Developing reactive Microservices with Quarkus

Niklas Heidloff
Developer Advocate, IBM
@nheidloff

IBM **Developer**

Buzzword Bingo

Reactive Manifesto
Reactive Systems
Reactive Programming
Functional Programming
Asynchronous Programming
Reactive Streams
Reactive Operators

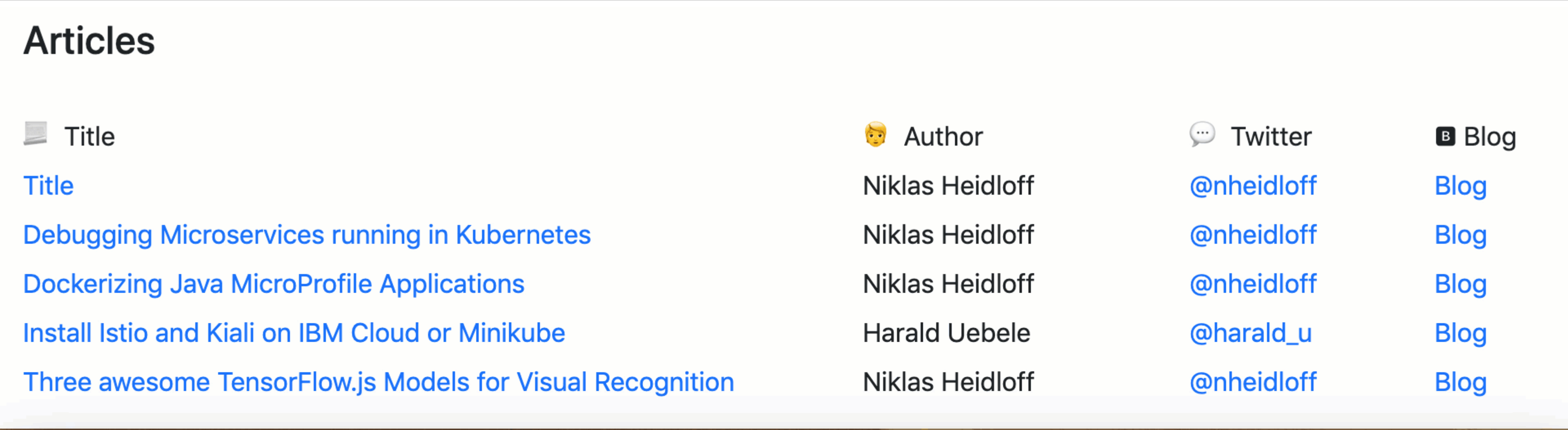
?

Let’s make it concrete

Reactive Web Application

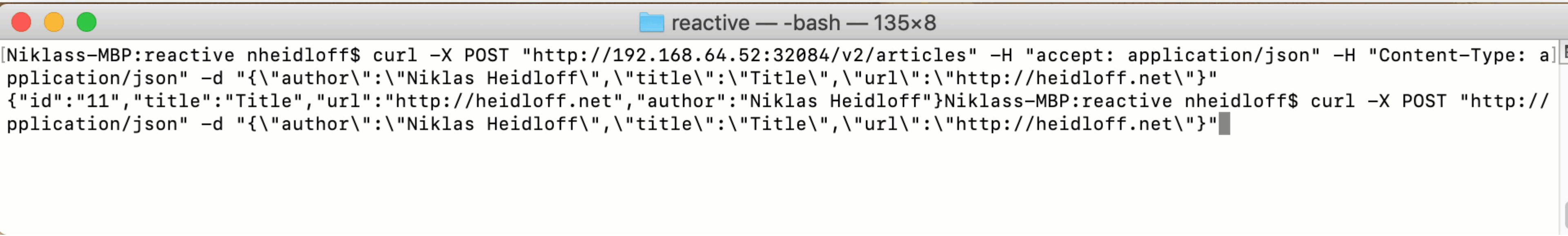
Reactive REST Endpoints

Reactive Web Application



Articles

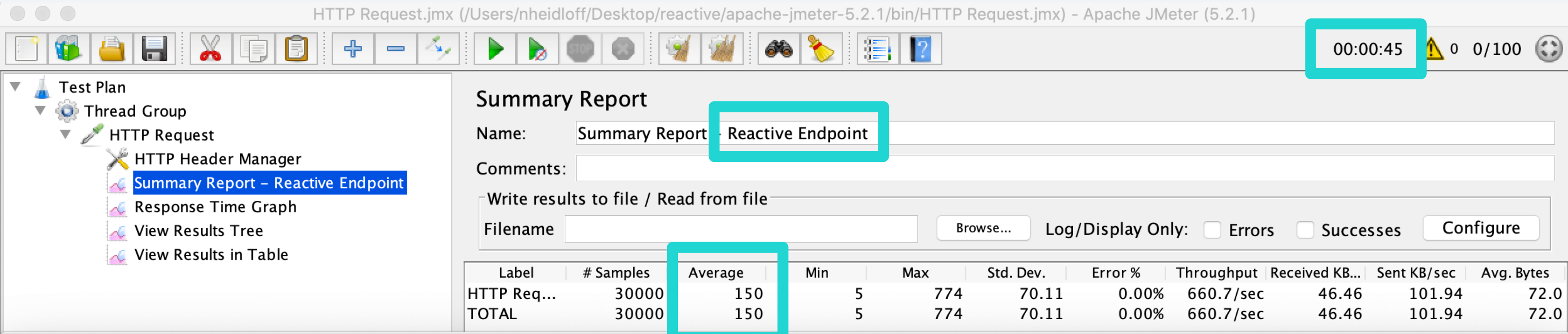
Title	Author	Twitter	Blog
Title	Niklas Heidloff	@nheidloff	Blog
Debugging Microservices running in Kubernetes	Niklas Heidloff	@nheidloff	Blog
Dockerizing Java MicroProfile Applications	Niklas Heidloff	@nheidloff	Blog
Install Istio and Kiali on IBM Cloud or Minikube	Harald Uebele	@harald_u	Blog
Three awesome TensorFlow.js Models for Visual Recognition	Niklas Heidloff	@nheidloff	Blog



reactive — -bash — 135×8

```
Niklass-MBP:reactive nheidloff$ curl -X POST "http://192.168.64.52:32084/v2/articles" -H "accept: application/json" -H "Content-Type: a
pplication/json" -d "{\"author\":\"Niklas Heidloff\",\"title\":\"Title\",\"url\":\"http://heidloff.net\"}"
{"id":"11","title":"Title","url":"http://heidloff.net","author":"Niklas Heidloff"}Niklass-MBP:reactive nheidloff$ curl -X POST "http://
pplication/json" -d "{\"author\":\"Niklas Heidloff\",\"title\":\"Title\",\"url\":\"http://heidloff.net\"}"
```

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 ⚠ 0 0/100

Summary Report

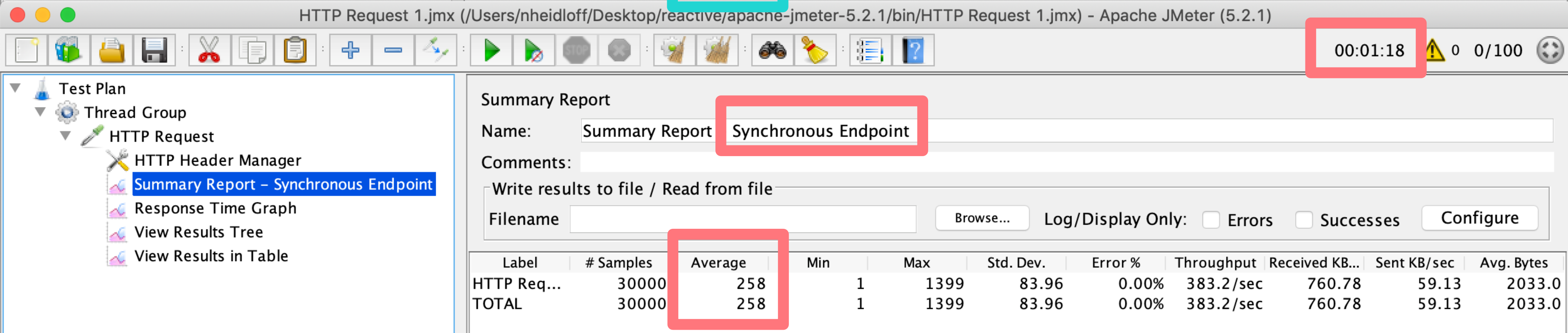
Name: Summary Report - Reactive Endpoint

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	72.0
TOTAL	30000	150	5	774	70.11	0.00%	660.7/sec	46.46	101.94	72.0



