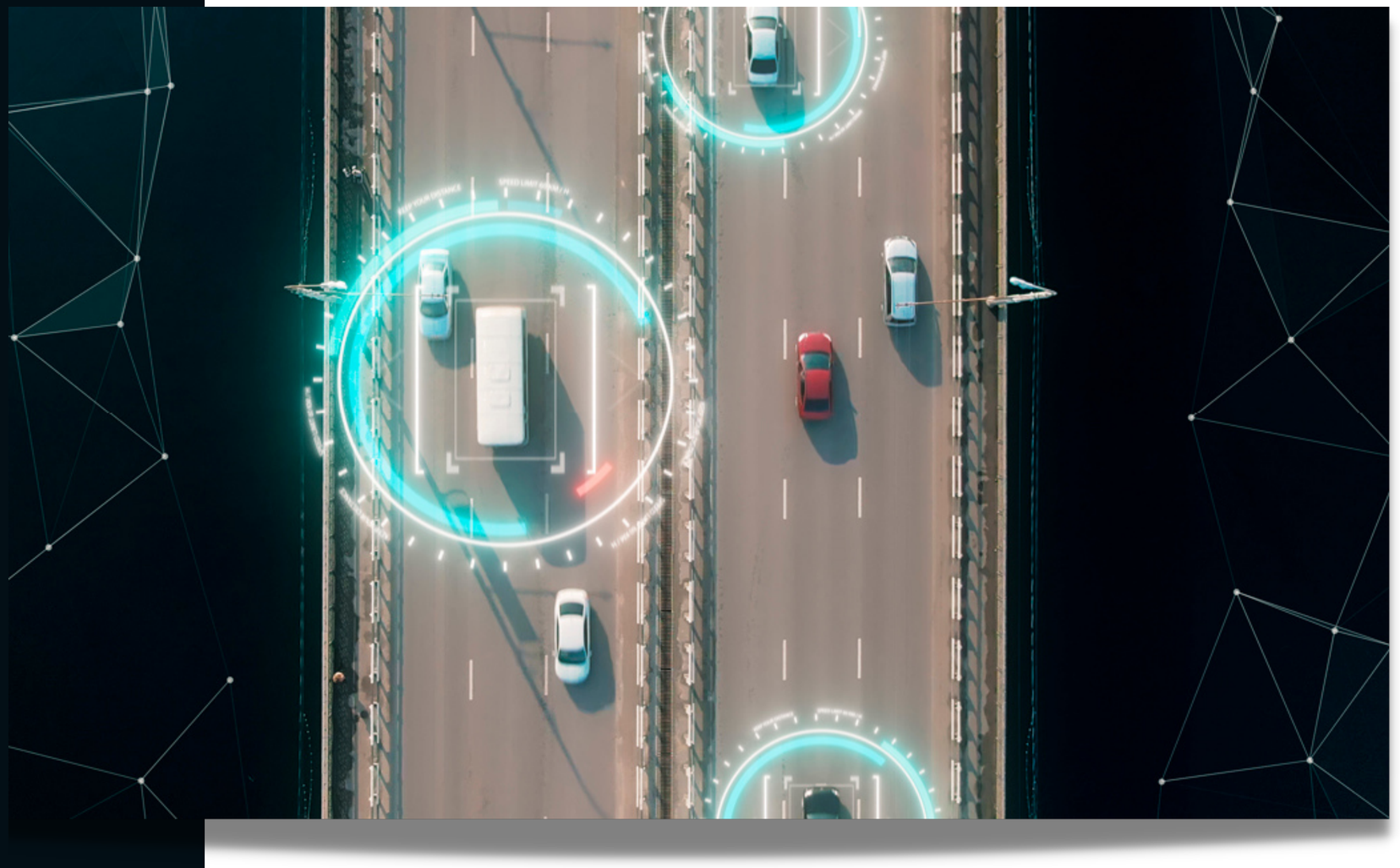
Optimisation du routage de données



Tuteur :
Adrien Sales

Étudiants :
Kenneth Winchester
Antoine Gindre

Dépot Github :
https://github.com/opt-nc/opt-temps-attente-agences-camel

kafka

OPT Nouvelle-Calédonie
LE MONDE À PORTÉE DE MAIN

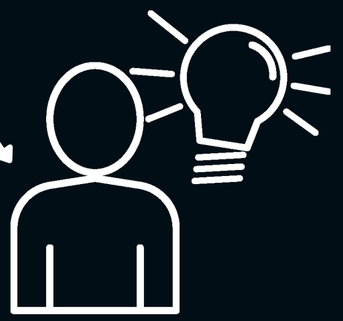
Sommaire

OPT Nouvelle-Calédonie TÉLÉCOMMUNICATIONS

1. Introduction

Postal
Services financiers
Télécommunications
Réseau de distribution

DSI



GLIA

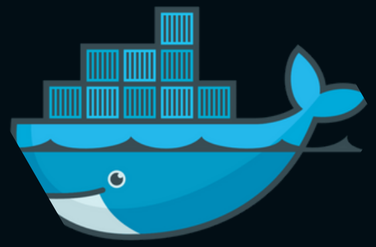
Solutions innovantes

2. Objectif clé

Mettre en place une route camel permettant de récupérer les temps d'attente en agence puis pousser ces données vers un topic Kafka afin de produire des analyses en temps réel (real time analytics).

OPT Nouvelle-Calédonie TÉLÉCOMMUNICATIONS

3. Missions



Consommer les données de l'API

Mise en place d'une route Camel permettant de récupérer les temps d'attentes en agence toute les 5 minutes

1

Alimentation d'un fichier

Via la route Camel précédente, pousser les données automatiquement dans un fichier json

2

Intégration Kafka

Grâce à un Poll régulier de 5 minutes et toujours en utilisant la route Camel, pousser les données dans un topic Kafka dédié avec l’aide d’un Producer

