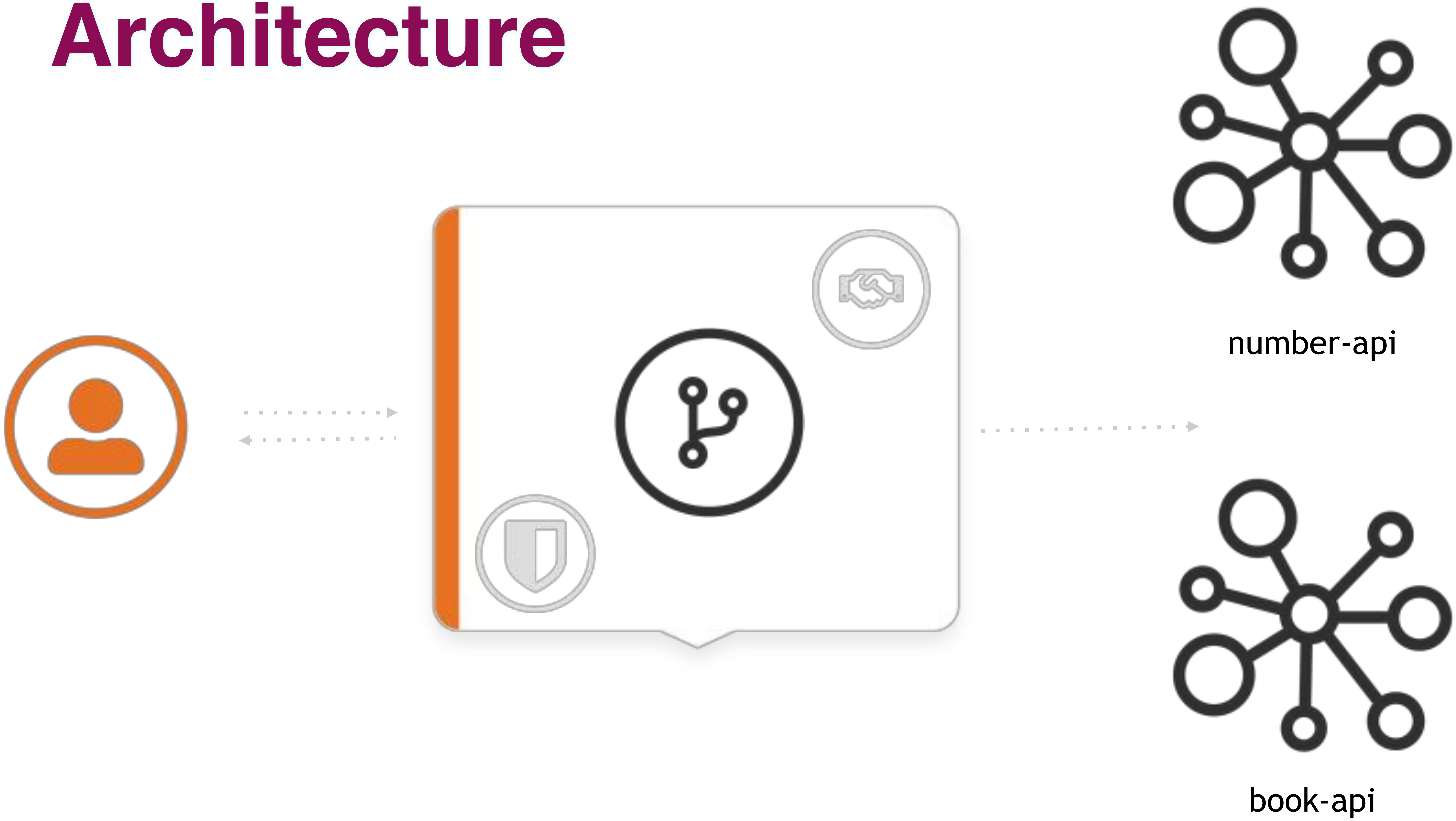
- Parse, Generate Transform and Query JSON
- Streaming API
- Object Model API

Putting it all together

```
@ApplicationScoped
@Path("books")
public class BookResource {
    @Inject
    private BookBean bookBean;

    @GET
    @Path("{id}")
    public Response findById(@PathParam("id") final Long id) {
        final Book book = bookBean.find(id);

        final JsonObjectBuilder builder =
                Json.createObjectBuilder()
                    .add("id", book.getId())
                    .add("name", book.getTitle());

     return Response.ok(builder.build().toString()).build();
    }
}
```

