# Rapport

## Étape 1

Après lecture de tous les fichiers, afin d'utiliser la variable NUMBER_WORD du fichier .env, j'ai trouvé que le fichier config.js avait pour rôle d'extraire les variables d'environnement. J'ai alors décidé d'y ajouter une nouvelle fonction qui récupère la variable NUMBER_WORD.

```
export const getNumberWord = () => {
    return process.env.NUMBER_WORD;
}
```

Dans server.js j'importe ensuite la fonction que je viens de créer pour pouvoir l'utiliser et je remplace alors l'ancienne valeur 3 par la fonction que je viens d'importer.

```
import {getConfigNumber, getDebug, getTimeOut, getTopic, getTypeMessage, getNumberWord} from "./config/config.js";
```

```
getStringMessage(getNumberWord())
```

Je remplace ici le HOST_IP dans le fichier .env par une adresse ip aléatoire, dans ce cas ci, c'est celui fourni dans le read.me du projet initial.

```
### Configuration du server RedPanda
HOST_IP="172.16.0.23"
TOPIC="mon-super-topic"
```

En suivant alors le read.me j'ai build mon projet en tant qu'image docker puis lancer la commande run, une erreur est alors survenu disant qu'il n'arrivait pas à trouver le fichier .env ou un autre problème avec ce fichier. Après consultation avec le professeur, on a remplacé le chemin du .env par un chemin absolu. Pour faire fonctionner alors la commande suivante il faudra y rajouter votre chemin absolu au fichier .env
docker run -d --name RedPanda-Producer
--volume="C:\Users\LINos\OneDrive\Bureau\Travaux\3A\Dev avancé\prod-red-panda\.env:/home/node/app/.env"
--network=redpanda-quickstart_redpanda_network red-panda-producer

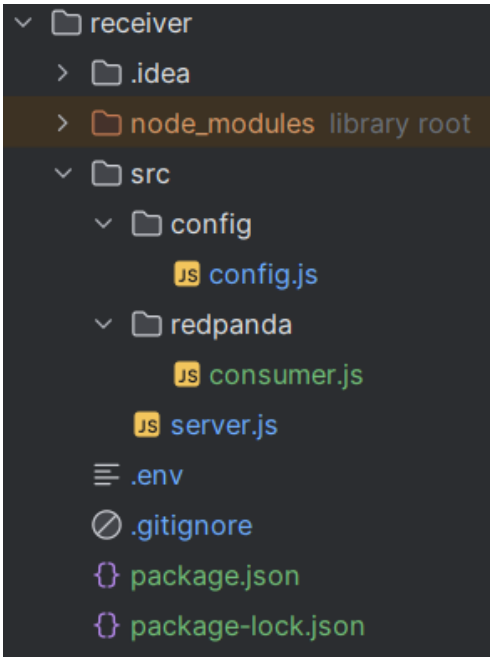
Après la conteneurisation réussie, en me dirigeant vers la console du conteneur en cours d'exécution, je vois bien que des messages y sont générés.

Je me dirige ensuite vers l'interface web de la console redpanda pour y voir si les messages sont bien enregistrés dans le topic "mon-super-topic", j'y trouve bien les différents messages générés en me dirigeant dessus.

```
2024-02-06 16:55:12   user: 'Floki',
2024-02-06 16:55:12   message: 'proident ea aliquip'
2024-02-06 16:55:12 }
2024-02-06 16:55:13 {
2024-02-06 16:55:13   topic: 'mon-super-topic',
2024-02-06 16:55:13   user: 'Thor',
2024-02-06 16:55:13   message: 'aliqua consectetur reprehenderit'
2024-02-06 16:55:13 }
2024-02-06 16:55:14 { topic: 'mon-super-topic', user: 'Viggo', message: 'aliqua non sint' }
2024-02-06 16:55:15 { topic: 'mon-super-topic', user: 'Liv', message: 'pariatur do sunt' }
2024-02-06 16:55:16 {
2024-02-06 16:55:16   topic: 'mon-super-topic',
2024-02-06 16:55:16   user: 'Viggo',
2024-02-06 16:55:16   message: 'magna incididunt sunt'
2024-02-06 16:55:16 }
2024-02-06 16:55:17 {
2024-02-06 16:55:17   topic: 'mon-super-topic',
2024-02-06 16:55:17   user: 'Astrid',
2024-02-06 16:55:17   message: 'quis velit occaecat'
2024-02-06 16:55:17 }
2024-02-06 16:55:18 {
2024-02-06 16:55:18   topic: 'mon-super-topic',
2024-02-06 16:55:18   user: 'Astrid',
2024-02-06 16:55:18   message: 'laborum deserunt ut'
2024-02-06 16:55:18 }
2024-02-06 16:55:19 { topic: 'mon-super-topic', user: 'Harald', message: 'id Lorem quis' }
2024-02-06 16:55:20 {
2024-02-06 16:55:20   topic: 'mon-super-topic',
2024-02-06 16:55:20   user: 'Thora',
2024-02-06 16:55:20   message: 'exercitation dolore nostrud'
2024-02-06 16:55:20 }
```