3

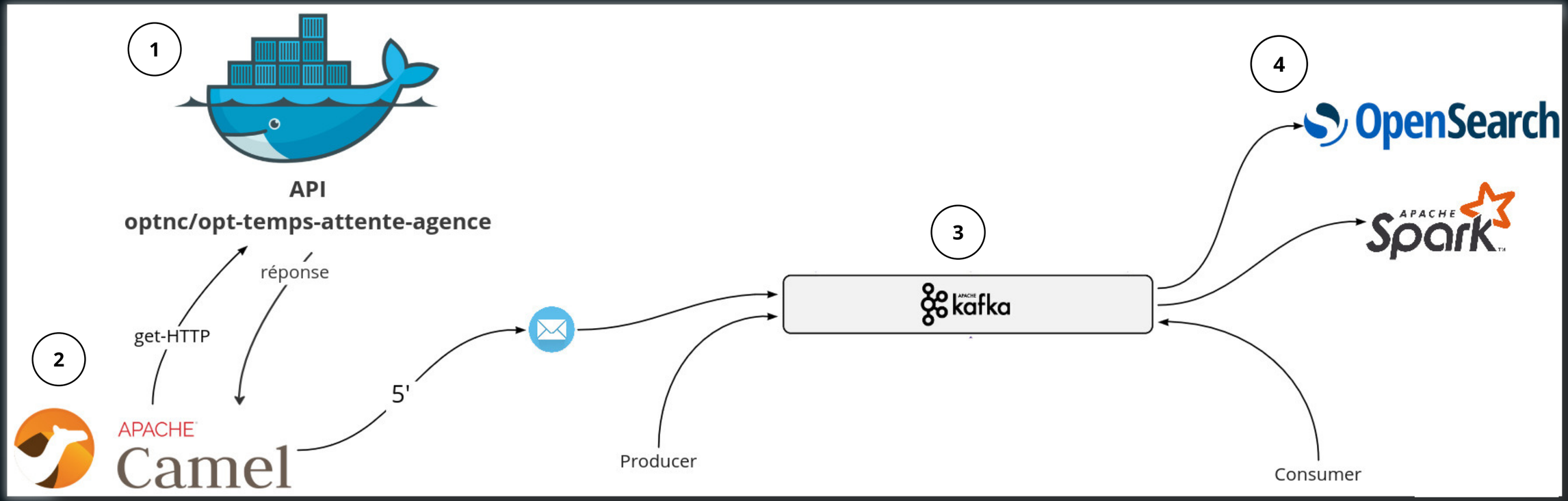
Stockage et data visualisation

Analyse des données en temps réel grâce à une solution de data visualisation tel que OpenSearch

4

OPT Nouvelle-Calédonie TÉLÉCOMMUNICATIONS

Architecture cible



4. Projet

1. API des temps d'attentes

- Déploiement du conteneur docker optnc/opt-temps-attente-agence
- Découverte de la structure de donnée via l'outil Httpie

```
PS C:\Users\5056win> http -v http://localhost:8080/temps-attente/ag
GET /temps-attente/agences HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: localhost:8080
User-Agent: HTTPie/3.2.1



HTTP/1.1 200
Connection: keep-alive
Content-Type: application/json
Date: Mon, 27 Feb 2023 22:12:36 GMT
Keep-Alive: timeout=60
Transfer-Encoding: chunked
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers

[
    {
        "codeESirius": "null",
        "codePostal": "98819",
        "codePostalRefloc": "98819",
        "commune": "MOINDOU",
        "coordonneeX": 367105.9452000037,
        "coordonneeXPrecise": 366555,
        "coordonneeY": 278873.54089999944,
        "coordonneeYPrecise": 278703,
        "designation": "Agence de MOINDOU",
        "idAgence": 4310,
        "idRefloc": "TMP22241",
        "lieuDitOuTribu": "VILLAGE DE MOINDOU",
        "localite": "MOINDOU",
        "localiteRefloc": "MOINDOU",
        "position": {
            "lat": -21.690519046609122,
            "lon": 165.68210414425718
        },
        "realMaxWaitingTimeMs": 0,
        "type": "ANNEXE"
    },
```

4. Projet

1. API des temps d'attentes

Script de test PowerShell

- Health Test
- Test code retour (200)
- Accès aux données
- Test export des données au format CSV

```powershell
### Construction des URL de tests
$actuatorUri = "http://localhost:8080/actuator"
$healthUri = $actuatorUri +"/health"
