

# How to develop your first cloud-native Applications with Java

Niklas Heidloff  
Developer Advocate, IBM  
@nheidloff

Harald Uebel  
Developer Advocate, IBM  
@Harald\_U

“Microservices are a software development technique [...] that structures an application as a collection of loosely coupled services.”

Wikipedia

# What are cloud-native Applications?

Elasticity

→ App stays responsive

Continuous delivery

→ DevOps

Cloud-native applications provide new capabilities, which challenge developers

- New tasks
- Old tasks in new context

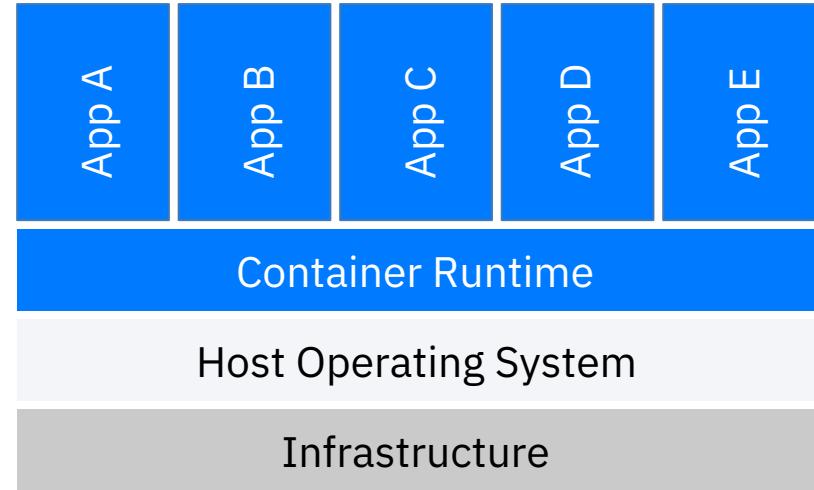
Service discovery and traffic management

Distributed monitoring, logging and tracing

Authentication and authorization between services

“A container image is a lightweight, standalone, executable package of software that includes everything needed to run an application.”

[docker.com](https://docker.com)



# Java Image

## Open source stack

OpenJ9 0.12.1

OpenJDK 8u202-b08 from AdoptOpenJDK

Open Liberty 18.0.0.4

MicroProfile 2.1

## Dockerfile

```
FROM open-liberty:microProfile2-java11
```

```
COPY liberty/server.xml /config/
```

```
COPY target/authors.war /config/apps/
```

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

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



# kubernetes

# Example Application

Cloud Native Starter

user@demo.email ▾

## 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](#)

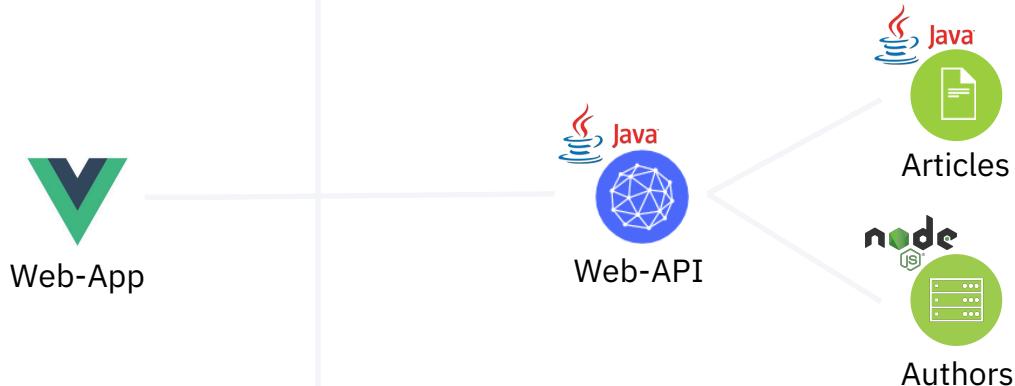
[Blog](#)

[Blog](#)

# Example Application – REST APIs

Browser

Kubernetes with Istio



# Exposing REST APIs

## JAX-RS

Java API for RESTful Web Services

### GetArticles.java

```
@RequestScoped
@Path("/v1")
@OpenAPIDefinition(info = @Info(title = "Web-API Service",
|   version = "1.0", description = "Web-API Service APIs"))
public class GetArticles {

    @Inject
    com.ibm.webapi.business.Service service;

    @GET
    @Path("/getmultiple")
    @Produces(MediaType.APPLICATION_JSON)
    @APIResponses(value = {
        @APIResponse(responseCode = "200",
            description = "Get most recently added articles",
            content = @Content(mediaType = "application/json",
                schema = @Schema(type = SchemaType.ARRAY,
                    implementation = Article.class))),
        @APIResponse(responseCode = "500",
            description = "Internal service error") })
    @Operation(summary = "Get most recently added articles",
        description = "Get most recently added articles")
    public Response getArticles() {
```

# Exposing REST APIs

**Open API** (formerly Swagger)

API description format for REST APIs

## GetArticles.java

```
@RequestScoped
@Path("/v1")
@OpenAPIDefinition(info = @Info(title = "Web-API Service",
|   version = "1.0", description = "Web-API Service APIs"))
public class GetArticles {

    @Inject
    com.ibm.webapi.business.Service service;

    @GET
    @Path("/getmultiple")
    @Produces(MediaType.APPLICATION_JSON)
    @APIResponses(value = {
        @APIResponse(responseCode = "200",
            description = "Get most recently added articles",
            content = @Content(mediaType = "application/json",
                schema = @Schema(type = SchemaType.ARRAY,
                    implementation = Article.class))),
        @APIResponse(responseCode = "500",
            description = "Internal service error") })
    @Operation(summary = "Get most recently added articles",
        description = "Get most recently added articles")
    public Response getArticles() {
```

# Exposing REST APIs

**Open API** (formerly Swagger)

API description format for REST APIs

The screenshot shows the Open Liberty Web-API Service interface. At the top, there's a logo and the text "Open Liberty". Below it, the title "Web-API Service" is displayed with a "1.0" badge and an "OAS3" badge. A sub-header "Web-API Service APIs" follows.

The main area is titled "Server" and shows a dropdown menu set to "http://192.168.99.100:31380/web-api".

A section titled "default" contains several API endpoints:

- POST /v1/create**: Create a new article.
- GET /v1/getmultiple**: Get most recently added articles.

Below these, under "Parameters", it says "No parameters". There are "Execute" and "Clear" buttons at the bottom of this section.

