Get started with Security for your Java Microservices Application

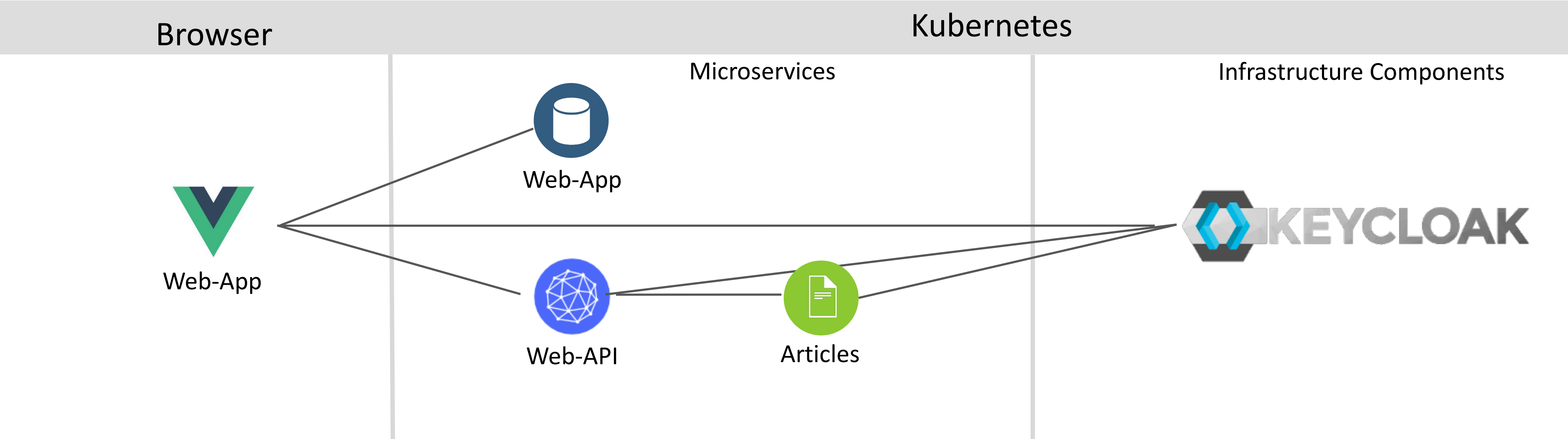
Harald Uebele
Developer Advocate, IBM
@harald_u

Thomas Südbröcker
Developer Advocate, IBM
@tsuedbroecker

IBM **Developer**

As a developer you should ask yourself: "How can I make my application (more) secure?"

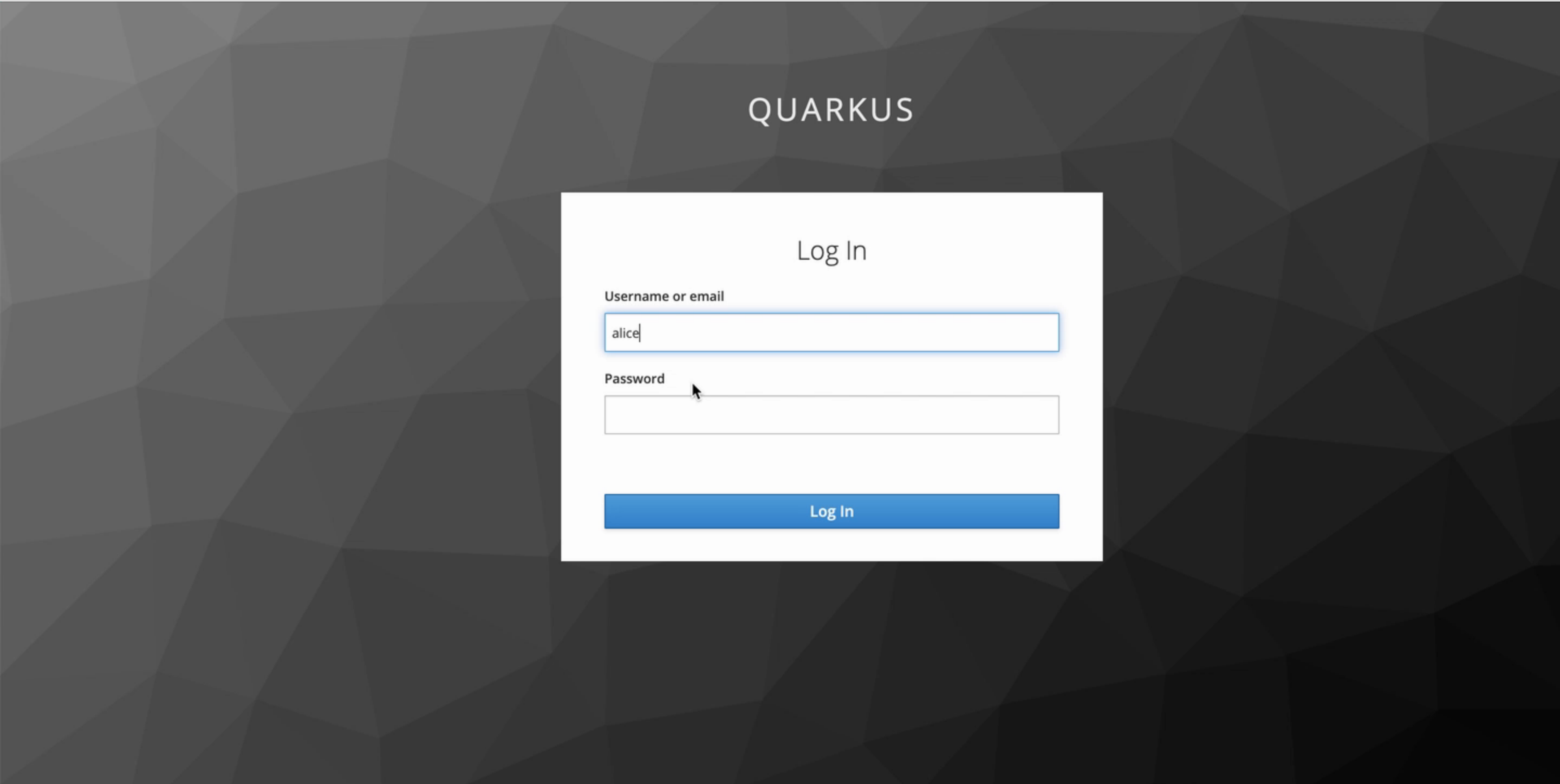
Application Security Example



Authentication and Authorization implementation

Platform security

The Example *Cloud Native Starter* Web Application



Username or email

alice

Password

"A Kubernetes Native Java stack tailored for OpenJDK, HotSpot, and GraalVM, crafted from the best of breed Java libraries and standards."

quarkus.io

QUARKUS

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

“Optimizing Enterprise Java for a Microservices Architecture.“

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

microprofile.io

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

“Open Source Identity and Access Management For Modern Applications and Services”

