How to develop your first cloud-native Applications with Java

Niklas Heidloff
Developer Advocate, IBM
@nheidloff

IBM **Developer**

IBM

“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

New Options → New Challenges

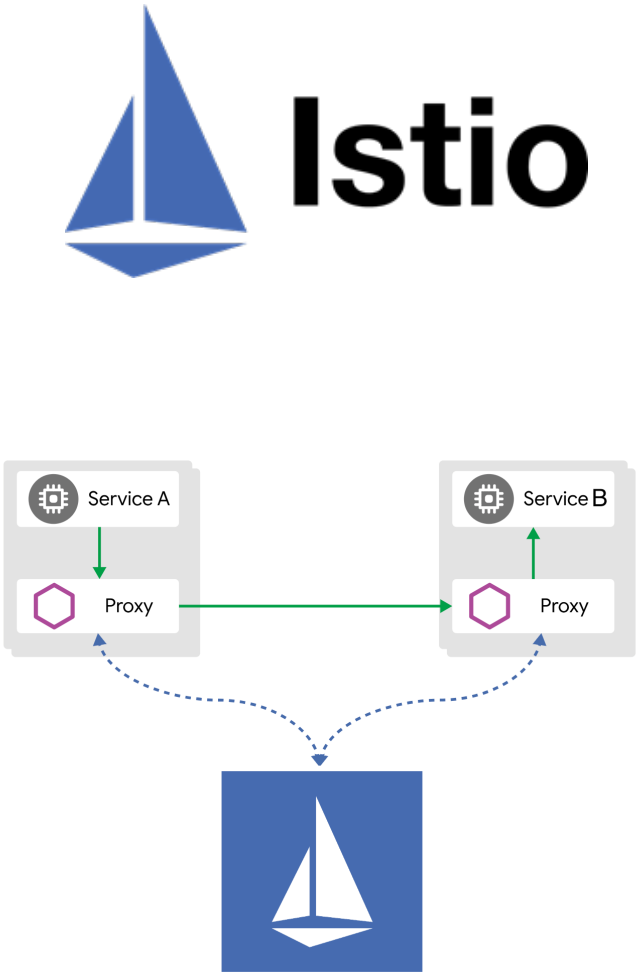
“Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.”

kubernetes.io

kubernetes

“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`

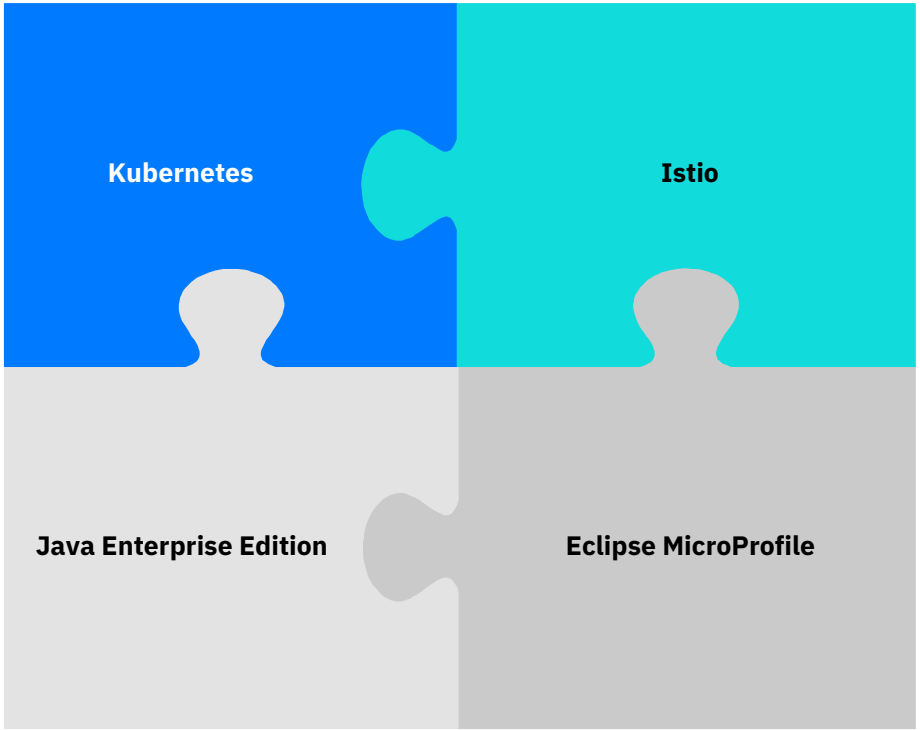


“Optimizing Enterprise Java for a Microservices Architecture.

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

`microprofile.io`

How to use all Pieces together?



Leverage platforms as much as possible.

Use frameworks for app specific logic.

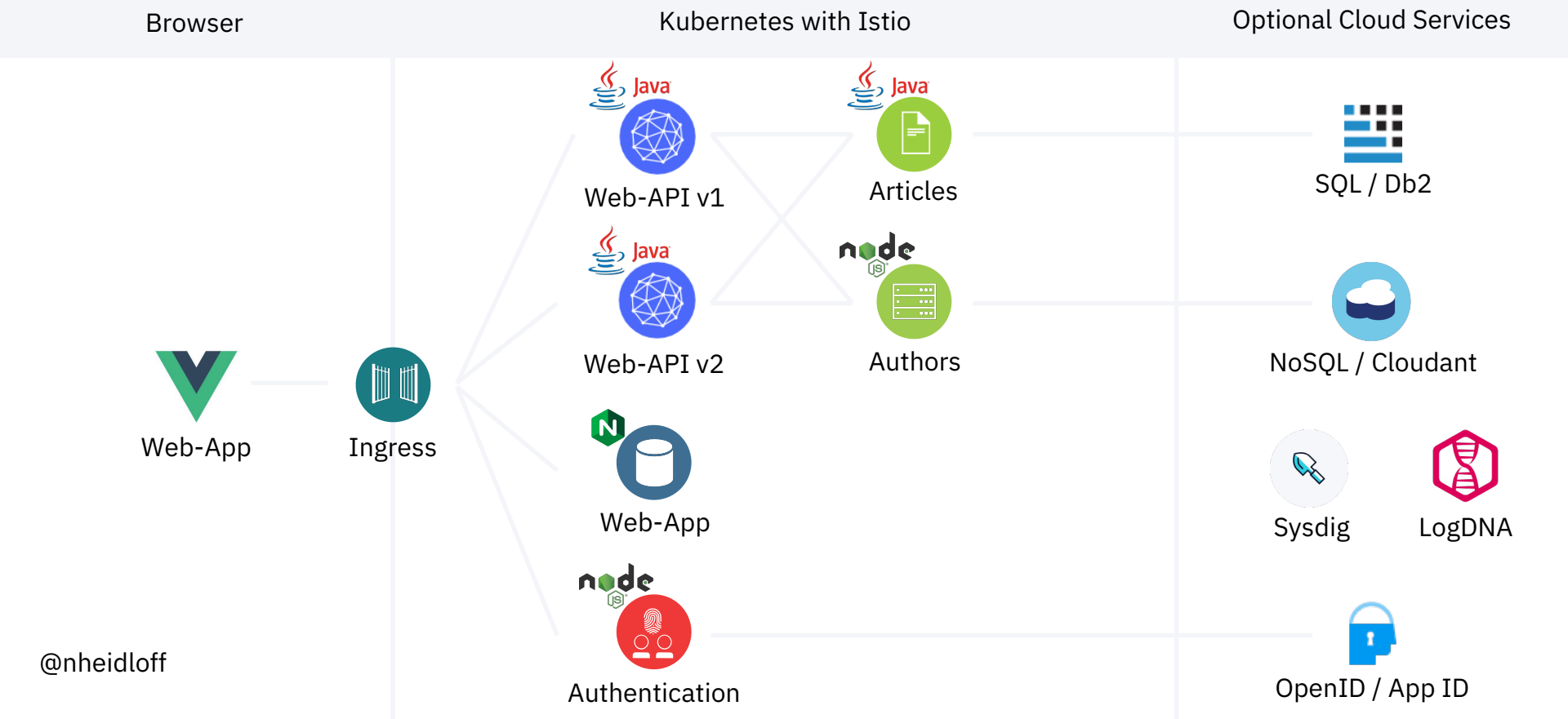
Design principles for the end-to-end example ‘cloud-native-starter’

