

# Developing reactive Microservices with Quarkus

Niklas Heidloff  
Developer Advocate, IBM  
@nheidloff

# Buzzword Bingo

Reactive Manifesto

Reactive Systems

Reactive Programming

Functional Programming

Asynchronous Programming

Reactive Streams

Reactive Operators



# Let's make it concrete

Reactive Web Application

Reactive REST Endpoints

# Reactive Web Application

## Articles

|  Title |  Author |  Twitter |  Blog |
|---|--|---|--|
| <a href="#">Debugging Microservices running in Kubernetes</a>                         | Niklas Heidloff  | <a href="#">@nheidloff</a>  | <a href="#">Blog</a>   |
| <a href="#">Dockerizing Java MicroProfile Applications</a>                            | Niklas Heidloff  | <a href="#">@nheidloff</a>  | <a href="#">Blog</a>   |
| <a href="#">Install Istio and Kiali on IBM Cloud or Minikube</a>                      | Harald Uebele  | <a href="#">@harald_u</a>   | <a href="#">Blog</a>   |
| <a href="#">Three awesome TensorFlow.js Models for Visual Recognition</a>             | Niklas Heidloff  | <a href="#">@nheidloff</a>  | <a href="#">Blog</a>   |
| <a href="#">Blue Cloud Mirror Architecture Diagrams</a>                               | Niklas Heidloff  | <a href="#">@nheidloff</a>  | <a href="#">Blog</a>   |



reactive — -bash — 135x8

Niklass-MBP:reactive nheidloff\$

# Reactive REST Endpoints

HTTP Request.jmx (/Users/nheidloff/Desktop/reactive/apache-jmeter-5.2.1/bin/HTTP Request.jmx) - Apache JMeter (5.2.1)

00:00:45

Test Plan

- Thread Group
  - HTTP Request
    - Summary Report – Reactive Endpoint
  - HTTP Header Manager
  - Response Time Graph
  - View Results Tree
  - View Results in Table

Summary Report

Name: Summary Report – Reactive Endpoint

Comments:

Write results to file / Read from file

Filename  Browse... Log/Display Only:  Errors  Successes Configure

| Label       | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB... | Sent KB/sec | Avg. Bytes |
|-------------|-----------|---------|-----|-----|-----------|---------|------------|----------------|-------------|------------|
| HTTP Req... | 30000     | 150     | 5   | 774 | 70.11     | 0.00%   | 660.7/sec  | 46.46          | 101.94      | 72.00      |
| TOTAL       | 30000     | 150     | 5   | 774 | 70.11     | 0.00%   | 660.7/sec  | 46.46          | 101.94      | 72.00      |

HTTP Request 1.jmx (/Users/nheidloff/Desktop/reactive/apache-jmeter-5.2.1/bin/HTTP Request 1.jmx) - Apache JMeter (5.2.1)

00:01:18

Test Plan

- Thread Group
  - HTTP Request
    - Summary Report – Synchronous Endpoint
  - HTTP Header Manager
  - Response Time Graph
  - View Results Tree
  - View Results in Table

Summary Report

Name: Summary Report – Synchronous Endpoint

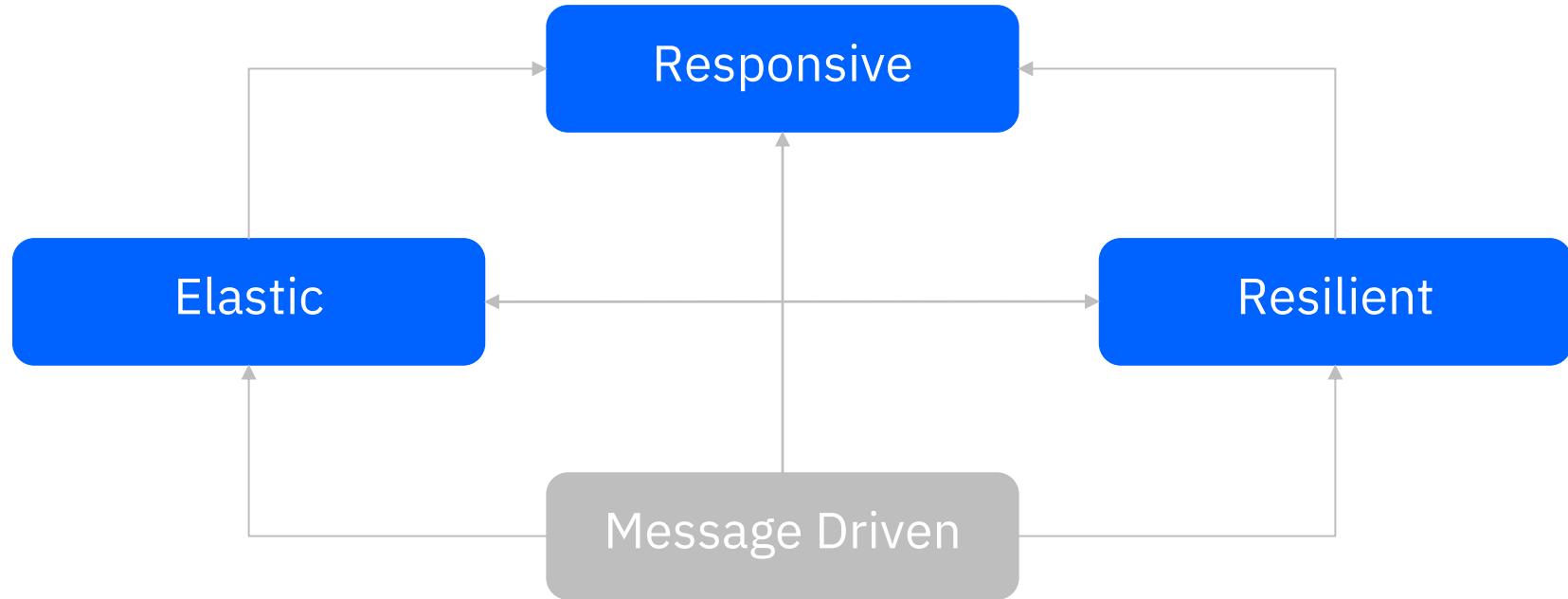
Comments:

Write results to file / Read from file

Filename  Browse... Log/Display Only:  Errors  Successes Configure

| Label       | # Samples | Average | Min | Max  | Std. Dev. | Error % | Throughput | Received KB... | Sent KB/sec | Avg. Bytes |
|-------------|-----------|---------|-----|------|-----------|---------|------------|----------------|-------------|------------|
| HTTP Req... | 30000     | 258     | 1   | 1399 | 83.96     | 0.00%   | 383.2/sec  | 760.78         | 59.13       | 2033.00    |
| TOTAL       | 30000     | 258     | 1   | 1399 | 83.96     | 0.00%   | 383.2/sec  | 760.78         | 59.13       | 2033.00    |

# Reactive Manifesto



# Reactive Systems

!=

# Reactive Programming

# Reactive Programming is ...

```
@GET
@Path("/articles")
@Produces(MediaType.APPLICATION_JSON)
public CompletionStage<Response> getArticlesReactive(int amount) {
    return articleService.getArticlesReactive(amount)
        .thenApply(articles -> convertArticlesToJsonArray(articles))
        .thenApply(jsonArray -> Response.ok(jsonArray).build())
        .exceptionally(throwable -> {
            if (throwable.getCause().toString().equals(InvalidInputParameter.class.getName()))
                return Response.status(Response.Status.BAD_REQUEST).build();
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
        });
}
```