Evolving your MicroProfile App

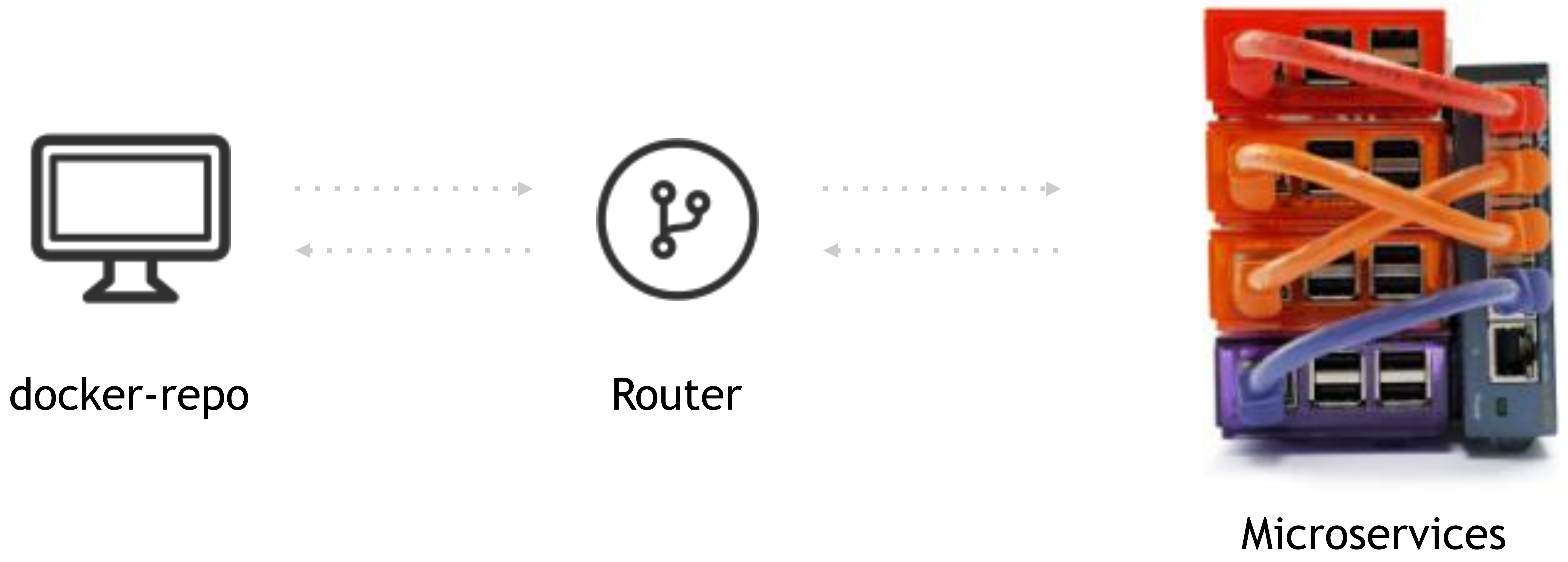
Architecture



Runtime

- Raspberry PI v3 (HypriotOS)
- Package everything with Docker
- Local Docker Registry
- Push Docker Images with Ansible

Deployment



MicroProfile Configuration

Configuration

- Applications need configuration based on their running environment
- It must be possible to change configuration without repacking the application
- Based on DeltaSpike Config, Apache Tamaya and Sabot

Configuration

- Standalone or in a CDI Container
- Default Values
- Supports the most common Java Type (including Optional)

Configuration

```
@Inject
@ConfigProperty(
   name="NUMBER_TARGET_API",
   defaultValue="http://localhost:8081/
   number-api/numbers/generate")
private String numberApiTargetUrl;
```

Configuration

```
Config config = ConfigProvider.getConfig();

final String url =
  config.getValue("NUMBER_TARGET_API",
    String.class);
```

Config Sources

- META-INF/microprofile-config.properties
- System properties
- Environment variables
- Pluggable to allow custom config sources and providers through the ServiceLoader mechanism

MicroProfile Fault Tolerance

Fault Tolerance

- Different strategies to guide the execution and result of some logic
- Inspired by Hystrix and Failsafe
- Supports Sync and Async execution

Fault Tolerance

- Depends on CDI
- Interceptor bindings
- Business method invocation
- Integrates with MP Config and Metrics

