جامعة شعيب الدكالي
universite chouaib doukkali

ENSA
Ecole Nationale des Sciences Appliquées d'El Jadida

Filière : IITE3

Module : Architecture et Infrastructure Big Data

Année universitaire : 2023/2024

Travaux Pratique 8 :

Apache Kafka+ Spark Streaming + Flume

Rédigé par :

HMAYDA Abdessamad

Encadré par :

Pr. Kalloubi Fouad

جامعة شعيب الدكالي
universite chouaib doukkali

ENSA
Ecole Nationale des Sciences Appliquées d'El Jadida

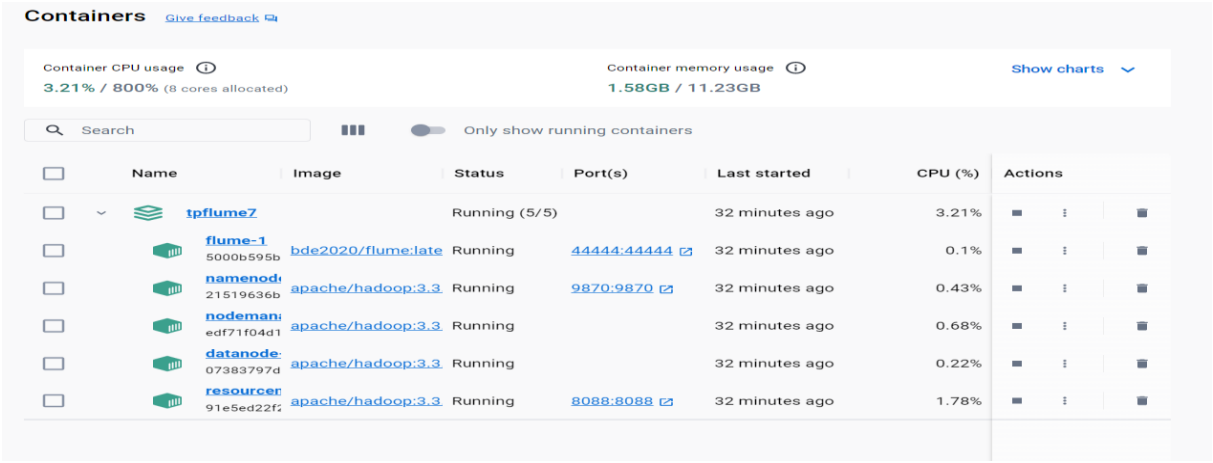
Table des matières

I. Apache Kafka :

1. Mise en place

-Cloner ce projet, et exécuter tapez « docker compose up » à l'intérieure du dossier du projet

- tapez « docker ps » pour s'assurer que les conteneurs marchent bien comme l'image d'apres docker desktop montre



2. Introduction :

a. Création d'un topic « ensademo » :

```
PS C:\Users\Hmayda\OneDrive\Bureau\ensaj\Architecture et infrastructure big data\TP\TP 8 KAFKA> docker exec -it  kafka bash
root@52caf236345a:/# kafka-topics.sh --bootstrap-server localhost:9092 --create --topic ensademo  --replication-factor 1 --partitions 1
Created topic ensademo.
root@52caf236345a:/#
```

b. S'assurer que le topic a été créé :

```
root@52caf236345a:/# kafka-topics.sh --list --bootstrap-server localhost:9092
ensademo
root@52caf236345a:/#
```

c. Démarrage du producteur :

```
root@52caf236345a:/# kafka-console-producer.sh --broker-list localhost:9092 --topic ensademo
>Bonjour je suis en train de produire hh
```

d. Démarrage du consommateur :

```
PS C:\Users\Hmayda\OneDrive\Bureau\ensaj\Architecture et infrastructure big data\TP\TP 8 KAFKA> docker exec -it kafka bash
root@52caf236345a:/# kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic ensademo --from-beginning
Bonjour je suis en train de produire hh
```

e. Affichage du contenu de notre topic :

f. Travail à faire :

i. Créer un topic « ensaDemoPart » :

