Déploiement Kafka en Haute Disponibilité

Mode KRaft avec Docker Compose

01	INTRODUCTION	Contexte,problematique, Objectif
2	RAPPELS SUR KAFKA	Les mots clés
03	MÉCANISMES DE HAUTE DISPONIBILITÉ	Docker compose ,Réplication des partitions,ISR (In-Sync Replicas)? Quorum des controllers,Election du leader
04	DÉMO	Déploiement en haute disponibilité et Scénarios de tests de résilience
05	CONCLUSION	Rappels des bonnes pratiques

Introduction

01

CONTEXTE

Pourquoi la haute disponibilité est essentielle dans les systèmes distribués.

2

EXEMPLE

Un nœud peut tomber en panne

03

PROBLÉMATIQUE

Tolérance aux pannes et continuité de service.

04

EXEMPLE

Assurer la continuité du service malgré les pannes.

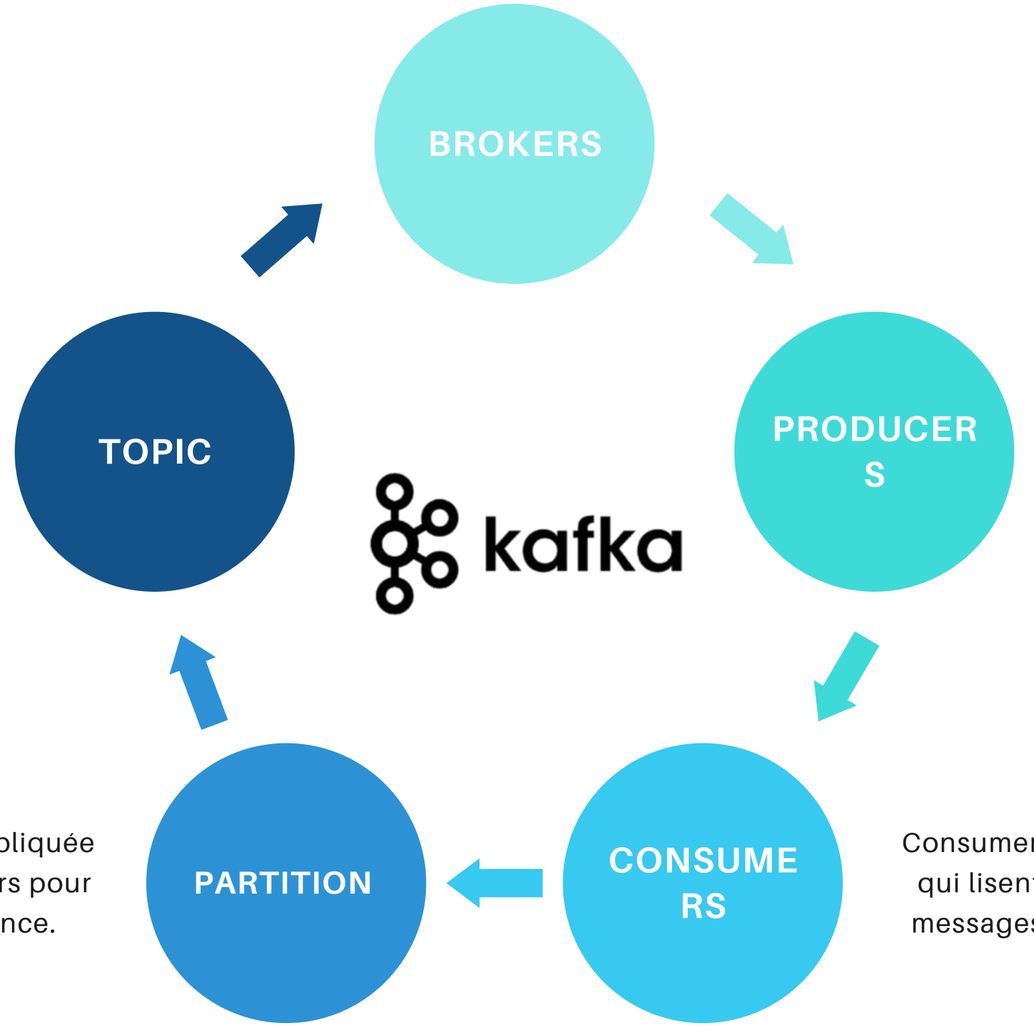
05

OBJECTIF

Montrer comment Kafka assure la haute disponibilité.

Rappels sur Kafka

Brokers : serveurs qui stockent et distribuent les données.