keycloak.org

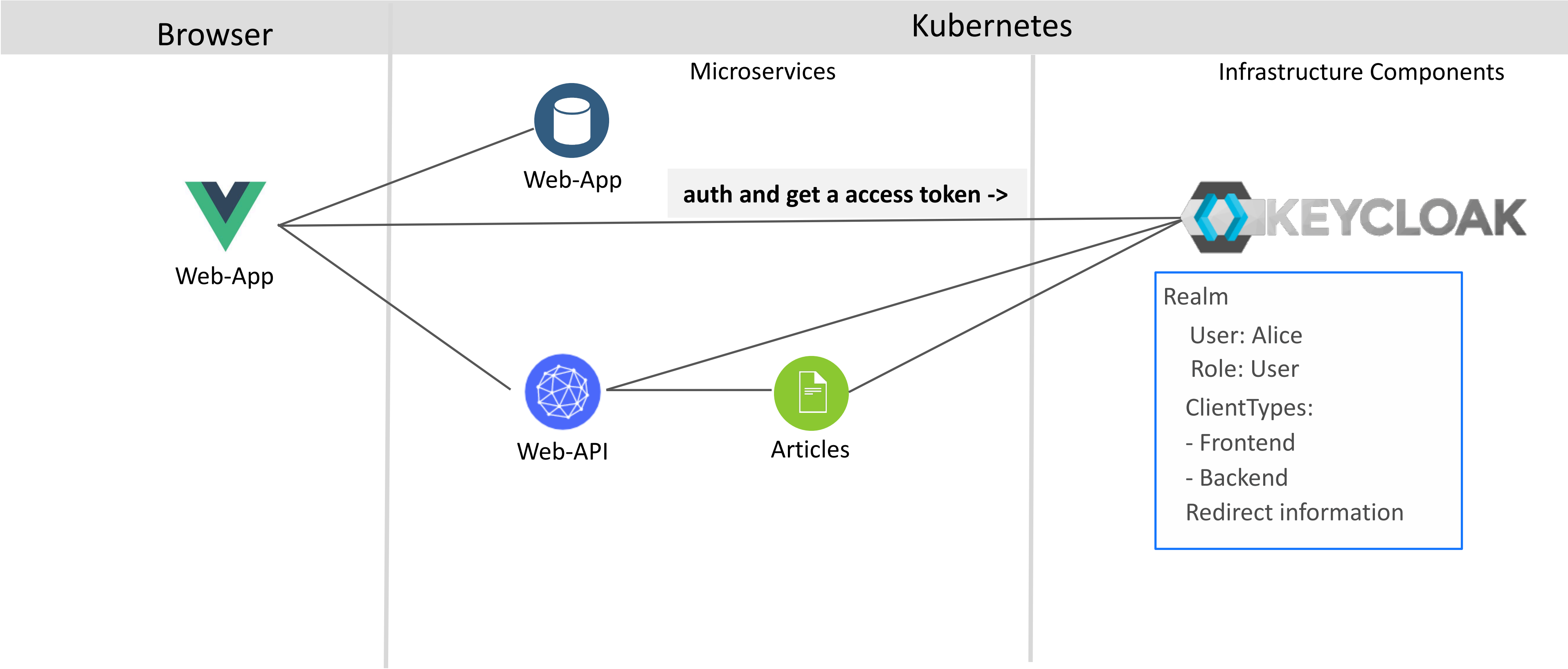
@harald_u @tsuedbroecker

"JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties."

jwt.io

JWT

Let’s make it concrete



Browser

Kubernetes

Microservices

Infrastructure Components

Web-App

Web-App

auth and get a access token ->

KEYCLOAK

Web-API

Articles

Realm

User: Alice

Role: User

ClientTypes:

- Frontend
- Backend

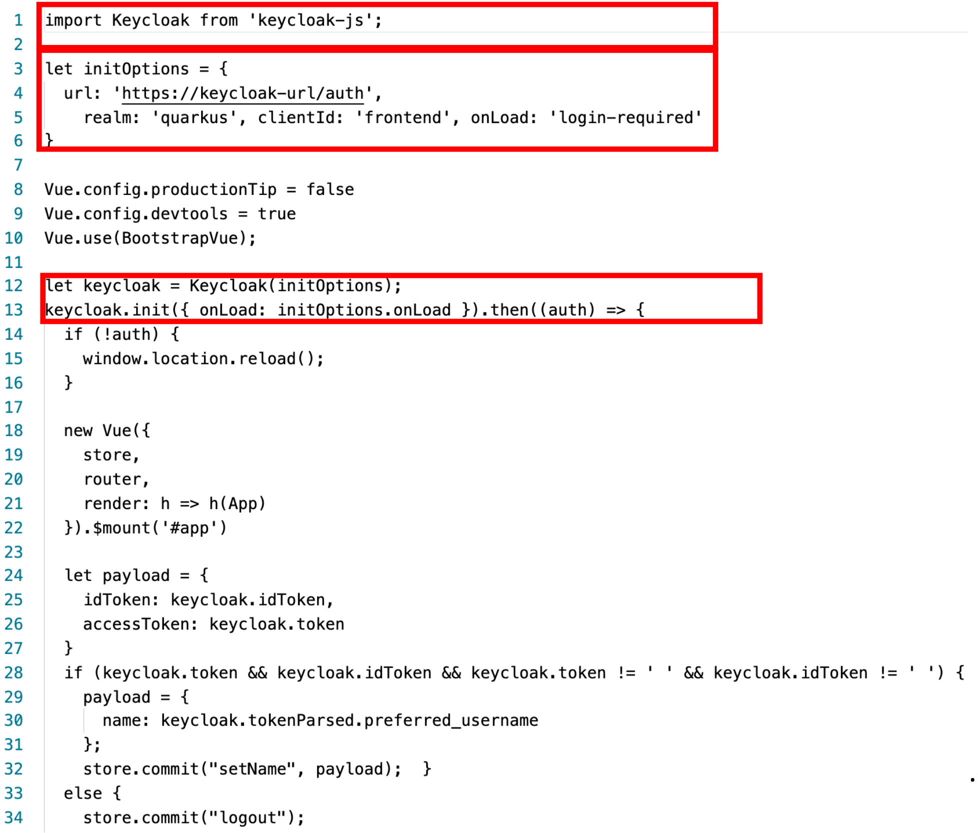
Redirect information

Authentication with Keycloak

Browser

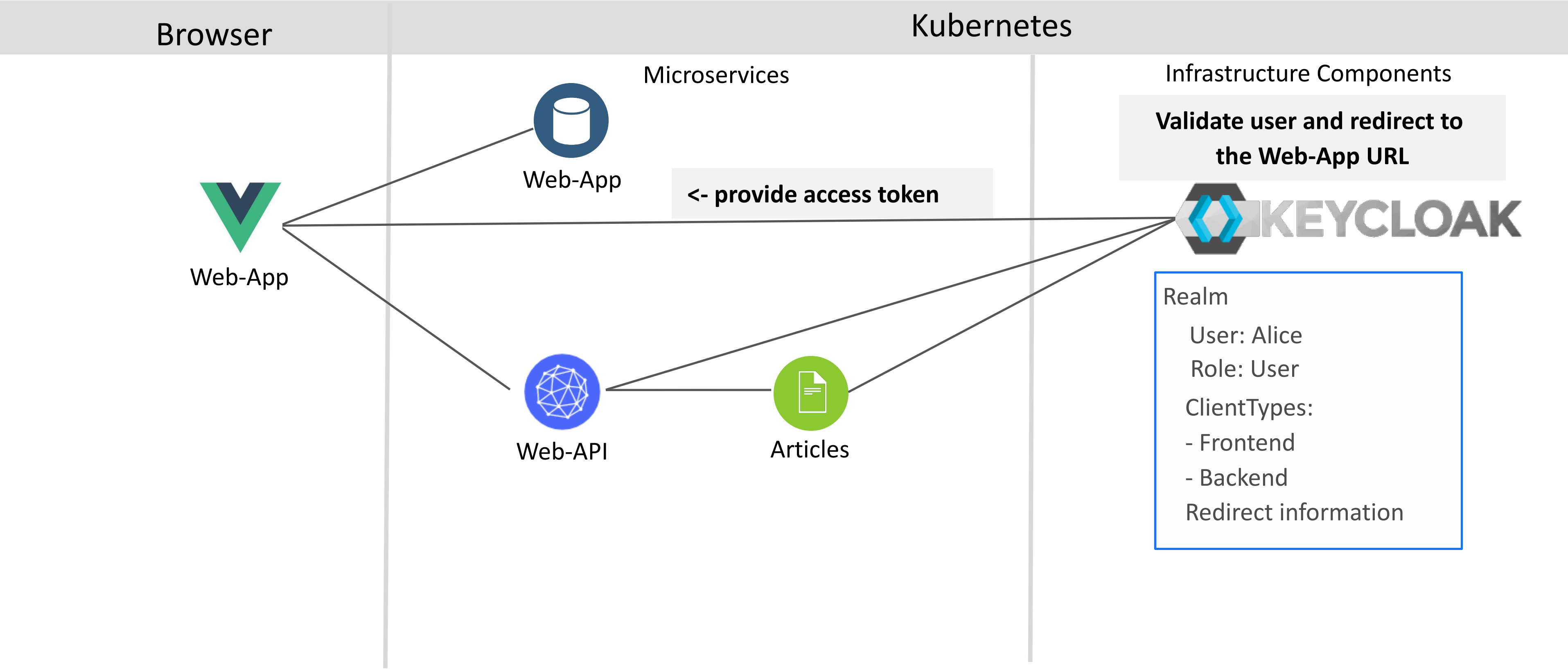
Web-App

Code: “main.js”



```
import Keycloak from 'keycloak-js';

let initOptions = {
  url: 'https://keycloak-url/auth',
    realm: 'quarkus', clientId: 'frontend', onLoad: 'login-required'
}

Vue.config.productionTip = false
Vue.config.devtools = true
Vue.use(BootstrapVue);

let keycloak = Keycloak(initOptions);
keycloak.init({ onLoad: initOptions.onLoad }).then((auth) => {
  if (!auth) {
    window.location.reload();
  }

  new Vue({
    store,
    router,
    render: h => h(App)
  }).$mount('#app')

  let payload = {
    idToken: keycloak.idToken,
    accessToken: keycloak.token
  }
  if (keycloak.token && keycloak.idToken && keycloak.token != ' ' && keycloak.idToken != ' ') {
    payload = {
      name: keycloak.tokenParsed.preferred_username
    };
    store.commit("setName", payload);  }
  else {
    store.commit("logout");
```

