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

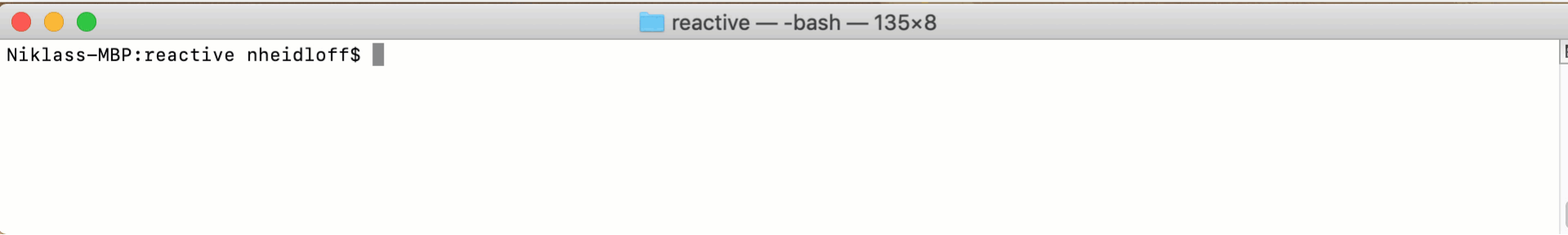
Reactive Web Application

Reactive REST Endpoints

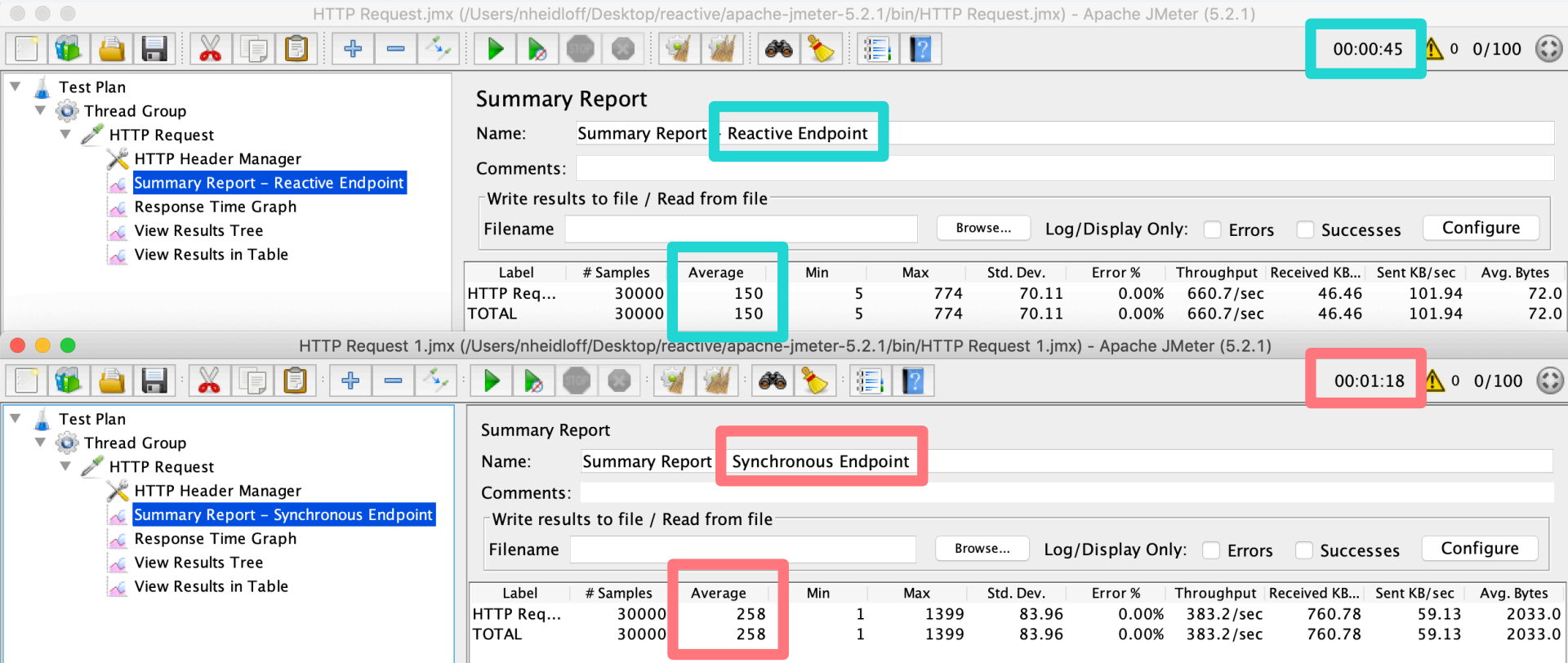
Reactive Web Application

Articles

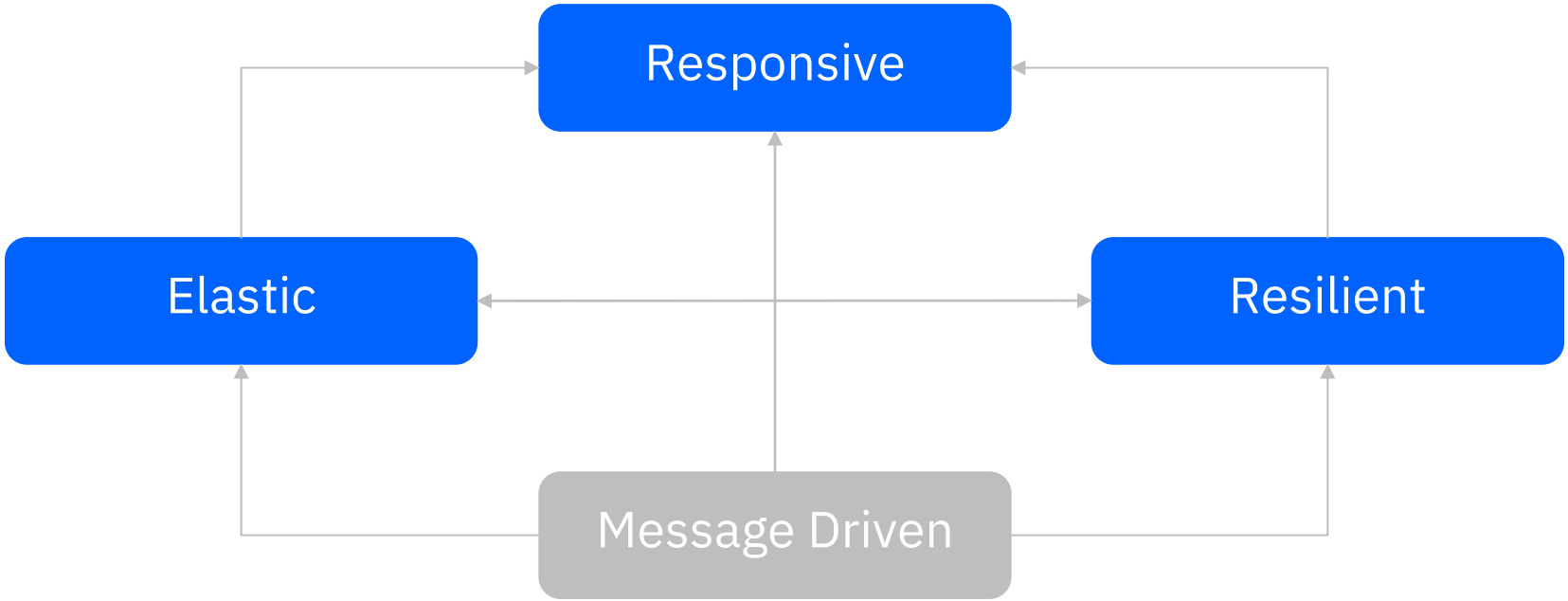
Title	Author	Twitter	Blog
Debugging Microservices running in Kubernetes	Niklas Heidloff	@nheidloff	Blog
Dockerizing Java MicroProfile Applications	Niklas Heidloff	@nheidloff	Blog
Install Istio and Kiali on IBM Cloud or Minikube	Harald Uebele	@harald_u	Blog
