Get started with Security for your Java Microservices Application

Harald Uebele
Developer Advocate, IBM
@Harald_U

Thomas Südbröcker
Developer Advocate, IBM
@tsuedbroecker

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

What is Application Security?

"Application security encompasses measures taken to improve the security of an application often by finding, fixing and preventing security vulnerabilities."

Source: https://en.wikipedia.org/wiki/Application_security

Terms

Asset

“Resource of value such as the data in a database, money in an account, file on the filesystem or any system resource.”

Vulnerability

“A weakness or gap in security program that can be exploited by threats to gain unauthorized access to an asset.”

Attack (or exploit)

An action taken to harm an asset.

Threat

Anything that can exploit a vulnerability and obtain, damage, or destroy an asset.

Source: https://en.wikipedia.org/wiki/Application_security

Categories

Category	
Input Validation	Buffer overflow; cross-site
Software Tampering	Attacker modifies an existi… code extension
Authentication	Network eavesdropping; B…
Authorization	Elevation of p…
Configuration management	Unauthorized… individual acc…
Sensitive information	Access sensitive code or d…
Session management	Session hijacking; session…
Cryptography	Poor key generation or key…

INFORMATIONAL
Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 6819
Category: Informational
ISSN: 2070-1721

T. Lodderstedt, Ed.
Deutsche Telekom AG
M. McGloin
IBM
P. Hunt
Oracle Corporation
January 2013

OAuth 2.0 Threat Model and Security Considerations

Abstract

© https://tools.ietf.org/html/rfc6819

Threat & Attacks are not in our scope

Source: https://en.wikipedia.org/wiki/Application_security

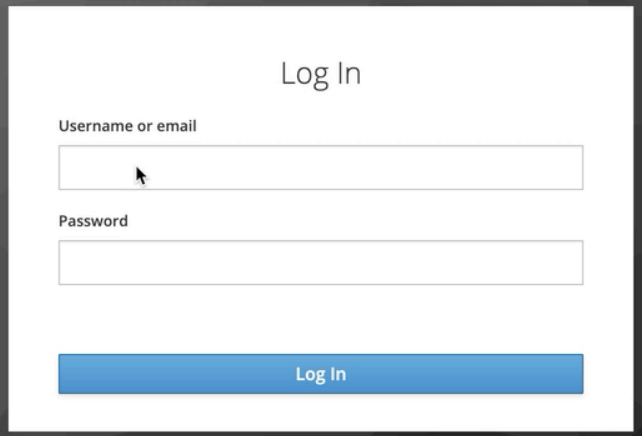
Developer point of view

Category	Threats & Attacks
Input Validation	Buffer overflow; cross-site scripting; SQL injection; canonicalization
Software Tampering	Attacker modifies an existing application's runtime behavior to perform unauthorized actions; exploited via binary patching, code substitution, or code extension
Authentication	Network eavesdropping; Brute force attack; dictionary attacks; cookie replay; credential theft
Authorization	Elevation of privile
Configuration management	Unauthorized acc individual account
Sensitive information	Access sensitive
Session management	Sessio hijacking
Cryptography	Poor key generation or key management; weak or custom encryption

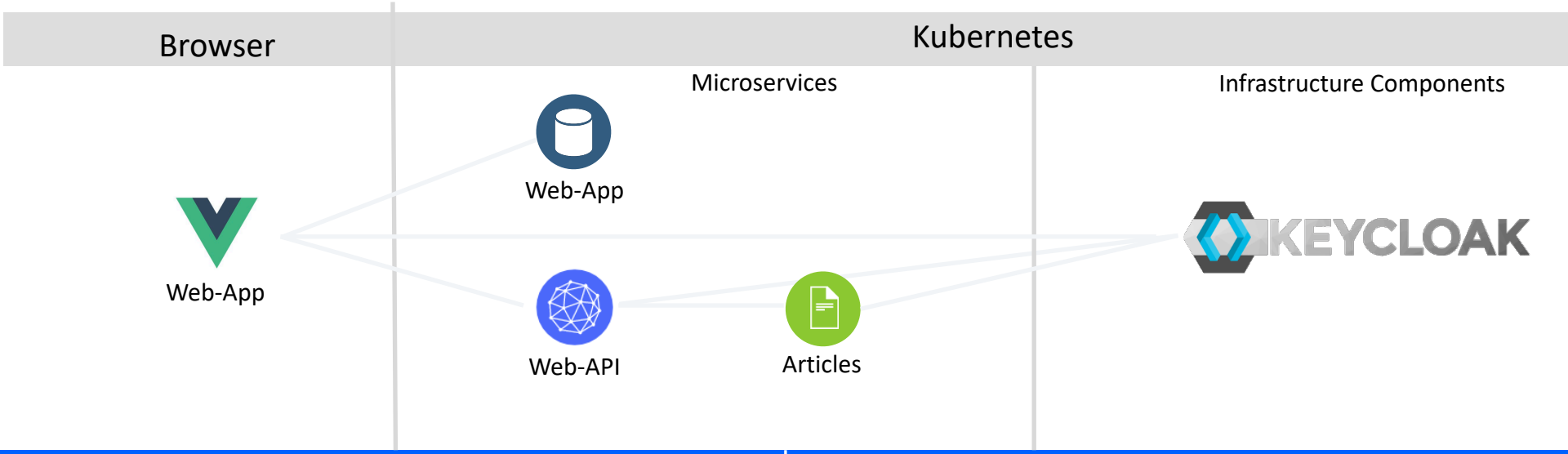
How to implement or configure these categories for a Microservices based Cloud Native application?

