Architecture du projet Kafka

## **Introduction**

Ce rapport présente l'architecture technique du projet, en mettant en évidence la structure des différents composants et leur interaction. L'objectif de cette architecture est de garantir une gestion efficace des abonnements, une diffusion ciblée des messages et une tolérance aux pannes tout en assurant la sécurité des données utilisateur via un système d'authentification basé sur une adresse email et un mot de passe.

### **Architecture générale**

L'architecture est composée des éléments suivants :

**Frontend** : Interface utilisateur permettant aux utilisateurs de s'inscrire, se connecter, gérer leurs abonnements et consulter leurs alertes et de publier des messages.

**Backend** : Serveur applicatif implémenté en Java qui gère la logique métier, la gestion des abonnements, et la communication avec Apache Kafka.

**Apache Kafka** : Composant de message-broker assurant la gestion des topics et des messages entre publishers et subscribers.

**Apache Zookeeper** : Coordination et gestion des métadonnées des clusters Kafka pour garantir la tolérance aux pannes.

**Base de données** : Système Mysql contenant les tables des utilisateurs, des abonnements et des topics.

**Docker** : Utilisé pour la conteneurisation et l'orchestration des composants du système (Kafka, Zookeeper, backend, etc.).

### **Description des composants principaux**

* **Base de données (Mysql)**

La base de données structure les données comme suit :

***users*** : Gère les informations utilisateur (email, nom, prénom, mot de passe, rôle).

***subscription*** : Associe un utilisateur à un ou plusieurs topics auxquels il est abonné.

***topics*** : Liste des thèmes disponibles pour les abonnements.

**mailplaning** : reçoit les plannings d’envoi de message des utilisateurs

* **Backend**

***Gestion des utilisateurs*** :

Authentification des utilisateurs via email et mot de passe.

Sécurité garantie par le stockage des mots de passe hachés (non inclus dans le code actuel).

***Gestion des abonnements*** :

Les utilisateurs peuvent s’abonner à des topics ou en créer de nouveaux. Les abonnements sont stockés dans la table subscription.

***Interaction avec Kafka*** :

Les publishers publient des messages dans les topics correspondants.

Les subscribers reçoivent les messages correspondant à leurs abonnements.

* **Apache Kafka et Zookeeper**

***Apache Kafka*** : gère les topics et les messages, garantit une distribution efficace des messages entre les producteurs et les consommateurs.

***Zookeeper*** :permet la synchronisation des nœuds Kafka, assure une haute disponibilité en cas de panne d'un broker.

* **Docker**

***Rôle de Docker*** :sert à la conteneurisation des composants pour assurer leur portabilité et leur isolation, à l’orchestration des instances Kafka (brokers et contrôleurs) et Zookeeper pour simplifier leur déploiement.

* **Conteneurs actifs**

***Kafka*** : plusieurs brokers Kafka (kafka1, kafka2, kafka3) sont déployés, chacun étant accessible via un port distinct.

***Zookeeper*** : trois contrôleurs (controller-1, controller-2, controller-3) pour gérer la coordination des clusters Kafka.

***Avantages*** :

* Déploiement rapide et standardisé.
* Surveillance centralisée des ressources via Docker Desktop.
* **Interface utilisateur (Frontend)**

***Fonctionnalités principales*** :

* Inscription et connexion des utilisateurs.
* Publication de messages par les publishers.
* Gestion des abonnements et consultation des alertes.
* Personnalisation des réglages de notification (heure et fréquence).

### **Interaction avec l’application**

***Cas d'utilisation principal*** : Publication et réception de messages

**Inscription et connexion** :Un utilisateur s'inscrit ou se connecte via l'interface utilisateur. Les informations sont validées et stockées dans la table users.

***Abonnement à un topic*** :Un utilisateur choisit un topic existant ou en crée un nouveau. L'abonnement est enregistré dans la table subscription.

***Publication d'un message*** :Un publisher publie un message via l'interface. Le message est envoyé à Kafka et stocké dans le topic correspondant.

***Réception par les subscribers*** :Les subscribers reçoivent les messages correspondant à leurs abonnements en temps réel ou selon leur planification.