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 ⚠ 0 0/100

Summary Report

Name: Summary Report - Synchronous Endpoint

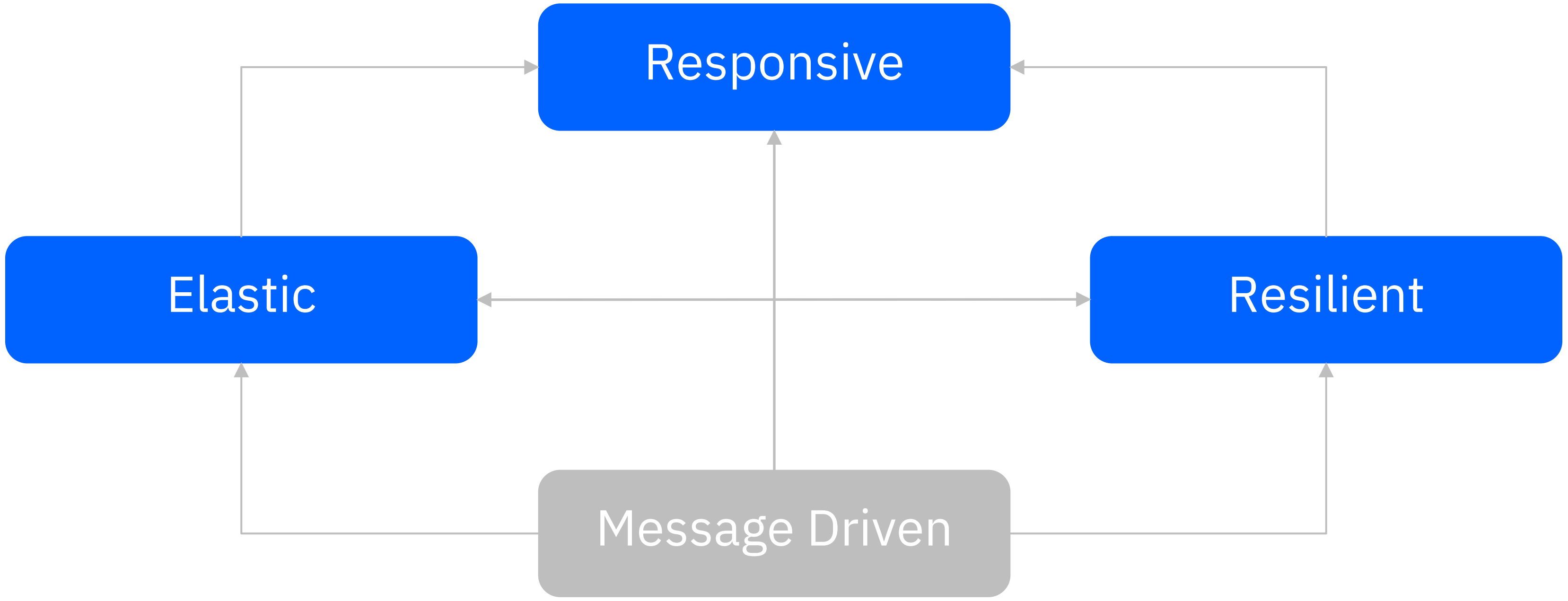
Comments:

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	2033.0
TOTAL	30000	258	1	1399	83.96	0.00%	383.2/sec	760.78	59.13	2033.0

Reactive Manifesto



Reactive Systems

!=

Reactive Programming

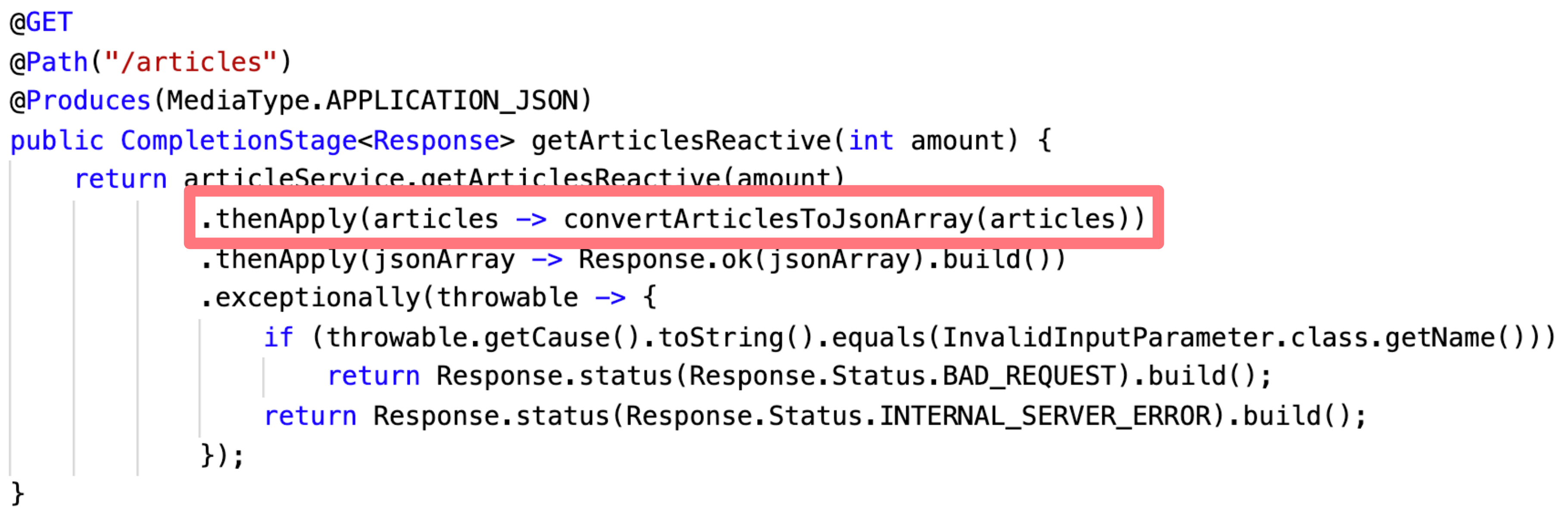
Reactive Programming is ...

```java
@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’

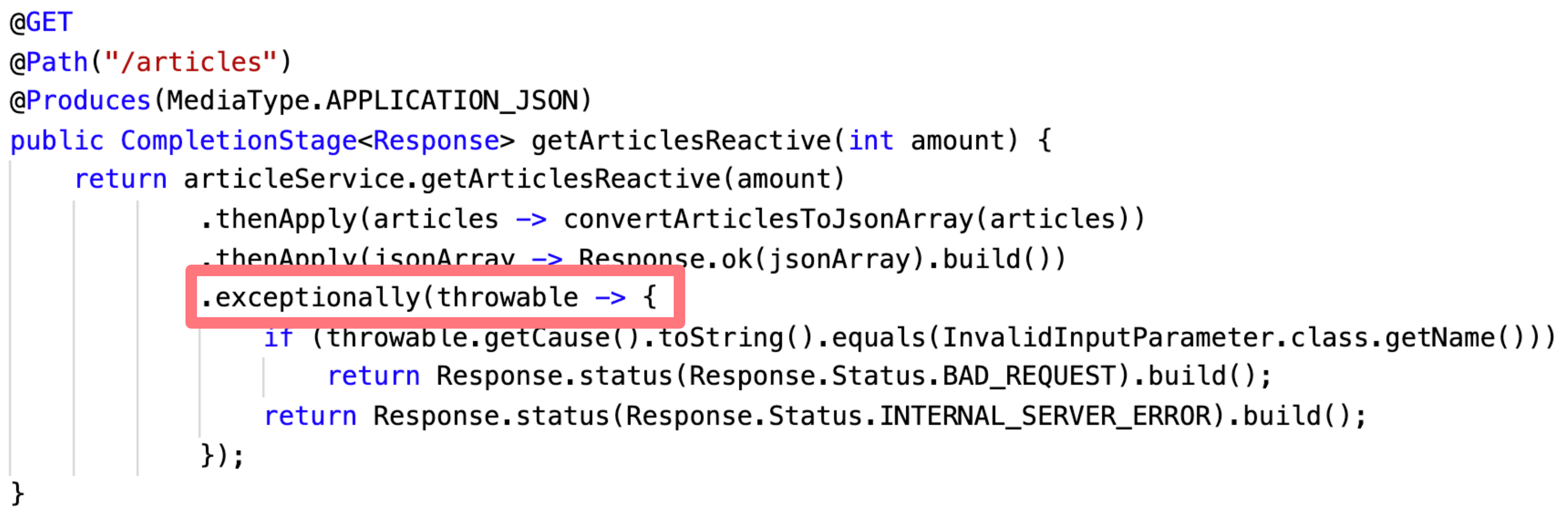
```java
@GET
@Path("/articles")
@Produces(MediaType.APPLICATION_JSON)
public CompletionStage<Response> getArticlesReactive(int amount) {
    return articleService.getArticlesReactive(amount)
            .thenApply(articles -> convertArticlesToJsonArray(articles))
            .thenApply(jsonArray -> Response.ok(jsonArray).build())
            .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’