# Reactive Programming is ‘unusual’

```
@GET  
@Path("/articles")  
@Produces(MediaType.APPLICATION_JSON)  
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build())  
        .exceptionally(throwable -> {  
            if (throwable.getCause().toString().equals(InvalidInputParameter.class.getName()))  
                return Response.status(Response.Status.BAD_REQUEST).build();  
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();  
        });  
}
```

# Reactive Programming is ‘unusual’

```
@GET  
@Path("/articles")  
@Produces(MediaType.APPLICATION_JSON)  
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build())  
        .exceptionally(throwable -> {  
            if (throwable.getCause().toString().equals(InvalidInputParameter.class.getName()))  
                return Response.status(Response.Status.BAD_REQUEST).build();  
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();  
        });  
}
```

# Reactive Programming is ‘unusual’

```
@GET  
@Path("/articles")  
@Produces(MediaType.APPLICATION_JSON)  
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build())  
        .exceptionally(throwable -> {  
            if (throwable.getCause().toString().equals(InvalidInputParameter.class.getName()))  
                return Response.status(Response.Status.BAD_REQUEST).build();  
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();  
        });  
}
```

# Javadoc to the Rescue?

```
public <U> CompletionStage<U> thenApply(Function<? super T, ? extends U> fn);

    /**
     * Returns a new CompletionStage that, when this stage completes normally, is executed
     * with this stage's result as the argument to the supplied function.
     *
     * This method is analogous to Optional.map and Stream.map.
     *
     * See the CompletionStage documentation for rules covering exceptional completion.
     *
     * 


     *   - Type Parameters:
     *

     *           - <U> the function's return type

     *
     *

     *   - Parameters:
     *

     *           - fn: the function to apply to the stage's result

     *
     *

     * 

     */
    public CompletionStage<Response> thenApply(CompletionStage<Response> java.util.concurrent.CompletionStage.thenApply(Function<? super JsonArray, ? extends Response> fn)) {
        return thenApply(jsonArray -> {
            return Response.ok(jsonArray).build();
        }).exceptionally(throwable -> {
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
        }).whenComplete((response, throwable) -> {
            future.complete(response);
        });
    }

    /**
     * Returns a new CompletionStage that, when this stage completes normally, is executed
     * with this stage's result as the argument to the supplied function.
     *
     * This method is analogous to Optional.map and Stream.map.
     *
     * See the CompletionStage documentation for rules covering exceptional completion.
     *
     * 


     *   - Type Parameters:
     *

     *           - <U> the function's return type

     *
     *

     *   - Parameters:
     *

     *           - fn: the function to apply to the stage's result

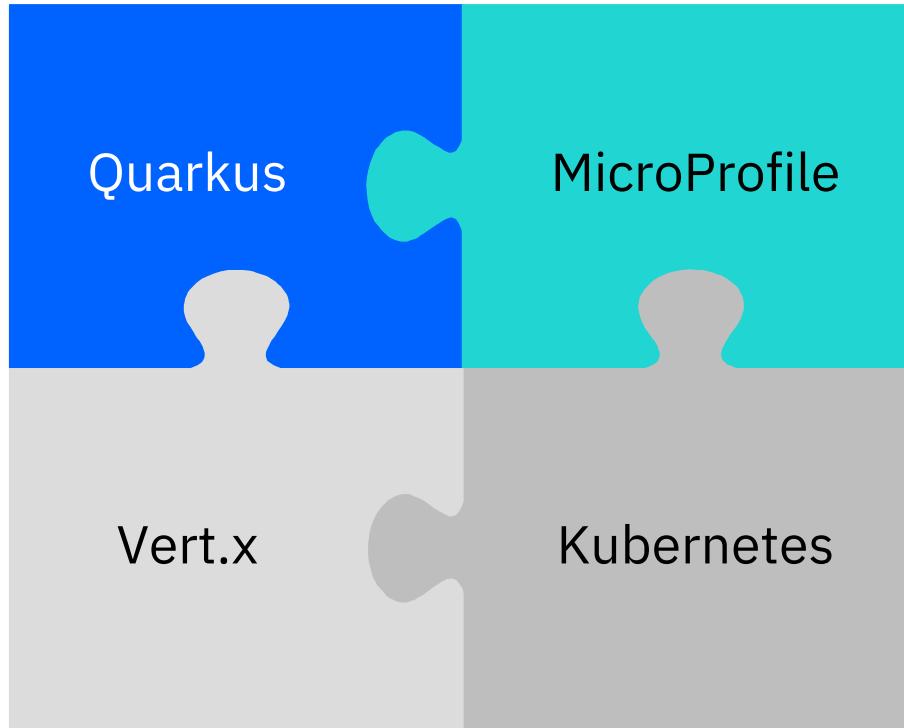
     *
     *

     * 

     */
    public CompletionStage<U> thenApply(Function<? super T, ? extends U> fn) {
        return thenApply(thenApply((JsonArray) -> {
            return Response.ok((JsonArray) jsonArray).build();
        }));
    }
}
```

Reactive programming is  
extremely powerful, but  
not the right tool for  
all jobs!

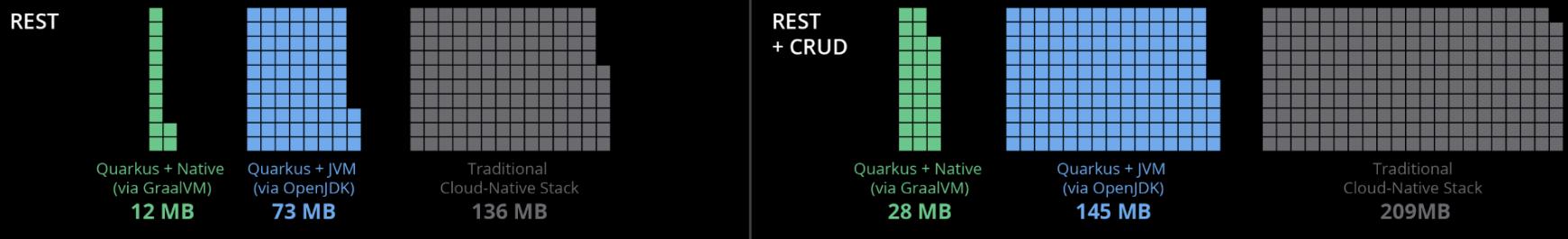
# Technologies to build reactive Applications