Timeout

- Prevents the execution from waiting forever
- Fail the execution if the timeout is hit
- Useful when calling other services

Retry

- Invoke the same operation again
- Specify criteria on when to retry
- Recover from network glitchs

Fallback

- Invoked when the original execution fails
- Specify when it fails
- Fallback to some other execution to prevent failures

Circuit Breaker

- Prevent repeated failures
- Fail fast
- Can be open, half-open or closed
- Protect services

Bulkhead

- Limit number of concurrent requests
- Access from multiple contexts
- Prevent failures from cascading

MicroProfile Healthchecks

Healthchecks

- Probe the state of a computer node
- Primary target cloud infrastructure
- Automated processes to maintain the state of nodes

Healthchecks

- Simple API to specify health status
- /health JAX-RS endpoint reporting server health
- Response status indicates if the health check passed
- Payload can include more detail

DevDays 2019

Healthchecks

```
@Health
@ApplicationScoped
public class CheckDiskSpace implements HealthCheck {
    public HealthCheckResponse call() {

    }
}
```

Healthchecks

GET /health

```
{
  "outcome": "UP",
  "checks": [{
    "name": "diskspace",
    "state": "UP",
    "data": {
      "key": "freebytes",
      "freebytes": "126000000000"
    }
  }]
}
```

DevDays 2019

Healthcheck Demo

MicroProfile Metrics

Metrics

- Monitor essential System Parameters
- Ensure reliable operation of software
- Monitoring endpoints to collect data

Metrics

- Accessible via REST interface
- /metrics/base for MP compliant servers
- /metrics/application for Application specific Metrics
- /metrics/vendor for Server specific metrics
- OPTIONS provides metadata, such as Unit of measure

GET /metrics/base

```
{
   "thread.count" : 33,
   "thread.max.count" : 47,
   "memory.maxHeap" : 3817863211,
   "memory.usedHeap" : 16859081,
   "memory.committedHeap" : 64703546
}
```

OPTIONS /metrics/base

```
{
  "fooVal": {
    "unit": "milliseconds",
    "type": "gauge",
    "description": "The average duration of foo requests during last 5
minutes",
    "displayName": "Duration of foo",
    "tags": "app=webshop"
  },
  "barVal": {
    "unit": "megabytes",
    "type": "gauge",
    "tags": "component=backend,app=webshop"
  }
}
```

Metrics

- @Counted
- @Metered
- @Timed
- @Gauge
- Histogram

Application Metrics

```
@GET
@Path("/{id}")
@Metered(name = "BookResource.findById_meter")
@Timed(name = "BookResource.findById_timer",
        unit = MetricUnits.MILLISECONDS,
        absolute = true)
public Response findById(@PathParam("id") final Long id) {
    ...
}
```

Meter

GET /metrics/application/{meter}

```
{
  "{meter}": {
    "count": 1,
    "fifteenMinRate": 0.2,
    "fiveMinRate": 0.2,
    "meanRate": 0.015384615384615385,
    "oneMinRate": 0.2,
    "unit": "per_second"
  }
}
```

Timed

GET /metrics/application/{timer}

```
{
  "{timer}": {
    "count": 1,
    "fifteenMinRate": 0.2,
    "fiveMinRate": 0.2,
    "max": 13644163,
    "mean": 13644163,
    "meanRate": 0.015384615384615385,
    "min": 13644163,
    "oneMinRate": 0.2,
    "p50": 13644163,
    "p75": 13644163,
    "p95": 13644163,
    "p98": 13644163,
    "p99": 13644163,
    "p999": 13644163,
    "stddev": 0
  }
}
```

DevDays 2019

Counted

GET /metrics/application/{counter}

```
{
  "{counter}": 156825
}
```

Gauge

```
public class BookResource {

    private AtomicLong booksAdded = new AtomicLong();

    @Gauge(name="booksadded", unit = MetricUnits.NONE)
    public long count() {
        return booksAdded.get();
    }
}
```

Gauge

GET /metrics/application/{gauge}

```
{
  "{gauge}": 156825
}
```

Histogram

```
public class BookResource {
    @Inject
    @Metric(name = "bookcount")
    private Histogram histo;

    public void update(long count) {
        histo.update(count);
    }
}
```

MicroProfile OpenTracing

OpenTracing

- Trace the flow of a request across services
- OpenTracing is a distributed tracing standard for applications
- Java Binding to OpenTracing Implementation