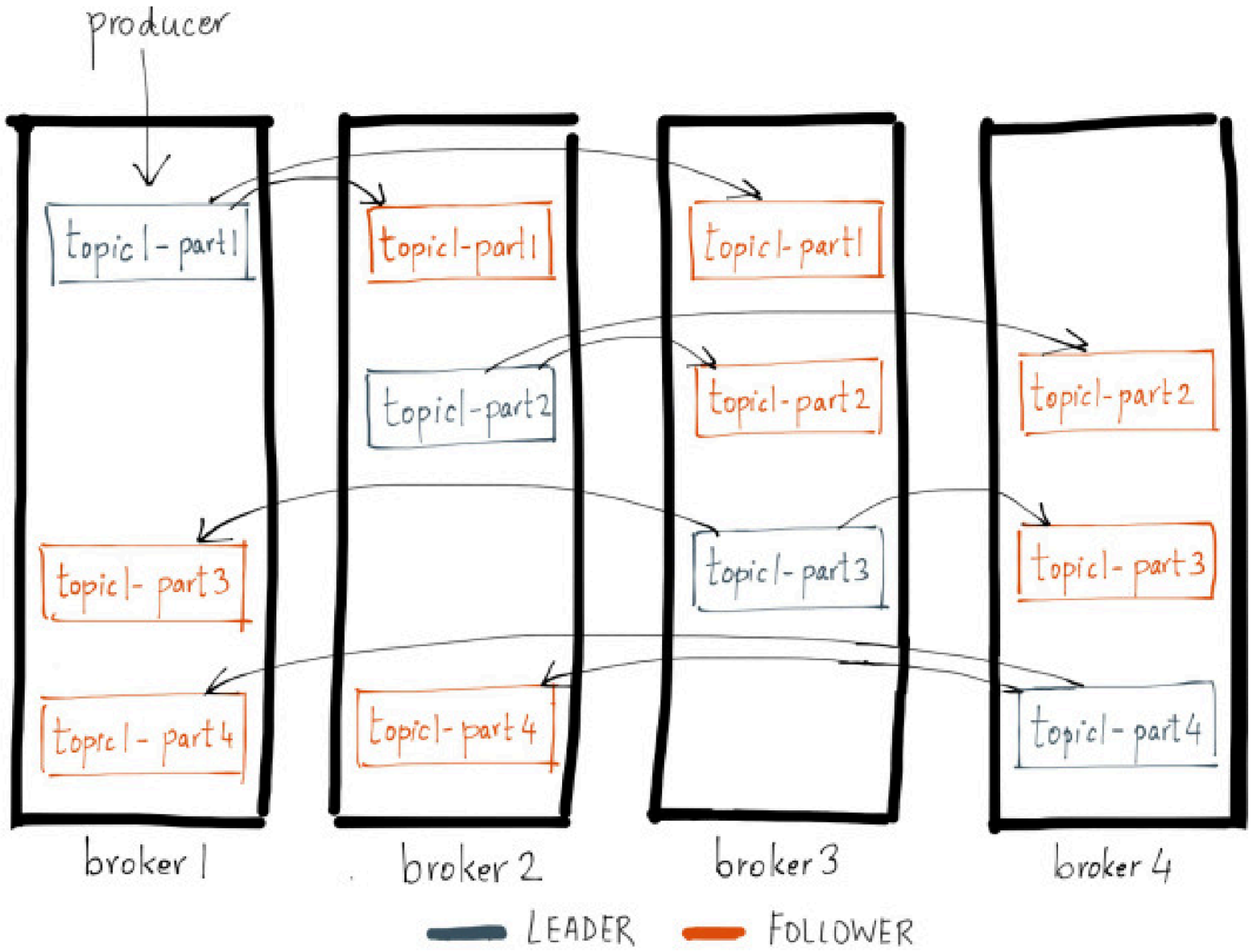
Producers : applications ou services qui publient des messages dans Kafka.

Consumers : applications qui lisent et traitent les messages depuis Kafka.

Une partition est répliquée sur plusieurs brokers pour garantir la résilience.

Un topic est une catégorie où les messages sont publiés (ex. : userlogin).

Mécanismes de haute disponibilité



Mécanismes de haute disponibilité

01

RÉPLICATION DES PARTITIONS

Chaque partition d'un topic peut être répliquée sur plusieurs brokers.
Parmi les réplicas, l'un est désigné comme leader, les autres sont des followers.

2

ISR (IN-SYNC REPLICAS)

Les ISR représentent l'ensemble des réplicas synchronisés avec le leader.
Tant que le leader et au moins un follower sont dans l'ISR, les données sont considérées comme durables.

