Description du projet sur le Déploiement de Kafka sur un Cluster Kubernetes pour la Programmation Distribuée

Dans le dossier 01-Presentation + Quickstart du projet, vous trouverez les fichiers suivants :

- Quickstart Apache Kafka (Local Kafka)
- Quickstart Strimzi Kafka on Kubernetes (Strimzi k8s)
- Marche à suivre set up complet (Full Setup)

Introduction

Ce rapport décrit la mise en place d'un cluster Kubernetes (k8s) intégrant Apache Kafka, Kafka Connect et Kafka Bridge pour mettre en œuvre une solution de programmation distribuée. Il explique comment connecter une source de données (connecteur Telegram) à Kafka Connect pour publier sur un topic Kafka, et comment utiliser un sink pour exporter les données vers AWS SQS. Ce projet montre comment les concepts de calcul distribué peuvent être appliqués pour créer une architecture résiliente, évolutive et performante.

Objectifs du Projet

1. **Déployer un cluster Kubernetes** pour orchestrer les conteneurs nécessaires.
2. **Installer et configurer Apache Kafka** sur Kubernetes pour le traitement des flux de données en temps réel.
3. **Utiliser Kafka Connect** pour intégrer une source de données Telegram et publier des messages sur un topic Kafka.
4. **Configurer un sink** pour exporter les données des topics Kafka vers AWS SQS.
5. **Utiliser Kafka Bridge** pour permettre la communication via des API REST.

Architecture du Système Distribué

Composants Principaux

1. **Kubernetes (k8s)** : Orchestration de conteneurs pour la gestion automatisée du déploiement, du scaling et des opérations.
2. **Apache Kafka** : Plateforme de streaming distribuée pour la publication, le stockage et la consommation de flux de données en temps réel.
3. **Kafka Connect** : Cluster pour intégrer Kafka avec des systèmes externes (sources et sinks).

4. **Kafka Bridge** : Permet aux applications de communiquer avec Kafka via des API REST.

5. **Sources de Données** : Connecteur Telegram source pour publier des messages sur Kafka.

Schéma d'Architecture

Mise en Œuvre

Déploiement de Kubernetes

1. **Installation de Kubernetes** : Utiliser Minikube, Kubeadm ou un cluster Kubernetes managé (comme AWS EKS, Google GKE, ou Azure AKS).

2. **Configuration du Cluster** : Configurer les nœuds maîtres et les nœuds de travail pour exécuter les charges de travail de conteneurs.

Installation et Configuration de Kafka

1. **Déploiement de Kafka** : Utiliser des Helm charts ou des opérateurs Kubernetes pour déployer Kafka sur le cluster.

2. **Configuration de Kafka** : Configurer les brokers Kafka et Zookeeper pour gérer le stockage des données et la coordination des services.

Mise en Place de Kafka Connect

1. **Déploiement de Kafka Connect**

2. **Configuration du Connecteur Telegram Source** : Déployer et configurer le connecteur pour récupérer les messages de Telegram et les publier sur un topic Kafka.

3. **Configuration du Sink AWS SQS**

    - **Déploiement du Sink** : Déployer et configurer le connecteur AWS SQS pour consommer les messages des topics Kafka et les stocker dans un bucket SQS.

Utilisation de Kafka Bridge

1. **Déploiement de Kafka Bridge**

2. **Configuration de Kafka Bridge** : Configurer les API REST pour permettre aux applications de communiquer avec Kafka.

Avantages de la Solution Distribuée

Scalabilité

La solution distribuée basée sur Kubernetes et Kafka permet d'ajouter facilement de nouveaux nœuds pour gérer une augmentation de la charge de travail sans interruption du service.

Disponibilité

La tolérance aux pannes intégrée de Kubernetes et Kafka assure que le système continue de fonctionner même en cas de défaillance de certains nœuds.

Consistance

Les données sont gérées de manière cohérente à travers tous les nœuds du cluster, garantissant l'intégrité des informations stockées.

Transparence

Les utilisateurs peuvent interagir avec le système comme s'il s'agissait d'un seul ordinateur, sans se soucier des détails de la configuration et de la gestion des machines individuelles.

Efficacité

La solution distribuée utilise de manière optimale les ressources matérielles disponibles, assurant des performances rapides et fiables pour le traitement des données.

Conclusion

Ce projet démontre comment les concepts de calcul distribué peuvent être appliqués pour créer une solution robuste et évolutive en utilisant Kubernetes, Kafka, et Kafka Connect. L'intégration de diverses sources de données et la gestion de l'exportation des données vers des services de cloud comme AWS SQS montrent la puissance et la flexibilité des architectures distribuées.

