Congo Digital Services

Architecture Technique : ASP.NET Core + Kafka + Docker + CQRS + Event Sourcing

1. Introduction

Cette architecture repose sur ASP.NET Core avec une approche modulaire basée sur les principes de la Clean Architecture et de l'Hexagonal Architecture. Elle intègre Docker pour l'orchestration des services, Kafka pour la gestion asynchrone des événements, et le pattern CQRS avec Event Sourcing pour assurer la scalabilité, la maintenabilité et le respect des principes SOLID.

2. Technologies Utilisées

- Framework principal: ASP.NET Core (C#)
- Architecture: Clean Architecture / Hexagonal Architecture
- Pattern: CQRS (Séparation lecture/écriture) avec MediatR
- Gestion des événements: Event Sourcing + Kafka
- Authentification: JWT (JSON Web Token)
- ORM: Entity Framework Core
- Documentation API: Swagger / Scalar
- Conteneurisation: Docker & Docker Compose
- Base de données: MicroSoft Sql server/ PostgreSQL
- Cache / Queue: Redis
- Monitoring: Prometheus + Grafana + Seq / ELK Stack
- Messaging: Kafka + Zookeeper + Schema Registry
- Supervision et administration: Kafka UI / Confluent Control Center

3. Rôle de chaque composant

• ASP.NET Core API : cœur métier de l'application(ARPCE Homologie) utilisant MediatR pour gérer les commandes et événements.
• Kafka : bus d'événements asynchrone qui va permettre de publier et consommer des événements entre backends.
• Docker : outil d'orchestration des services pour garantir la portabilité et l'isolation des environnements.
• MicroSoft Sql Server / PostgreSQL : stockage principal et gestion de l'Event Store.
• Redis : cache distribué et stockage temporaire pour améliorer la performance.
• Prometheus & Grafana : outils de supervision et visualisation des performances.
• Seq / ELK : centralisation et analyse des logs applicatifs.
• Swagger / Scalar : documentation interactive de l'API (ARPCE Homologie).
• JWT : système d'authentification sécurisé par jetons.

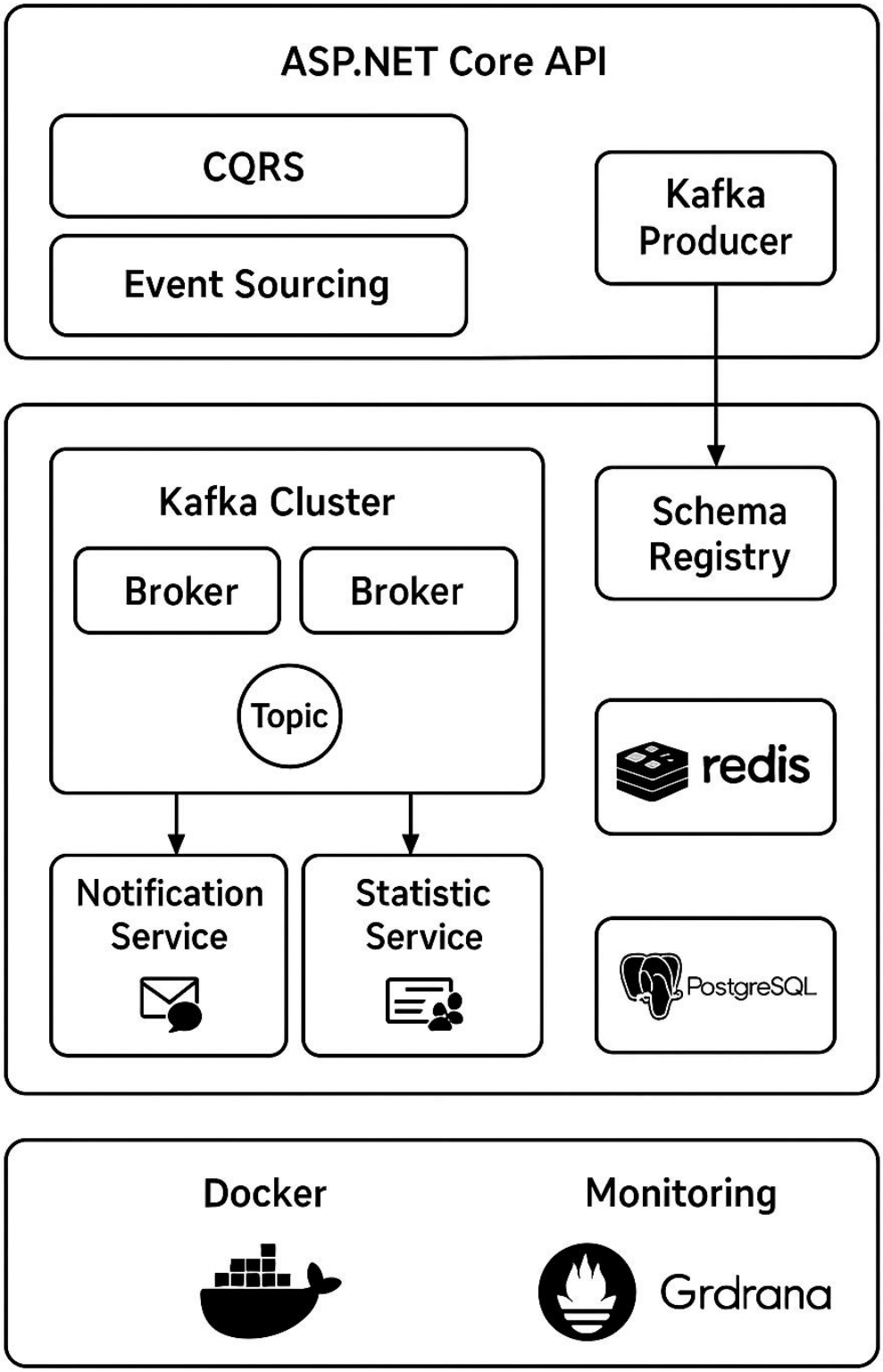
4. Flux de données avec Kafka et CQRS

1. L'utilisateur envoie une requête HTTP vers l'API ASP.NET Core.
2. Une commande MediatR est exécutée, créant un événement de domaine.
3. L'événement est stocké dans l'Event Store et publié dans un topic Kafka.
4. Les consommateurs Kafka (NotificationService, StatistiqueService, ...) reçoivent l'événement.
5. Les vues de lecture (Read Models) sont mises à jour.
6. Les conteneurs Docker assurent l'exécution et la communication entre les services.

5. Avantages de l'architecture

- Respect des principes SOLID et du découplage fort.
- Scalabilité horizontale via Docker et Kafka.
- Gestion asynchrone des événements métier.
- Séparation claire entre les responsabilités (CQRS).
- Haute résilience et tolérance aux pannes.
- Observabilité complète via Grafana et Prometheus.
- Déploiement portable et reproductible grâce à Docker.

6. Shéma visuel


ASP.NET Core API
CQRS
Event Sourcing
Kafka Producer
Kafka Cluster
Broker
Broker
Topic
Schema Registry
redis
Notification Service
Statistic Service
PostgreSQL
Docker
Monitoring
Grdrana

Congo Digital Services