### **Technologies utilisées**

* **Docker**

Utilisé pour conteneuriser les composants tels que Kafka, Zookeeper, et le backend.

Simplifie le déploiement et la gestion des différentes instances.

* **Apache Kafka**

Utilisé pour sa capacité à gérer un grand volume de messages avec une faible latence.

Scalabilité horizontale pour supporter un nombre croissant d’utilisateurs.

* **Apache Zookeeper**

Gestion des clusters Kafka et coordination des brokers pour garantir une tolérance aux pannes.

* **Mysql**

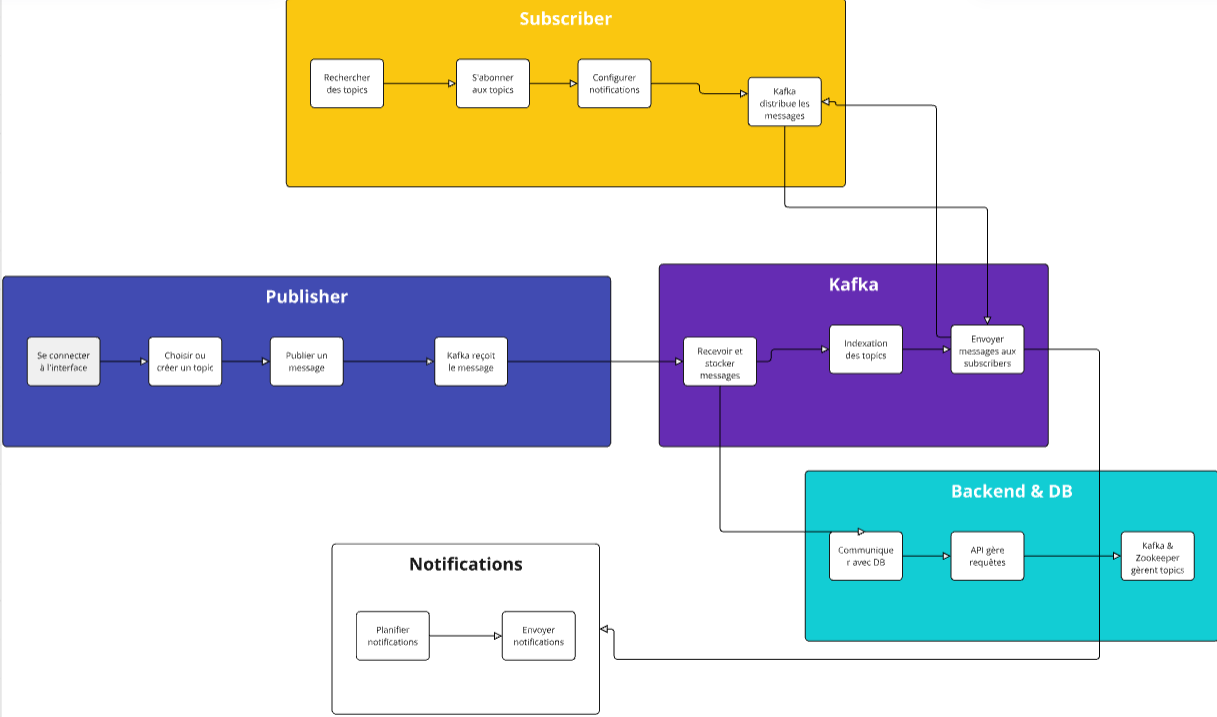
Stockage des données utilisateurs, des abonnements et des topics. Intégrité assurée par les contraintes de clés étrangères.

* **Java**

Langage principal pour l'implémentation du backend.

Compatible avec les API Kafka pour une intégration efficace.

### **Graphe des flux**



### **Aspects de performance et de scalabilité**

***Performance*** : Kafka garantit une diffusion rapide des messages avec une faible latence.

***Scalabilité*** : Possibilité d'ajouter des brokers Kafka et des partitions pour répartir la charge.

***Tolérance aux pannes*** : Zookeeper et Kafka assurent la continuité du service même en cas de défaillance d'un composant.

* **Sécurité**

***Authentification*** : Basée sur l’email et le mot de passe.