

# **BUT 3 INFORMATIQUE**

## **Parcours A FI**

### **Réalisations d'applications**

#### **RAPPORT TP3**

#### **Analyse de mots**

**Prénom et Nom de l'étudiant : Akash Selvaratnam**

**Groupe : 303**

**Promotion : 2023-2024**

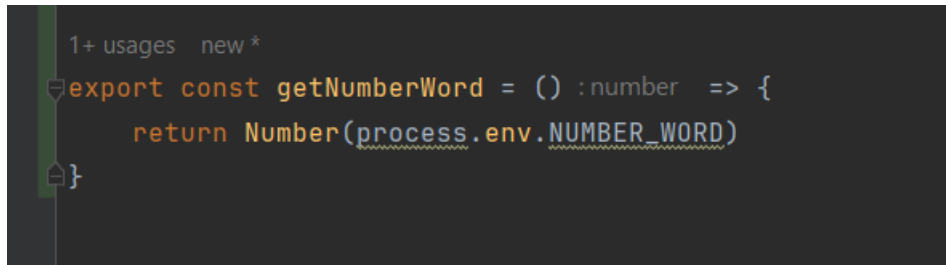
# Sommaire

## Table des matières

Sommaire .....	2
Étape 1 – « Vive les petits producteurs » .....	3
Étape 2 – Star de Cinéma : « 26l / 100 km » - Qui suis-je ? .....	5
Étape 3 –« Are you Redis »? .....	7

## Étape 1 – « Vive les petits producteurs »

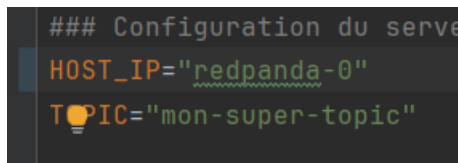
Pour l'étape 1, j'ai créé une nouvelle fonction `getNumberWord()` dans le fichier `config.js` permettant de retourner le nombre de mots, le nombre de mots est une variable déclarée dans le fichier `.env` qu'on peut récupérer à l'aide du module `dotenv` à partir de `process`.



```
1+ usages  new *  
export const getNumberWord = () : number => {  
    return Number(process.env.NUMBER_WORD)  
}
```

Figure 1 : Fonction permettant de retourner le nombre de mots

J'ai ensuite ajouté les configurations nécessaires dans le fichier `.env` pour la configuration de `redPanda` en fonction des valeurs présent dans le `readfile`, avec le host ip et le nom du topic.



```
### Configuration du serveur  
HOST_IP="redpanda-0"  
TOPIC="mon-super-topic"
```

Figure 2 : Configuration du serveur `redPanda`

J'ai construit mon conteneur avec la commande suivante `docker build --tag red-panda-producer .` et j'ai pu lancer mon conteneur avec la commande `docker run -d --name RedPanda-Producer --volume=C:/TP_NODEJS/TP3/prod-red-panda/.env:/home/node/app/.env --network=redpanda-quickstart_redpanda_network red-panda-producer`.

J'ai pu créer mon topic avec mes enregistrements.

```

2024-02-06 16:54:30 topic: 'mon-super-topic',
2024-02-06 16:54:30 user: 'Thor',
2024-02-06 16:54:30 message: 'cillum pariatur nulla'
2024-02-06 16:54:30 }
2024-02-06 16:54:31 {
2024-02-06 16:54:31 topic: 'mon-super-topic',
2024-02-06 16:54:31 user: 'Liv',
2024-02-06 16:54:31 message: 'etiam laborum minim'
2024-02-06 16:54:31 }
2024-02-06 16:54:32 {
2024-02-06 16:54:32 topic: 'mon-super-topic',
2024-02-06 16:54:32 user: 'Viggo',
2024-02-06 16:54:32 message: 'exercitation incididunt consequat'
2024-02-06 16:54:32 }
2024-02-06 16:54:33 { topic: 'mon-super-topic', user: 'Viggo', message: 'ex occaecat ut' }
2024-02-06 16:54:34 { topic: 'mon-super-topic', user: 'Frigg', message: 'id labore quis' }
2024-02-06 16:54:35 { topic: 'mon-super-topic', user: 'Harald', message: 'ea ex mollit' }
2024-02-06 16:54:36 { topic: 'mon-super-topic', user: 'Harald', message: 'do irure sunt' }
2024-02-06 16:54:37 {
2024-02-06 16:54:37 topic: 'mon-super-topic',
2024-02-06 16:54:37 user: 'Thor',
2024-02-06 16:54:37 message: 'excepteur laboris sit'
2024-02-06 16:54:37 }
2024-02-06 16:54:38 { topic: 'mon-super-topic', user: 'Floki', message: 'qui amet tempor' }
2024-02-06 16:54:39 {
2024-02-06 16:54:39 topic: 'mon-super-topic',

```

Figure 3 : Console de redPanda

On peut également visualiser mes enregistrements sur l'interface web de redPanda.