Source: https://en.wikipedia.org/wiki/Application_security

The example Cloud Native Starter – Web application



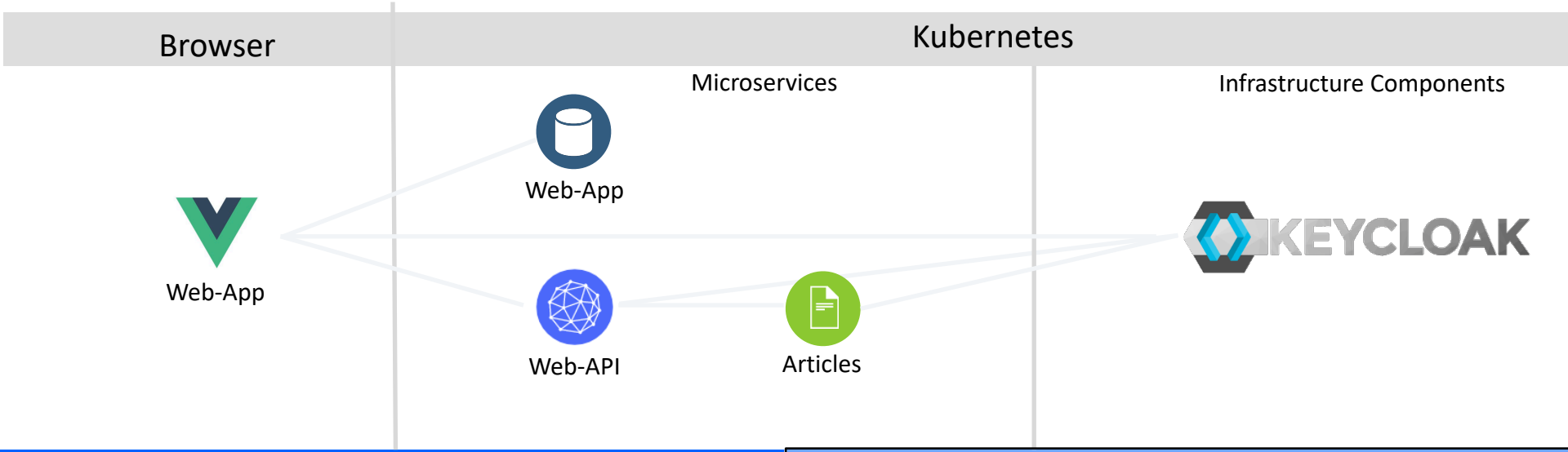
Let’s make it concrete



Cryptography

Authentication and Authorization

Let’s make it concrete



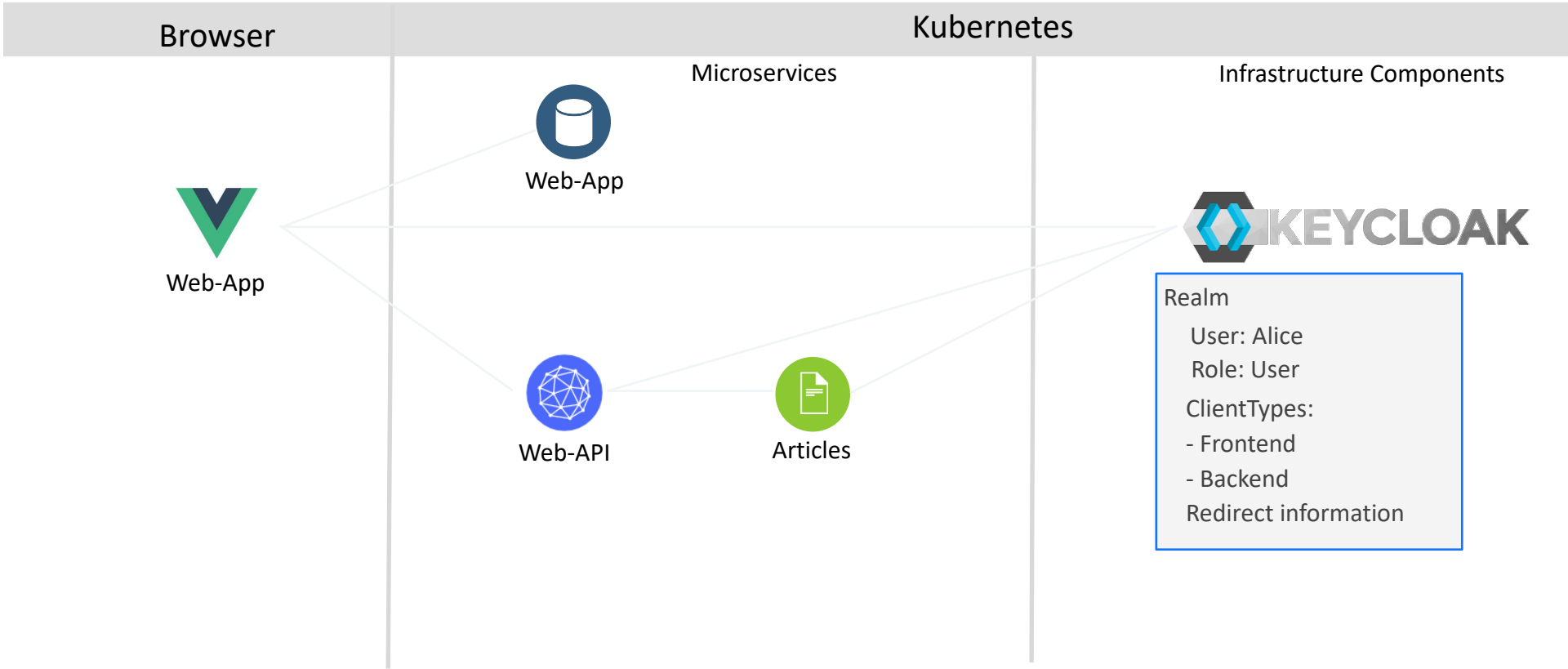
Cryptography

@Harald_U @tsuedbroecker

Authentication and Authorization

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

Let’s make it concrete



Authentication with Keycloak

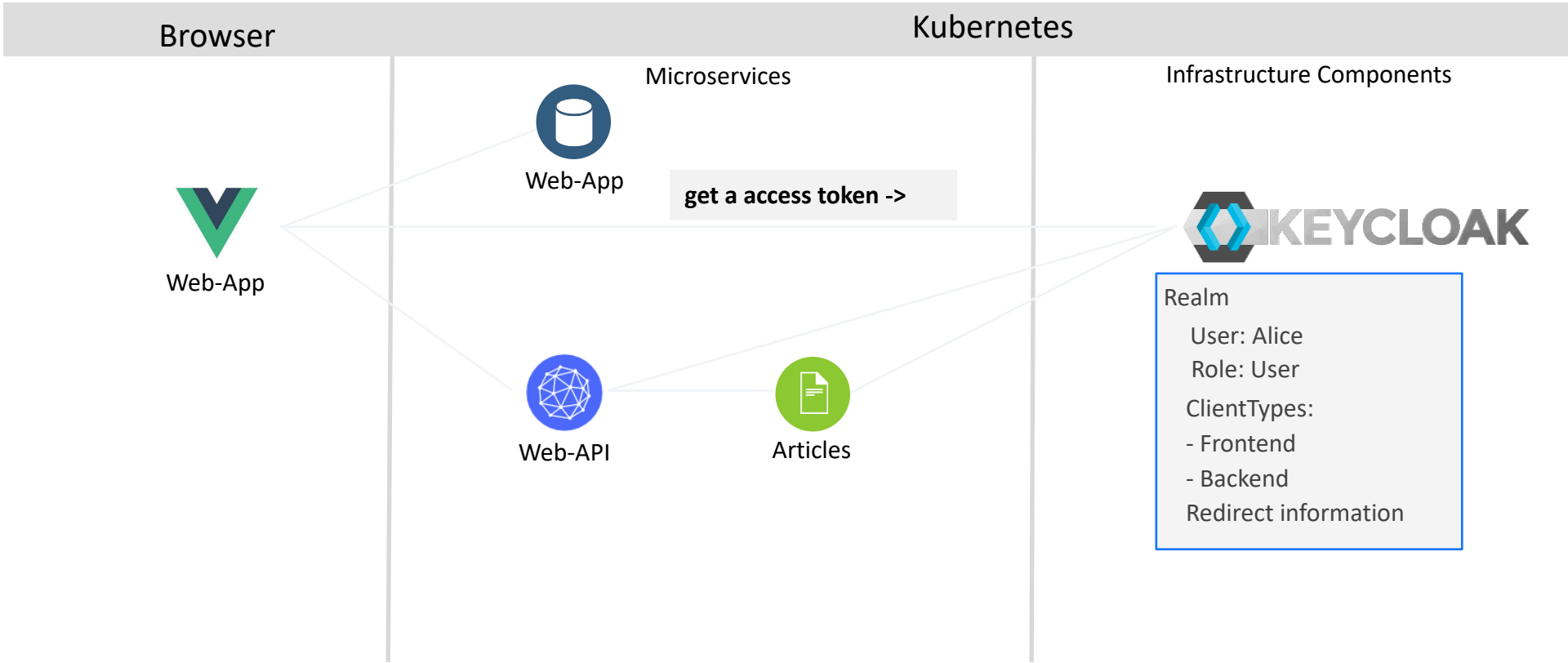
Browser

Web-App

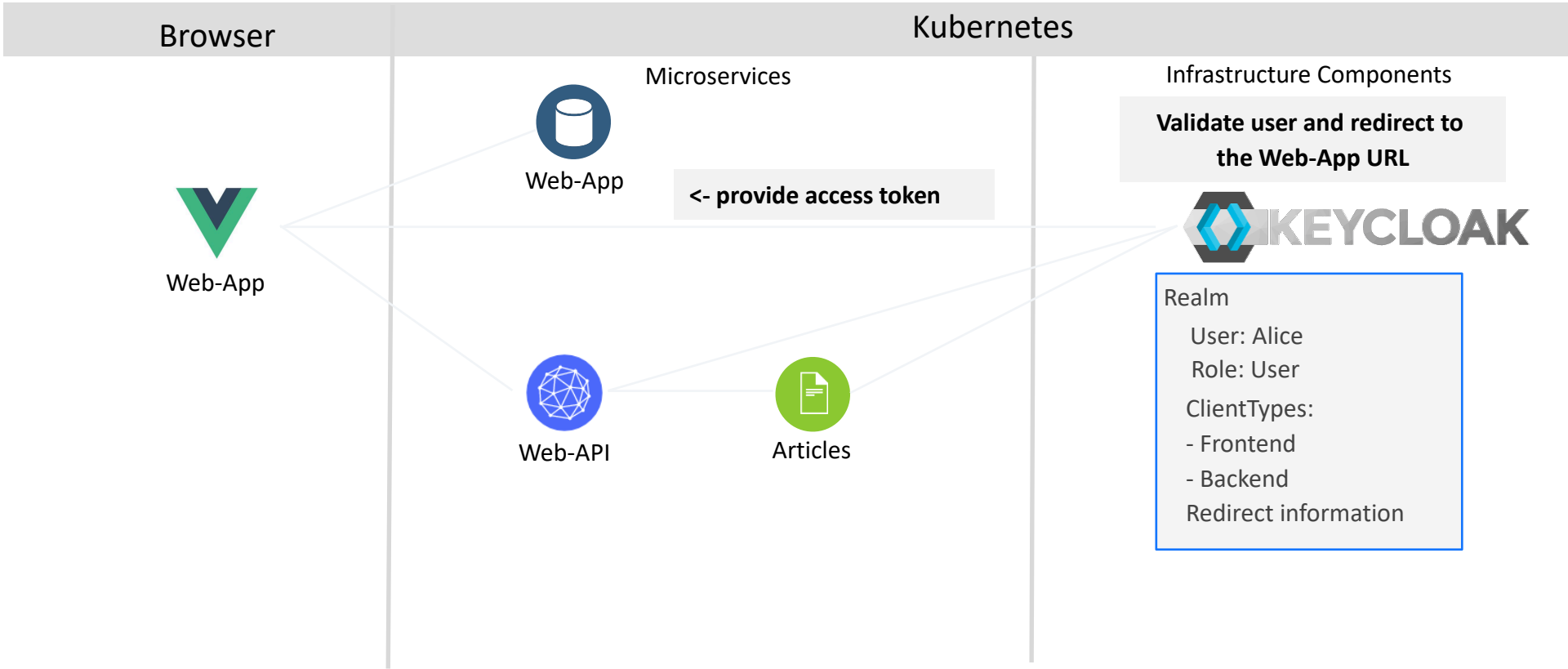
Code: “main.js”

```
1  import Keycloak from 'keycloak-js';
2
3  let initOptions = {
4    url: 'https://keycloak-url/auth',
5      realm: 'quarkus', clientId: 'frontend', onLoad: 'login-required'
6  }
7
8  Vue.config.productionTip = false
9  Vue.config.devtools = true
10 Vue.use(BootstrapVue);
11
12 let keycloak = Keycloak(initOptions);
13 keycloak.init({ onLoad: initOptions.onLoad }).then((auth) => {
14   if (!auth) {
15     window.location.reload();
16   }
17
18   new Vue({
19     store,
20     router,
21     render: h => h(App)
22   }).$mount('#app')
23
24   let payload = {
25     idToken: keycloak.idToken,
26     accessToken: keycloak.token
27   }
28   if (keycloak.token && keycloak.idToken && keycloak.token != ' ' && keycloak.idToken != ' ') {
29     payload = {
30       name: keycloak.tokenParsed.preferred_username
31     };
32     store.commit("setName", payload);  }
33   else {
34     store.commit("logout");
```