Under "Responses", there's a "Curl" section with the command:  
curl -X GET "http://192.168.99.100:31380/web-api/v1/getmultiple" -H "accept: application/json"

Further down, there's a "Request URL" field containing "http://192.168.99.100:31380/web-api/v1/getmultiple".

The "Server response" section shows a table with one row for code 200. The "Details" column for code 200 shows the "Response body" which contains a JSON array of articles:

```
[{"id": "1555051929394", "title": "Example Java App running in the Cloud via Kubernetes", "url": "http://heidloff.net/article/example-java-app-cloud-kubernetes", "authorName": "Niklas Heidloff", "authorBlog": "http://heidloff.net", "authorTwitter": "@nheidloff"}, {"id": "1555051929395", "title": "Java Application Containerization with Docker", "url": "http://heidloff.net/article/java-application-containerization-docker", "authorName": "Niklas Heidloff", "authorBlog": "http://heidloff.net", "authorTwitter": "@nheidloff"}]
```

# Consuming REST APIs

## MicroProfile Rest Client

Type-safe approach to invoke RESTful services

### AuthorsService.java

```
@RegisterProvider(ExceptionMapperArticles.class)
public interface AuthorsService {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Author getAuthor(String name) throws NonexistentAuthor;
}
```

### AuthorsServiceDataAccess.java

```
public class AuthorsServiceDataAccess implements AuthorsDataAccess {
    public AuthorsServiceDataAccess() {}

    static final String BASE_URL = "http://authors:3000/api/v1/";

    public Author getAuthor(String name) throws NoConnectivity, NonexistentAuthor {
        try {
            name = URLEncoder.encode(name, "UTF-8").replace("+", "%20");
            URL apiUrl = new URL(BASE_URL + "getauthor?name=" + name);
            AuthorsService customRestClient;
            customRestClient = RestClientBuilder.newBuilder().baseUrl(apiUrl)
                .register(ExceptionMapperAuthors.class).build(AuthorsService.class);

            Author output = customRestClient.getAuthor(name);
            return output;
        } catch (NonexistentAuthor e) {
            e.printStackTrace();
            throw new NonexistentAuthor(e);
        } catch (Exception e) {
            throw new NoConnectivity(e);
        }
    }
}
```

# Consuming REST APIs

## MicroProfile Rest Client

Type-safe approach to invoke RESTful services

### AuthorsService.java

```
@RegisterProvider(ExceptionMapperArticles.class)
public interface AuthorsService {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Author getAuthor(String name) throws NonexistentAuthor;
}
```

### AuthorsServiceDataAccess.java

```
public class AuthorsServiceDataAccess implements AuthorsDataAccess {
    public AuthorsServiceDataAccess() {}

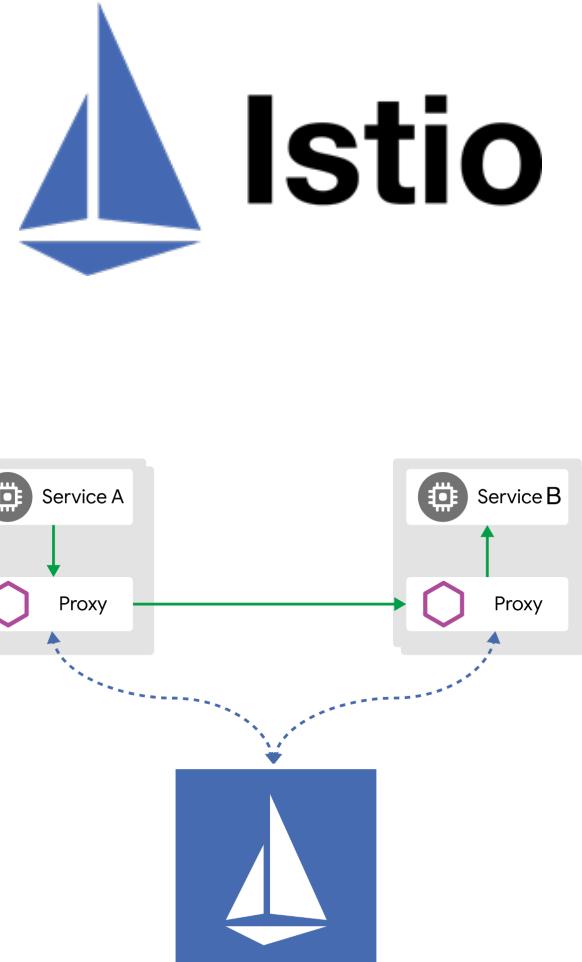
    static final String BASE_URL = "http://authors:3000/api/v1/";

    public Author getAuthor(String name) throws NoConnectivity, NonexistentAuthor {
        try {
            name = URLEncoder.encode(name, "UTF-8").replace("+", "%20");
            URL apiUrl = new URL(BASE_URL + "getauthor?name=" + name);
            AuthorsService customRestClient;
            customRestClient = RestClientBuilder.newBuilder().baseUrl(apiUrl)
                .register(ExceptionMapperAuthors.class).build(AuthorsService.class);

            Author output = customRestClient.getAuthor(name);
            return output;
        } catch (NonexistentAuthor e) {
            e.printStackTrace();
            throw new NonexistentAuthor(e);
        } catch (Exception e) {
            throw new NoConnectivity(e);
        }
    }
}
```

“Istio is an open platform for providing a uniform way to integrate microservices, manage traffic flow across microservices, enforce policies and aggregate telemetry data.”

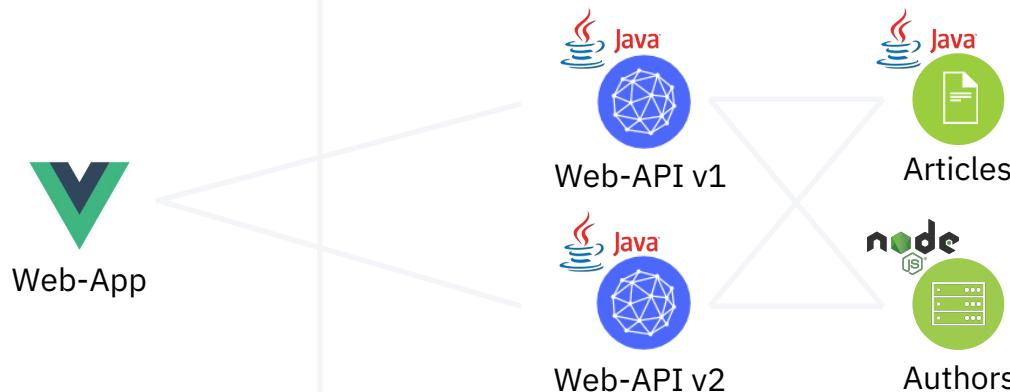
[github.com/istio/istio](https://github.com/istio/istio)



