Laksan MANOHARAN 302

COMPTE RENDU TRAVAUX PRATIQUE – SUJET 3

Etape 1 :

Dans cette première étape, j'ai récupéré le dépôt du producteur depuis GitHub, contenant les fichiers nécessaires tels que ReadMe.md, .env, et le Dockerfile. Après avoir configuré l'environnement avec Docker et adapté le fichier .env pour correspondre à mon environnement, j'ai modifié les fichiers server.js et config.js afin d'utiliser la variable NUMBER_WORD pour la génération des phrases. Le producteur, développé en Node.js avec la bibliothèque kafkajs, a ensuite été conteneurisé grâce à Docker. Après la création de l'image et le lancement du conteneur, des messages de débogage ont confirmé le bon fonctionnement du producteur. J'ai également vérifié la présence des messages via la console RedPanda.

Etape 2 :

La deuxième étape a consisté à développer un consommateur Kafka capable de récupérer les messages du topic mon-super-topic et de les afficher dans la console. J'ai créé un nouveau projet Node.js, en m'appuyant sur kafkajs pour gérer la connexion au broker et la consommation des messages. J'ai implémenté une fonction connexion() permettant de m'abonner au topic et de confirmer la connexion par un message dans la console. Pour chaque message reçu, j'ai affiché son contenu et ajouté une fonction formatTimestamp() pour convertir le timestamp en un format lisible (dd/mm/yyyy à hh:mm). Après quelques ajustements, le consommateur a pu afficher correctement les messages avec la date et l'heure de réception.

Etape 3 :

J'ai commencé par configurer Redis en utilisant Docker, ce qui m'a permis de lancer facilement un serveur Redis local. Ensuite, j'ai intégré la bibliothèque redis dans le fichier consumer.js pour établir la connexion entre l'application Node.js et Redis.

Chaque message consommé depuis le topic Kafka a été découpé en mots. J'ai appliqué un nettoyage des caractères spéciaux afin de ne conserver que des mots valides. Pour chaque mot, j'ai utilisé la commande INCR de Redis afin d'incrémenter le compteur d'occurrences associé.

Pour rendre ces statistiques accessibles, j'ai mis en place un serveur HTTP en utilisant express. J'ai créé un endpoint /stats qui récupère toutes les clés stockées dans Redis ainsi que leur valeur, représentant le nombre d'occurrences de chaque mot.