Authentication with Keycloak



Redirect


Browser
Kubernetes
Microservices
Infrastructure Components
Web-App
Web-App
<- provide access token
Validate user and redirect to the Web-App URL
KEYCLOAK
Realm
User: Alice
Role: User
ClientTypes:
- Frontend
- Backend
Redirect information
Web-API
Articles

Access Token

HEADER: ALGORITHM & TOKEN TYPE

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

Source: jwt.io

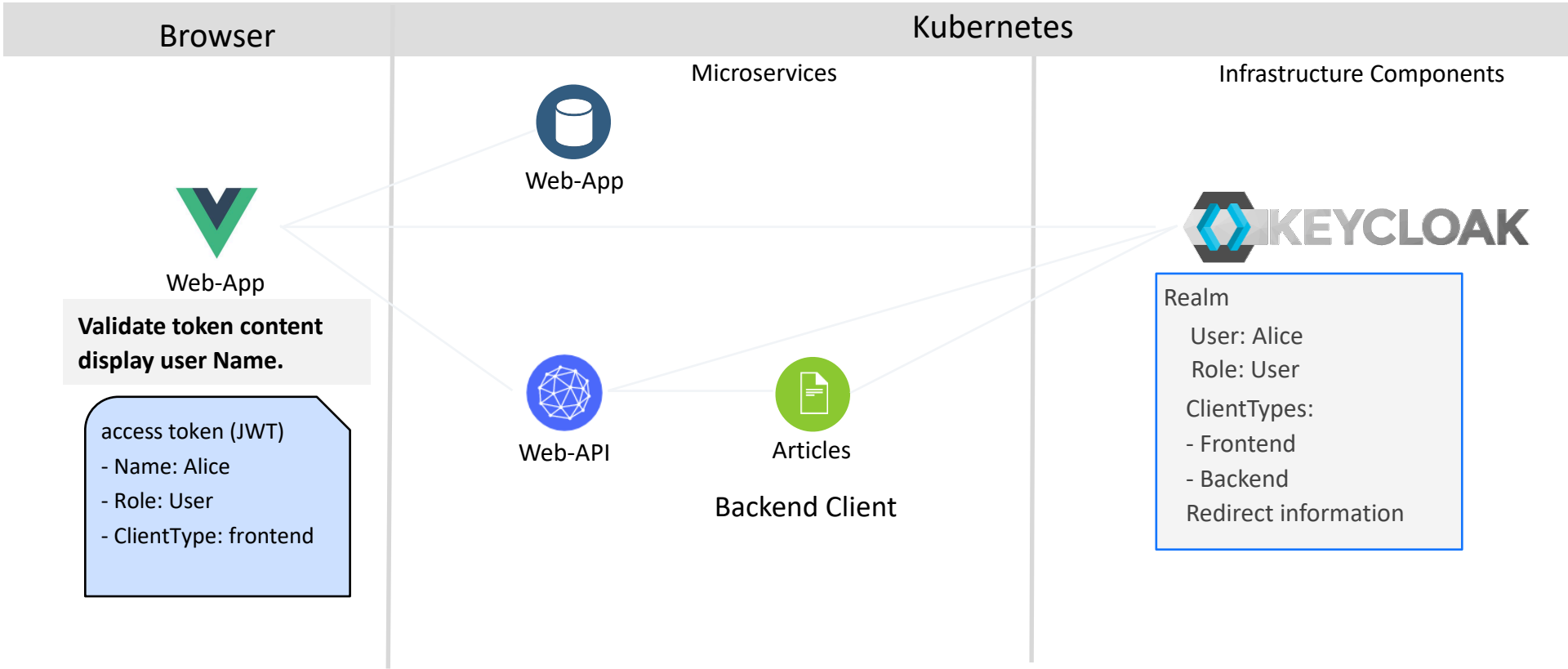
VERIFY SIGNATURE

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