# Example Application – Traffic Management

Browser

Kubernetes with Istio



# Traffic Management – Web API Version 1

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 Uebel

Niklas Heidloff

Niklas Heidloff



Twitter

[@nheidloff](#)

[@nheidloff](#)

[@harald\\_u](#)

[@nheidloff](#)

[@nheidloff](#)



Blog

[Blog](#)

[Blog](#)

[Blog](#)

[Blog](#)

[Blog](#)

# Traffic Management – Web API Version 2

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](#)

[Three Minutes Demo of Blue Cloud Mirror](#)

[IBM announced Managed Istio and Managed Knative](#)

[Developing and debugging Microservices with Java](#)

[Recent Java Updates from IBM](#)

[Blue Cloud Mirror — \(Don't\) Open The Doors!](#)



Author

Niklas Heidloff

Niklas Heidloff

Harald Uebel

Niklas Heidloff

Niklas Heidloff

Niklas Heidloff

Niklas Heidloff

Niklas Heidloff

Niklas Heidloff

Harald Uebel



Twitter

[@nheidloff](#)

[@nheidloff](#)

[@harald\\_u](#)

[@nheidloff](#)

[@nheidloff](#)

[@nheidloff](#)

[@nheidloff](#)

[@nheidloff](#)

[@nheidloff](#)

[@harald\\_u](#)



Blog

[Blog](#)

# Traffic Management

Example: 80% / 20% splitting

## ingress.yaml

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: virtualservice-ingress-web-api-web-app
spec:
  hosts:
  - "*"
  gateways:
  - default-gateway-ingress-http
  http:
  - match:
    - uri:
        prefix: /web-api/v1/getmultiple
      route:
      - destination:
          host: web-api
          subset: v1
          weight: 80
      - destination:
          host: web-api
          subset: v2
          weight: 20
```

# Traffic Management Demo

**kiali**

Namespace: default

Graph [?](#)

Display [?](#) Edge Labels [?](#) Graph Type Versioned app [?](#) Find... [x](#) Hide... [x](#) [?](#) Fetching Last min [?](#) Every 5 sec [?](#)

Apr 11, 11:25:29 ... Apr 11, 11:26:29

The graph displays the following traffic distribution:

- From istio-ingressgateway (istio-system) to web-app: 85.7% (OK)
- From istio-ingressgateway (istio-system) to web-api: 14.3% (OK)
- From web-app to v1 (web): 100% (OK)
- From web-api to v1 (web): 80% (OK)
- From web-api to v2: 20% (OK)
- From v1 (web) to articles: 16.7% (OK)
- From v1 (web) to authors: 83.3% (OK)
- From v2 to v1 (articles): 8.8% (OK)
- From v2 to v1 (authors): 91.2% (OK)
- From articles to v1 (articles): 100% (OK)
- From authors to v1 (authors): 100% (OK)

Namespace: default  
applications, services, workloads

Current Graph:  
6 apps  
4 services  
11 edges

HTTP Traffic (requests per second):  
Total %Success %Error  
6.65 100.00 0.00

0 25 50 75 100 %

OK 3xx 4xx 5xx

# “Optimizing Enterprise Java for a Microservices Architecture.

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

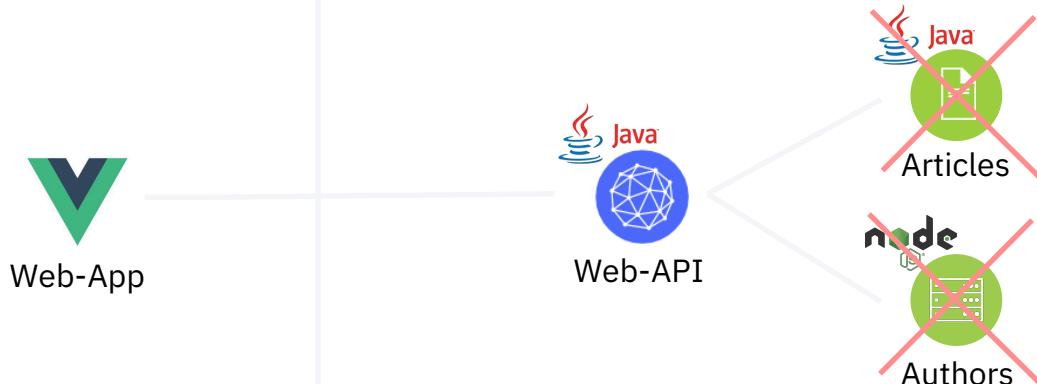
[micrometer.io](https://micrometer.io)



# Example Application – Resiliency

Browser

Kubernetes with Istio



# Resiliency

Authors service not available

## Usage of default values

### Service.java

```
private List<Article> lastReadArticles;

public List<Article> fallbackNoArticlesService() {
    return lastReadArticles;
}

@Fallback(fallbackMethod = "fallbackNoArticlesService")
public List<Article> getArticles() throws NoDataAccess {

    List<Article> articles = new ArrayList<Article>();
    List<CoreArticle> coreArticles = new ArrayList<CoreArticle>();

    try {
        coreArticles = DataAccessManager.getArticlesDataAccess().getArticles(5);
    } catch (NoConnectivity e) {
        throw new NoDataAccess(e);
    }

    for (int index = 0; index < coreArticles.size(); index++) {
        CoreArticle coreArticle = coreArticles.get(index);
        Article article = new Article(coreArticle.id, coreArticle.title,
            coreArticle.title, coreArticle.author);
        try {
            Author author;
            author = DataAccessManager.getAuthorsDataAccess().getAuthor(coreArticle.author);
            article.authorBlog = author.blog;
            article.authorTwitter = author.twitter;
        } catch (Exception e) {
            article.authorBlog = "";
            article.authorTwitter = "";
        }
        articles.add(article);
    }
}
```

# Resiliency

Articles service not available

## MicroProfile Fallback annotation

### Service.java

```
private List<Article> lastReadArticles;

public List<Article> fallbackNoArticlesService() {
    return lastReadArticles;
}

@Fallback(fallbackMethod = "fallbackNoArticlesService")
public List<Article> getArticles() throws NoDataAccess {