Redirect

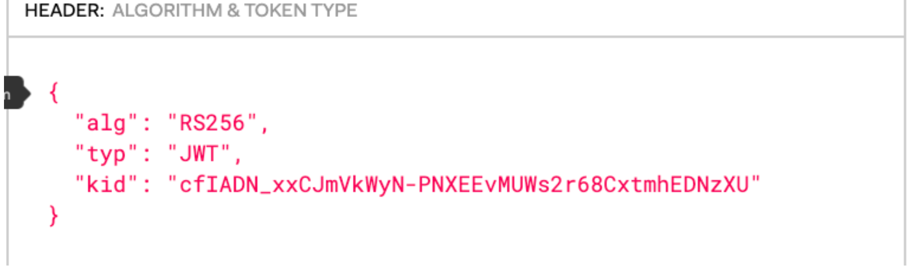


Validate user and redirect to the Web-App URL

<- provide access token

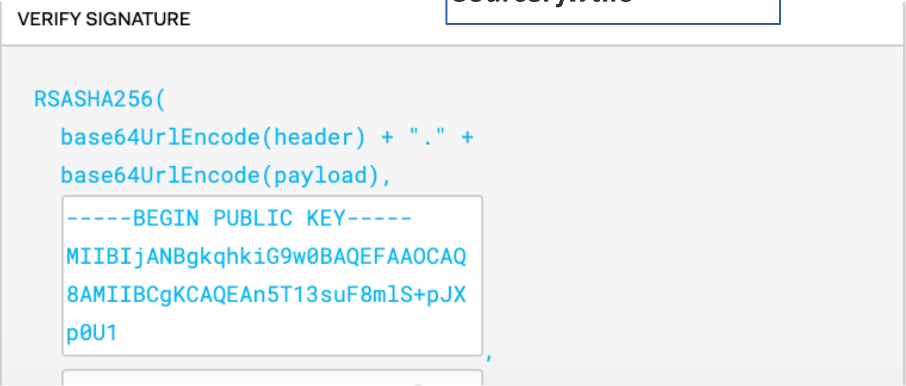
Realm
User: Alice
Role: User
ClientTypes:
- Frontend
- Backend
Redirect information

JSON Web Token Content



HEADER: ALGORITHM & TOKEN TYPE

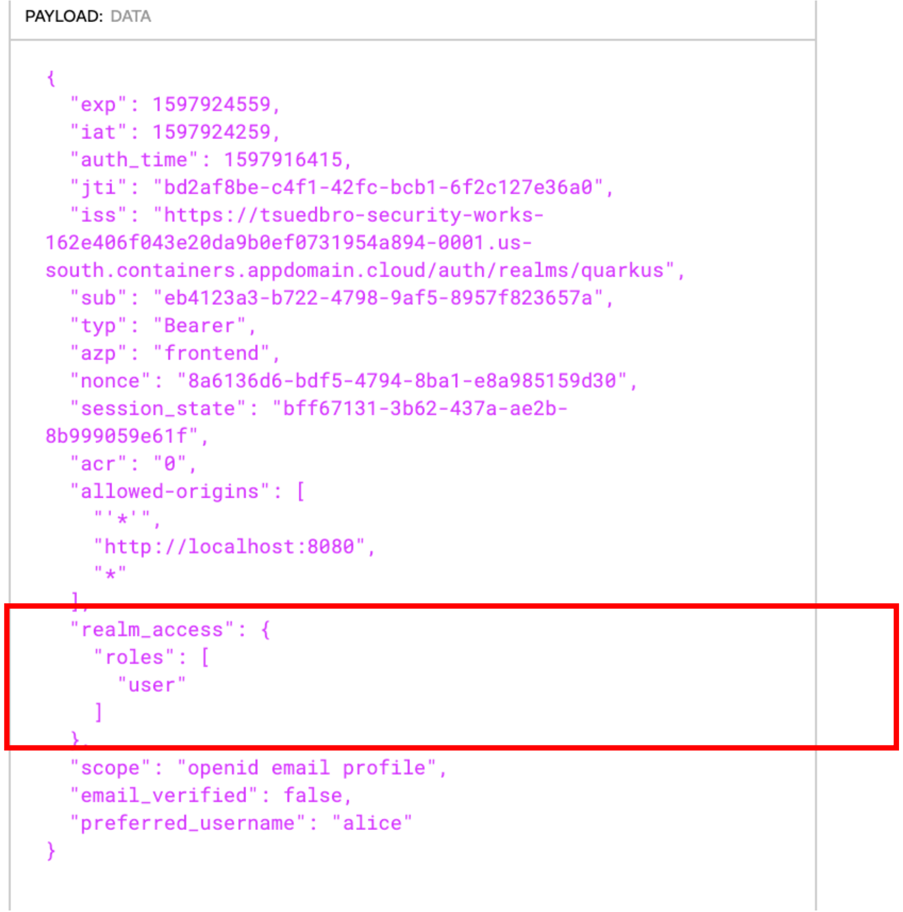
```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "cfIADN_xxCJmVkWyN-PNXEEvMUWs2r68CxtmhEDNzXU"
}
```



VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ
  8AMIIBCgKCAQEAn5T13suF8mlS+pJX
  p0U1
  ,
```

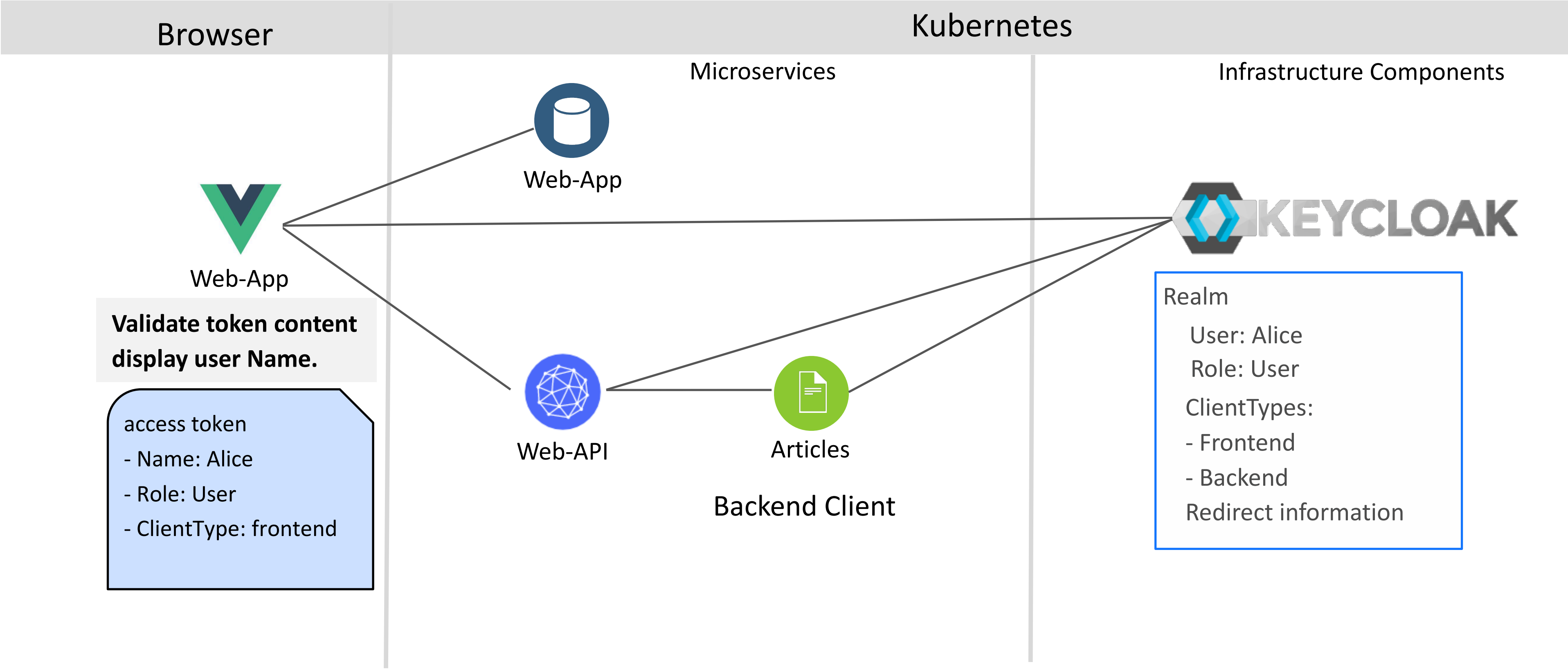
Source: jwt.io



PAYLOAD: DATA