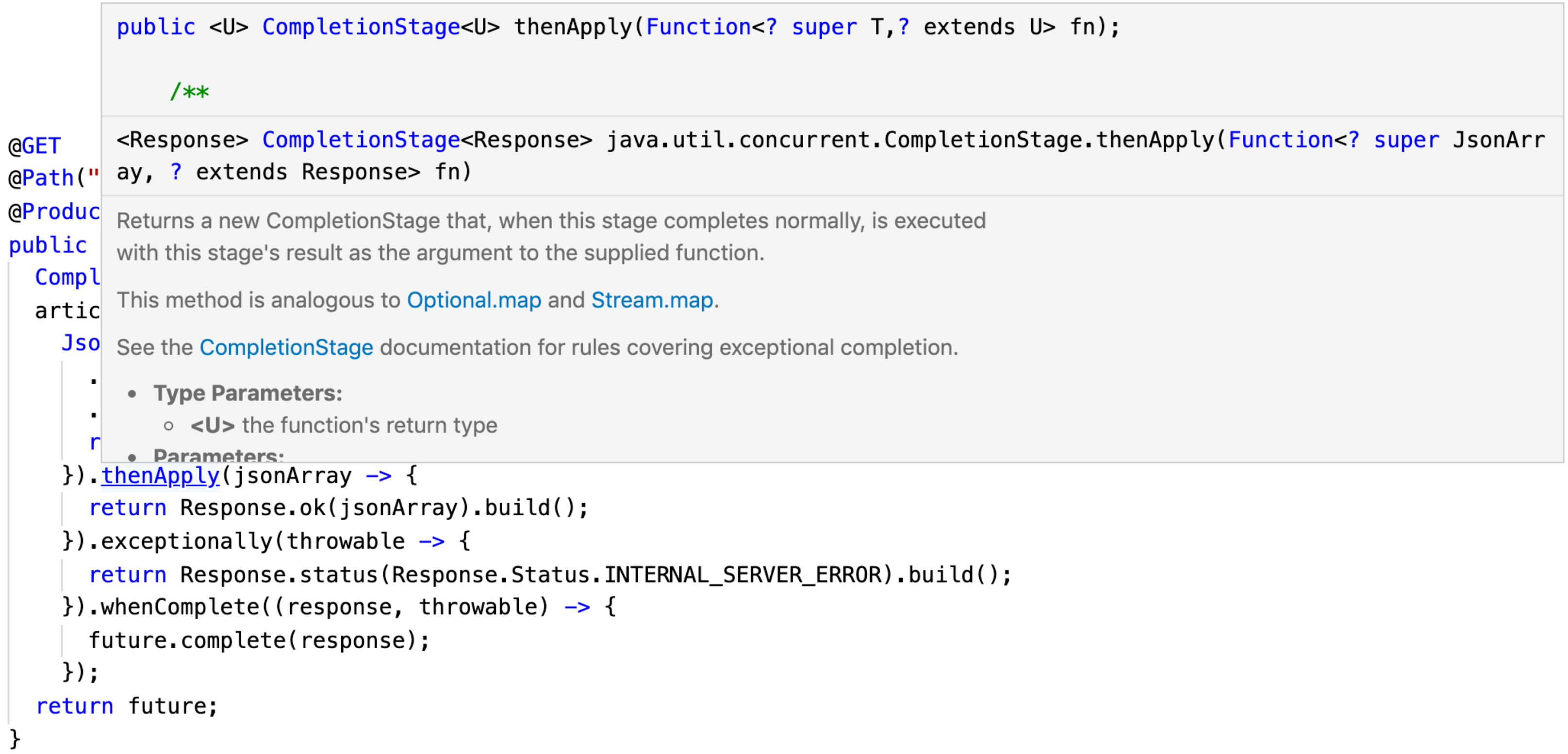
```java
@GET
@Path("/articles")
@Produces(MediaType.APPLICATION_JSON)
public CompletionStage<Response> getArticlesReactive(int amount) {
    return articleService.getArticlesReactive(amount)
            .thenApply(articles -> convertArticlesToJsonArray(articles))
            .thenApply(jsonArray -> Response.ok(jsonArray).build())
            .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’



```java
@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);

    /**
```

```
<Response> CompletionStage<Response> java.util.concurrent.CompletionStage.thenApply(Function<? super JsonArr
ay, ? extends Response> 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.

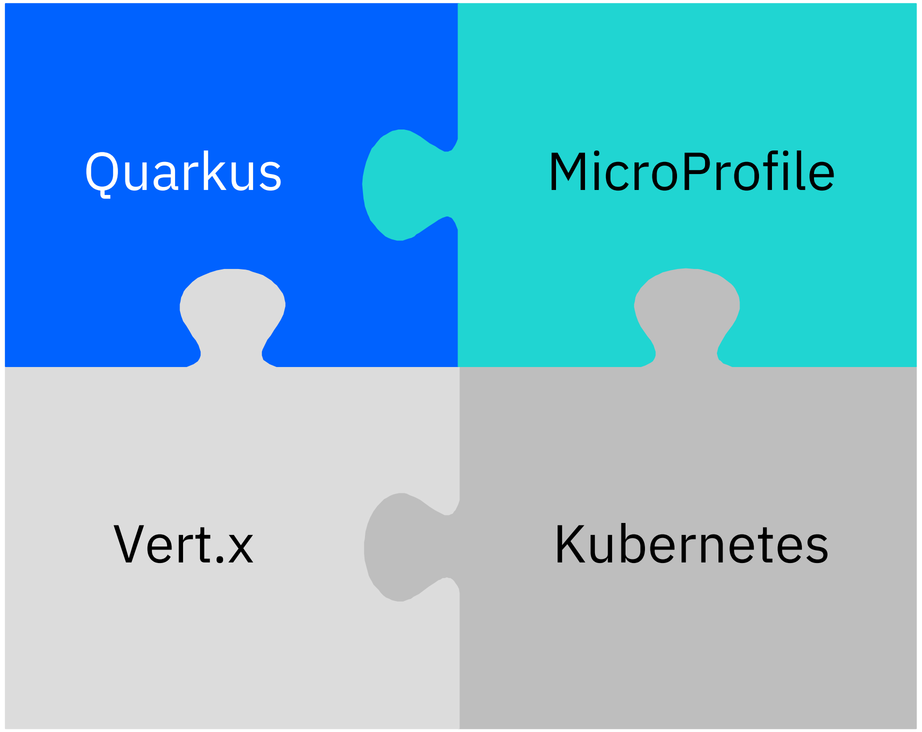
See the CompletionStage documentation for rules covering exceptional completion.

- **Type Parameters:**
  - **<U>** the function's return type
- **Parameters:**

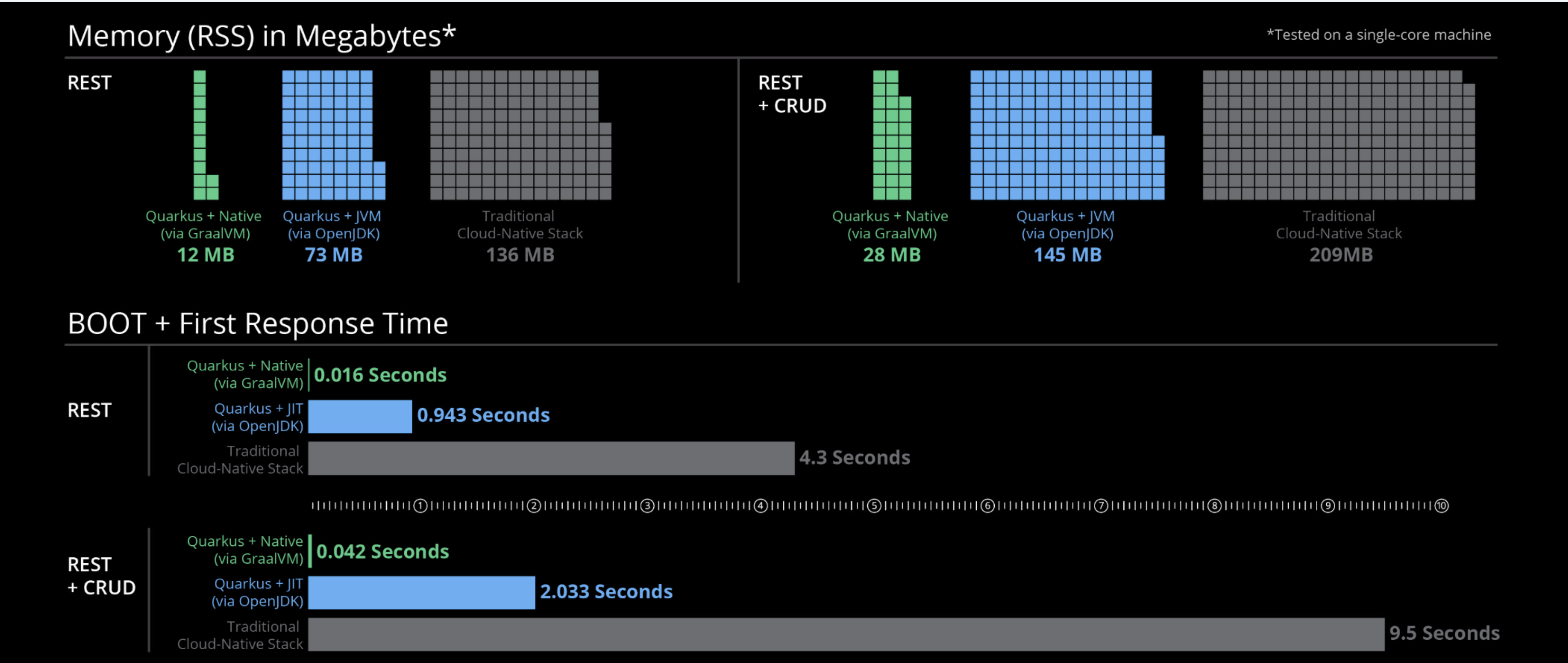
```
@GET
@Path("
@Produc
public
  Compl
  artic
    Jso
      .
      .
      r
    }).thenApply(jsonArray -> {
      return Response.ok(jsonArray).build();
    }).exceptionally(throwable -> {
      return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
    }).whenComplete((response, throwable) -> {
      future.complete(response);
    });
  return future;
}
```