    List<Article> articles = new ArrayList<Article>();
    List<CoreArticle> coreArticles = new ArrayList<CoreArticle>();

    try {
        coreArticles = DataAccessManager.getArticlesDataAccess().getArticles(5);
    } catch (NoConnectivity e) {
        throw new NoDataAccess(e);
    }

    for (int index = 0; index < coreArticles.size(); index++) {
        CoreArticle coreArticle = coreArticles.get(index);
        Article article = new Article(coreArticle.id, coreArticle.title,
        coreArticle.title, coreArticle.author);
        try {
            Author author;
            author = DataAccessManager.getAuthorsDataAccess().getAuthor(coreArticle.author);
            article.authorBlog = author.blog;
            article.authorTwitter = author.twitter;
        } catch (Exception e) {
            article.authorBlog = "";
            article.authorTwitter = "";
        }
        articles.add(article);
    }
}
```

# Resiliency Demo

Cloud Native Starter

user@demo.email ▾

## 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](#)

# Resiliency Demo

Cloud Native Starter

user@demo.email ▾

## 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



Blog

# Authentication and Authorization

## OpenID Connect

Identity layer on top of the OAuth 2.0 protocol

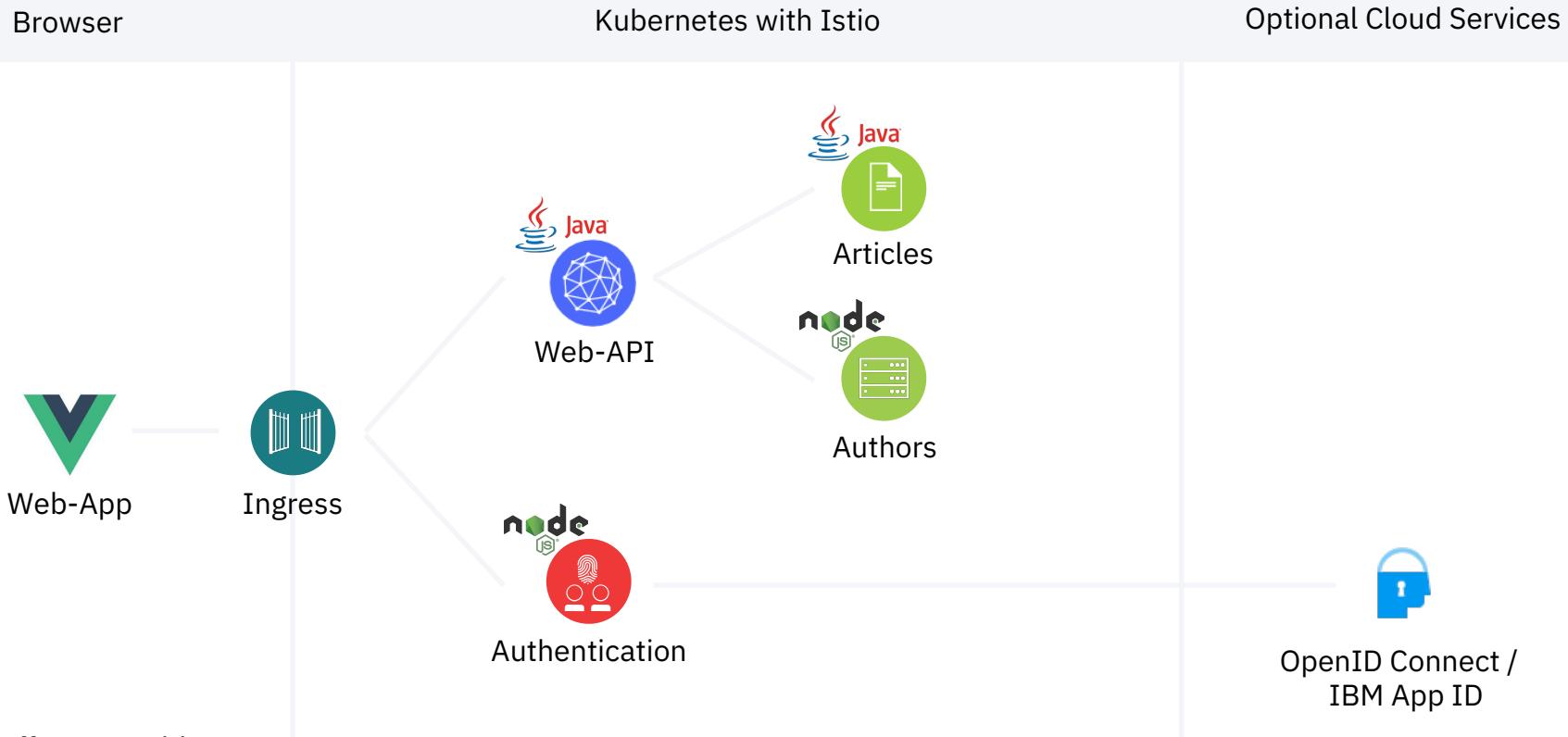
## IBM App ID

IBM service to authenticate users and protect APIs

## policy.yaml

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "protect-web-api"
spec:
  targets:
  - name: web-api
  origins:
  - jwt:
      issuer: "https://us-south.appid.cloud.ibm.com/oauth/v4/xxx"
      jwksUri: "https://us-south.appid.cloud.ibm.com/oauth/v4/xxx/publickeys"
      trigger_rules:
      - included_paths:
          - exact: /web-api/v1/create
  principalBinding: USE_ORIGIN
```

# Example Application – Authentication and Authorization



# Authentication



Email:

A text input field for entering an email address. It features a small icon of a person in a grey box to its left. The placeholder text "user@demo.email" is visible inside the input field.

Password:

A password input field. It features a small icon of a lock in a grey box to its left. The placeholder text "....." is visible inside the input field.