```
{
  "exp": 1597924559,
  "iat": 1597924259,
  "auth_time": 1597916415,
  "jti": "bd2af8be-c4f1-42fc-bcb1-6f2c127e36a0",
  "iss": "https://tsuedbro-security-works-
162e406f043e20da9b0ef0731954a894-0001.us-
south.containers.appdomain.cloud/auth/realms/quarkus",
  "sub": "eb4123a3-b722-4798-9af5-8957f823657a",
  "typ": "Bearer",
  "azp": "frontend",
  "nonce": "8a6136d6-bdf5-4794-8ba1-e8a985159d30",
  "session_state": "bff67131-3b62-437a-ae2b-
8b999059e61f",
  "acr": "0",
  "allowed-origins": [
    "'*'",
    "http://localhost:8080",
    "*"
  ],
  "realm_access": {
    "roles": [
      "user"
    ]
  },
  "scope": "openid email profile",
  "email_verified": false,
  "preferred_username": "alice"
}
```

Validate Token Content



Browser

Kubernetes

Microservices

Infrastructure Components

Web-App

Web-App

Validate token content display user Name.

access token
- Name: Alice
- Role: User
- ClientType: frontend

Web-API

Articles

Backend Client

KEYCLOAK

Realm

User: Alice

Role: User

ClientTypes:
- Frontend
- Backend

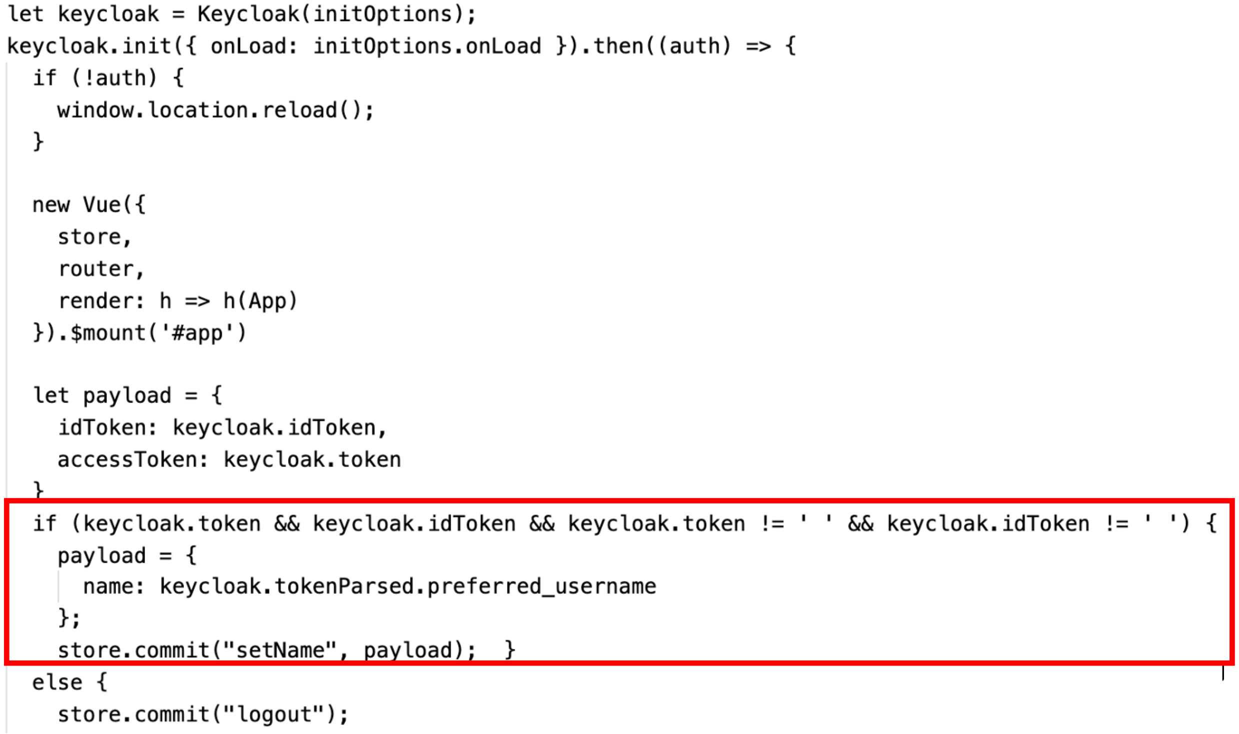
Redirect information

Validate Token Content

Browser

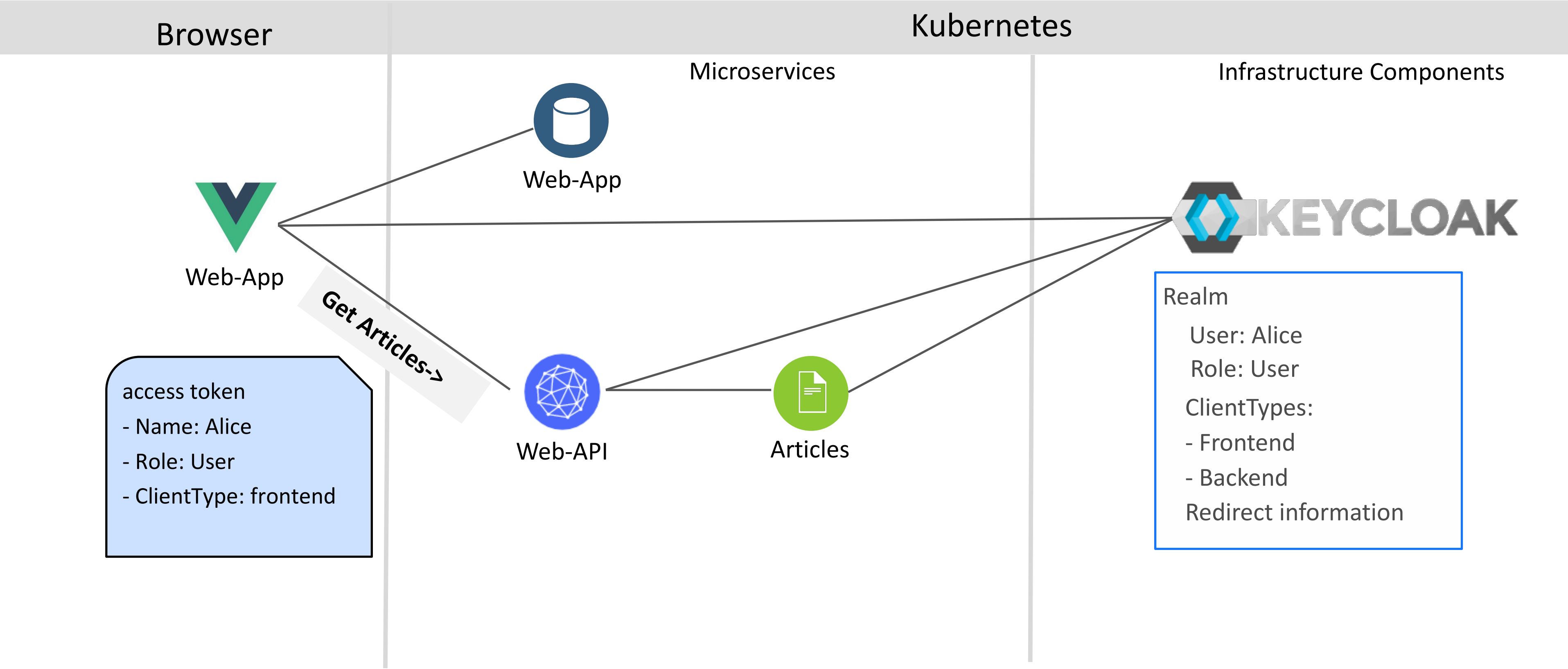
Web-App

Code: “main.js”



