APACHE

kafka

Application Synchronise vs Asynchronies

Application Synchronise : Application sends a message directly to another application example service call.
In this scenario if application 1 sends a message it just sends it and forget about it. There is a problem here .What about if application 2 is down ? Application 2 never receive a message . (Banking system or critical system when we lose 1 message it will be disaster

Asynchronies : means two application may not be up at the same time . For example application 1 is trying to send a message to application 2 but application 2 is down or its not available or there is a network problem then application 2 is not receiving a message .
In Asynchronies communication is blocking call**. In another word application 1 remains block till application 2 sends an acknowledgement**

To avoid all this issue we use **messaging system** .
Is a data structure and application itself which receive the message stores it and when the receiving application is available ,it pulls the message from the queue.
The basis of messaging system is a **queue or topic.**

Publisher /Subscriber vs Point to Point Communication

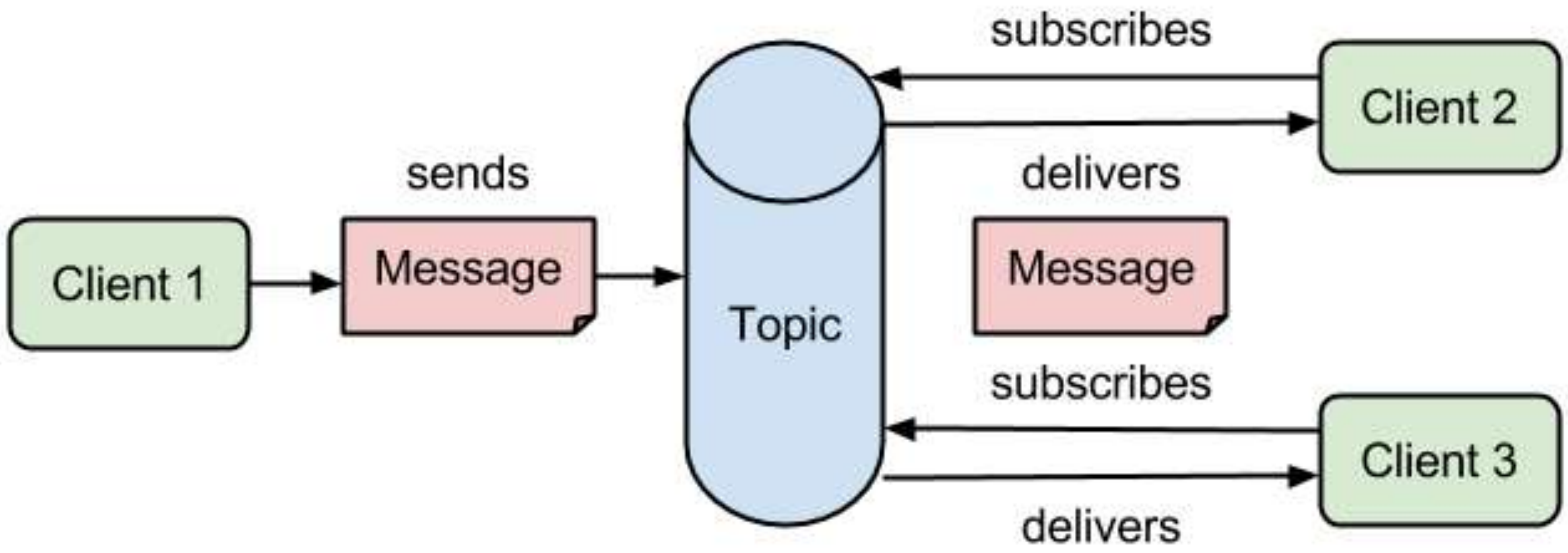
We have two types of communications : Point to points and Publisher /Subscriber communication

Point to points : Application 1 is a producer and application 2 is a consumer .Application 1 sends a message to queue and application 2 reads it from queue then deletes it from the queue.

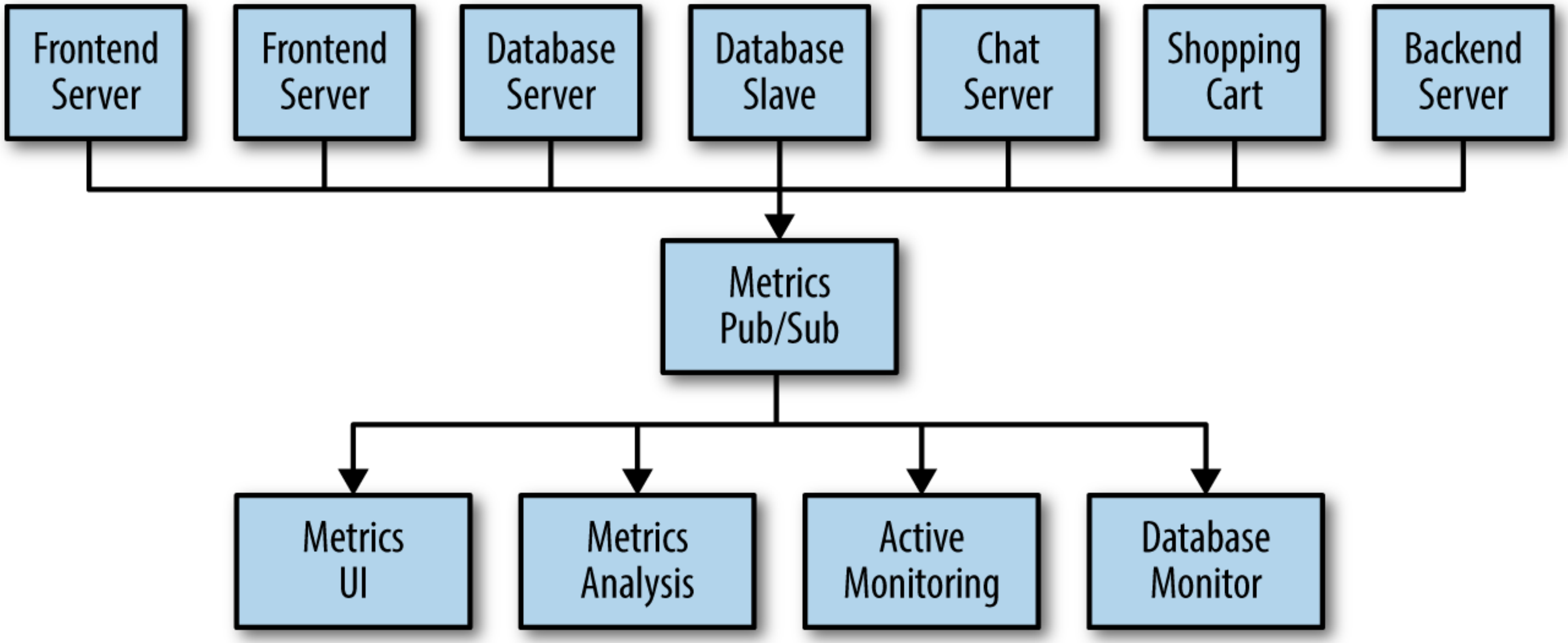
Publisher /Subscriber : Client 1 sends a message to topic and any clients who has subscribed the topic will receive a copy of the message. One producer and multiple consumers.

Only clients who has subscribed to the topic to be able to get the message.

Publish/Subscribe Messaging



Publish/Subscribe Messaging - Advanced



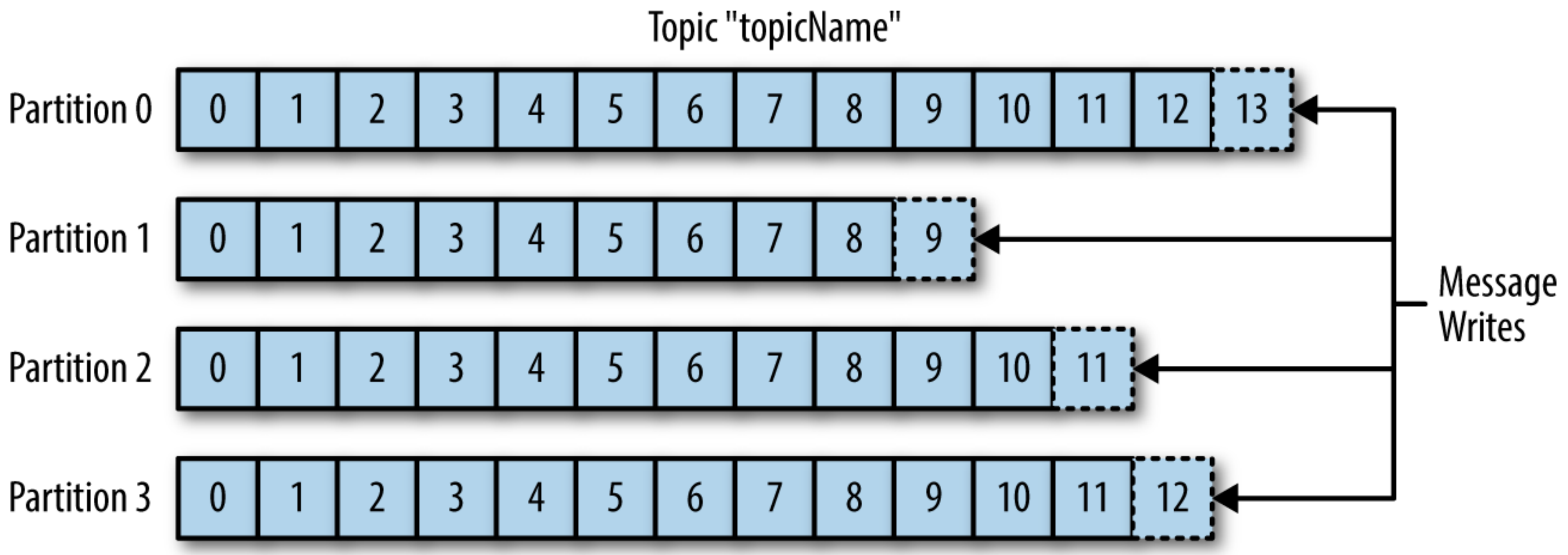
Messages & Baches

- The unit of data within Kafka is called a message.
- A message is simply an array of bytes as far as Kafka is concerned, and hence has its contents have no meaning to it
- A message can have an optional bit of metadata, which is referred to as a key
- The key is also a byte array and, as with the message, has no specific meaning to Kafka
- Keys are useful when messages need to be partitioned
- For efficiency, a Kafka producer writes messages to a **Kafka broker in batches**
- Batching is done because posting individual messages to the broker takes way too much time due to the network latency involved
- Though, this reduces the throughput
- Batches are also typically compressed, providing more efficient data transfer and storage at the cost of some processing power

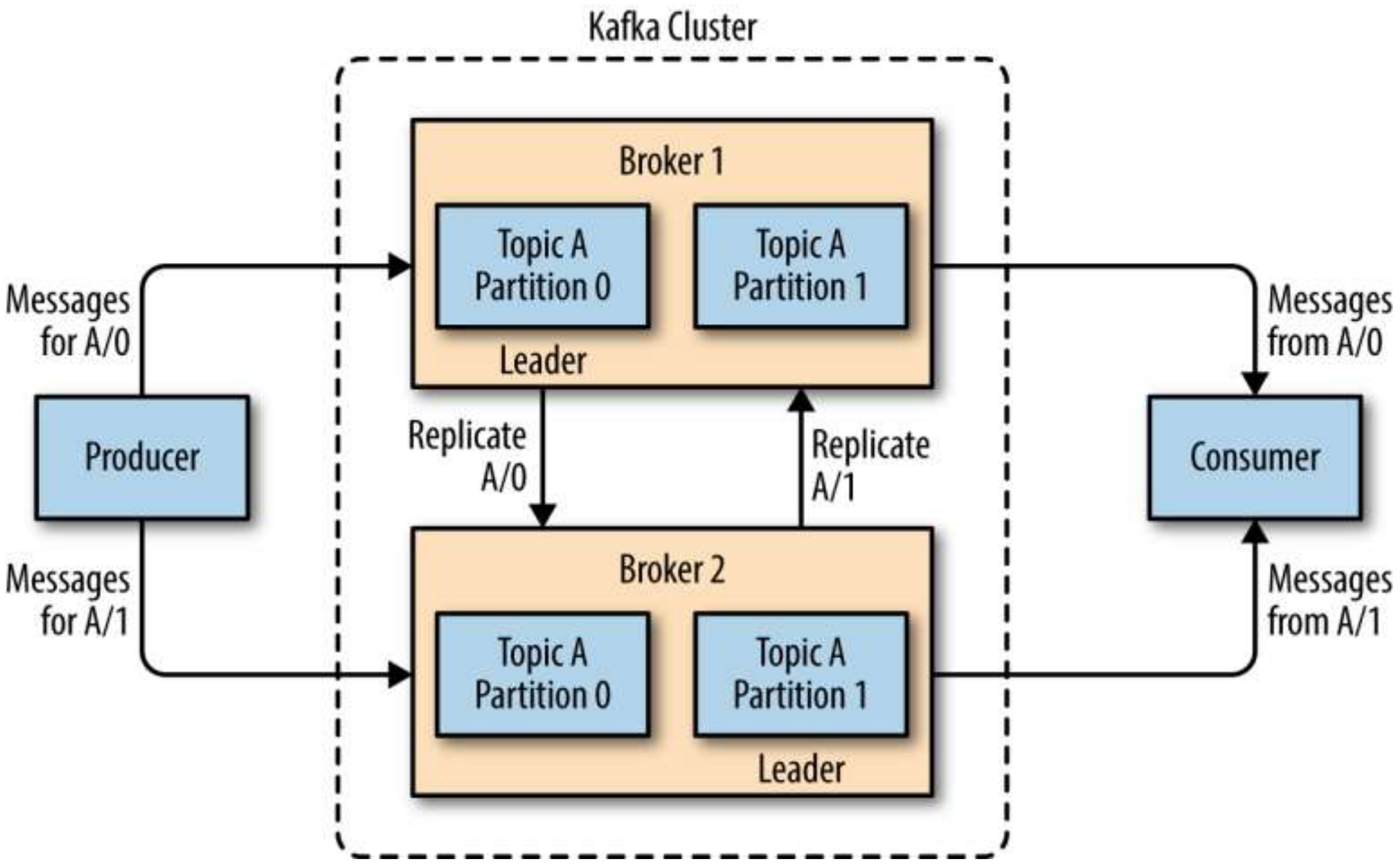
Topics & Partitions

- A topic is where messages accumulate
- Topics are broken down into partitions
- Each partition has a commit log (log ,data structure which record entry of data and exits of data from each partition) associated with it
- Messages are written to a partition in append-only fashion
- Messages are read from a partition from the beginning
- Order of messages are guaranteed within a particular partition, not across all the partitions of the topic
- Since each partition can be hosted on different servers, it accounts for scalability

Topics & Partitions

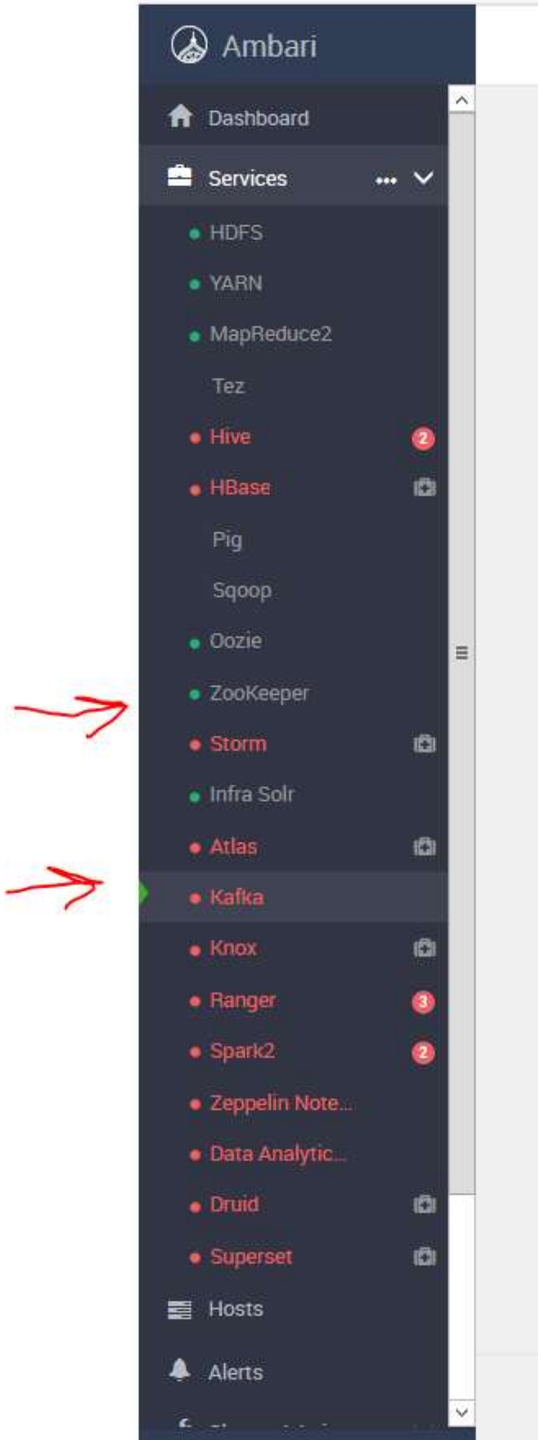


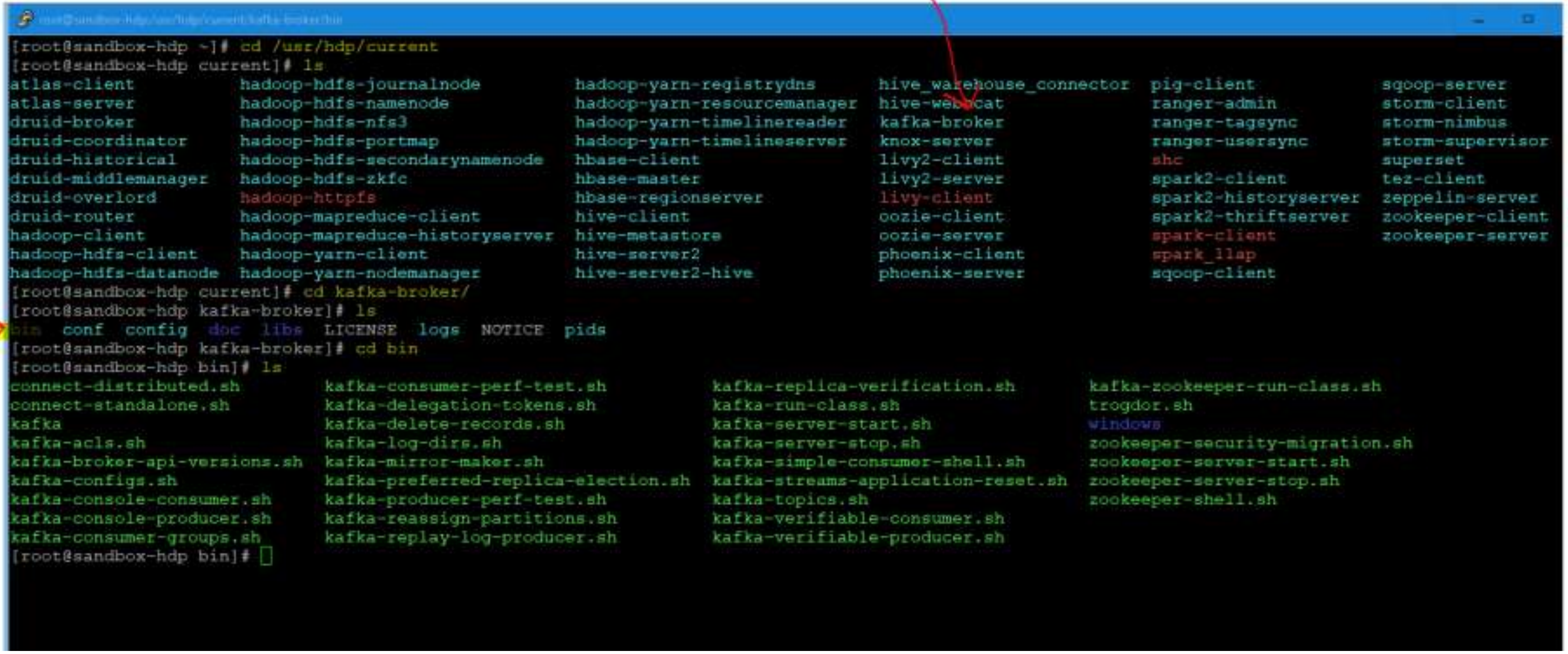
Brokers & Cluster



Kafka Modules

- Producers
- Consumers
- Kafka Connect
- Kafka Streams


Ambari
Dashboard
Services
HDFS
YARN
MapReduce2
Tez
Hive 2
HBase
Pig
Sqoop
Oozie
ZooKeeper
Storm
Infra Solr
Atlas
Kafka
Knox
Ranger 3
Spark2 2
Zeppelin Note...
Data Analytic...
Druid
Superset
Hosts
Alerts

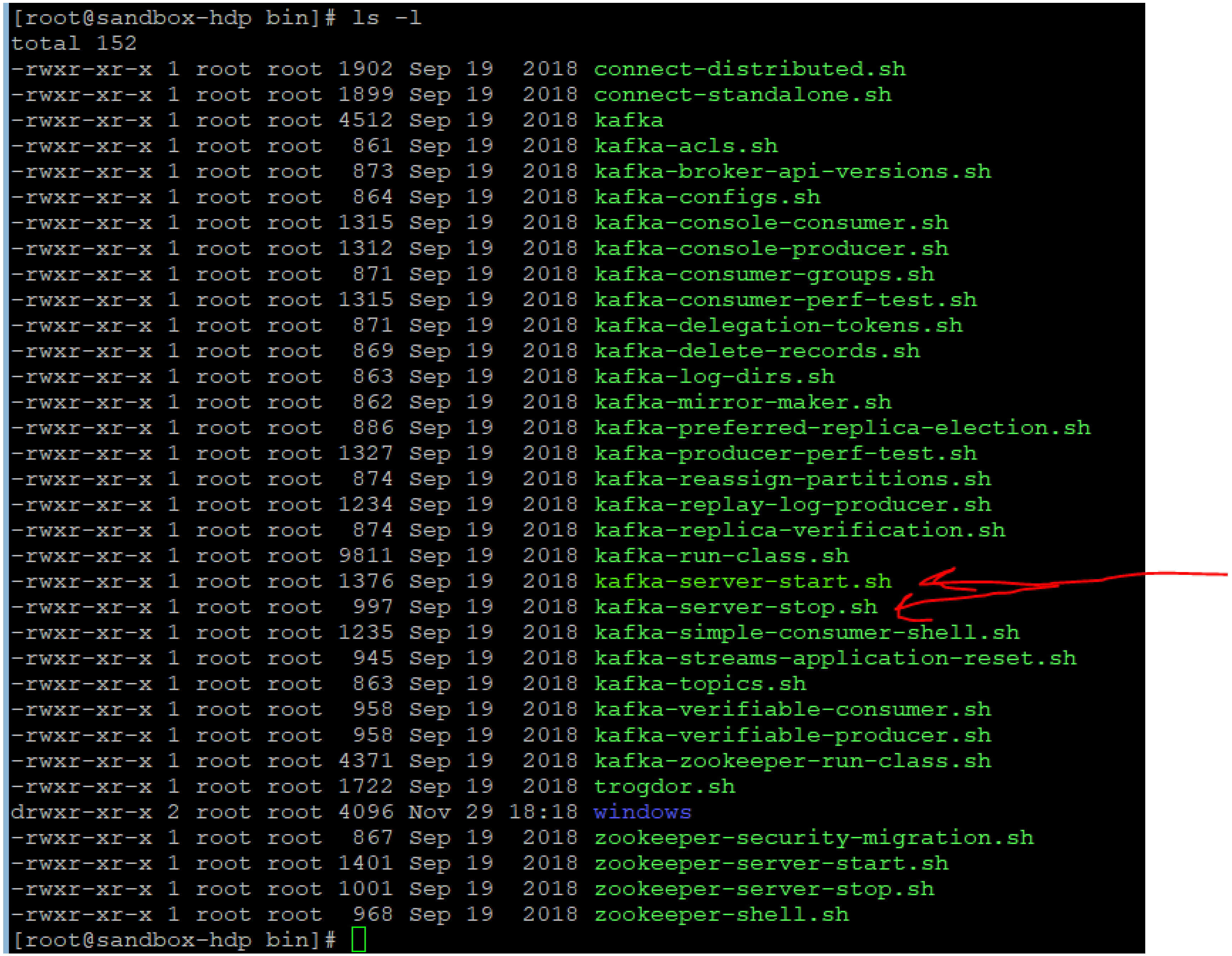


```
[root@sandbox-hdp ~]# cd /usr/hdp/current
[root@sandbox-hdp current]# ls
atlas-client          hadoop-hdfs-journalnode          hadoop-yarn-registrydns         hive_warehouse_connector  pig-client             sqoop-server
atlas-server          hadoop-hdfs-namenode             hadoop-yarn-resourcemanager     hive-webhcat              ranger-admin           storm-client
druid-broker          hadoop-hdfs-nfs3                 hadoop-yarn-timelinereader      kafka-broker              ranger-tagsync         storm-nimbus
druid-coordinator     hadoop-hdfs-portmap              hadoop-yarn-timelineserver      knox-server               ranger-usersync        storm-supervisor
druid-historical      hadoop-hdfs-secondarynamenode    hbase-client                    livy2-client              shc                    superset
druid-middlemanager   hadoop-hdfs-zkfc                 hbase-master                    livy2-server              spark2-client          tez-client
druid-overlord        hadoop-httpfs                    hbase-regionserver              livy-client               spark2-historyserver   zeppelin-server
druid-router          hadoop-mapreduce-client          hive-client                     oozie-client              spark2-thriftserver    zookeeper-client
hadoop-client         hadoop-mapreduce-historyserver   hive-metastore                  oozie-server              spark-client           zookeeper-server
hadoop-hdfs-client    hadoop-yarn-client               hive-server2                    phoenix-client            spark_llap
hadoop-hdfs-datanode  hadoop-yarn-nodemanager          hive-server2-hive               phoenix-server            sqoop-client
[root@sandbox-hdp current]# cd kafka-broker/
[root@sandbox-hdp kafka-broker]# ls
bin  conf  config  doc  libs  LICENSE  logs  NOTICE  pids
[root@sandbox-hdp kafka-broker]# cd bin
[root@sandbox-hdp bin]# ls
connect-distributed.sh         kafka-consumer-perf-test.sh          kafka-replica-verification.sh        kafka-zookeeper-run-class.sh
connect-standalone.sh          kafka-delegation-tokens.sh           kafka-run-class.sh                   trogdor.sh
kafka                          kafka-delete-records.sh              kafka-server-start.sh                windows
kafka-acls.sh                  kafka-log-dirs.sh                    kafka-server-stop.sh                 zookeeper-security-migration.sh
kafka-broker-api-versions.sh   kafka-mirror-maker.sh                kafka-simple-consumer-shell.sh       zookeeper-server-start.sh
kafka-configs.sh               kafka-preferred-replica-election.sh  kafka-streams-application-reset.sh   zookeeper-server-stop.sh
kafka-console-consumer.sh      kafka-producer-perf-test.sh          kafka-topics.sh                      zookeeper-shell.sh
kafka-console-producer.sh      kafka-reassign-partitions.sh         kafka-verifiable-consumer.sh
kafka-consumer-groups.sh       kafka-replay-log-producer.sh         kafka-verifiable-producer.sh
[root@sandbox-hdp bin]# 
```

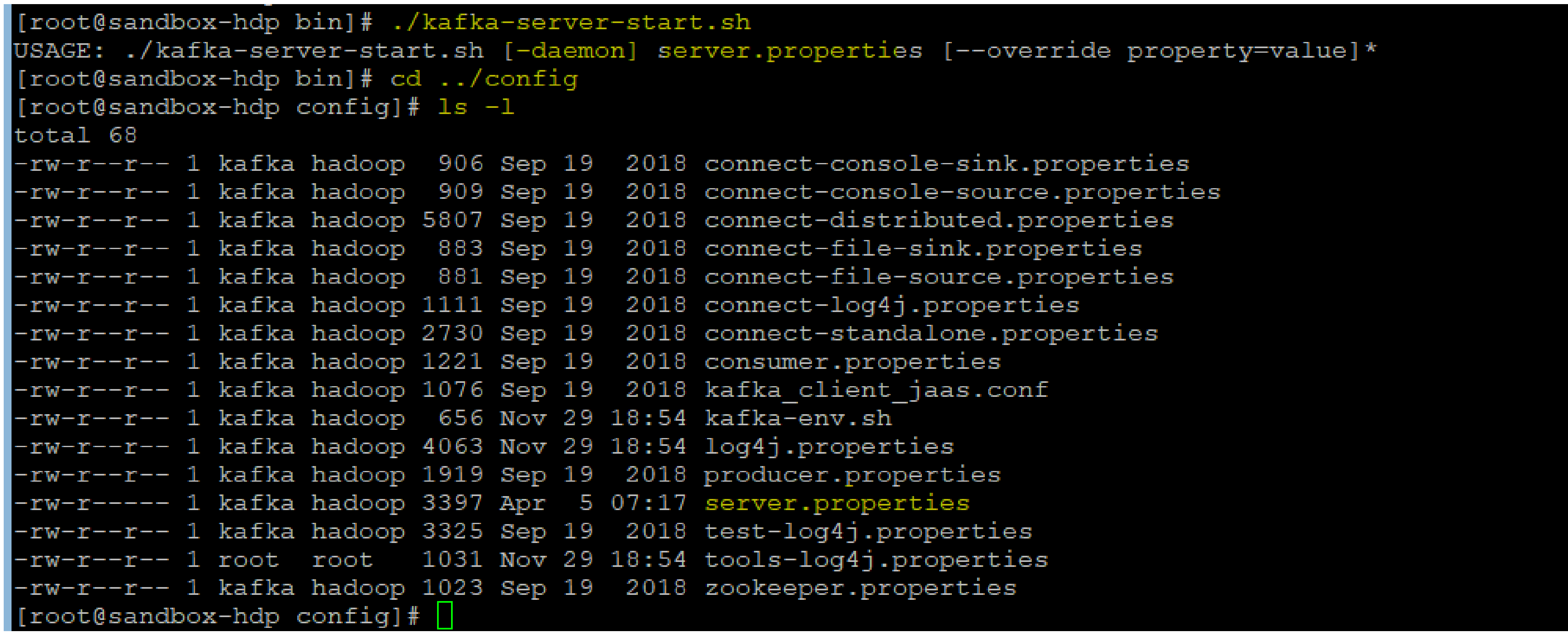
Kafka relays on something calls zookeeper

Zookeeper is distributed configuration management tools and belongs to Hadoop .For example if there is a shared configuration system that your Hadoop ecosystem need it then zookeeper can be your choice which you can store those information.