[Forgot Password?](#)

**Login**

Don't have an account? [Sign up](#)

# Authorization

Cloud Native Starter

user@demo.email ▾

## Create new Article

Title:

Example Java App running in the Cloud via Kubernetes

URL:

<http://heidloff.net/article/example-java-app-cloud-kubernetes>

Author:

Niklas Heidloff

Submit

Show Articles

# Authorization

Cloud Native Starter user@demo.email ▾

Articles

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

Network

Filter Hide data URLs

All XHR JS CSS Img Media Font Doc WS Manifest Other

Name	Headers	Preview	Response	Timing
auth				
callback?code=w...				
loginwithtoken?na...				
app.9cd06f5a.css				
chunk-vendors.73...				
app.bfe2e0c6.js				
chunk-vendors.3e...				
getmultiple				
create				
getmultiple				

General

Request URL: <http://192.168.99.100:31380/web-api/v1/create>  
Request Method: POST  
Status Code: 201 Created  
Remote Address: 192.168.99.100:31380  
Referrer Policy: no-referrer-when-downgrade

Response Headers

access-control-allow-credentials: true  
access-control-allow-headers: origin, content-type, accept, aut...  
access-control-allow-methods: GET, POST, PUT, DELETE, OPTIONS,  
access-control-allow-origin: \*  
content-language: en-US  
content-length: 182  
content-type: application/json  
date: Fri, 12 Apr 2019 06:52:09 GMT  
server: istio-envoy  
x-envoy-upstream-service-time: 346  
x-powered-by: Servlet/4.0

Request Headers

⚠ Provisional headers are shown

Accept: application/json, text/plain, \*/\*  
Authorization: Bearer eyJhbGci...  
KLTrmMTQ2NGZnLmNKOTMNdMS5i...  
Iy0jM1ljk5NCi...InZlcii6NH0.e...  
3VKLmlibS5jb20vb2f1dGvdjQvN...  
ZTY0liwiyXVkiobImVLOtYxNDcZL...

# Authorization

## Via MicroProfile

### server.xml

```
<server description="OpenLiberty Server">

    <featureManager>
        <feature>webProfile-8.0</feature>
        <feature>microProfile-2.1</feature>
        <feature>usr:opentracingZipkin-0.31</feature>
        <feature>monitor-1.0</feature>
        <feature>mpJwt-1.1</feature>
        <feature>appSecurity-3.0</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint" host="*"
        httpPort="9080" httpsPort="9443"/>

    <mpMetrics authentication="false"/>

    <mpJwt id="jwt"
        issuer="https://us-south.appid.cloud.ibm.com/oauth/v4/xxx"
        jwksUri="https://us-south.appid.cloud.ibm.com/oauth/v4/xxx/publickeys"
        userNameAttribute="sub"
        audiences="ALL_AUDIENCES"/>

    <sslDefault sslRef="RpSSLConfig"/>
    <ssl id="RpSSLConfig" keyStoreRef="defaultKeyStore"
        trustStoreRef="validationKeystore"/>
    <keyStore id="defaultKeyStore" location="keystore.jceks"
        type="JCEKS" password="secret" />
    <keyStore id="validationKeystore" location="/config/key.jks"
        type="jks" password="changeit"/>
</server>
```

# Authorization

## Via MicroProfile

### Manage.java

```
@RequestScoped  
@Path("/v1")  
public class Manage {  
  
    @Inject  
    private JsonWebToken jwtPrincipal;  
  
    @POST  
    @Path("/manage")  
    @Produces(MediaType.APPLICATION_JSON)  
    @Operation(summary = "Manage app", description = "Manage app")  
    public Response manage() {  
        System.out.println("com.ibm.web-api.apis.Manage.manage");  
        System.out.println(this.jwtPrincipal);  
  
        String principalEmail = this.jwtPrincipal.getClaim("email");  
        if (principalEmail.equalsIgnoreCase("admin@demo.email")) {  
            JsonObject output = Json.createObjectBuilder()  
                .add("message", "success").build();  
            return Response.ok(output).build();  
        }  
    }  
}
```

# Observability

Tracing

Logging

Monitoring

Metrics

Healthchecks

Microservices vs monolith

→ Higher complexity

→ Ephemeral

Chained invocations

Kubernetes

→ 1 service = N pods

# Example Application – Distributed Logging

Browser

Kubernetes with Istio



# Distributed Logging

The screenshot shows a distributed logging interface with the following elements:

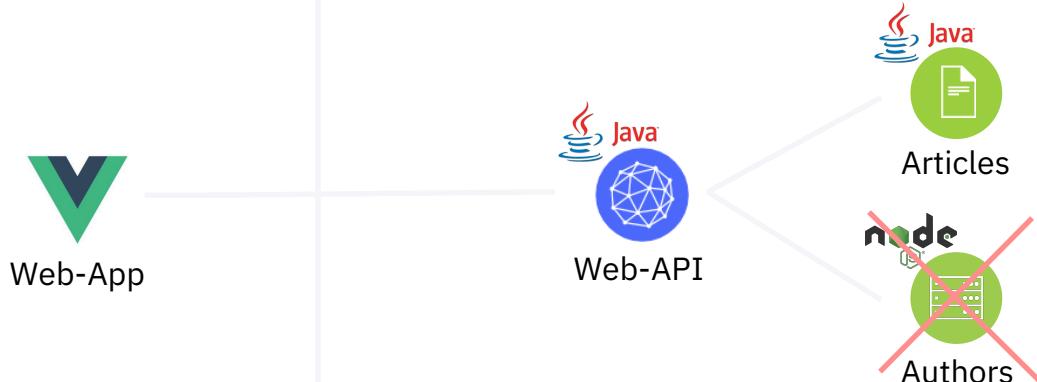
- Left Sidebar:** Icons for Find a View, EVERYTHING, VIEWS, CloudNative, and various monitoring and configuration tools.
- Top Bar:** Filter dropdowns for CloudNative, All Tags, All Sources, 4 Apps, and All Levels.
- Log List:** A list of log entries from different services. One entry is highlighted with a blue border:

```
May 16 14:44:24 articles-76bbfcdfb7-mqxm4 articles com.ibm.articles.apis.GetArticles (not retained)
May 16 14:44:24 web-api-v1-fc5f475c5-f5vlp web-api com.ibm.web-api.apis.GetArticles.getArticles (not retained)
May 16 14:44:24 authors-6d48cff748-gjzh8 authors [2019-05-16T12:44:24.367] [INFO] authors_service - Query
for: Niklas Heidloff (not retained)
```
- Bottom Bar:** Includes a search input, a "Jump to timeframe" button, and other navigation controls.

# Example Application – Distributed Logging

Browser

Kubernetes with Istio



# Distributed Logging

The screenshot shows a distributed logging interface with the following interface elements:

- Left Sidebar:** Icons for Find a View, EVERYTHING, VIEWS, CloudNative, and various monitoring metrics.
- Top Bar:** Filter dropdowns for CloudNative, All Tags, All Sources, 4 Apps, and All Levels.
- Log List:** A list of log entries. The first five entries are from the `web-api` service, and the subsequent entries are from the `articles` service. The log entries all indicate a failure to connect to the `authors` service.
- Selected Log Entry:** The third log entry from the `articles` service is highlighted with a blue border.
- Bottom Bar:** Includes a search bar, a "Jump to timeframe" button, and other navigation controls.

