EXERCICE : premier pas apache kafka

1. Création de topic
2. Envoyer un flux dans un topic via un producer
3. Consommer le flux d’un topic

Solution :

En amant :

* 1. Donnez les droits sur l’arbo de kafka
     1. sudo chmod 777 -R kafka\_2.11-2.0.0/
  2. Ouvrir un onglet ou fenetre a votre guise pour chacune des commandes que vous allez exécuter

1. Lancer le serveur
   1. Zookeeper :

bin/zookeeper-server-start.sh config/zookeeper.properties

* 1. Kafka :

bin/kafka-server-start.sh config/server.properties

1. Créer un topic

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic jems

bin/kafka-topics.sh --list --zookeeper localhost:2181

1. Envoyer message

bin/kafka-console-producer.sh --broker-list localhost:9092 --topic jems

1. Consommer message

bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic jems --from-beginning

Exercice : multiples consumer et Producer

1. Transférer via ligne de commande le contenu d’un fichier à partir d’un Producer a un topic
2. Consommer le flux d’un topic dans un autre consumer

Exercice : production de flux via scala

Sur built.sbt :

*name* := "kafkaproject"  
  
*version* := "0.1"  
  
*scalaVersion* := "2.11.0"  
  
  
//kafka-clients  
  
*libraryDependencies* ++= *Seq*(  
 // The excludes of jms, jmxtools and jmxri are required as per https://issues.apache.org/jira/browse/KAFKA-974.  
 // The exclude of slf4j-simple is because it overlaps with our use of logback with slf4j facade; without the exclude  
 // we get slf4j warnings and logback's configuration is not picked up.  
 "org.apache.kafka" % "kafka-clients" % "0.10.1.0"  
 exclude("javax.jms", "jms")  
 exclude("com.sun.jdmk", "jmxtools")  
 exclude("com.sun.jmx", "jmxri")  
 exclude("org.slf4j", "slf4j-simple"),  
 // Logback with slf4j facade  
 "ch.qos.logback" % "logback-classic" % "1.0.13",  
 "ch.qos.logback" % "logback-core" % "1.0.13",  
 "org.slf4j" % "slf4j-api" % "1.7.5"  
)

creation du scala producer

**import** java.util.Properties  
**import** org.apache.kafka.clients.producer.\_  
  
**object** KafkaProducer **extends** App {  
  
 **val** *props* = **new** Properties()  
 *props*.put("bootstrap.servers", "localhost:9092")  
  
 *props*.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer")  
 *props*.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer")  
  
 **val** *producer* = **new** KafkaProducer[String, String](*props*)  
  
 **val** *TOPIC*="jems"  
 **var** *i*:Int=0;  
 **while**(**true**){  
 *i*+=1  
 **val** record = **new** ProducerRecord(*TOPIC*, "key", s"toto **$***i*")  
 *producer*.send(record)  
 }  
  
  
 *producer*.close()  
}

Exercice : création consommation de flux via scala

**import** java.util  
  
**import** java.util.Properties  
  
**import** org.apache.kafka.clients.consumer.KafkaConsumer  
**import** scala.collection.JavaConverters.\_  
  
**object** KafkaConsumer **extends** App {  
 **val** *TOPIC*="jems"  
  
 **val** *props* = **new** Properties()  
 *props*.put("bootstrap.servers", "localhost:9092")  
  
 *props*.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer")  
 *props*.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer")  
 *props*.put("group.id", "something")  
  
 **val** *consumer* = **new** KafkaConsumer[String, String](*props*)  
  
 *consumer*.subscribe(util.Collections.*singletonList*(*TOPIC*))  
  
 **while**(**true**){  
 **val** records=*consumer*.poll(100)  
 **for** (record<-records.asScala){  
 *println*(record.value)  
 }  
 }  
  
}