/usr/hdp/current/kafka-broker
You can start zookeeper from GUI

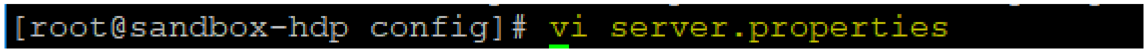


```
[root@sandbox-hdp bin]# ls -l
total 152
-rwxr-xr-x 1 root root 1902 Sep 19  2018 connect-distributed.sh
-rwxr-xr-x 1 root root 1899 Sep 19  2018 connect-standalone.sh
-rwxr-xr-x 1 root root 4512 Sep 19  2018 kafka
-rwxr-xr-x 1 root root  861 Sep 19  2018 kafka-acls.sh
-rwxr-xr-x 1 root root  873 Sep 19  2018 kafka-broker-api-versions.sh
-rwxr-xr-x 1 root root  864 Sep 19  2018 kafka-configs.sh
-rwxr-xr-x 1 root root 1315 Sep 19  2018 kafka-console-consumer.sh
-rwxr-xr-x 1 root root 1312 Sep 19  2018 kafka-console-producer.sh
-rwxr-xr-x 1 root root  871 Sep 19  2018 kafka-consumer-groups.sh
-rwxr-xr-x 1 root root 1315 Sep 19  2018 kafka-consumer-perf-test.sh
-rwxr-xr-x 1 root root  871 Sep 19  2018 kafka-delegation-tokens.sh
-rwxr-xr-x 1 root root  869 Sep 19  2018 kafka-delete-records.sh
-rwxr-xr-x 1 root root  863 Sep 19  2018 kafka-log-dirs.sh
-rwxr-xr-x 1 root root  862 Sep 19  2018 kafka-mirror-maker.sh
-rwxr-xr-x 1 root root  886 Sep 19  2018 kafka-preferred-replica-election.sh
-rwxr-xr-x 1 root root 1327 Sep 19  2018 kafka-producer-perf-test.sh
-rwxr-xr-x 1 root root  874 Sep 19  2018 kafka-reassign-partitions.sh
-rwxr-xr-x 1 root root 1234 Sep 19  2018 kafka-replay-log-producer.sh
-rwxr-xr-x 1 root root  874 Sep 19  2018 kafka-replica-verification.sh
-rwxr-xr-x 1 root root 9811 Sep 19  2018 kafka-run-class.sh
-rwxr-xr-x 1 root root 1376 Sep 19  2018 kafka-server-start.sh
-rwxr-xr-x 1 root root  997 Sep 19  2018 kafka-server-stop.sh
-rwxr-xr-x 1 root root 1235 Sep 19  2018 kafka-simple-consumer-shell.sh
-rwxr-xr-x 1 root root  945 Sep 19  2018 kafka-streams-application-reset.sh
-rwxr-xr-x 1 root root  863 Sep 19  2018 kafka-topics.sh
-rwxr-xr-x 1 root root  958 Sep 19  2018 kafka-verifiable-consumer.sh
-rwxr-xr-x 1 root root  958 Sep 19  2018 kafka-verifiable-producer.sh
-rwxr-xr-x 1 root root 4371 Sep 19  2018 kafka-zookeeper-run-class.sh
-rwxr-xr-x 1 root root 1722 Sep 19  2018 trogdor.sh
drwxr-xr-x 2 root root 4096 Nov 29 18:18 windows
-rwxr-xr-x 1 root root  867 Sep 19  2018 zookeeper-security-migration.sh
-rwxr-xr-x 1 root root 1401 Sep 19  2018 zookeeper-server-start.sh
-rwxr-xr-x 1 root root 1001 Sep 19  2018 zookeeper-server-stop.sh
-rwxr-xr-x 1 root root  968 Sep 19  2018 zookeeper-shell.sh
[root@sandbox-hdp bin]#
```

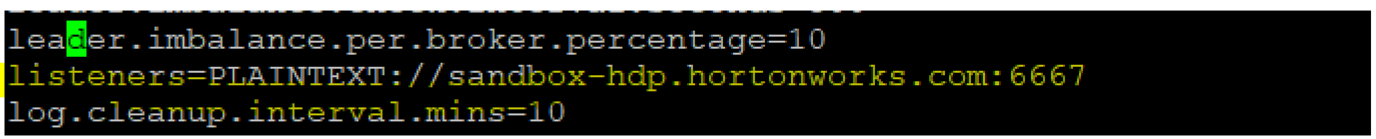


```
[root@sandbox-hdp bin]# ./kafka-server-start.sh
USAGE: ./kafka-server-start.sh [-daemon] server.properties [--override property=value]*
[root@sandbox-hdp bin]# cd ../config
[root@sandbox-hdp config]# ls -l
total 68
-rw-r--r-- 1 kafka hadoop  906 Sep 19  2018 connect-console-sink.properties
-rw-r--r-- 1 kafka hadoop  909 Sep 19  2018 connect-console-source.properties
-rw-r--r-- 1 kafka hadoop 5807 Sep 19  2018 connect-distributed.properties
-rw-r--r-- 1 kafka hadoop  883 Sep 19  2018 connect-file-sink.properties
-rw-r--r-- 1 kafka hadoop  881 Sep 19  2018 connect-file-source.properties
-rw-r--r-- 1 kafka hadoop 1111 Sep 19  2018 connect-log4j.properties
-rw-r--r-- 1 kafka hadoop 2730 Sep 19  2018 connect-standalone.properties
-rw-r--r-- 1 kafka hadoop 1221 Sep 19  2018 consumer.properties
-rw-r--r-- 1 kafka hadoop 1076 Sep 19  2018 kafka_client_jaas.conf
-rw-r--r-- 1 kafka hadoop  656 Nov 29 18:54 kafka-env.sh
-rw-r--r-- 1 kafka hadoop 4063 Nov 29 18:54 log4j.properties
-rw-r--r-- 1 kafka hadoop 1919 Sep 19  2018 producer.properties
-rw-r----- 1 kafka hadoop 3397 Apr  5 07:17 server.properties
-rw-r--r-- 1 kafka hadoop 3325 Sep 19  2018 test-log4j.properties
-rw-r--r-- 1 root  root   1031 Nov 29 18:54 tools-log4j.properties
-rw-r--r-- 1 kafka hadoop 1023 Sep 19  2018 zookeeper.properties
[root@sandbox-hdp config]#
```

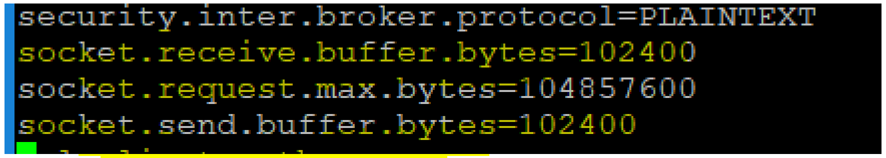
We start Kafka-services we need to specifies few thing like server.properties



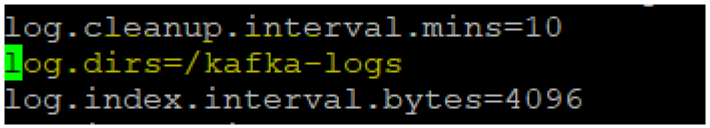
```
[root@sandbox-hdp config]# vi server.properties
```



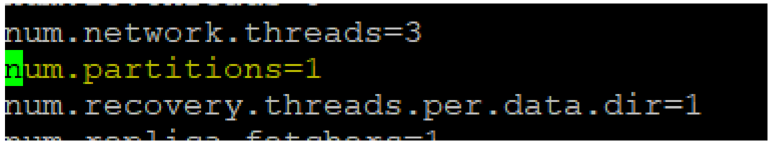
```
leader.imbalance.per.broker.percentage=10
listeners=PLAINTEXT://sandbox-hdp.hortonworks.com:6667
log.cleanup.interval.mins=10
```



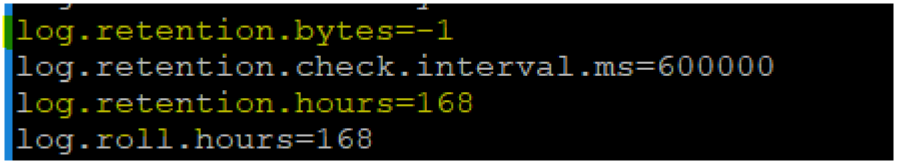
```
security.inter.broker.protocol=PLAINTEXT
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
socket.send.buffer.bytes=102400
```



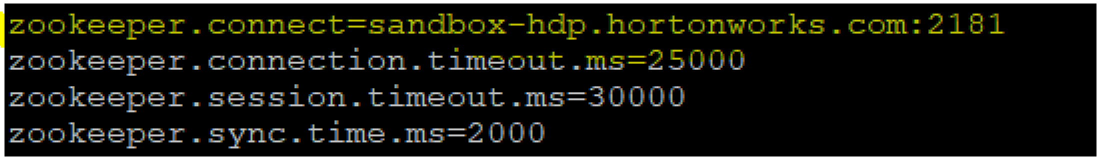
```
log.cleanup.interval.mins=10
log.dirs=/kafka-logs
log.index.interval.bytes=4096
```



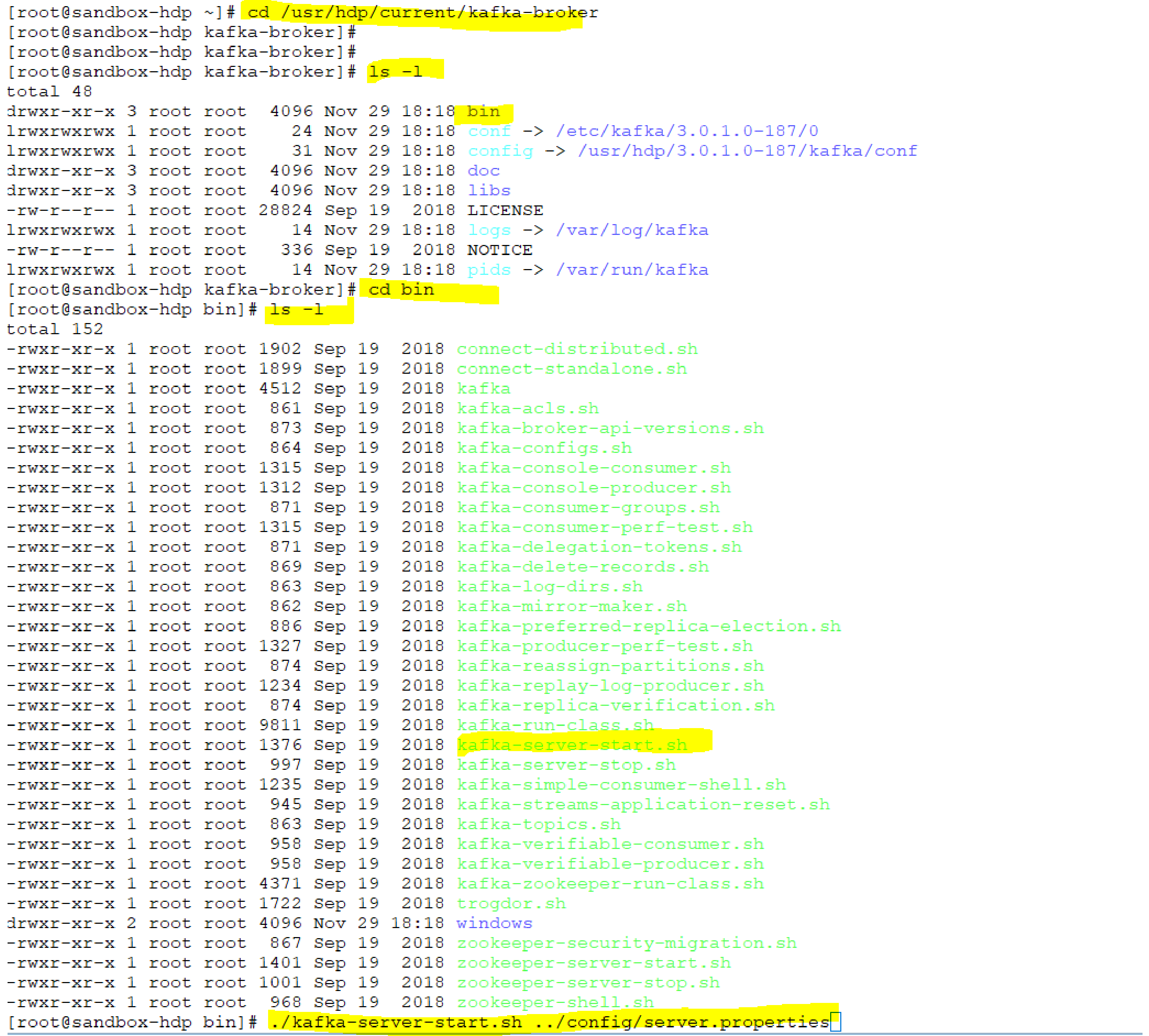
```
num.network.threads=3
num.partitions=1
num.recovery.threads.per.data.dir=1
```



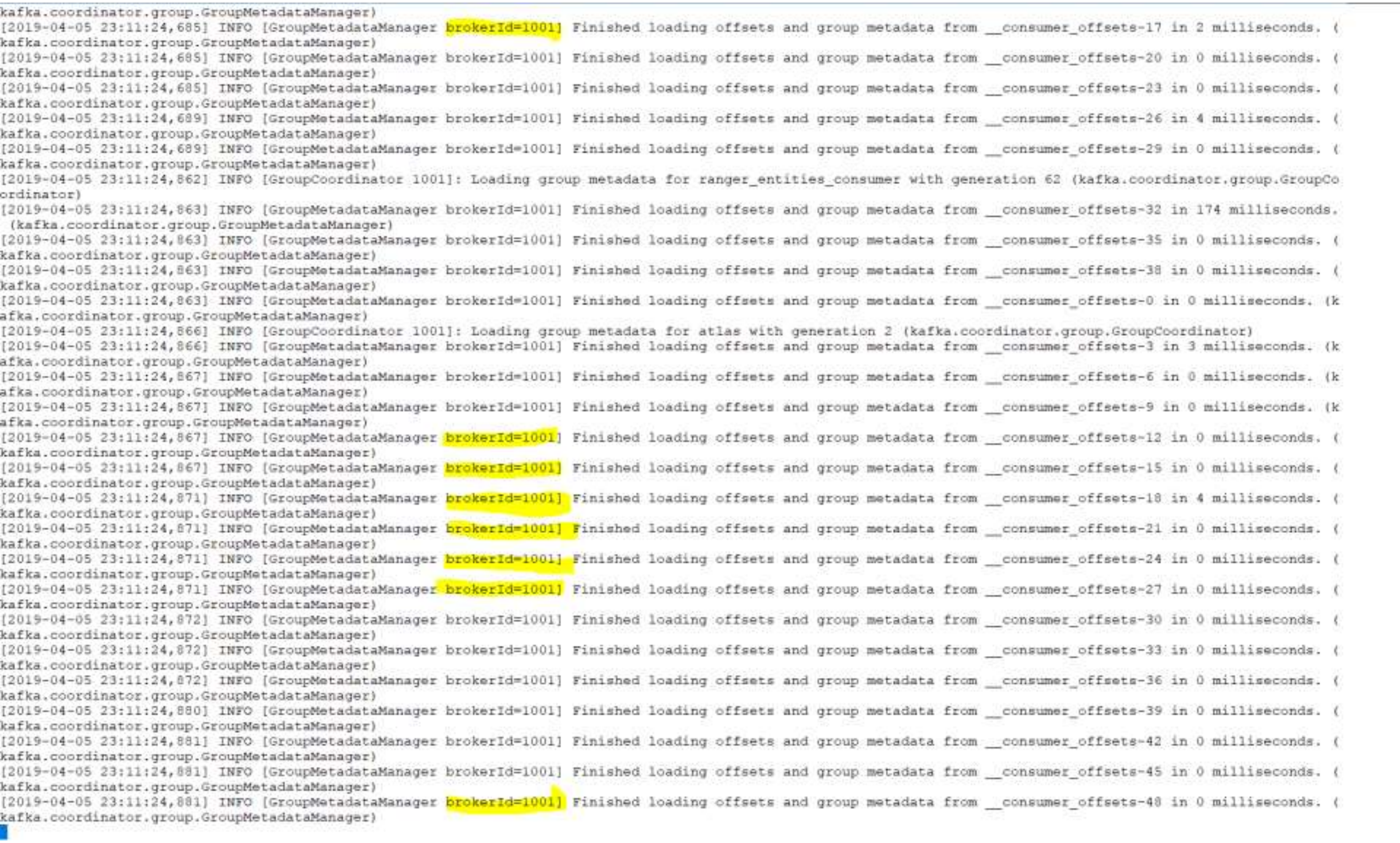
```
log.retention.bytes=-1
log.retention.check.interval.ms=600000
log.retention.hours=168
log.roll.hours=168
```



```
zookeeper.connect=sandbox-hdp.hortonworks.com:2181
zookeeper.connection.timeout.ms=25000
zookeeper.session.timeout.ms=30000
zookeeper.sync.time.ms=2000
```

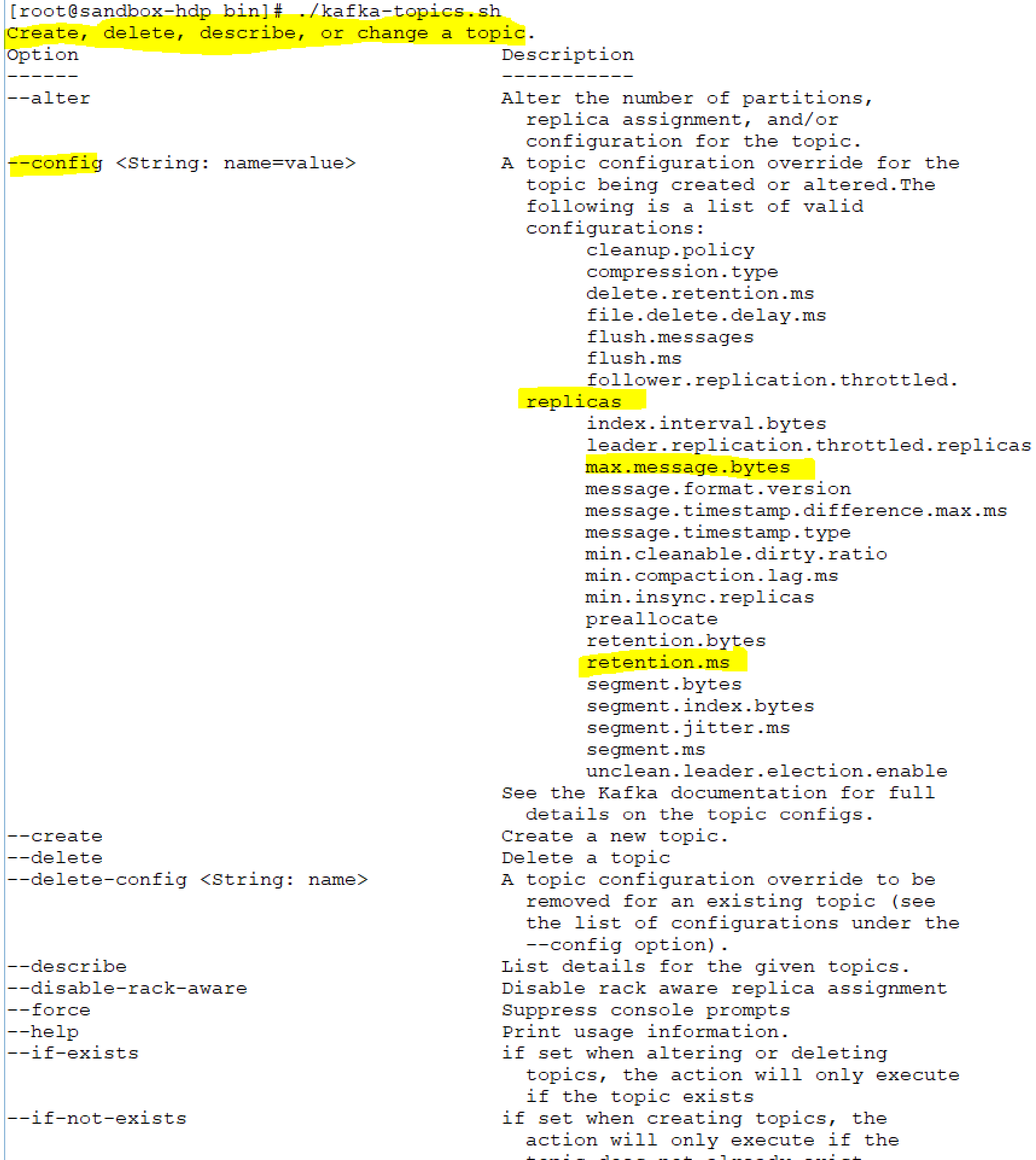


```
[root@sandbox-hdp ~]# cd /usr/hdp/current/kafka-broker
[root@sandbox-hdp kafka-broker]#
[root@sandbox-hdp kafka-broker]#
[root@sandbox-hdp kafka-broker]# ls -l
total 48
drwxr-xr-x 3 root root  4096 Nov 29 18:18 bin
lrwxrwxrwx 1 root root    24 Nov 29 18:18 conf -> /etc/kafka/3.0.1.0-187/0
lrwxrwxrwx 1 root root    31 Nov 29 18:18 config -> /usr/hdp/3.0.1.0-187/kafka/conf
drwxr-xr-x 3 root root  4096 Nov 29 18:18 doc
drwxr-xr-x 3 root root  4096 Nov 29 18:18 libs
-rw-r--r-- 1 root root 28824 Sep 19  2018 LICENSE
lrwxrwxrwx 1 root root    14 Nov 29 18:18 logs -> /var/log/kafka
-rw-r--r-- 1 root root   336 Sep 19  2018 NOTICE
lrwxrwxrwx 1 root root    14 Nov 29 18:18 pids -> /var/run/kafka
[root@sandbox-hdp kafka-broker]# cd bin
[root@sandbox-hdp bin]# ls -l
total 152
-rwxr-xr-x 1 root root 1902 Sep 19  2018 connect-distributed.sh
-rwxr-xr-x 1 root root 1899 Sep 19  2018 connect-standalone.sh
-rwxr-xr-x 1 root root 4512 Sep 19  2018 kafka
-rwxr-xr-x 1 root root  861 Sep 19  2018 kafka-acls.sh
-rwxr-xr-x 1 root root  873 Sep 19  2018 kafka-broker-api-versions.sh
-rwxr-xr-x 1 root root  864 Sep 19  2018 kafka-configs.sh
-rwxr-xr-x 1 root root 1315 Sep 19  2018 kafka-console-consumer.sh
-rwxr-xr-x 1 root root 1312 Sep 19  2018 kafka-console-producer.sh
-rwxr-xr-x 1 root root  871 Sep 19  2018 kafka-consumer-groups.sh
-rwxr-xr-x 1 root root 1315 Sep 19  2018 kafka-consumer-perf-test.sh
-rwxr-xr-x 1 root root  871 Sep 19  2018 kafka-delegation-tokens.sh
-rwxr-xr-x 1 root root  869 Sep 19  2018 kafka-delete-records.sh
-rwxr-xr-x 1 root root  863 Sep 19  2018 kafka-log-dirs.sh
-rwxr-xr-x 1 root root  862 Sep 19  2018 kafka-mirror-maker.sh
-rwxr-xr-x 1 root root  886 Sep 19  2018 kafka-preferred-replica-election.sh
-rwxr-xr-x 1 root root 1327 Sep 19  2018 kafka-producer-perf-test.sh
-rwxr-xr-x 1 root root  874 Sep 19  2018 kafka-reassign-partitions.sh
-rwxr-xr-x 1 root root 1234 Sep 19  2018 kafka-replay-log-producer.sh
-rwxr-xr-x 1 root root  874 Sep 19  2018 kafka-replica-verification.sh
-rwxr-xr-x 1 root root 9811 Sep 19  2018 kafka-run-class.sh
-rwxr-xr-x 1 root root 1376 Sep 19  2018 kafka-server-start.sh
-rwxr-xr-x 1 root root  997 Sep 19  2018 kafka-server-stop.sh
-rwxr-xr-x 1 root root 1235 Sep 19  2018 kafka-simple-consumer-shell.sh
-rwxr-xr-x 1 root root  945 Sep 19  2018 kafka-streams-application-reset.sh
-rwxr-xr-x 1 root root  863 Sep 19  2018 kafka-topics.sh
-rwxr-xr-x 1 root root  958 Sep 19  2018 kafka-verifiable-consumer.sh
-rwxr-xr-x 1 root root  958 Sep 19  2018 kafka-verifiable-producer.sh
-rwxr-xr-x 1 root root 4371 Sep 19  2018 kafka-zookeeper-run-class.sh
-rwxr-xr-x 1 root root 1722 Sep 19  2018 trogdor.sh
drwxr-xr-x 2 root root 4096 Nov 29 18:18 windows
-rwxr-xr-x 1 root root  867 Sep 19  2018 zookeeper-security-migration.sh
-rwxr-xr-x 1 root root 1401 Sep 19  2018 zookeeper-server-start.sh
-rwxr-xr-x 1 root root 1001 Sep 19  2018 zookeeper-server-stop.sh
-rwxr-xr-x 1 root root  968 Sep 19  2018 zookeeper-shell.sh
[root@sandbox-hdp bin]# ./kafka-server-start.sh ../config/server.properties
```

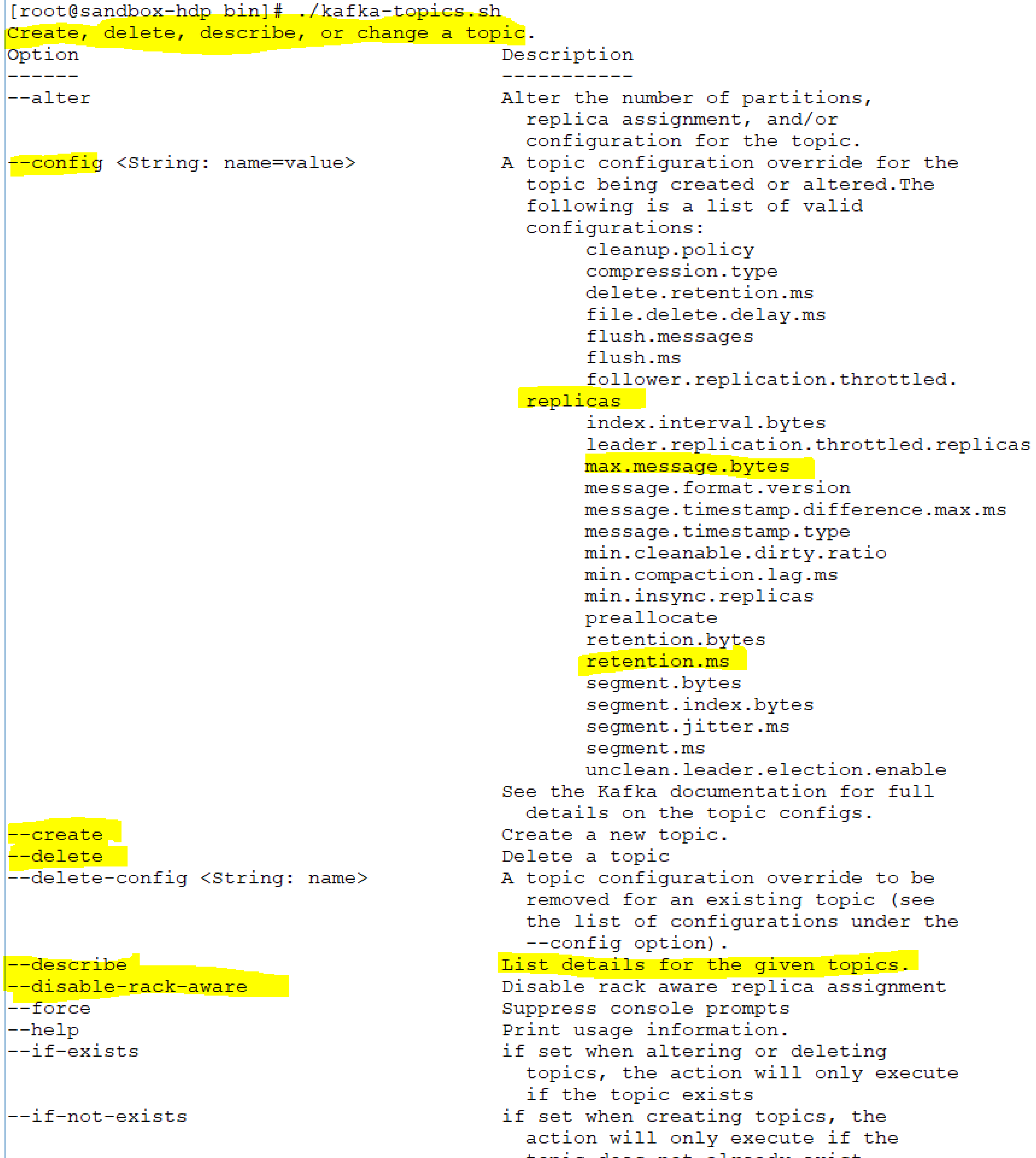


```
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,685] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-17 in 2 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,685] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-20 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,685] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-23 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,689] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-26 in 4 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,689] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-29 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,862] INFO [GroupCoordinator 1001]: Loading group metadata for ranger_entities_consumer with generation 62 (kafka.coordinator.group.GroupCo
ordinator)
[2019-04-05 23:11:24,863] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-32 in 174 milliseconds.
 (kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,863] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-35 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,863] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-38 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,863] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-0 in 0 milliseconds. (k
afka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,866] INFO [GroupCoordinator 1001]: Loading group metadata for atlas with generation 2 (kafka.coordinator.group.GroupCoordinator)
[2019-04-05 23:11:24,866] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-3 in 3 milliseconds. (k
afka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,867] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-6 in 0 milliseconds. (k
afka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,867] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-9 in 0 milliseconds. (k
afka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,867] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-12 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,867] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-15 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,871] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-18 in 4 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,871] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-21 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,871] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-24 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,871] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-27 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,872] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-30 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,872] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-33 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,872] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-36 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,880] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-39 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,881] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-42 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,881] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-45 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
[2019-04-05 23:11:24,881] INFO [GroupMetadataManager brokerId=1001] Finished loading offsets and group metadata from __consumer_offsets-48 in 0 milliseconds. (
kafka.coordinator.group.GroupMetadataManager)
```

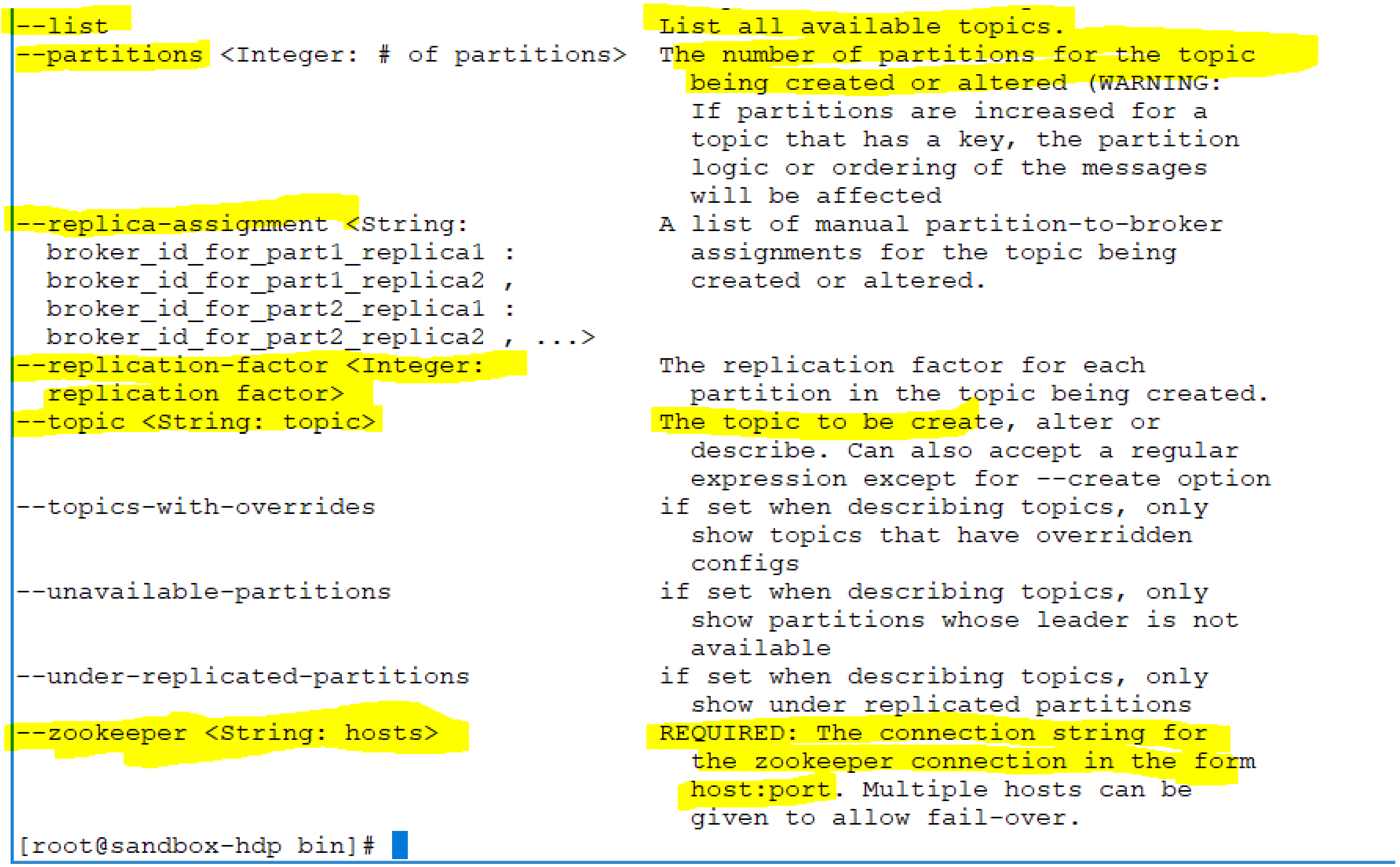
In my physical machine I am running Kafka cluster with a single broker.