Time	Service	Source	Level	Message
May 16 14:48:48	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:48	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:48	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:48	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:48	articles-76bbfcdfb7-mqxm4	articles	com.ibm.articles.apis.GetArticles	(not retained)
May 16 14:48:49	web-api-v1-fc5f475c5-f5vlp	web-api	com.ibm.web-api.apis.GetArticles	.getArticles (not retained)
May 16 14:48:49	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:49	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:49	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:49	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:49	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:49	web-api-v1-fc5f475c5-f5vlp	web-api	stderr	com.ibm.webapi.business.getArticles: Cannot connect to authors service (not retained)
May 16 14:48:49	articles-76bbfcdfb7-mqxm4	articles	com.ibm.articles.apis.GetArticles	(not retained)

# Tracing

## OpenTracing

Vendor-neutral APIs and instrumentation for distributed tracing

## Jaeger and Zipkin

Open source distributed tracing systems

## server.xml

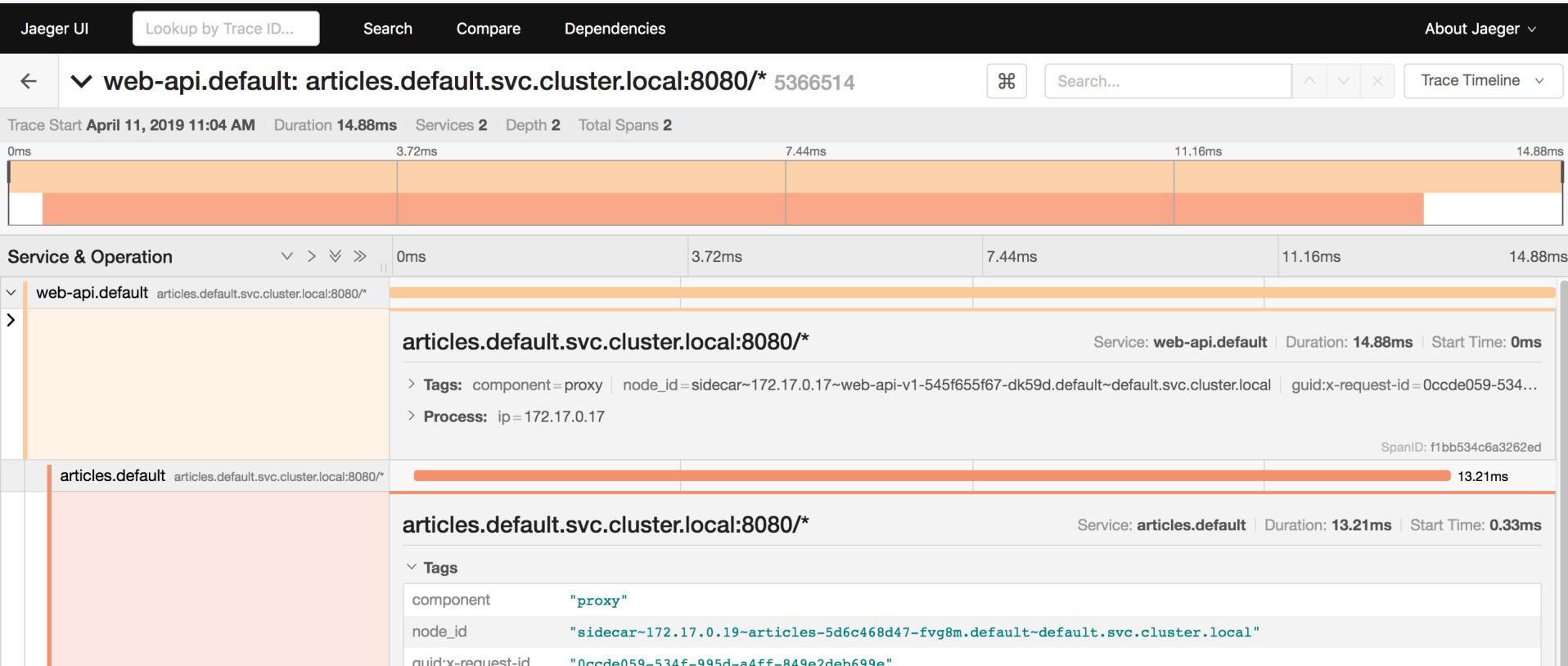
```
<?xml version="1.0" encoding="UTF-8"?>
<server description="OpenLiberty Server">

    <featureManager>
        <feature>webProfile-8.0</feature>
        <feature>microProfile-2.1</feature>
        <feature>usr:opentracingZipkin-0.31</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint" host="*"
        httpPort="8080" httpsPort="9443"/>

</server>
```