Three awesome TensorFlow.js Models for Visual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



Reactive REST Endpoints



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);

    /**
```

```
<Response> CompletionStage<Response> java.util.concurrent.CompletionStage.thenApply(Function<? super JsonArray, ? 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.

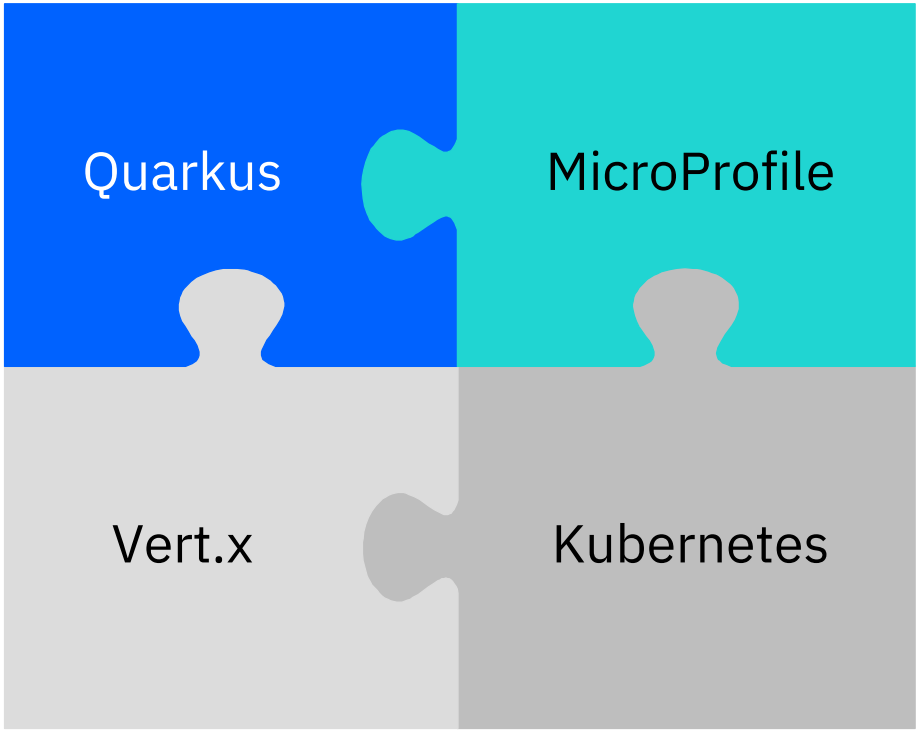
See the CompletionStage documentation for rules covering exceptional completion.