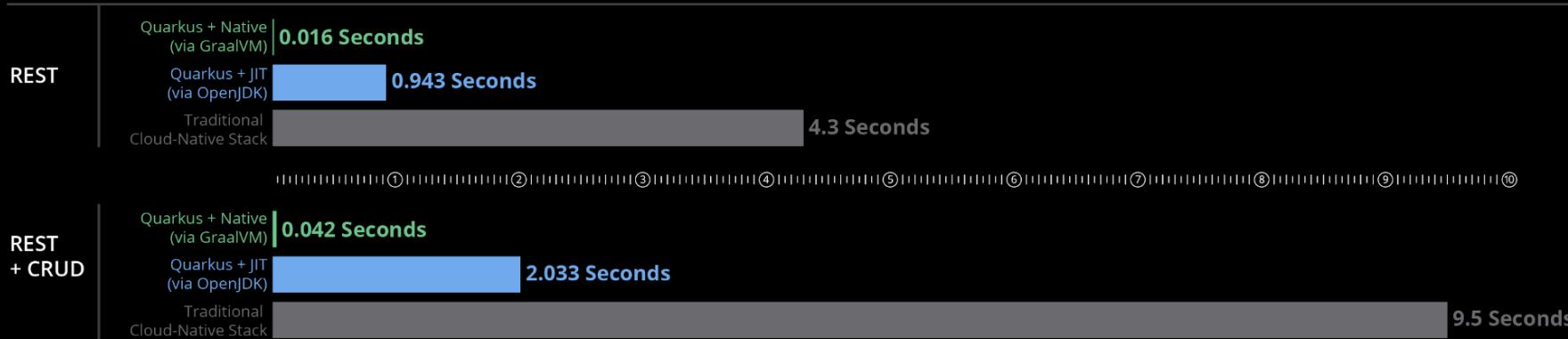
# Quarkus – Supersonic Subatomic Java

## Memory (RSS) in Megabytes\*

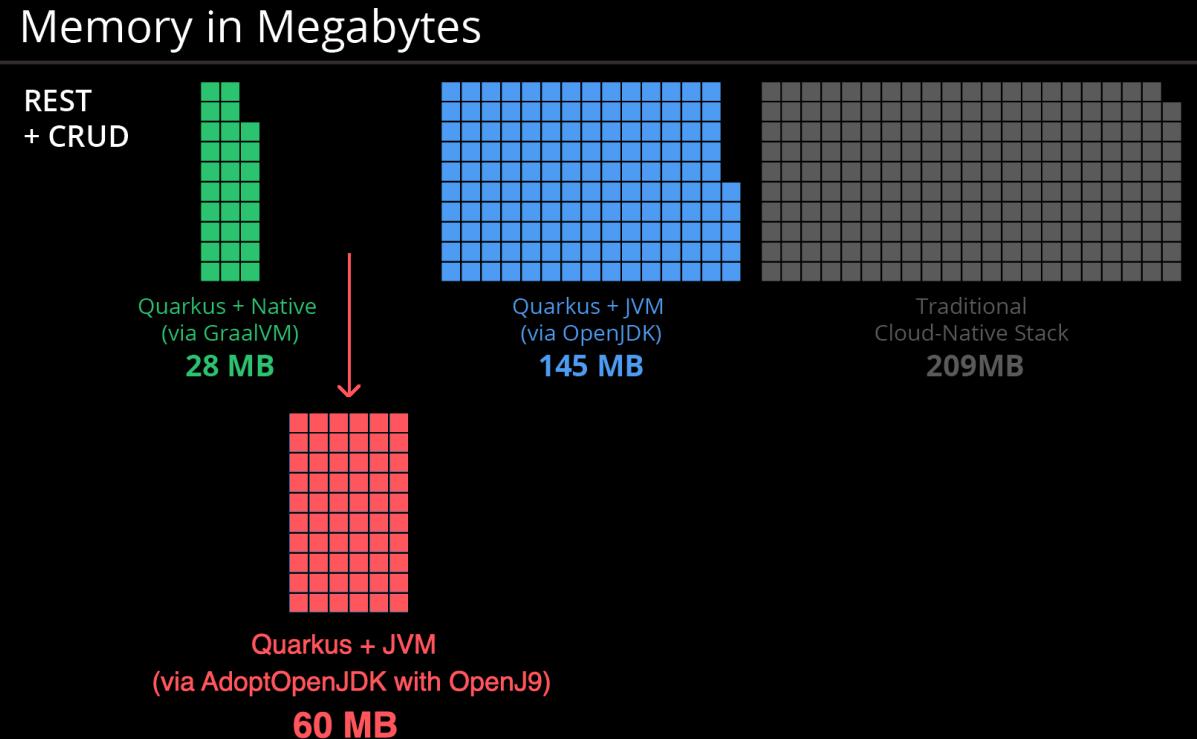
\*Tested on a single-core machine



## BOOT + First Response Time



# Quarkus using OpenJ9



# “Optimizing Enterprise Java for a Microservices Architecture.”

“[...] by innovating [...] with a  
goal of standardization.”

[micrometer.io](https://micrometer.io)



“Eclipse Vert.x is a tool-kit for building reactive applications on the JVM.”

“Eclipse Vert.x is event driven and non blocking [...] and lets your app scale with minimal hardware.”

[vertx.io](http://vertx.io)

@nheidloff



#IBMDeveloper [github.com/ibm/cloud-native-starter](https://github.com/ibm/cloud-native-starter)

“Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.”

[kubernetes.io](https://kubernetes.io)



# kubernetes

@nheidloff

#IBMDeveloper [github.com/ibm/cloud-native-starter](https://github.com/ibm/cloud-native-starter)

# Example Application

## Cloud Native Starter

### Articles



Title

[Debugging Microservices running in Kubernetes](#)

[Dockerizing Java MicroProfile Applications](#)

[Install Istio and Kiali on IBM Cloud or Minikube](#)

[Three awesome TensorFlow.js Models for Visual Recognition](#)

[Blue Cloud Mirror Architecture Diagrams](#)



Author

Niklas Heidloff

Niklas Heidloff

Harald Uebele

Niklas Heidloff

Niklas Heidloff



Twitter

[@nheidloff](#)

[@nheidloff](#)

[@harald\\_u](#)

[@nheidloff](#)

[@nheidloff](#)



Blog

[Blog](#)

[Blog](#)

[Blog](#)

[Blog](#)

[Blog](#)

# Architecture

Clients



Web-App



API Client

Kubernetes

Microservices



Web-App



Web-API



Authors

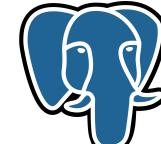


Articles

Infrastructure Components



Kafka



Postgres

# Reactive Web Application

## Articles

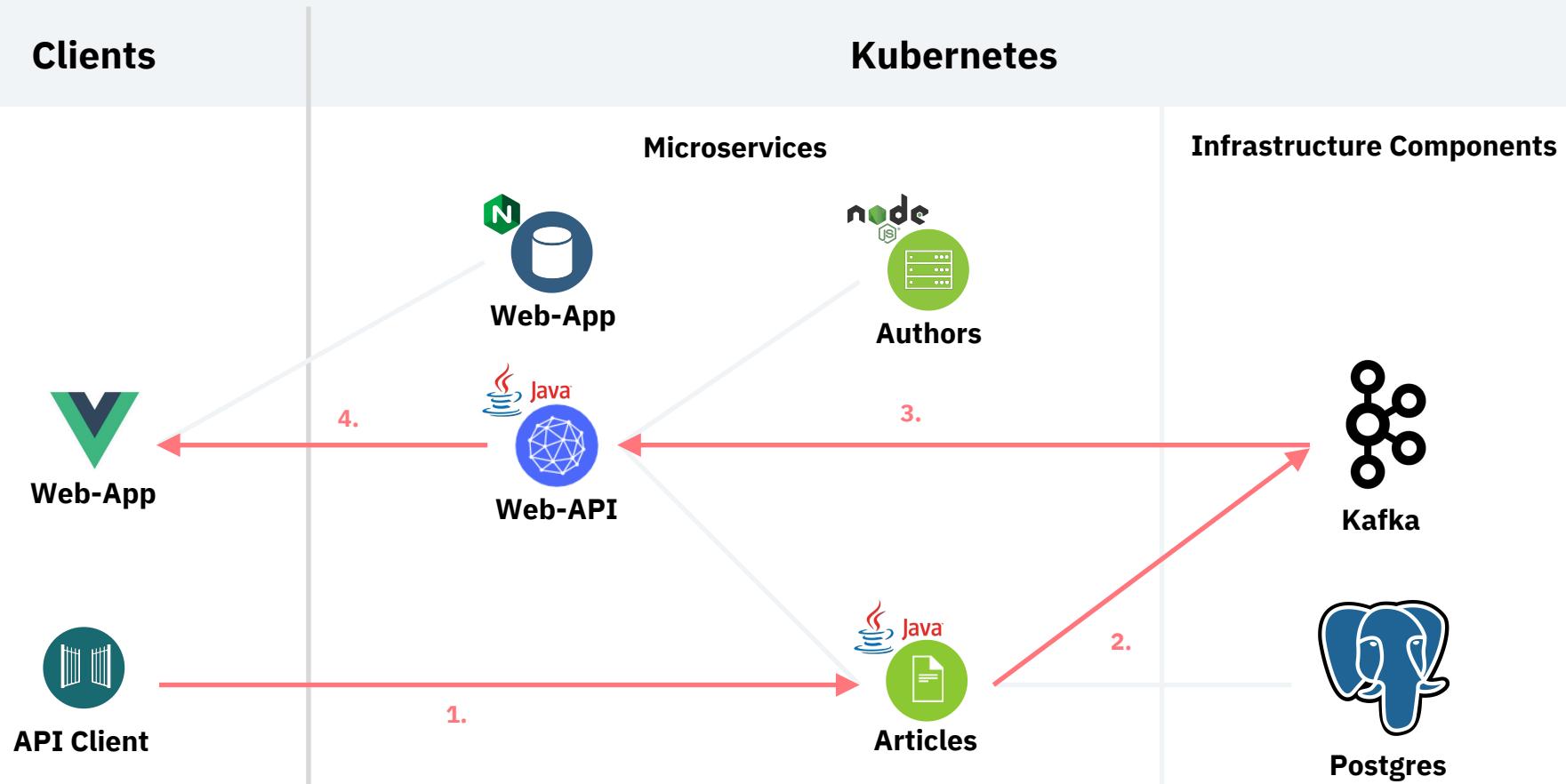
|  Title |  Author |  Twitter |  Blog |
|---|--|---|--|
| <a href="#">Debugging Microservices running in Kubernetes</a>                         | Niklas Heidloff  | <a href="#">@nheidloff</a>  | <a href="#">Blog</a>   |
| <a href="#">Dockerizing Java MicroProfile Applications</a>                            | Niklas Heidloff  | <a href="#">@nheidloff</a>  | <a href="#">Blog</a>   |
| <a href="#">Install Istio and Kiali on IBM Cloud or Minikube</a>                      | Harald Uebele  | <a href="#">@harald_u</a>   | <a href="#">Blog</a>   |
| <a href="#">Three awesome TensorFlow.js Models for Visual Recognition</a>             | Niklas Heidloff  | <a href="#">@nheidloff</a>  | <a href="#">Blog</a>   |
| <a href="#">Blue Cloud Mirror Architecture Diagrams</a>                               | Niklas Heidloff  | <a href="#">@nheidloff</a>  | <a href="#">Blog</a>   |



