Développement Avancé

TP3 – Analyse de mots – Wania Jean-Pierre (302)

Thèmes

- Échange de messages
- Le cluster RedPanda
- Producteur / Consommateur
- Utilisation de conteneurs Docker

Étape 1 – « Vive les petits producteurs »

J'ai dans un premier temps fait les changements nécessaires pour exploiter NUMBER_WORD, un champ de paramétrage du fichier .env. Pour ce faire, j'ai d'abord créé une méthode getNumberWord() dans le fichier config.js, qui est utilisée dans le fichier server.js lors de l'appel de la méthode getStringMessage(...).

J'ai ensuite modifié le champ HOST_IP en lui donnant la valeur "red-panda-0:9092" puisque le service sera lancé depuis Docker, il faut donc que l'HOST-IP corresponde pour que le service fonctionne.

La prochaine étape a été de construire mon image Docker — contrairement au dernier TP, j'ai réussi, puisque cette fois-ci j'ai pensé à lancer Docker, donc c'était beaucoup plus simple.

docker build -t producteur "C:\Users\wania\OneDrive\Documents\GitHub\prod-red-panda"

J'ai ensuite démarré mon conteneur en utilisant la commande : docker run producteur.

```
{ topic: 'mon-super-topic', user: 'Liv', message: 'dolore' }
{ topic: 'mon-super-topic', user: 'Frigg', message: 'sit' }
{ topic: 'mon-super-topic', user: 'Astrid', message: 'eu' }
{ topic: 'mon-super-topic', user: 'Thor', message: 'exercitation' }
{ topic: 'mon-super-topic', user: 'Thor', message: 'id' }
{ topic: 'mon-super-topic', user: 'Harald', message: 'in' }
```

Étape 2 – Star de Cinéma : « 26l / 100 km » – Qui suis-je ?

J'ai créé un nouveau projet appelé "topic" — oui, le nom est très mal choisi. Ce projet est le "consommateur" — consumer, c'est un service qui a pour but de s'abonner à notre topic afin de récupérer l'ensemble des messages.

Pour atteindre cet objectif, j'ai créé un fichier consumer.js sur le modèle de producer.js. On y retrouve la fonction asynchrone connexion qui s'abonne au topic, qui lui est passé en paramètre, puis qui traite les messages en parsant la valeur du message en JSON, puis en affichant les messages dans la console. Et ça marche, puisque j'obtiens dans la console ce type de log :

```
{
  date: '12/02/2024 21:06:07',
  utilisateur: 'Harald',
  message: 'consequat'
}
```

- La date qui est retrouvée à partir du timestamp en utilisant la classe Date — cf : getDateFromTimestamp(timestamp).
- L'utilisateur qui est une des informations du message.
- Le message de l'utilisateur.

Étape 3 – « Are you Redis »?

L'objectif de cette troisième étape était de pouvoir compter l'occurrence d'apparition de certains mots dans le topic. C'est l'étape qui m'a pris le plus de temps.

Après avoir ajouté la modification dans le Consommateur pour découper les messages en mots, j'ai essayé de comprendre la manière de procéder pour me connecter à Redis depuis mon projet consommateur.

To connect to a different host or port, use a connection string in the format `redis[s]://[[username][:password]@][host][:port][/db-number]`:

```
createClient({
  url: 'redis://alice:foobared@awesome.redis.server:6380'
});
```

J'ai été induite en erreur par l'information du dessus, qui n'a absolument pas fonctionné dans mon cas. Un autre élément qui a posé problème est le fait que Redis accepte une connexion sans mot de passe, et qu'une erreur n'arrive que lorsqu'on essaye de manipuler les données.

Pour avoir une connexion réellement réussie, j'ai donc procédé de cette manière :

```
export const getClient = async() : Promise<...>  => {
    return createClient( options: {
        host : url,
        username : 'default',
        password: pwd,
        port : "6379"
    });
}
```

Je ne sais pas si c'est recommandé, mais dans mon cas cela fonctionne.

Un autre élément qui m'a posé problème est l'ajout/la modification de données. J'ai essayé d'utiliser INCR directement, mais encore une fois ça ne fonctionnait pas. Donc, je vérifie d'abord si un mot existe avant de l'incrémenter. S'il n'existe pas, je l'ajoute.

```
export const setValue = async(words) : Promise<void>  => {

    for (const word of words){

        if( await client.exists(word) !== 1){
            await client.set(word, 0)
        }
        await client.incr(word);
    }
}
```