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

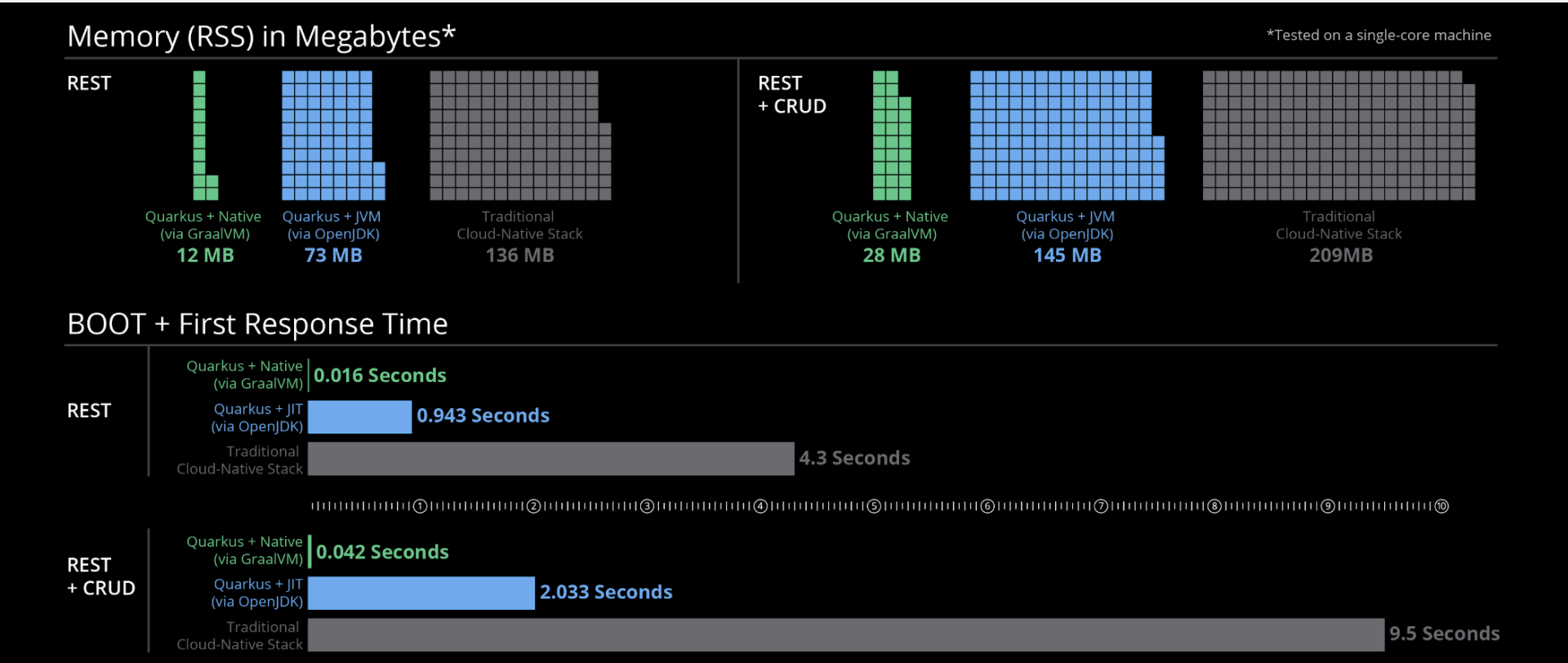
```
@GET
@Path("
@Produc
public
  Compl
  artic
    Jso
      .
      .
      r
    }).thenApply(jsonArray -> {
      return Response.ok(jsonArray).build();
    }).exceptionally(throwable -> {
      return Response.status(Response.Status.INTERNAL_SERVER_ERROR).build();
    }).whenComplete((response,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