reactive — -bash — 135x8

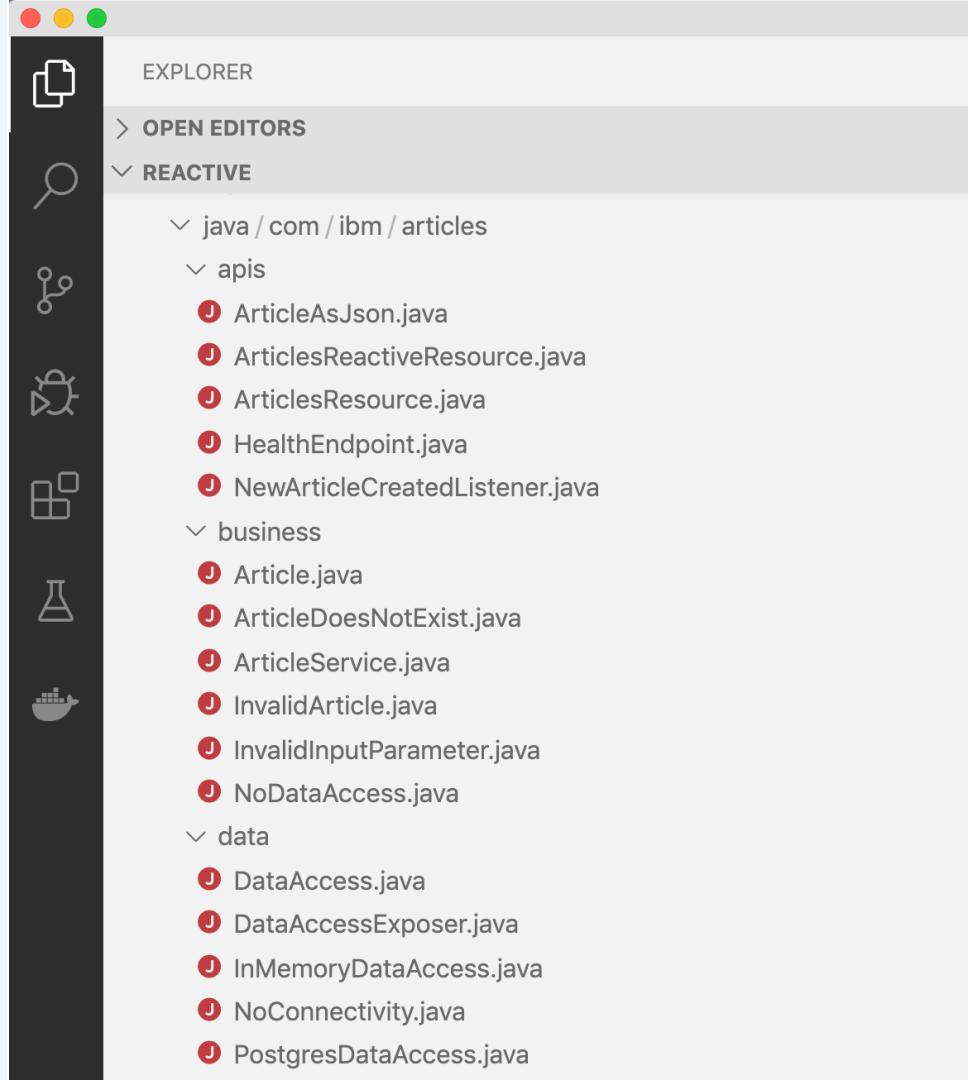
Niklass-MBP:reactive nheidloff\$

# Notifications for Web Applications



# Clean Architecture

1. APIs  
REST endpoints and messaging
2. Business  
Logic of services and entities
3. Data  
Access to databases or other services



# Vert.x Event Bus

```
import io.vertx.axle.core.eventbus.EventBus;

public class ArticleService {

    @Inject
    EventBus bus;

    private void sendMessageToKafka(Article article) {
        bus.publish("com.ibm.articles.apis.NewArticleCreatedListener", article.id);
    }
}
```

```
import io.quarkus.vertx.ConsumeEvent;

public class NewArticleCreatedListener {

    @ConsumeEvent
    public void sendMessageToKafka(String articleId) {
        // run logic
    }
}
```

# Vert.x Event Bus

```
import io.vertx.axle.core.eventbus.EventBus;

public class ArticleService {

    @Inject
    EventBus bus;

    private void sendMessageToKafka(Article article) {
        bus.publish("com.ibm.articles.apis.NewArticleCreatedListener", article.id);
    }
}
```

```
import io.quarkus.vertx.ConsumeEvent;

public class NewArticleCreatedListener {

    @ConsumeEvent
    public void sendMessageToKafka(String articleId) {
        // run logic
    }
}
```

# Vert.x Event Bus

```
import io.vertx.axle.core.eventbus.EventBus;

public class ArticleService {

    @Inject
    EventBus bus;

    private void sendMessageToKafka(Article article) {
        bus.publish("com.ibm.articles.apis.NewArticleCreatedListener", article.id);
    }
}
```

```
import io.quarkus.vertx.ConsumeEvent;

public class NewArticleCreatedListener {

    @ConsumeEvent
    public void sendMessageToKafka(String articleId) {
        // run logic
    }
}
```