OpenTracing

- JAX-RS calls are traced
- Context propagated to services called with REST client
  - X-B3-TraceId, X-B3-ParentSpanId, X-B3-SpanId
- Traces are collected and pushed to a database

OpenTracing

- Operations are measured in Spans
- Spans are grouped together into Traces

Tracing CDI Calls

```java
@Traced
@GET
@Path(“/books")
public Response findBooks() { }
```

OpenTracing

```
@ApplicationScoped
public class BookBean {

    @Inject
    private Tracer tracer;

    public Book create(final Book book) {
        final Span activeSpan = tracer.activeSpan();
        final Tracer.SpanBuilder spanBuilder = tracer.buildSpan("create");

        if (activeSpan != null) {
            spanBuilder.asChildOf(activeSpan.context());
        }

        final Span span = spanBuilder.withTag("created", true).start();
        tracer.scopeManager().activate(span, true);

    // do work
        span.finish();

        return book;
    }
}
```

Zipkin Demo

MicroProfile JWT Propagation

DevDays 2019

Challenges in security

- Who is the caller?
- What can he do?
- How to propagate the security context?

JWT Propagation

- Security Tokens
- Most common: OAuth2, OpenID Connect JWT
- Lightweight way to propagate identities across different services

JWT Propagation

- Role based access control
- Keys (JWKS)
- Standard configuration (MP Config)

Goals

- Extract and verify the token
- Identify the caller
- Enforce authorization policies

What is a JWT?

- Pronounced “JOT”
- SAML like but less verbose
- Fancy JSON map
- BASE 64 URL encoded
- Digitally signed (RSA-SHA256, HMAC-SHA512, etc)
- Possibly encrypted
- Built-in expiration

DevDays 2019

```
@LoginConfig(authMethod = "MP-JWT")
public class ApplicationConfig extends Application {
    // let the server discover the endpoints
}

—
@Inject
private JsonWebToken jwtPrincipal;

@Context
private SecurityContext securityContext;

@Inject
@Claim("username")
private ClaimValue<String> username;

@Inject
@Claim("email")
private ClaimValue<String> email;

—
@RolesAllowed("create")
public Response create(final Book book, @Context UriInfo uriInfo) {
 ...
}
```

DevDays 2019

MicroProfile OpenAPI

Open API

- Java API for the OpenAPI v3 specification
- OpenAPI v3 is based on Swagger v2
- Annotations should be similar

Configuration

```
@GET
@Path("/{id}")
@Operation(summary = "Find a Book by Id")
@APIResponse(responseCode = "200",
  content = {@Content(schema =
   @Schema(implementation = Book.class))})
public Response findById(@PathParam("id") final Long id) {
    return bookBean.findById(id)
                .map(Response::ok)
                .orElse(status(NOT_FOUND))
                .build();
}
```

GET /openapi

```
paths:
  /books/{id}:
    get:
      parameters:
      - name: "id"
        required: true
        style: "simple"
        schema:
          readOnly: false
          deprecated: false
          description: "The id of the Book"
          writeOnly: false
      deprecated: false
      summary: "Find a Book by Id"
      responses:
        200:
          content:
            application/json:
              schema:
        …
          description: ""
      operationId: "findById"
```

OpenAPI Demo

MicroProfile REST Client

REST Client

- Microservices typically talk REST to other services
- Several JAX-RS implementations already support interface definition
- Consistent and easy to reuse

REST Client

- Type safe REST Service over HTTP
- Extends JAX-RS 2.0 API's
- More natural code style
- Similar to Feign

Configuration

```
@RegisterRestClient
@Path("/books")
public interface BookService {
  @GET
  @Path("/{id}")
  Response findById(@PathParam("id") final Long id);
}

@ApplicationScoped
public class BookStore {
     @Inject
     @RestClient
     private BookService client;
}
```

MicroProfile Starter

https://start.microprofile.io

DevDays 2019

Future

DevDays 2019

MicroProfile Roadmap

- MicroProfile 3.0 by June 2019
- Major updates to Health and Metrics
- GraphQl, Long Running Actions, Concurrency, Reactive Messaging, Event Data, Reactive Relational DB Access

Get Involved

Resources

- http://microprofile.io/
- http://tomee.apache.org
- https://github.com/radcortez/microprofile-samples

Thank you!