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

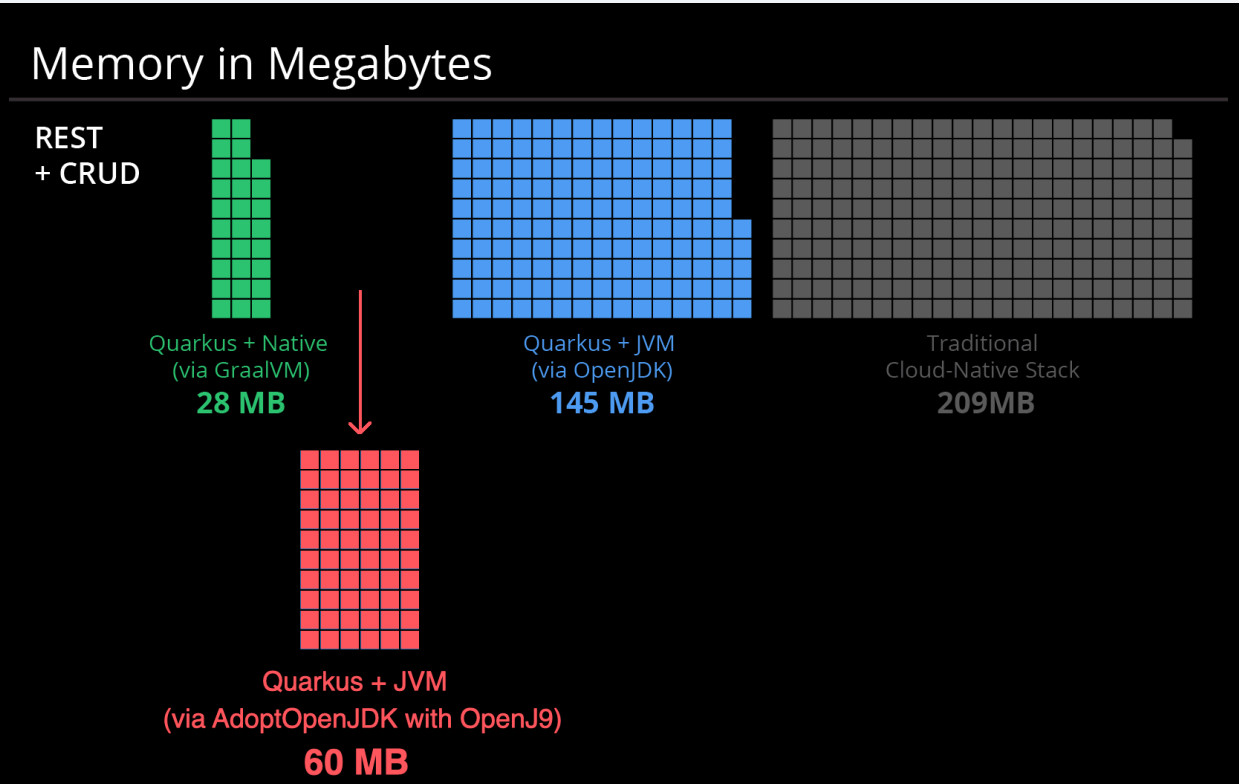
Technologies to build reactive Applications



Quarkus – Supersonic Subatomic Java



Quarkus using OpenJ9



“Optimizing Enterprise Java for a Microservices Architecture.“

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

`microprofile.io`

“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

kubernetes

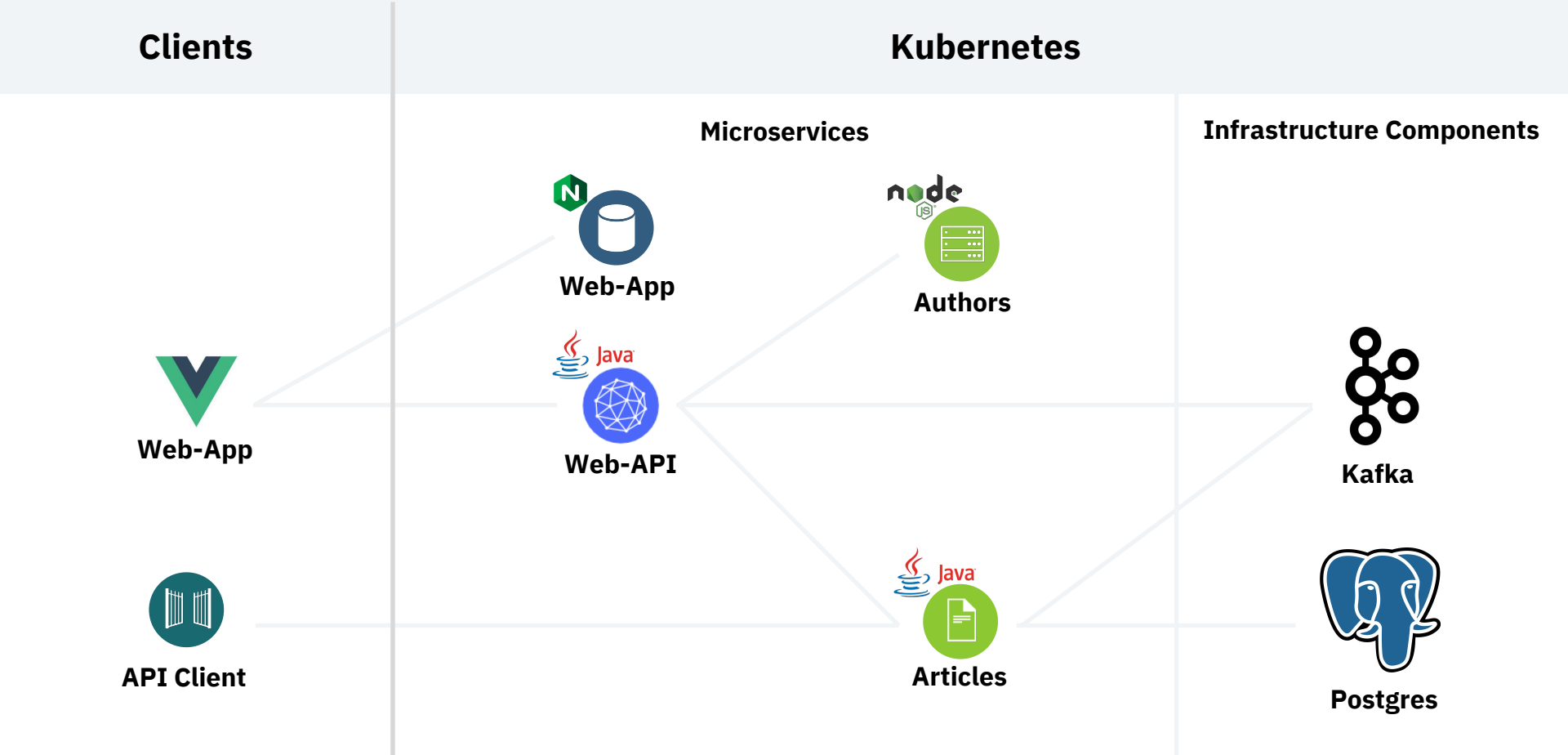