# Vert.x Event Bus

```
import io.vertx.axle.core.eventbus.EventBus;

public class ArticleService {

    @Inject
    EventBus bus;

    private void sendMessageToKafka(Article article) {
        bus.publish("com.ibm.articles.apis.NewArticleCreatedListener", article.id);
    }
}
```

```
import io.quarkus.vertx.ConsumeEvent;

public class NewArticleCreatedListener {

    @ConsumeEvent
    public void sendMessageToKafka(String articleId) {
        // run logic
    }
}
```

# Kafka API

```
@Inject
io.vertx.core.Vertx vertx;

private io.vertx.kafka.client.producer.KafkaProducer<String, String> producer;

@PostConstruct
void initKafkaClient() {
    Map<String, String> config = new HashMap<>();
    config.put("bootstrap.servers", kafkaBootstrapServer);
    producer = KafkaProducer.create(vertx, config);
}

@ConsumeEvent
public void sendMessageToKafka(String articleId) {
    try {
        io.vertx.kafka.client.producer.KafkaProducerRecord<String, String> record =
            KafkaProducerRecord.create("new-article-created", articleId);
        producer.write(record, done -> System.out.println("Kafka message sent"));
    } catch (Exception e) {
    }
}
```

# Kafka API

```
@Inject
io.vertx.core.Vertx vertx;

private io.vertx.kafka.client.producer.KafkaProducer<String, String> producer;

@PostConstruct
void initKafkaClient() {
    Map<String, String> config = new HashMap<>();
    config.put("bootstrap.servers", kafkaBootstrapServer);
    producer = KafkaProducer.create(vertx, config);
}

@ConsumeEvent
public void sendMessageToKafka(String articleId) {
    try {
        io.vertx.kafka.client.producer.KafkaProducerRecord<String, String> record =
            KafkaProducerRecord.create("new-article-created", articleId);
        producer.write(record, done -> System.out.println("Kafka message sent"));
    } catch (Exception e) {
    }
}
```

# Kafka API

```
@Inject
io.vertx.core.Vertx vertx;

private io.vertx.kafka.client.producer.KafkaProducer<String, String> producer;

@PostConstruct
void initKafkaClient() {
    Map<String, String> config = new HashMap<>();
    config.put("bootstrap.servers", kafkaBootstrapServer);
    producer = KafkaProducer.create(vertx, config);
}

@ConsumeEvent
public void sendMessageToKafka(String articleId) {
    try {
        io.vertx.kafka.client.producer.KafkaProducerRecord<String, String> record =
            KafkaProducerRecord.create("new-article-created", articleId);
        producer.write(record, done -> System.out.println("Kafka message sent"));
    } catch (Exception e) {
    }
}
```

# MicroProfile Reactive Messaging

```
import org.eclipse.microprofile.reactive.messaging.Incoming;
import org.eclipse.microprofile.reactive.messaging.Outgoing;
import io.smallrye.reactive.messaging.annotations.Broadcast;

public class NewArticleListener {

    @Incoming("new-article-created")
    @Outgoing("stream-new-article")
    @Broadcast
    public String process(String articleId) {
        System.out.println("Kafka message received: new-article-created - " + articleId);
        return articleId;
    }
}
```

# MicroProfile Reactive Messaging

```
import org.eclipse.microprofile.reactive.messaging.Incoming;
import org.eclipse.microprofile.reactive.messaging.Outgoing;
import io.smallrye.reactive.messaging.annotations.Broadcast;

public class NewArticleListener {

    @Incoming("new-article-created")
    @Outgoing("stream-new-article")
    @Broadcast
    public String process(String articleId) {
        System.out.println("Kafka message received: new-article-created - " + articleId);
        return articleId;
    }
}
```

Reactive Streams is an initiative to provide a standard for asynchronous stream processing [...] aimed at runtime environments (JVM and JavaScript)."

[reactive-streams.org](http://reactive-streams.org)

Components:

1. Subscriber
2. Publisher
3. Processor

Java:

- JDK9: `java.util.concurrent.Flow`
- MicroProfile: `org.reactivestreams`

# MicroProfile Reactive Messaging

```
import org.eclipse.microprofile.reactive.messaging.Incoming;
import org.eclipse.microprofile.reactive.messaging.Outgoing;
import io.smallrye.reactive.messaging.annotations.Broadcast;

public class NewArticleListener {
    @Incoming("new-article-created")  
    @Outgoing("stream-new-article")  
    @Broadcast  
    public String process(String articleId) {  
        System.out.println("Kafka message received: new-article-created - " + articleId);  
        return articleId;  
    }
}
```

Subscriber

Publisher

# Server Sent Events

```
import org.reactivestreams.Publisher;
import io.smallrye.reactive.messaging.annotations.Channel;
import org.jboss.resteasy.annotations.SseElementType;

public class NewArticlesStreamResource {

    @Inject
    @Channel("stream-new-article") Publisher<String> newArticles;

    @GET
    @Path("/server-sent-events")
    @Produces(MediaType.SERVER_SENT_EVENTS)
    @SseElementType("text/plain")
    public Publisher<String> stream() {
        return newArticles;
    }
}
```

```
let url = this.$store.state.endpoints.api +
| "server-sent-events";
this.readArticles();
let source = new EventSource(url);
let that = this;
source.onmessage = function (event) {
| that.readArticles();
};
```

# Server Sent Events

```
import org.reactivestreams.Publisher;
import io.smallrye.reactive.messaging.annotations.Channel;
import org.jboss.resteasy.annotations.SseElementType;

public class NewArticlesStreamResource {
    ...
    @Channel("stream-new-article") Publisher<String> newArticles;

    @GET
    @Path("/server-sent-events")
    @Produces(MediaType.SERVER_SENT_EVENTS)
    @SseElementType("text/plain")
    public Publisher<String> stream() {
        return newArticles;
    }
}
```

```
let url = this.$store.state.endpoints.api +
  "server-sent-events";
this.readArticles();
let source = new EventSource(url);
let that = this;
source.onmessage = function (event) {
  that.readArticles();
};
```

# Server Sent Events

```
import org.reactivestreams.Publisher;
import io.smallrye.reactive.messaging.annotations.Channel;
import org.jboss.resteasy.annotations.SseElementType;

public class NewArticlesStreamResource {

    @Inject
    @Channel("stream-new-article") Publisher<String> newArticles;

    @GET
    @Path("/server-sent-events")
    @Produces(MediaType.SERVER_SENT_EVENTS)
    @SseElementType("text/plain")
    public Publisher<String> stream() {
        return newArticles;
    }
}
```

```
let url = this.$store.state.endpoints.api +
  "server-sent-events";
this.readArticles();
let source = new EventSource(url);
let that = this;
source.onmessage = function (event) {
  that.readArticles();
};
```

# Server Sent Events

```
import org.reactivestreams.Publisher;
import io.smallrye.reactive.messaging.annotations.Channel;
import org.jboss.resteasy.annotations.SseElementType;

public class NewArticlesStreamResource {

    @Inject
    @Channel("stream-new-article") Publisher<String> newArticles;

    @GET
    @Path("/server-sent-events")
    @Produces(MediaType.SERVER_SENT_EVENTS)
    @SseElementType("text/plain")
    public Publisher<String> stream() {
        return newArticles;
    }
}
```

```
let url = this.$store.state.endpoints.api +
  "server-sent-events";
this.readArticles();

let source = new EventSource(url);
let that = this;
source.onmessage = function (event) {
  that.readArticles(),
};

}
```

# Reactive REST Endpoints

HTTP Request.jmx (/Users/nheidloff/Desktop/reactive/apache-jmeter-5.2.1/bin/HTTP Request.jmx) - Apache JMeter (5.2.1)

00:00:45

Test Plan

- Thread Group
  - HTTP Request
    - Summary Report – Reactive Endpoint
  - HTTP Header Manager
  - Response Time Graph
  - View Results Tree
  - View Results in Table