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

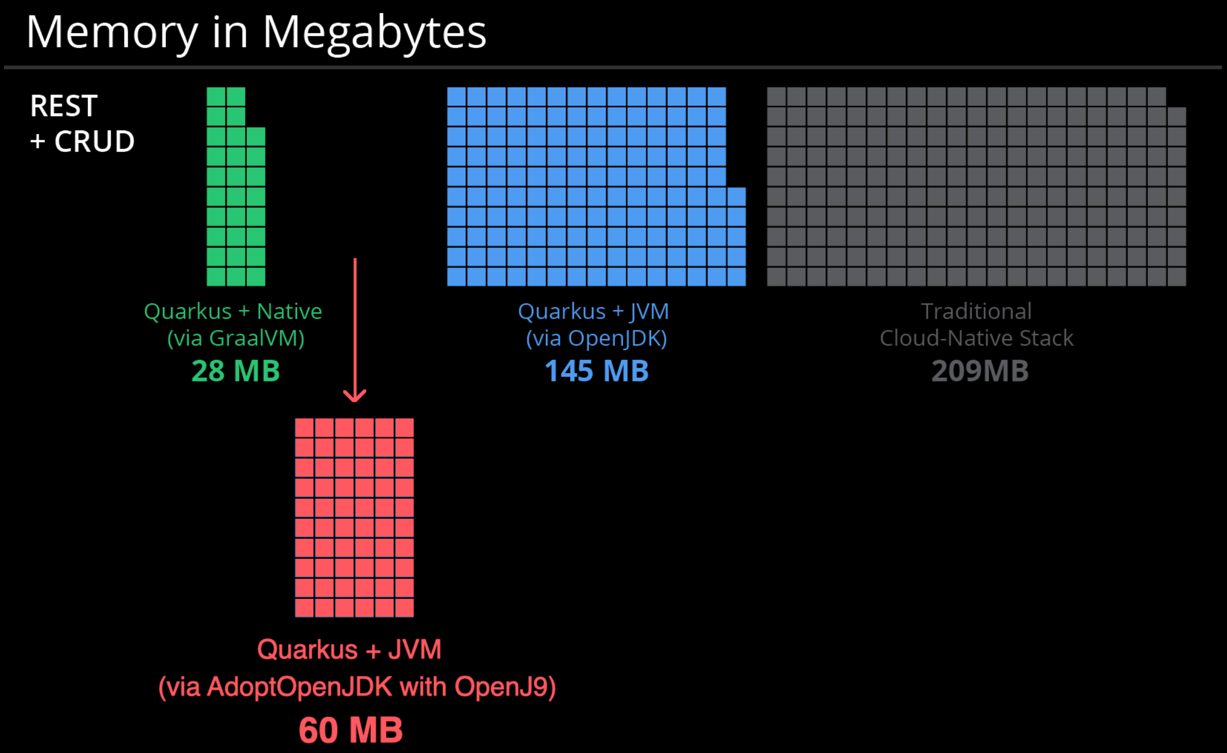
Technologies to build reactive Applications



Quarkus – Supersonic Subatomic Java



Quarkus using OpenJ9



“Optimizing Enterprise Java for a Microservices Architecture.“

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

`microprofile.io`

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

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

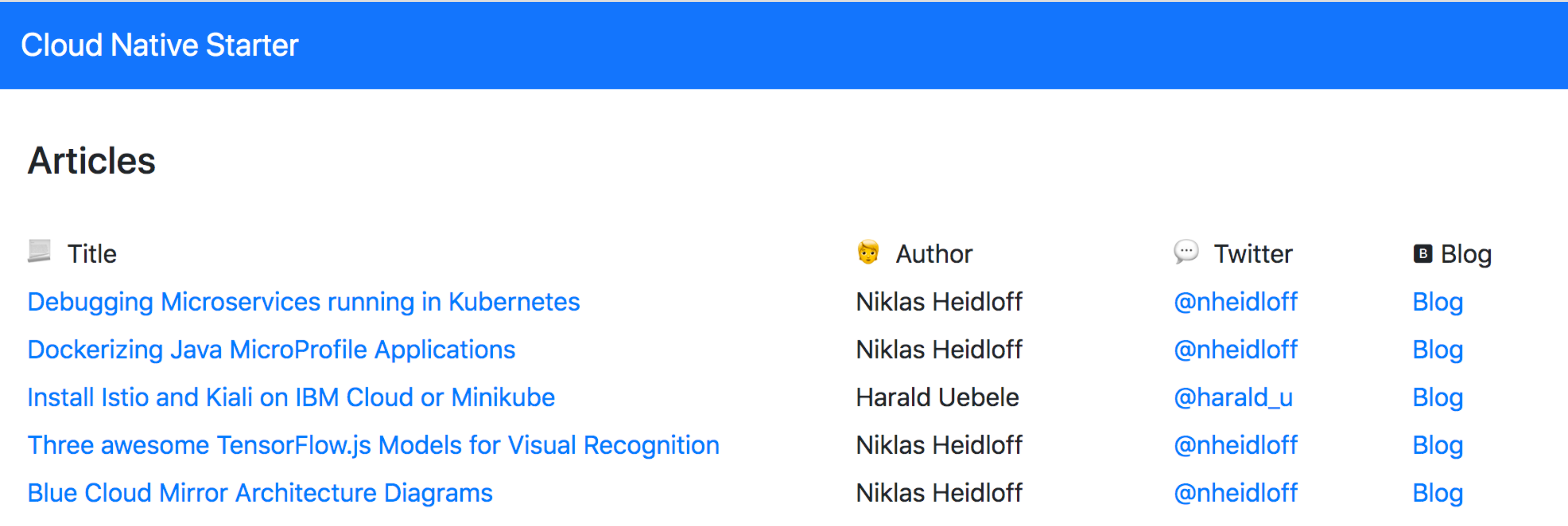
"Eclipse Vert.x is event driven and non blocking [...] and lets your app scale with minimal hardware."