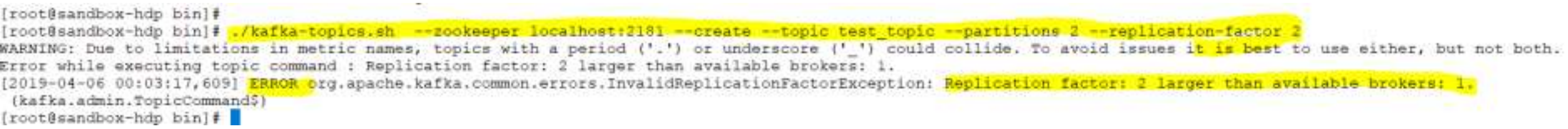
```
[root@sandbox-hdp bin]# ./kafka-topics.sh
Create, delete, describe, or change a topic.
Option                                  Description
------                                  -----------
--alter                                 Alter the number of partitions,
                                          replica assignment, and/or
                                          configuration for the topic.
--config <String: name=value>           A topic configuration override for the
                                          topic being created or altered.The
                                          following is a list of valid
                                          configurations:
                                                cleanup.policy
                                                compression.type
                                                delete.retention.ms
                                                file.delete.delay.ms
                                                flush.messages
                                                flush.ms
                                                follower.replication.throttled.
                                          replicas
                                                index.interval.bytes
                                                leader.replication.throttled.replicas
                                                max.message.bytes
                                                message.format.version
                                                message.timestamp.difference.max.ms
                                                message.timestamp.type
                                                min.cleanable.dirty.ratio
                                                min.compaction.lag.ms
                                                min.insync.replicas
                                                preallocate
                                                retention.bytes
                                                retention.ms
                                                segment.bytes
                                                segment.index.bytes
                                                segment.jitter.ms
                                                segment.ms
                                                unclean.leader.election.enable
                                        See the Kafka documentation for full
                                          details on the topic configs.
--create                                Create a new topic.
--delete                                Delete a topic
--delete-config <String: name>          A topic configuration override to be
                                          removed for an existing topic (see
                                          the list of configurations under the
                                          --config option).
--describe                              List details for the given topics.
--disable-rack-aware                    Disable rack aware replica assignment
--force                                 Suppress console prompts
--help                                  Print usage information.
--if-exists                             if set when altering or deleting
                                          topics, the action will only execute
                                          if the topic exists
--if-not-exists                         if set when creating topics, the
                                          action will only execute if the
```



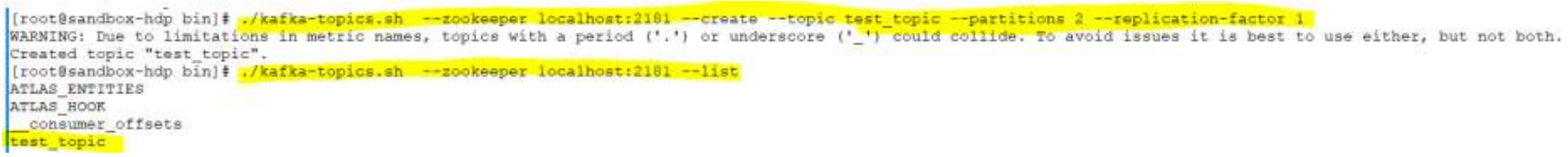
```
[root@sandbox-hdp bin]# ./kafka-topics.sh
Create, delete, describe, or change a topic.
Option                                   Description
------                                   -----------
--alter                                  Alter the number of partitions,
                                           replica assignment, and/or
                                           configuration for the topic.
--config <String: name=value>            A topic configuration override for the
                                           topic being created or altered.The
                                           following is a list of valid
                                           configurations:
                                                cleanup.policy
                                                compression.type
                                                delete.retention.ms
                                                file.delete.delay.ms
                                                flush.messages
                                                flush.ms
                                                follower.replication.throttled.
                                           replicas
                                                index.interval.bytes
                                                leader.replication.throttled.replicas
                                                max.message.bytes
                                                message.format.version
                                                message.timestamp.difference.max.ms
                                                message.timestamp.type
                                                min.cleanable.dirty.ratio
                                                min.compaction.lag.ms
                                                min.insync.replicas
                                                preallocate
                                                retention.bytes
                                                retention.ms
                                                segment.bytes
                                                segment.index.bytes
                                                segment.jitter.ms
                                                segment.ms
                                                unclean.leader.election.enable
                                         See the Kafka documentation for full
                                           details on the topic configs.
--create                                 Create a new topic.
--delete                                 Delete a topic
--delete-config <String: name>           A topic configuration override to be
                                           removed for an existing topic (see
                                           the list of configurations under the
                                           --config option).
--describe                               List details for the given topics.
--disable-rack-aware                     Disable rack aware replica assignment
--force                                  Suppress console prompts
--help                                   Print usage information.
--if-exists                              if set when altering or deleting
                                           topics, the action will only execute
                                           if the topic exists
--if-not-exists                          if set when creating topics, the
                                           action will only execute if the
```



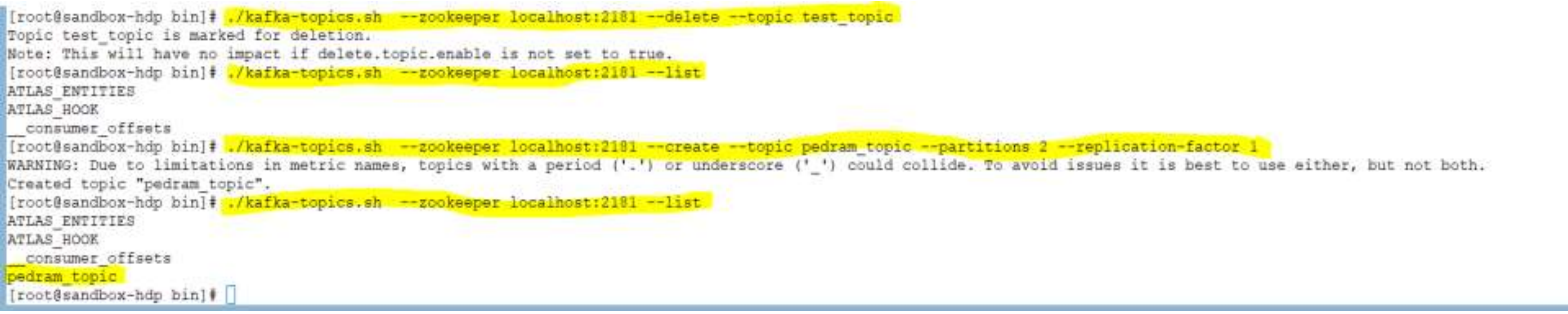
```
--list                                   List all available topics.
--partitions <Integer: # of partitions>  The number of partitions for the topic
                                           being created or altered (WARNING:
                                           If partitions are increased for a
                                           topic that has a key, the partition
                                           logic or ordering of the messages
                                           will be affected
--replica-assignment <String:            A list of manual partition-to-broker
  broker_id_for_part1_replica1 :           assignments for the topic being
  broker_id_for_part1_replica2 ,           created or altered.
  broker_id_for_part2_replica1 :
  broker_id_for_part2_replica2 , ...>
--replication-factor <Integer:           The replication factor for each
  replication factor>                      partition in the topic being created.
--topic <String: topic>                  The topic to be create, alter or
                                           describe. Can also accept a regular
                                           expression except for --create option
--topics-with-overrides                  if set when describing topics, only
                                           show topics that have overridden
                                           configs
--unavailable-partitions                 if set when describing topics, only
                                           show partitions whose leader is not
                                           available
--under-replicated-partitions            if set when describing topics, only
                                           show under replicated partitions
--zookeeper <String: hosts>              REQUIRED: The connection string for
                                           the zookeeper connection in the form
                                           host:port. Multiple hosts can be
                                           given to allow fail-over.
[root@sandbox-hdp bin]#
```



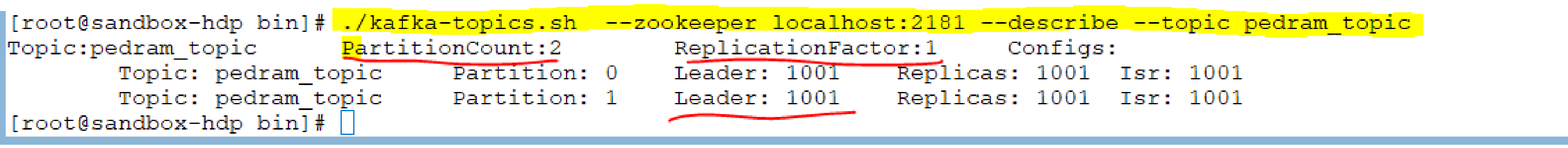
```
[root@sandbox-hdp bin]#
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --create --topic test_topic --partitions 2 --replication-factor 2
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Error while executing topic command : Replication factor: 2 larger than available brokers: 1.
[2019-04-06 00:03:17,609] ERROR org.apache.kafka.common.errors.InvalidReplicationFactorException: Replication factor: 2 larger than available brokers: 1.
 (kafka.admin.TopicCommand$)
[root@sandbox-hdp bin]#
```



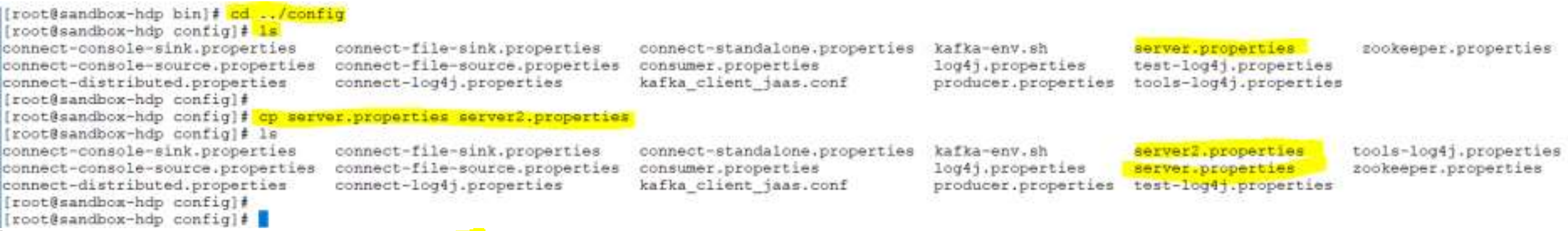
```
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --create --topic test_topic --partitions 2 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic "test_topic".
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --list
ATLAS_ENTITIES
ATLAS_HOOK
__consumer_offsets
test_topic
```



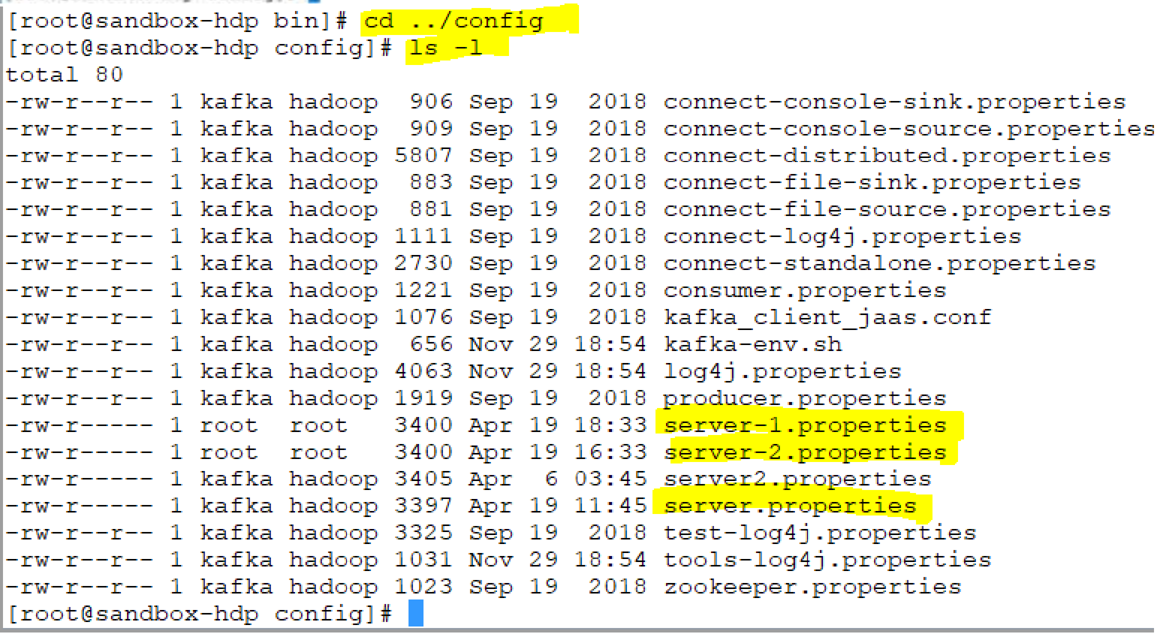
```
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --delete --topic test_topic
Topic test_topic is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --list
ATLAS_ENTITIES
ATLAS_HOOK
__consumer_offsets
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --create --topic pedram_topic --partitions 2 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic "pedram_topic".
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --list
ATLAS_ENTITIES
ATLAS_HOOK
__consumer_offsets
pedram_topic
[root@sandbox-hdp bin]#
```



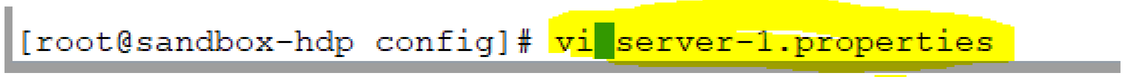
```
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:1     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1001    Replicas: 1001  Isr: 1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001  Isr: 1001
[root@sandbox-hdp bin]#
```



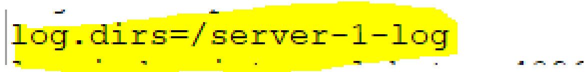
```
[root@sandbox-hdp bin]# cd ../config
[root@sandbox-hdp config]# ls
connect-console-sink.properties    connect-file-sink.properties    connect-standalone.properties  kafka-env.sh          server.properties        zookeeper.properties
connect-console-source.properties  connect-file-source.properties  consumer.properties            log4j.properties      test-log4j.properties
connect-distributed.properties     connect-log4j.properties        kafka_client_jaas.conf         producer.properties   tools-log4j.properties
[root@sandbox-hdp config]#
[root@sandbox-hdp config]# cp server.properties server2.properties
[root@sandbox-hdp config]# ls
connect-console-sink.properties    connect-file-sink.properties    connect-standalone.properties  kafka-env.sh          server2.properties      tools-log4j.properties
connect-console-source.properties  connect-file-source.properties  consumer.properties            log4j.properties      server.properties       zookeeper.properties
connect-distributed.properties     connect-log4j.properties        kafka_client_jaas.conf         producer.properties   test-log4j.properties
[root@sandbox-hdp config]#
[root@sandbox-hdp config]#
```



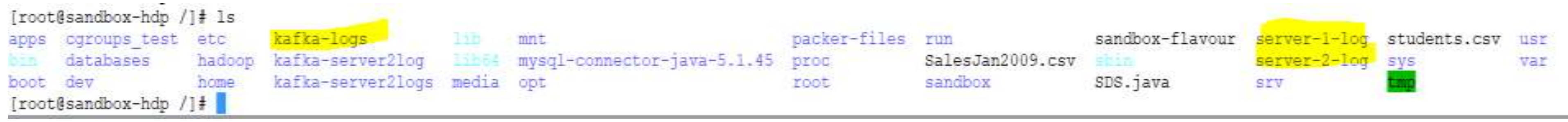
```
[root@sandbox-hdp bin]# cd ../config
[root@sandbox-hdp config]# ls -l
total 80
-rw-r--r-- 1 kafka hadoop  906 Sep 19  2018 connect-console-sink.properties
-rw-r--r-- 1 kafka hadoop  909 Sep 19  2018 connect-console-source.properties
-rw-r--r-- 1 kafka hadoop 5807 Sep 19  2018 connect-distributed.properties
-rw-r--r-- 1 kafka hadoop  883 Sep 19  2018 connect-file-sink.properties
-rw-r--r-- 1 kafka hadoop  881 Sep 19  2018 connect-file-source.properties
-rw-r--r-- 1 kafka hadoop 1111 Sep 19  2018 connect-log4j.properties
-rw-r--r-- 1 kafka hadoop 2730 Sep 19  2018 connect-standalone.properties
-rw-r--r-- 1 kafka hadoop 1221 Sep 19  2018 consumer.properties
-rw-r--r-- 1 kafka hadoop 1076 Sep 19  2018 kafka_client_jaas.conf
-rw-r--r-- 1 kafka hadoop  656 Nov 29 18:54 kafka-env.sh
-rw-r--r-- 1 kafka hadoop 4063 Nov 29 18:54 log4j.properties
-rw-r--r-- 1 kafka hadoop 1919 Sep 19  2018 producer.properties
-rw-r----- 1 root  root   3400 Apr 19 18:33 server-1.properties
-rw-r----- 1 root  root   3400 Apr 19 16:33 server-2.properties
-rw-r----- 1 kafka hadoop 3405 Apr  6 03:45 server2.properties
-rw-r----- 1 kafka hadoop 3397 Apr 19 11:45 server.properties
-rw-r--r-- 1 kafka hadoop 3325 Sep 19  2018 test-log4j.properties
-rw-r--r-- 1 kafka hadoop 1031 Nov 29 18:54 tools-log4j.properties
-rw-r--r-- 1 kafka hadoop 1023 Sep 19  2018 zookeeper.properties
[root@sandbox-hdp config]#
```



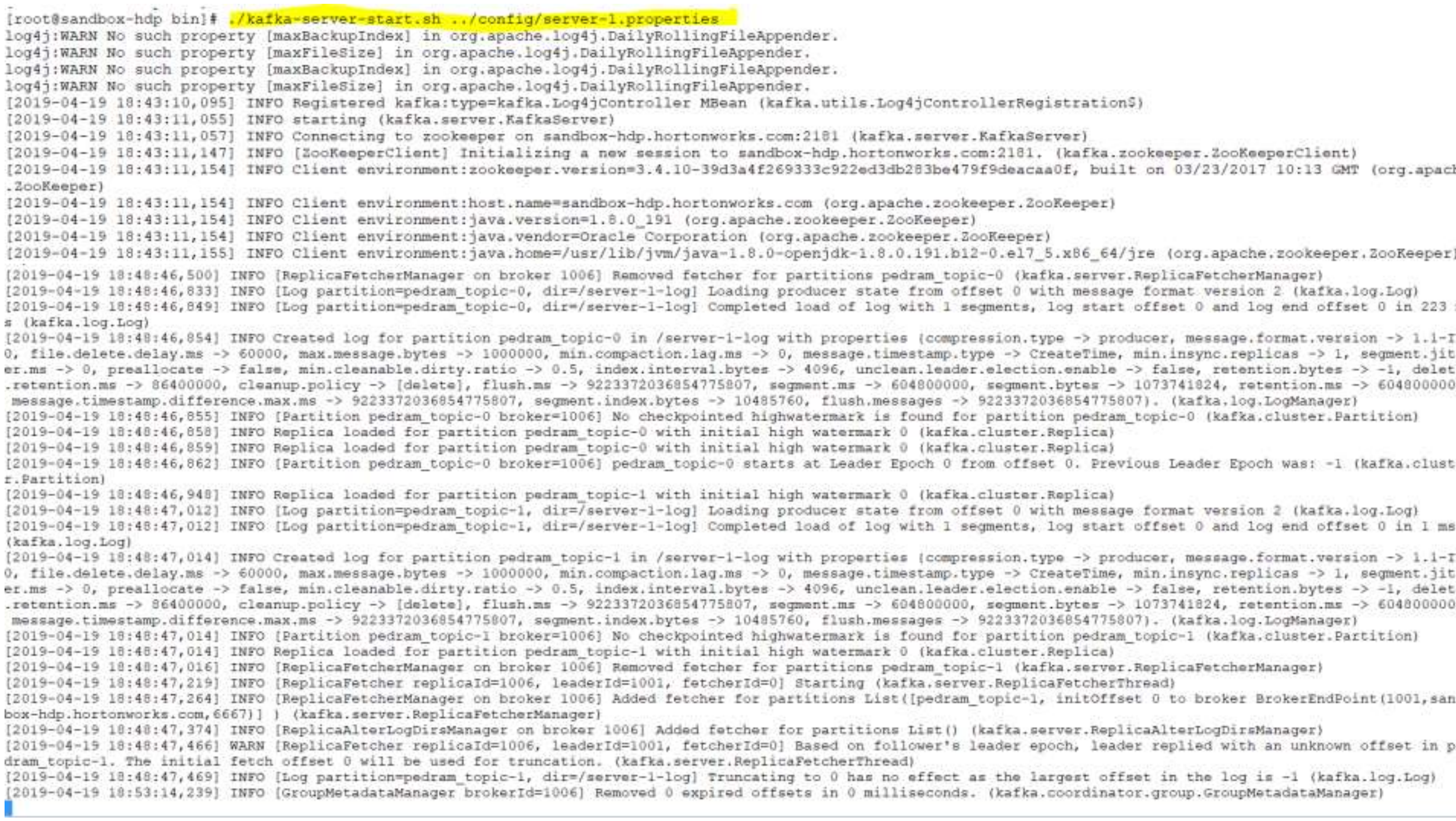
```
[root@sandbox-hdp config]# vi server-1.properties
```



```
log.dirs=/server-1-log
```

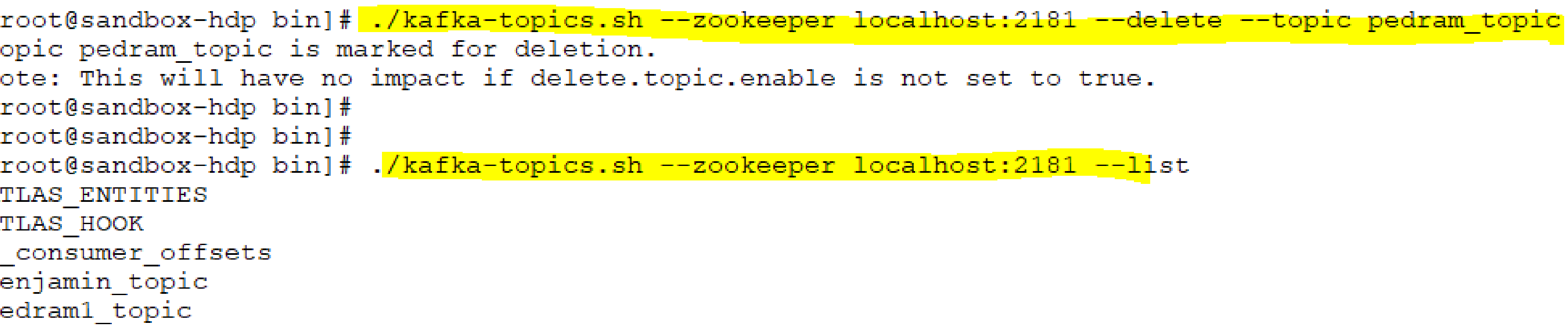


```
[root@sandbox-hdp /]# ls
apps  cgroups_test  etc     kafka-logs         lib    mnt                           packer-files  run               sandbox-flavour  server-1-log  students.csv  usr
bin   databases     hadoop  kafka-server2log   lib64  mysql-connector-java-5.1.45   proc          SalesJan2009.csv  sbin             server-2-log  sys           var
boot  dev           home    kafka-server2logs  media  opt                           root          sandbox           SDS.java         srv           tmp
[root@sandbox-hdp /]#
```

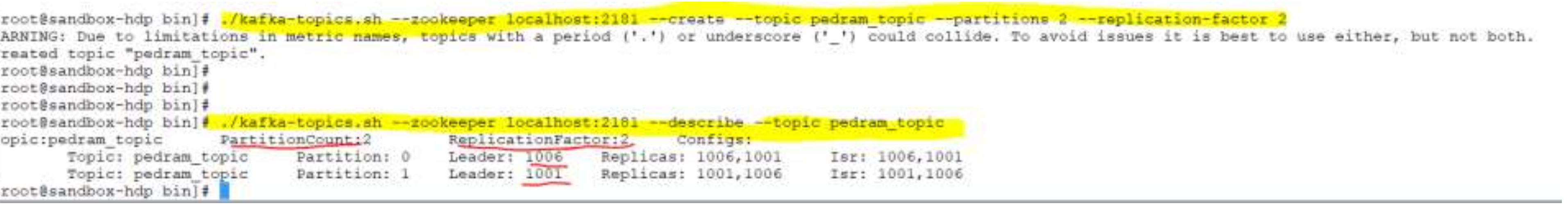


```
[root@sandbox-hdp bin]# ./kafka-server-start.sh ../config/server-1.properties
log4j:WARN No such property [maxBackupIndex] in org.apache.log4j.DailyRollingFileAppender.
log4j:WARN No such property [maxFileSize] in org.apache.log4j.DailyRollingFileAppender.
log4j:WARN No such property [maxBackupIndex] in org.apache.log4j.DailyRollingFileAppender.
log4j:WARN No such property [maxFileSize] in org.apache.log4j.DailyRollingFileAppender.
[2019-04-19 18:43:10,095] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2019-04-19 18:43:11,055] INFO starting (kafka.server.KafkaServer)
[2019-04-19 18:43:11,057] INFO Connecting to zookeeper on sandbox-hdp.hortonworks.com:2181 (kafka.server.KafkaServer)
[2019-04-19 18:43:11,147] INFO [ZooKeeperClient] Initializing a new session to sandbox-hdp.hortonworks.com:2181. (kafka.zookeeper.ZooKeeperClient)
[2019-04-19 18:43:11,154] INFO Client environment:zookeeper.version=3.4.10-39d3a4f269333c922ed3db283be479f9deacaa0f, built on 03/23/2017 10:13 GMT (org.apach
.ZooKeeper)
[2019-04-19 18:43:11,154] INFO Client environment:host.name=sandbox-hdp.hortonworks.com (org.apache.zookeeper.ZooKeeper)
[2019-04-19 18:43:11,154] INFO Client environment:java.version=1.8.0_191 (org.apache.zookeeper.ZooKeeper)
[2019-04-19 18:43:11,154] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2019-04-19 18:43:11,155] INFO Client environment:java.home=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.191.b12-0.el7_5.x86_64/jre (org.apache.zookeeper.ZooKeeper)
[2019-04-19 18:48:46,500] INFO [ReplicaFetcherManager on broker 1006] Removed fetcher for partitions pedram_topic-0 (kafka.server.ReplicaFetcherManager)
[2019-04-19 18:48:46,833] INFO [Log partition=pedram_topic-0, dir=/server-1-log] Loading producer state from offset 0 with message format version 2 (kafka.log.Log)
[2019-04-19 18:48:46,849] INFO [Log partition=pedram_topic-0, dir=/server-1-log] Completed load of log with 1 segments, log start offset 0 and log end offset 0 in 223
s (kafka.log.Log)
[2019-04-19 18:48:46,854] INFO Created log for partition pedram_topic-0 in /server-1-log with properties {compression.type -> producer, message.format.version -> 1.1-I
0, file.delete.delay.ms -> 60000, max.message.bytes -> 1000000, min.compaction.lag.ms -> 0, message.timestamp.type -> CreateTime, min.insync.replicas -> 1, segment.jit
er.ms -> 0, preallocate -> false, min.cleanable.dirty.ratio -> 0.5, index.interval.bytes -> 4096, unclean.leader.election.enable -> false, retention.bytes -> -1, delet
.retention.ms -> 86400000, cleanup.policy -> [delete], flush.ms -> 9223372036854775807, segment.ms -> 604800000, segment.bytes -> 1073741824, retention.ms -> 604800000
 message.timestamp.difference.max.ms -> 9223372036854775807, segment.index.bytes -> 10485760, flush.messages -> 9223372036854775807}. (kafka.log.LogManager)
[2019-04-19 18:48:46,855] INFO [Partition pedram_topic-0 broker=1006] No checkpointed highwatermark is found for partition pedram_topic-0 (kafka.cluster.Partition)
[2019-04-19 18:48:46,858] INFO Replica loaded for partition pedram_topic-0 with initial high watermark 0 (kafka.cluster.Replica)
[2019-04-19 18:48:46,859] INFO Replica loaded for partition pedram_topic-0 with initial high watermark 0 (kafka.cluster.Replica)
[2019-04-19 18:48:46,862] INFO [Partition pedram_topic-0 broker=1006] pedram_topic-0 starts at Leader Epoch 0 from offset 0. Previous Leader Epoch was: -1 (kafka.clust
r.Partition)
[2019-04-19 18:48:46,948] INFO Replica loaded for partition pedram_topic-1 with initial high watermark 0 (kafka.cluster.Replica)
[2019-04-19 18:48:47,012] INFO [Log partition=pedram_topic-1, dir=/server-1-log] Loading producer state from offset 0 with message format version 2 (kafka.log.Log)
[2019-04-19 18:48:47,012] INFO [Log partition=pedram_topic-1, dir=/server-1-log] Completed load of log with 1 segments, log start offset 0 and log end offset 0 in 1 ms
(kafka.log.Log)
[2019-04-19 18:48:47,014] INFO Created log for partition pedram_topic-1 in /server-1-log with properties {compression.type -> producer, message.format.version -> 1.1-I
0, file.delete.delay.ms -> 60000, max.message.bytes -> 1000000, min.compaction.lag.ms -> 0, message.timestamp.type -> CreateTime, min.insync.replicas -> 1, segment.jit
er.ms -> 0, preallocate -> false, min.cleanable.dirty.ratio -> 0.5, index.interval.bytes -> 4096, unclean.leader.election.enable -> false, retention.bytes -> -1, delet
.retention.ms -> 86400000, cleanup.policy -> [delete], flush.ms -> 9223372036854775807, segment.ms -> 604800000, segment.bytes -> 1073741824, retention.ms -> 604800000
 message.timestamp.difference.max.ms -> 9223372036854775807, segment.index.bytes -> 10485760, flush.messages -> 9223372036854775807}. (kafka.log.LogManager)
[2019-04-19 18:48:47,014] INFO [Partition pedram_topic-1 broker=1006] No checkpointed highwatermark is found for partition pedram_topic-1 (kafka.cluster.Partition)
[2019-04-19 18:48:47,014] INFO Replica loaded for partition pedram_topic-1 with initial high watermark 0 (kafka.cluster.Replica)
[2019-04-19 18:48:47,016] INFO [ReplicaFetcherManager on broker 1006] Removed fetcher for partitions pedram_topic-1 (kafka.server.ReplicaFetcherManager)
[2019-04-19 18:48:47,219] INFO [ReplicaFetcher replicaId=1006, leaderId=1001, fetcherId=0] Starting (kafka.server.ReplicaFetcherThread)
[2019-04-19 18:48:47,264] INFO [ReplicaFetcherManager on broker 1006] Added fetcher for partitions List([pedram_topic-1, initOffset 0 to broker BrokerEndPoint(1001,san
box-hdp.hortonworks.com,6667)] ) (kafka.server.ReplicaFetcherManager)
[2019-04-19 18:48:47,374] INFO [ReplicaAlterLogDirsManager on broker 1006] Added fetcher for partitions List() (kafka.server.ReplicaAlterLogDirsManager)
[2019-04-19 18:48:47,466] WARN [ReplicaFetcher replicaId=1006, leaderId=1001, fetcherId=0] Based on follower's leader epoch, leader replied with an unknown offset in p
dram_topic-1. The initial fetch offset 0 will be used for truncation. (kafka.server.ReplicaFetcherThread)
[2019-04-19 18:48:47,469] INFO [Log partition=pedram_topic-1, dir=/server-1-log] Truncating to 0 has no effect as the largest offset in the log is -1 (kafka.log.Log)
[2019-04-19 18:53:14,239] INFO [GroupMetadataManager brokerId=1006] Removed 0 expired offsets in 0 milliseconds. (kafka.coordinator.group.GroupMetadataManager)
```

Create the second broker in the same machine

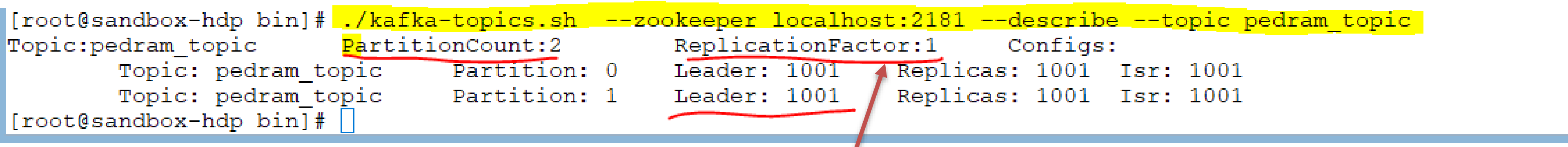


```
root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --delete --topic pedram_topic
opic pedram_topic is marked for deletion.
ote: This will have no impact if delete.topic.enable is not set to true.
root@sandbox-hdp bin]#
root@sandbox-hdp bin]#
root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --list
TLAS_ENTITIES
TLAS_HOOK
_consumer_offsets
enjamin_topic
edram1_topic
```

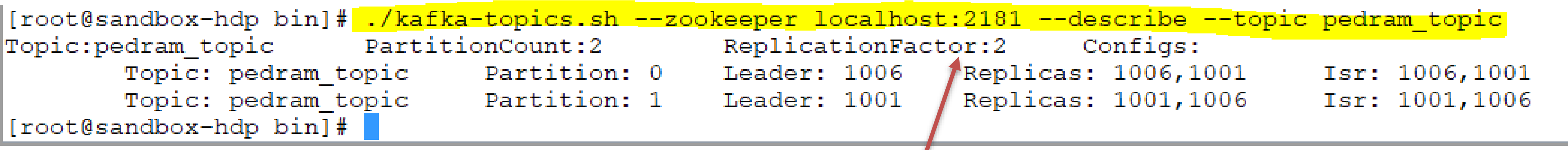


```
root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --create --topic pedram_topic --partitions 2 --replication-factor 2
ARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
reated topic "pedram_topic".
root@sandbox-hdp bin]#
root@sandbox-hdp bin]#
root@sandbox-hdp bin]#
root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
opic:pedram_topic       PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1006    Replicas: 1006,1001     Isr: 1006,1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001,1006
root@sandbox-hdp bin]#
```

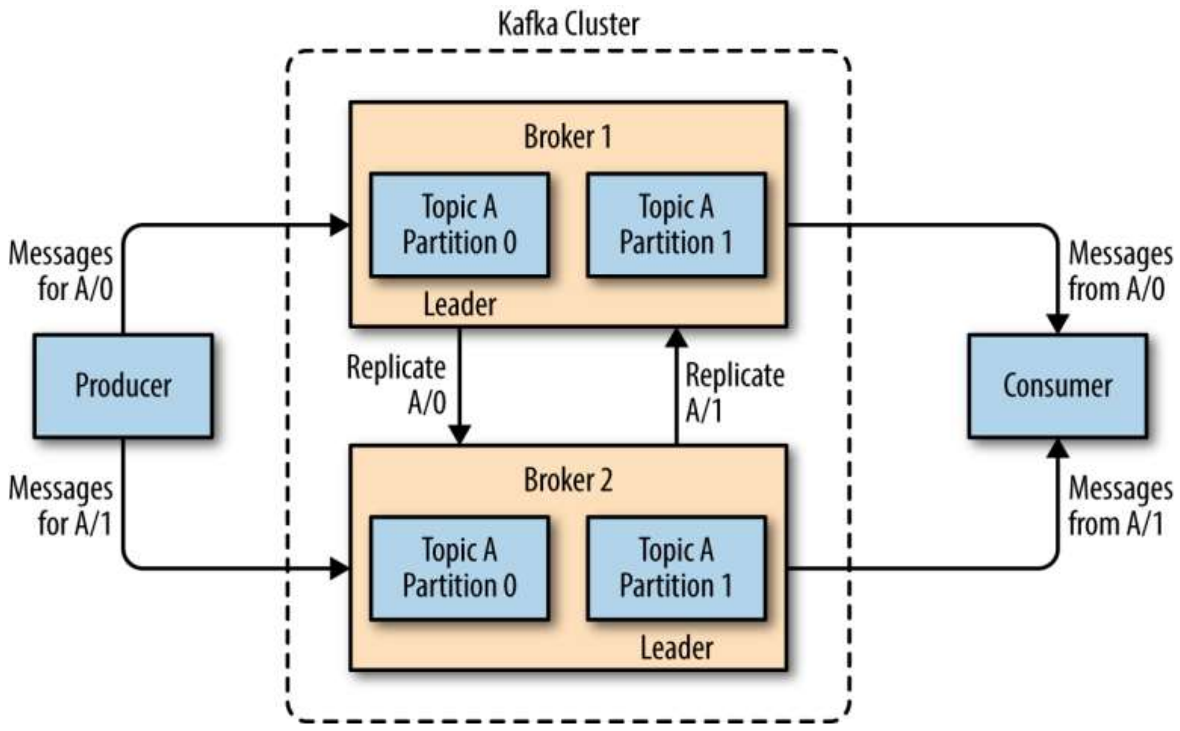
We delete the previous topic and we create again as the replication facto is different now.



```
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:1     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1001    Replicas: 1001  Isr: 1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001  Isr: 1001
[root@sandbox-hdp bin]#
```