+	0	0	06/02/2024 16:52:30	🔍	{"message":"ipsum deserunt reprehenderit","user":"Liv"}
+	1	0	06/02/2024 16:52:31	🔍	{"message":"voluptate veniam ipsum","user":"Harald"}
+	2	0	06/02/2024 16:52:32	🔍	{"message":"ex laborum non","user":"Astrid"}
+	3	0	06/02/2024 16:52:33	🔍	{"message":"reprehenderit esse eiusmod","user":"Thor"}
+	4	0	06/02/2024 16:52:34	🔍	{"message":"non ea enim","user":"Viggo"}
+	5	0	06/02/2024 16:52:35	🔍	{"message":"consectetur pariatur non","user":"Viggo"}
+	6	0	06/02/2024 16:52:36	🔍	{"message":"esse exercitation consequat","user":"Floki"}
+	7	0	06/02/2024 16:52:37	🔍	{"message":"dolor consectetur ut","user":"Viggo"}

Figure 4 : Interface redpanda

Avec cette étape, j'ai pu comprendre comment fonctionne le serveur RedPanda particulièrement au niveau du producteur en analysant le code fourni ainsi qu'en le démarrant avec les différentes commandes.

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

Pour l'étape 2, j'ai me suis connecté au serveur redpanda au port 19092.

```
const isLocalBroker :boolean = getLocalBroker()
const redpanda :Kafka = new Kafka( config: {
  brokers: [
    isLocalBroker ? `${process.env.HOST_IP}:19092` : 'redpanda-0:19092',
    'localhost:19092'],
});
```

Figure 5 : Configuration redpanda

Ensuite, j'ai créé une fonction consommateur() qui permet de se connecter à mon topic et d'afficher l'ensemble des enregistrement de ces topic.

```
1+ usages Akashos23
export const consommateur = async () :Promise<void> => {
  const consumer :Consumer = redpanda.consumer( config: {groupId: 'test-group'})

  await consumer.connect()
  await consumer.subscribe( subscription: {topic: 'mon-super-topic'})
  await consumer.run( config: {
    eachMessage: async ({topic:string , partition :number , message :KafkaMessage }) :Promise<void> => {
      console.log(getDate(message.timestamp) + `${message.value}`)
      const mes = JSON.parse(message.value.toString());
      mes.message.toString().split( separator: " ").map((mot :string ) =>client.incr(mot));
    },
  })
}
```

Figure 6 : Fonction Consommateur()

J'ai également réalisé une fonction getDate() permettant de transformer le timestamp sous format de date et de changer le format de la date avec le format suivant dd/mm/aaaa à hh : mm.

```
const getDate = (date) :string =>{
  let foo :number = Number.parseInt(date);
  var date :Date = new Date(foo);
  return date.getDate() + "/" + (date.getMonth()+1) + "/" + date.getFullYear() + " à " + date.getHours() + " : " + date.getMinutes()
}
```

Figure 7 : Fonction getDate()

On peut ensuite lancer le serveur afin d'apercevoir le message que nous affiche la console.

```
8/2/2024 à 10 :14{"message":"tempor incidunt labore","user":"Thor"}
8/2/2024 à 10 :14{"message":"eu dolore enim","user":"Frigg"}
8/2/2024 à 10 :14{"message":"ea sint do","user":"Harald"}
8/2/2024 à 10 :14{"message":"cillum aute esse","user":"Liv"}
8/2/2024 à 10 :14{"message":"anim commodo voluptate","user":"Astrid"}
8/2/2024 à 10 :14{"message":"duis officia enim","user":"Astrid"}
8/2/2024 à 10 :14{"message":"quis ipsum culpa","user":"Thora"}
```

*Figure 8 : Résultat console de la fonction consommateur*

Au cours de cette étape, j'ai pu apprendre à créer mon propre consommateur avec la fonction `consumer` en indiquant le nom de mon topic afin de pouvoir le contenu produit par la production et l'affiché via une `console.log`. De plus, cette étape, m'a permis de comprendre comment transformer un timestamp sous format de date et de changer sa forme sous la forme que je souhaite.

## Étape 3 –« Are you Redis »?

Pour l'étape 3, j'ai découpé le message qu'obtient le consommateur et je le découpe sous format de mot afin de pouvoir exécuter la fonction incr de la bibliothèque redis sur chacun des mots que le consommateur obtient.

Tout d'abord, il faut se connecter à notre serveur redis en indiquant le host, le port et le mot de passe.

```
const client = await createClient( options: {  
  host: "myredis",  
  port: 6379,  
  password : "redispwd"  
})  
  
.on('error', err => console.log('Redis Client Error', err))  
.connect();
```

Figure 9 : Connexion au serveur redis

Ensuite dans la fonction connexion du Consommateur, je transforme le message sous format json afin de pouvoir découper ce message sous format de mot grâce à la fonction split() et pour chaque mot, j'utilise la fonction incr pour pouvoir incrémenter l'occurrence des mots (clé).

```
eachMessage: async ({topic:string , partition:number , message :KafkaMessage }) :Promise<void> => {  
  console.log(getDate(message.timestamp) + `${message.value}`)  
  const mes = JSON.parse(message.value.toString());  
  mes.message.toString().split( separator: " ").map((mot:string) => client.incr(mot));  
},
```

Figure 10 : Incrémentation de l'occurrence des mots

On peut ensuite, lancer le serveur et consulter les résultats de la fonction consommateur() sur le site web React.

Occurrence Des Mots	
Nom du mot	Nombre d'occurrence
ut	72
eu	69
velit	69
anim	65
do	65
enim	64
quis	62
ad	61
ipsum	60
qui	57
exercitation	55

Figure 11 : Page React (pour pouvoir consulter les résultats de l'opération INCR de redis)

Pour cette dernière étape, j'ai appris à me connecter à mon serveur redis et j'ai compris comment utiliser la fonction inc proposée par le module redis.