`vertx.io`

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

`kubernetes.io`

Example Application

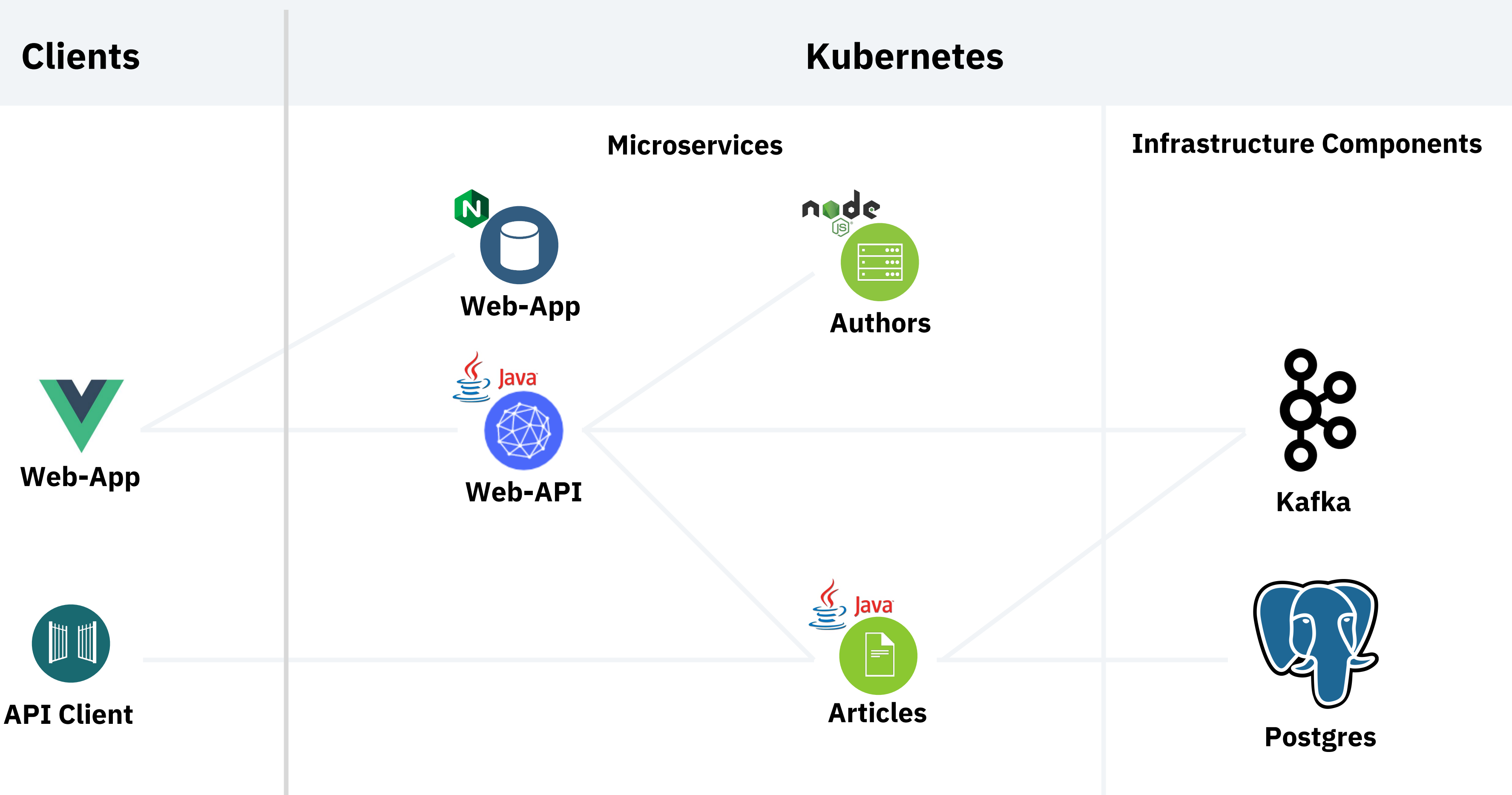


Cloud Native Starter

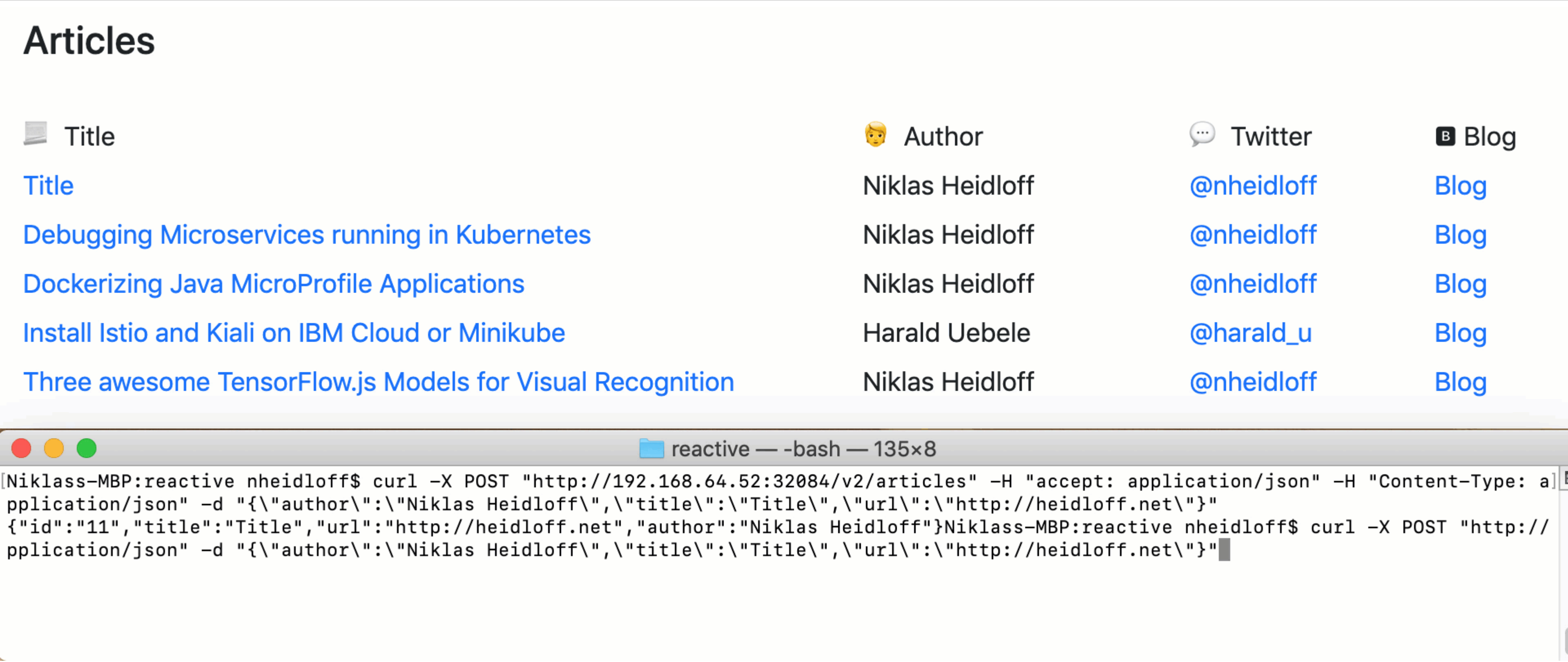
Articles

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

Architecture



Reactive Web Application



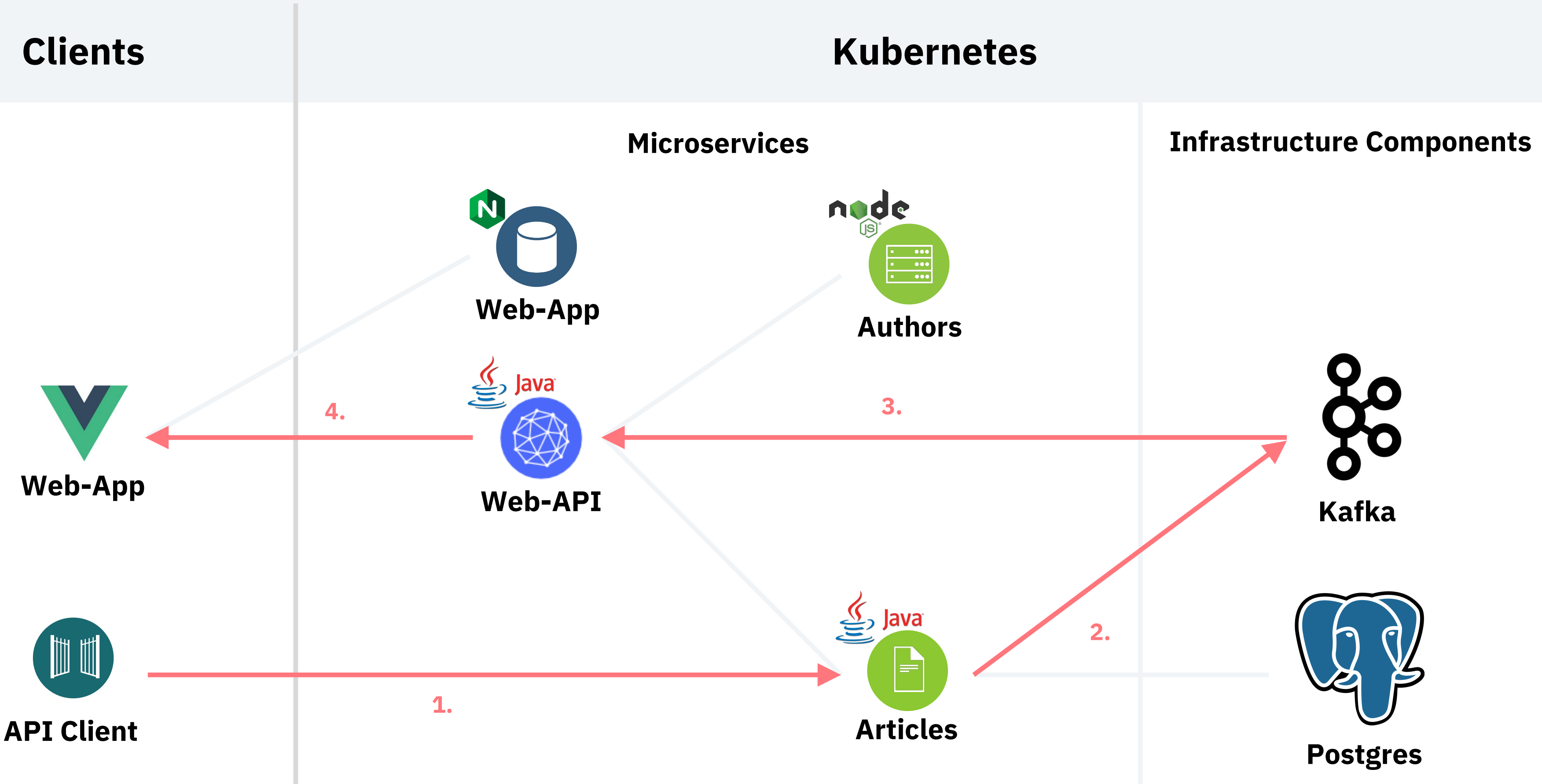
Articles

Title	Author	Twitter	Blog
Title	Niklas Heidloff	@nheidloff	Blog
Debugging Microservices running in Kubernetes	Niklas Heidloff	@nheidloff	Blog
Dockerizing Java MicroProfile Applications	Niklas Heidloff	@nheidloff	Blog
Install Istio and Kiali on IBM Cloud or Minikube	Harald Uebele	@harald_u	Blog
Three awesome TensorFlow.js Models for Visual Recognition	Niklas Heidloff	@nheidloff	Blog

reactive — -bash — 135×8

```
Niklass-MBP:reactive nheidloff$ curl -X POST "http://192.168.64.52:32084/v2/articles" -H "accept: application/json" -H "Content-Type: a
pplication/json" -d "{\"author\":\"Niklas Heidloff\",\"title\":\"Title\",\"url\":\"http://heidloff.net\"}"
{"id":"11","title":"Title","url":"http://heidloff.net","author":"Niklas Heidloff"}Niklass-MBP:reactive nheidloff$ curl -X POST "http://
pplication/json" -d "{\"author\":\"Niklas Heidloff\",\"title\":\"Title\",\"url\":\"http://heidloff.net\"}"
```