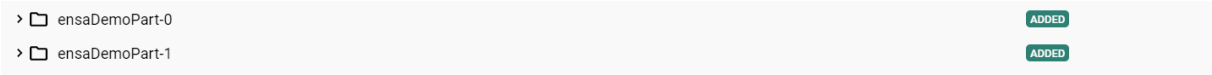
-Cette fois on va créer le topic avec un nombre de partitions égal à 2 :

```
root@52caf236345a:/# kafka-topics.sh  --bootstrap-server localhost:9092 --create --topic ensaDemoPart --replication-factor 1 --partitions 2
Created topic ensaDemoPart.
root@52caf236345a:/#
```

-la liste des topics

```
root@52caf236345a:/# kafka-topics.sh --list --bootstrap-server localhost:9092
__consumer_offsets
ensaDemoPart
ensademo
```

➔ Puisque le nombre de partitions de ensaDemoPart est 2, donc au niveau de « /tmp/kafka-logs » on aura deux partitions des logs de ce topic, comme indiqué dans l'image suivante :



ii. Créer un topic avec une facteur de réplication qui vaut 2 :

```
root@52caf236345a:/# kafka-topics.sh  --bootstrap-server localhost:9092 --create --topic ensaDemoRep --replication-factor 2 --partitions 1
Error while executing topic command : Replication factor: 2 larger than available brokers: 1.
[2023-12-08 20:19:46,594] ERROR org.apache.kafka.common.errors.InvalidReplicationFactorException: Replication factor: 2 larger than available brokers: 1.
 (kafka.admin.TopicCommand$)
root@52caf236345a:/#
```

Le message d'erreur Facteur de réplication : 2 plus grand que les courtiers(brokers) disponibles : 1 indique qu'on essaye de créer un sujet Kafka avec un facteur de réplication supérieur au nombre de courtiers disponibles dans notre cluster Kafka.

II. Spark streaming :

1) Lancement de SparkStreamingContext

1)- Tout d'abord allez au terminal de votre namenode et tapez : hdfs dfs-chmod-R 777 /

2)- lancez le mode interactif de spark :

```
bash-5.0# ./spark-shell --master yarn --conf spark.ui.port=12378
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/12/08 23:52:16 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
23/12/08 23:52:19 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Spark context Web UI available at http://6f3e91cea76c:12378
Spark context available as 'sc' (master = yarn, app id = application_1702070899983_0002).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.3.0
      /_/

Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 1.8.0_275)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

- ✓ Remarques :
  - Spark streaming est utilisé pour traiter les donnees en mode streaming, il s'appuie sur un service web appelé StreamingContext (on utilisera netcat dans notre cas pour simuler un service web).
  - StreamingContext exécute un traitement perpétuel de données à intervalles réguliers

- Nous ne pouvons pas avoir plusieurs contextes en meme temps, donc on doit arrêter sparkContexte avant de lancer sparkStreaming

3) -arrêtez SparkContext et lancez SparkStreamingContext :

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

import org.apache.spark.streaming._
import org.apache.spark.SparkConf
sc.stop
val conf=new SparkConf().setAppName("Streaming").setMaster("yarn")
val ssc=new StreamingContext(conf,Seconds(10))

// Exiting paste mode, now interpreting.

23/12/09 00:24:29 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
import org.apache.spark.streaming._
import org.apache.spark.SparkConf
conf: org.apache.spark.SparkConf = org.apache.spark.SparkConf@78ceb8cd
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@afee49e

scala>
```

2)-Développement d'une application « wordCount » en utilisant Spark Streaming :

Tout d'abord nous allons créer une application Spark en utilisant Scala(2.11.0) et sbt (1.3.3) et apres importer spark et spark-streaming

SparkStreaming.scala | build.sbt

```
ThisBuild / version := "v1"

ThisBuild / scalaVersion := "2.11.0"



libraryDependencies += "org.apache.spark" %% "spark-core" % "1.6.0"
libraryDependencies += "org.apache.spark" %% "spark-streaming" % "1.6.0"
libraryDependencies += "org.apache.logging.log4j" % "log4j-api" % "2.17.1"
libraryDependencies += "org.apache.logging.log4j" % "log4j-core" % "2.17.1"

lazy val root = (project in file("."))
  .settings(
    name := "SparkStreaming"
  )
```