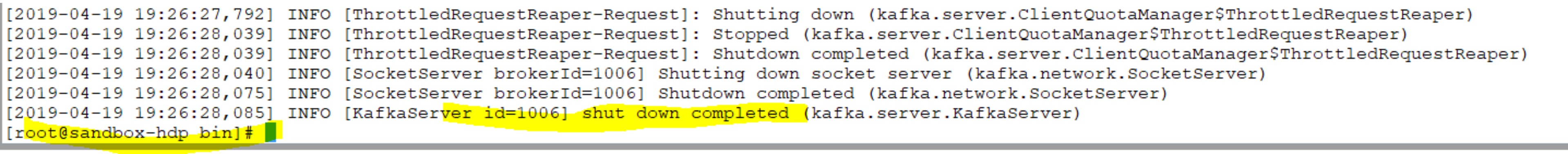
```
[root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1006    Replicas: 1006,1001     Isr: 1006,1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001,1006
[root@sandbox-hdp bin]#
```



./kafka-server-stop.sh ../config/server.properties

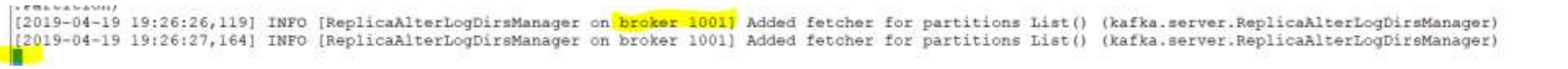
As both brokers are in the same machine running this command will stop both of them so to avoid this , I just use **ctrl+c**

broker 1006

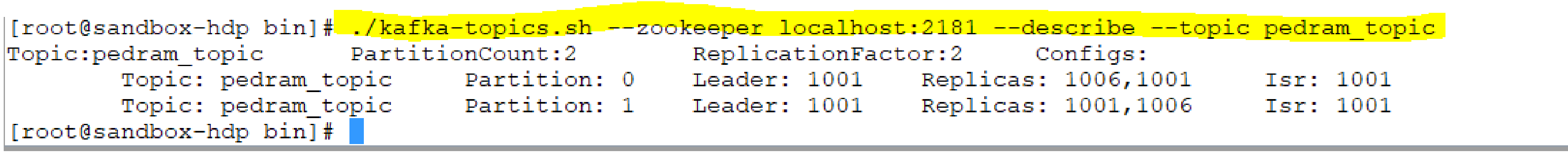


```
[2019-04-19 19:26:27,792] INFO [ThrottledRequestReaper-Request]: Shutting down (kafka.server.ClientQuotaManager$ThrottledRequestReaper)
[2019-04-19 19:26:28,039] INFO [ThrottledRequestReaper-Request]: Stopped (kafka.server.ClientQuotaManager$ThrottledRequestReaper)
[2019-04-19 19:26:28,039] INFO [ThrottledRequestReaper-Request]: Shutdown completed (kafka.server.ClientQuotaManager$ThrottledRequestReaper)
[2019-04-19 19:26:28,040] INFO [SocketServer brokerId=1006] Shutting down socket server (kafka.network.SocketServer)
[2019-04-19 19:26:28,075] INFO [SocketServer brokerId=1006] Shutdown completed (kafka.network.SocketServer)
[2019-04-19 19:26:28,085] INFO [KafkaServer id=1006] shut down completed (kafka.server.KafkaServer)
[root@sandbox-hdp bin]#
```

broker 1001

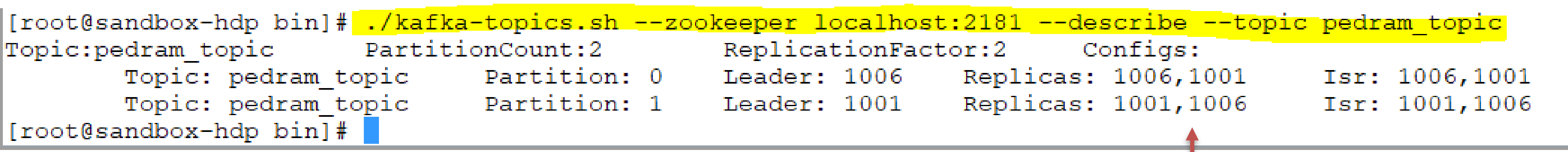


```
[2019-04-19 19:26:26,119] INFO [ReplicaAlterLogDirsManager on broker 1001] Added fetcher for partitions List() (kafka.server.ReplicaAlterLogDirsManager)
[2019-04-19 19:26:27,164] INFO [ReplicaAlterLogDirsManager on broker 1001] Added fetcher for partitions List() (kafka.server.ReplicaAlterLogDirsManager)
```



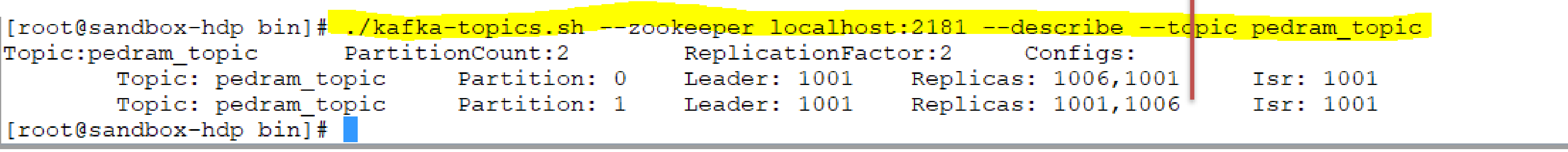
```
[root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1001    Replicas: 1006,1001     Isr: 1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001
[root@sandbox-hdp bin]#
```

Before shut down broker 1006



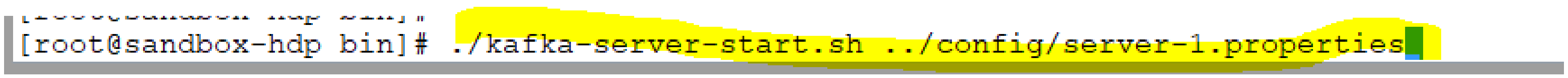
```
[root@sandbox-hdp bin]#  ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1006    Replicas: 1006,1001     Isr: 1006,1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001,1006
[root@sandbox-hdp bin]#
```

After shut down broker 1006

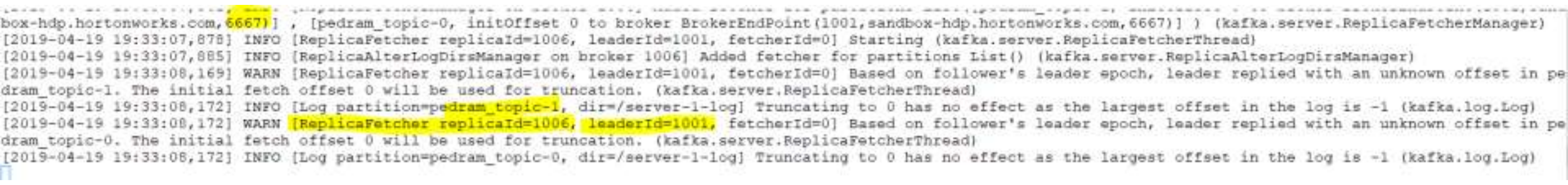


```
[root@sandbox-hdp bin]#  ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1001    Replicas: 1006,1001     Isr: 1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001
[root@sandbox-hdp bin]#
```

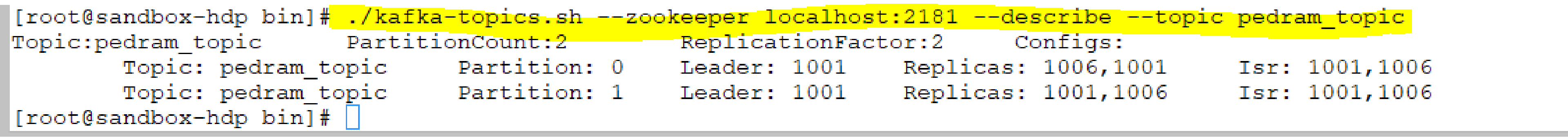
Replica tells you the historic picture .What is supposed to be there .
In sync replica tells you what is currently there .



```
[root@sandbox-hdp bin]# ./kafka-server-start.sh ../config/server-1.properties
```



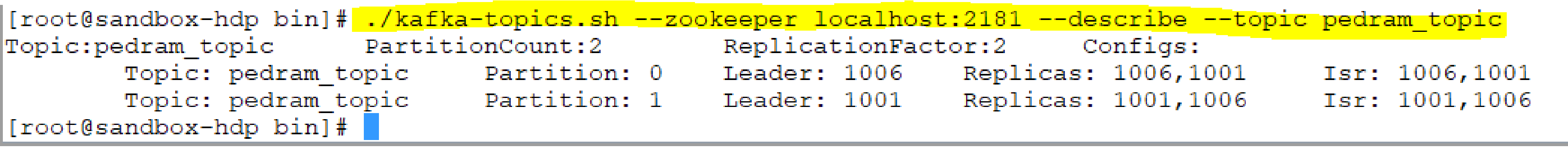
```
box-hdp.hortonworks.com,6667)] , [pedram_topic-0, initOffset 0 to broker BrokerEndPoint(1001,sandbox-hdp.hortonworks.com,6667)] ) (kafka.server.ReplicaFetcherManager)
[2019-04-19 19:33:07,878] INFO [ReplicaFetcher replicaId=1006, leaderId=1001, fetcherId=0] Starting (kafka.server.ReplicaFetcherThread)
[2019-04-19 19:33:07,885] INFO [ReplicaAlterLogDirsManager on broker 1006] Added fetcher for partitions List() (kafka.server.ReplicaAlterLogDirsManager)
[2019-04-19 19:33:08,169] WARN [ReplicaFetcher replicaId=1006, leaderId=1001, fetcherId=0] Based on follower's leader epoch, leader replied with an unknown offset in pe
dram_topic-1. The initial fetch offset 0 will be used for truncation. (kafka.server.ReplicaFetcherThread)
[2019-04-19 19:33:08,172] INFO [Log partition=pedram_topic-1, dir=/server-1-log] Truncating to 0 has no effect as the largest offset in the log is -1 (kafka.log.Log)
[2019-04-19 19:33:08,172] WARN [ReplicaFetcher replicaId=1006, leaderId=1001, fetcherId=0] Based on follower's leader epoch, leader replied with an unknown offset in pe
dram_topic-0. The initial fetch offset 0 will be used for truncation. (kafka.server.ReplicaFetcherThread)
[2019-04-19 19:33:08,172] INFO [Log partition=pedram_topic-0, dir=/server-1-log] Truncating to 0 has no effect as the largest offset in the log is -1 (kafka.log.Log)
```



```
[root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1001    Replicas: 1006,1001     Isr: 1001,1006
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001,1006
[root@sandbox-hdp bin]#
```

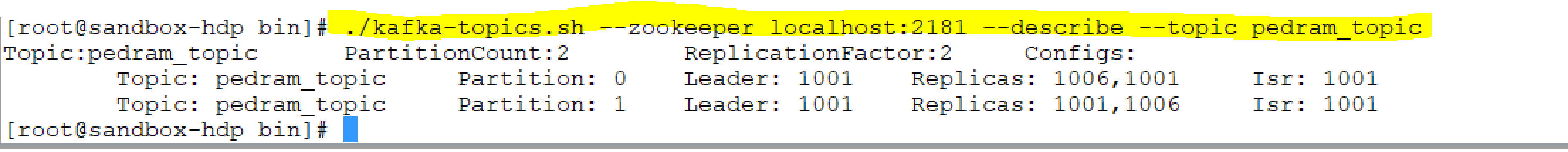
Bring broker 1 up again
Leader will still be 1001 (leader election)

Before shut down broker 1006



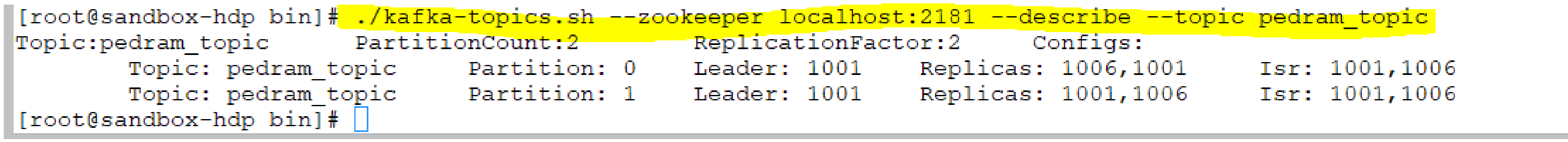
```
[root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1006    Replicas: 1006,1001     Isr: 1006,1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001,1006
[root@sandbox-hdp bin]#
```

After shut down broker 1006



```
[root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1001    Replicas: 1006,1001     Isr: 1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001
[root@sandbox-hdp bin]#
```

After bring up broker 1006



```
[root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1001    Replicas: 1006,1001     Isr: 1001,1006
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001,1006
[root@sandbox-hdp bin]#
```

Kafka Topics - Demo

<u>One Broker – Broker 0</u>

```
Topic:pedram_test	PartitionCount:2	ReplicationFactor:1	Configs:
	Topic: pedram_test	Partition: 0	Leader: 0	Replicas: 0	Isr: 0
	Topic: pedram_test	Partition: 1	Leader: 0	Replicas: 0	Isr: 0
```

<u>Two Brokers – Broker 0 & Broker 1</u>

```
Topic:pedram_test	PartitionCount:2	ReplicationFactor:2	Configs:
	Topic: pedram_test	Partition: 0	Leader: 1	Replicas: 1,0	Isr: 1,0
	Topic: pedram_test	Partition: 1	Leader: 0	Replicas: 0,1	Isr: 0,1
```

<u>Two Brokers – Broker 0 Up & Broker 1 Down</u>

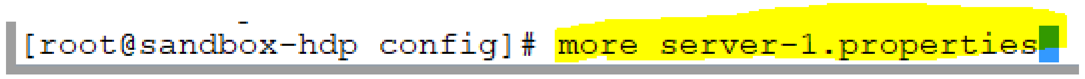
```
Topic:pedram_test	PartitionCount:2	ReplicationFactor:2	Configs:
	Topic: pedram_test	Partition: 0	Leader: 0	Replicas: 1,0	Isr: 0
	Topic: pedram_test	Partition: 1	Leader: 0	Replicas: 0,1	Isr: 0
```

<u>Two Brokers – Broker 0 Up & Broker 1 Up Again</u>

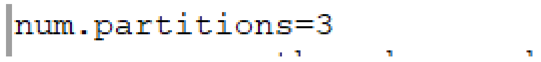
```
Topic:pedram_test	PartitionCount:2	ReplicationFactor:2	Configs:
	Topic: pedram_test	Partition: 0	Leader: 0	Replicas: 1,0	Isr: 0,1
	Topic: pedram_test	Partition: 1	Leader: 0	Replicas: 0,1	Isr: 0,1
```

If I have three broker and 4 partition how it will be distribute ?

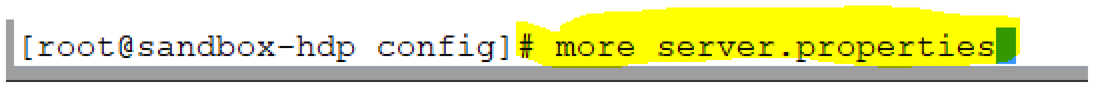
Each broker has a limit to hold the number of partition.



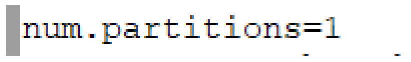
```
[root@sandbox-hdp config]# more server-1.properties
```



```
num.partitions=3
```



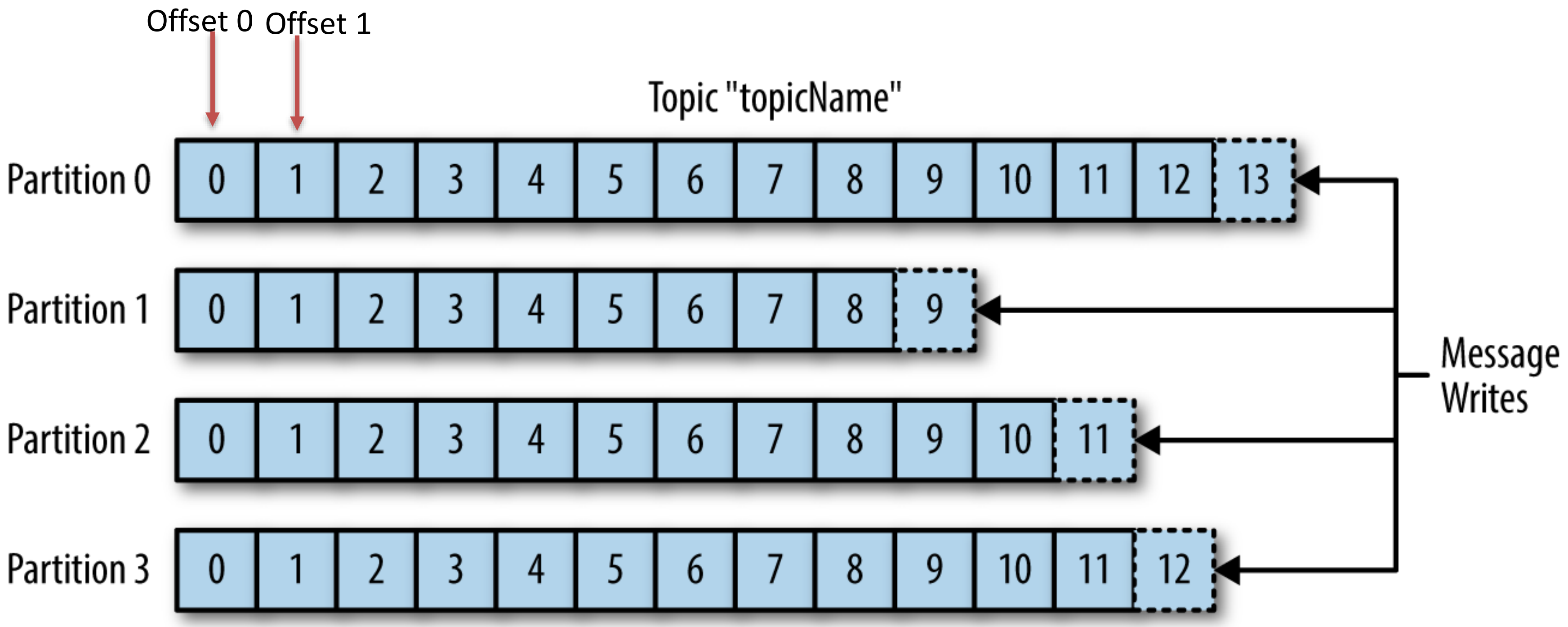
```
[root@sandbox-hdp config]# more server.properties
```



```
num.partitions=1
```

Architecture

- At the core of Kafka is a *topic*
- Each topic consists of multiple *partitions*
- Each partition is an immutable ordered sequence of records
- The ordered sequence is appended to a *commit log*
- Each record in a partition has a position index associated with it known as the *offset*
- Each record in a partition is retained up to a certain configurable period of time
- The partitions are replicated over the cluster nodes according to a configurable *replication factor*
- While distributed, one partition acts as the *leader* while the others act as *followers*

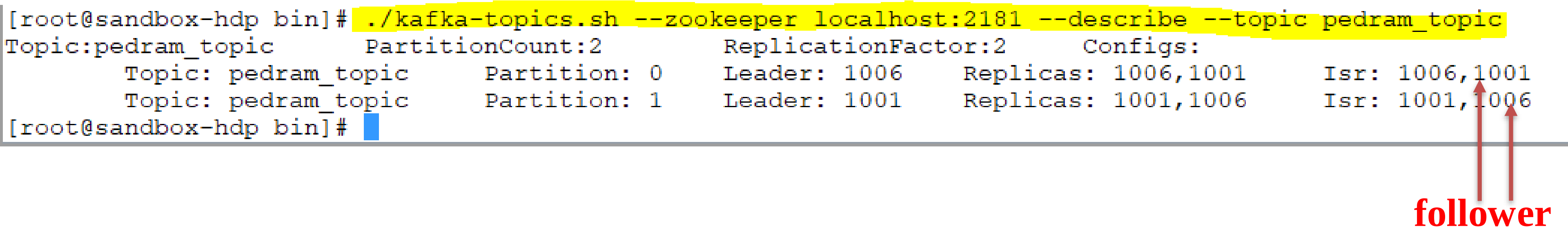


Within each partition each record has an offset

Offset is nothing than the position of the message or record within the partition

Architecture

- The leader serves as read-write requests while the followers passively replicate themselves to be *in-sync* with the leader
- If the leader fails, one of the followers is chosen as the leader
- The leader election is done by another Hadoop ecosystem component known as ***Zookeeper***
- The Zookeeper cluster maintains the *state information* for the Kafka cluster (decide who is the leader)
- Each *broker* in the Kafka cluster acts as the leader for some partitions while acting as the **follower** for other partitions
- Kafka implements the publish-subscribe model



```
[root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic pedram_topic
Topic:pedram_topic      PartitionCount:2        ReplicationFactor:2     Configs:
        Topic: pedram_topic     Partition: 0    Leader: 1006    Replicas: 1006,1001     Isr: 1006,1001
        Topic: pedram_topic     Partition: 1    Leader: 1001    Replicas: 1001,1006     Isr: 1001,1006
[root@sandbox-hdp bin]#
```

For partition 0 read and write will happens on the broker 1006. The other partitions are sync with leader 1006 silently

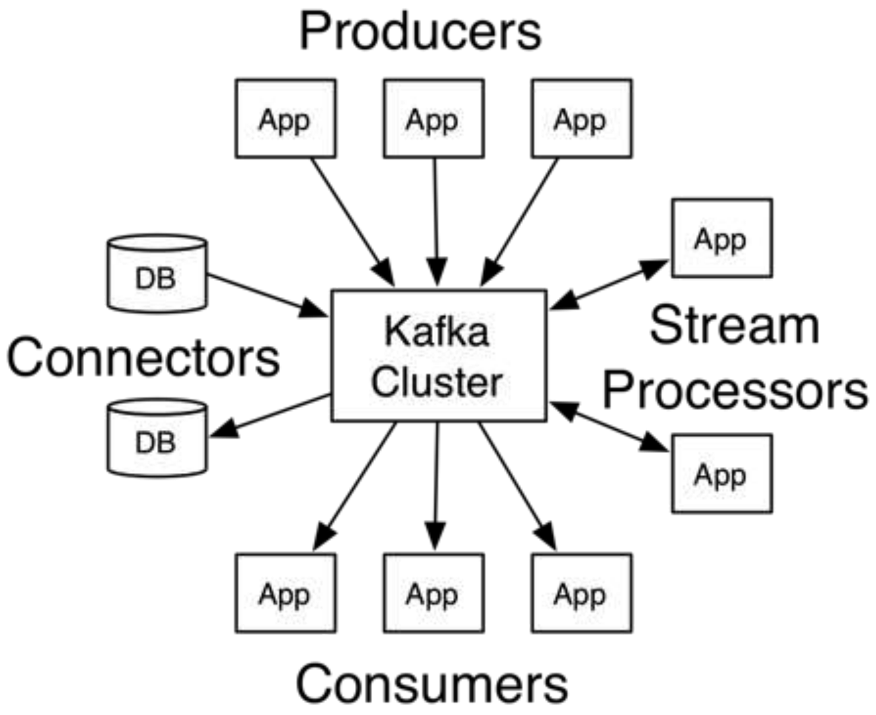
For partition 1 read and write will happens on the broker 1001. The other partitions are sync with leader 1001 silently

Producers

- Producers write messages to a Kafka topic
- They don’t care which partition the data would go to, though they may direct a particular message to a particular partition
- If a Producer has to direct a particular message to a particular partition, it has to use a **practitioner** will hash the key of the message & then map it to a particular topic

Features

- Distributed streaming platform
- Stream of records are stored in a fault-tolerant way
- Used as a messaging system plus as a data storage channel
- Consists of a central cluster and four API's viz. Producer, Consumer, Connector & Streams



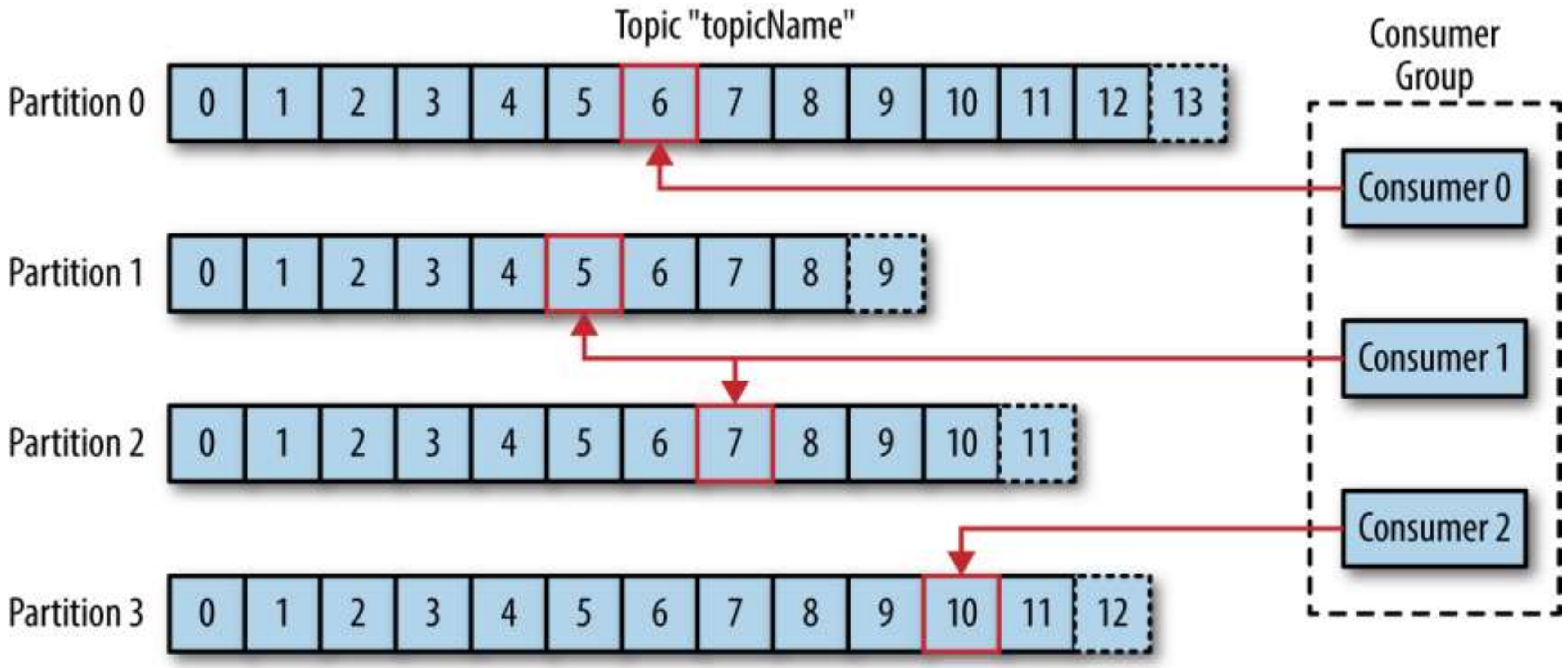
Consumers

- Consumers read messages from a Kafka topic
- Consumers maintain track of messages they have read from a particular topic using an offset
- Each message in a particular partition is assigned an offset (sequence number) by Kafka
- Offsets at a partition level for consumers is maintained in Zookeeper

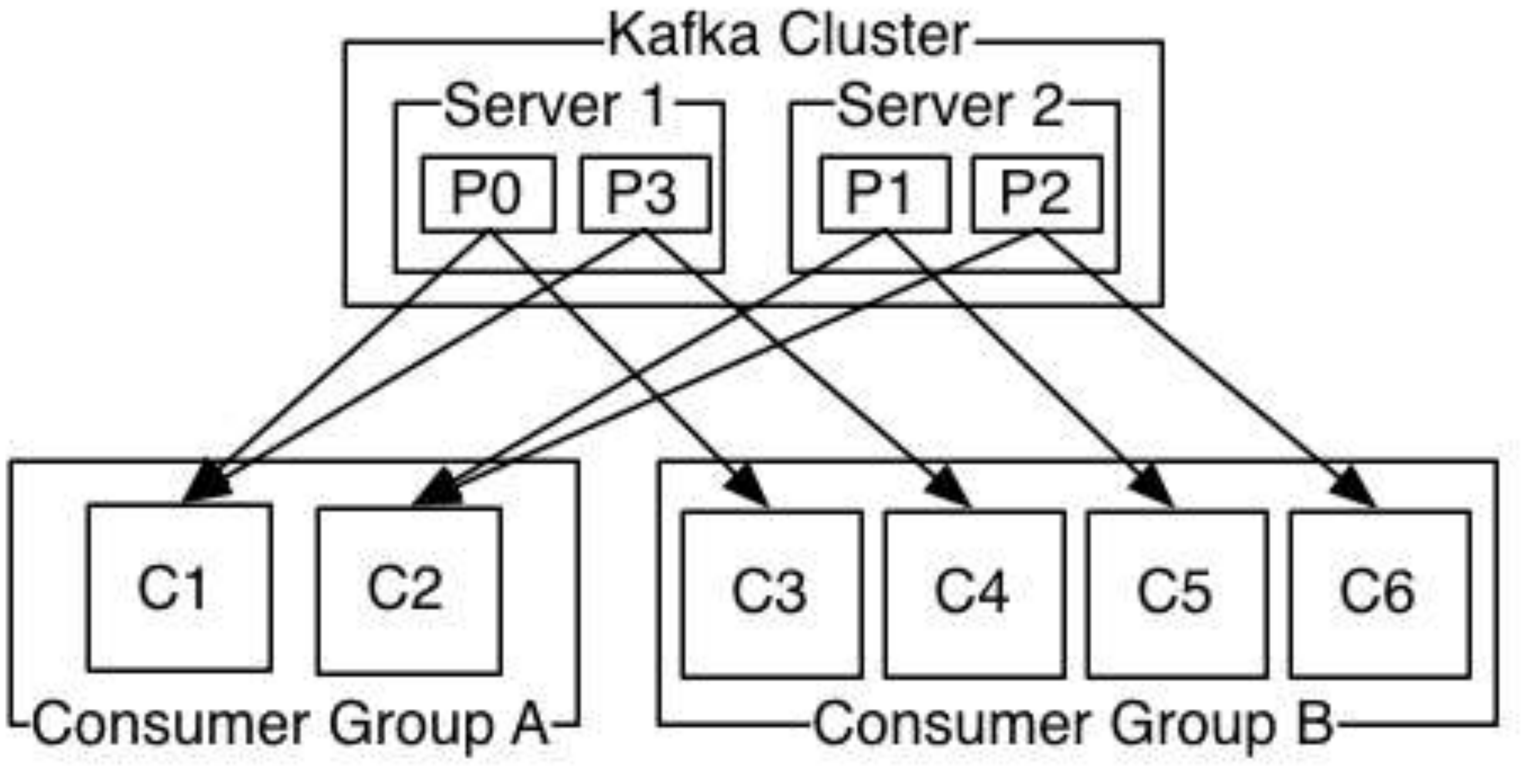
Consumer Groups

- Consumers that read from a single Kafka topic
- Partitions of the topic are evenly distributed amongst the consumers in the group
- The mapping of a consumer to a partition is often called **ownership of the partition by the consumer**
- Ensures scalability in terms of the Consumers
- If a particular Consumer fails, Kafka rebalances the distribution of partitions amongst the remaining consumers in the group