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

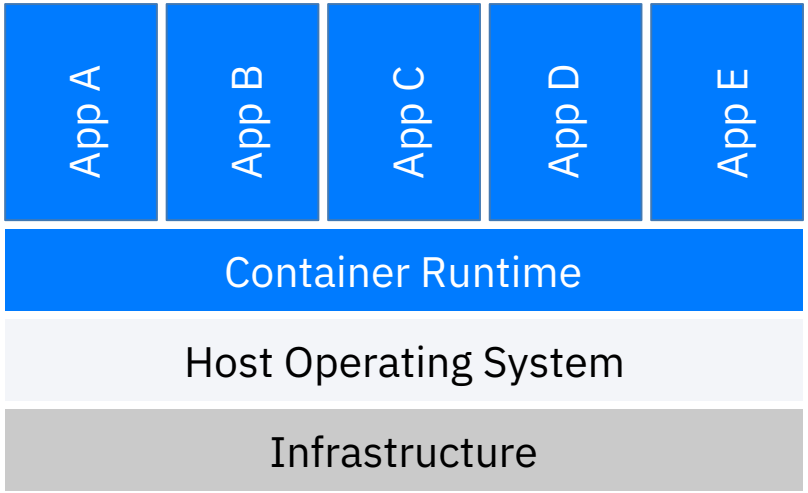
Architecture: End-to-End Example ‘cloud-native-starter’



@nheidloff

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

docker.com



Portable Containers

Java Image

Open source stack

OpenJ9 0.14.0

OpenJDK 11.0.3+7 from AdoptOpenJDK

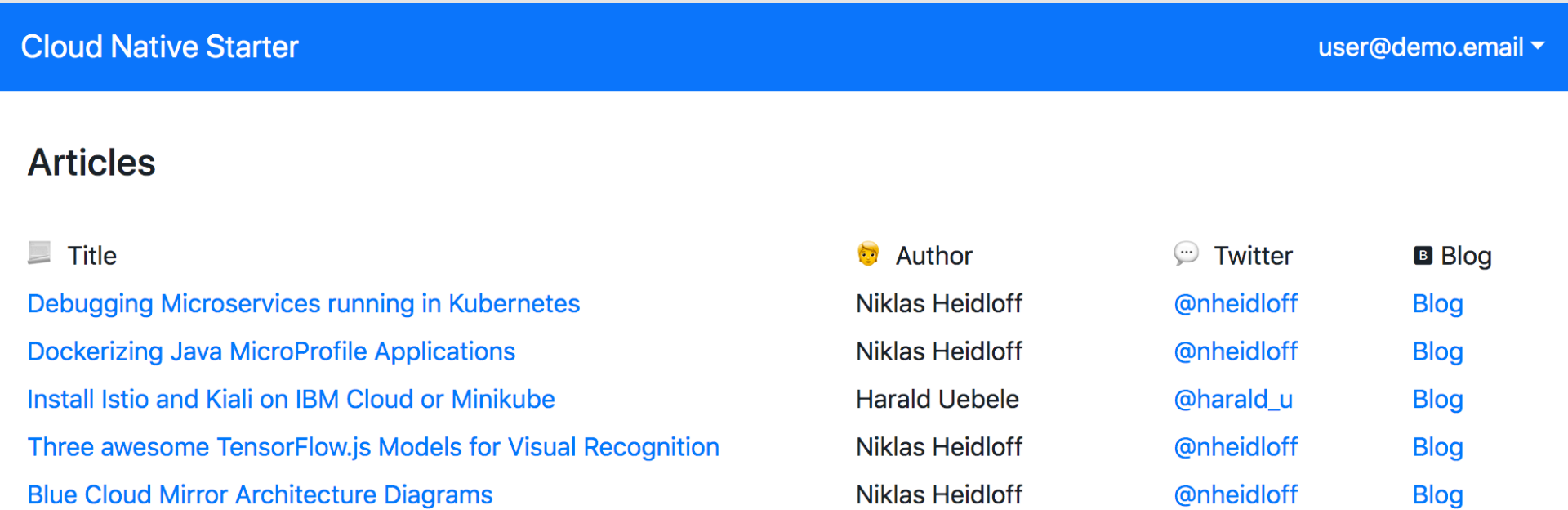
Open Liberty 19.0.0.5

MicroProfile 2.2

Dockerfile

```
FROM open-liberty:microProfile2-java11

COPY liberty/server.xml /config/

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

Example Application



Example Application – REST APIs

Browser

Kubernetes with Istio

Exposing REST APIs

JAX-RS

Java API for RESTful Web Services

@nheidloff

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

@nheidloff

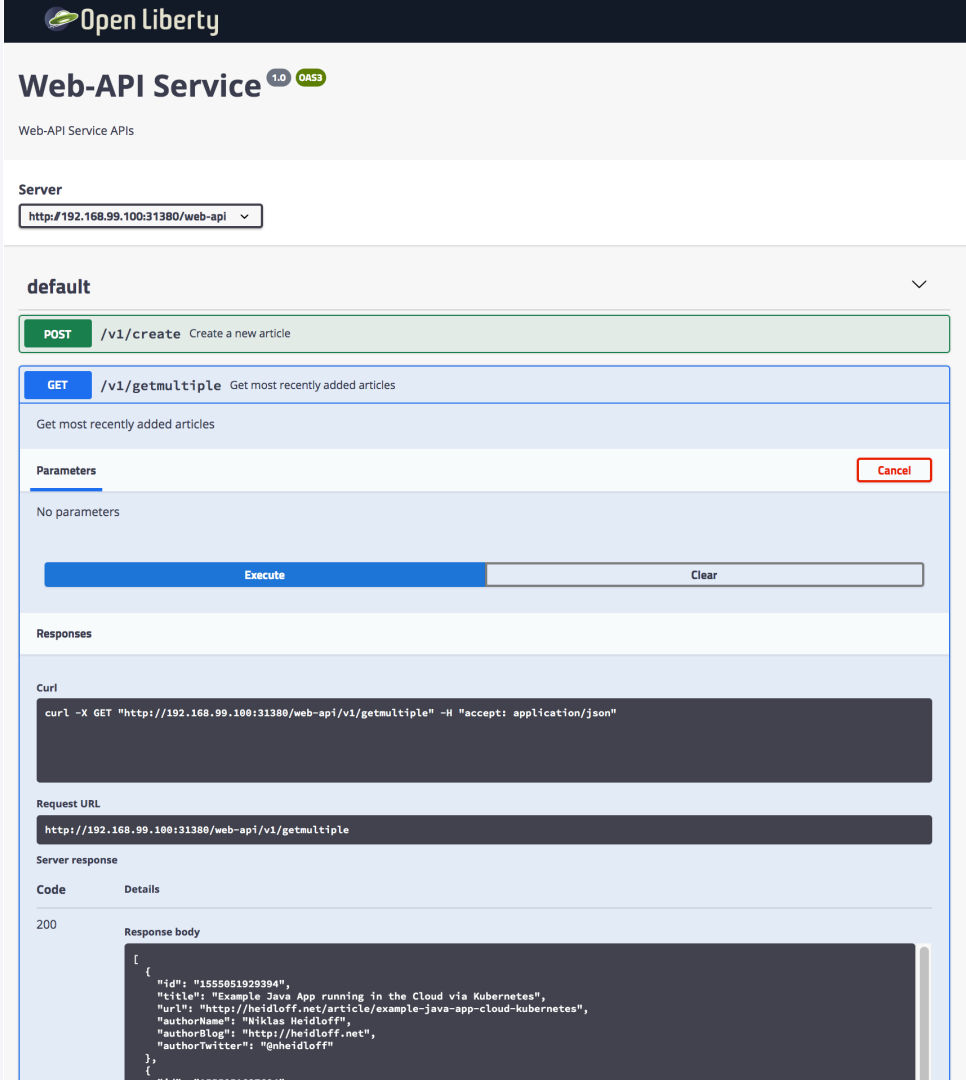
GetArticles.java