Notifications for Web Applications



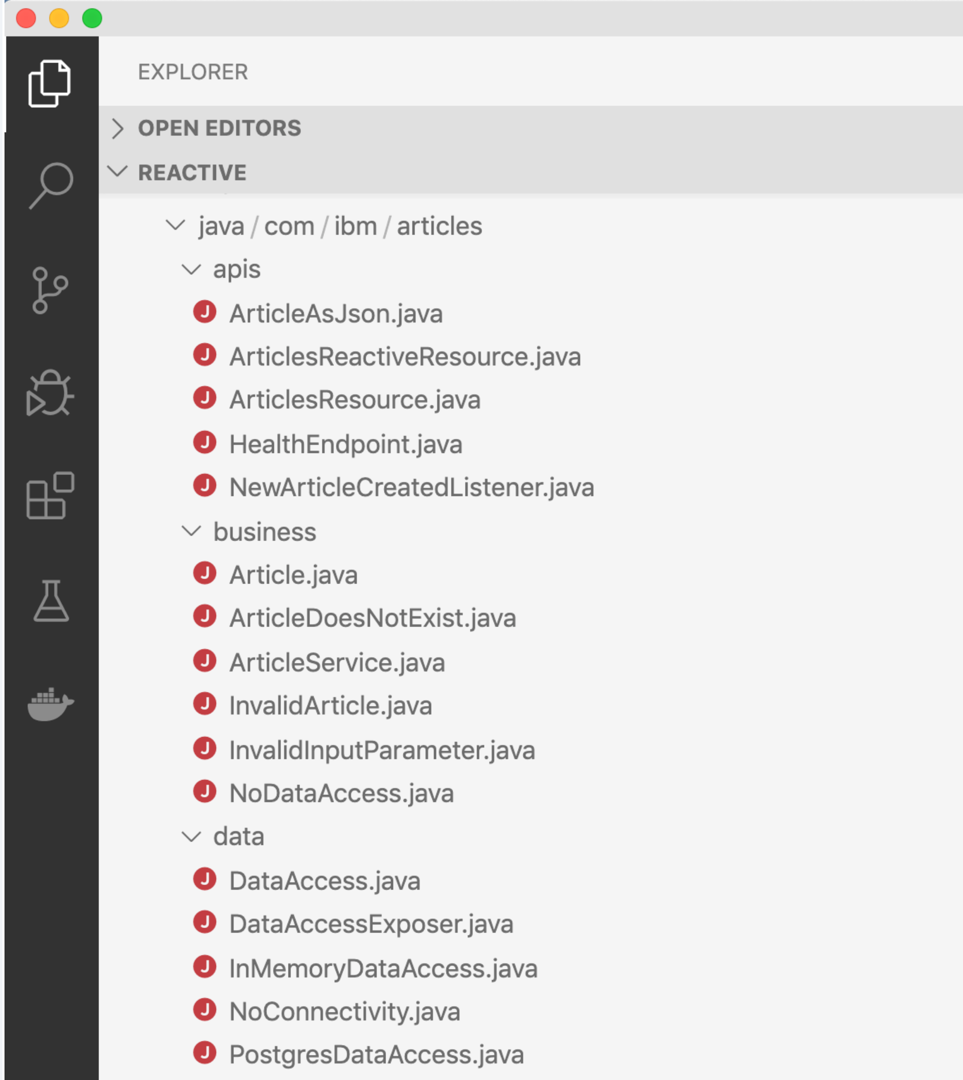
Clean Architecture

1. APIs
   REST endpoints and messaging

2. Business
   Logic of services and entities

3. Data
   Access to databases or other services

@nheidloff



EXPLORER

> OPEN EDITORS

REACTIVE

```
java / com / ibm / articles
  apis
    ArticleAsJson.java
    ArticlesReactiveResource.java
    ArticlesResource.java
    HealthEndpoint.java
    NewArticleCreatedListener.java
  business
    Article.java
    ArticleDoesNotExist.java
    ArticleService.java
    InvalidArticle.java
    InvalidInputParameter.java
    NoDataAccess.java
  data
    DataAccess.java
    DataAccessExposer.java
    InMemoryDataAccess.java
    NoConnectivity.java
    PostgresDataAccess.java
```

Vert.x Event Bus

```java
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);
    }
```

```java
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

```java
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);
    }
```

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

public class NewArticleCreatedListener {

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

Vert.x Event Bus

```java
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);
    }
```

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

public class NewArticleCreatedListener {

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

Kafka API

```java
@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

```java
@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

```java
@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

```java
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

```java
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`

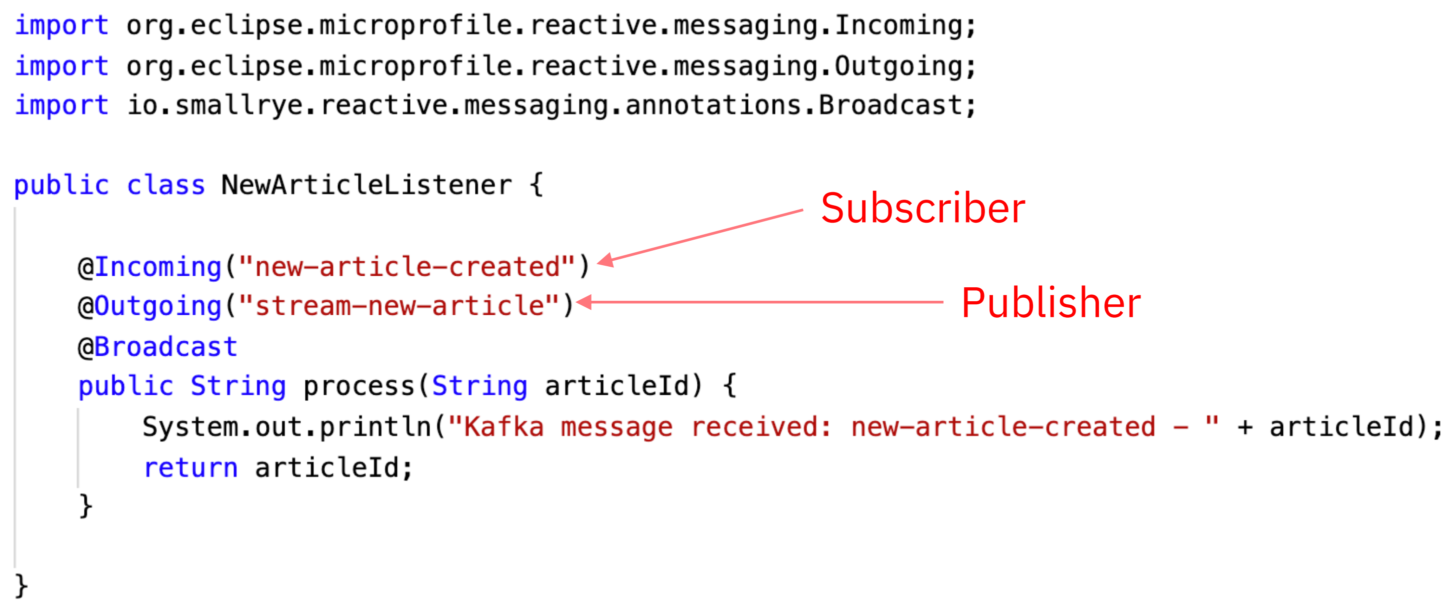
Components:

1. Subscriber
2. Publisher
3. Processor

Java:

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

MicroProfile Reactive Messaging



```java
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;
    }

}
```

Server Sent Events

```java
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;
    }
}
```

```javascript
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

```java
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;
    }
}
```

```javascript
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

```java
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;
    }
}
```

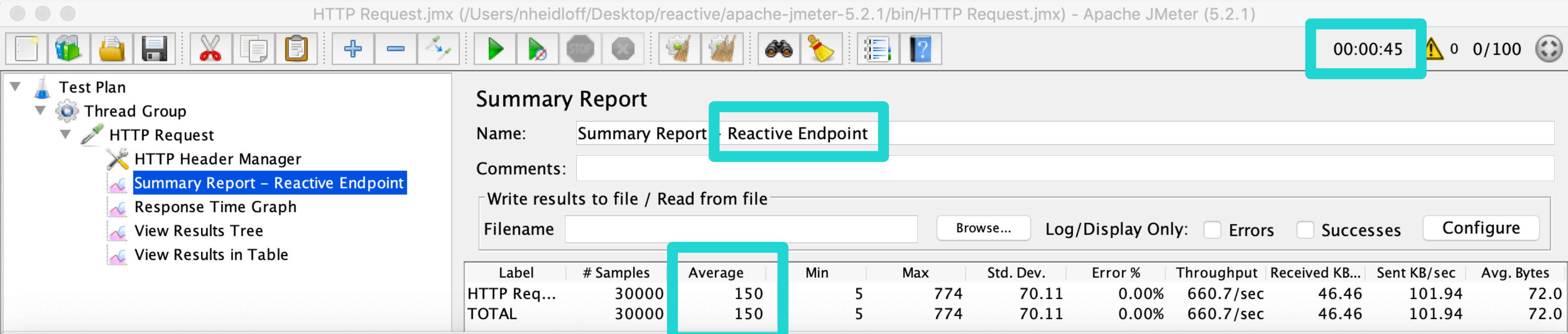
```javascript
let url = this.$store.state.endpoints.api +
  "server-sent-events";
this.readArticles();
let source = new EventSource(url);
let that = this;
source.onmessage = function (event) {
  that.readArticles();
};
```