Consumer Groups



Consumer Groups



Producer & Consumer

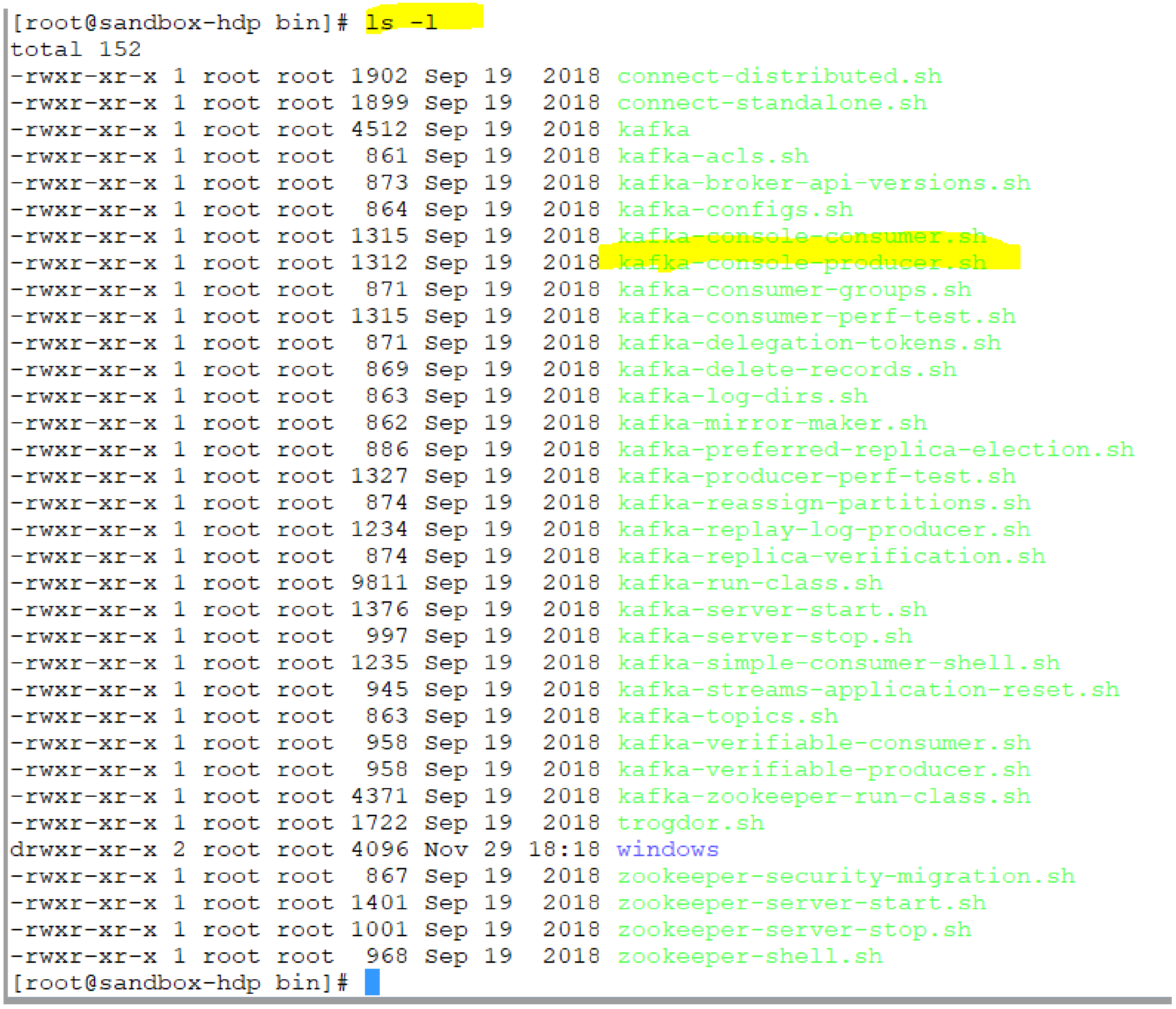
Producer

- Producers publish data to a particular partition of a particular topic
- The data is automatically written to the leader partition
- It is the responsibility of the producer to implement a partitioning mechanism of its data & then push it to the relevant partition

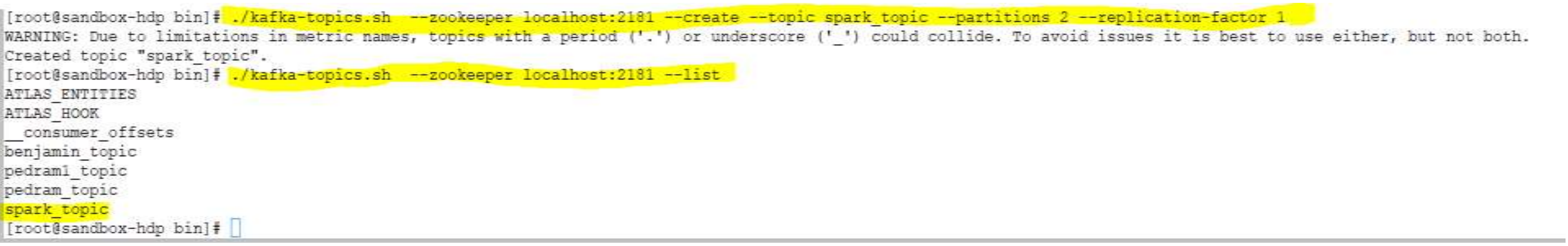
Consumer

- Consumers belong to a *consumer group*
- Each record in a topic is delivered to one instance in a particular consumer group

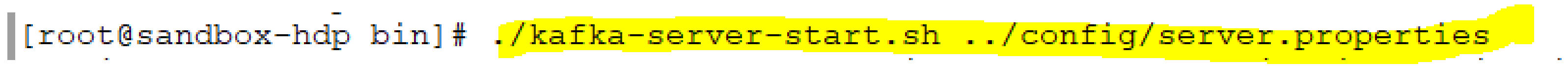
Console-producer is a command line tool to produce messages to Kafka



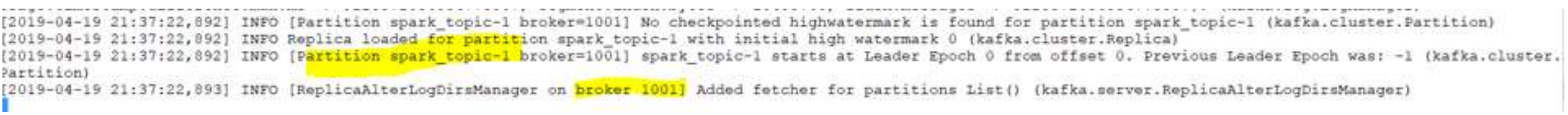
```
[root@sandbox-hdp bin]# ls -l
total 152
-rwxr-xr-x 1 root root 1902 Sep 19  2018 connect-distributed.sh
-rwxr-xr-x 1 root root 1899 Sep 19  2018 connect-standalone.sh
-rwxr-xr-x 1 root root 4512 Sep 19  2018 kafka
-rwxr-xr-x 1 root root  861 Sep 19  2018 kafka-acls.sh
-rwxr-xr-x 1 root root  873 Sep 19  2018 kafka-broker-api-versions.sh
-rwxr-xr-x 1 root root  864 Sep 19  2018 kafka-configs.sh
-rwxr-xr-x 1 root root 1315 Sep 19  2018 kafka-console-consumer.sh
-rwxr-xr-x 1 root root 1312 Sep 19  2018 kafka-console-producer.sh
-rwxr-xr-x 1 root root  871 Sep 19  2018 kafka-consumer-groups.sh
-rwxr-xr-x 1 root root 1315 Sep 19  2018 kafka-consumer-perf-test.sh
-rwxr-xr-x 1 root root  871 Sep 19  2018 kafka-delegation-tokens.sh
-rwxr-xr-x 1 root root  869 Sep 19  2018 kafka-delete-records.sh
-rwxr-xr-x 1 root root  863 Sep 19  2018 kafka-log-dirs.sh
-rwxr-xr-x 1 root root  862 Sep 19  2018 kafka-mirror-maker.sh
-rwxr-xr-x 1 root root  886 Sep 19  2018 kafka-preferred-replica-election.sh
-rwxr-xr-x 1 root root 1327 Sep 19  2018 kafka-producer-perf-test.sh
-rwxr-xr-x 1 root root  874 Sep 19  2018 kafka-reassign-partitions.sh
-rwxr-xr-x 1 root root 1234 Sep 19  2018 kafka-replay-log-producer.sh
-rwxr-xr-x 1 root root  874 Sep 19  2018 kafka-replica-verification.sh
-rwxr-xr-x 1 root root 9811 Sep 19  2018 kafka-run-class.sh
-rwxr-xr-x 1 root root 1376 Sep 19  2018 kafka-server-start.sh
-rwxr-xr-x 1 root root  997 Sep 19  2018 kafka-server-stop.sh
-rwxr-xr-x 1 root root 1235 Sep 19  2018 kafka-simple-consumer-shell.sh
-rwxr-xr-x 1 root root  945 Sep 19  2018 kafka-streams-application-reset.sh
-rwxr-xr-x 1 root root  863 Sep 19  2018 kafka-topics.sh
-rwxr-xr-x 1 root root  958 Sep 19  2018 kafka-verifiable-consumer.sh
-rwxr-xr-x 1 root root  958 Sep 19  2018 kafka-verifiable-producer.sh
-rwxr-xr-x 1 root root 4371 Sep 19  2018 kafka-zookeeper-run-class.sh
-rwxr-xr-x 1 root root 1722 Sep 19  2018 trogdor.sh
drwxr-xr-x 2 root root 4096 Nov 29 18:18 windows
-rwxr-xr-x 1 root root  867 Sep 19  2018 zookeeper-security-migration.sh
-rwxr-xr-x 1 root root 1401 Sep 19  2018 zookeeper-server-start.sh
-rwxr-xr-x 1 root root 1001 Sep 19  2018 zookeeper-server-stop.sh
-rwxr-xr-x 1 root root  968 Sep 19  2018 zookeeper-shell.sh
[root@sandbox-hdp bin]#
```



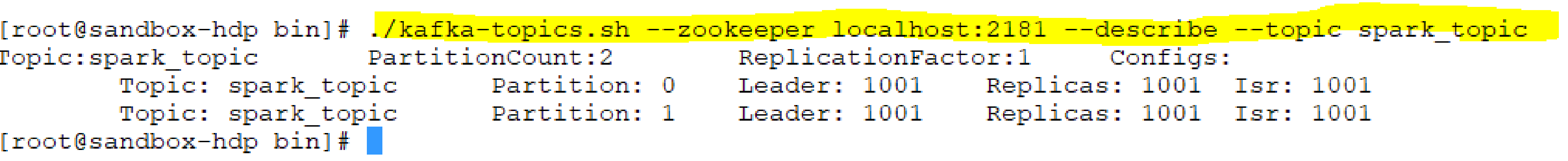
```
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --create --topic spark_topic --partitions 2 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic "spark_topic".
[root@sandbox-hdp bin]# ./kafka-topics.sh  --zookeeper localhost:2181 --list
ATLAS_ENTITIES
ATLAS_HOOK
__consumer_offsets
benjamin_topic
pedram1_topic
pedram_topic
spark_topic
[root@sandbox-hdp bin]#
```



```
[root@sandbox-hdp bin]# ./kafka-server-start.sh ../config/server.properties
```



```
[2019-04-19 21:37:22,892] INFO [Partition spark_topic-1 broker=1001] No checkpointed highwatermark is found for partition spark_topic-1 (kafka.cluster.Partition)
[2019-04-19 21:37:22,892] INFO Replica loaded for partition spark_topic-1 with initial high watermark 0 (kafka.cluster.Replica)
[2019-04-19 21:37:22,892] INFO [Partition spark_topic-1 broker=1001] spark_topic-1 starts at Leader Epoch 0 from offset 0. Previous Leader Epoch was: -1 (kafka.cluster.
Partition)
[2019-04-19 21:37:22,893] INFO [ReplicaAlterLogDirsManager on broker 1001] Added fetcher for partitions List() (kafka.server.ReplicaAlterLogDirsManager)
```



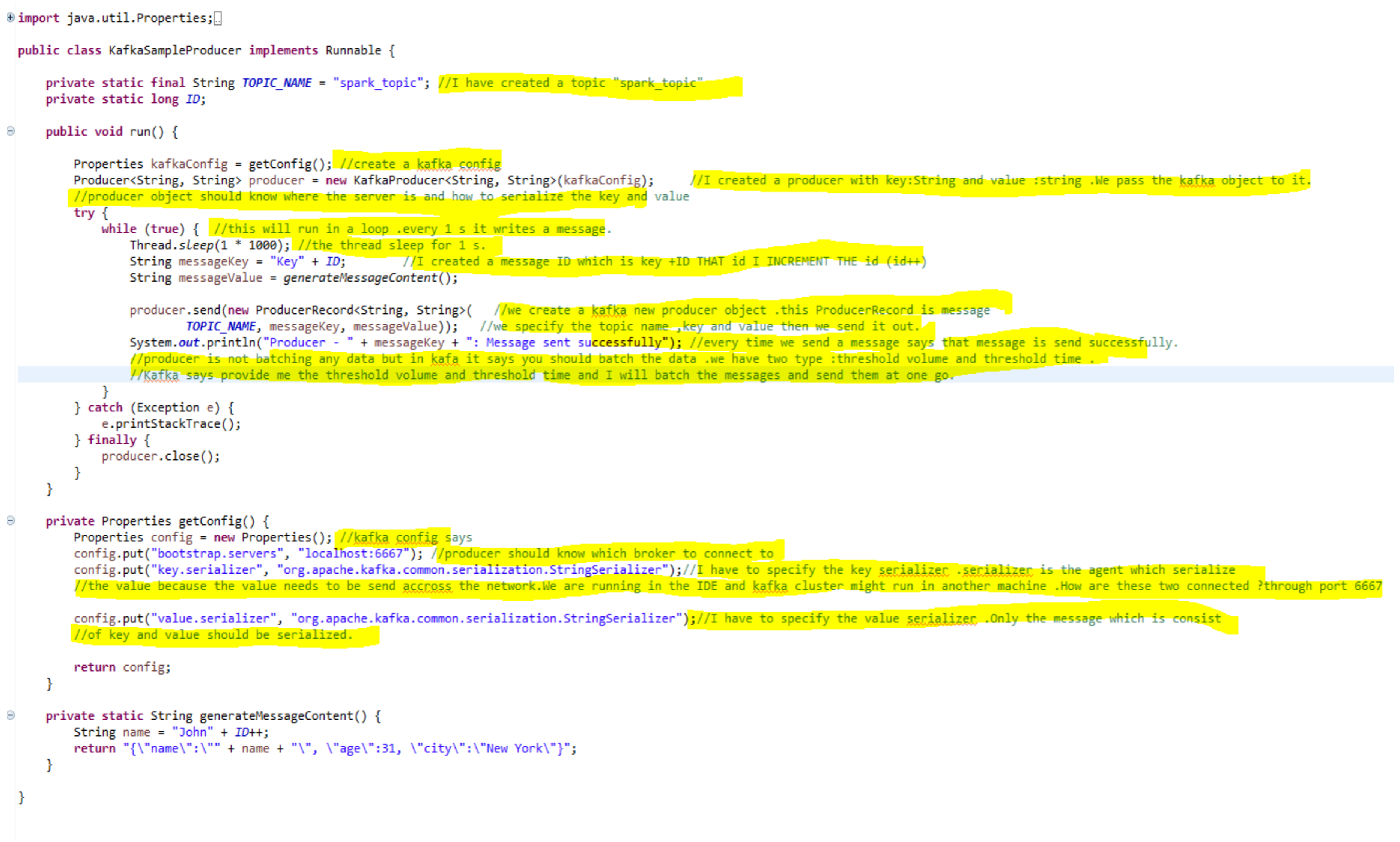
```
[root@sandbox-hdp bin]# ./kafka-topics.sh --zookeeper localhost:2181 --describe --topic spark_topic
Topic:spark_topic       PartitionCount:2        ReplicationFactor:1     Configs:
        Topic: spark_topic      Partition: 0    Leader: 1001    Replicas: 1001  Isr: 1001
        Topic: spark_topic      Partition: 1    Leader: 1001    Replicas: 1001  Isr: 1001
[root@sandbox-hdp bin]#
```

Three scenarios

- Console consumer /App producer
- Console producer /App consumer
- App producer /App consumer

The Kafka code is written in Java so you need to install Eclipse for Java .

Console consumer /App producer



```
import java.util.Properties;

public class KafkaSampleProducer implements Runnable {

    private static final String TOPIC_NAME = "spark_topic"; //I have created a topic "spark_topic"
    private static long ID;

    public void run() {

        Properties kafkaConfig = getConfig(); //create a kafka config
        Producer<String, String> producer = new KafkaProducer<String, String>(kafkaConfig);     //I created a producer with key:String and value :string .We pass the kafka object to it.
        //producer object should know where the server is and how to serialize the key and value
        try {
            while (true) {  //this will run in a loop .every 1 s it writes a message.
                Thread.sleep(1 * 1000); //the thread sleep for 1 s.
                String messageKey = "Key" + ID;        //I created a message ID which is key +ID THAT id I INCREMENT THE id (id++)
                String messageValue = generateMessageContent();

                producer.send(new ProducerRecord<String, String>(   //we create a kafka new producer object .this ProducerRecord is message
                        TOPIC_NAME, messageKey, messageValue));   //we specify the topic name ,key and value then we send it out.
                System.out.println("Producer - " + messageKey + ": Message sent successfully"); //every time we send a message says that message is send successfully.
                //producer is not batching any data but in kafa it says you should batch the data .we have two type :threshold volume and threshold time .
                //Kafka says provide me the threshold volume and threshold time and I will batch the messages and send them at one go.
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            producer.close();
        }
    }

    private Properties getConfig() {
        Properties config = new Properties(); //kafka config says
        config.put("bootstrap.servers", "localhost:6667"); //producer should know which broker to connect to
        config.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");//I have to specify the key serializer .serializer is the agent which serialize
        //the value because the value needs to be send accross the network.We are running in the IDE and kafka cluster might run in another machine .How are these two connected ?through port 6667

        config.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");//I have to specify the value serializer .Only the message which is consist
        //of key and value should be serialized.

        return config;
    }

    private static String generateMessageContent() {
        String name = "John" + ID++;
        return "{\"name\":\"" + name + "\", \"age\":31, \"city\":\"New York\"}";
    }

}
```

Example 1:

Volume threshold - 100 MB
Time Threshold - 60 sec
1 message /second –
Each msg -10 MB
10 SEC – 10 msg – 10 (10mb) =100 MB .It means your volume threshold will be reach first and it will send out all .Kafka producer does not send one message at a time.
Kafka producer will send batches of the messages then send them out.
======================================

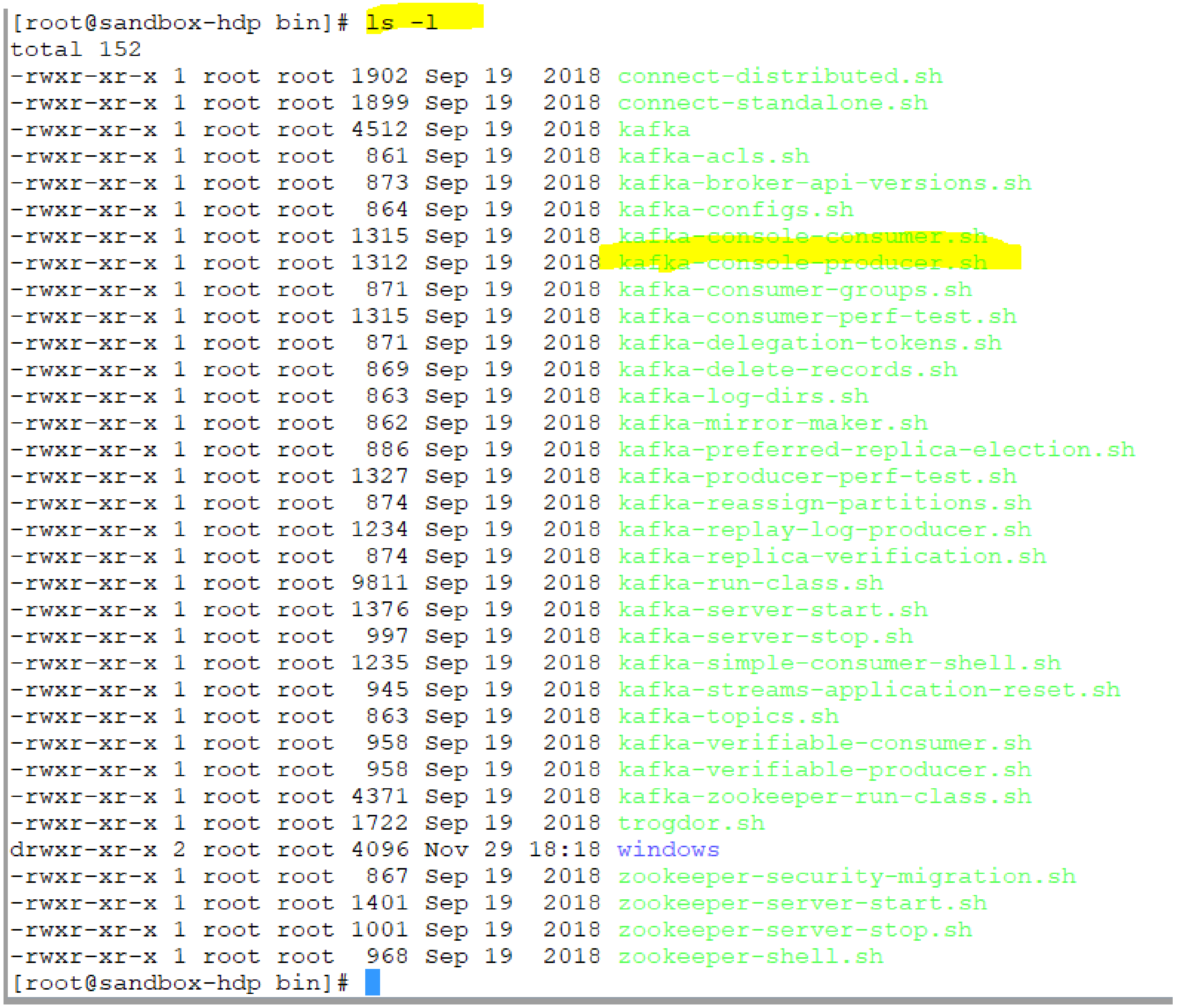
Example 2:

Volume threshold - 100 MB
Time Threshold - 60 sec

1 message /second –
Each msg -1 MB

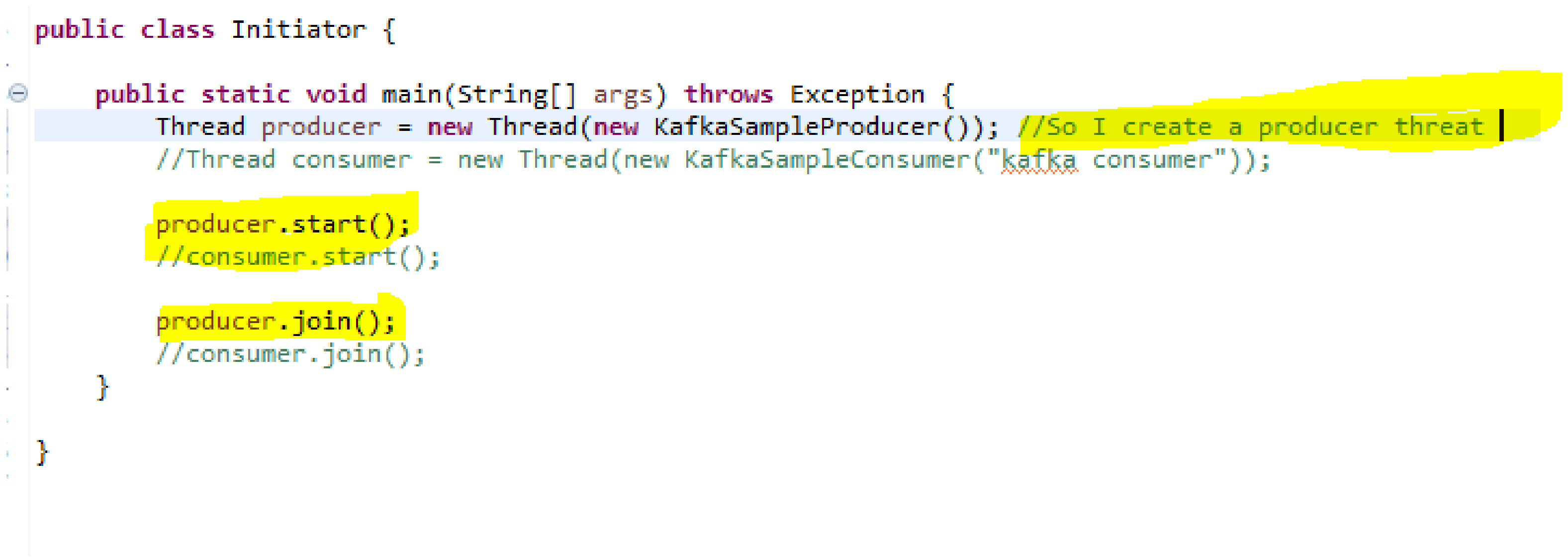
10 SEC – 10 msg – 10 (1mb) =10 MB
60 sec -60 msg -60 (1MB) -60 mb
So kafka producer will send 60 messages in 60 second to kafka broker.

Console-producer is a command line tool to produce messages to Kafka



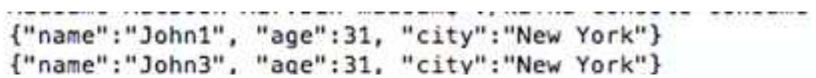
```
[root@sandbox-hdp bin]# ls -l
total 152
-rwxr-xr-x 1 root root 1902 Sep 19  2018 connect-distributed.sh
-rwxr-xr-x 1 root root 1899 Sep 19  2018 connect-standalone.sh
-rwxr-xr-x 1 root root 4512 Sep 19  2018 kafka
-rwxr-xr-x 1 root root  861 Sep 19  2018 kafka-acls.sh
-rwxr-xr-x 1 root root  873 Sep 19  2018 kafka-broker-api-versions.sh
-rwxr-xr-x 1 root root  864 Sep 19  2018 kafka-configs.sh
-rwxr-xr-x 1 root root 1315 Sep 19  2018 kafka-console-consumer.sh
-rwxr-xr-x 1 root root 1312 Sep 19  2018 kafka-console-producer.sh
-rwxr-xr-x 1 root root  871 Sep 19  2018 kafka-consumer-groups.sh
-rwxr-xr-x 1 root root 1315 Sep 19  2018 kafka-consumer-perf-test.sh
-rwxr-xr-x 1 root root  871 Sep 19  2018 kafka-delegation-tokens.sh
-rwxr-xr-x 1 root root  869 Sep 19  2018 kafka-delete-records.sh
-rwxr-xr-x 1 root root  863 Sep 19  2018 kafka-log-dirs.sh
-rwxr-xr-x 1 root root  862 Sep 19  2018 kafka-mirror-maker.sh
-rwxr-xr-x 1 root root  886 Sep 19  2018 kafka-preferred-replica-election.sh
-rwxr-xr-x 1 root root 1327 Sep 19  2018 kafka-producer-perf-test.sh
-rwxr-xr-x 1 root root  874 Sep 19  2018 kafka-reassign-partitions.sh
-rwxr-xr-x 1 root root 1234 Sep 19  2018 kafka-replay-log-producer.sh
-rwxr-xr-x 1 root root  874 Sep 19  2018 kafka-replica-verification.sh
-rwxr-xr-x 1 root root 9811 Sep 19  2018 kafka-run-class.sh
-rwxr-xr-x 1 root root 1376 Sep 19  2018 kafka-server-start.sh
-rwxr-xr-x 1 root root  997 Sep 19  2018 kafka-server-stop.sh
-rwxr-xr-x 1 root root 1235 Sep 19  2018 kafka-simple-consumer-shell.sh
-rwxr-xr-x 1 root root  945 Sep 19  2018 kafka-streams-application-reset.sh
-rwxr-xr-x 1 root root  863 Sep 19  2018 kafka-topics.sh
-rwxr-xr-x 1 root root  958 Sep 19  2018 kafka-verifiable-consumer.sh
-rwxr-xr-x 1 root root  958 Sep 19  2018 kafka-verifiable-producer.sh
-rwxr-xr-x 1 root root 4371 Sep 19  2018 kafka-zookeeper-run-class.sh
-rwxr-xr-x 1 root root 1722 Sep 19  2018 trogdor.sh
drwxr-xr-x 2 root root 4096 Nov 29 18:18 windows
-rwxr-xr-x 1 root root  867 Sep 19  2018 zookeeper-security-migration.sh
-rwxr-xr-x 1 root root 1401 Sep 19  2018 zookeeper-server-start.sh
-rwxr-xr-x 1 root root 1001 Sep 19  2018 zookeeper-server-stop.sh
-rwxr-xr-x 1 root root  968 Sep 19  2018 zookeeper-shell.sh
[root@sandbox-hdp bin]#
```

```
[root@sandbox-hdp bin]# ./kafka-console-consumer.sh --bootstrap-server localhost:6667 --topic spark_topic --from-beginning
```

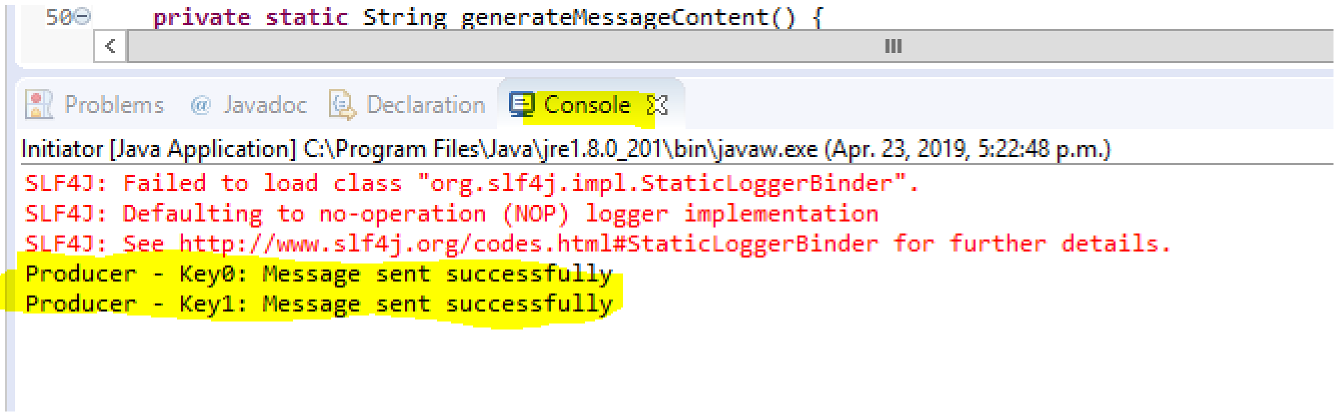


```java
public class Initiator {

    public static void main(String[] args) throws Exception {
        Thread producer = new Thread(new KafkaSampleProducer()); //So I create a producer threat
        //Thread consumer = new Thread(new KafkaSampleConsumer("kafka consumer"));

        producer.start();
        //consumer.start();

        producer.join();
        //consumer.join();
    }

}
```

```
[root@sandbox-hdp bin]# ./kafka-console-consumer.sh --bootstrap-server localhost:6667 --topic spark_topic --from-beginning
```



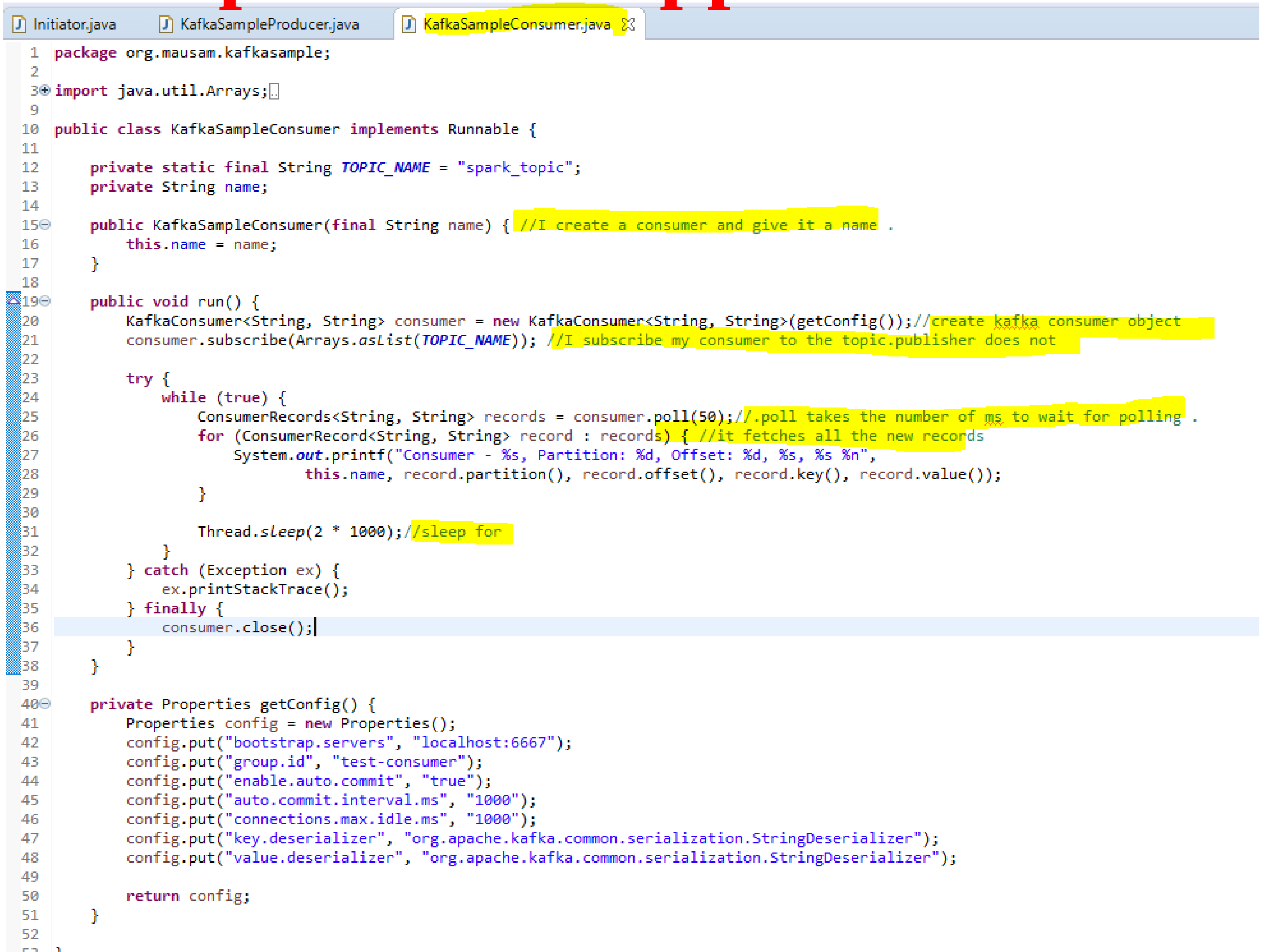
```
{"name":"John1", "age":31, "city":"New York"}
{"name":"John3", "age":31, "city":"New York"}
```



```
    private static String generateMessageContent() {
```