```
@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



@nheidloff

Consuming REST APIs

MicroProfile Rest Client

Type-safe approach to invoke RESTful services

@nheidloff

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

@nheidloff

AuthorsService.java

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

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

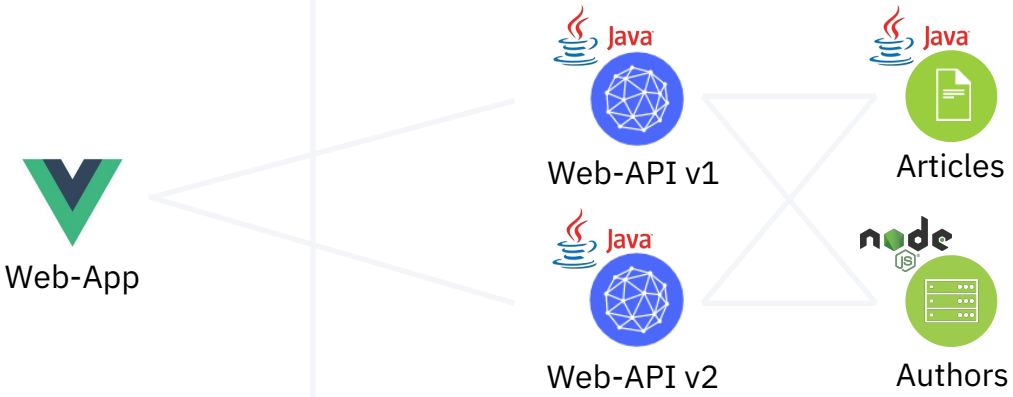
AuthorsServiceDataAccess.java

```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);
```

Example Application – Traffic Management

Browser

Kubernetes with Istio



Traffic Management – Web API Version 1

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

Traffic Management – Web API Version 2

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
Three Minutes Demo of Blue Cloud Mirror	Niklas Heidloff	@nheidloff	Blog
IBM announced Managed Istio and Managed Knative	Niklas Heidloff	@nheidloff	Blog
Developing and debugging Microservices with Java	Niklas Heidloff	@nheidloff	Blog
Recent Java Updates from IBM	Niklas Heidloff	@nheidloff	Blog
Blue Cloud Mirror — (Don't) Open The Doors!	Harald Uebele	@harald_u	Blog

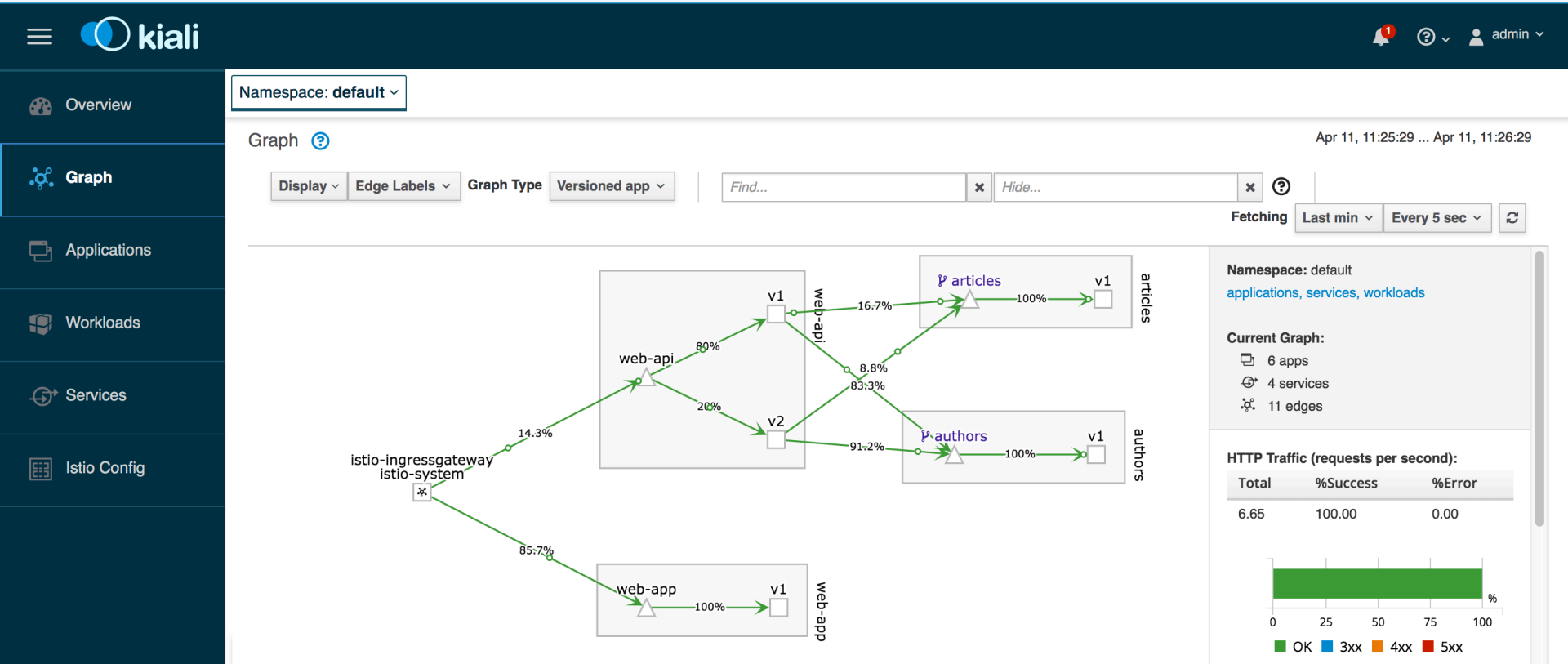
Traffic Management

Example: 80% / 20% splitting

ingress.yaml

```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
```

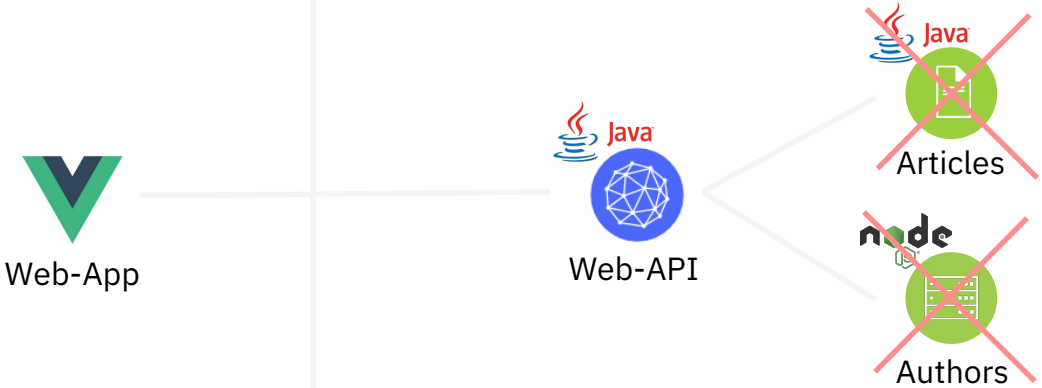
Traffic Management Demo



Example Application – Resiliency

Browser

Kubernetes with Istio



Resiliency

Authors service not available

Usage of default values

@nheidloff

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

@nheidloff

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	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