03

ÉLECTION DU LEADER

En cas de panne du broker leader :
Le controller du cluster déclenche une élection automatique.
Un follower dans l'ISR devient le nouveau leader.

04

QUORUM DES CONTROLLERS

Dans le mode KRaft, les controllers utilisent un protocole de consensus (Raft) pour gérer les métadonnées.

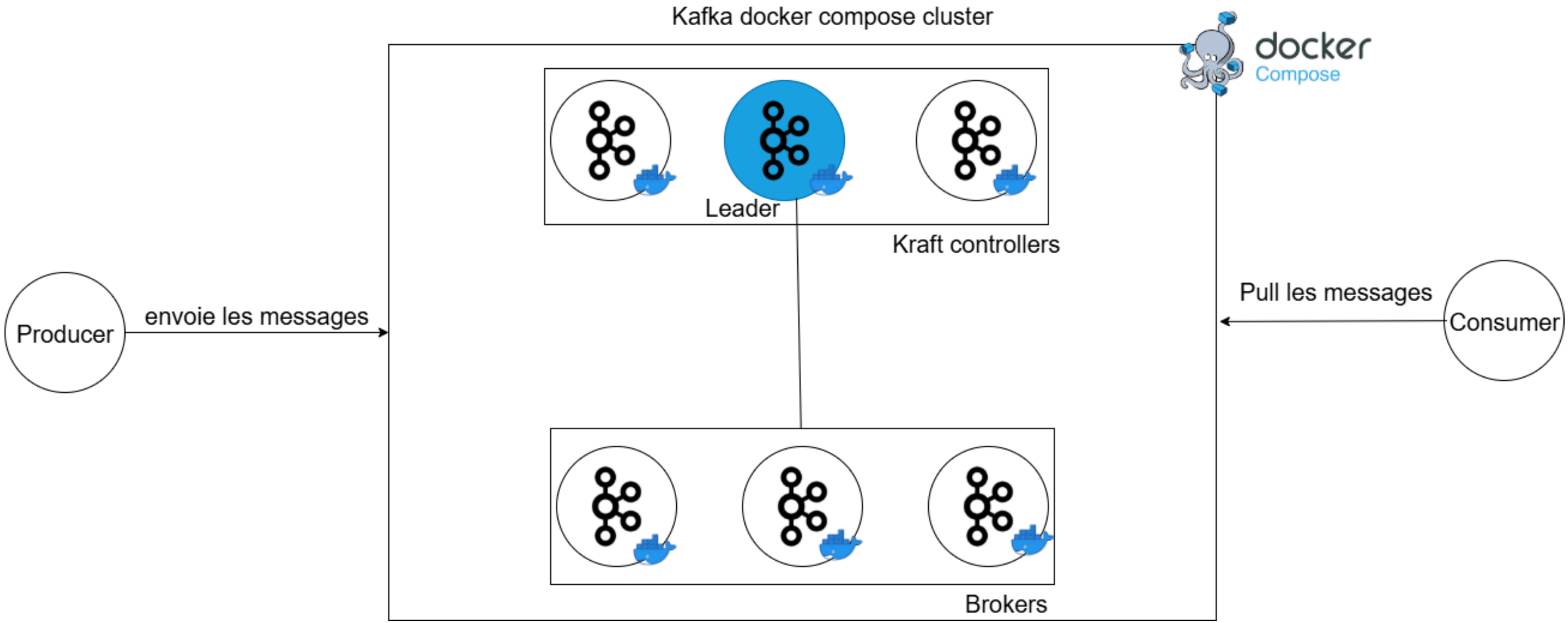
05

CONCLUSION

Pour que le cluster continue de fonctionner, il faut que la majorité des controllers soit disponible (ex. : sur 3 controllers, 2 doivent être actifs).

Mécanismes de haute disponibilité

Diagramme Architecture



Mécanismes de haute disponibilité

https://docs.docker.com/manuals

DOCKER	DOCKER	DOCKER	DOCKER	DOCKER
Docker est une plateforme permettant de créer, déployer et exécuter des applications dans des conteneurs	Les conteneurs sont légers, portables et rapides	Ils assurent la cohérence entre environnements de développement, test et production.	Portabilité entre différents environnements (Windows, Linux, Cloud).	Gain de temps et de ressources par rapport aux machines virtuelles.