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



Validate Token Content

Browser

Web-App

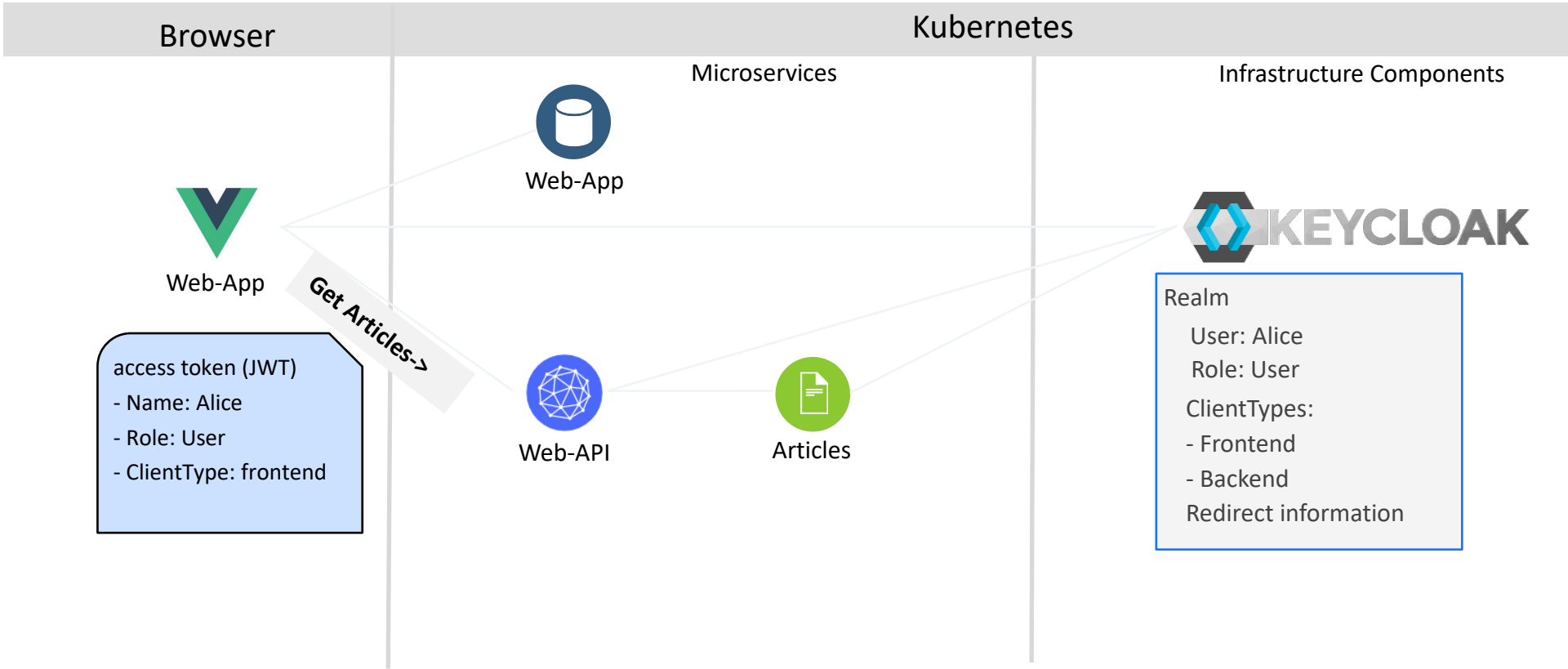
Code: “main.js”

```
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



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

Web-API

Code: ArticelsResource.java and application.properties

```
@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");
```

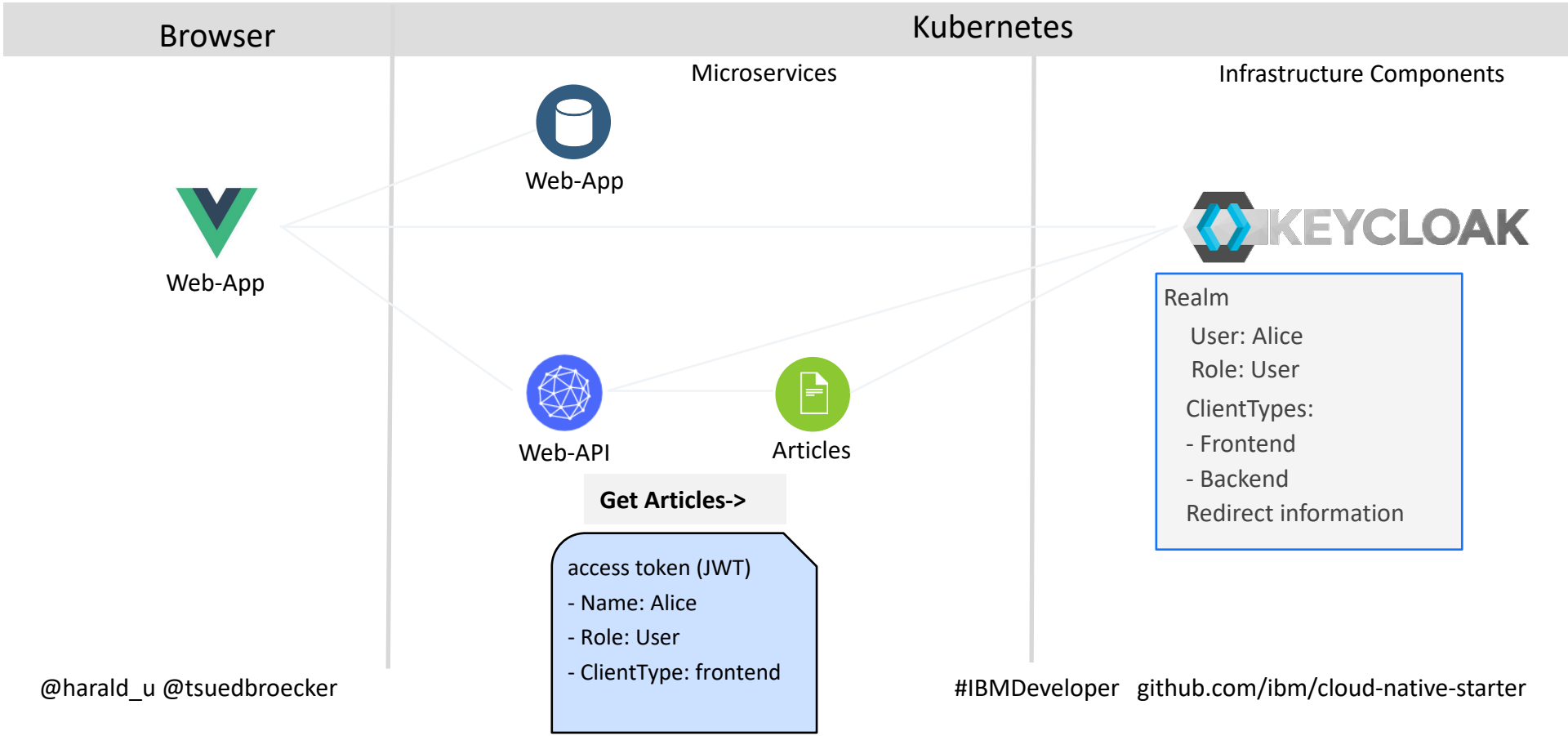
```
quarkus.oidc.auth-server-url=YOUR-URL/auth/realms/quarkus