Resiliency Demo

Cloud Native Starter user@demo.email

Articles

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

Authentication and Authorization

OpenID Connect

Identity layer on top of the OAuth 2.0 protocol

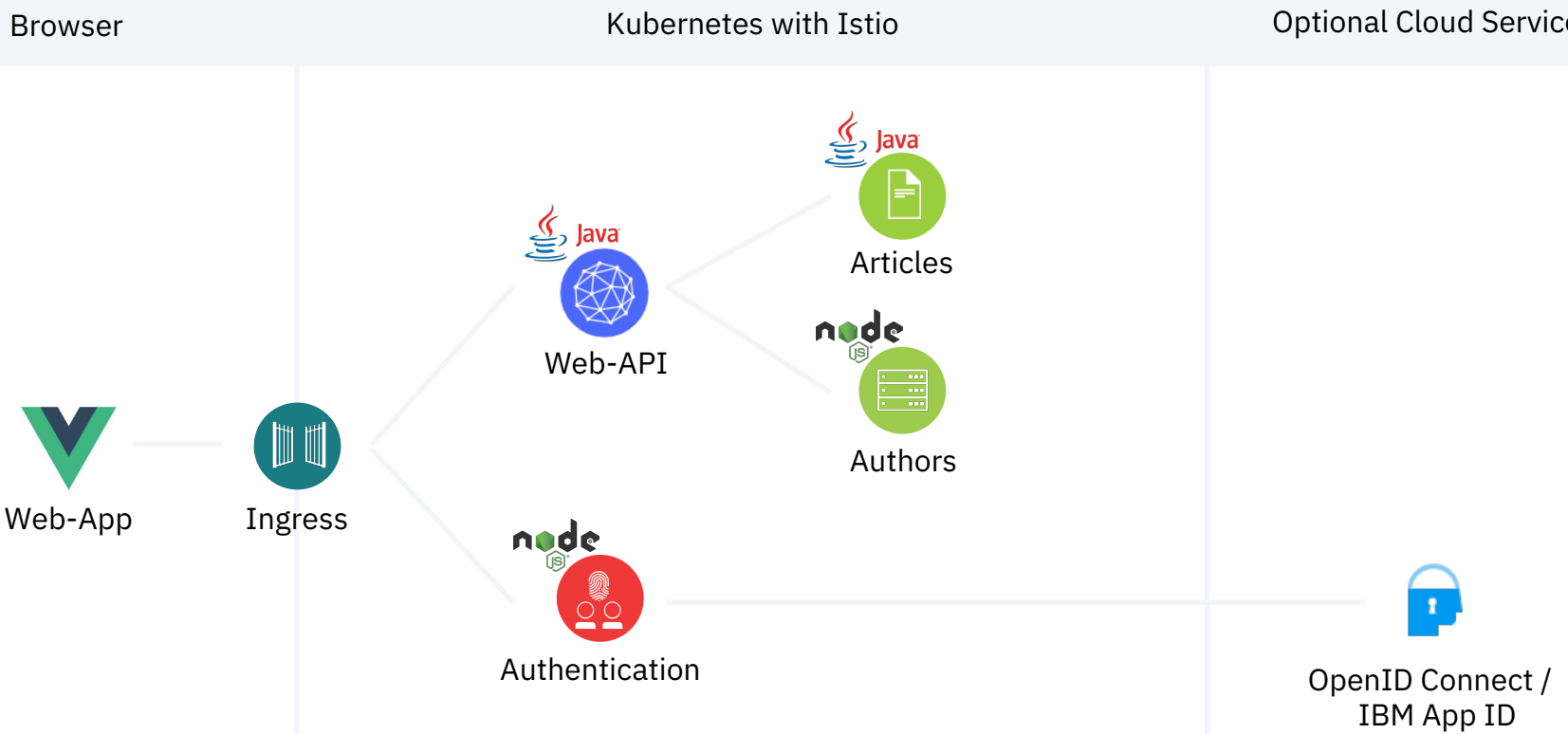
IBM App ID

IBM service to authenticate users and protect APIs

policy.yaml

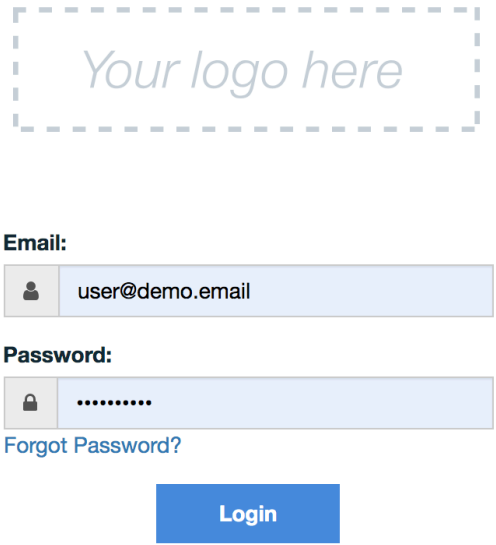
```yaml
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "protect-web-api"
spec:
  targets:
  - name: web-api
  origins:
  - jwt:
      issuer: "https://us-south.appid.cloud.ibm.com/oauth/v4/xxx"