| | | | | | |
|---|---|---|---|---|---|
| ⊞ | 85 | 0 | 06/02/2024 16:53:51 | ⊘ | {"message":"occaecat ea elit","user":"Fl… |
| ⊞ | 86 | 0 | 06/02/2024 16:53:52 | ⊘ | {"message":"commodo officia do","user":"… |
| ⊞ | 87 | 0 | 06/02/2024 16:53:53 | ⊘ | {"message":"id sit non","user":"Floki"} |
| ⊞ | 88 | 0 | 06/02/2024 16:53:54 | ⊘ | {"message":"labore et tempor","user":"Fl… |
| ⊞ | 89 | 0 | 06/02/2024 16:53:55 | ⊘ | {"message":"eu irure aliqua","user":"Tho… |
| ⊞ | 90 | 0 | 06/02/2024 16:53:56 | ⊘ | {"message":"fugiat mollit enim","user":"… |
| ⊞ | 91 | 0 | 06/02/2024 16:53:57 | ⊘ | {"message":"consequat ipsum occaecat","u… |
| ⊞ | 92 | 0 | 06/02/2024 16:53:58 | ⊘ | {"message":"ut dolor elit","user":"Frigg… |
| ⊞ | 93 | 0 | 06/02/2024 16:53:59 | ⊘ | {"message":"laboris tempor elit","user":… |
| ⊞ | 94 | 0 | 06/02/2024 16:54:00 | ⊘ | {"message":"excepteur eu pariatur","user… |

Étape 2




Je déplace le projet actuel dans un nouveau dossier nommé sender et créer un projet dans un fichier nommé sender pour le consumer. En me basant sur la structure du projet précédent j'y garde le server.js et config.js et ajoute un fichier consumer.js.

Dans le fichier consumer.js, je crée un consumer avec la bibliothèque redpanda et je lui affecte un groupeId, dans ce cas le groupId est "group" mais il peut être un string quelconque. J'initialise aussi la constante topic par la valeur de celle dans le fichier .env, il aurait peut être préférable de faire une fonction dans le fichier config.js pour récupérer cette variable. (Ceci a été implémenté lors de l'étape 3)

Je crée ensuite une fonction connexion qui lancera la fonction du consumer, je connecte alors le consumer au serveur redpanda qui a été initialisé avant puis je l'abonne au topic qui nous intéresse, je mets aussi le paramètre fromBeginning à false pour recevoir seulement les messages envoyés pendant que le consumer est en marche. Je mets ensuite en route le consumer et je configure qu'à chaque message qu'il reçoit il fera afficher le message avec le timestamp avant.

```
const consumer = redpanda.consumer( config: { groupId: 'group' })

const topic = process.env.TOPIC;
1+ usages  new *
export const connexion = async () => {
    await consumer.connect()
    await consumer.subscribe( subscription: { topic, fromBeginning: false })
    await consumer.run( config: {
        eachMessage: async ({ topic, partition, message }) => {
            let json = await JSON.parse(message.value)
            console.log(`[${convertTS(message.timestamp)}] ${json.user} : ${json.message}`)
        },
    })
}
```

Le server.js est assez simple, il importe Consumer puis lance la fonction connexion montré juste avant.

