Architektur

### Ablauf der Authentifizierung (Keycloak broker und External IdP)

1. **Benutzerauthentifizierung**:
   * **Schritt 1**: Der Benutzer besucht internet-website.com und klickt auf "Login".
   * **Schritt 2**: Die Website leitet den Benutzer zu https://keycloak.companyxyz.com/auth weiter.
   * **Schritt 3**: Keycloak zeigt die Anmeldeseite an oder leitet zum externen Identitätsanbieter weiter.
   * **Schritt 4**: Der Benutzer authentifiziert sich beim externen Identitätsanbieter (z. B. Google).
   * **Schritt 5**: Der externe Identitätsanbieter sendet ein Authentifizierungs-Token an Keycloak.
   * **Schritt 6**: Keycloak validiert das Token und erstellt eine Sitzung für den Benutzer.
   * **Schritt 7**: Keycloak leitet den Benutzer mit einem Token zurück zu internet-website.com.
2. **Zugriff auf Backend-Service**:
   * **Schritt 1**: Authentifizierter Benutzer fordert den Zugriff auf eine Ressource über https://api.companyxyz.com/resource an.
   * **Schritt 2**: Das API-Gateway erhält die Anfrage und überprüft das Benutzertoken bei Keycloak.
   * **Schritt 3**: Wenn das Token gültig ist, leitet das API-Gateway die Anfrage an den entsprechenden Backend-Dienst weiter.
   * **Schritt 4**: Der Backend-Dienst verarbeitet die Anfrage und sendet eine Antwort über das API-Gateway zurück.

**Komponentenübersicht**

1. **Kubernetes-Cluster**:
   * Hostet Keycloak, API-Gateway und Backend-Service.
   * Läuft innerhalb des lokalen Netzwerks von Unternehmen XYZ.
2. **Keycloak**:
   * Fungiert als zentraler Identitätsbroker.
   * Integriert mit einem externen Identitätsanbieter (z. B. Google, Azure AD).
3. **MS SQL Server 2022**:
   * Managed Service für die Keycloak-Datenbank.
   * Innerhalb des lokalen Netzwerks des Unternehmens zugänglich.
4. **API-Gateway**:
   * Sichert und verwaltet den API-Zugriff auf Backend-Service.
   * Authentifiziert und autorisiert Anfragen über Keycloak.
5. **Externer Identitätsanbieter**:
   * Bietet Authentifizierungsdienste (z. B. Google, Azure AD).
6. **Internet-Website**:
   * Öffentliche Anwendung für Benutzerinteraktionen.
   * Verwendet Keycloak zur Authentifizierung.
7. **Lokale Netzwerksicherheit**:
   * Stellt sicher, dass die Kommunikation zwischen internen Services sicher ist.

**Konfiguration**

**Kubernetes-Cluster-Konfiguration**

**1. Keycloak-Bereitstellung**:

* Deployment YAML:

apiVersion: apps/v1

kind: Deployment

metadata:

name: keycloak

namespace: keycloak

spec:

replicas: 3

selector:

matchLabels:

app: keycloak

template:

metadata:

labels:

app: keycloak

spec:

containers:

- name: keycloak

image: jboss/keycloak:latest

ports:

- containerPort: 8080

env:

- name: DB\_VENDOR

value: mssql

- name: DB\_ADDR

value: mssql-server

- name: DB\_DATABASE

value: keycloak

- name: DB\_USER

valueFrom:

secretKeyRef:

name: keycloak-db-secret

key: username

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: keycloak-db-secret

key: password

* Service YAML**:**

apiVersion: v1

kind: Service

metadata:

name: keycloak

namespace: keycloak

spec:

selector:

app: keycloak

ports:

- protocol: TCP

port: 80

targetPort: 8080

* Ingress YAML**:**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: keycloak-ingress

namespace: keycloak

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: keycloak.companyxyz.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: keycloak

port:

number: 80

tls:

- hosts:

- keycloak.companyxyz.com

secretName: keycloak-tls

**2. MS SQL Server 2022 Configuration**:

* **Service Connection**:

apiVersion: v1

kind: Service

metadata:

name: mssql-server

namespace: keycloak

spec:

ports:

- port: 1433

selector:

app: mssql-server

* Secret für Datenbank-Zugangsdaten**:**

apiVersion: v1

kind: Secret

metadata:

name: keycloak-db-secret

namespace: keycloak

type: Opaque

data:

username: bXl1c2Vy # base64 codiert

password: bXlwYXNzd29yZA== # base64 codiert

**3. API-Gateway-Bereitstellung**:

* **Deployment YAML**:

apiVersion: apps/v1

kind: Deployment

metadata:

name: api-gateway

namespace: backend-services

spec:

replicas: 2

selector:

matchLabels:

app: api-gateway

template:

metadata:

labels:

app: api-gateway

spec:

containers:

- name: api-gateway

image: your-api-gateway-image:latest

ports:

- containerPort: 80

* Service YAML**:**

apiVersion: v1

kind: Service

metadata:

name: api-gateway

namespace: backend-services

spec:

selector:

app: api-gateway

ports:

- protocol: TCP

port: 80

targetPort: 80

* **Ingress YAML**:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: api-gateway-ingress

namespace: backend-services

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: api.companyxyz.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: api-gateway

port:

number: 80

tls:

- hosts:

- api.companyxyz.com

secretName: api-gateway-tls

**Netzwerksicherheit**

* **SSL/TLS-Zertifikate**:
  + Verwende Let's Encrypt für öffentliche Zertifikate oder eine interne Zertifizierungsstelle (CA) für interne Zertifikate.
  + Stelle sicher, dass alle Verbindungen zu keycloak.companyxyz.com und api.companyxyz.com über HTTPS erfolgen..
  + Verwende Kubernetes Secrets, um sensible Daten wie Zugangsdaten und API-verchlüsselt sicher zu speichern.
* **Netzwerk-Policies:** 
  + Anwenden von Kubernetes-Netzwerk-Policies, um den Datenverkehr zwischen Namespaces zu beschränken und sicherzustellen, dass nur notwendige Service kommunizieren könnenapiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: allow-keycloak

namespace: keycloak

spec:

podSelector:

matchLabels:

app: keycloak

ingress:

- from:

- namespaceSelector:

matchLabels:

name: backend-services

ports:

- protocol: TCP

port: 8080

* + . Netzwerk-Policies helfen, den Datenverkehr innerhalb des Kubernetes-Clusters zu steuern und einzuschränken, welcher Pod mit welchem kommunizieren darf.
  + Netzwerk-Policy für den Zugriff auf Keycloak nur vom API-Gateway erlaubt.
  + Netzwerk-Policy für den Zugriff auf MS SQL Server nur von Keycloak erlaubt:
* **Firewall**
  + Verwende die Firewall, um den Netzwerkverkehr zwischen verschiedenen Bereichen des internen Netzwerks zu kontrollieren.
  + Konfiguriere die Firewall so, dass nur der notwendige Datenverkehr zu den Kubernetes-Worker-Knoten zugelassen wird.
* **Monitoring und Auditing**
  + Erfasse und speichere Logs von Keycloak, API-Gateway und anderen kritischen Services.