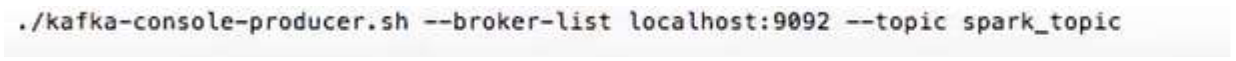
```
Initiator [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Apr. 23, 2019, 5:22:48 p.m.)
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Producer - Key0: Message sent successfully
Producer - Key1: Message sent successfully
```

Console producer /App consumer

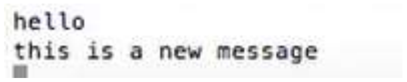


Initiator.java   KafkaSampleProducer.java   KafkaSampleConsumer.java

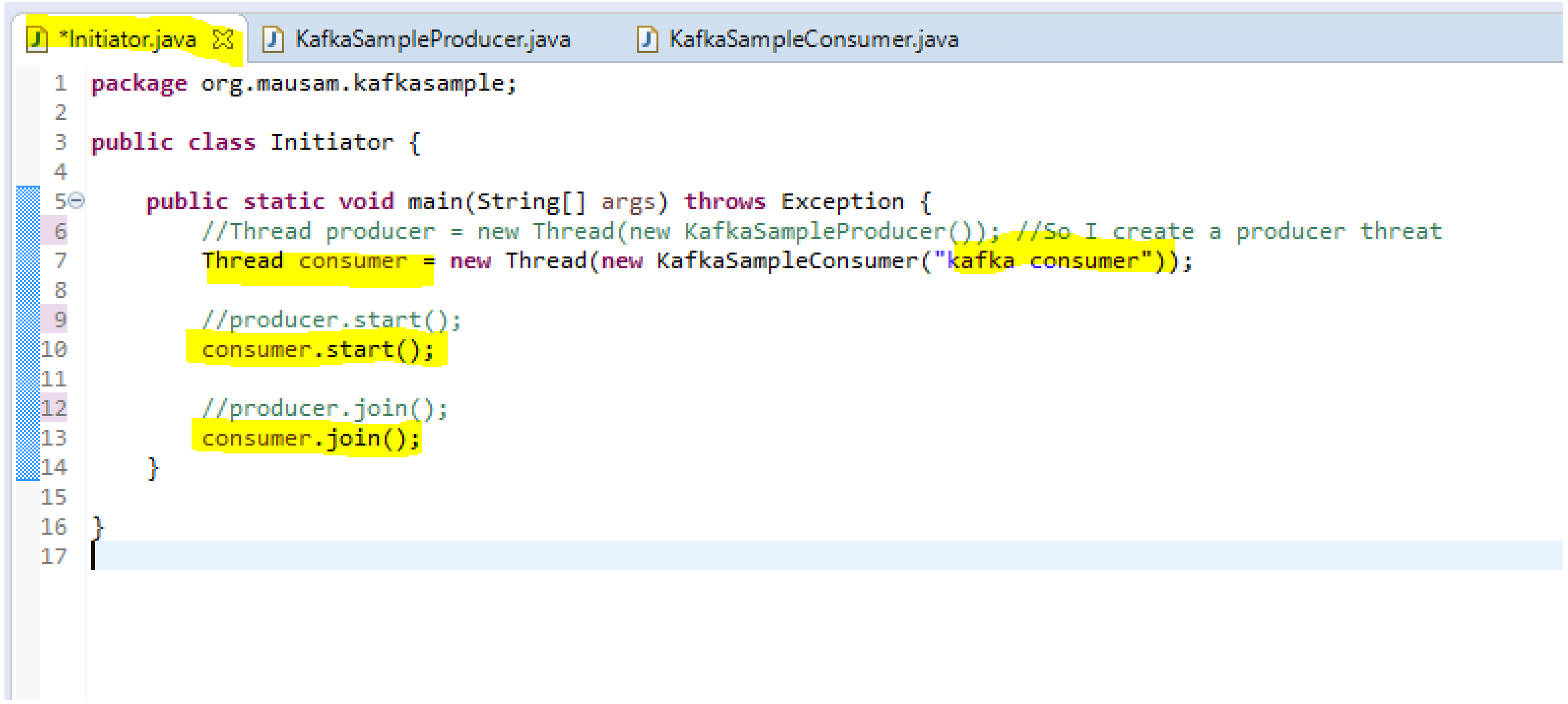
```java
package org.mausam.kafkasample;

import java.util.Arrays;

public class KafkaSampleConsumer implements Runnable {

    private static final String TOPIC_NAME = "spark_topic";
    private String name;

    public KafkaSampleConsumer(final String name) { //I create a consumer and give it a name .
        this.name = name;
    }

    public void run() {
        KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(getConfig());//create kafka consumer object
        consumer.subscribe(Arrays.asList(TOPIC_NAME)); //I subscribe my consumer to the topic.publisher does not

        try {
            while (true) {
                ConsumerRecords<String, String> records = consumer.poll(50);//.poll takes the number of ms to wait for polling .
                for (ConsumerRecord<String, String> record : records) { //it fetches all the new records
                    System.out.printf("Consumer - %s, Partition: %d, Offset: %d, %s, %s %n",
                            this.name, record.partition(), record.offset(), record.key(), record.value());
                }

                Thread.sleep(2 * 1000);//sleep for
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            consumer.close();
        }
    }

    private Properties getConfig() {
        Properties config = new Properties();
        config.put("bootstrap.servers", "localhost:6667");
        config.put("group.id", "test-consumer");
        config.put("enable.auto.commit", "true");
        config.put("auto.commit.interval.ms", "1000");
        config.put("connections.max.idle.ms", "1000");
        config.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        config.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

        return config;
    }

}
```



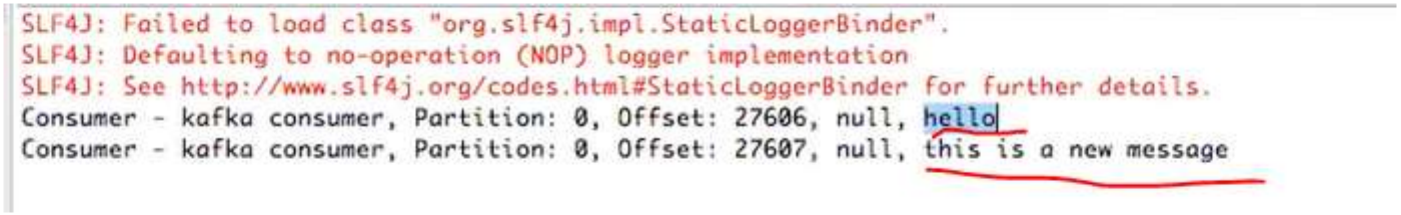
```
./kafka-console-producer.sh --broker-list localhost:9092 --topic spark_topic
```



```
hello
this is a new message
```

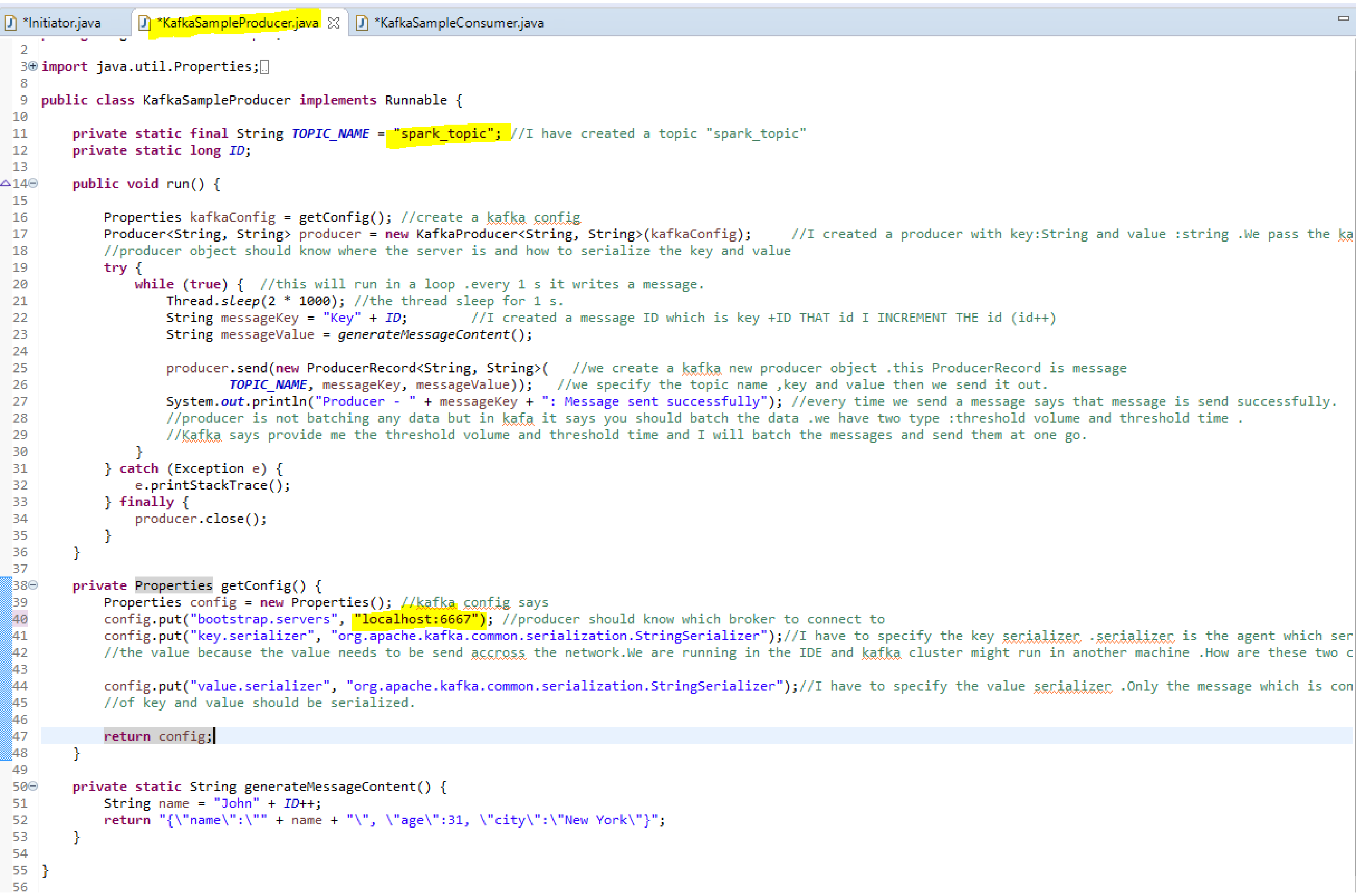


```java
package org.mausam.kafkasample;

public class Initiator {

    public static void main(String[] args) throws Exception {
        //Thread producer = new Thread(new KafkaSampleProducer()); //So I create a producer threat
        Thread consumer = new Thread(new KafkaSampleConsumer("kafka consumer"));

        //producer.start();
        consumer.start();

        //producer.join();
        consumer.join();
    }

}
```

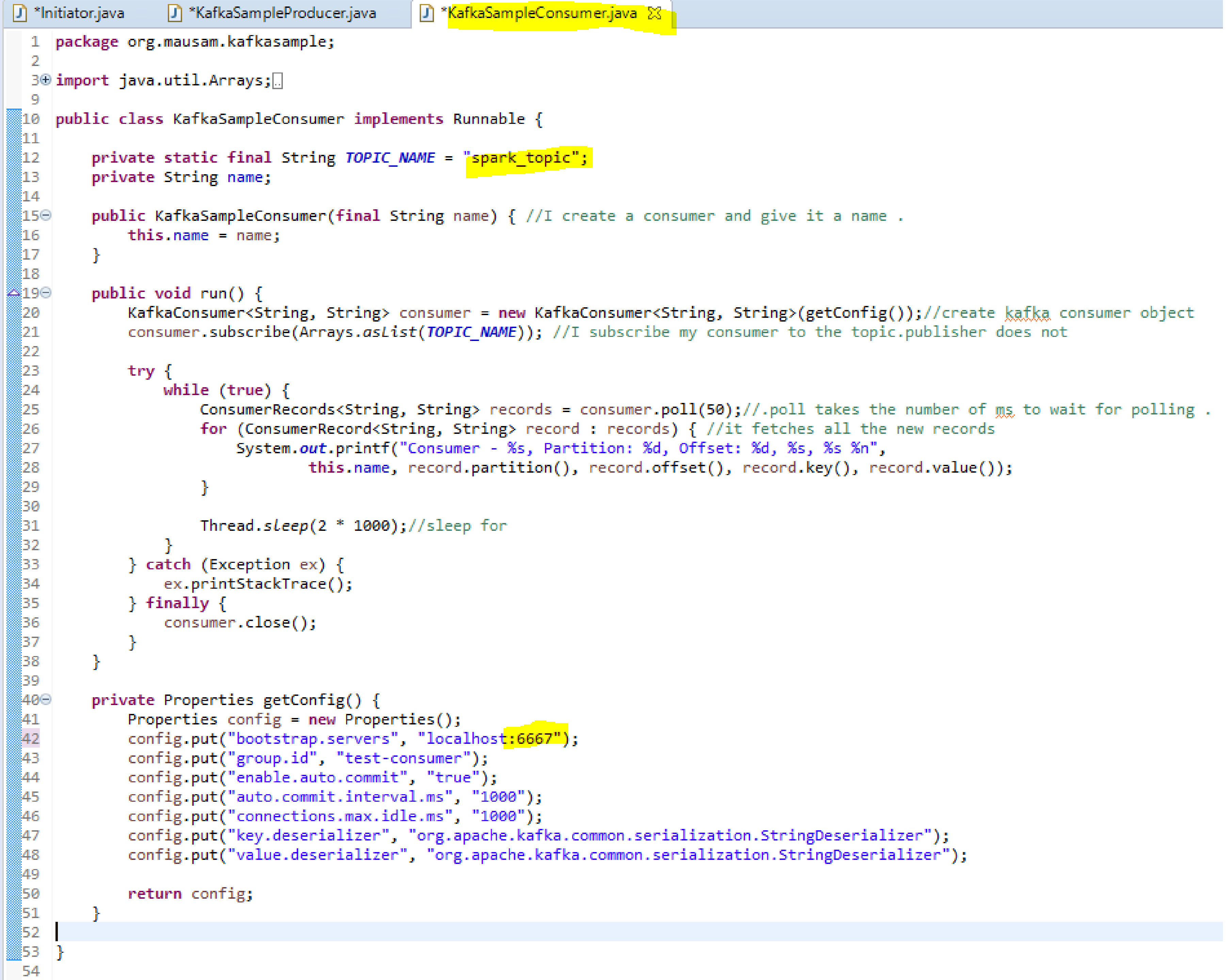


```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Consumer - kafka consumer, Partition: 0, Offset: 27606, null, hello
Consumer - kafka consumer, Partition: 0, Offset: 27607, null, this is a new message
```

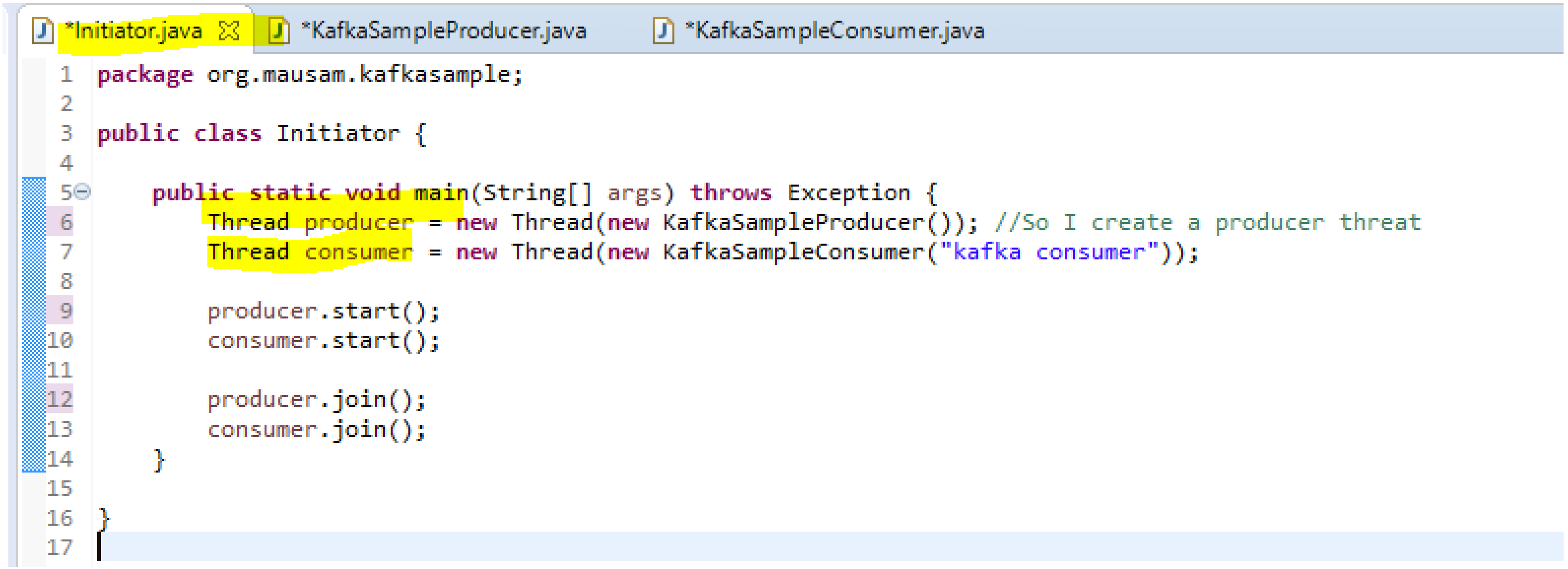
App producer /App consumer



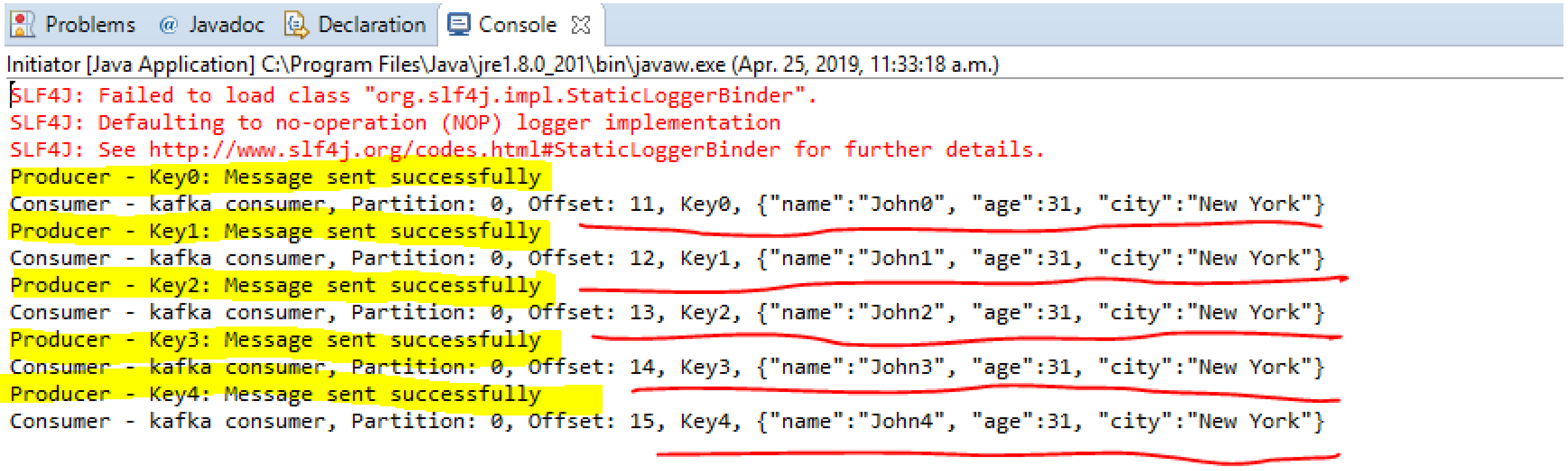
```java

import java.util.Properties;

public class KafkaSampleProducer implements Runnable {

    private static final String TOPIC_NAME = "spark_topic"; //I have created a topic "spark_topic"
    private static long ID;

    public void run() {

        Properties kafkaConfig = getConfig(); //create a kafka config
        Producer<String, String> producer = new KafkaProducer<String, String>(kafkaConfig);     //I created a producer with key:String and value :string .We pass the ka
        //producer object should know where the server is and how to serialize the key and value
        try {
            while (true) {  //this will run in a loop .every 1 s it writes a message.
                Thread.sleep(2 * 1000); //the thread sleep for 1 s.
                String messageKey = "Key" + ID;         //I created a message ID which is key +ID THAT id I INCREMENT THE id (id++)
                String messageValue = generateMessageContent();

                producer.send(new ProducerRecord<String, String>(   //we create a kafka new producer object .this ProducerRecord is message
                        TOPIC_NAME, messageKey, messageValue));   //we specify the topic name ,key and value then we send it out.
                System.out.println("Producer - " + messageKey + ": Message sent successfully"); //every time we send a message says that message is send successfully.
                //producer is not batching any data but in kafa it says you should batch the data .we have two type :threshold volume and threshold time .
                //Kafka says provide me the threshold volume and threshold time and I will batch the messages and send them at one go.
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            producer.close();
        }
    }

    private Properties getConfig() {
        Properties config = new Properties(); //kafka config says
        config.put("bootstrap.servers", "localhost:6667"); //producer should know which broker to connect to
        config.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");//I have to specify the key serializer .serializer is the agent which ser
        //the value because the value needs to be send accross the network.We are running in the IDE and kafka cluster might run in another machine .How are these two c

        config.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");//I have to specify the value serializer .Only the message which is con
        //of key and value should be serialized.

        return config;
    }

    private static String generateMessageContent() {
        String name = "John" + ID++;
        return "{\"name\":\"" + name + "\", \"age\":31, \"city\":\"New York\"}";
    }

}
```


*Initiator.java   *KafkaSampleProducer.java   *KafkaSampleConsumer.java

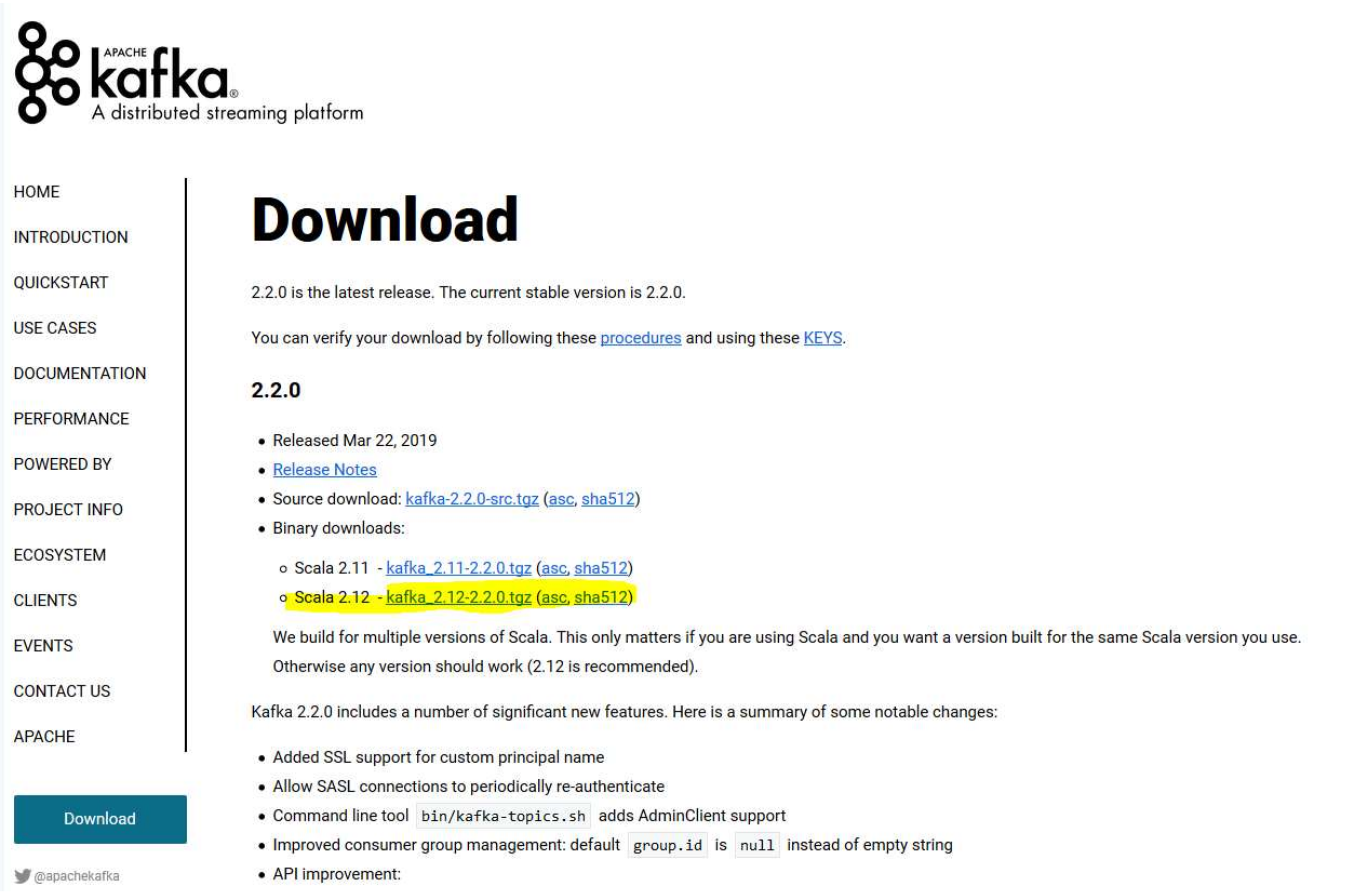
```java
package org.mausam.kafkasample;

import java.util.Arrays;

public class KafkaSampleConsumer implements Runnable {

    private static final String TOPIC_NAME = "spark_topic";
    private String name;

    public KafkaSampleConsumer(final String name) { //I create a consumer and give it a name .
        this.name = name;
    }

    public void run() {
        KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(getConfig());//create kafka consumer object
        consumer.subscribe(Arrays.asList(TOPIC_NAME)); //I subscribe my consumer to the topic.publisher does not

        try {
            while (true) {
                ConsumerRecords<String, String> records = consumer.poll(50);//.poll takes the number of ms to wait for polling .
                for (ConsumerRecord<String, String> record : records) { //it fetches all the new records
                    System.out.printf("Consumer - %s, Partition: %d, Offset: %d, %s, %s %n",
                            this.name, record.partition(), record.offset(), record.key(), record.value());
                }

                Thread.sleep(2 * 1000);//sleep for
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            consumer.close();
        }
    }

    private Properties getConfig() {
        Properties config = new Properties();
        config.put("bootstrap.servers", "localhost:6667");
        config.put("group.id", "test-consumer");
        config.put("enable.auto.commit", "true");
        config.put("auto.commit.interval.ms", "1000");
        config.put("connections.max.idle.ms", "1000");
        config.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        config.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

        return config;
    }

}
```



```java
package org.mausam.kafkasample;

public class Initiator {

    public static void main(String[] args) throws Exception {
        Thread producer = new Thread(new KafkaSampleProducer()); //So I create a producer threat
        Thread consumer = new Thread(new KafkaSampleConsumer("kafka consumer"));

        producer.start();
        consumer.start();

        producer.join();
        consumer.join();
    }

}
```



```
Initiator [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Apr. 25, 2019, 11:33:18 a.m.)
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Producer - Key0: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 11, Key0, {"name":"John0", "age":31, "city":"New York"}
Producer - Key1: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 12, Key1, {"name":"John1", "age":31, "city":"New York"}
Producer - Key2: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 13, Key2, {"name":"John2", "age":31, "city":"New York"}
Producer - Key3: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 14, Key3, {"name":"John3", "age":31, "city":"New York"}
Producer - Key4: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 15, Key4, {"name":"John4", "age":31, "city":"New York"}
```



Download

2.2.0 is the latest release. The current stable version is 2.2.0.

You can verify your download by following these procedures and using these KEYS.

2.2.0

- Released Mar 22, 2019
- Release Notes
- Source download: kafka-2.2.0-src.tgz (asc, sha512)
- Binary downloads:
  - Scala 2.11 - kafka_2.11-2.2.0.tgz (asc, sha512)
  - Scala 2.12 - kafka_2.12-2.2.0.tgz (asc, sha512)

  We build for multiple versions of Scala. This only matters if you are using Scala and you want a version built for the same Scala version you use. Otherwise any version should work (2.12 is recommended).