# Distributed Tracing



# Metrics

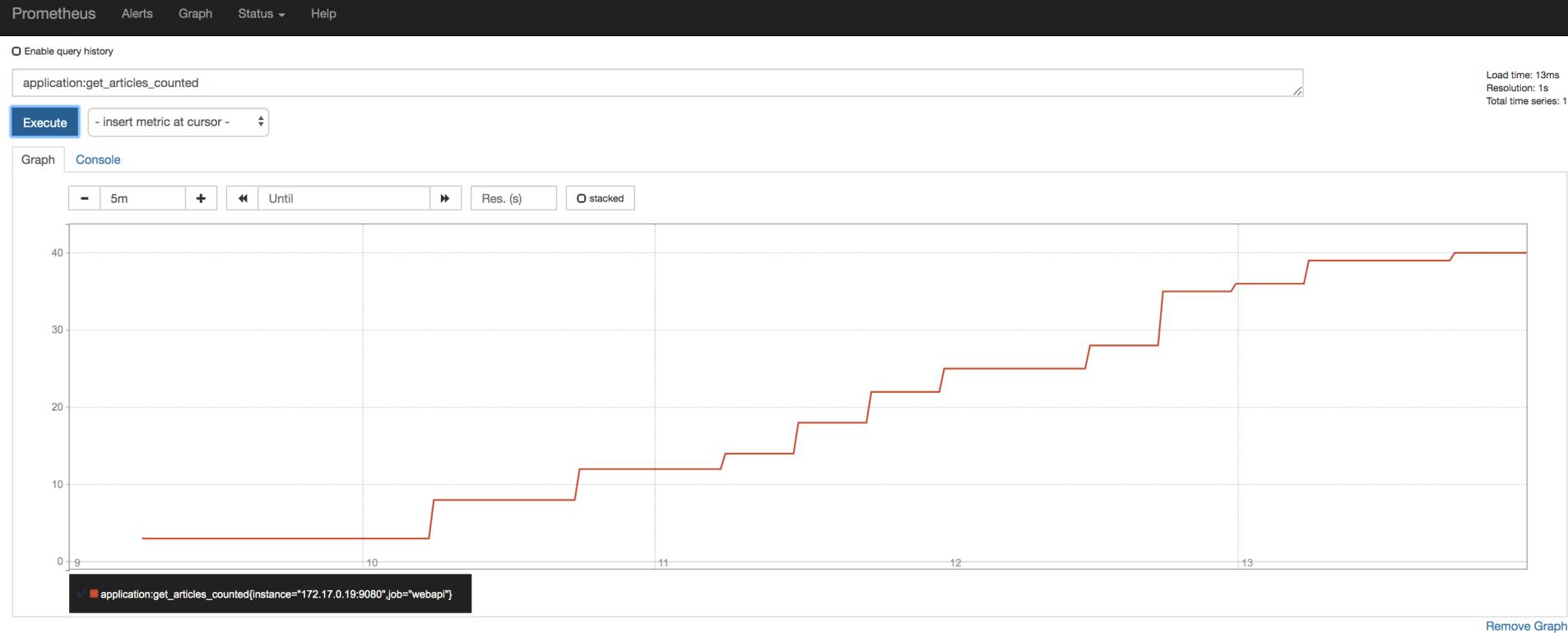
## Prometheus

Monitoring system and time series database

### GetArticles.java

```
@Timed(name = "getArticlesTimed",
        absolute = true,
        displayName = "web-api /getmultiple timer",
        description = "Time taken by getArticles")
@Counted(name = "getArticlesCounted",
        absolute = true,
        displayName = "web-api /getmultiple count",
        description = "Number of times getArticles has been invoked",
        monotonic = true)
@Metered(name = "getArticlesMetered",
        displayName = "web-api /getmultiple frequency",
        description = "Rate the throughput of getArticles")
@GET
@Path("/getmultiple")
@Produces(MediaType.APPLICATION_JSON)
public Response getArticles() {
```

# Metrics



# Sysdig

 Dashboards

 Search dashboards

 My Dashboards 

- Istio 1.0 Overview
- Istio 1.0 Service

 My Shared Dashboards 

- HTTP Overview
- Network Overview
- Overview by Container
- Overview by Host
- Overview by Process
- Top Processes

 Dashboards Shared With Me 

- No dashboards

 Add Dashboard

 EXPLORE

 DASHBOARDS

 ALERTS

 EVENTS

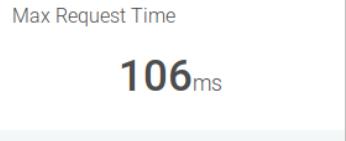
 CAPTURES

## HTTP Overview

Everywhere

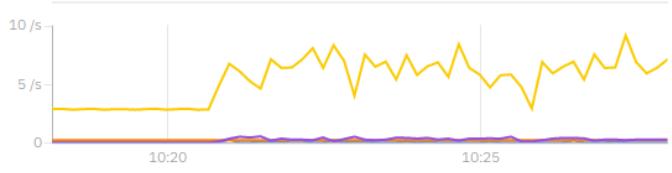
  
Request Count **11.4/s**

  
HTTP Error Count **0.21/s**

  
Average Request Time **6.90ms**

  
Max Request Time **106ms**

### Status Codes Over Time



### Average and Max Request Time



# Healthchecks

## MicroProfile Health

Liveness probes and readiness probes

@nheidloff @Harald\_U

### HealthEndpoint.java

```
@Health
@ApplicationScoped
public class HealthEndpoint implements HealthCheck {

    @Override
    public HealthCheckResponse call() {
        return HealthCheckResponse.named("web-api").withData("web-api", "ok").up().build();
    }
}
```

### Service.yaml

```
kind: Deployment
apiVersion: apps/v1beta1
metadata:
  name: web-api-v1
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: web-api
        version: v1
    spec:
      containers:
        - name: web-api
          image: web-api:1
          ports:
            - containerPort: 9080
          livenessProbe:
            exec:
              command: ["sh", "-c", "curl -s http://localhost:9080/"]
            initialDelaySeconds: 20
          readinessProbe:
            exec:
              command: ["sh", "-c", "curl -s http://localhost:9080/health | grep -q web-api"]
            initialDelaySeconds: 40
  restartPolicy: Always
```

# Persistence

Via JPA

## server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="OpenLiberty Server">

    <featureManager>
        <feature>webProfile-8.0</feature>
        <feature>microProfile-2.1</feature>
        <feature>usr:opentracingZipkin-0.31</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint" host="*"
                  httpPort="8080" httpsPort="9443"/>

    <library id="DB2JCCLib">
        <fileset dir="${shared.resource.dir}"
                 includes="jcc*.jar"/>
    </library>

    <dataSource id="articlejpadatasource"
               jndiName="jdbc/articlejpadatasource">
        <jdbcDriver libraryRef="DB2JCCLib" />
        <properties.db2.jcc databaseName="BLUDB"
                            portNumber="50000"
                            serverName="DB2-SERVER"
                            user="DB2-USER"
                            password="DB2-PASSWORD" />
    </dataSource>
</server>
```

# Persistence

Via JPA

## ArticleEntity.java

```
@Entity(name = "Article")
@Table(name = "Article")
@NamedQuery(name = "Article.findAll", query = "SELECT e FROM Article e")
@NamedQuery(name = "Article.findArticle", query = "SELECT a FROM Article a WHERE "
    + "a.title = :title AND a.url = :url AND a.author = :author")
public class ArticleEntity implements Serializable {
    private static final long serialVersionUID = 1L;

    @GeneratedValue(strategy = GenerationType.AUTO)
    @Id
    @Column(name = "articleId")
    private int id;

    @Column(name = "articleTitle")
    private String title;