Summary Report

Name: Summary Report – Reactive Endpoint

Comments:

Write results to file / Read from file

Filename  Browse... Log/Display Only:  Errors  Successes Configure

| Label       | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB... | Sent KB/sec | Avg. Bytes |
|-------------|-----------|---------|-----|-----|-----------|---------|------------|----------------|-------------|------------|
| HTTP Req... | 30000     | 150     | 5   | 774 | 70.11     | 0.00%   | 660.7/sec  | 46.46          | 101.94      | 72.00      |
| TOTAL       | 30000     | 150     | 5   | 774 | 70.11     | 0.00%   | 660.7/sec  | 46.46          | 101.94      | 72.00      |

HTTP Request 1.jmx (/Users/nheidloff/Desktop/reactive/apache-jmeter-5.2.1/bin/HTTP Request 1.jmx) - Apache JMeter (5.2.1)

00:01:18

Test Plan

- Thread Group
  - HTTP Request
    - Summary Report – Synchronous Endpoint
  - HTTP Header Manager
  - Response Time Graph
  - View Results Tree
  - View Results in Table

Summary Report

Name: Summary Report – Synchronous Endpoint

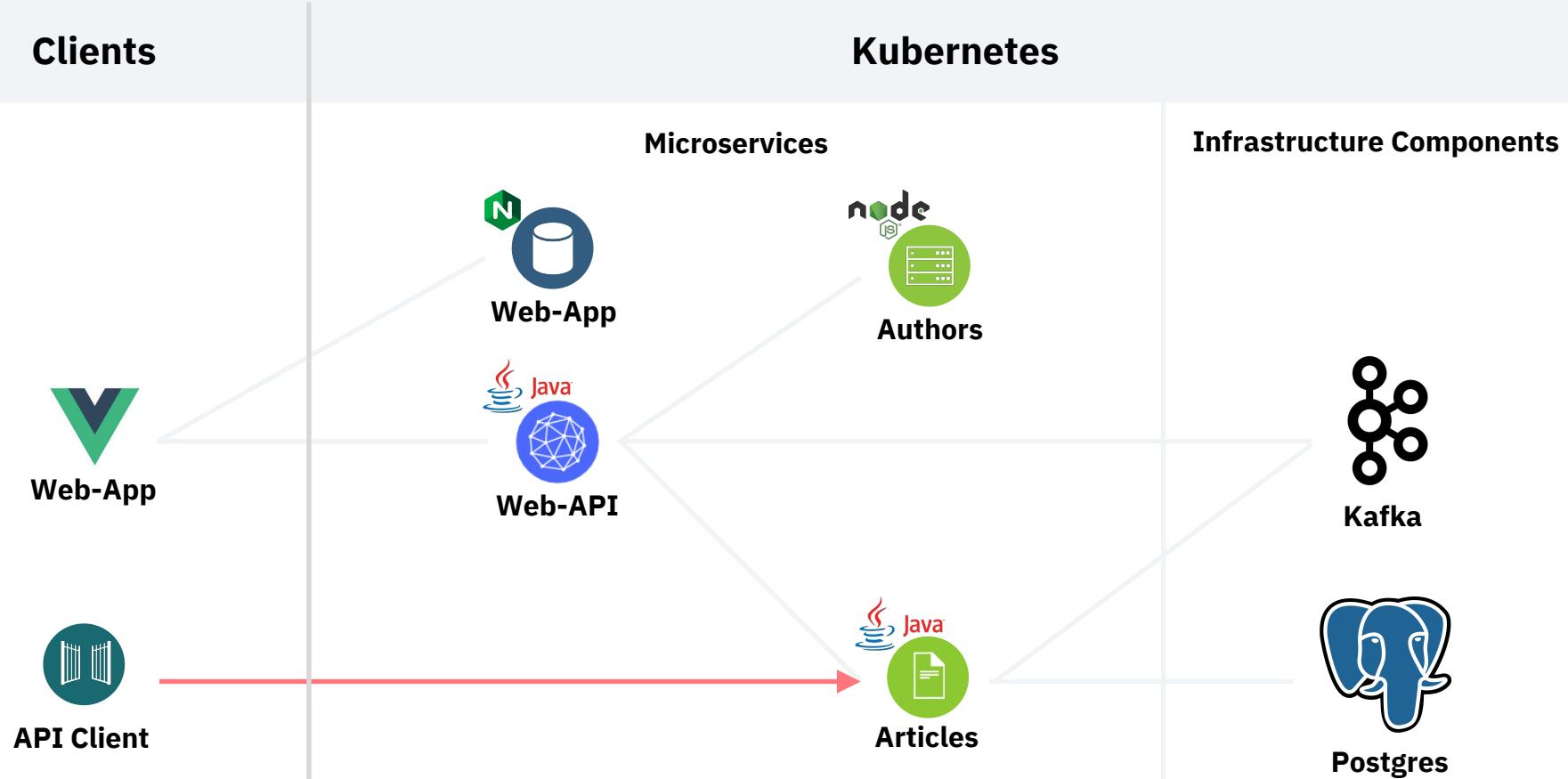
Comments:

Write results to file / Read from file

Filename  Browse... Log/Display Only:  Errors  Successes Configure

| Label       | # Samples | Average | Min | Max  | Std. Dev. | Error % | Throughput | Received KB... | Sent KB/sec | Avg. Bytes |
|-------------|-----------|---------|-----|------|-----------|---------|------------|----------------|-------------|------------|
| HTTP Req... | 30000     | 258     | 1   | 1399 | 83.96     | 0.00%   | 383.2/sec  | 760.78         | 59.13       | 2033.00    |
| TOTAL       | 30000     | 258     | 1   | 1399 | 83.96     | 0.00%   | 383.2/sec  | 760.78         | 59.13       | 2033.00    |

# Reactive REST Endpoint



# Reactive REST Endpoint

```
@GET
@Path("/articles")
@Produces(MediaType.APPLICATION_JSON)
public CompletionStage<Response> getArticlesReactive(int amount) {
    return articleService.getArticlesReactive(amount)
        .thenApply(articles -> convertArticlesToJsonArray(articles))
        .thenApply(jsonArray -> Response.ok(jsonArray).build())
        .exceptionally(throwable -> {
            if (throwable.getCause().toString().equals(InvalidInputParameter.class.getName()))
                return Response.status(Response.Status.BAD_REQUEST).build();
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
        });
}
```

# Completion Stage

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build());  
}
```

# Completion Stage

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleservice.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build());  
}
```

# Completion Stage and Completable Future

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
    CompletableFuture<Response> completableFuture = new CompletableFuture<Response>();  
  
    articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build())  
        .whenComplete((response, throwable) -> {  
            completableFuture.complete(response);  
        });  
  
    return completableFuture;  
}
```

# Completion Stage and Completable Future

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
  
    CompletableFuture<Response> completableFuture = new CompletableFuture<Response>();  
  
    articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build())  
        .whenComplete((response, throwable) -> {  
            completableFuture.complete(response);  
        });  
  
    return completableFuture;  
}
```

# Chained Completion Stages

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build());  
}
```

# Chained Completion Stages

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build());  
}
```

# Chained Completion Stages

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
  
    CompletionStage<List<Article>> completionStageArticles = articleService.getArticlesReactive(amount);  
    CompletionStage<Response> output;  
  
    output = completionStageArticles  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build());  
  
    return output;  
}
```

# Chained Completion Stages

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
  
    CompletionStage<List<Article>> completionStageArticles = articleService.getArticlesReactive(amount);  
    CompletionStage<Response> output;  
  
    output = completionStageArticles  
        .thenApply((articles) -> {  
            return convertArticlesToJsonArray(articles);  
        })  
        .thenApply((jsonArray) -> {  
            return Response.ok(jsonArray).build();  
        });  
  
    return output;  
}
```