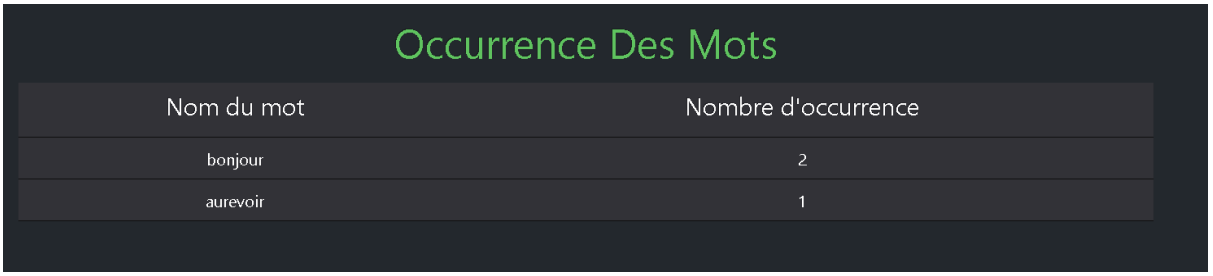
```
import * as Consumer from "./redpanda/consumer.js"

1+ usages
async function start() {

    console.log("Connecting...")
    Consumer.connexion().catch(e => console.error(`[example/consumer] ${e.message}`, e))

}

start()
```

Pour le timestamp, je convertis simplement le string de chiffre que je reçois en nombre puis en date et je récupère chaque valeur dans l'ordre demandé, Il faut noter que le mois commence à 0 il faut donc ajouter 1 à cette valeur.

```
const convertTS = (ts) => {
    let date = new Date(Number.parseInt(ts));
    return `${date.getDate()}/${date.getMonth()+1}/${date.getFullYear()} à ${date.getHours()}:${date.getMinutes()}`
}
```

Étape 3

Après avoir lancé le nouveau projet puis mettre dirigé dans la console, en suivant les consignes du read.me, j'ai pu comprendre le fonctionnement de ce dernier et pu incrémenter la récurrence de certains mots.




Il me faut maintenant connecté le consumer au serveur redis, une première tentative a été faite d'abord en copiant le redisOptions du projet récupéré mais ceci n'avait pas fonctionné, après alors une rapide recherche sur internet j'ai changé la syntaxe par la suivante :

```
1+ usages
export const redisOptions = {
    host: "myredis",
    port: 6379,
    password : "redispwd"
}
```

Après l'importation des fonctions nécessaires ainsi que de la bibliothèque je crée alors une connexion avec le serveur redis. que je stocke dans la constante client

```
const client = await createClient(redisOptions)
    .on('error', err => console.log('Redis Client Error', err))
    .connect()
```

DAns l'éxecution du consumer, je change l'affichage par l'ajout de chaque mot, je coupe simplement le message de chaque utilisateur en une liste de mots avec la fonction split et j'effectue ensuite une boucle sur cette liste pour effectuer l'ajout sur chacun des éléments de la liste qui corresponde à un mot.

```
await consumer.run( config: {
    eachMessage: async ({ topic, partition, message }) => {
        let json = await JSON.parse(message.value)
        let words = json.message.split(" ")
        words.forEach((word)=>{
            client.incr(word)
        })
    },
})
```

Le résultat sur le site est alors le suivant :

## Occurrence Des Mots

| Nom du mot | Nombre d'occurrence |
|---|---|
| id | 24 |
| officia | 23 |
| reprehenderit | 21 |
| dolor | 20 |
| enim | 20 |
| sint | 19 |
| qui | 18 |
| est | 18 |
| laborum | 17 |
| cillum | 17 |
| ullamco | 17 |
| anim | 16 |
| ex | 16 |
| consequat | 16 |
| et | 16 |

Conclusion :

Ce TP m’a permis d’en apprendre plus sur les possibilités que peut offrir différentes bibliothèques, que ce soit la gestion de topic de redpanda où l’implémentation de noSQL avec Redis.