Server Sent Events

```java
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;
    }
}
```

```javascript
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 ⚠ 0 0/100

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

Summary Report

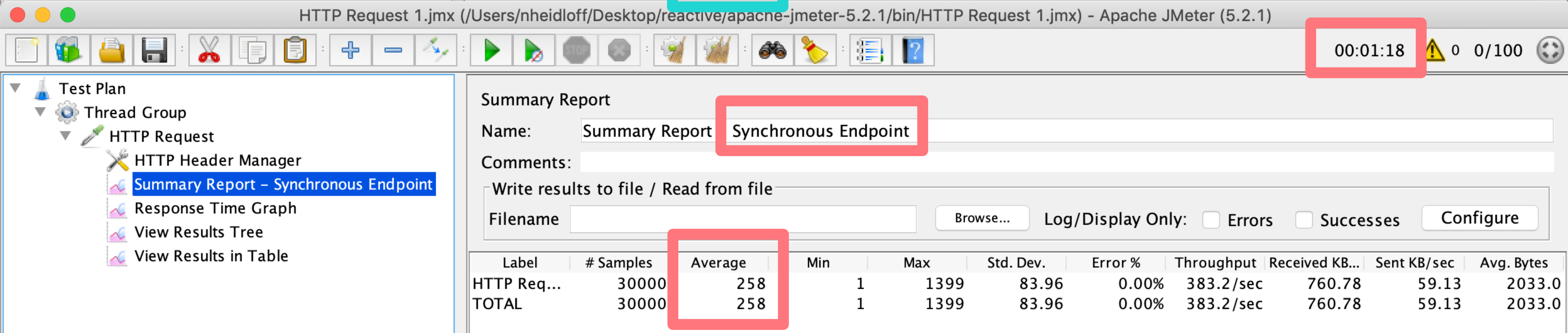
Name: Summary Report – Reactive Endpoint

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	72.0
TOTAL	30000	150	5	774	70.11	0.00%	660.7/sec	46.46	101.94	72.0



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 ⚠ 0 0/100

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

Summary Report

Name: Summary Report – Synchronous Endpoint

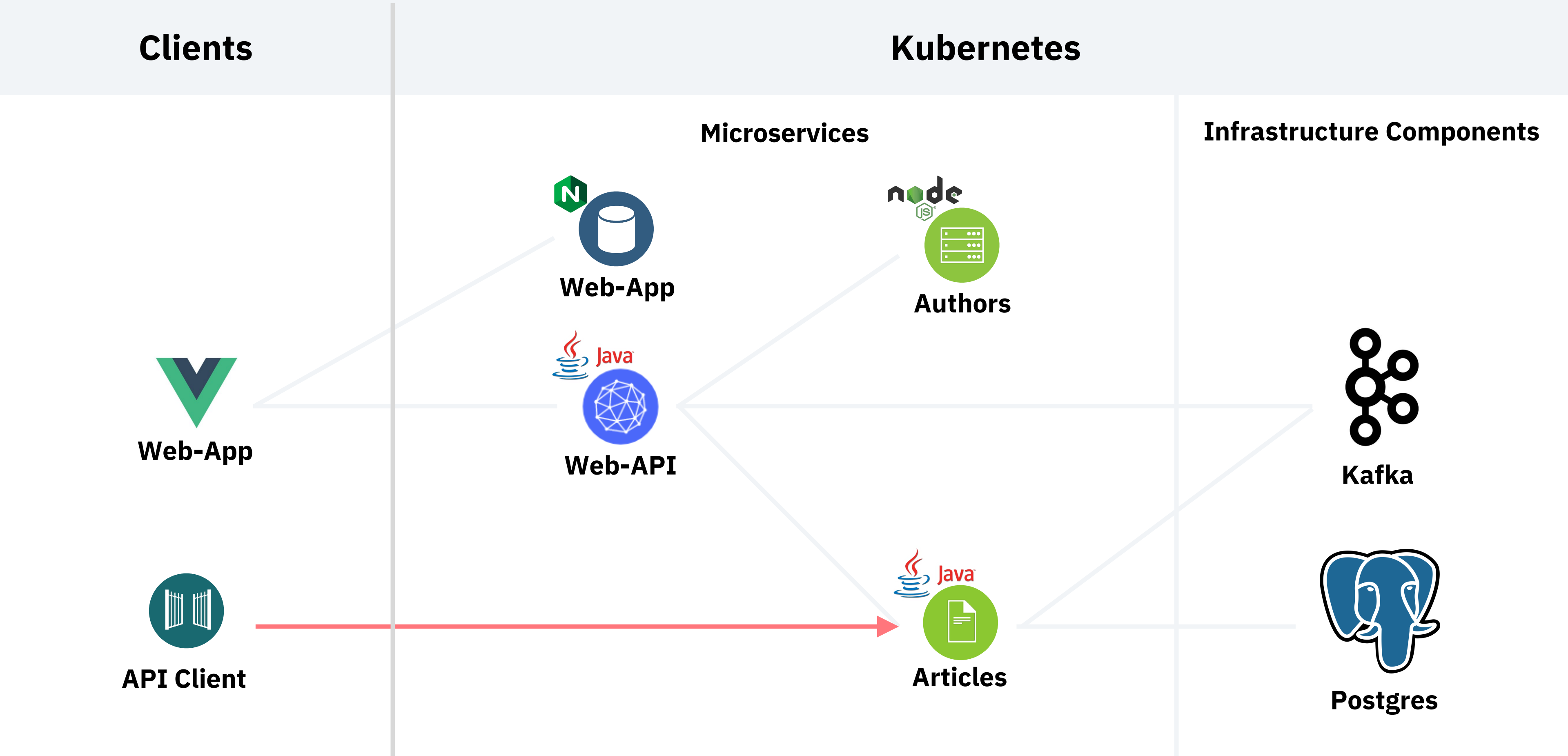
Comments:

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	2033.0
TOTAL	30000	258	1	1399	83.96	0.00%	383.2/sec	760.78	59.13	2033.0

Reactive REST Endpoint



Reactive REST Endpoint

```java
@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

```java
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

```java
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

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

Chained Completion Stages

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

Chained Completion Stages

```java
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

```java
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

```java
public class ArticleService {

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

```java
@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

```java
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

```java
public class ArticleService {

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

        if (requestedAmount < 0)
            return CompletableFuture.failedFuture(new InvalidInputParameter());
```

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

Exception Handling with reactive Code

```java
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();
        });
}
```

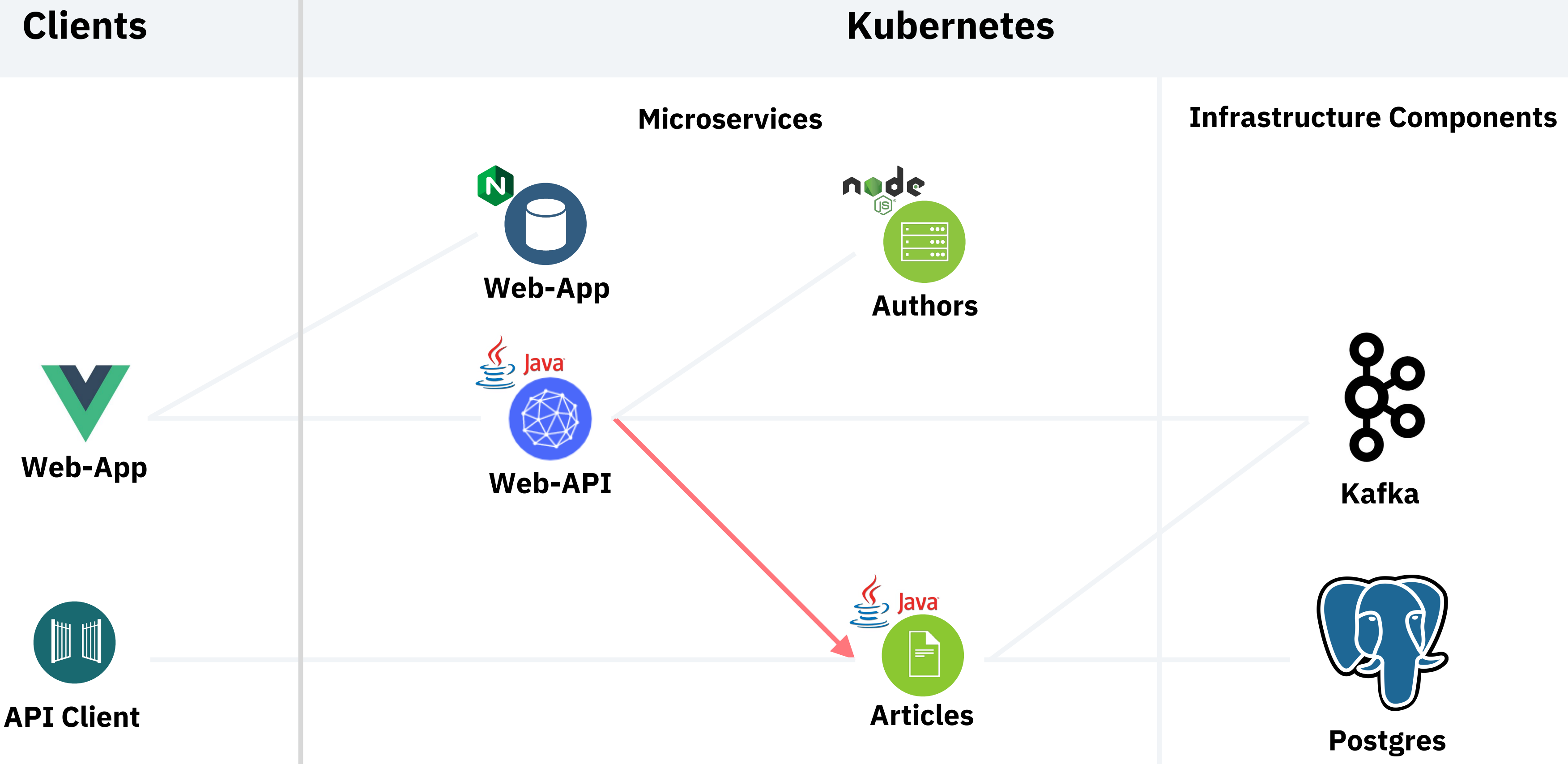
Timeouts

```
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

```java
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