$infoUri = $actuatorUri +"/info"

### HealtTestURI
Try {
    $healthTestUri =  Invoke-WebRequest -Uri $healthUri -UseBasicParsing
    $healthStatusCode = $healthTestUri.StatusCode

    If ($healthStatusCode -eq "200"){
    Write-Host "API HealthTest OK :" $healthStatusCode
    }Else {
        Write-Host "API HealthTest KO :" $healthStatusCode
    }

}Catch{
    $healthStatusCode = $_.Exception.Response.StatusCode.value__
}

### InfoTestURI
Try {
    $infoTestUri =  Invoke-WebRequest -Uri $infoUri -UseBasicParsing
    $infoStatusCode = $healthTestUri.StatusCode

    If ($infoStatusCode -eq "200"){
        Write-Host "API InfoTest OK :" $infoStatusCode
    }Else {
        Write-Host "API InfoTest KO :" $infoStatusCode
    }

}Catch{
    $infoStatusCode = $_.Exception.Response.StatusCode.value__
}


### Test de récupération du csv de toutes les agences
If (($healthTestUri.StatusCode -eq "200") -and  ($healthTestUri.StatusCode -eq "200")) {
    $webdata = Invoke-WebRequest -Uri http://localhost:8080/temps-attente/agences -UseBasicParsing

    write-host "Récupération des données des agences en cours..."
    $releases = ConvertFrom-Json $webdata.Content
    write-host "Récupération des données des agences terminée."

    ### Export des données dans un fichier CSV
    if ($releases) {
       $releases | Export-Csv .\agences.csv -delimiter ';' -NoTypeInformation
        $file = Get-ChildItem agences.csv
        Write-Host "Le fichier" $file.Name "a été créé dans le répertoire courant."
    }

    Write-Host "L'API contient" $releases.Count "agences."

    ### Test pour une agence
    $4161 = Invoke-WebRequest -Uri http://localhost:8080/temps-attente/agence/4161 -UseBasicParsing
    Write-host "Détail de l'agence 4161" -ForegroundColor Green
    $4161
}
```