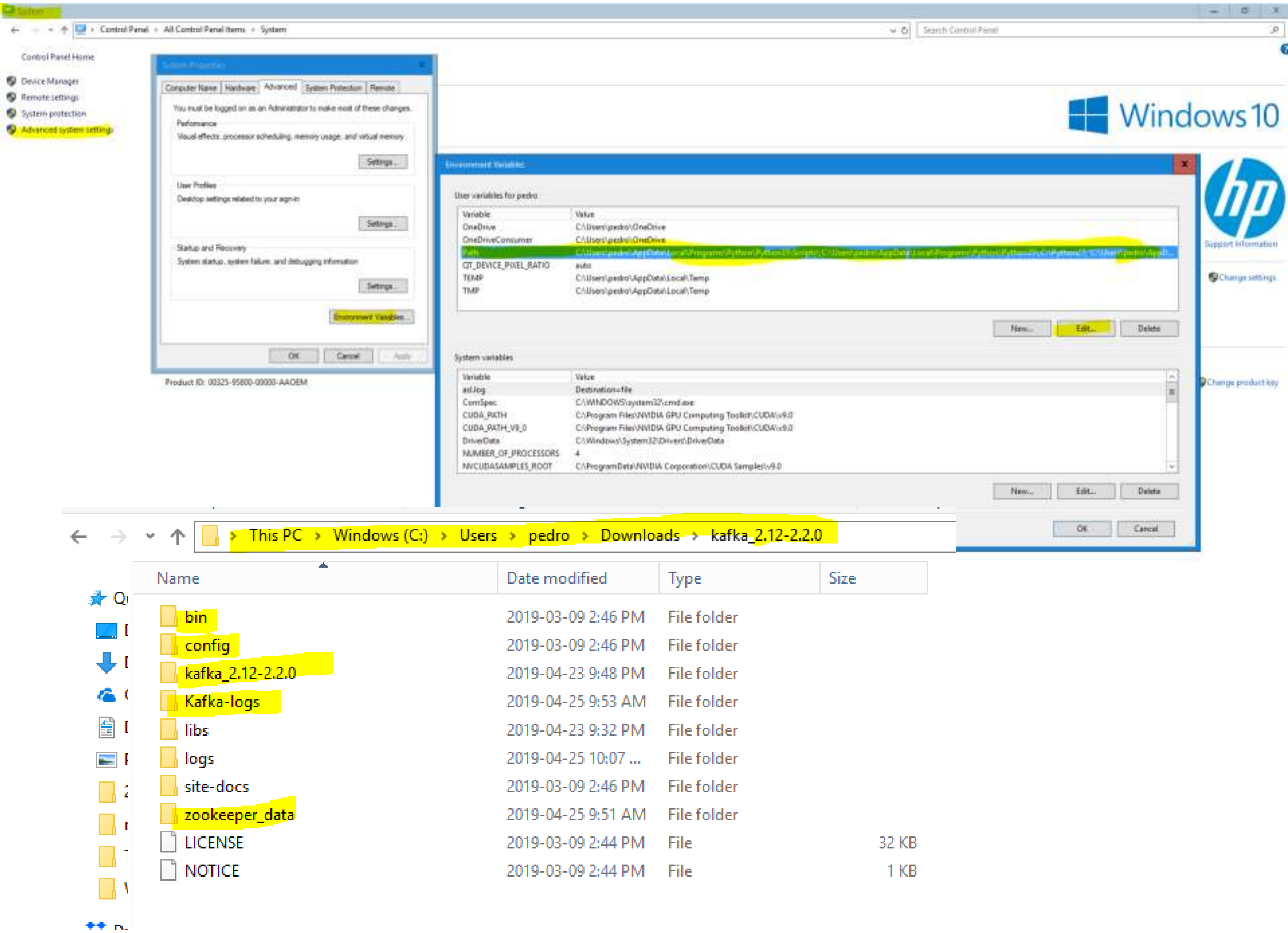
Kafka 2.2.0 includes a number of significant new features. Here is a summary of some notable changes:

- Added SSL support for custom principal name
- Allow SASL connections to periodically re-authenticate
- Command line tool `bin/kafka-topics.sh` adds AdminClient support
- Improved consumer group management: default `group.id` is `null` instead of empty string
- API improvement:

https://kafka.apache.org/downloads

Kafka Streams - Installing Kafka on Windows

https://www.youtube.com/watch?v=TTsOoQ6_QB0



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK | Cancel | Apply

Product ID: 00325-95800-00000-AAOEM

Environment Variables

User variables for pedro

Variable	Value
OneDrive	C:\Users\pedro\OneDrive
OneDriveConsumer	C:\Users\pedro\OneDrive
Path	C:\Users\pedro\AppData\Local\Programs\Python\Python37\Scripts\;C:\Users\pedro\AppData\Local\Programs\Python\Python37\;C:\Python27\;C:\Users\pedro\AppD...
QT_DEVICE_PIXEL_RATIO	auto
TEMP	C:\Users\pedro\AppData\Local\Temp
TMP	C:\Users\pedro\AppData\Local\Temp

New... | Edit... | Delete

System variables

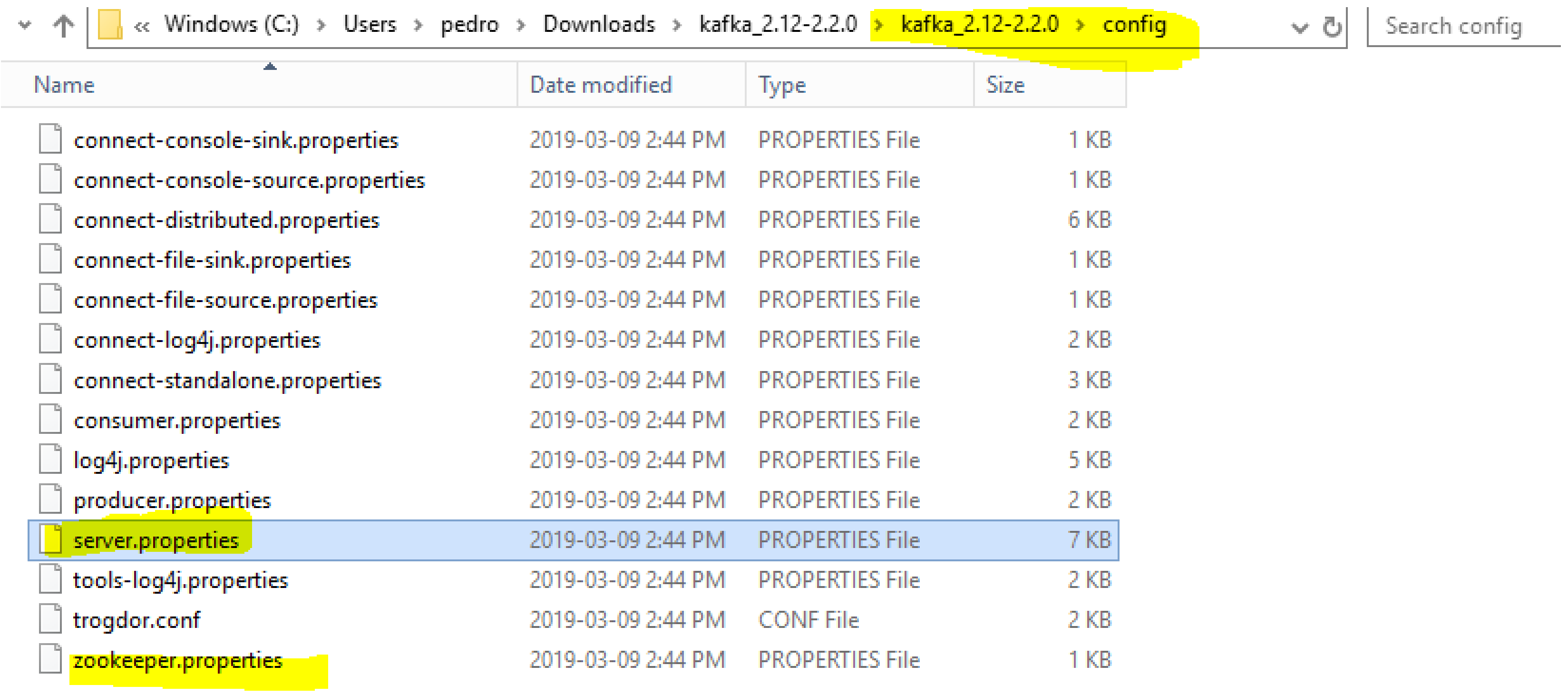
Variable	Value
asl.log	Destination=file
ComSpec	C:\WINDOWS\system32\cmd.exe
CUDA_PATH	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0
CUDA_PATH_V9_0	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	4
NVCUDASAMPLES_ROOT	C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.0

New... | Edit... | Delete

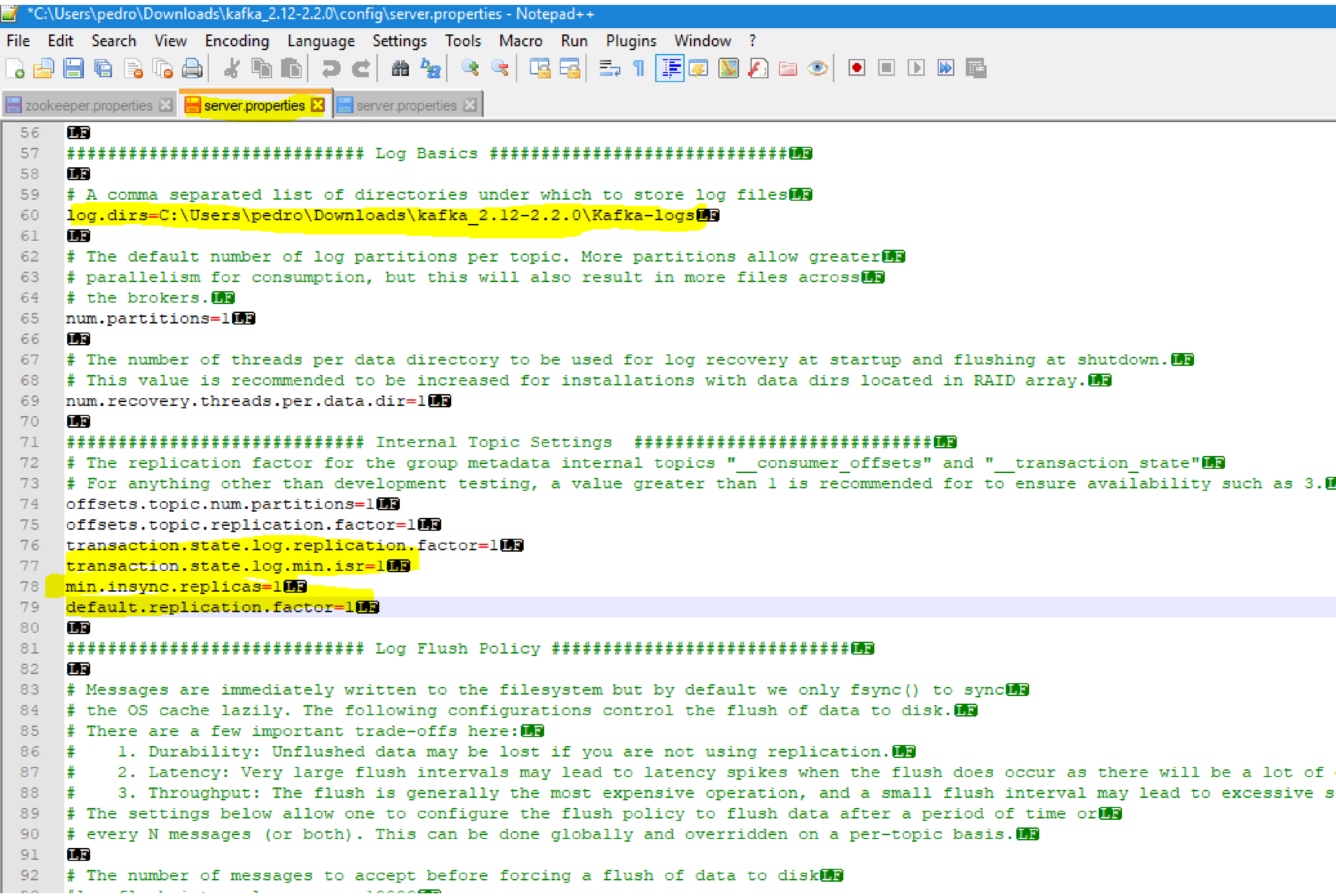
OK | Cancel

This PC › Windows (C:) › Users › pedro › Downloads › kafka_2.12-2.2.0

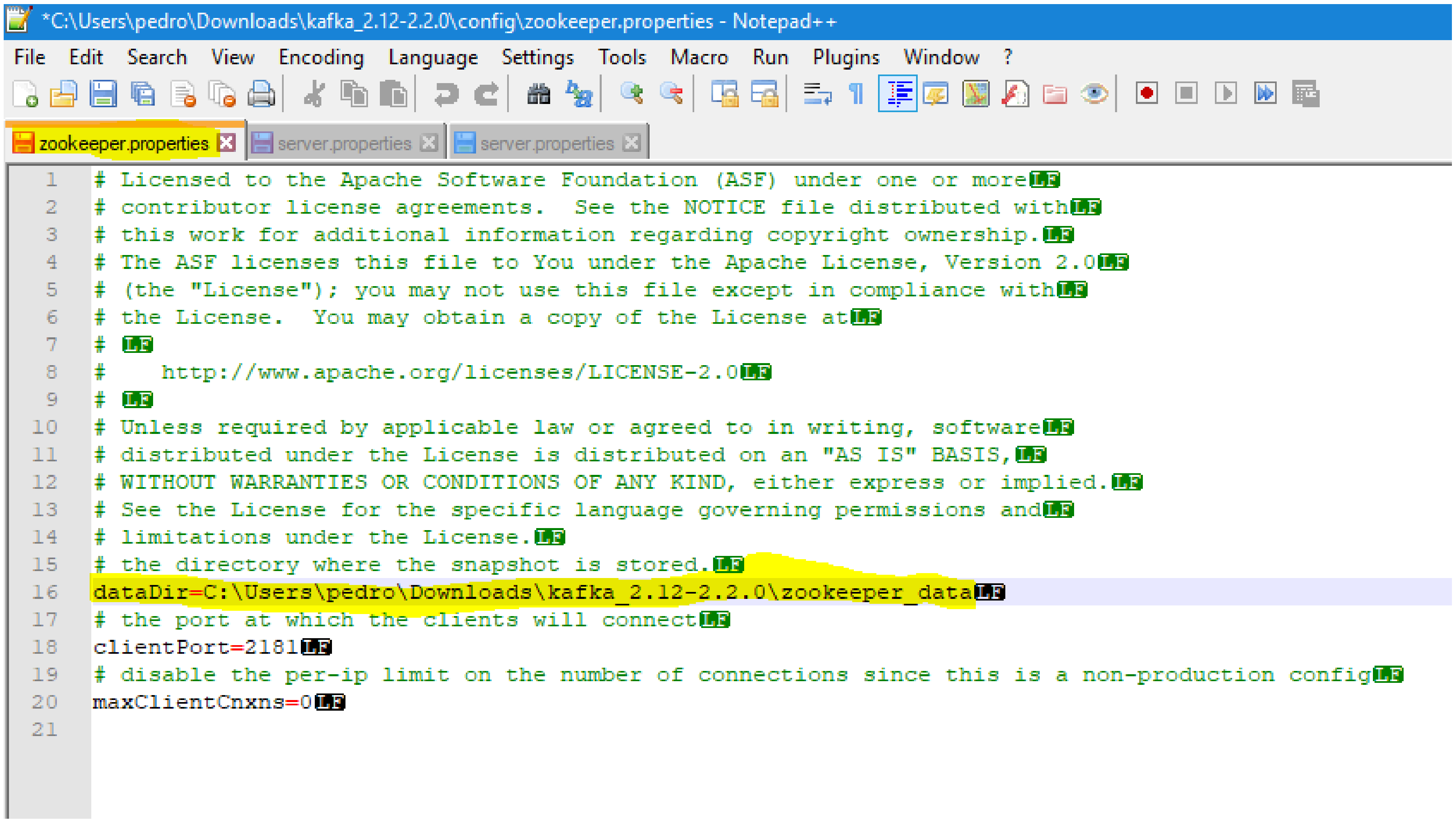
Name	Date modified	Type	Size
bin	2019-03-09 2:46 PM	File folder	
config	2019-03-09 2:46 PM	File folder	
kafka_2.12-2.2.0	2019-04-23 9:48 PM	File folder	
Kafka-logs	2019-04-25 9:53 AM	File folder	
libs	2019-04-23 9:32 PM	File folder	
logs	2019-04-25 10:07 ...	File folder	
site-docs	2019-03-09 2:46 PM	File folder	
zookeeper_data	2019-04-25 9:51 AM	File folder	
LICENSE	2019-03-09 2:44 PM	File	32 KB
NOTICE	2019-03-09 2:44 PM	File	1 KB


« Windows (C:) › Users › pedro › Downloads › kafka_2.12-2.2.0 › kafka_2.12-2.2.0 › config
Search config

Name	Date modified	Type	Size
connect-console-sink.properties	2019-03-09 2:44 PM	PROPERTIES File	1 KB
connect-console-source.properties	2019-03-09 2:44 PM	PROPERTIES File	1 KB
connect-distributed.properties	2019-03-09 2:44 PM	PROPERTIES File	6 KB
connect-file-sink.properties	2019-03-09 2:44 PM	PROPERTIES File	1 KB
connect-file-source.properties	2019-03-09 2:44 PM	PROPERTIES File	1 KB
connect-log4j.properties	2019-03-09 2:44 PM	PROPERTIES File	2 KB
connect-standalone.properties	2019-03-09 2:44 PM	PROPERTIES File	3 KB
consumer.properties	2019-03-09 2:44 PM	PROPERTIES File	2 KB
log4j.properties	2019-03-09 2:44 PM	PROPERTIES File	5 KB
producer.properties	2019-03-09 2:44 PM	PROPERTIES File	2 KB
server.properties	2019-03-09 2:44 PM	PROPERTIES File	7 KB
tools-log4j.properties	2019-03-09 2:44 PM	PROPERTIES File	2 KB
trogdor.conf	2019-03-09 2:44 PM	CONF File	2 KB
zookeeper.properties	2019-03-09 2:44 PM	PROPERTIES File	1 KB

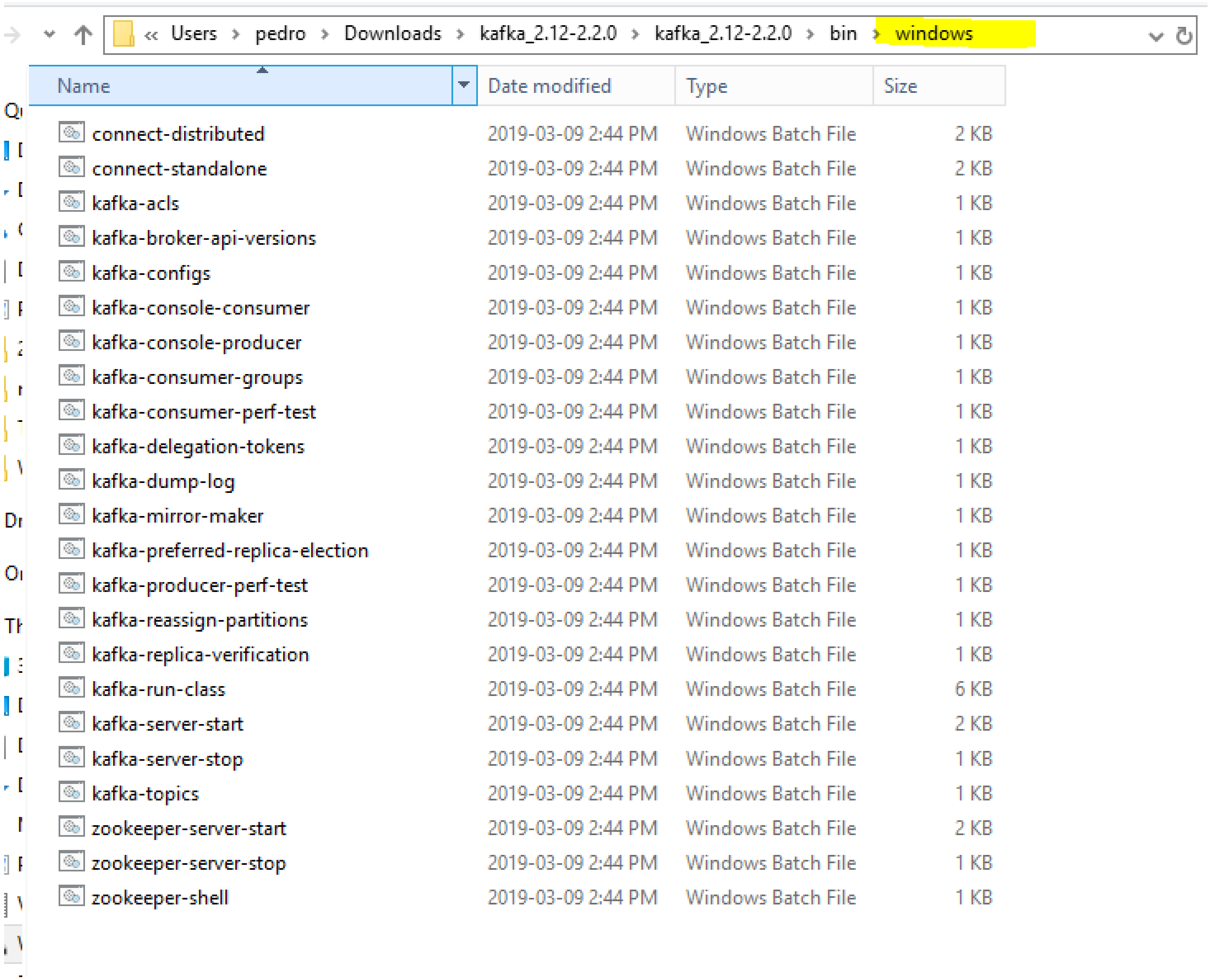

*C:\Users\pedro\Downloads\kafka_2.12-2.2.0\config\server.properties - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
zookeeper.properties
server.properties
server.properties

```

############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=C:\Users\pedro\Downloads\kafka_2.12-2.2.0\Kafka-logs

# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=1

# The number of threads per data directory to be used for log recovery at startup and flushing at shutdown.
# This value is recommended to be increased for installations with data dirs located in RAID array.
num.recovery.threads.per.data.dir=1

############################# Internal Topic Settings  #############################
# The replication factor for the group metadata internal topics "__consumer_offsets" and "__transaction_state"
# For anything other than development testing, a value greater than 1 is recommended for to ensure availability such as 3.
offsets.topic.num.partitions=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
min.insync.replicas=1
default.replication.factor=1

############################# Log Flush Policy #############################

# Messages are immediately written to the filesystem but by default we only fsync() to sync
# the OS cache lazily. The following configurations control the flush of data to disk.
# There are a few important trade-offs here:
#    1. Durability: Unflushed data may be lost if you are not using replication.
#    2. Latency: Very large flush intervals may lead to latency spikes when the flush does occur as there will be a lot of
#    3. Throughput: The flush is generally the most expensive operation, and a small flush interval may lead to excessive s
# The settings below allow one to configure the flush policy to flush data after a period of time or
# every N messages (or both). This can be done globally and overridden on a per-topic basis.

# The number of messages to accept before forcing a flush of data to disk
```



*C:\Users\pedro\Downloads\kafka_2.12-2.2.0\config\zookeeper.properties - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
zookeeper.properties
server.properties
server.properties

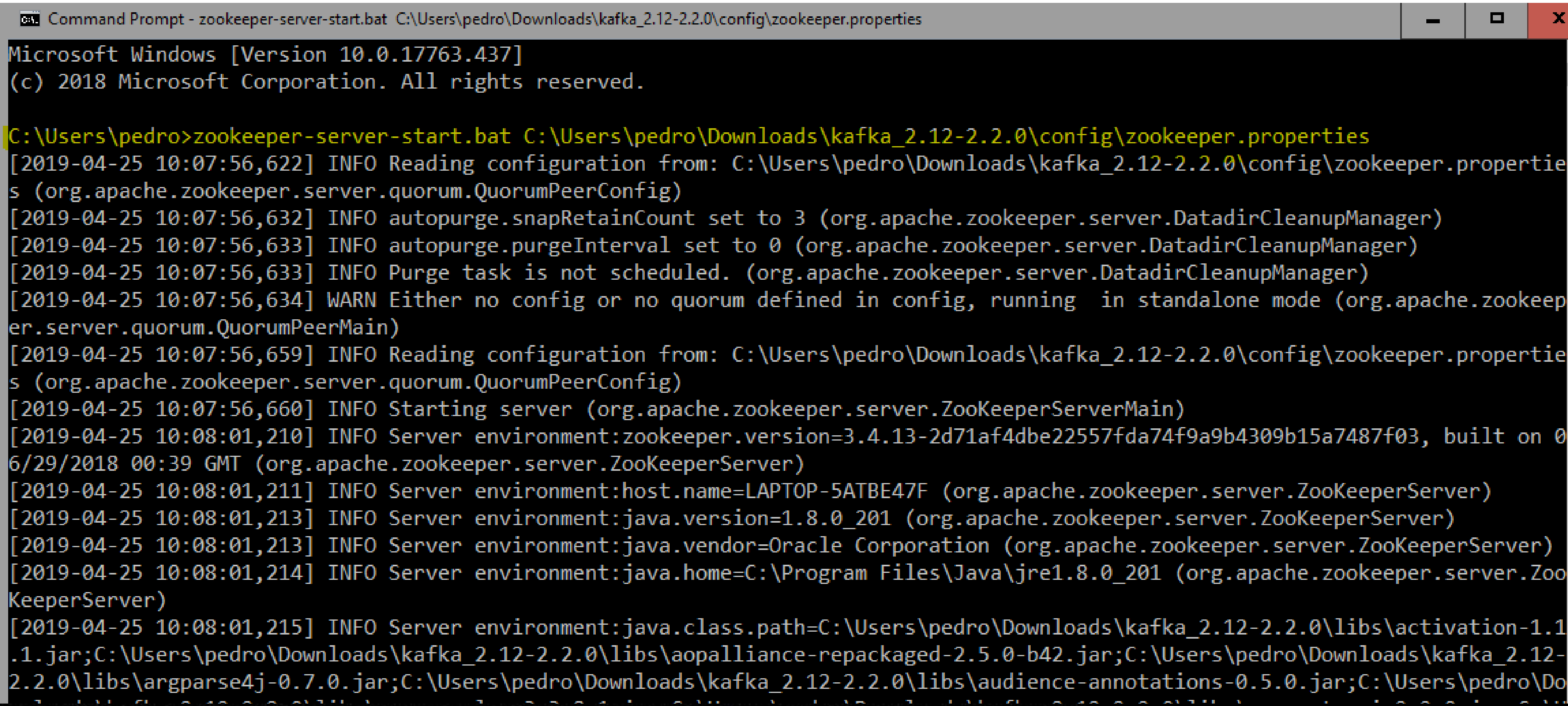
```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
# 
#    http://www.apache.org/licenses/LICENSE-2.0
# 
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# the directory where the snapshot is stored.
dataDir=C:\Users\pedro\Downloads\kafka_2.12-2.2.0\zookeeper_data
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections since this is a non-production config
maxClientCnxns=0
```



« Users › pedro › Downloads › kafka_2.12-2.2.0 › kafka_2.12-2.2.0 › bin › windows

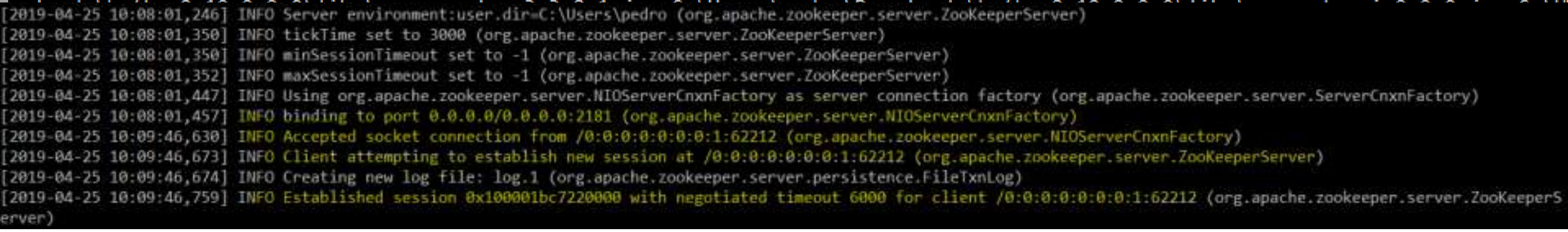
Name	Date modified	Type	Size
connect-distributed	2019-03-09 2:44 PM	Windows Batch File	2 KB
connect-standalone	2019-03-09 2:44 PM	Windows Batch File	2 KB
kafka-acls	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-broker-api-versions	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-configs	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-console-consumer	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-console-producer	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-consumer-groups	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-consumer-perf-test	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-delegation-tokens	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-dump-log	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-mirror-maker	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-preferred-replica-election	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-producer-perf-test	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-reassign-partitions	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-replica-verification	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-run-class	2019-03-09 2:44 PM	Windows Batch File	6 KB
kafka-server-start	2019-03-09 2:44 PM	Windows Batch File	2 KB
kafka-server-stop	2019-03-09 2:44 PM	Windows Batch File	1 KB
kafka-topics	2019-03-09 2:44 PM	Windows Batch File	1 KB
zookeeper-server-start	2019-03-09 2:44 PM	Windows Batch File	2 KB
zookeeper-server-stop	2019-03-09 2:44 PM	Windows Batch File	1 KB
zookeeper-shell	2019-03-09 2:44 PM	Windows Batch File	1 KB

Start zookeeper then Kafka server



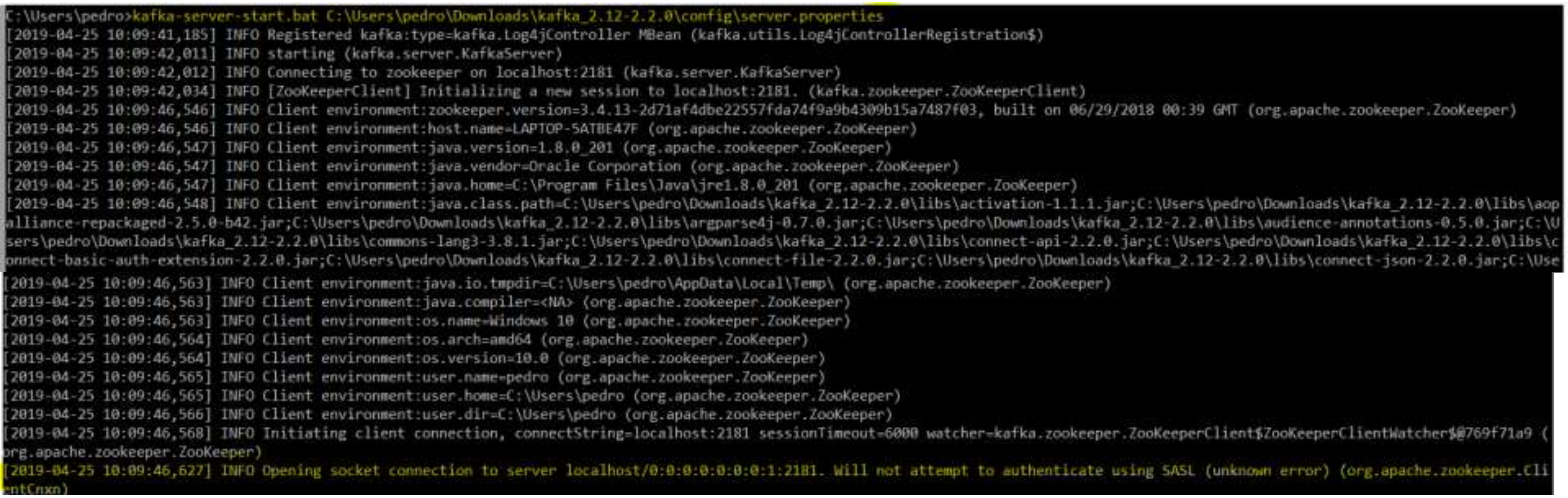
Command Prompt - zookeeper-server-start.bat C:\Users\pedro\Downloads\kafka_2.12-2.2.0\config\zookeeper.properties

```
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\pedro>zookeeper-server-start.bat C:\Users\pedro\Downloads\kafka_2.12-2.2.0\config\zookeeper.properties
[2019-04-25 10:07:56,622] INFO Reading configuration from: C:\Users\pedro\Downloads\kafka_2.12-2.2.0\config\zookeeper.propertie
s (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2019-04-25 10:07:56,632] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2019-04-25 10:07:56,633] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2019-04-25 10:07:56,633] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2019-04-25 10:07:56,634] WARN Either no config or no quorum defined in config, running  in standalone mode (org.apache.zookeep
er.server.quorum.QuorumPeerMain)
[2019-04-25 10:07:56,659] INFO Reading configuration from: C:\Users\pedro\Downloads\kafka_2.12-2.2.0\config\zookeeper.propertie
s (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2019-04-25 10:07:56,660] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2019-04-25 10:08:01,210] INFO Server environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 0
6/29/2018 00:39 GMT (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:08:01,211] INFO Server environment:host.name=LAPTOP-5ATBE47F (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:08:01,213] INFO Server environment:java.version=1.8.0_201 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:08:01,213] INFO Server environment:java.vendor=Oracle Corporation (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:08:01,214] INFO Server environment:java.home=C:\Program Files\Java\jre1.8.0_201 (org.apache.zookeeper.server.Zoo
KeeperServer)
[2019-04-25 10:08:01,215] INFO Server environment:java.class.path=C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\activation-1.1
.1.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\aopalliance-repackaged-2.5.0-b42.jar;C:\Users\pedro\Downloads\kafka_2.12-
2.2.0\libs\argparse4j-0.7.0.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\audience-annotations-0.5.0.jar;C:\Users\pedro\Do
```

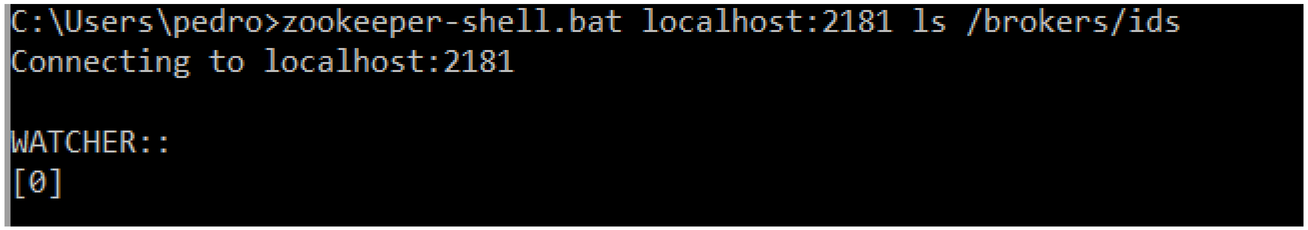


```
[2019-04-25 10:08:01,246] INFO Server environment:user.dir=C:\Users\pedro (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:08:01,350] INFO tickTime set to 3000 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:08:01,350] INFO minSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:08:01,352] INFO maxSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:08:01,447] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apache.zookeeper.server.ServerCnxnFactory)
[2019-04-25 10:08:01,457] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2019-04-25 10:09:46,630] INFO Accepted socket connection from /0:0:0:0:0:0:0:1:62212 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2019-04-25 10:09:46,673] INFO Client attempting to establish new session at /0:0:0:0:0:0:0:1:62212 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-04-25 10:09:46,674] INFO Creating new log file: log.1 (org.apache.zookeeper.server.persistence.FileTxnLog)
[2019-04-25 10:09:46,759] INFO Established session 0x100001bc7220000 with negotiated timeout 6000 for client /0:0:0:0:0:0:0:1:62212 (org.apache.zookeeper.server.ZooKeeperS
erver)
```

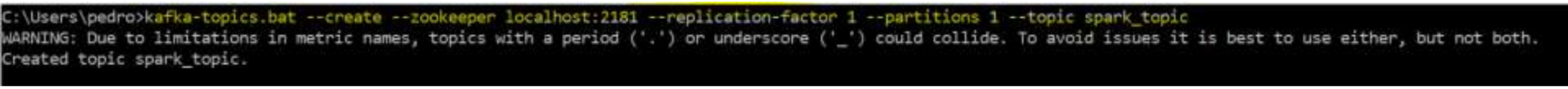
<span style="color:red">Start zookeeper then Kafka server</span>



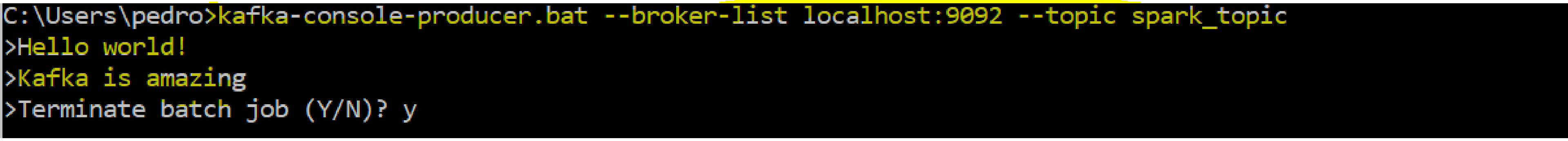
```
C:\Users\pedro>kafka-server-start.bat C:\Users\pedro\Downloads\kafka_2.12-2.2.0\config\server.properties
[2019-04-25 10:09:41,185] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2019-04-25 10:09:42,011] INFO starting (kafka.server.KafkaServer)
[2019-04-25 10:09:42,012] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2019-04-25 10:09:42,034] INFO [ZooKeeperClient] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2019-04-25 10:09:46,546] INFO Client environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 00:39 GMT (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,546] INFO Client environment:host.name=LAPTOP-5ATBE47F (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,547] INFO Client environment:java.version=1.8.0_201 (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,547] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,547] INFO Client environment:java.home=C:\Program Files\Java\jre1.8.0_201 (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,548] INFO Client environment:java.class.path=C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\activation-1.1.1.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\aop
alliance-repackaged-2.5.0-b42.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\argparse4j-0.7.0.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\audience-annotations-0.5.0.jar;C:\U
sers\pedro\Downloads\kafka_2.12-2.2.0\libs\commons-lang3-3.8.1.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\connect-api-2.2.0.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\c
onnect-basic-auth-extension-2.2.0.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\connect-file-2.2.0.jar;C:\Users\pedro\Downloads\kafka_2.12-2.2.0\libs\connect-json-2.2.0.jar;C:\Use
[2019-04-25 10:09:46,563] INFO Client environment:java.io.tmpdir=C:\Users\pedro\AppData\Local\Temp\ (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,563] INFO Client environment:java.compiler=<NA> (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,563] INFO Client environment:os.name=Windows 10 (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,564] INFO Client environment:os.arch=amd64 (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,564] INFO Client environment:os.version=10.0 (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,565] INFO Client environment:user.name=pedro (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,565] INFO Client environment:user.home=C:\Users\pedro (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,566] INFO Client environment:user.dir=C:\Users\pedro (org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,568] INFO Initiating client connection, connectString=localhost:2181 sessionTimeout=6000 watcher=kafka.zookeeper.ZooKeeperClient$ZooKeeperClientWatcher$@769f71a9 (
org.apache.zookeeper.ZooKeeper)
[2019-04-25 10:09:46,627] INFO Opening socket connection to server localhost/0:0:0:0:0:0:0:1:2181. Will not attempt to authenticate using SASL (unknown error) (org.apache.zookeeper.Cli
entCnxn)
```