جامعة شعيب الدكالي
universite chouaib doukkali

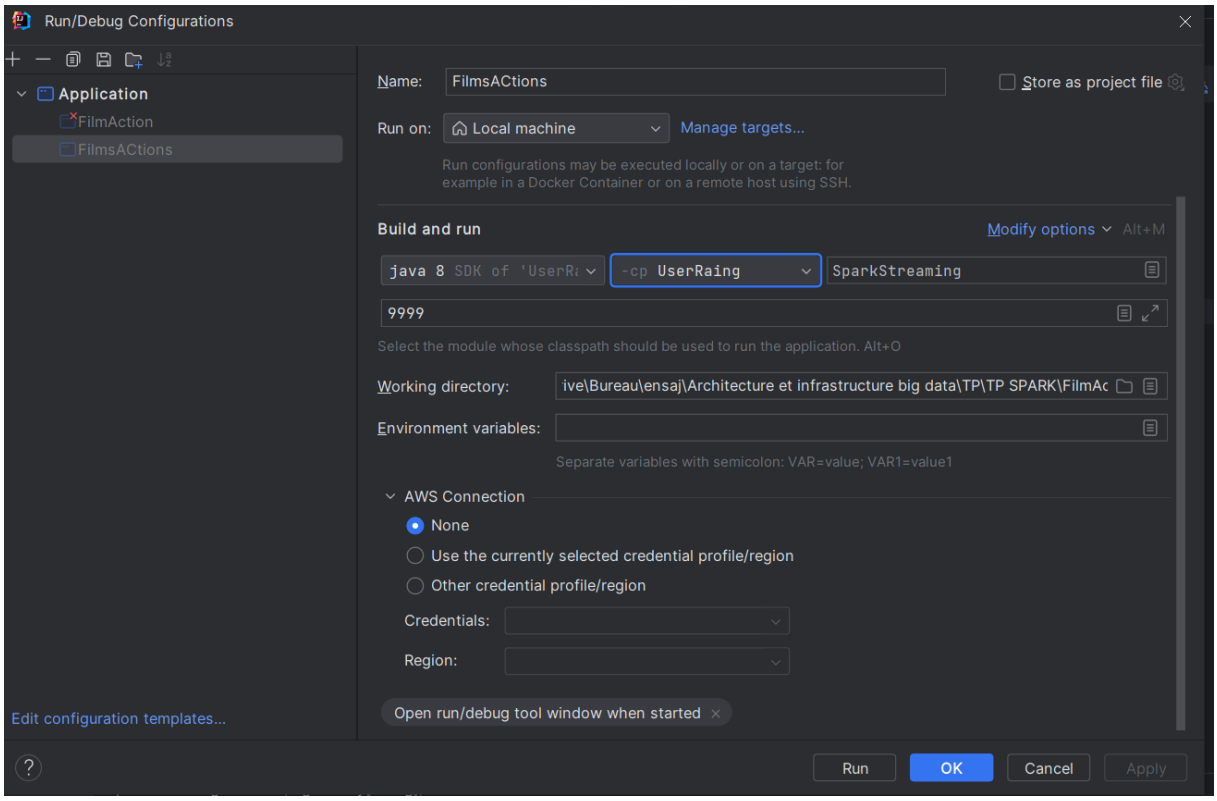
ENSA
Ecole Nationale des Sciences Appliquées d'El Jadida