      jwksUri: "https://us-south.appid.cloud.ibm.com/oauth/v4/xxx/publickeys"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

Example Application – Authentication and Authorization

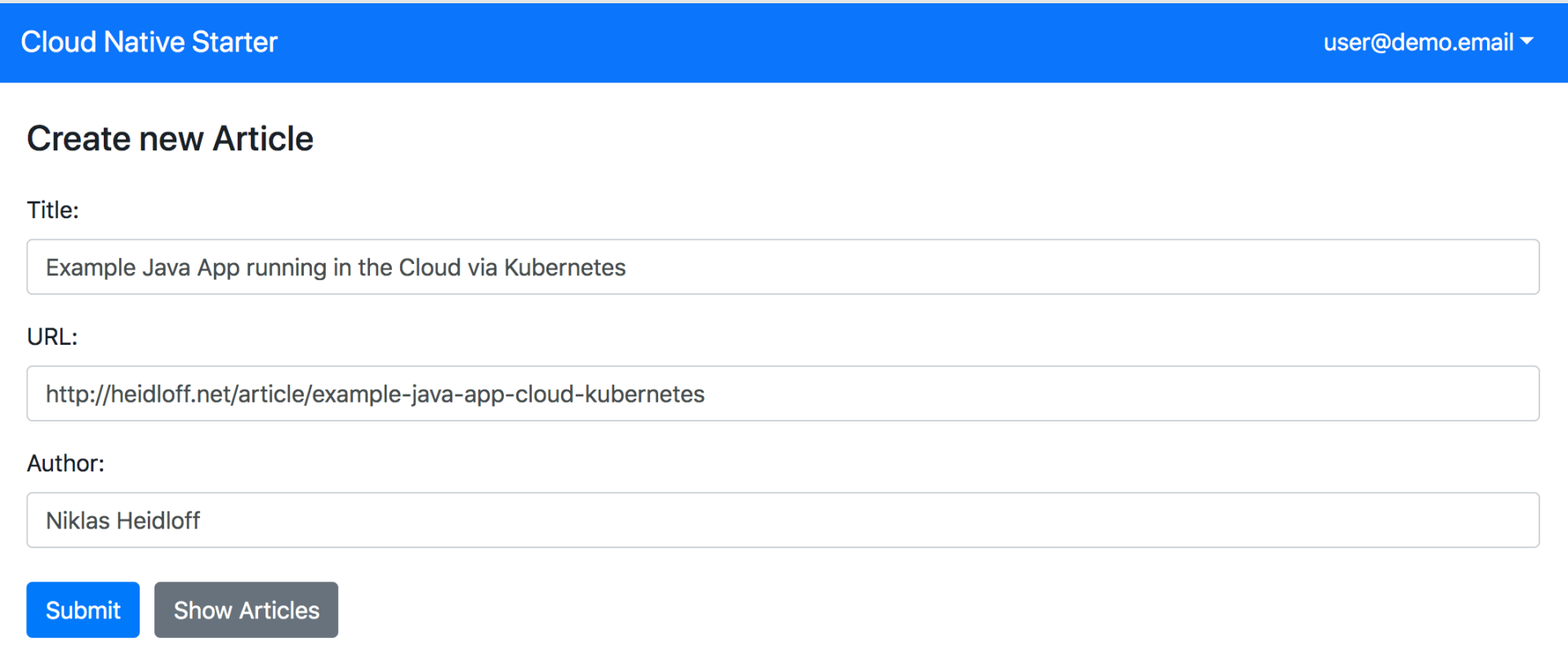


@nheidloff

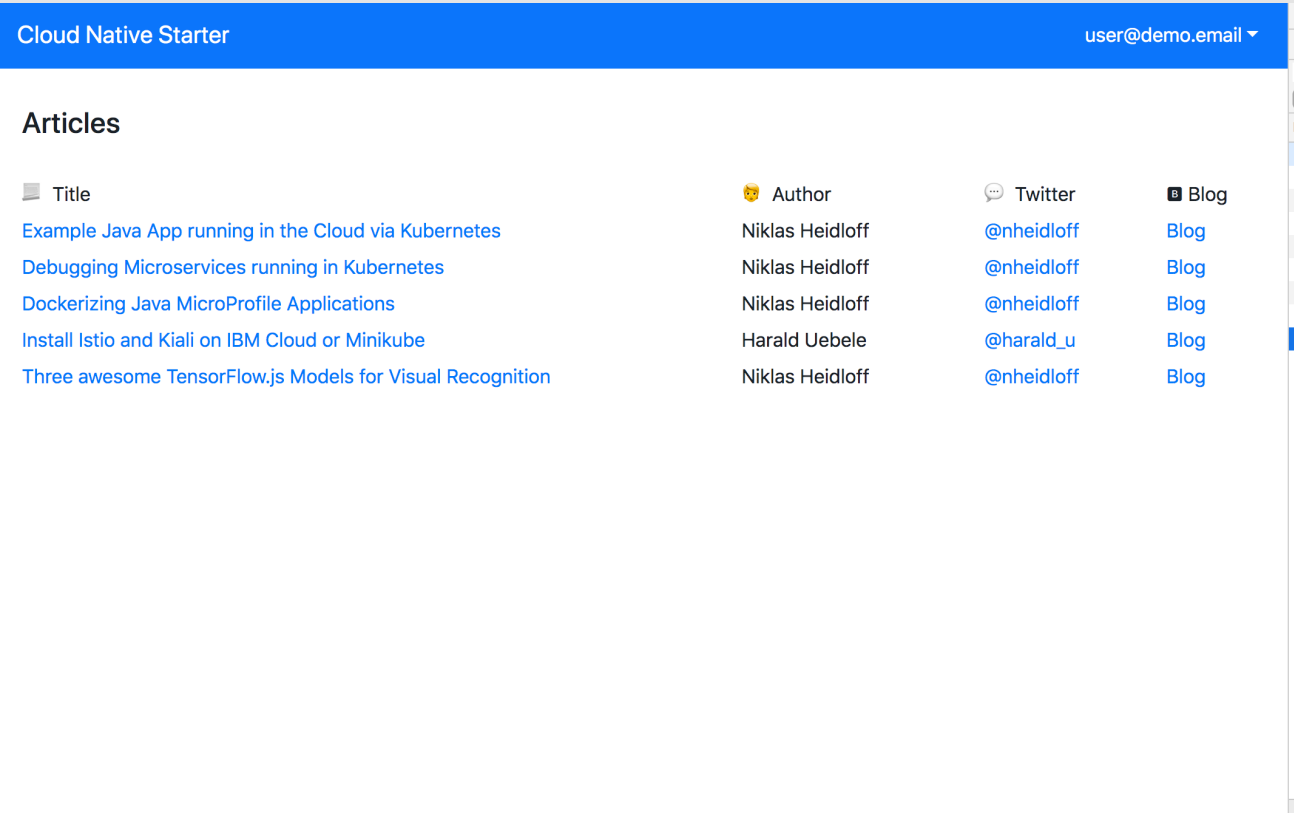
Authentication

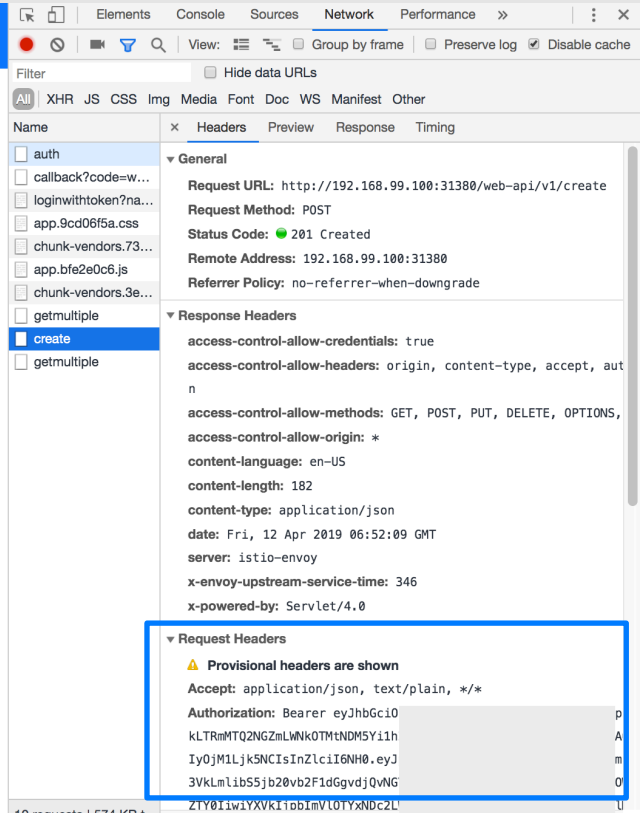


Authorization



Authorization





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

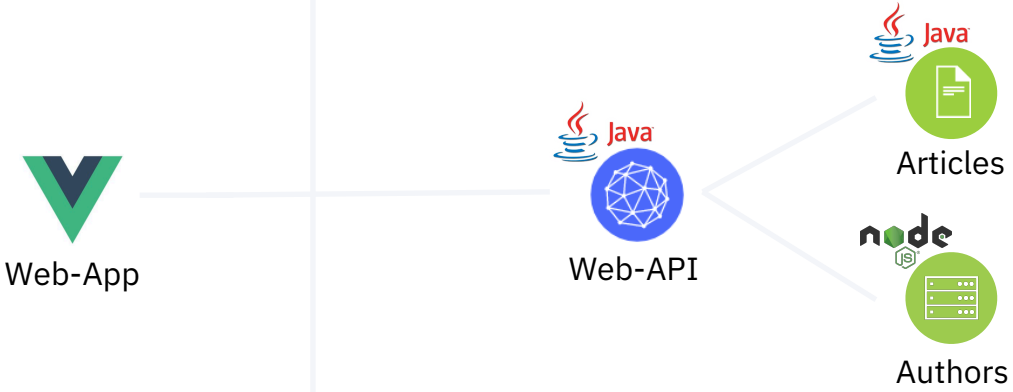
Kubernetes

→ 1 service = N pods