4. Projet

2. Première route Camel

Script XML avec Camel JBang

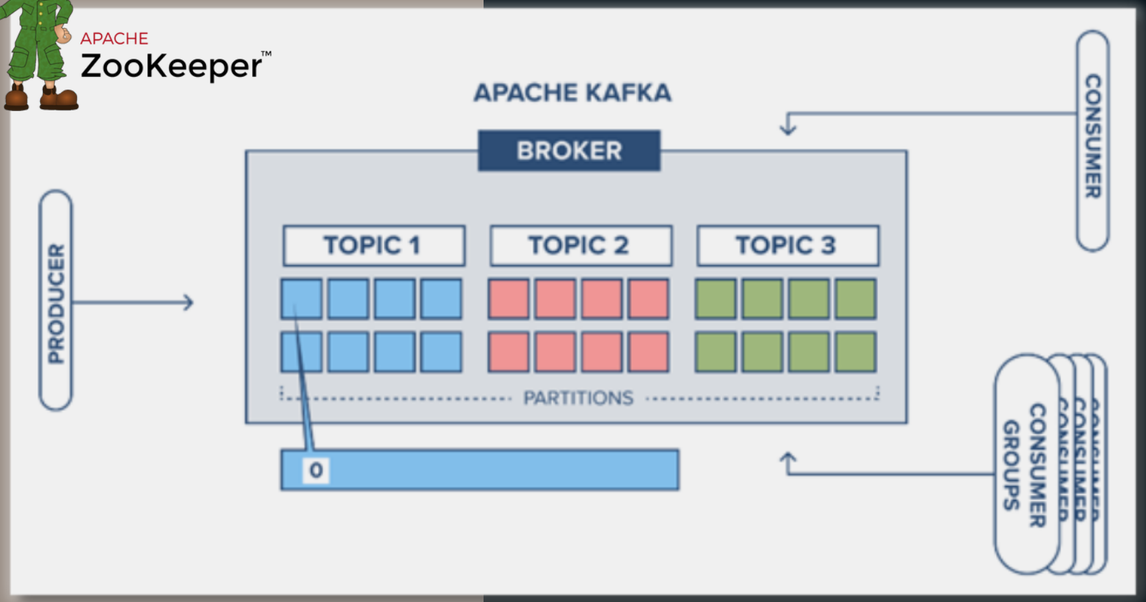
- Connection à l'API
- Méthode GET
- Sortie fichier JSON
- Facilité de developpement
  - *Camel init*
  - *Syntaxe concise*
  - *Lisibilité du code/route*
  - *Dépendances Camel incluent dans JBang*
- Facilité de maintenance
  - Un seul fichier

```
///usr/bin/env jbang "$0" "$@" ; exit $?
// Use org.apache.camel:camel-bom:3.19.0@pom in order to avoid repeating the version in each line
//DEPS org.apache.camel:camel-core:3.19.0
//DEPS org.apache.camel:camel-main:3.19.0
//DEPS org.apache.camel:camel-stream:3.19.0
//DEPS org.apache.camel:camel-http:3.19.0
//DEPS org.apache.camel:camel-jackson:3.19.0

import org.apache.camel.*;
import org.apache.camel.builder.*;
import org.apache.camel.main.*;
import org.apache.camel.model.dataformat.JsonLibrary;

import static java.lang.System.*;

public class TempsAttenteAgenceCamel {

    public static void main(String... args) throws Exception {
        out.println("Hello World");
        out.println("Running camel route...");

        Main main = new Main();

        // Configuration de la route
        main.configure().addRoutesBuilder(new RouteBuilder() {
            public void configure() throws Exception {
                from("timer://foo?period=5000")
                .to("http://localhost:8080/temps-attente/agences")
                .unmarshal().json(JsonLibrary.Jackson)
                .process(exchange -> {
                    // On print la réponse
                    out.println(exchange.getIn().getBody());
                });

            }
        });
        main.run();
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- camel-k: language=xml -->

<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://camel.apache.org/schema/spring"
        xsi:schemaLocation="
            http://camel.apache.org/schema/spring
            https://camel.apache.org/schema/spring/camel-spring.xsd">

    <!-- Write your routes here, for example: -->
    <route id="http-get-route">

        <!-- Ordonnance de l'exécution de la route toutes les 5 minutes -->
        <from uri="timer:http-get-timer??period={{time:300000}}"/>


        <!-- URL source de l'API avec la méthode GET -->
        <to uri="http://localhost:8080/temps-attente/agences?httpMethod=GET"/>

        <!-- Déserialisation des données au format JSON -->
        <unmarshal>
            <json library="Jackson" prettyPrint="true" />
        </unmarshal>

        <!-- Affichage du JSON dans le terminal -->
        <log message="Received data: ${body}"/>

        <!-- JSON -->
        <!-- Conversion des données au format JSON -->
        <marshal>
            <json library="Jackson" prettyPrint="true" />
        </marshal>
        <to uri="file:./data?fileExist=Append&amp;fileName=temps-attente.json"/>

        <!-- Affichage du JSON dans le terminal -->
        <log message="Received data: ${body}"/>

    </route>
</routes>
```