```
let keycloak = Keycloak(initOptions);
keycloak.init({ onLoad: initOptions.onLoad }).then((auth) => {
  if (!auth) {
    window.location.reload();
  }

  new Vue({
    store,
    router,
    render: h => h(App)
  }).$mount('#app')

  let payload = {
    idToken: keycloak.idToken,
    accessToken: keycloak.token
  }
  if (keycloak.token && keycloak.idToken && keycloak.token != ' ' && keycloak.idToken != ' ') {
    payload = {
      name: keycloak.tokenParsed.preferred_username
    };
    store.commit("setName", payload);  }
  else {
    store.commit("logout");
```

Invoke the Web-API



access token
- Name: Alice
- Role: User
- ClientType: frontend

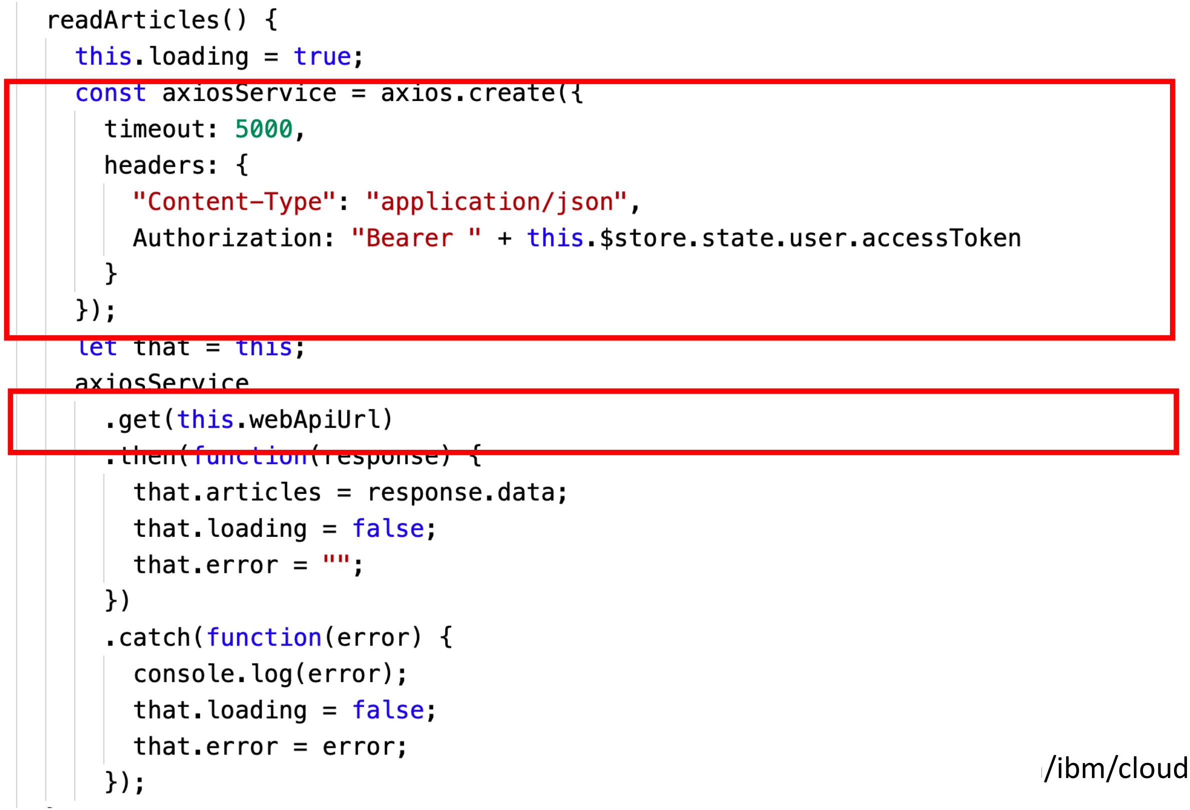
Realm
User: Alice
Role: User
ClientTypes:
- Frontend
- Backend
Redirect information

Invoke Web-API

Browser

Web-App

Code: “Home.vue”



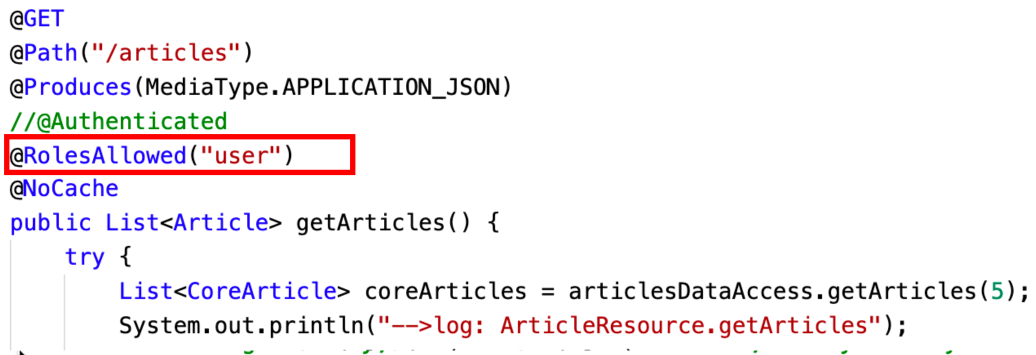
```
readArticles() {
  this.loading = true;
  const axiosService = axios.create({
    timeout: 5000,
    headers: {
      "Content-Type": "application/json",
      Authorization: "Bearer " + this.$store.state.user.accessToken
    }
  });
  let that = this;
  axiosService
    .get(this.webApiUrl)
    .then(function(response) {
      that.articles = response.data;
      that.loading = false;
      that.error = "";
    })
    .catch(function(error) {
      console.log(error);
      that.loading = false;
      that.error = error;
    });
```

Defintion of REST Endpoint for the Web-API

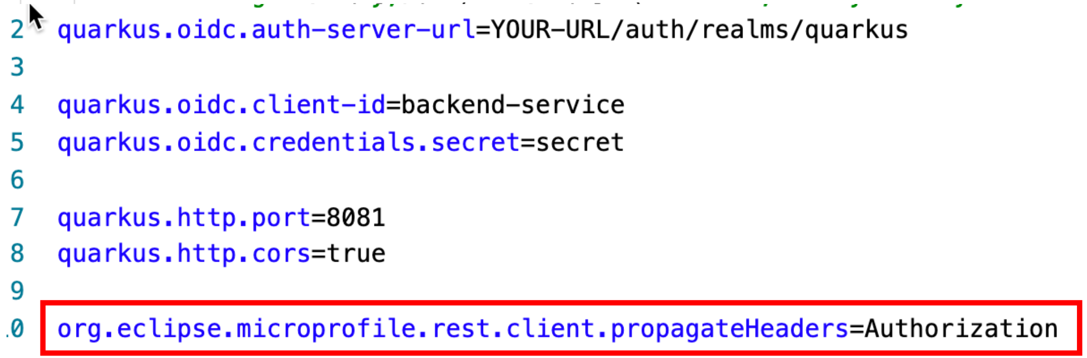
Kubernetes

Code: ArticleIsResource.java and application.properties

Web-API

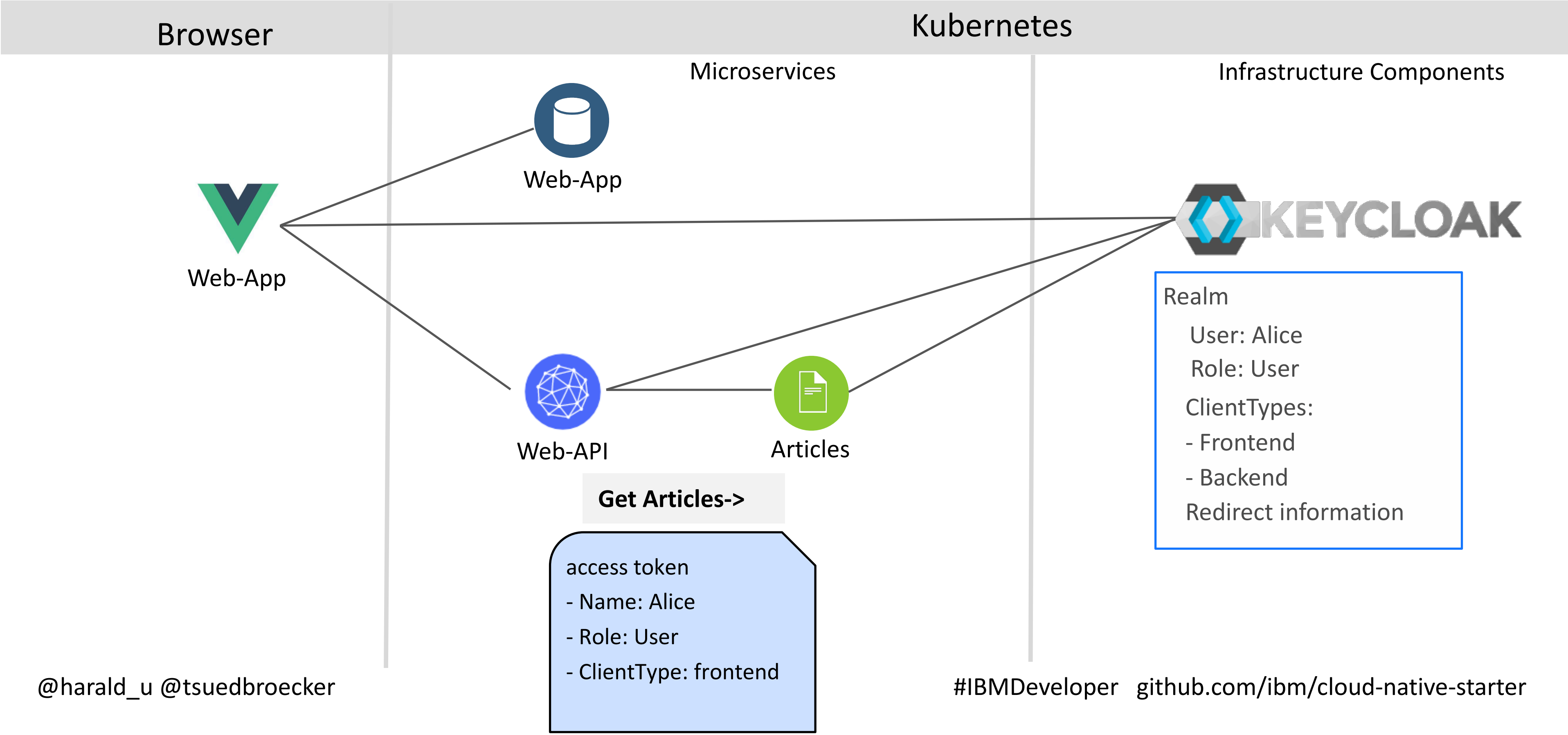