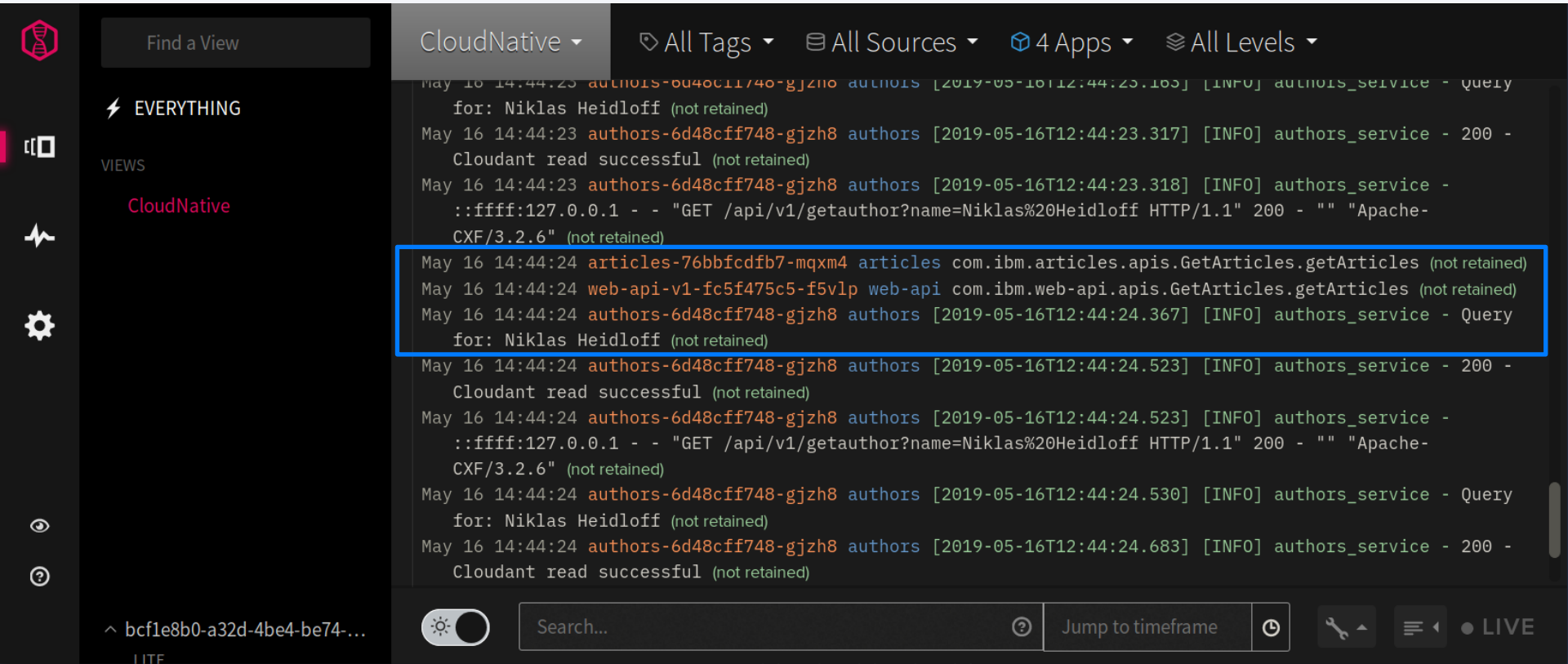
Example Application – Distributed Logging

Browser

Kubernetes with Istio



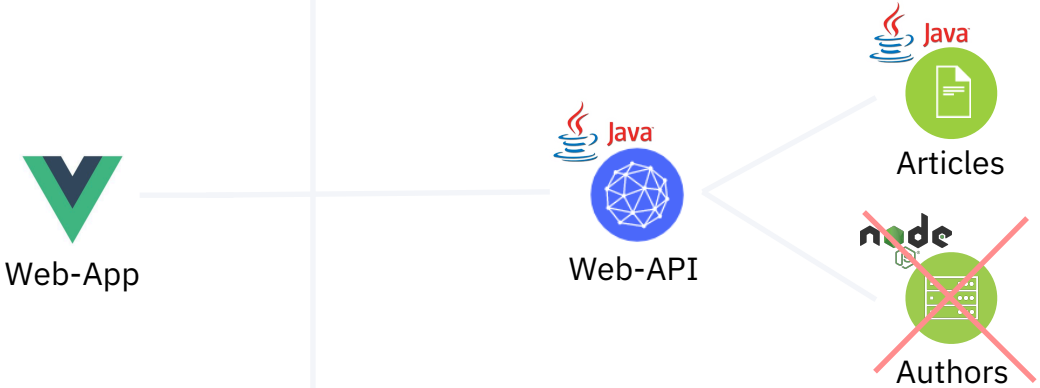
Distributed Logging



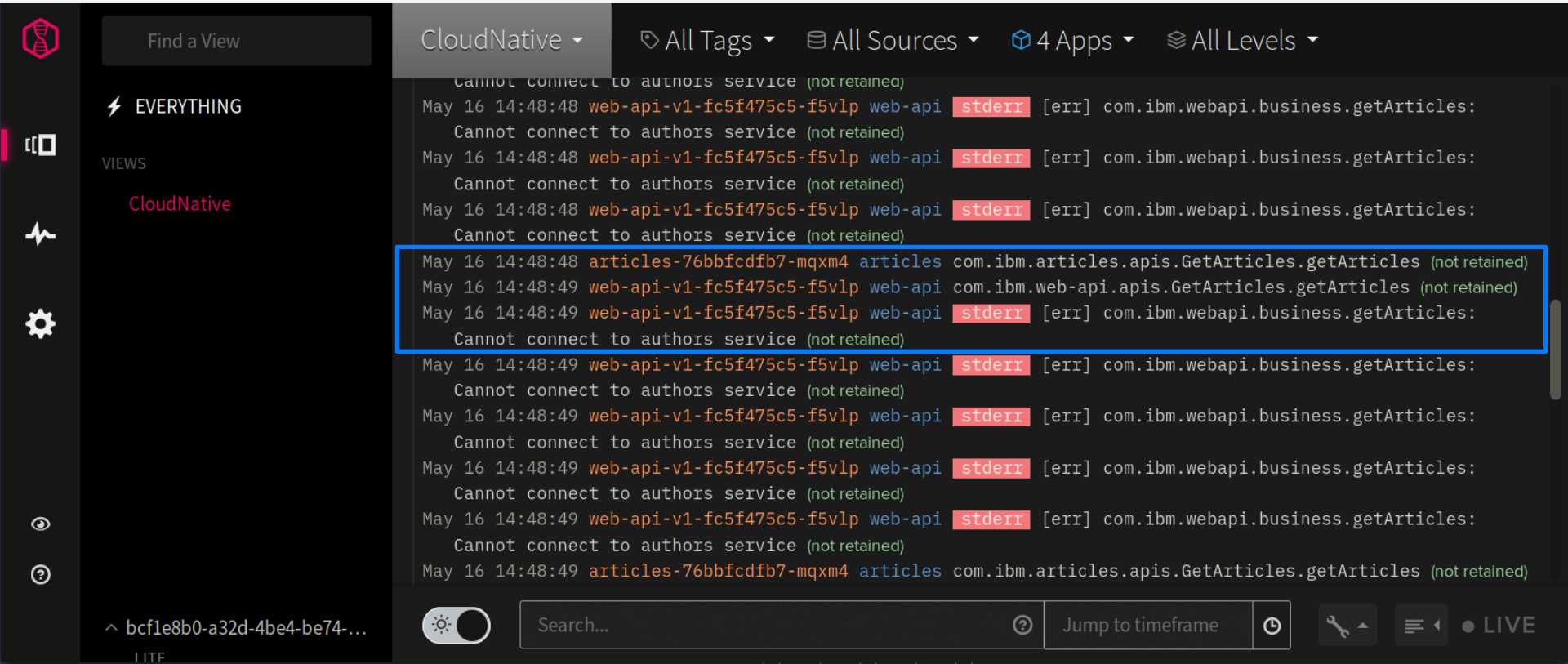
Example Application – Distributed Logging

Browser

Kubernetes with Istio



Distributed Logging



Tracing

OpenTracing

Vendor-neutral APIs and instrumentation for distributed tracing

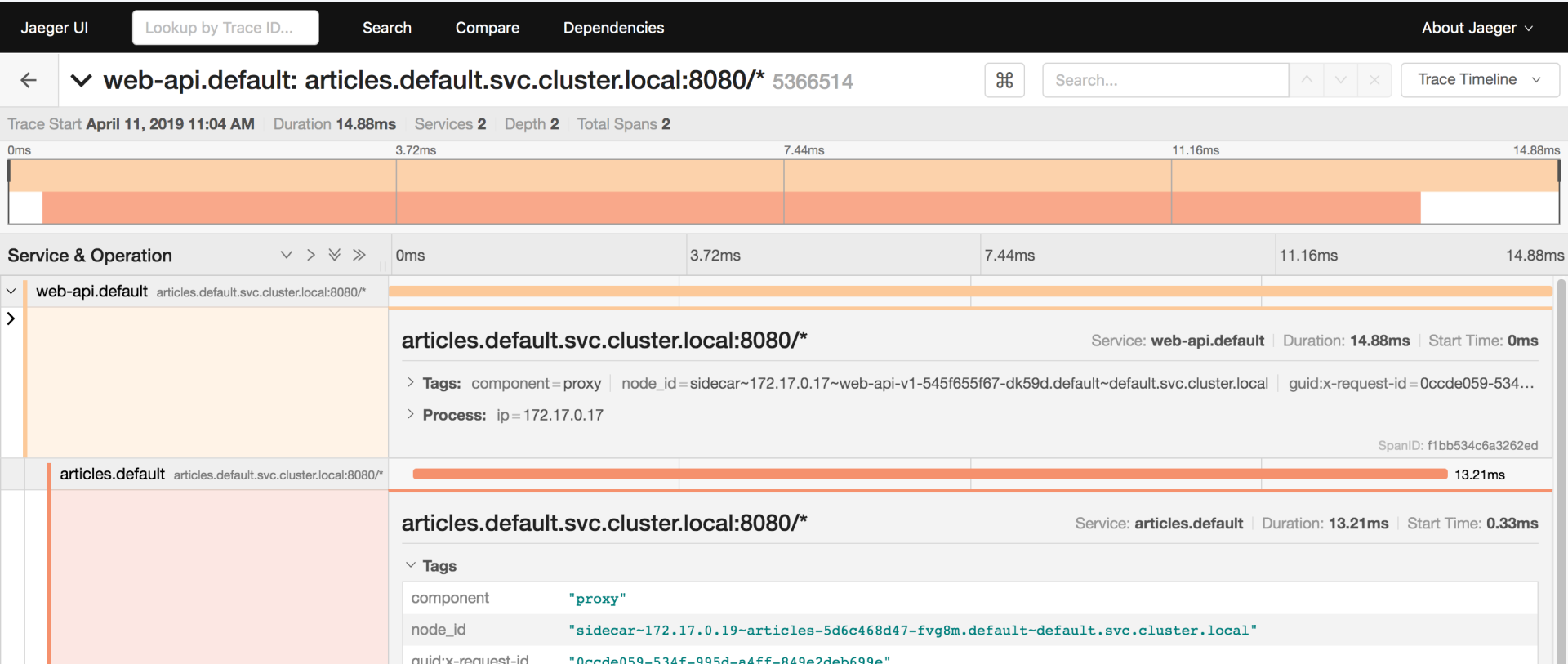
Jaeger and Zipkin

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

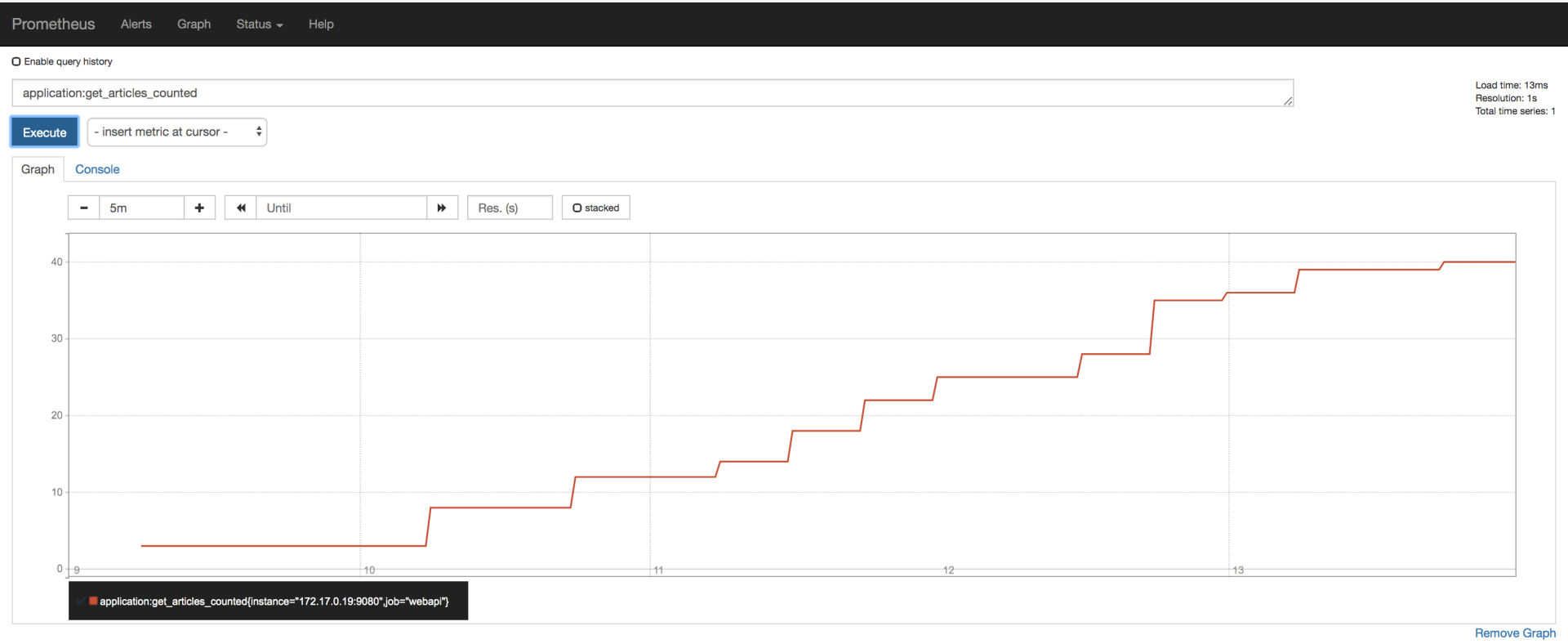