4. Projet

3. Intégration Kafka



- **Système de gestion de message en temps réel**
  Broker / Topic / Producer / Consumer

- **Zookeeper**
  Stocke les métadonnées
  Garanti la fiabilité de Kafka
  Assure la haute disponibilité

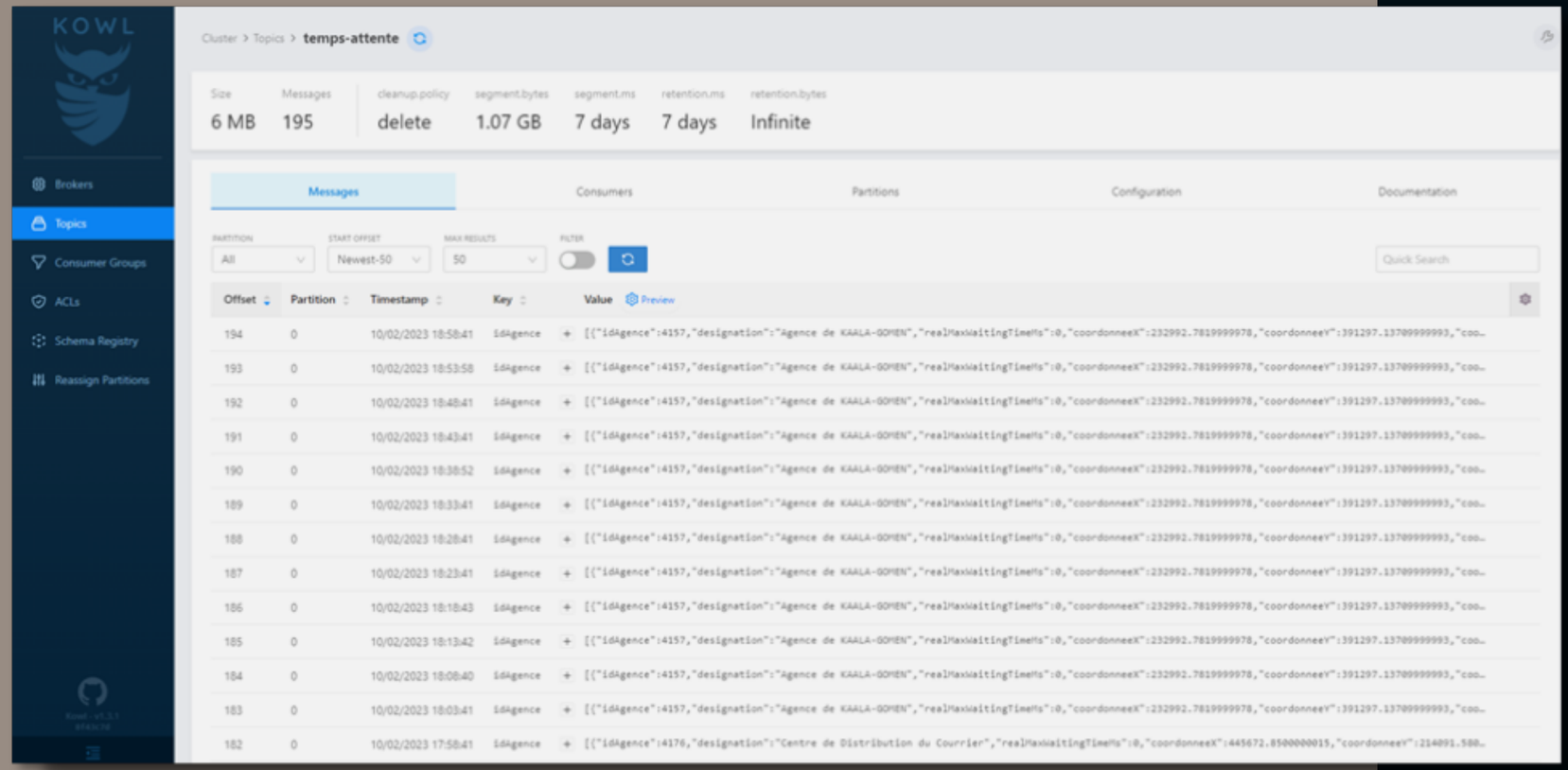
- **Commandes :**
  - **Kafka**
    kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic temps-attente
  - **Kafkacat**
    kafkacat -P -b kafka:9092 -t temps-attente &

4. Projet

4. Kowl pour Kafka

Kowl

- Interface utilisateur Web
- Fonction de recherche et de filtrage
- ...



4. Projet

5. Docker

Docker compose

- Simplification de la gestion
- Rapidité de déploiement
- Centralisation de la configuration dans un fichier YAML

- Portabilité 

```yaml
version: "3"

services:
  optnc :
    hostname : optnc
    image : optnc/opt-temps-attente-agences-api
    container_name: optnc
    ports:
      - "8080:8080"

  zookeeper:
    hostname : zookeeper
    image: ubuntu/zookeeper:latest
    container_name: zookeeper
    ports:
      - "2181:2181"
    healthcheck:
      test: ["CMD", "echo", "ruok"]
      interval: 30s
      timeout: 5s
      retries: 3
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes

  kafka:
    hostname : kafka
    image: ubuntu/kafka:latest
    container_name: kafka
    ports:
      - "9092:9092"
      - "9093:9093"
    depends_on:
      - zookeeper
    environment:
      - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
      - ALLOW_PLAINTEXT_LISTENER=yes

  kowl:
    image: quay.io/cloudhut/kowl:v1.3.1
    container_name: kowl
    restart: always
    ports:
      - "8888:8888"
    depends_on:
      - kafka
    environment:
      - KAFKA_BROKERS=kafka:9092
      - SERVER_LISTENPORT=8888
```

4. Projet

6. Producer Camel Kafka

- Gestion des dépendances
- Ordonnancement (5 minutes)
- Collecte des données
- Alimentation topic kafka
- Contraintes Kafka (clé / format de données)
- Simplicité d'exécution de la route

  camel run route-kafka.xml --dep camel.dependencies=org.apache.camel:camel-kafka

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- camel-k: language=xml -->

<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://camel.apache.org/schema/spring"
        xsi:schemaLocation="
            http://camel.apache.org/schema/spring
            https://camel.apache.org/schema/spring/camel-spring.xsd">