```
@GET
@Path("/articles")
@Produces(MediaType.APPLICATION_JSON)
//@Authenticated
@RolesAllowed("user")
@NoCache
public List<Article> getArticles() {
    try {
        List<CoreArticle> coreArticles = articlesDataAccess.getArticles(5);
        System.out.println("-->log: ArticleResource.getArticles");
```



```
quarkus.oidc.auth-server-url=YOUR-URL/auth/realms/quarkus

quarkus.oidc.client-id=backend-service
quarkus.oidc.credentials.secret=secret

quarkus.http.port=8081
quarkus.http.cors=true

org.eclipse.microprofile.rest.client.propagateHeaders=Authorization
```

Invoke Articles Service REST Endpoint



Realm

User: Alice

Role: User

ClientTypes:

- Frontend
- Backend

Redirect information

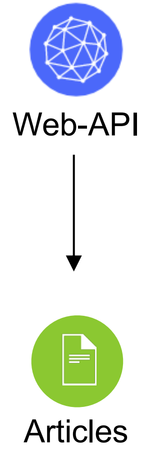
access token

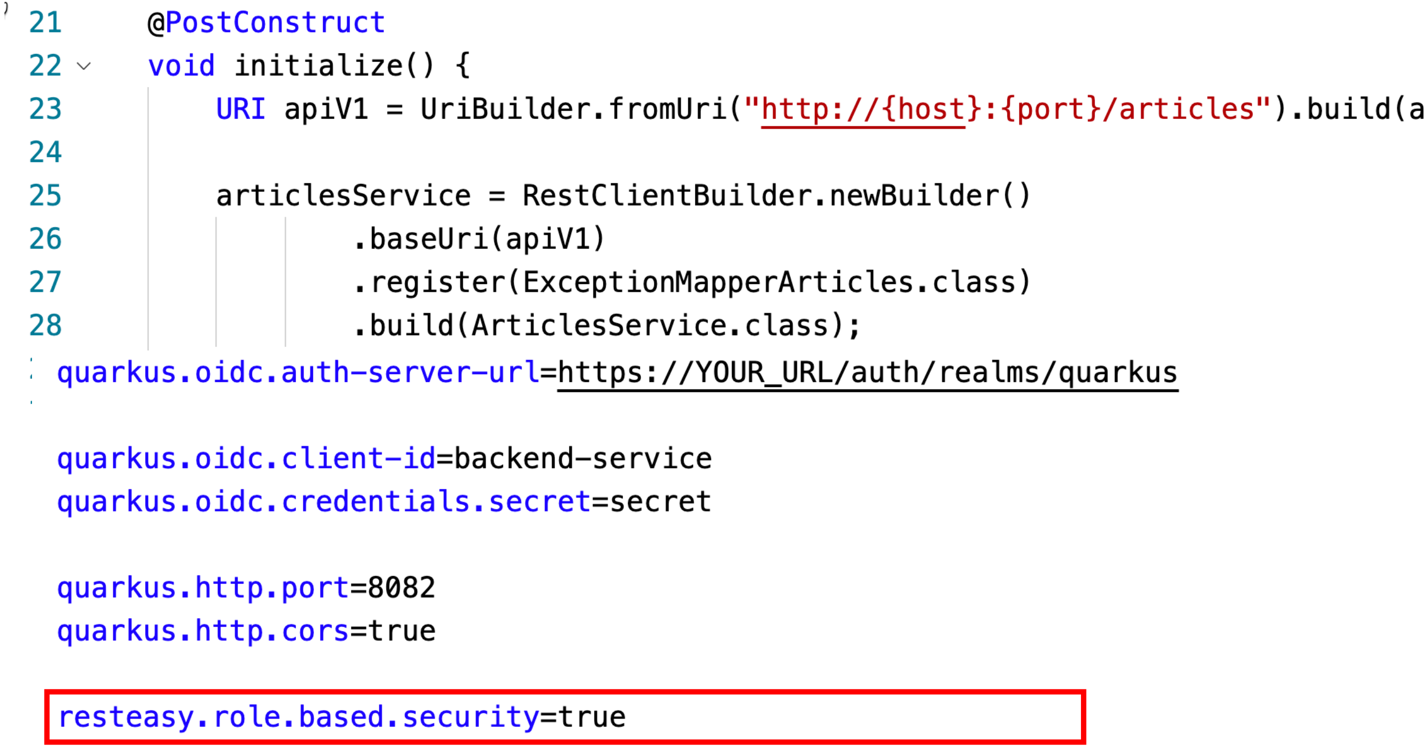
- Name: Alice
- Role: User
- ClientType: frontend

Invoke Articles Service REST Endpoint

Kubernetes

Code: **Web-API:** ArticlesDataAccess.java and **Articles**: application.propertie





```
    @PostConstruct
    void initialize() {
        URI apiV1 = UriBuilder.fromUri("http://{host}:{port}/articles").build(a

        articlesService = RestClientBuilder.newBuilder()
                .baseUri(apiV1)
                .register(ExceptionMapperArticles.class)
                .build(ArticlesService.class);
```