Prometheus

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


Prometheus
Alerts
Graph
Status
Help
Enable query history
application:get_articles_counted
Load time: 13ms
Resolution: 1s
Total time series: 1
Execute
- insert metric at cursor -
Graph
Console
5m
Until
Res. (s)
stacked
40
30
20
10
0
9
10
11
12
13
application:get_articles_counted{instance="172.17.0.19:9080",job="webapi"}
Remove Graph

Healthchecks

MicroProfile Health

Liveness probes and readiness probes

@nheidloff

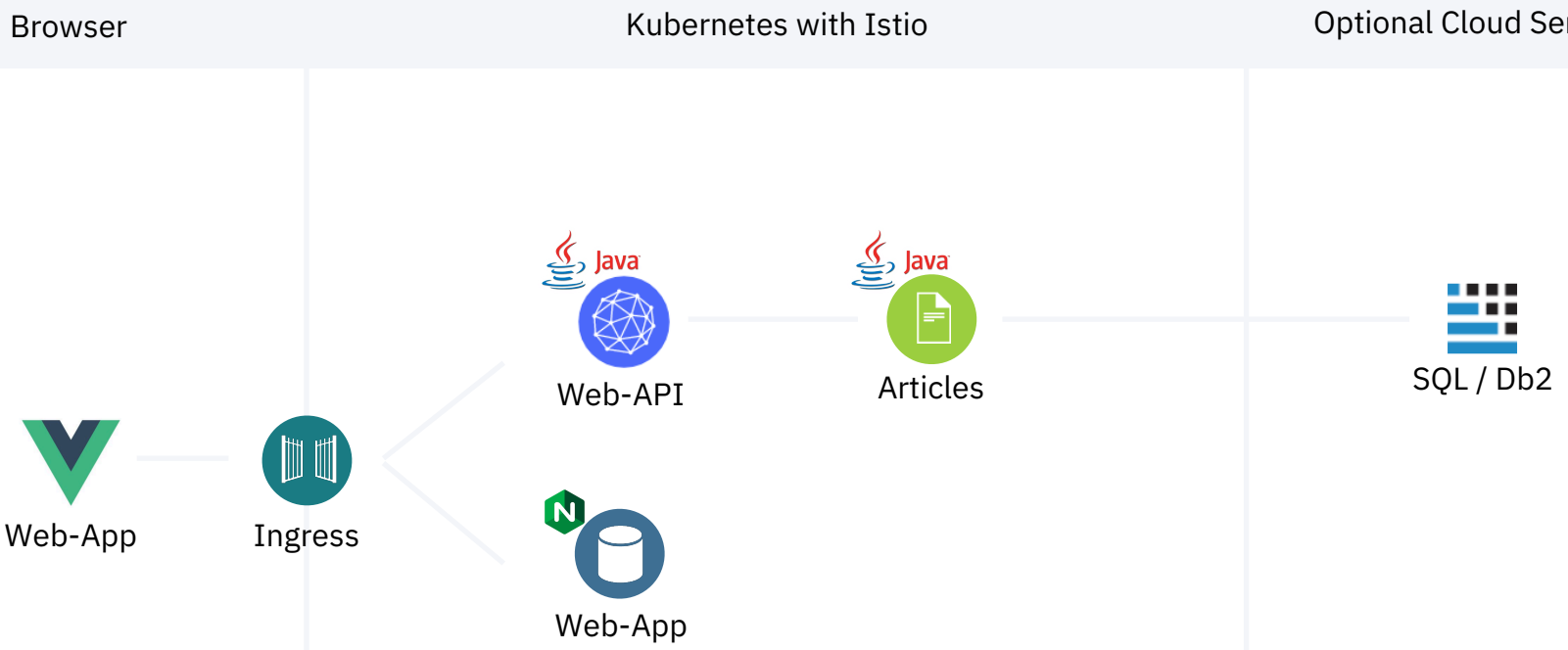
HealthEndpoint.java