Mécanismes de haute disponibilité

https://docs.docker.com/manuals

DOCKER COMPOSE	DOCKER COMPOSE	DOCKER COMPOSE	DOCKER COMPOSE	DOCKER COMPOSE
Outil permettant de définir et gérer des applications multi-conteneurs	Utilise un fichier YAML pour configurer les services, réseaux et volumes.	Facilite le déploiement et la mise en place d'environnements complets.	Permet de lancer tous les services avec une seule commande : docker-compose up.	Facilite le partage et la reproduction d'environnements de travail.

Mécanismes de haute disponibilité

Exemple de docker compose

```yaml
services:
  kafka:
    image: apache/kafka:4.1.0
    container_name: broker-1
    ports:
      - "9092:9092"
      - "9093:9093"
    environment:
      KAFKA_KRAFT_MODE: "true"  # Active le mode KRaft.
      KAFKA_PROCESS_ROLES: controller,broker  # Kafka agit à la fois comme contrôleur et broker.
      KAFKA_NODE_ID: 1  # ID unique pour l'instance Kafka.
      KAFKA_CONTROLLER_QUORUM_VOTERS: "1@localhost:9093"  # Quorum du contrôleur.
      KAFKA_LISTENERS: PLAINTEXT://localhost:9092,CONTROLLER://localhost:9093
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,CONTROLLER:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
      KAFKA_LOG_DIRS: /var/lib/kafka/data  # Emplacement de stockage des journaux (logs).
      KAFKA_AUTO_CREATE_TOPICS_ENABLE: "true"  # Active la création automatique des topics.
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1  # Réplication simple (un seul replica, pour simplifier).
      KAFKA_LOG_RETENTION_HOURS: 168  # Période de rétention des journaux (7 jours).
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0  # Aucun délai pour le rééquilibrage initial.
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ./data:/var/lib/kafka/data  # Monte les journaux (logs) dans un dossier local.
```

Mécanismes de haute disponibilité

Exemple plus détaillé

```
services: # Définition des services Docker (chaque service est un conteneur)
  kafka: # Nom du service (ici, Kafka)
    image: apache/kafka:4.1.0 # Image Docker officielle d'Apache Kafka en version 4.1.0
    container_name: broker-1 # Nom explicite du conteneur (facilite la gestion)
    ports: # Mappage des ports entre la machine hôte et le conteneur
      - "9092:9092" # Port principal pour la communication client Kafka
      - "9093:9093" # Port utilisé pour la communication du contrôleur (KRaft)
    environment: # Variables d'environnement pour configurer Kafka
      KAFKA_KRAFT_MODE: "true"  # Active le mode KRaft (remplace ZooKeeper)
      KAFKA_PROCESS_ROLES: controller,broker  # Kafka joue le rôle de contrôleur et broker en même temps
      KAFKA_NODE_ID: 1  # Identifiant unique du nœud dans le cluster Kafka
      KAFKA_CONTROLLER_QUORUM_VOTERS: "1@localhost:9093"  # Définit les membres du quorum contrôleur (ici un seul contrôleur)
      KAFKA_LISTENERS: PLAINTEXT://localhost:9092,CONTROLLER://localhost:9093 # Définit les adresses et protocoles des listeners
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,CONTROLLER:PLAINTEXT # Mappe les protocoles de sécurité aux listeners
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT # Définit le listener utilisé pour la communication interne entre brokers
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER # Définit le listener utilisé par le contrôleur
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092 # Adresse annoncée aux clients Kafka
      KAFKA_LOG_DIRS: /var/lib/kafka/data  # Dossier où Kafka stocke ses journaux (logs) et données
      KAFKA_AUTO_CREATE_TOPICS_ENABLE: "true"  # Autorise la création automatique de topics lors de la première utilisation
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1  # Réplication minimale (un seul replica, car un seul broker)
      KAFKA_LOG_RETENTION_HOURS: 168  # Durée de rétention des logs en heures (168h = 7 jours)
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0  # Aucun délai lors du premier rééquilibrage des groupes de consommateurs
    volumes: # Volumes pour persister les données et interagir avec Docker
      - /var/run/docker.sock:/var/run/docker.sock # Monte le socket Docker (souvent utilisé pour monitoring ou outils de gestion)
      - ./data:/var/lib/kafka/data  # Persistance locale : stocke les données Kafka dans ./data sur l'hôte
```

DEMO

Scénarios

https://github.com/ngatcheu/kafka_docker_compose

01

RÉPLICATION DES DONNÉES

Vérifier que les topics sont répliqués entre les brokers.

02

RÉSILIENCE DES BROKERS

Stopper un broker → le cluster reste disponible.

03

TOLÉRANCE AUX PANNES DES CONTROLLERS

Stopper un controller leader → un autre est élu automatiquement.

Conclusion