Example Application

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

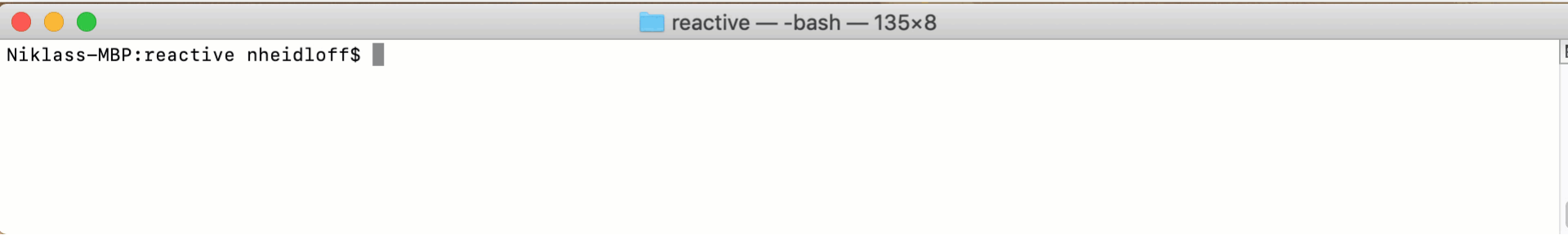
Architecture



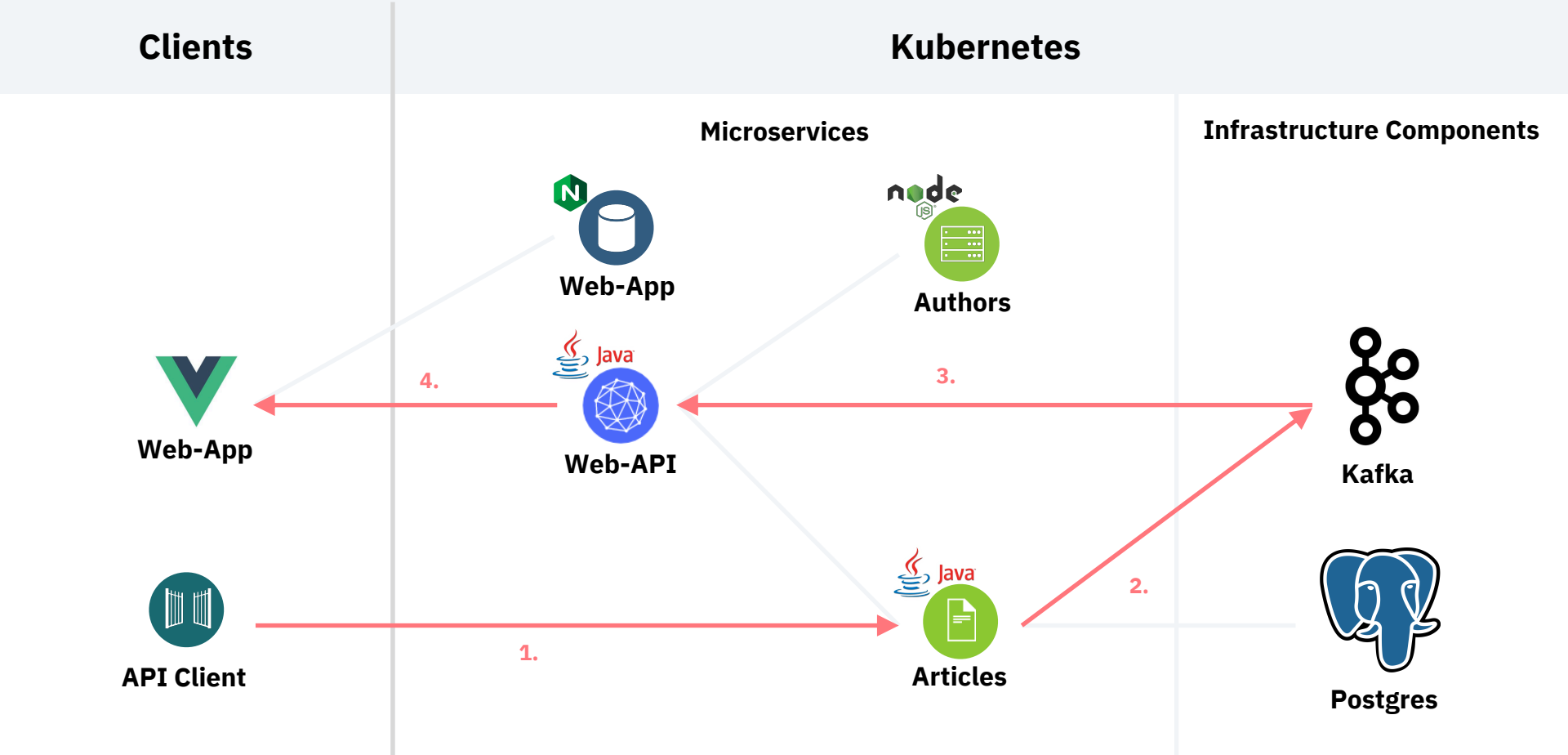
Reactive Web Application

Articles

Title	Author	Twitter	Blog
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



Notifications for Web Applications

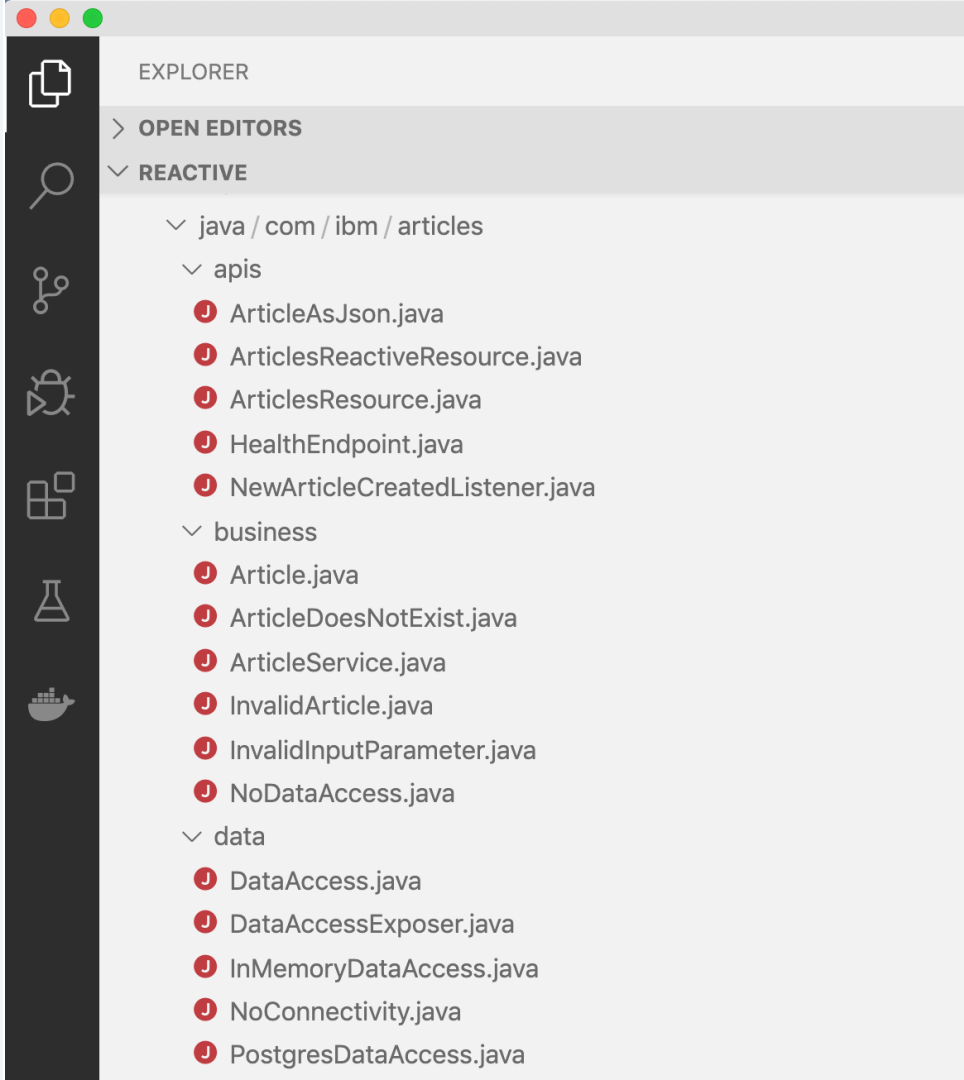


Clean Architecture

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