```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

```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



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() {
    }
```

Configuration

Via MicroProfile

kubernetes.yaml

```yaml
kind: Deployment
apiVersion: apps/v1beta1
metadata:
  name: articles
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: articles
        version: v1
    spec:
      containers:
      - name: articles
        image: articles:1
        ports:
        - containerPort: 8080
        env:
        - name: samplescreation
          valueFrom:
            configMapKeyRef:
              name: articles-config
              key: samplescreation
        - name: inmemory
          valueFrom:
            configMapKeyRef:
              name: articles-config
              key: inmemory
      restartPolicy: Always
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: articles-config
data:
  samplescreation: CREATE
  inmemory: USE_IN_MEMORY_STORE
---
```

Configuration

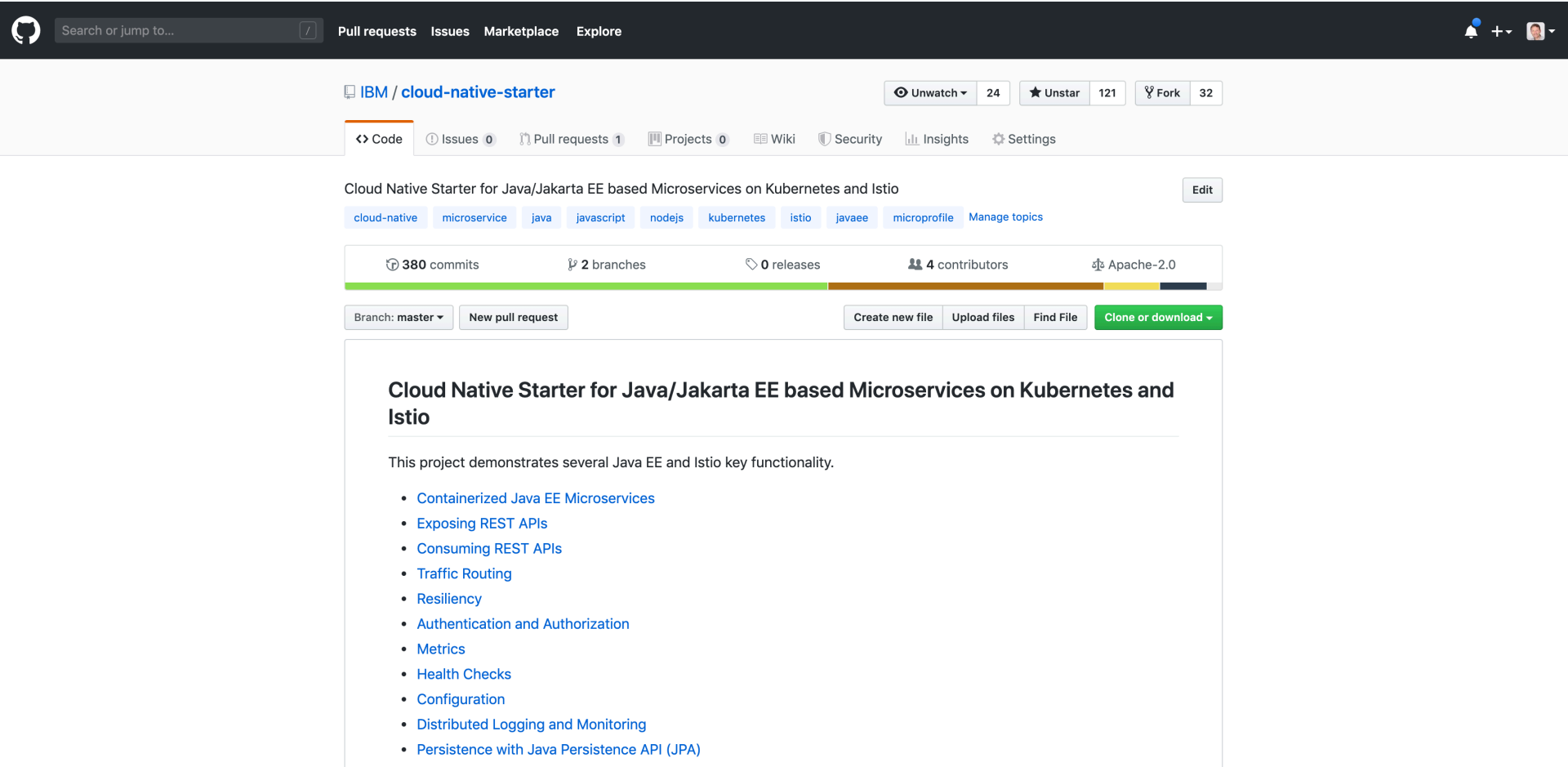
Via MicroProfile

CoreService.java

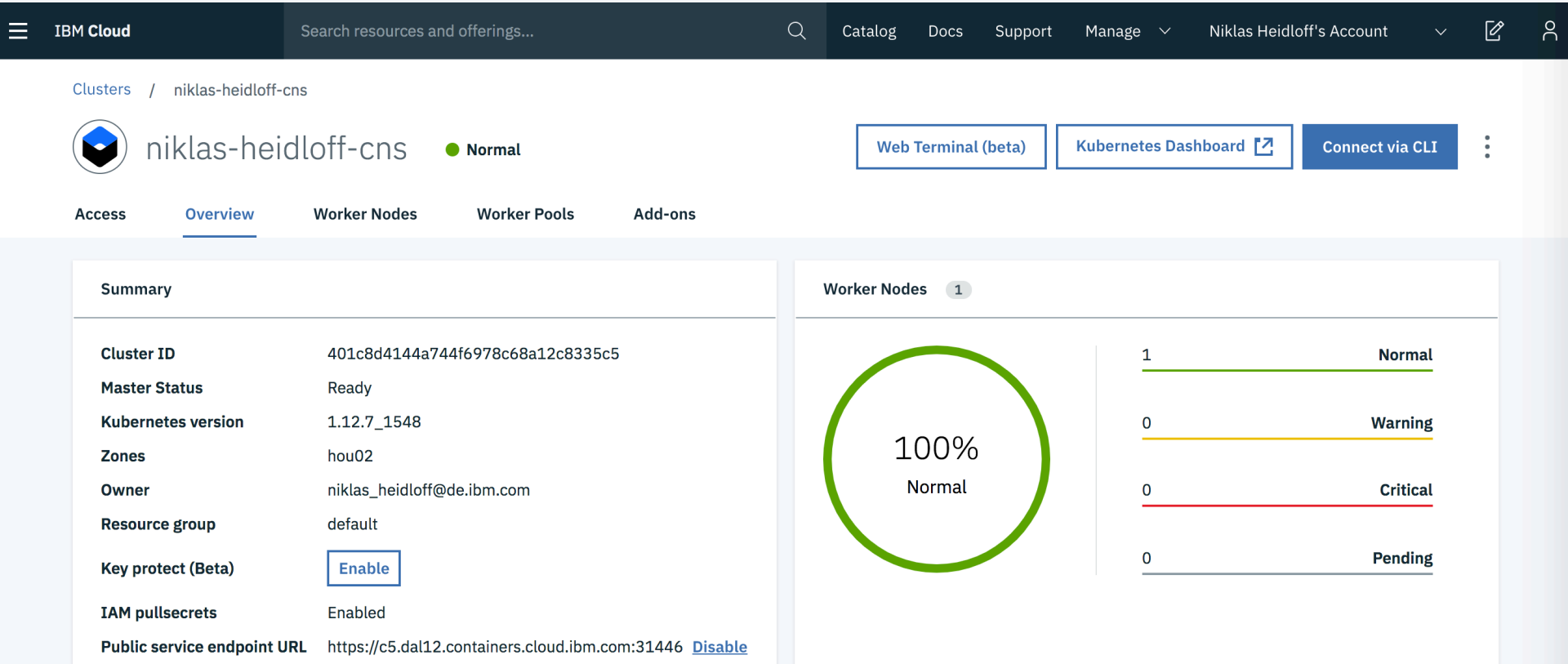
```
@ApplicationScoped
public class CoreService {

  private static final String CREATE_SAMPLES = "CREATE";
  private static final String USE_IN_MEMORY_STORE = "USE_IN_MEMORY_STORE";

  @Inject
  @ConfigProperty(name = "inmemory", defaultValue = USE_IN_MEMORY_STORE)
  private String inmemory;

  @Inject
  @ConfigProperty(name = "samplescreation", defaultValue = "dontcreate")
  private String samplescreation;

  @Inject
  private DataAccessManager dataAccessManager;

  @PostConstruct
  private void addArticles() {
    if (inmemory.equalsIgnoreCase(USE_IN_MEMORY_STORE)) {
      if (samplescreation.equalsIgnoreCase(CREATE_SAMPLES))
        addSampleArticles();
    }
  }
```

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

Documentation



IBM Cloud Kubernetes Service including Istio and Knative



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

IBM Cloud Lite account

ibm.biz/nheidloff


IBM