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



Invoke Articles Service REST Endpoint

Kubernetes

Web-API

Articles

Code: **WebAPI:** ArticlesDataAccess.java and **Articles**: application.properties

```
21      @PostConstruct
22      void initialize() {
23          URI apiV1 = UriBuilder.fromUri("http://{host}:{port}/articles").build(a
24
25          articlesService = RestClientBuilder.newBuilder()
26                  .baseUri(apiV1)
27                  .register(ExceptionMapperArticles.class)
28                  .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
```

Platform Security

IBM Cloud

Compliance: GDPR, HIPAA, PCI, SOC2, ISO 9001, etc.
( )

Identity and Access Management (IAM) for the platform

Key Management System aaS

IBM Cloud Kubernetes Service (IKS)

Protecting sensitive information

Istio Security

Encryption

Access control

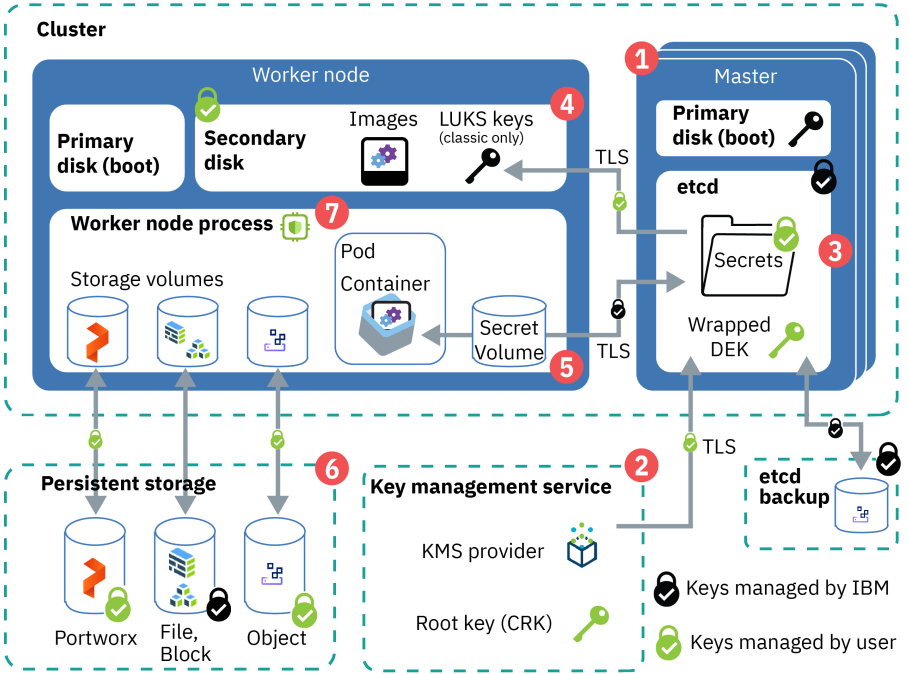
Security by default: no changes needed to application code and infrastructure

IBM Cloud Kubernetes Service (IKS)