Éléments de la Programmation Distribuée

Éléments de la Programmation Distribuée

1. **Kubernetes (k8s)** :
    - Kubernetes orchestre les conteneurs dans un cluster, permettant la gestion automatisée du déploiement, du scaling et des opérations des applications conteneurisées. Chaque nœud du cluster Kubernetes peut être considéré comme un composant distribué.
2. **Apache Kafka** :
    - Kafka est une plateforme de streaming distribuée. Il permet la publication, le stockage et la consommation de flux de données en temps réel. Kafka fonctionne sur

un cluster de serveurs, offrant tolérance aux pannes, haute disponibilité et scalabilité.

3. **Kafka Connect** :

    - Kafka Connect est un framework pour intégrer Kafka avec des systèmes externes (sources et sinks). Il permet de connecter diverses sources de données et de publier les données sur Kafka, ainsi que de connecter des destinations pour consommer ces données. Cela ajoute un autre niveau de distribution en connectant différents systèmes.

    - Le sink pour AWS SQS dans Kafka Connect consomme les données d'un topic Kafka et les écrit dans SQS. C'est un exemple d'exportation de données distribuée.

    - Le connecteur Telegram source dans Kafka Connect extrait des messages de Telegram et les publie sur un topic Kafka. C'est un exemple d'ingestion de données distribuée.

4. **Kafka Bridge** :

    - Kafka Bridge permet aux applications de communiquer avec un cluster Kafka via des API REST. Cela facilite l'intégration avec des systèmes et applications qui n'utilisent pas nativement les protocoles Kafka.

Technologies Utilisées

Azure

Azure est une plateforme de cloud computing de Microsoft, offrant une gamme de services cloud, notamment le calcul, le stockage, les bases de données, l'intelligence artificielle et l'analytique. Dans ce projet, Azure peut être utilisé pour héberger le cluster Kubernetes (via Azure Kubernetes Service - AKS), offrant une infrastructure évolutive, sécurisée et gérée pour déployer des applications distribuées.

Kubernetes

Kubernetes (k8s) est une plateforme open-source pour l'orchestration de conteneurs, permettant l'automatisation du déploiement, du scaling et de la gestion des applications conteneurisées. Kubernetes assure la résilience et la scalabilité du système en orchestrant les conteneurs Docker et en les répartissant sur les nœuds du cluster. Il gère également la communication entre les services et maintient la disponibilité des applications.

Apache Kafka

Apache Kafka est une plateforme de streaming distribuée capable de gérer des milliers de milliards d'événements par jour. Kafka est conçu pour permettre la publication, le stockage et la consommation de flux de données en temps réel. Dans ce projet, Kafka sert de backbone pour la communication des données entre différentes parties du système, garantissant une transmission rapide et fiable des messages.

Helm

Helm est un gestionnaire de packages pour Kubernetes, permettant de définir, installer et mettre à jour des applications Kubernetes à l'aide de packages appelés "charts". Helm facilite le déploiement et la gestion des applications complexes en encapsulant toutes les configurations nécessaires et en permettant des mises à jour et des rollbacks simplifiés.

Docker

Docker est une plateforme permettant de développer, expédier et exécuter des applications dans des conteneurs. Les conteneurs Docker encapsulent une application avec toutes ses dépendances, assurant ainsi une portabilité et une cohérence entre les environnements de développement, de test et de production. Docker est utilisé pour créer les images des applications qui sont ensuite déployées sur Kubernetes.

Operateur Kafka : Apache Strimzi

Apache Strimzi est un opérateur Kafka pour Kubernetes, facilitant le déploiement, la gestion et la surveillance des clusters Kafka sur Kubernetes. Strimzi simplifie la création de clusters Kafka, leur mise à jour, leur scaling et leur sauvegarde, tout en s'intégrant parfaitement avec les fonctionnalités de Kubernetes.

Amazon SQS (AWS)

Amazon Simple Queue Service (SQS) est un service de file d'attente de messages entièrement géré, offrant une manière fiable et évolutive de découpler et de communiquer entre les microservices. Dans ce projet, SQS peut être utilisé comme une solution de file d'attente complémentaire pour assurer la transmission fiable des messages entre différents composants de l'application.

Apache Camel

Apache Camel est un framework d'intégration open-source basé sur les patterns d'intégration d'entreprise (EIP). Camel permet de définir des règles de routage et de médiation utilisant un domaine spécifique de langage basé sur Java. Dans ce projet, Apache Camel peut être utilisé pour intégrer les différents composants du système distribué, facilitant la communication et la transformation des données entre les services