```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
    }
```

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

```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;
    }

}
```

Subscriber

Publisher

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

```
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;
    }
}
```

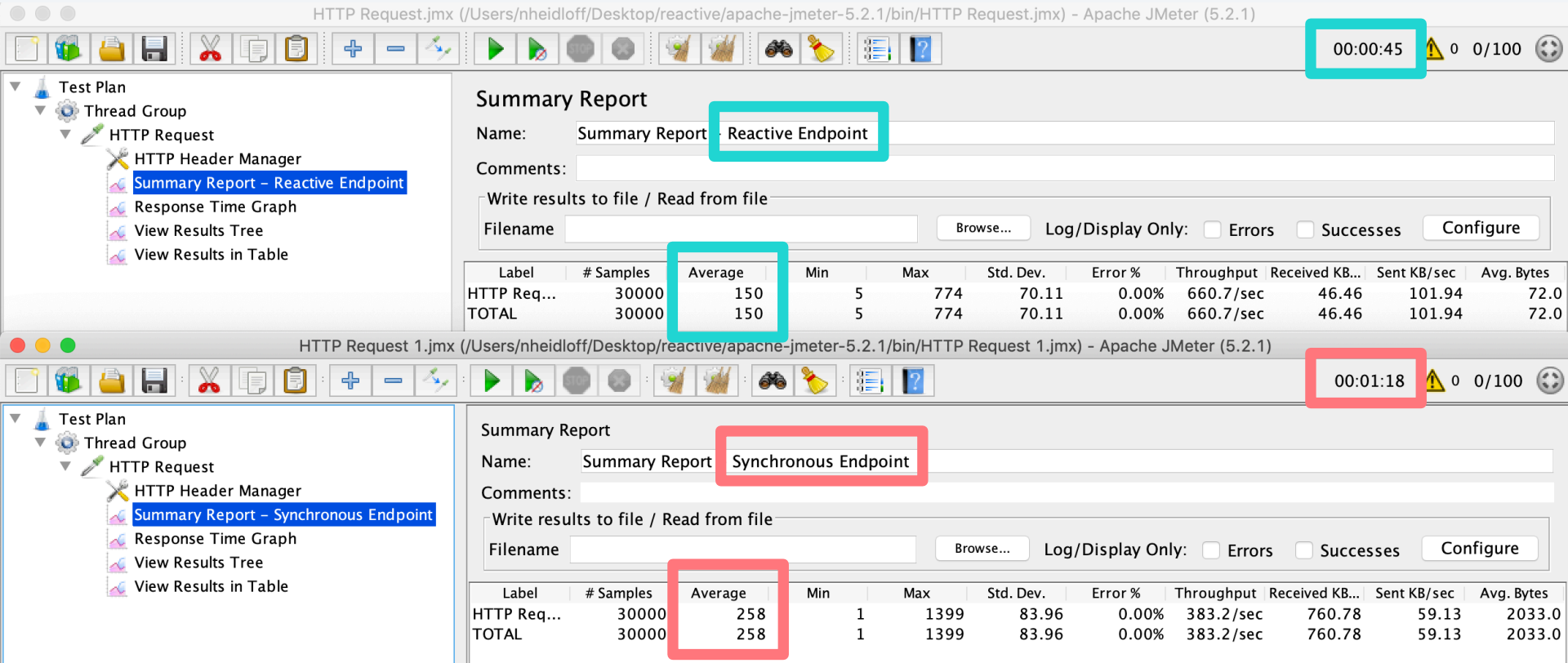
```
let url = this.$store.state.endpoints.api +
  "server-sent-events";
this.readArticles();
let source = new EventSource(url);
let that = this;
source.onmessage = function (event) {
  that.readArticles();
};
```

Server Sent Events

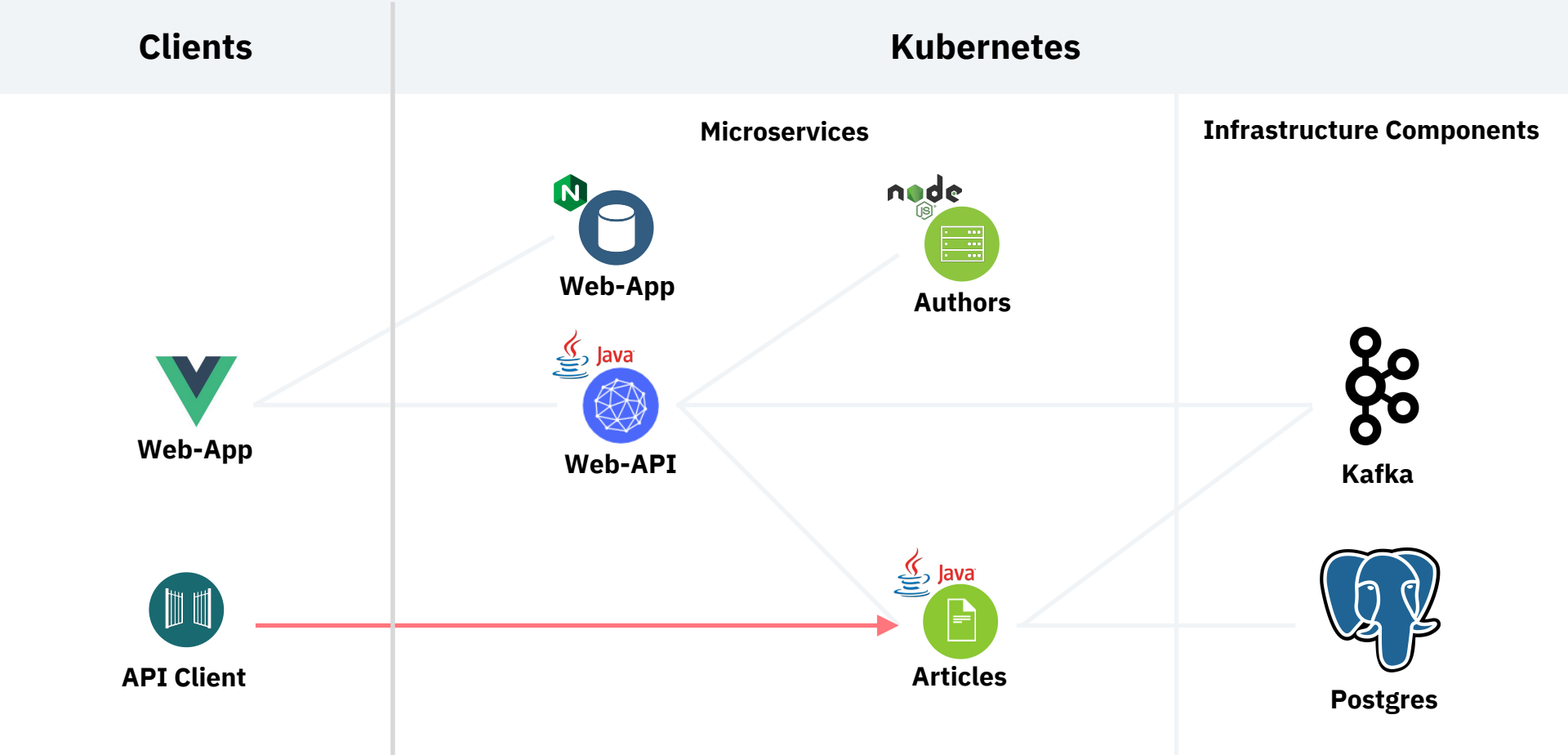
```java
import org.reactivestreams.Publisher;
import io.smallrye.reactive.messaging.annotations.Channel;
import org.jboss.resteasy.annotations.SseElementType;

