Examen Administration réseaux

et services

**Réalisé par :**

**Amine Akkari**

**Wael Abdallah**

* **Introduction :**

Ce projet consiste en la mise en œuvre de trois microservices interconnectés, utilisant Dapr pour la communication via l’invocation des services et la persistance des données via les bindings et les state stores. Les microservices gèrent différents aspects d'une application client : gestion des clients, des commandes et des livraisons. L'objectif est de créer une architecture résiliente, modulable et facilement maintenable.

* **Structure du Projet :**
  1. **Microservices :**

1. Customer Service : Gère les opérations relatives aux clients (ajout, modification, suppression, consultation).

|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| POST | /customer | Ajoute un nouveau client |
| GET | /customer/get/{id] | Récupère les détails d'un client |
| PUT | /customer/update/{id} | Met à jour les détails d’un client |
| DELETE | /customer/delete/{id} | Supprime un client |

1. Order Service : Gère la gestion des commandes. Il crée, récupère et valide les commandes passées par les clients pour des produits spécifiques.

|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| POST | /order | Place une nouvelle commande |
| GET | /order/get/{id] | Récupère les détails d'une commande par ID |
| PUT | /order/update/{id} | Met à jour les détails d’une commande |
| DELETE | /order/delete/{id} | Supprime une commande |

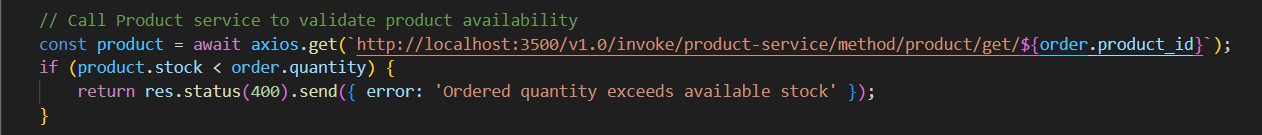
1. Product Service : Assure le suivi des livraisons, de l'enregistrement à la complétion.

|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| POST | /product | Ajoute un nouveau produit au stock |
| GET | /product/get/{id] | Récupère les détails d'un produit par ID |
| PUT | /product/update/{id} | Met à jour les niveaux de stock des produits |
| DELETE | /product/delete/{id} | Supprime une commande |

* 1. **Communication Entre Services :**
     + **Order Service ↔ Customer Service :** Avant de créer une commande, le Order Service valide le customer\_id en invoquant le Customer Service.
     + **Order Service ↔ Product Service :** Après avoir créé une commande, le Order Service invoque le Product Service pour mettre à jour les niveaux de stock.
     + **Product Service ↔ State Store :** Le Product Service pourrait utiliser l'API de gestion d'état de Dapr pour persister les niveaux de stock pour une résilience accrue.
  2. **Architecture :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Microservice | Langage | Framework/Librairie | Base de données | Dapr Component |
| Customer Service | **Python** | **Flask** | **PostgreSQL** | **Binding** |
| |  |  | | --- | --- | | **Order Service** |  | | **Node.js** | **Express, Mongoose** | **MongoDB** | **State Store** |
| Product Service | **Java** | **Quarkus** | **MySQL** | **Binding** |

* + - Chaque microservice est déployé dans un conteneur Docker.
    - Les communications interservices utilisent les API HTTP via Dapr.
    - Kubernetes orchestre les microservices, gère leur déploiement et la tolérance aux pannes.
  1. **Choix Technologiques :**
* **Dapr Components :**
  1. **Service Invocation :** Dapr permet une invocation directe des microservices via HTTP ou gRPC. Par exemple, **Order-Service** appelle une méthodede **Product-Service** pour vérifier la disponibilité de la quantité demandée via l’URL spécifique au service invocation component de Dapr à travers **Dapr side-car** :



* 1. **Bindings :** Les bindings permettent d'interagir avec des ressources externes comme des bases de données. Exemple pour **PostgreSQL** :Une image contenant texte, capture d’écran, Police

     Description générée automatiquement
  2. Une image contenant texte, capture d’écran, Police