```
quarkus.oidc.auth-server-url=https://YOUR_URL/auth/realms/quarkus

quarkus.oidc.client-id=backend-service
quarkus.oidc.credentials.secret=secret

quarkus.http.port=8082
quarkus.http.cors=true

resteasy.role.based.security=true
```

@Harald_U @tsuedbroecker

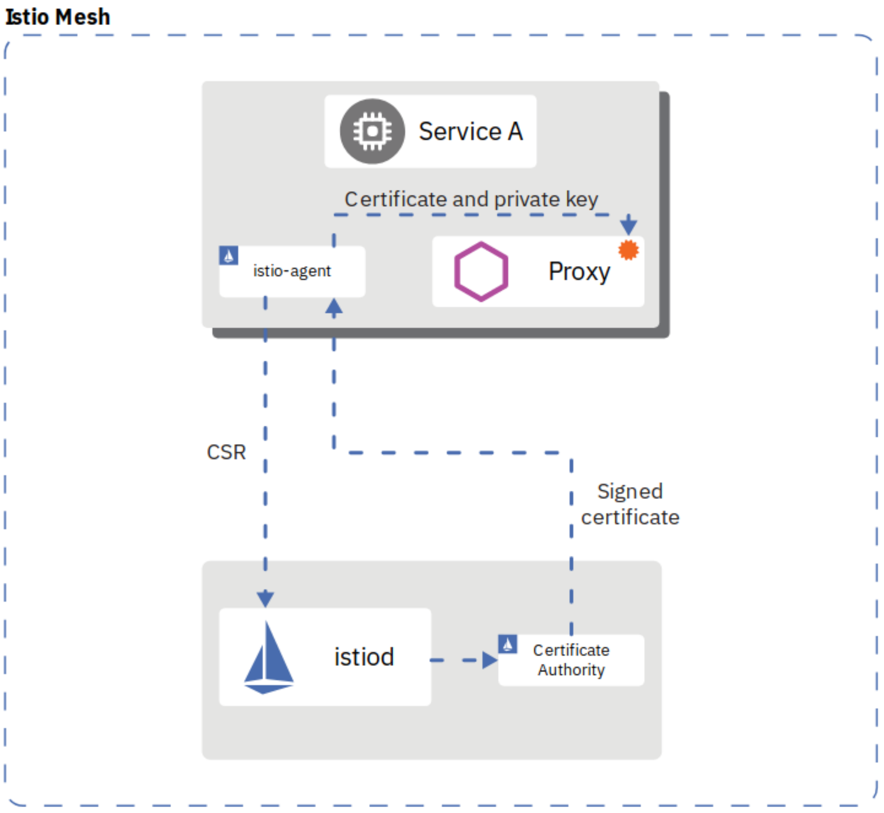
#IBMDeveloper github.com/ibm/cloud-native-starter

Platform Security

Istio Security

Certificate Authority (CA)

- Manages X.509 certificates
- Key and certificate rotation



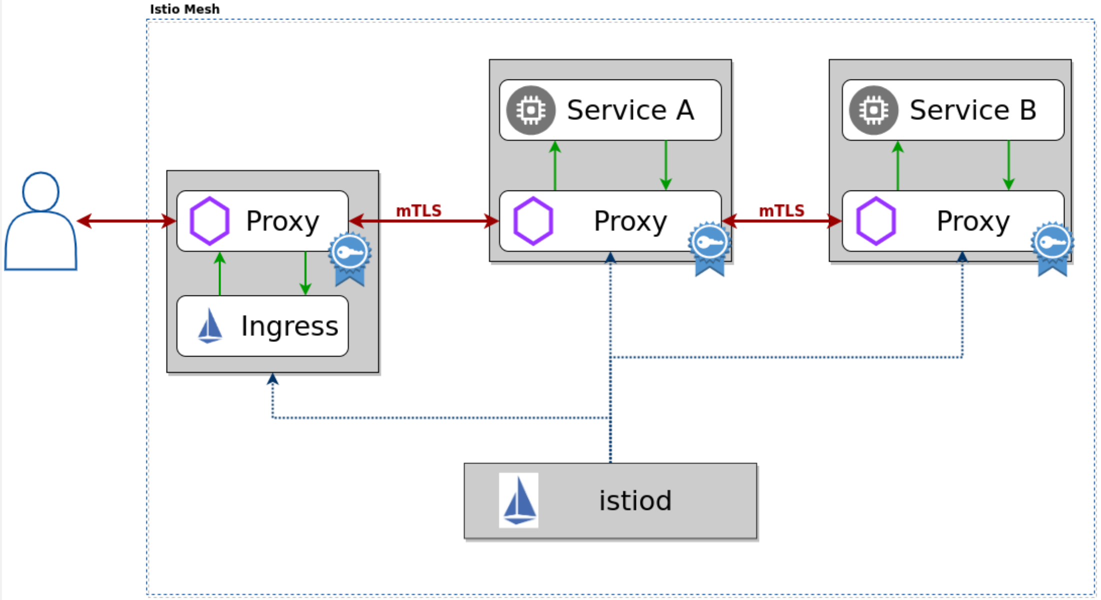
#IBMDeveloper github.com/ibm/cloud-native-starter

@harald_u @tsuedbroecker

Istio Security

Mutual TLS (mTLS) authentication

- Traffic between services is routed through Envoy proxies
- Use Istio Ingress to route traffic into Service Mesh, includes Envoy
- Envoys establish mTLS connection
- Connection is encrypted and identity of service verified
- mTLS is enabled by default (permissive)



@harald_u @tsuedbroecker

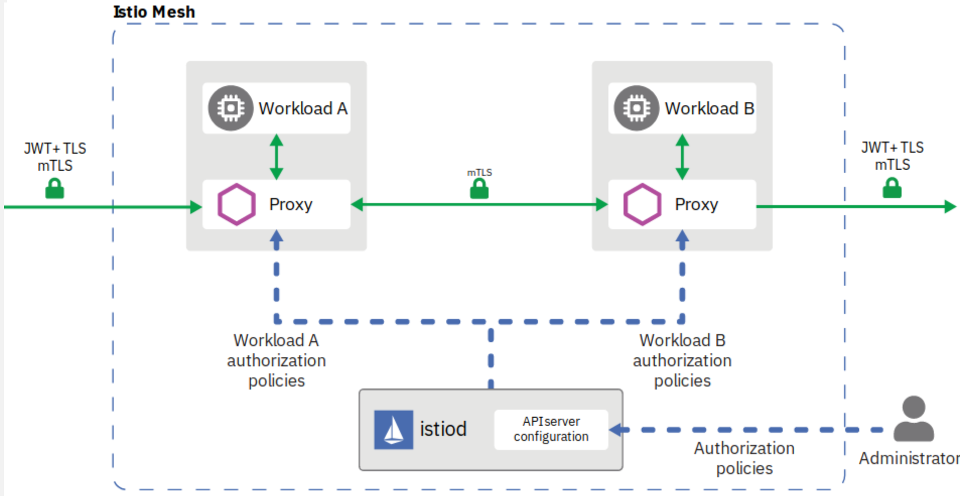
#IBMDeveloper github.com/ibm/cloud-native-starter

Istio Security

Authorization

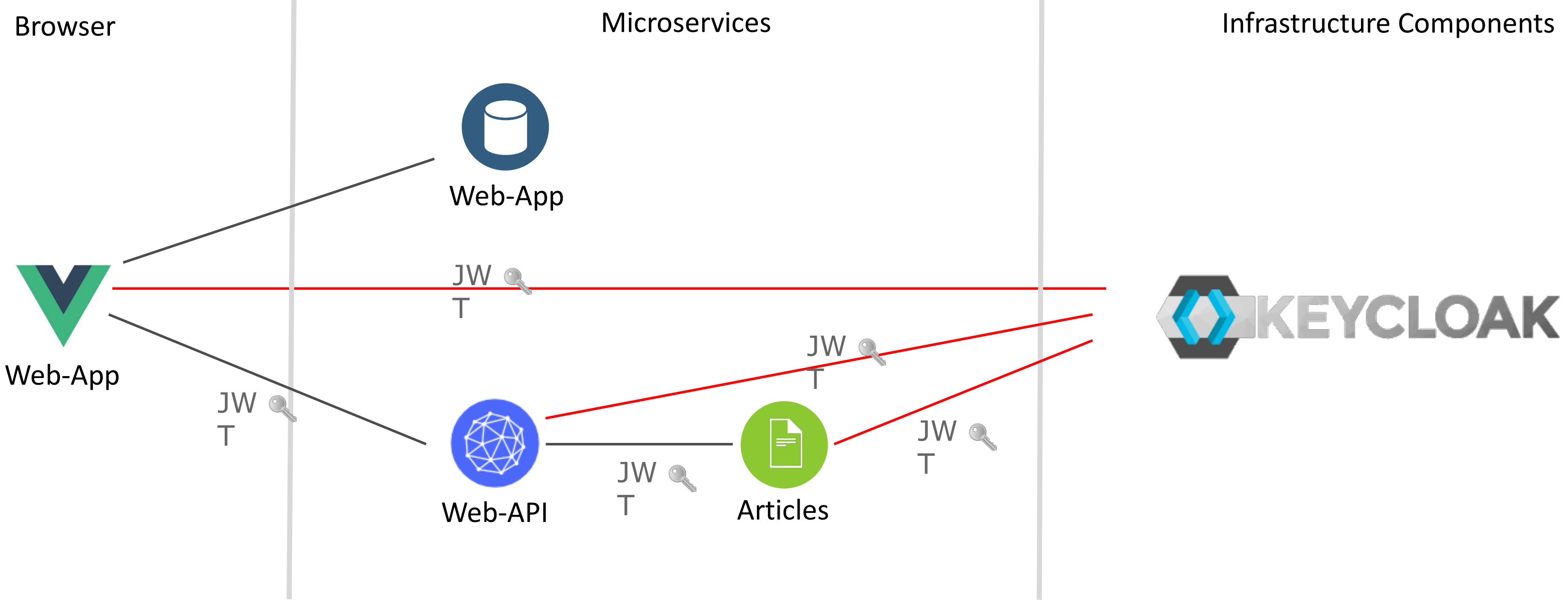
Policies based on

- mTLS certificates (internal)
- JWT (external, e.g. from Keycloak)



@harald_u @tsuedbroecker