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



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();
        });
}
```

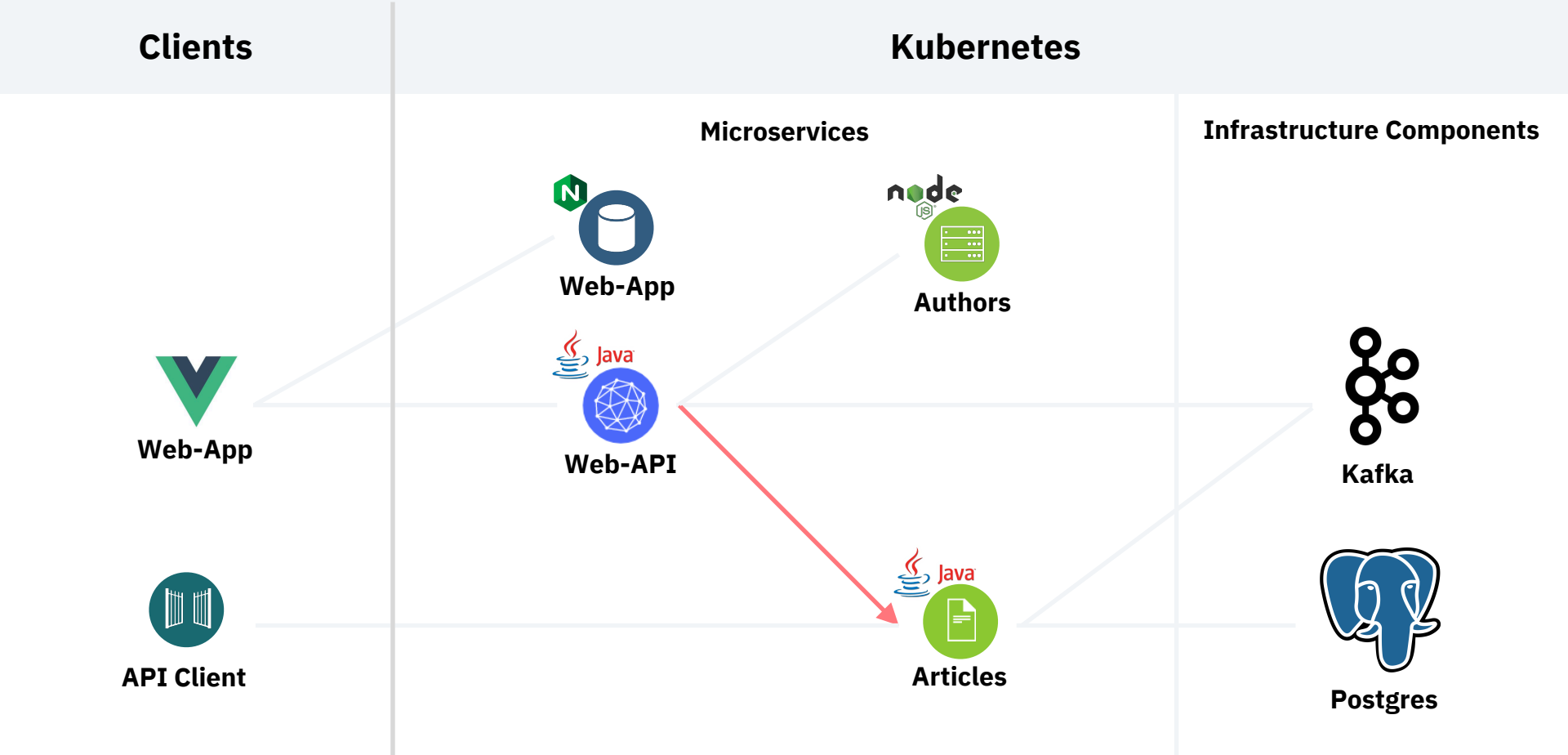
Timeouts

```
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

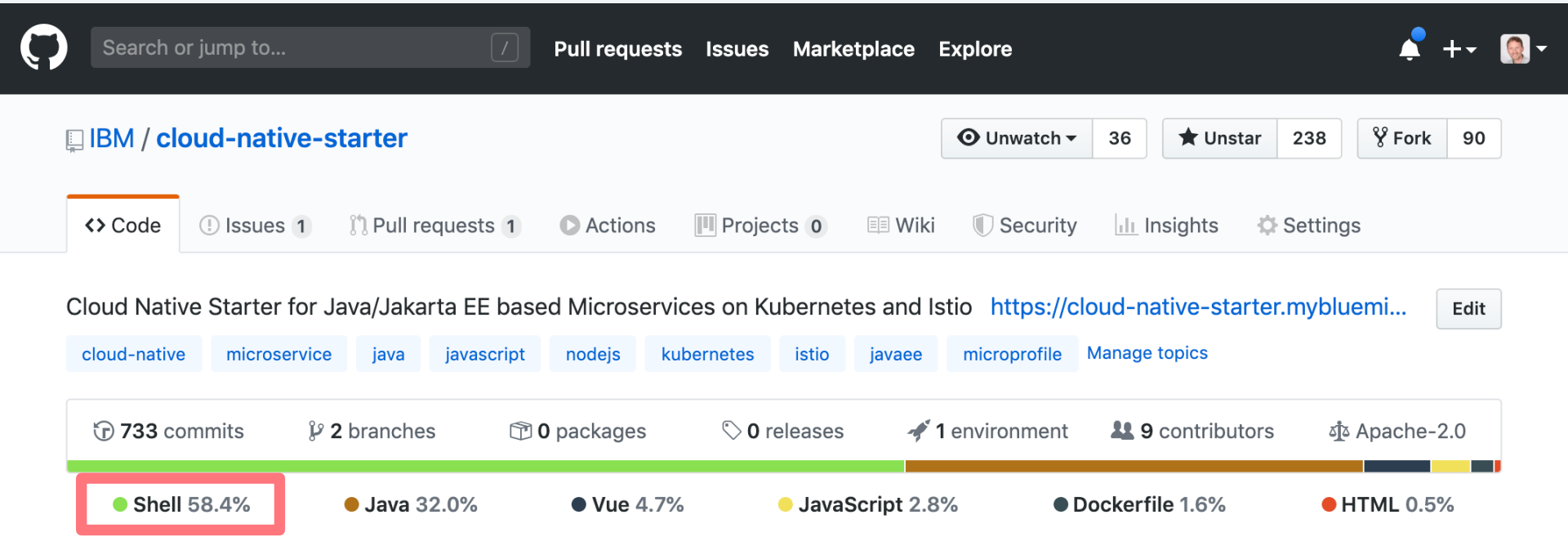
```
private ArticlesServiceReactive articlesServiceReactive;

@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

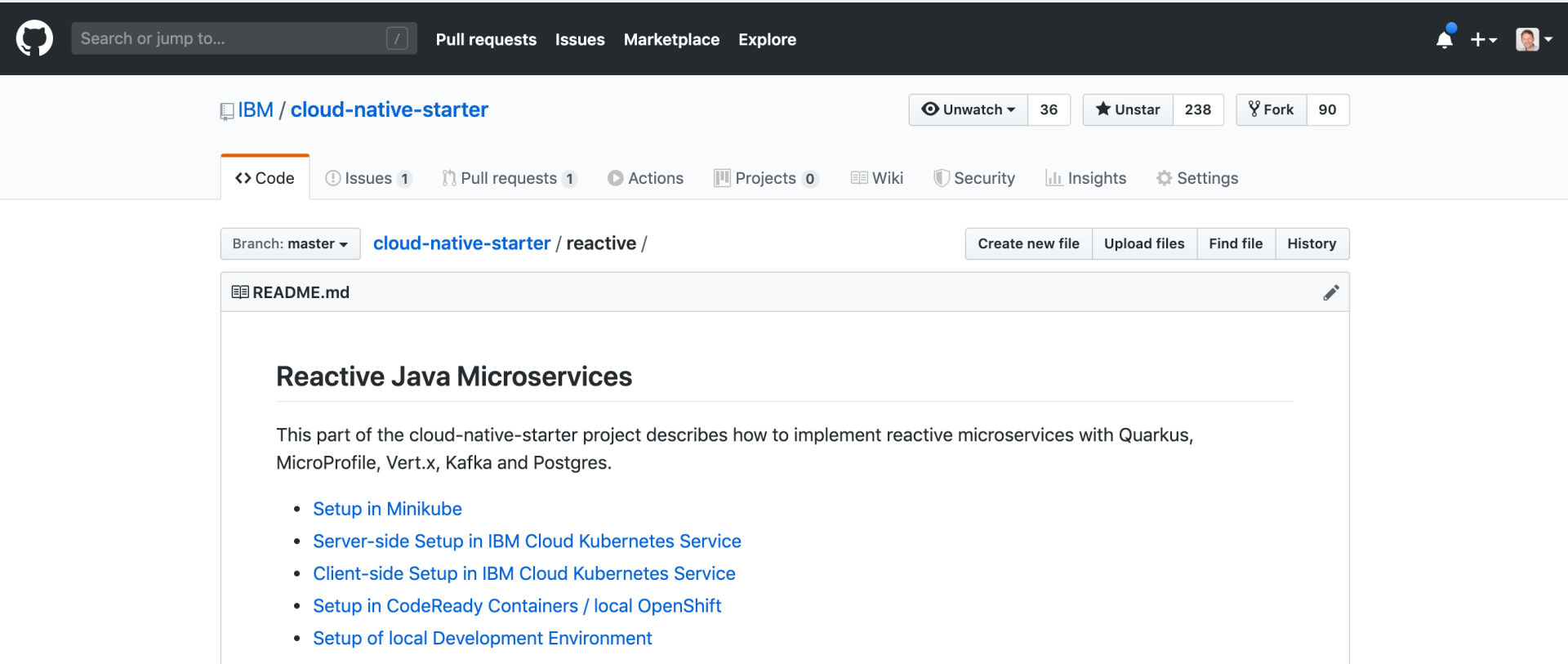
```
private ArticlesServiceReactive articlesServiceReactive;