     Description générée automatiquement**State Stores :** Pour garantir une résilience accrue, **Order** **Service** utilise un **state store Dapr** pour persister les commandes :
* **Activités d’Administration :**
  1. **Supervision :**
     + **Dapr Dashboard :**

Une image contenant capture d’écran, texte, Logiciel multimédia, logiciel

Description générée automatiquement

Une image contenant texte, Logiciel multimédia, capture d’écran

Description générée automatiquement

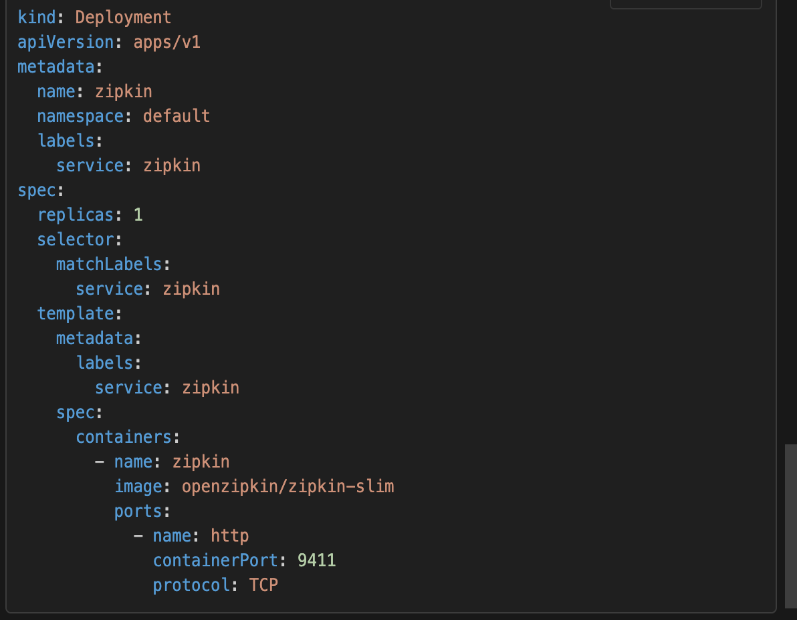
* + - **Prometheus :** La configuration Prometheus est définie dans **prometheus-deployment.yml**. Elle représente la récupération des métriques à partir de **Dapr sidecars** des 3 microservices :

****

* + - **Grafana :** La configuration de Grafana est définie dans grafana-deployment.yml. Il est exécuté sur le port 3000 :

****

* + - **Zipkin :** est un système de traçage distribué. Il aide à collecter les données de chronométrage nécessaires pour résoudre les problèmes de latence dans les architectures de services. La configuration de Zipkin est définie dans le fichier zipkin.yml. Zipkin est configuré pour fonctionner sur le port 9411 :



**2. Sécurité :**

* + - **mTLS :** Sécuriser les communications entre les services de Dapr en activant l’annotation mTLS géré implicitement par Dapr comme suit :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* + - **RBAC :** Configurer RBAC (Role-Based Control) dans Kubernetes pour limiter les accès en définissant des rôles pour chaque service et/ou cluster, par exemple :

Une image contenant texte, capture d’écran

Description générée automatiquement

* + - **WAF :** pare-feu applicatif (WAF) avec Istio pour protéger les services contre les attaques. Ici par exemple, on a configuré un proxy qui redirige toutes les requêtes vers le pare-feu pour les filtrer, particulièrement les requêtes de suppression avec la méthode DELETE :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* + - **K8s Audit :** Activer l’audit Kubernetes pour suivre les accès au cluster :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* **Conclusion :**

Ce projet a abouti à la création d'une architecture de microservices résiliente, utilisant **Dapr** pour faciliter la communication entre les services. En plus de la gestion des clients, des commandes et des produits, **deux activités d'administration** clés ont été mises en place pour garantir la fiabilité et la sécurité du système :

1. **Supervision :** Les outils **Prometheus/Grafana**, **Zipkin** et **Dapr Dashboard** permettent de surveiller les performances de tracer les appels entre services et de gérer les ressources du cluster.
2. **Sécurité :** L'authentification et l'autorisation sont assurées par **mTLS**, tandis que des pratiques de protection réseau avec **Istio** et **K8s Audit** renforcent la sécurité des services.