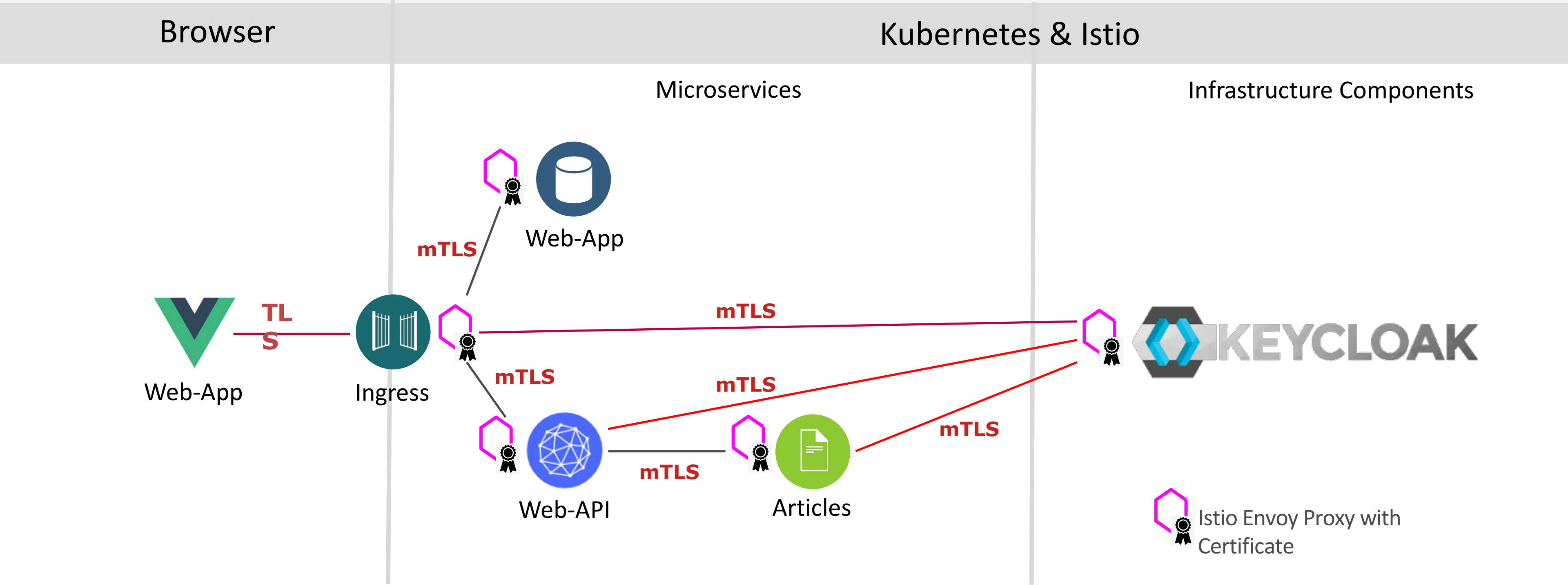
Application Security with Keycloak



@harald_u @tsuedbroecker

#IBMDeveloper github.com/ibm/cloud-native-starter

Platform Security with Istio



@harald_u @tsuedbroecker

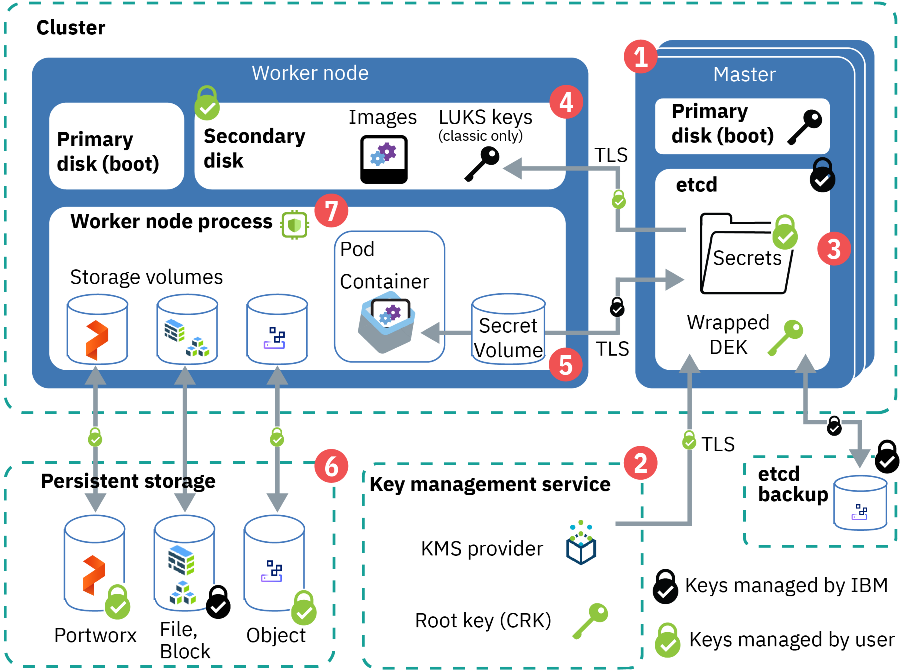
#IBMDeveloper github.com/ibm/cloud-native-starter

IBM Cloud Kubernetes Service

Protecting sensitive information:

https://cloud.ibm.com/docs/containers?topic=containers-encryption

- Encrypted disks
- Encrypted persistent storage (volumes)
- Automatically generate TLS certificates for Kubernetes services type LoadBalancer
- Certificate Management System (optional): Order, manage, rotate certificates
  - IBM Certificate Manager
- Key Management System (optional): Encrypt etcd and Kubernetes secrets
  - IBM Key Protect
  - IBM Cloud Hyper Protect Crypto Service



#IBMDeveloper github.com/ibm/cloud-native-starter

Try out the end-to-end security example for a Microservices application in the open source Cloud Native Starter project!

Summary

Authentication and Authorization with

- Quarkus
- MircoProfile
- Keycloak
- OpenID Connect
- JWT

Platform Security

- IBM Cloud
- Kubernetes
- Istio

IBM Developer

developer.ibm.com

IBM Cloud Lite account

ibm.biz/tbd

IBM