# Exception Handling with imperative Code

```
public class ArticleService {  
    public List<Article> getArticles(int requestedAmount) throws NoDataAccess, InvalidInputParameter {
```

```
@GET  
@Path("/articles")  
public Response getArticles(int amount) {  
    try {  
        JSONArray json = convertToJsonArray(articleService.getArticles(amount));  
        return Response.ok(json).build();  
    } catch (NoDataAccess e) {  
        e.printStackTrace();  
        return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();  
    } catch (InvalidInputParameter e) {  
        return Response.status(Response.Status.NO_CONTENT).build();  
    }  
}
```

# Exception Handling with reactive Code

```
public class ArticleService {  
    public CompletionStage<List<Article>> getArticlesReactive(int requestedAmount) {
```

# Exception Handling with reactive Code

```
public class ArticleService {  
    public CompletionStage<List<Article>> getArticlesReactive(int requestedAmount) {
```

# Exception Handling with reactive Code

```
public class ArticleService {  
  
    public CompletionStage<List<Article>> getArticlesReactive(int requestedAmount) {  
  
        if (requestedAmount < 0)  
            return CompletableFuture.failedFuture(new InvalidInputParameter());
```

```
articleService.getArticlesReactive(amount)  
    .thenApply(articles -> {  
        if (errorOccurred) {  
            completableFuture.completeExceptionally(new InvalidInputParameter());  
        }  
        return articles;  
    })
```

# Exception Handling with reactive Code

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build())  
        .exceptionally(throwable -> {  
            if (throwable.getCause().toString().equals(InvalidInputParameter.class.getName())) {  
                return Response.status(Response.Status.BAD_REQUEST).build();  
            }  
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();  
        });  
}
```

# Exception Handling with reactive Code

```
public CompletionStage<Response> getArticlesReactive(int amount) {  
    return articleService.getArticlesReactive(amount)  
        .thenApply(articles -> convertArticlesToJsonArray(articles))  
        .thenApply(jsonArray -> Response.ok(jsonArray).build())  
        .exceptionally(throwable -> {  
            if (throwable.getCause().toString().equals(InvalidInputParameter.class.getName())) {  
                return Response.status(Response.Status.BAD_REQUEST).build();  
            }  
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();  
        });  
}
```

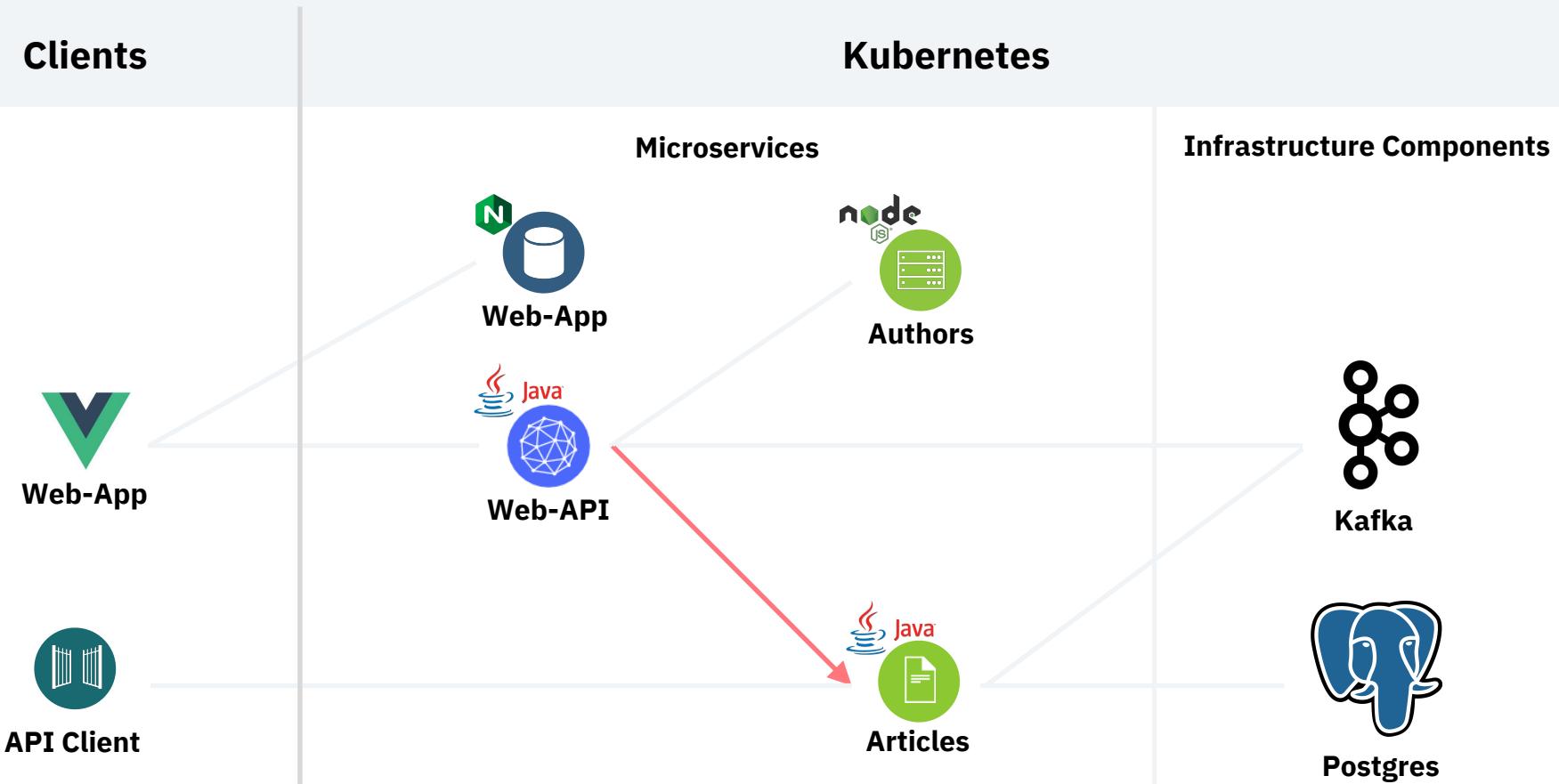
# Timeouts

```
public CompletionStage<List<Article>> getArticlesReactive() {  
    return client.query("SELECT id, title, url, author, creationdate FROM articles ORDER BY id ASC")  
        .toCompletableFuture()  
        .orTimeout(MAXIMAL_DURATION, TimeUnit.MILLISECONDS)  
        .thenApply(rowSet -> {  
            List<Article> list = new ArrayList<>(rowSet.size());  
            for (Row row : rowSet) {  
                list.add(fromRow(row));  
            }  
            return list;  
        }).exceptionally(throwable -> {  
            throw new NoConnectivity();  
       });  
}
```

# Timeouts

```
public CompletionStage<List<Article>> getArticlesReactive() {  
    return client.query("SELECT id title url author creationdate FROM articles ORDER BY id ASC")  
        .toCompletableFuture()  
        .orTimeout(MAXIMAL_DURATION, TimeUnit.MILLISECONDS)  
        .thenApply(rowSet -> {  
            List<Article> list = new ArrayList<>(rowSet.size());  
            for (Row row : rowSet) {  
                list.add(fromRow(row));  
            }  
            return list;  
        }).exceptionally(throwable -> {  
            throw new NoConnectivity();  
       });  
}
```

# Invoking REST APIs asynchronously



# MicroProfile Client

```
import org.eclipse.microprofile.rest.client.annotation.RegisterProvider;
import java.util.concurrent.CompletionStage;

