## Exercice : Intégration et Sécurisation d'une Architecture Événementielle avec Docker

## Étape 1 : Préparation et Mise en Place des Environnements

**Objectif :** Configurer les environnements nécessaires pour l'exercice en utilisant Docker.

**Tâche 1.1 :** Installer Docker Desktop sur vos machines locales.

## Étape 2 : Introduction et Normalisation des Données

**Objectif :** Assurer une base solide de compréhension des concepts de normalisation et de gestion des données.

**Tâche 2.1 :** Créer un document expliquant l'importance de la normalisation des données dans les échanges entre systèmes.

- **Points majeurs à inclure :**
  - Cohérence des données
  - Réduction de la redondance
  - Amélioration de la qualité des données
  - Facilité de maintenance et évolutivité

**Tâche 2.2 :** Créer et normaliser un jeu de données à partir de zéro. Documenter le processus utilisé.

1. **Étape 1 : Création du Jeu de Données**
   - Développez un jeu de données fictives représentant une base de données clients pour un scénario de commerce électronique. Le jeu de données doit inclure les informations suivantes pour chaque client :
     - Identifiant unique du client
     - Prénom et nom
     - Adresse email
     - Numéro de téléphone
     - Adresse postale (rue, ville, code postal, pays)
     - Date de création du compte
     - Historique des achats (au moins 5 transactions par client)
2. **Étape 2 : Normalisation du Jeu de Données**
   - Appliquez les principes de normalisation des bases de données pour structurer les données en différentes tables. Les tables doivent inclure :
     - **Table Clients** : Identifiant unique, prénom, nom, email, téléphone, date de création du compte.
     - **Table Adresses** : Identifiant unique, identifiant client (clé étrangère), rue, ville, code postal, pays.
     - **Table Achats** : Identifiant unique, identifiant client (clé étrangère), date d'achat, montant de l'achat, produit acheté (peut être une liste séparée si plusieurs produits).
     - **Table Produits** : Identifiant unique, nom du produit, catégorie, prix.
3. **Étape 3 : Documentation du Processus**

    - Documentez chaque étape du processus de normalisation, en expliquant les choix de conception et les avantages obtenus :
        - Décomposition initiale des données brutes en tables logiques.
        - Justification de la structure choisie pour chaque table.
        - Avantages obtenus en termes de cohérence, réduction de la redondance, et facilité de maintenance.

## Étape 3 : Développement des Microservices

**Objectif :** Développer une série de microservices pour une application de commerce électronique.

**Diagramme d'architecture :** Inclure un schéma illustrant l'interaction entre les microservices.

**Microservice 1 : Gestion des Utilisateurs**

- **Description :** Ce service gère les utilisateurs, y compris l'inscription, la connexion et la gestion des profils.
- **Technologie :** Node.js avec Express.
- **Endpoints :**
    - `POST /api/users/register` : Inscription d'un nouvel utilisateur.
    - `POST /api/users/login` : Connexion d'un utilisateur.
    - `GET /api/users/:id` : Récupérer les informations du profil utilisateur.
- **Base de données :** PostgreSQL

**Microservice 2 : Gestion des Produits**

- **Description :** Ce service gère les produits, y compris l'ajout, la mise à jour et la suppression de produits.
- **Technologie :** Python avec Flask.
- **Endpoints :**
    - `POST /api/products` : Ajouter un nouveau produit.
    - `GET /api/products` : Récupérer la liste des produits.
    - `PUT /api/products/:id` : Mettre à jour un produit.
    - `DELETE /api/products/:id` : Supprimer un produit.
- **Base de données :** PostgreSQL

**Microservice 3 : Gestion des Commandes**

- **Description :** Ce service gère les commandes, y compris la création, la mise à jour et l'état des commandes.
- **Technologie :** Node.js avec Express.
- **Endpoints :**
    - `POST /api/orders` : Créer une nouvelle commande.
    - `GET /api/orders/:id` : Récupérer les détails d'une commande.
    - `PUT /api/orders/:id` : Mettre à jour l'état d'une commande.
- **Base de données :** PostgreSQL

**Instructions :**

- Créez les microservices en suivant les descriptions ci-dessus.

- Testez les microservices de manière indépendante pour vérifier leur bon fonctionnement.
- Assurez la scalabilité des microservices en configurant Docker Compose.

## Étape 4 : Intégration et Orchestration des Microservices

**Objectif :** Utiliser Docker Compose pour orchestrer les microservices.

**Instructions :**

- Créez un fichier `Dockerfile` pour chaque microservice afin de les dockeriser.
- Créez un fichier `docker-compose.yml` pour orchestrer les microservices.

**Docker Compose File Exemple**

```
version: '3'
services:
  user-service:
    build: ./user-service
    ports:
      - "3001:3001"

  product-service:
    build: ./product-service
    ports:
      - "5000:5000"

  order-service:
    build: ./order-service
    ports:
      - "3002:3002"
```

## Étape 5 : Mise en Œuvre d'une Architecture Événementielle

**Objectif :** Utiliser Apache Kafka pour la communication entre microservices.

**Instructions :**

- Installez et configurez Apache Kafka en utilisant Docker Compose.
- Implémentez la production et la consommation d'événements entre les microservices en utilisant Kafka.

**Exemple : Configuration Apache Kafka avec Docker Compose**

```
version: '3'
services:
  kafka:
    image: bitnami/kafka:latest
    ports:
      - "9092:9092"
    environment:
      KAFKA_BROKER_ID: 1
```

```
      KAFKA_LISTENERS: PLAINTEXT://:9092
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181

  zookeeper:
    image: bitnami/zookeeper:latest
    ports:
      - "2181:2181"
```

## Étape 6 : Sécurisation des Services

**Objectif :** Implémenter des mécanismes de sécurité pour les services.

**Sécurité des Microservices avec JWT (JSON Web Tokens)**

- **Tâche 6.1 :** Implémenter l'authentification et l'autorisation à l'aide de JWT.

**Instructions :**

- Ajoutez une fonctionnalité d'authentification à vos services utilisateurs pour générer des tokens JWT lors de la connexion.
- Protégez les endpoints des autres microservices en validant les tokens JWT.

**Introduction API Gateway / Proxy**

- Implémentez un API Gateway pour gérer les requêtes entrantes et appliquer les politiques de sécurité.

## Étape 7 : Mise en Place de l'Application Frontend avec React

**Objectif :** Développer une interface utilisateur pour l'application de commerce électronique.

**Instructions :**

- Créez une application React pour interagir avec les microservices.
- Utilisez les endpoints suivants :
  - `GET /api/products` : Récupérer la liste des produits pour afficher sur la page principale.
  - `POST /api/orders` : Créer une commande à partir du panier.

## Étape 8 : Étude de Cas Pratique

**Objectif :** Appliquer l'ensemble des concepts dans un cas d'usage réel.

**Instructions :**

- Développez une application complète pour un scénario de commerce électronique, intégrant tous les services développés.
- Documentez chaque étape du développement, les défis rencontrés, et les solutions mises en place.
- Présentez le projet final avec une démonstration fonctionnelle.

## Évaluation et Livrables

**Documentation :** Chaque étape devra être accompagnée d'une documentation détaillant les décisions prises, les configurations effectuées, et les résultats obtenus.

**Code Source :** Le code de tous les services développés devra être soumis dans un repository Git, avec des instructions pour le déploiement.

**Présentation Finale :** Une présentation résumant le projet, les difficultés rencontrées, les solutions apportées, et une démonstration fonctionnelle de l'application.