Protecting sensitive information

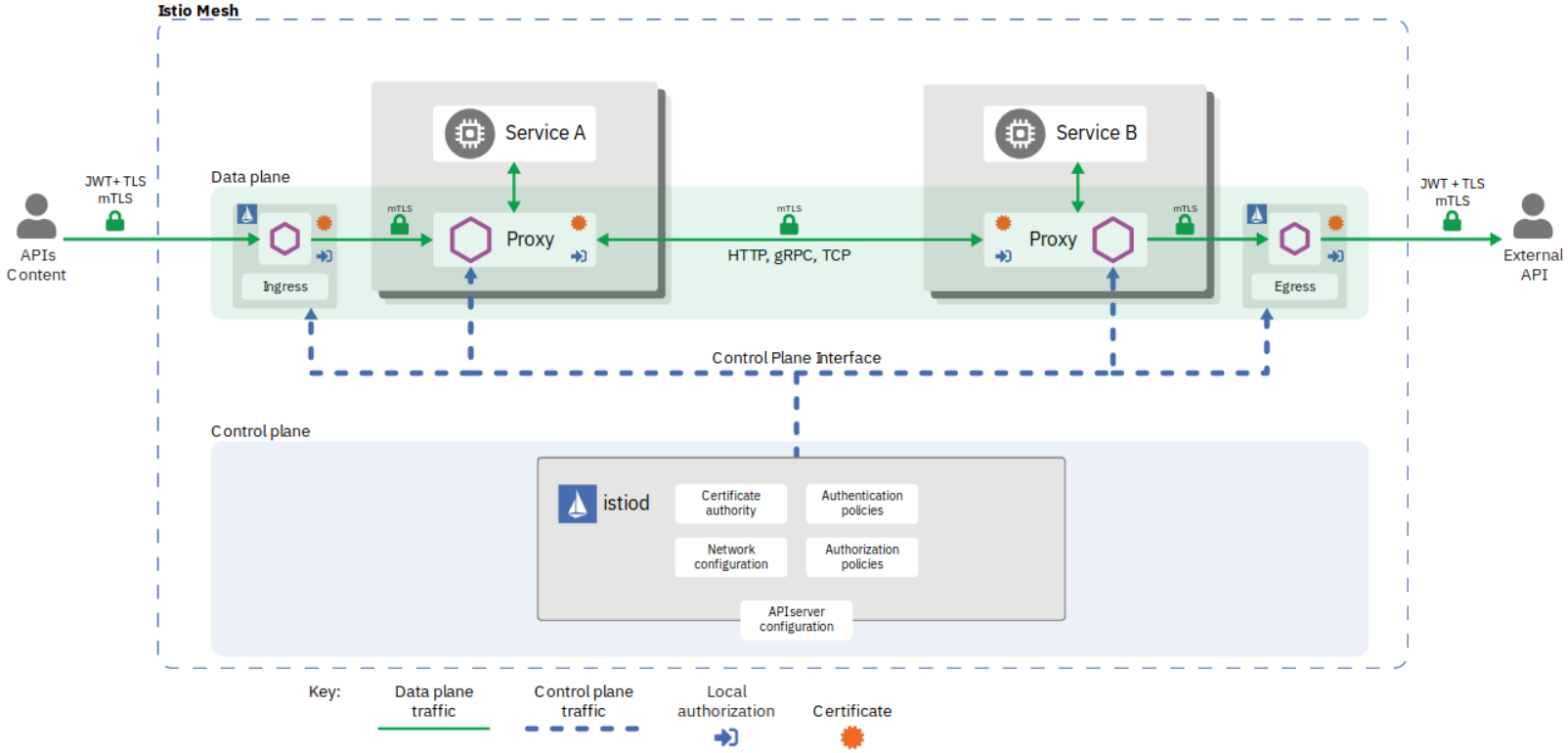
https://cloud.ibm.com/docs/containers?topic=containers-encryption

- Encrypted disks
- Optional Key Management System (KMS) to encrypt etcd and Kubernetes secrets
  - IBM Key Protect
  - IBM Cloud Hyper Protect Crypto Service
- Encrypted persistent storage
- Automatically generate TLS certificates for Kubernetes services type LoadBalancer
- IBM Cloud Container Registry
  - Signed Images (Integrity)
  - Vulnerability Advisor (Image security status)