@RegisterProvider(ExceptionMapperArticles.class)
public interface ArticlesServiceReactive {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    CompletionStage<List<CoreArticle>> getArticlesFromService(@QueryParam("amount") int amount);

}
```

# MicroProfile Client

```
import org.eclipse.microprofile.rest.client.annotation.RegisterProvider;
import java.util.concurrent.CompletionStage;

@RegisterProvider(ExceptionMapperArticles.class)
public interface ArticlesServiceReactive {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    CompletionStage<List<CoreArticle>> getArticlesFromService(@QueryParam("amount") int amount);

}
```

# MicroProfile Client

```
import org.eclipse.microprofile.rest.client.ext.ResponseExceptionMapper;
import javax.ws.rs.ext.Provider;

@Provider
public class ExceptionMapperArticles implements ResponseExceptionMapper<InvalidArticle> {

    @Override
    public InvalidArticle toThrowable(Response response) {
        if (response.getStatus() == 204)
            return new InvalidArticle();
        return null;
    }
}
```

# MicroProfile Client

```
import org.eclipse.microprofile.rest.client.ext.ResponseExceptionMapper;
import javax.ws.rs.ext.Provider;

@Provider
public class ExceptionMapperArticles implements ResponseExceptionMapper<InvalidArticle> {

    @Override
    public InvalidArticle toThrowable(Response response) {
        if (response.getStatus() == 204)
            return new InvalidArticle();
        return null;
    }
}
```

# MicroProfile Client

```
private ArticlesServiceReactive articlesServiceReactive;

@PostConstruct
void initialize() {
    URI api = UriBuilder.fromUri("http://{host}:{port}/v2/articles").build(articlesHost, articlesPort);
    articlesServiceReactive = RestClientBuilder.newBuilder()
        .baseUri(api)
        .register(ExceptionMapperArticles.class)
        .build(ArticlesServiceReactive.class);
}

public CompletionStage<List<CoreArticle>> getArticlesReactive(int amount) {
    return articlesServiceReactive.getArticlesFromService(amount)
        .toCompletableFuture()
        .orTimeout(MAXIMAL_DURATION, TimeUnit.MILLISECONDS);
}
```

# MicroProfile Client

```
private ArticlesServiceReactive articlesServiceReactive;

@PostConstruct
void initialize() {
    URI api = UriBuilder.fromUri("http://{host}:{port}/v2/articles").build(articlesHost, articlesPort);
    articlesServiceReactive = RestClientBuilder.newBuilder()
        .baseUri(api)
        .register(ExceptionMapperArticles.class)
        .build(ArticlesServiceReactive.class);
}

public CompletionStage<List<CoreArticle>> getArticlesReactive(int amount) {
    return articlesServiceReactive.getArticlesFromService(amount)
        .toCompletableFuture()
        .orTimeout(MAXIMAL_DURATION, TimeUnit.MILLISECONDS);
}
```

Try out the end-to-end  
microservices example  
cloud-native-starter!

# Focus on Developer Experience

Screenshot of a GitHub repository page for "IBM / cloud-native-starter".

The page includes a navigation bar with links for Pull requests, Issues, Marketplace, and Explore, along with user notifications and profile icons.

The repository title is "IBM / cloud-native-starter".

Action buttons include Unwatch (36), Unstar (238), Fork (90).

The main navigation tabs are Code, Issues (1), Pull requests (1), Actions, Projects (0), Wiki, Security, Insights, and Settings. The Code tab is selected.

A brief description of the repository: "Cloud Native Starter for Java/Jakarta EE based Microservices on Kubernetes and Istio" with a link to <https://cloud-native-starter.mybluemix.net>.

Topics listed: cloud-native, microservice, java, javascript, nodejs, kubernetes, istio, javaee, microprofile, Manage topics.

Repository statistics: 733 commits, 2 branches, 0 packages, 0 releases, 1 environment, 9 contributors, Apache-2.0 license.

A detailed breakdown of code coverage by language:

| Language   | Percentage |
|------------|------------|
| Shell      | 58.4%      |
| Java       | 32.0%      |
| Vue        | 4.7%       |
| JavaScript | 2.8%       |
| Dockerfile | 1.6%       |
| HTML       | 0.5%       |

# Several Kubernetes Environments

The screenshot shows a GitHub repository page for the 'cloud-native-starter' project under the 'IBM' organization. The repository name is 'cloud-native-starter'. The 'reactive' branch is selected. The page includes navigation links for Code, Issues (1), Pull requests (1), Actions, Projects (0), Wiki, Security, Insights, and Settings. There are buttons for Unwatch (36), Unstar, Fork (90), and a search bar. The main content area displays the 'README.md' file, which contains a section titled 'Reactive Java Microservices' and a description of how to implement reactive microservices with Quarkus, MicroProfile, Vert.x, Kafka, and Postgres. It also lists several setup options.

IBM / [cloud-native-starter](#)

Code Issues 1 Pull requests 1 Actions Projects 0 Wiki Security Insights Settings

Branch: master [cloud-native-starter / reactive /](#)

Create new file Upload files Find file History

## README.md

### Reactive Java Microservices

This part of the cloud-native-starter project describes how to implement reactive microservices with Quarkus, MicroProfile, Vert.x, Kafka and Postgres.

- [Setup in Minikube](#)
- [Server-side Setup in IBM Cloud Kubernetes Service](#)
- [Client-side Setup in IBM Cloud Kubernetes Service](#)
- [Setup in CodeReady Containers / local OpenShift](#)
- [Setup of local Development Environment](#)

# IBM Cloud Kubernetes Service including Istio and Knative

IBM Cloud Search resources and offerings... Catalog Docs Support Manage Niklas Heidloff's Account ⚙️ 🧑

Clusters / niklas-heidloff-cns

 niklas-heidloff-cns • Normal

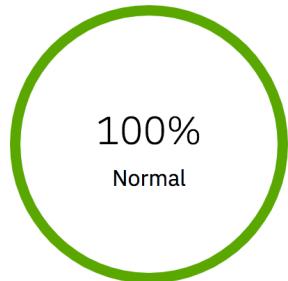
[Web Terminal \(beta\)](#) [Kubernetes Dashboard ↗](#) [Connect via CLI](#) ⋮

[Access](#) [Overview](#) [Worker Nodes](#) [Worker Pools](#) [Add-ons](#)

**Summary**

|                                    |   |
|------------------------------------|---|
| <b>Cluster ID</b>                  | 401c8d4144a744f6978c68a12c8335c5  |
| <b>Master Status</b>               | Ready   |
| <b>Kubernetes version</b>          | 1.12.7_1548   |
| <b>Zones</b>                       | hou02   |
| <b>Owner</b>                       | niklas_heidloff@de.ibm.com  |
| <b>Resource group</b>              | default   |
| <b>Key protect (Beta)</b>          | <a href="#">Enable</a>  |
| <b>IAM pullsecrets</b>             | Enabled   |
| <b>Public service endpoint URL</b> | <a href="https://c5.dal12.containers.cloud.ibm.com:31446">https://c5.dal12.containers.cloud.ibm.com:31446</a> <a href="#">Disable</a> |

**Worker Nodes** 1



|   |          |
|---|----------|
| 1 | Normal   |
| 0 | Warning  |
| 0 | Critical |
| 0 | Pending  |

# Summary

Get the code →



To be done

IBM loves open source

Kubernetes and Istio  
OpenJ9 & AdoptOpenJDK  
MicroProfile  
Quarkus and Open Liberty

IBM Developer

[developer.ibm.com](https://developer.ibm.com)

IBM Cloud Lite account

[ibm.biz/nheidloff](https://ibm.biz/nheidloff)