Voici le code de notre application

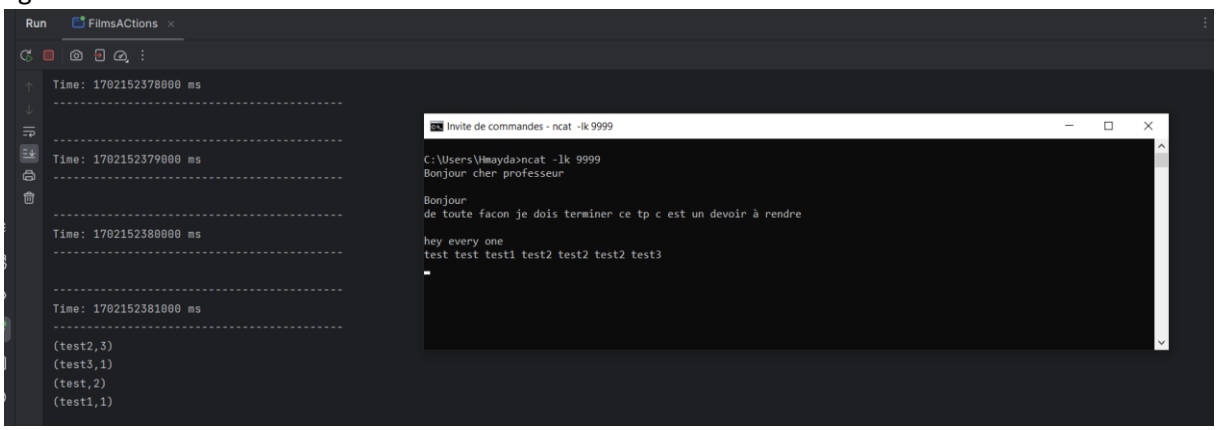
```scala
import org.apache.log4j.{Level, Logger}
import org.apache.spark._
import org.apache.spark.streaming._

object SparkStreaming{
  def main (args : Array[String]) : Unit = {
    if (args.length != 1) {
      println("Usage: NetcatReceiver <port>")
      System.exit( status = 1)
    }

    Logger.getLogger ( name = "org").setLevel(Level.ERROR)

    val port=args(0).toInt
    val conf = new SparkConf().setAppName("ActionMovies").setMaster("local[*]")
    val ssc=new StreamingContext(conf,Seconds(1))

    val lines = ssc.socketTextStream( hostname = "localhost", port)

    val words=lines.flatMap(_.split( regex = " "))
    val pairs=words.map(word=>(word,1))
    val wordCounts=pairs.reduceByKey(_+_)
    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()
  }
}
```

SparkStreaming › main(args: Array[String])

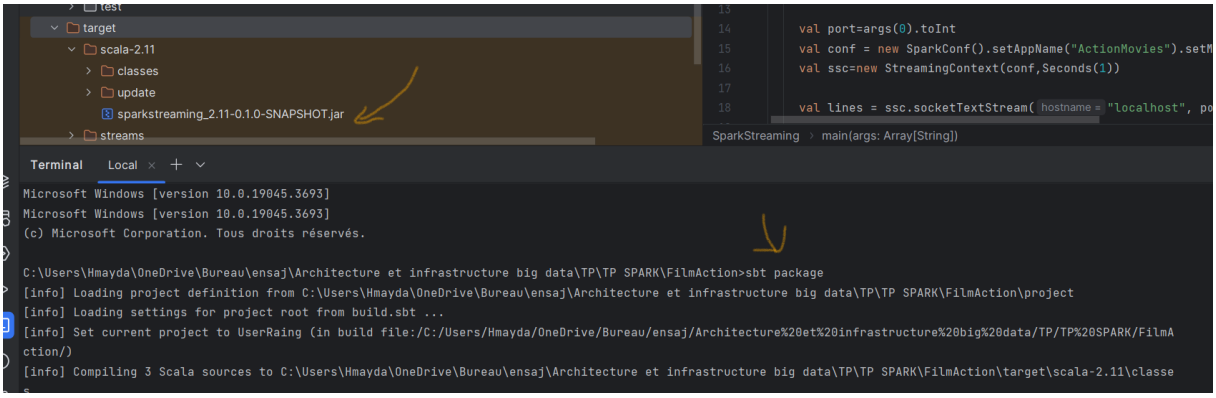
Exécuter cette application en passant 9999 comme argument



Également nous allons simuler un service web en utilisant netcat :



-donc notre application repond à nos besoin , on doit maintenant contsruire le jar de cette application :



- Ensuite, on doit mettre le jar dans HDFS, pour le faire deplacez le vers le dossier du volume hmayda et exécuter cette commande de hdfs :

```
sh-4.2$ hdfs dfs -put "/hmayda/sparkstreaming_2.11-v1.jar" /hmayda/
WARNING: log4j.properties is not found. HADOOP_CONF_DIR may be incomplete.
sh-4.2$
```

- Lancez le jar avec spark-submit

```
bash-5.0# ./spark-submit --class SparkStreaming  /hmayda/sparkstreaming_2.11-v1.jar 9999
```

-utiliser netcat et envoyez des messages sur le port 9999 et voici le résultat du traitement effectué par l'application spark