```java
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

```java
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

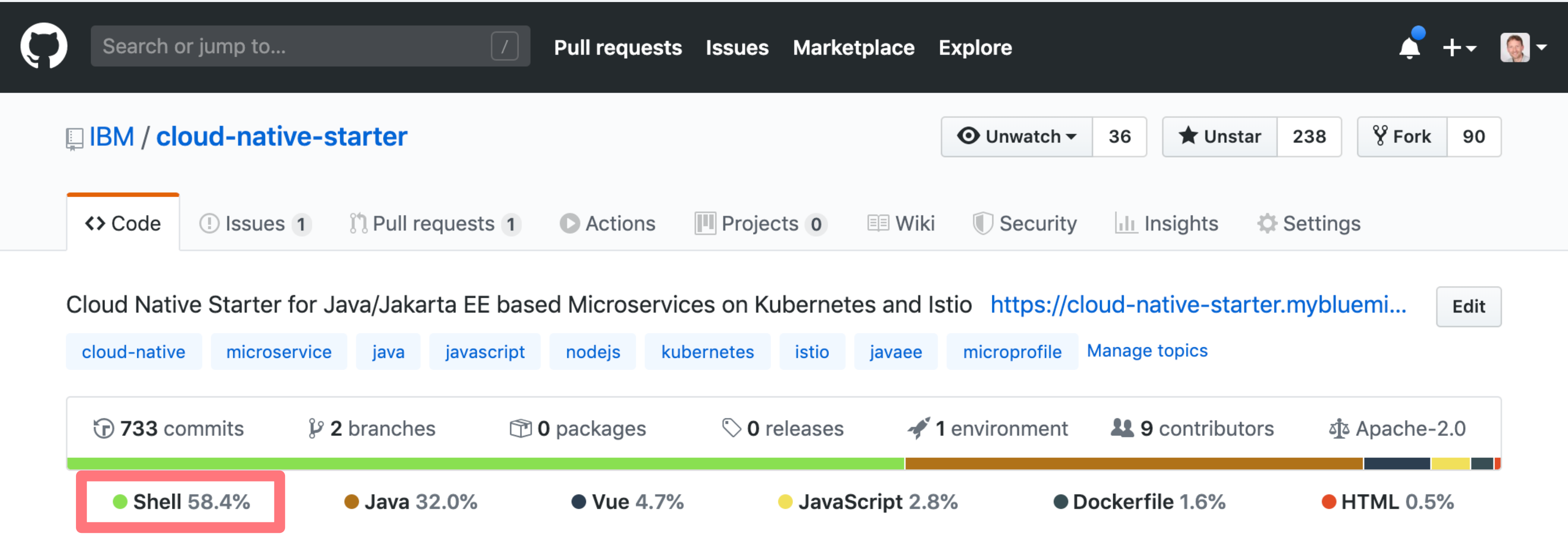
```java
private ArticlesServiceReactive articlesServiceReactive;

@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

```java
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



IBM / cloud-native-starter

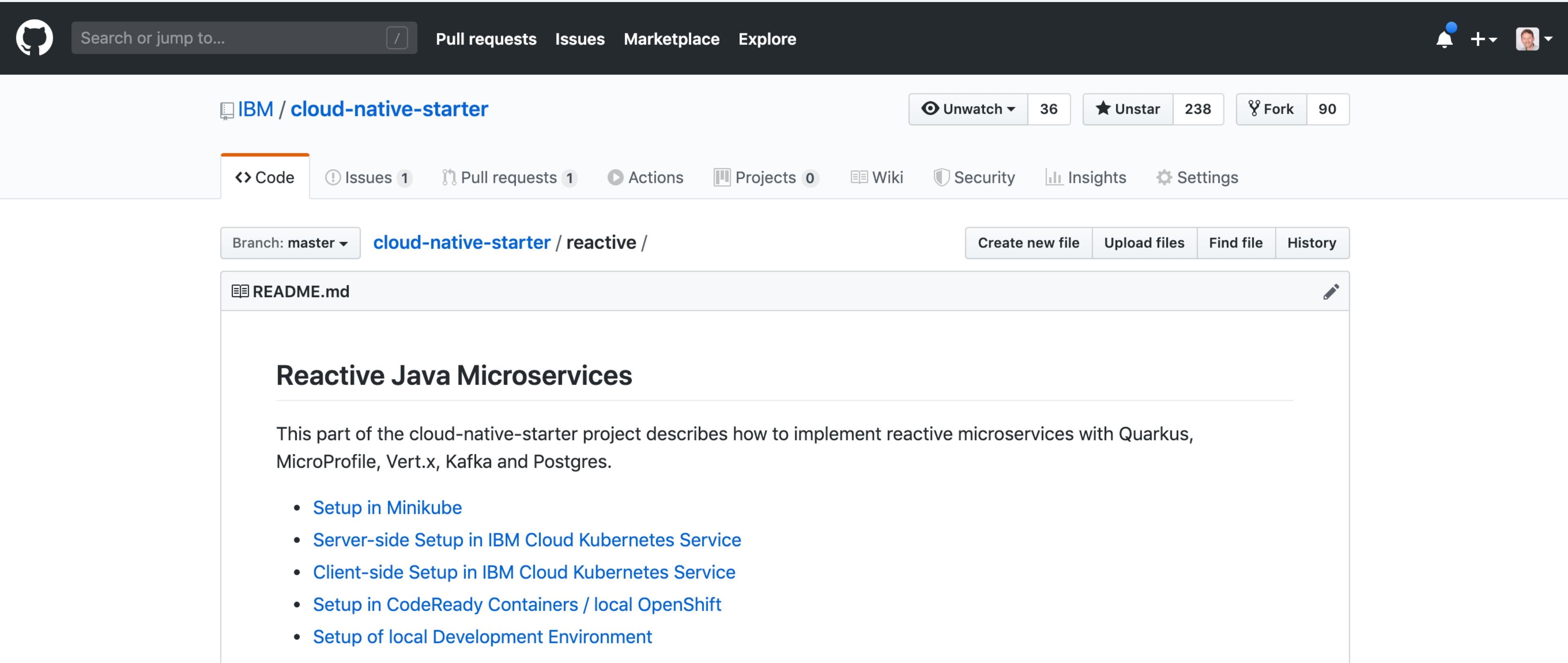
Cloud Native Starter for Java/Jakarta EE based Microservices on Kubernetes and Istio https://cloud-native-starter.mybluemi...

cloud-native microservice java javascript nodejs kubernetes istio javaee microprofile

733 commits · 2 branches · 0 packages · 0 releases · 1 environment · 9 contributors · Apache-2.0

● Shell 58.4% ● Java 32.0% ● Vue 4.7% ● JavaScript 2.8% ● Dockerfile 1.6% ● HTML 0.5%

Several Kubernetes Environments

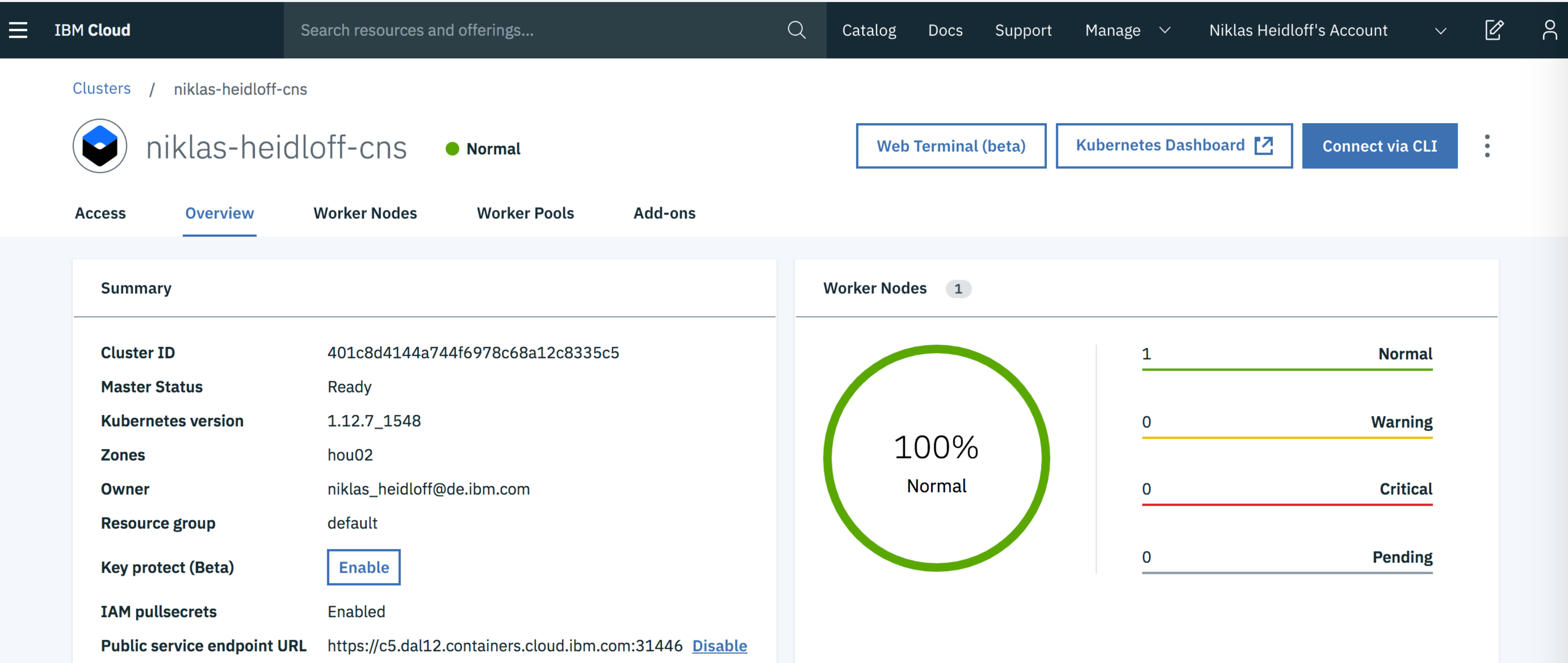


Reactive Java Microservices

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



Summary

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

Worker Nodes 1

100% Normal

1	Normal
0	Warning
0	Critical
0	Pending

Summary

Get the code →

Reactive systems improve user experiences and are more efficient

IBM loves open source

Kubernetes

OpenJ9

MicroProfile

Quarkus

IBM Developer

developer.ibm.com

IBM Cloud Lite account

ibm.biz/nheidloff

IBM