@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!

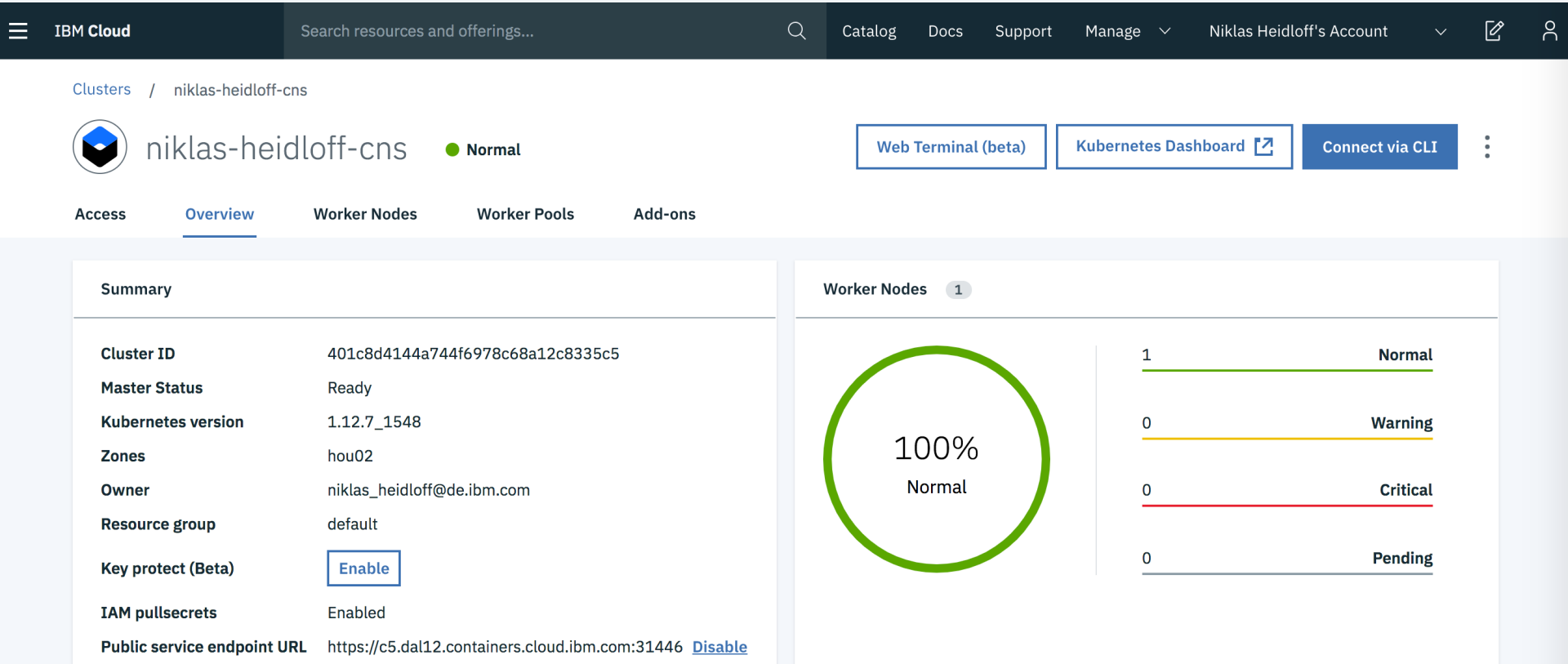
Focus on Developer Experience



Several Kubernetes Environments



IBM Cloud Kubernetes Service including Istio and Knative



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

IBM Cloud Lite account

ibm.biz/nheidloff