    <!-- Write your routes here, for example: -->
    <route id="http-get-route">

        <!-- Ordonnance de l'exécution de la route toutes les 5 minutes -->
        <from uri="timer:http-get-timer??period={{time:300000}}"/>

        <!-- URL source de l'API avec la méthode GET -->
        <to uri="http://localhost:8080/temps-attente/agences?httpMethod=GET"/>

        <!-- Déserialisation des données au format JSON -->
        <unmarshal>
            <json library="Jackson" prettyPrint="false" />
        </unmarshal>

        <!-- JSON -->
        <!-- Conversion des données au format JSON -->
        <marshal>
            <json library="Jackson" prettyPrint="false" />
        </marshal>

        <!-- URL de destination Kafka et paramétrage de la clé -->
        <to uri="kafka:temps-attente?brokers=localhost:9092&amp;key=idAgence"/>

    </route>
</routes>
```

4. Projet

7. Consumer Camel Kafka

- Consommation des données en temps réel
- Spécification d'un groupe id

- Alimentation d'un fichier Json en sortie
- Simplicité d'exécution des routes producer et consumer
  camel run route-kafka.xml route-kafka-json.xml --dep camel.dependencies=org.apache.camel:camel-kafka > camel-output.log &

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- camel-k: language=xml -->

<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://camel.apache.org/schema/spring"
        xsi:schemaLocation="
            http://camel.apache.org/schema/spring
            https://camel.apache.org/schema/spring/camel-spring.xsd">

    <!-- Write your routes here, for example: -->
    <route id="kafka-get-route">

        <!-- URL source Kafka et indication du groupid -->
        <from uri="kafka:temps-attente?brokers=localhost:9092&amp;groupId=temps"/>

        <!-- Déserialisation des données au format JSON -->
        <unmarshal>
            <json library="Jackson" prettyPrint="true" />
        </unmarshal>

        <!-- Affichage du JSON dans le terminal -->
        <log message="Received data: ${body}" loggingLevel="DEBUG"/>

        <!-- Conversion des données au format CSV -->
        <marshal>
            <json library="Jackson" prettyPrint="true" />
        </marshal>

        <!-- Route de destination du fichier avec ajout des données à la fin du csv-->
        <to uri="file:./data.csv?fileExist=Append&amp;fileName=temps-attente.json"/>

        <!-- Affichage du CSV dans le terminal -->
        <log message="Received data: ${body}" loggingLevel="DEBUG"/>

    </route>
</routes>
```

4. Projet

8. Stockage et data visualisation

OpenSearch

- Moteur de recherche et d'analyse de données
- Développé et maintenu par AWS
- Opensource
- Fonctionnalité de geolocalisation, IA et ML

```yaml
version: '3'

services:

  opensearch: # This is also the hostname of the container within the Docker network (i.e. https://opensearch/)
    image: opensearchproject/opensearch:latest # Specifying the latest available image - modify if you want a specific version
    container_name: opensearch
    environment:
      - cluster.name=opensearch-cluster # Name the cluster
      - node.name=opensearch # Name the node that will run in this container
      - discovery.seed_hosts=opensearch # Nodes to look for when discovering the cluster
      - cluster.initial_cluster_manager_nodes=opensearch # Nodes eligibile to serve as cluster manager
      - bootstrap.memory_lock=true # Disable JVM heap memory swapping
      - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m" # Set min and max JVM heap sizes to at least 50% of system RAM
    ulimits:
      memlock:
        soft: -1 # Set memlock to unlimited (no soft or hard limit)
        hard: -1
      nofile:
        soft: 65536 # Maximum number of open files for the opensearch user - set to at least 65536
        hard: 65536
    user: "1000:1001"
    #volumes:
      #- ./volumes/opensearch_data:/usr/share/opensearch/data:rw # Creates volume called opensearch-data and mounts it to the container
    ports:
      - 9200:9200 # REST API
      - 9600:9600 # Performance Analyzer
    networks:
      - opensearch-net # All of the containers will join the same Docker bridge network
### Si l'erreur  "[1]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]" s'affiche, exécuter la commande ci-dessous
### sudo sysctl -w vm.max_map_count=262144

  opensearch-dashboards:
    image: opensearchproject/opensearch-dashboards:latest # Make sure the version of opensearch-dashboards matches the version of opensearch installed on other nodes
    container_name: opensearch-dashboards
    ports:
      - 5601:5601 # Map host port 5601 to container port 5601
    expose:
      - "5601" # Expose port 5601 for web access to OpenSearch Dashboards
    environment:
      OPENSEARCH_HOSTS: '["https://opensearch:9200"]' # Define the OpenSearch nodes that OpenSearch Dashboards will query
    networks:
      - opensearch-net

networks:
  opensearch-net:
    driver: bridge

volumes:
  esdata:
    driver: local
    driver_opts:
      o: uid=1000,gid=1000
```