Istio Security Architecture

https://istio.io/latest/docs/concepts/security

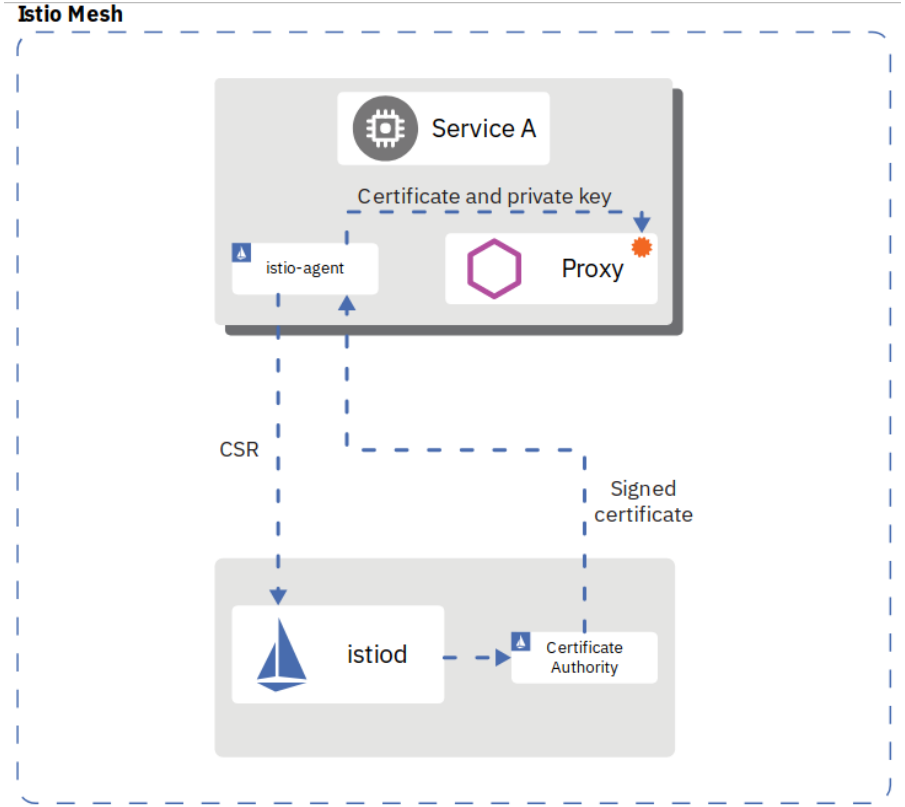


Istio Security

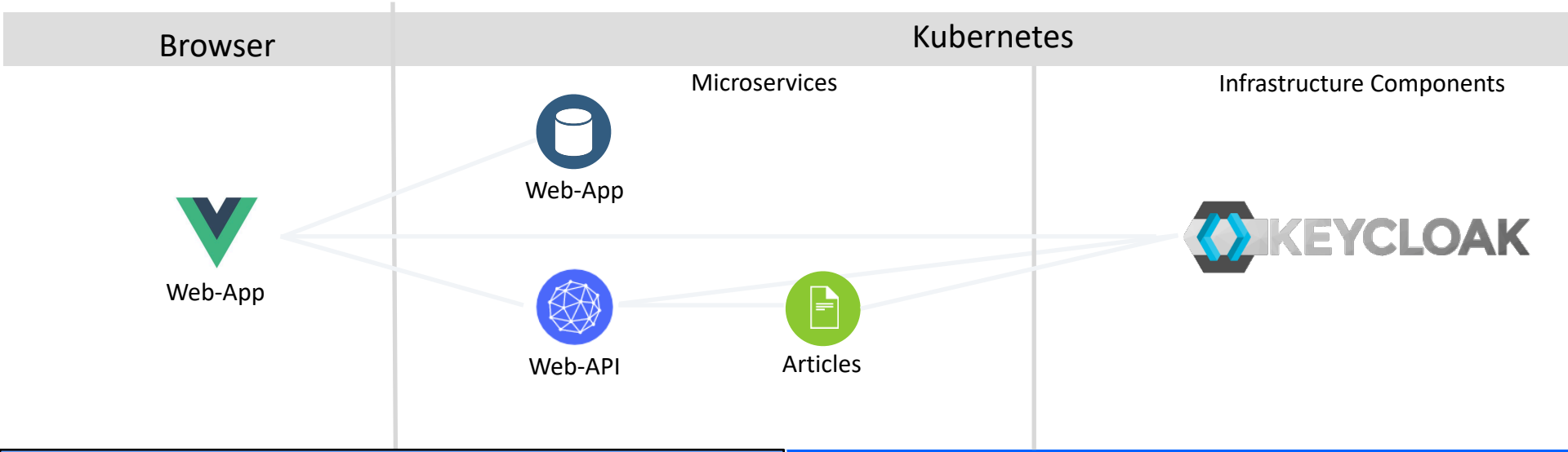
Identity and Access Management
- Certificate Authority
- Manages X.509 certificates
- Key and certificate rotation

Mutual TLS (mTLS) authentication
- Traffic between services is routed through Envoy proxies
- Envoys establish mTLS connection
- Connection is encrypted and identity of service verified
- mTLS is enabled by default

Authorization policies based on
- mTLS certificates (internal)
- JWT (external, e.g. from Keycloak)



Let’s make it concrete



Cryptography

@Harald_U @tsuedbroecker

Authentication and Authorization

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

“SUPERSONIC SUBATOMIC JAVA.“

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

quarkus.io

OpenJDK

kubernetes

“Optimizing Enterprise Java for a Microservices Architecture.“

“[...] by innovating [...] with a goal of standardization [...] microservices security are based on OAuth2, OpenID Connect(OIDC) and JSON Web Tokens(JWT) standards.”

microprofile.io

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

“... Add authentication to applications and secure services with minimum fuss. No need to deal with storing users or authenticating users … ”

https://www.keycloak.org/

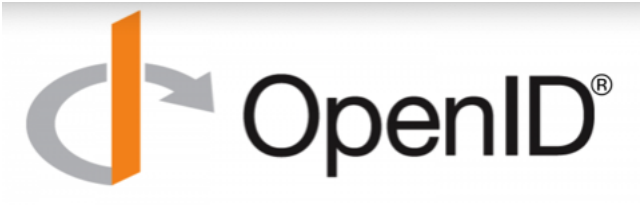
@harald_u @tsuedbroecker

Supported protocols:
Open ID Connect and SAML

“ a simple identity layer on top of the OAuth 2.0 protocol”

“It allows Clients to verify the identity of the End-User based on the authentication OpenID Connect specifies”

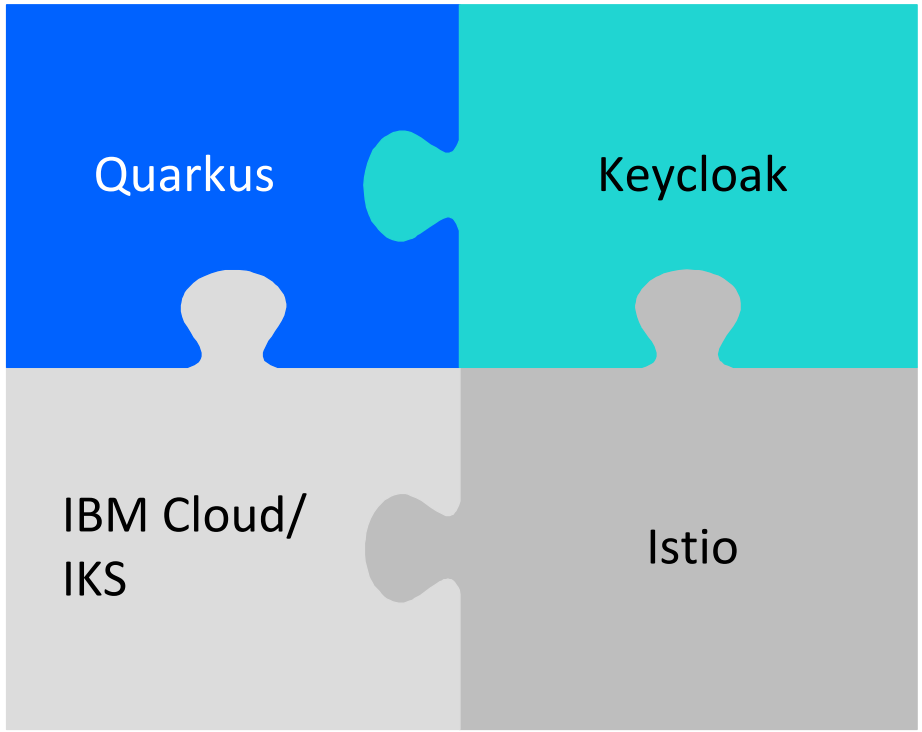
https://openid.net/connect/



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

https://jwt.io/

Technologies to secure the Microservice Application



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

Summary

Authentication and Authorization with

- Qurakus
- MircoProfile
- Keycloak
- OpenID Connect
- JWT

Cryptography

- IBM Cloud
- IKS
- Istio

IBM Developer

developer.ibm.com

IBM Cloud Lite account

ibm.biz/tbd