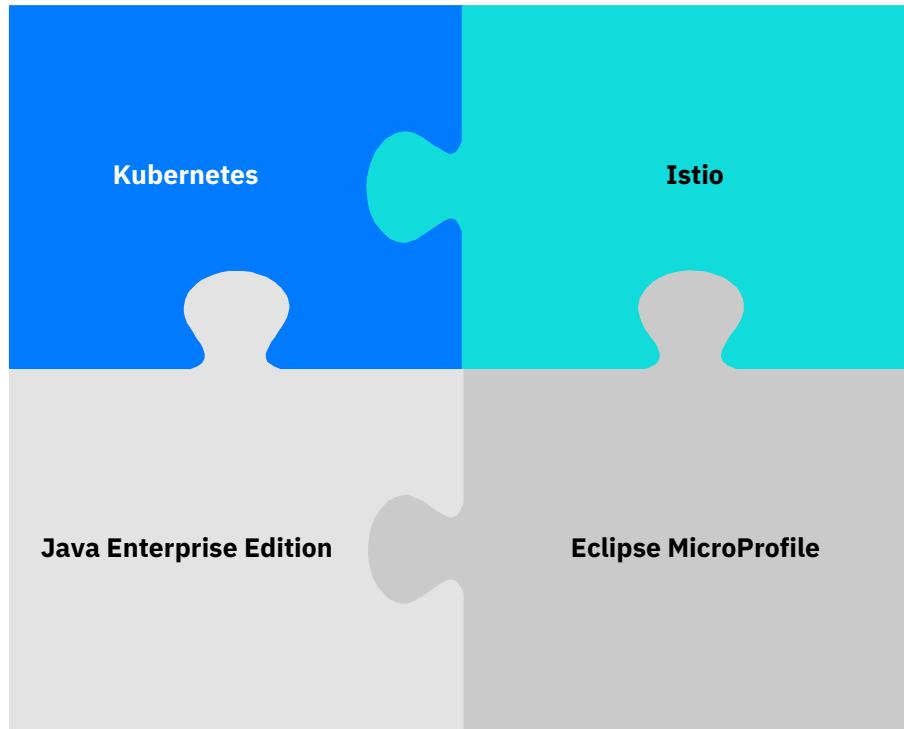
    @Column(name = "articleUrl")
    private String url;

    @Column(name = "articleAuthor")
    private String author;

    @Column(name = "creationDate")
    private String creationDate;

    public ArticleEntity() {
    }
```

# How to use all Pieces together?



Leverage platforms as  
much as possible.

Use frameworks for app  
specific logic.

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

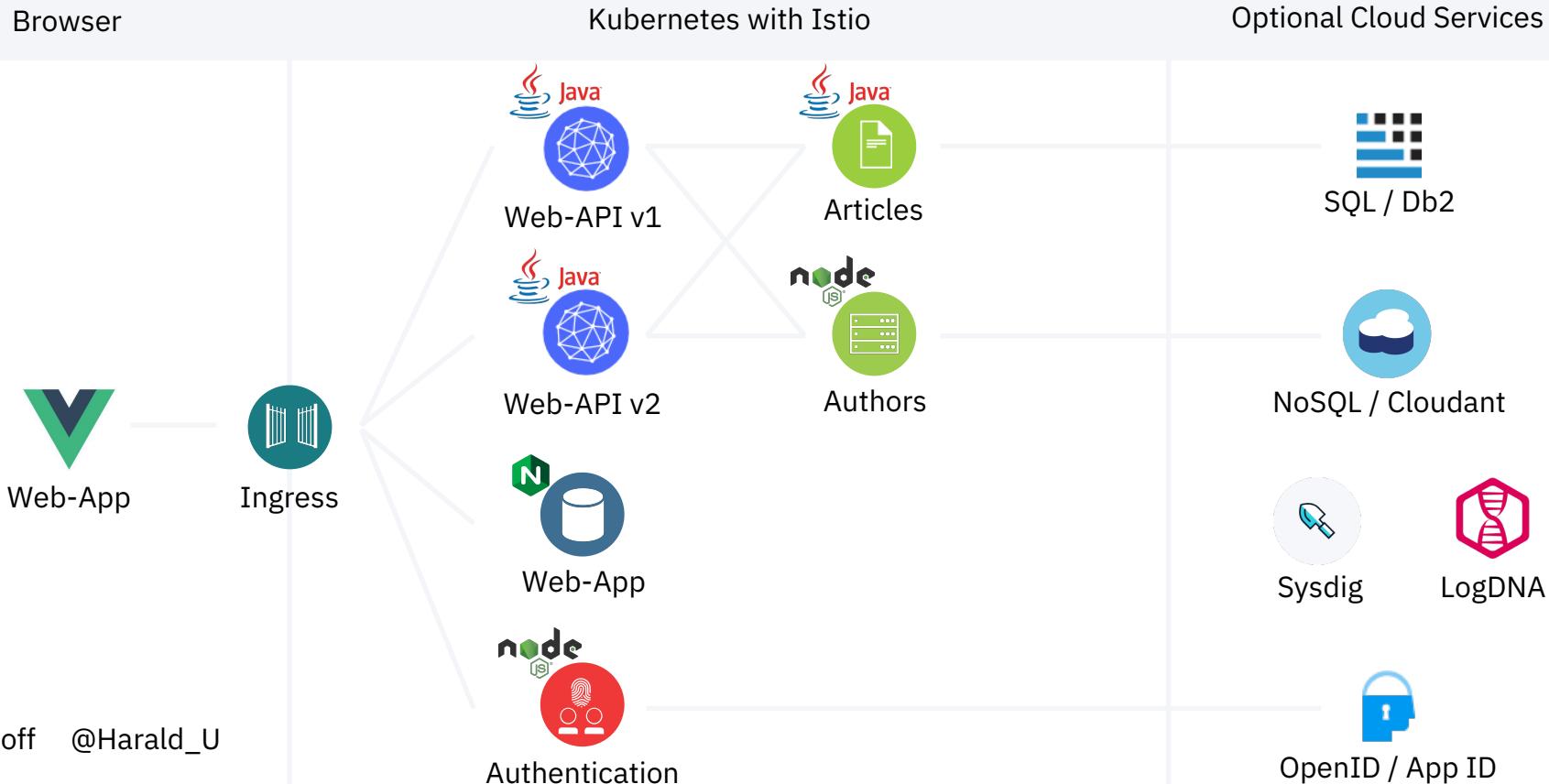
Design principles for the  
end-to-end example  
'cloud-native-starter'

Use only open-source  
components for the core  
services of the application

Make the first time  
experience as simple as  
possible

Be able to run the  
application in different  
environments

# Architecture: End-to-End Example ‘cloud-native-starter’



# 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

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)	<a href="#">Enable</a>
IAM pullsecrets	Enabled
Public service endpoint URL	<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 →



Leverage platforms as much as possible

Use frameworks for app specific logic

IBM loves open source

Kubernetes and Istio  
OpenJ9 & AdoptOpenJDK  
MicroProfile  
Open Liberty

IBM Developer

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

IBM Cloud Lite account

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