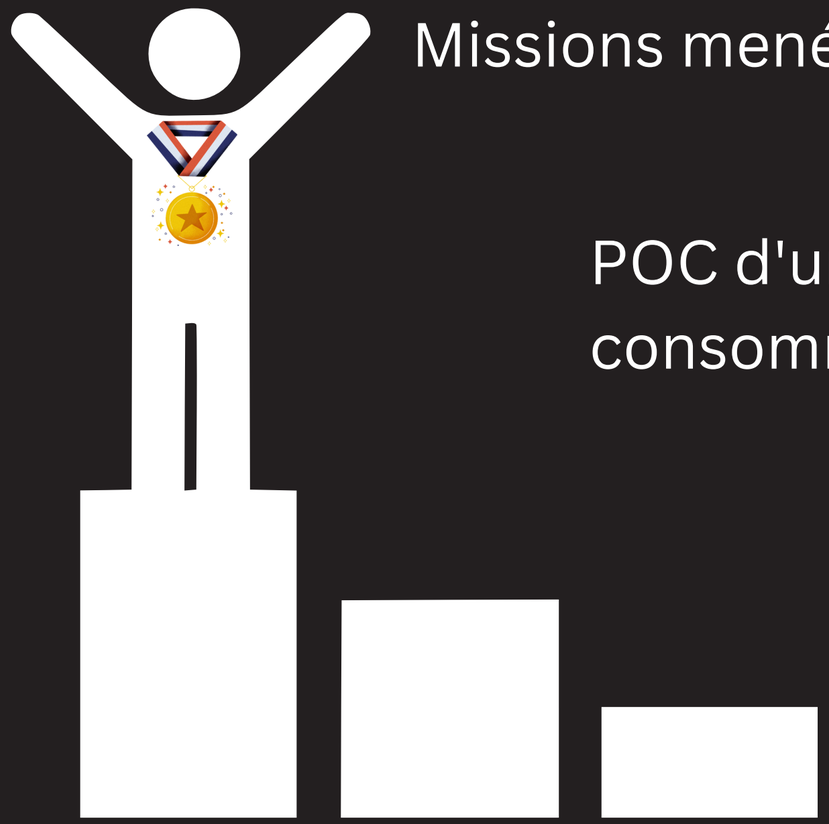
5. Résultats attendus

- Prise en main de l'API publique des temps d'attente
- Mise en place de la route Camel
- Interconnexion de chaque brique du projet
- Capitalisation du GLIA sur l'architecture mise en place

6. Améliorations futures

- Suppression progressive de Zookeeper par la communauté Kafka
- Les limites d'apache Camel
  - baisse de performance si volume de données trop important

Conclusion



Missions menées à bien

POC d'une nouvelle méthode de consommation des données

Montée en compétences importante

Dommage que ce soit déja fini !

OPT Nouvelle-Calédonie TÉLÉCOMMUNICATIONS

Merci !