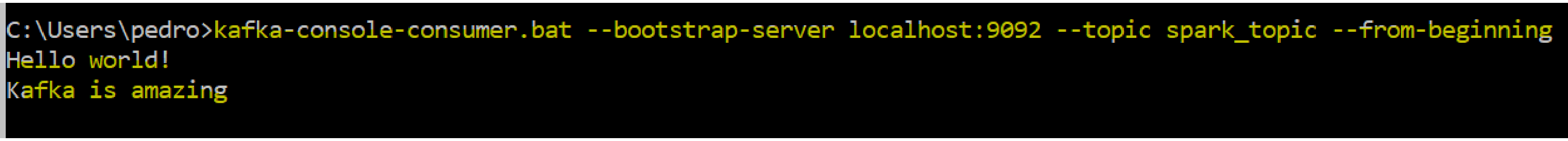
```
C:\Users\pedro>zookeeper-shell.bat localhost:2181 ls /brokers/ids
Connecting to localhost:2181

WATCHER::
[0]
```



```
C:\Users\pedro>kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic spark_topic
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic spark_topic.
```

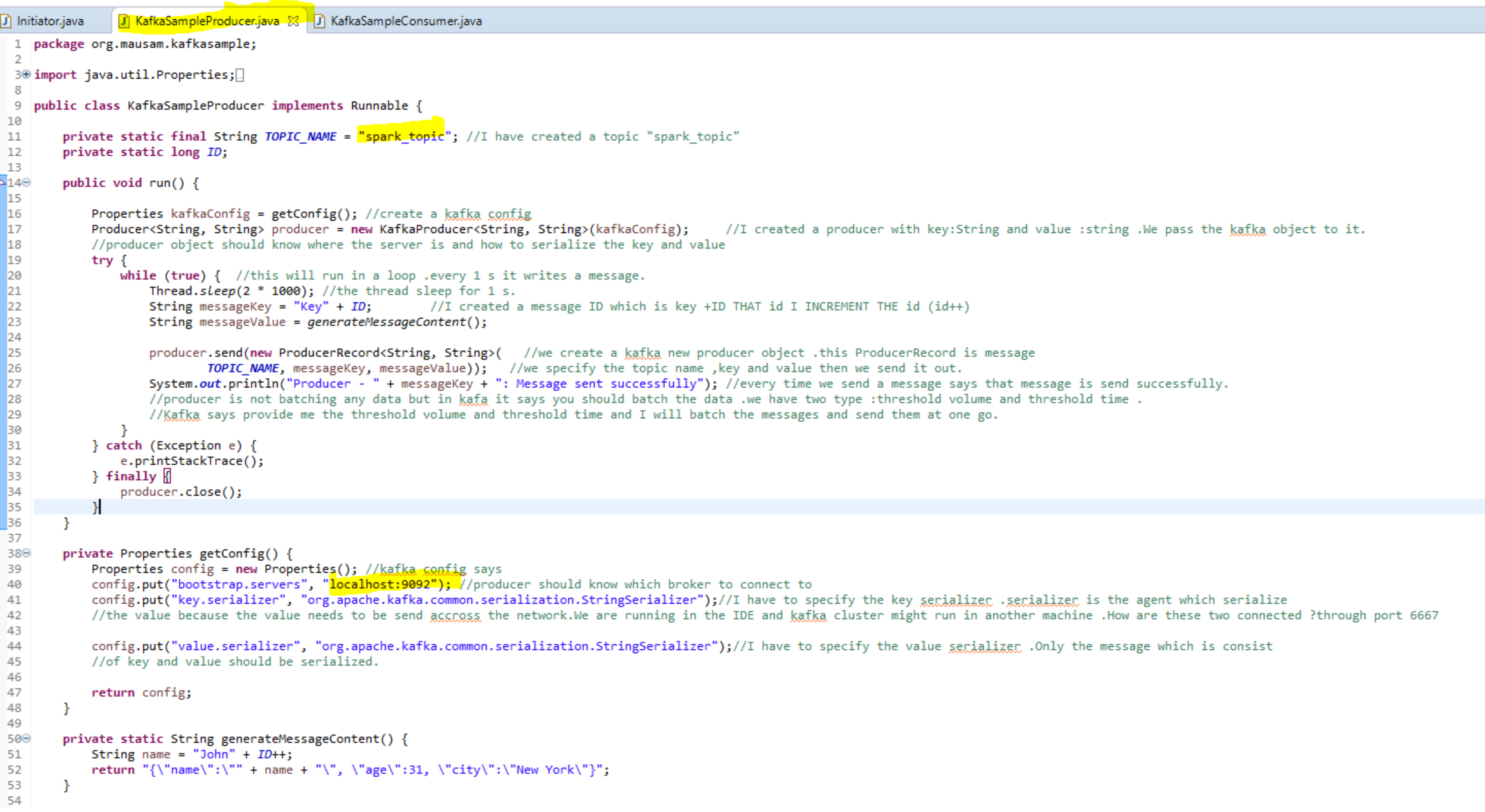


```
C:\Users\pedro>kafka-console-producer.bat --broker-list localhost:9092 --topic spark_topic
>Hello world!
>Kafka is amazing
>Terminate batch job (Y/N)? y
```

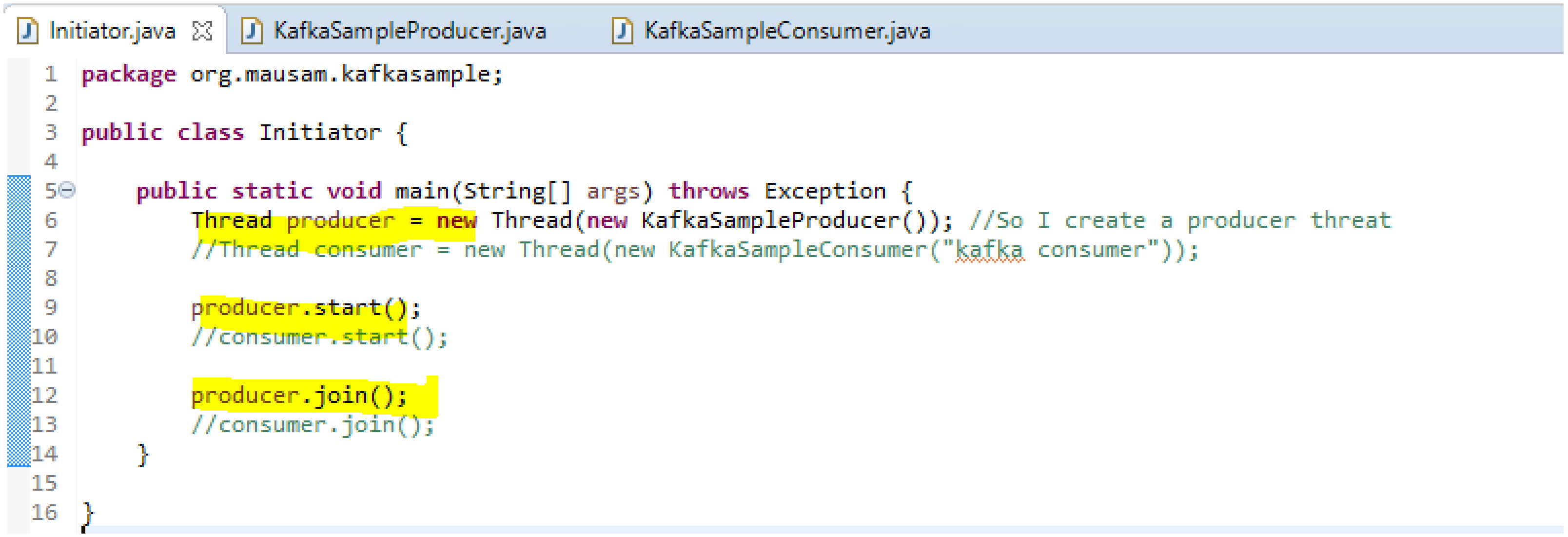


```
C:\Users\pedro>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic spark_topic --from-beginning
Hello world!
Kafka is amazing
```

Console consumer /App producer

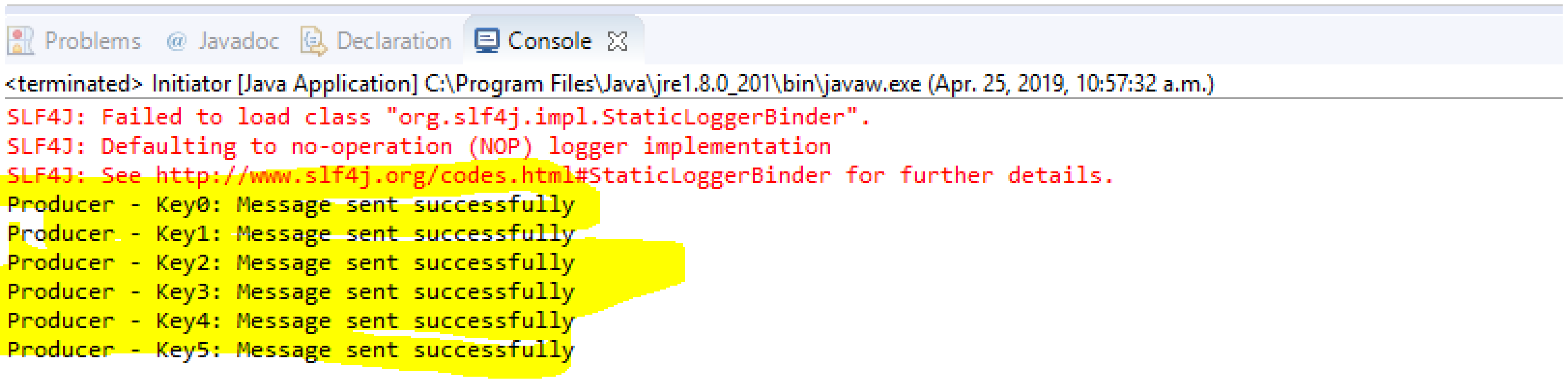


```java
package org.mausam.kafkasample;

import java.util.Properties;

public class KafkaSampleProducer implements Runnable {

    private static final String TOPIC_NAME = "spark_topic"; //I have created a topic "spark_topic"
    private static long ID;

    public void run() {

        Properties kafkaConfig = getConfig(); //create a kafka config
        Producer<String, String> producer = new KafkaProducer<String, String>(kafkaConfig);     //I created a producer with key:String and value :string .We pass the kafka object to it.
        //producer object should know where the server is and how to serialize the key and value
        try {
            while (true) {  //this will run in a loop .every 1 s it writes a message.
                Thread.sleep(2 * 1000); //the thread sleep for 1 s.
                String messageKey = "Key" + ID;        //I created a message ID which is key +ID THAT id I INCREMENT THE id (id++)
                String messageValue = generateMessageContent();

                producer.send(new ProducerRecord<String, String>(   //we create a kafka new producer object .this ProducerRecord is message
                        TOPIC_NAME, messageKey, messageValue));   //we specify the topic name ,key and value then we send it out.
                System.out.println("Producer - " + messageKey + ": Message sent successfully"); //every time we send a message says that message is send successfully.
                //producer is not batching any data but in kafa it says you should batch the data .we have two type :threshold volume and threshold time .
                //Kafka says provide me the threshold volume and threshold time and I will batch the messages and send them at one go.
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            producer.close();
        }
    }

    private Properties getConfig() {
        Properties config = new Properties(); //kafka config says
        config.put("bootstrap.servers", "localhost:9092"); //producer should know which broker to connect to
        config.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");//I have to specify the key serializer .serializer is the agent which serialize
        //the value because the value needs to be send accross the network.We are running in the IDE and kafka cluster might run in another machine .How are these two connected ?through port 6667

        config.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");//I have to specify the value serializer .Only the message which is consist
        //of key and value should be serialized.

        return config;
    }

    private static String generateMessageContent() {
        String name = "John" + ID++;
        return "{\"name\":\"" + name + "\", \"age\":31, \"city\":\"New York\"}";
    }
```

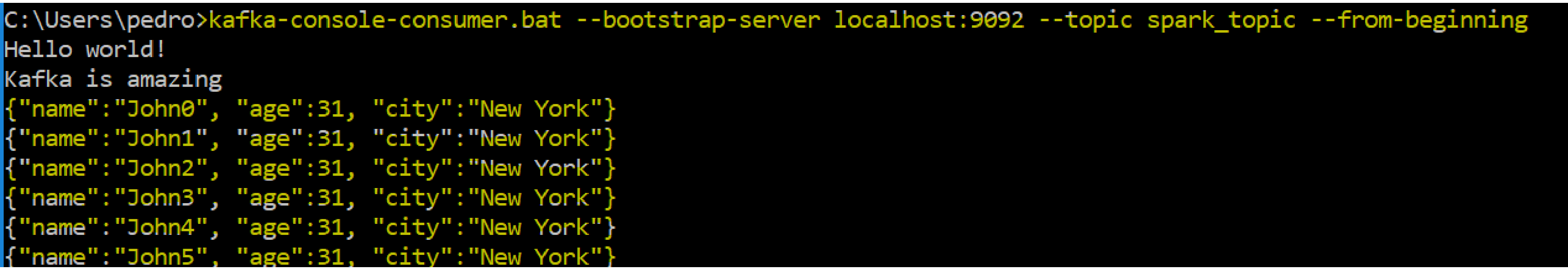


Initiator.java   KafkaSampleProducer.java   KafkaSampleConsumer.java

```java
package org.mausam.kafkasample;

public class Initiator {

    public static void main(String[] args) throws Exception {
        Thread producer = new Thread(new KafkaSampleProducer()); //So I create a producer threat
        //Thread consumer = new Thread(new KafkaSampleConsumer("kafka consumer"));

        producer.start();
        //consumer.start();

        producer.join();
        //consumer.join();
    }

}
```

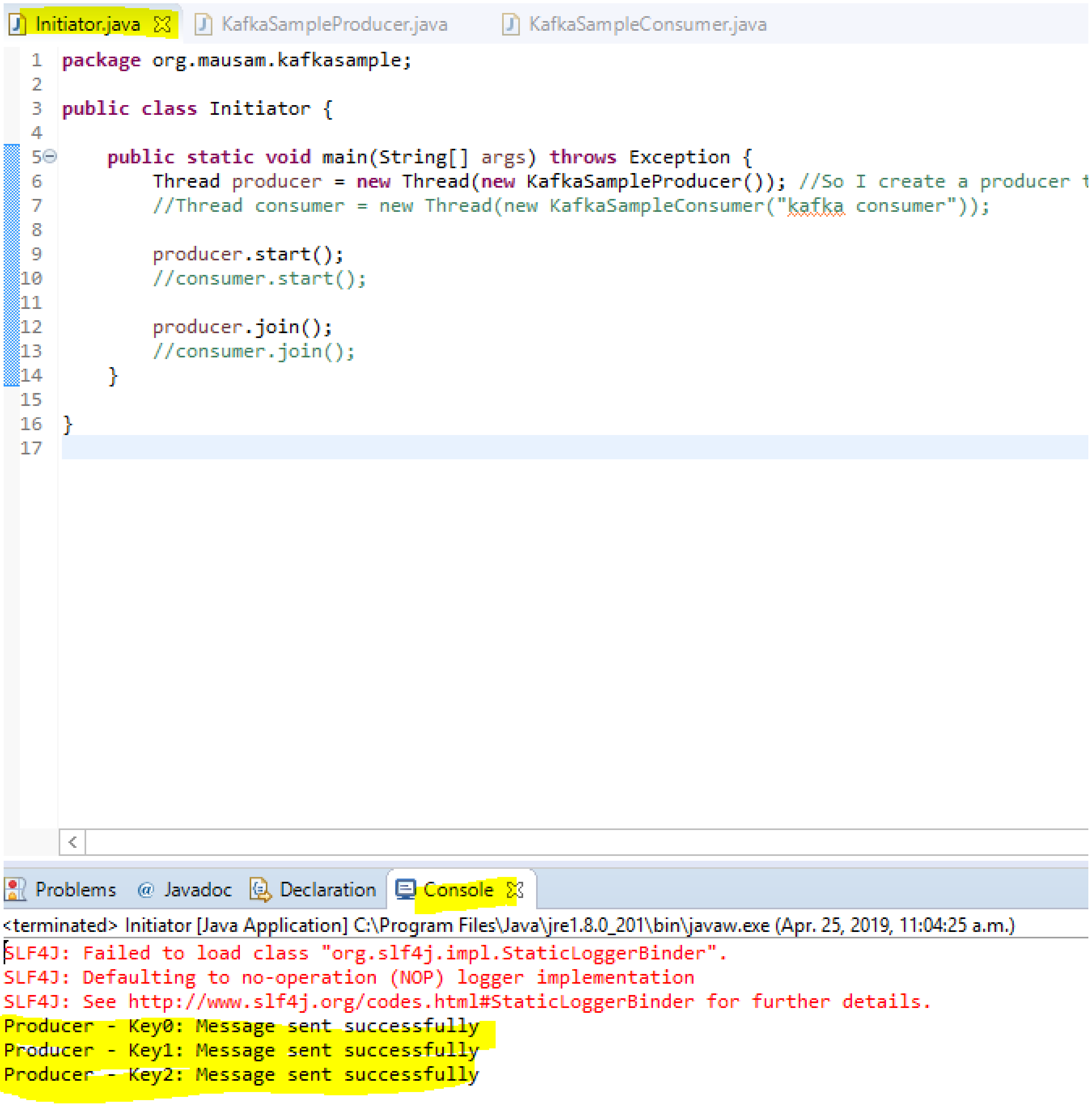


```
<terminated> Initiator [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Apr. 25, 2019, 10:57:32 a.m.)
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Producer - Key0: Message sent successfully
Producer - Key1: Message sent successfully
Producer - Key2: Message sent successfully
Producer - Key3: Message sent successfully
Producer - Key4: Message sent successfully
Producer - Key5: Message sent successfully
```



```
C:\Users\pedro>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic spark_topic --from-beginning
Hello world!
Kafka is amazing
{"name":"John0", "age":31, "city":"New York"}
{"name":"John1", "age":31, "city":"New York"}
{"name":"John2", "age":31, "city":"New York"}
{"name":"John3", "age":31, "city":"New York"}
{"name":"John4", "age":31, "city":"New York"}
{"name":"John5", "age":31, "city":"New York"}
```

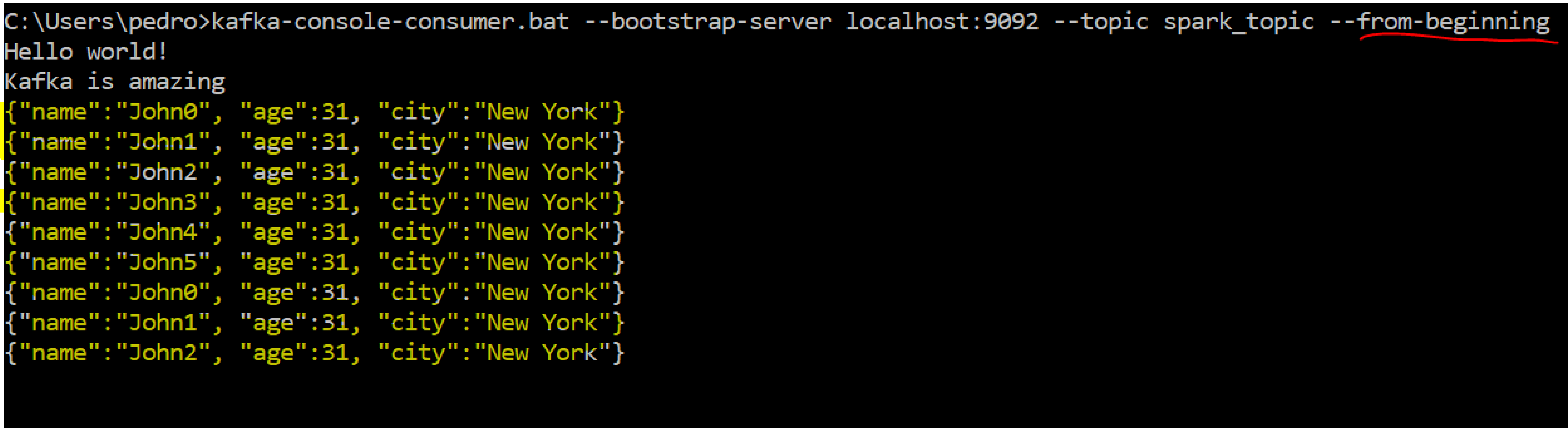
If I stop



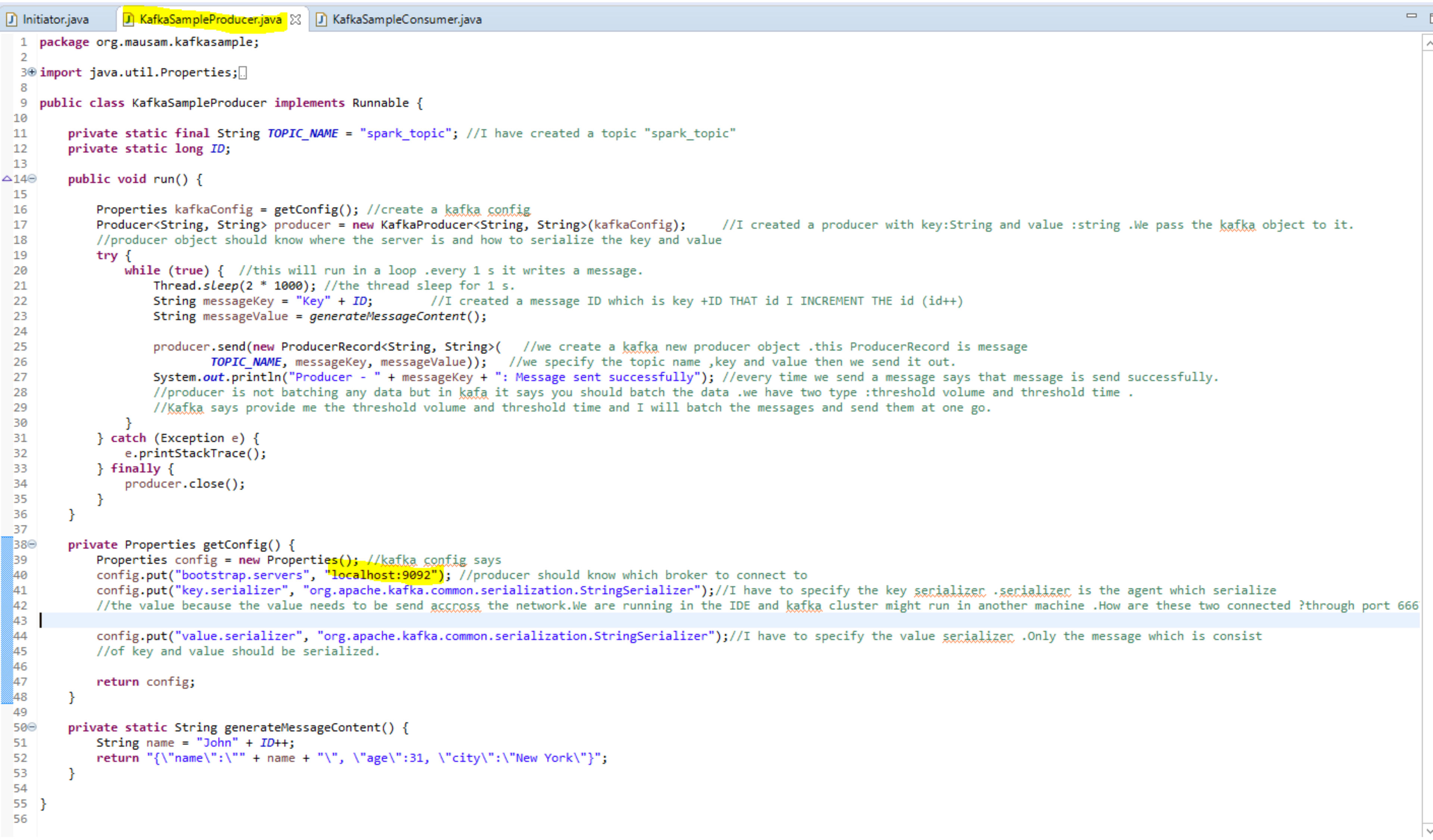
```
package org.mausam.kafkasample;

public class Initiator {

    public static void main(String[] args) throws Exception {
        Thread producer = new Thread(new KafkaSampleProducer()); //So I create a producer t
        //Thread consumer = new Thread(new KafkaSampleConsumer("kafka consumer"));

        producer.start();
        //consumer.start();

        producer.join();
        //consumer.join();
    }

}
```

```
<terminated> Initiator [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Apr. 25, 2019, 11:04:25 a.m.)
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Producer - Key0: Message sent successfully
Producer - Key1: Message sent successfully
Producer - Key2: Message sent successfully
```

We get all the messages .

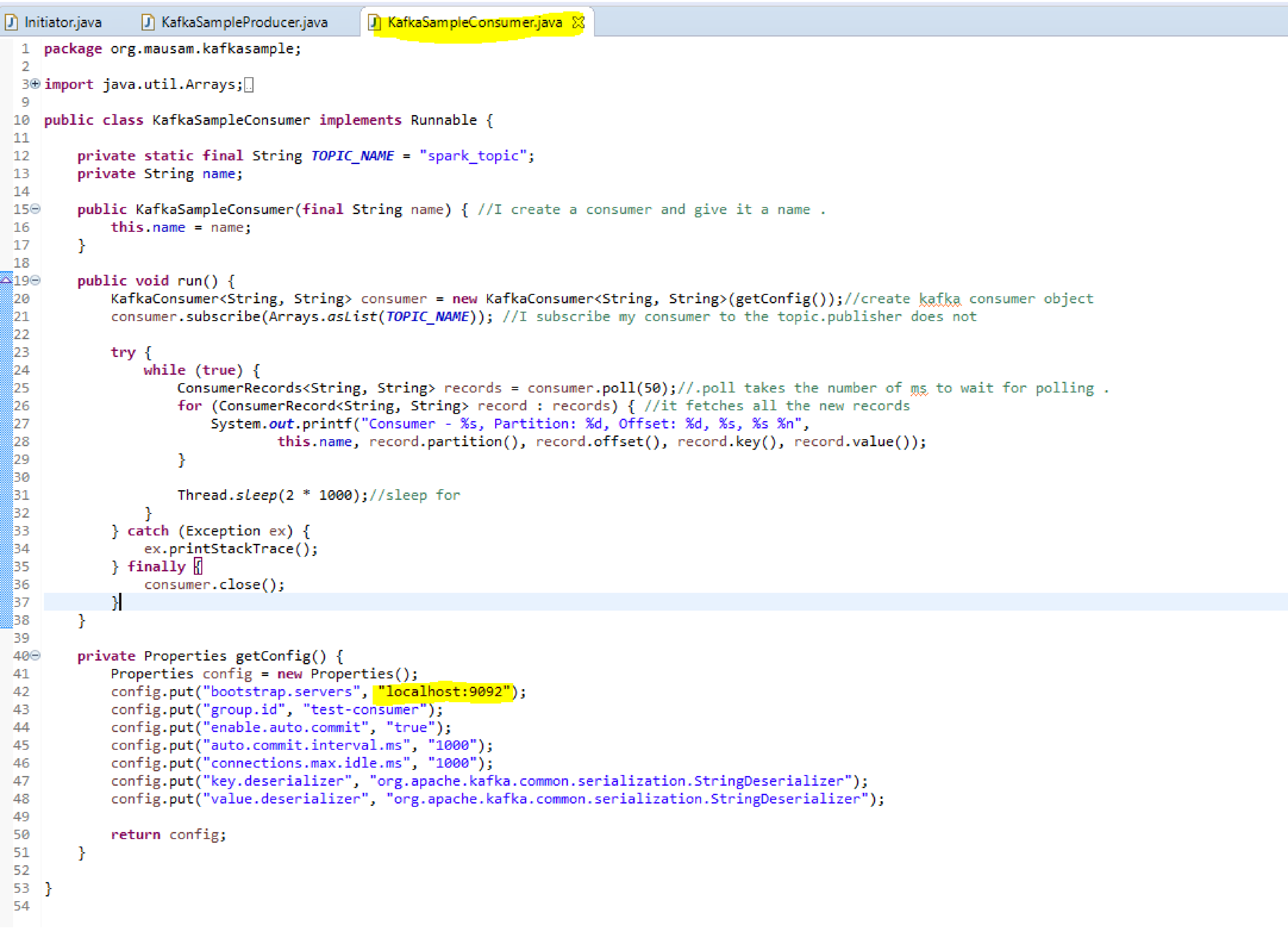

```
C:\Users\pedro>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic spark_topic --from-beginning
Hello world!
Kafka is amazing
{"name":"John0", "age":31, "city":"New York"}
{"name":"John1", "age":31, "city":"New York"}
{"name":"John2", "age":31, "city":"New York"}
{"name":"John3", "age":31, "city":"New York"}
{"name":"John4", "age":31, "city":"New York"}
{"name":"John5", "age":31, "city":"New York"}
{"name":"John0", "age":31, "city":"New York"}
{"name":"John1", "age":31, "city":"New York"}
{"name":"John2", "age":31, "city":"New York"}
```

App consumer /App producer



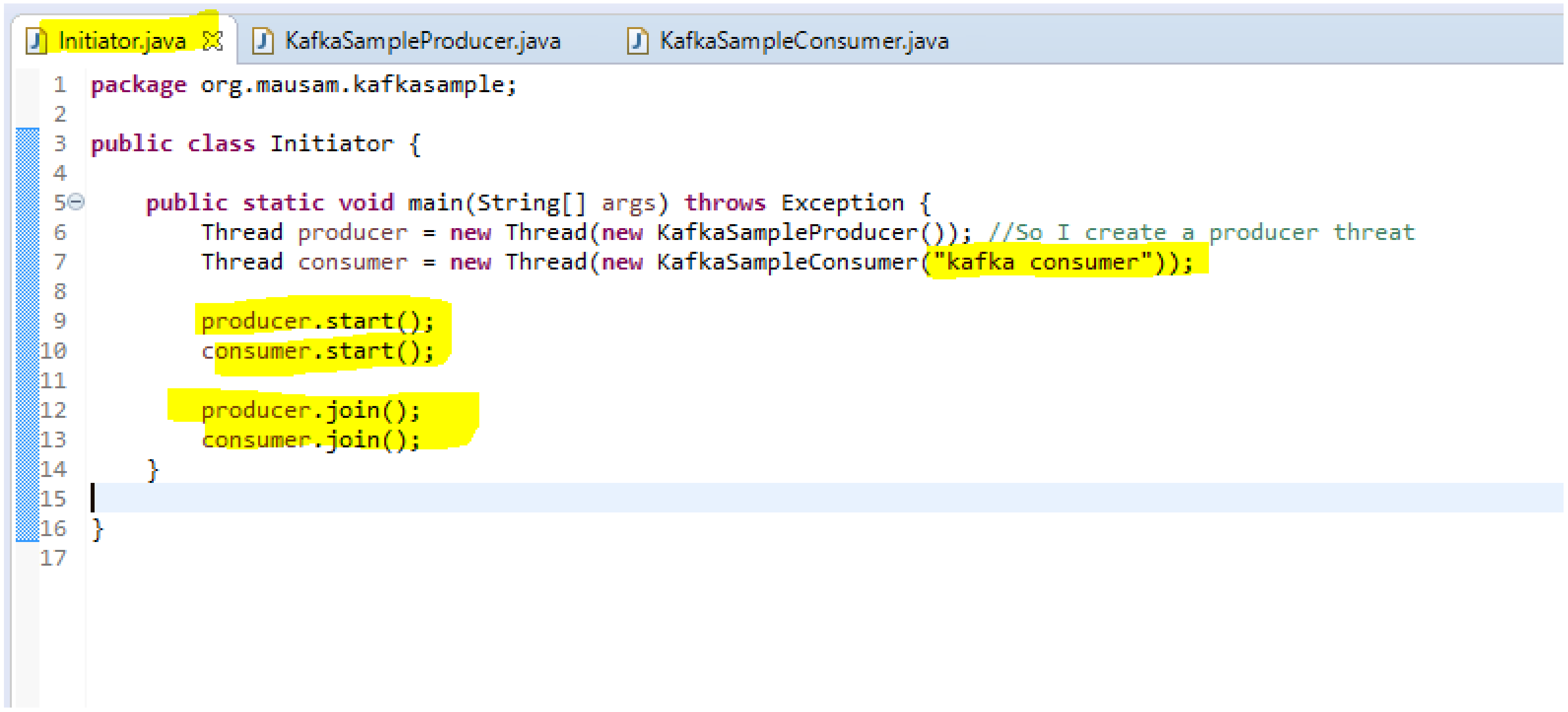
Initiator.java | KafkaSampleProducer.java | KafkaSampleConsumer.java

```java
package org.mausam.kafkasample;

import java.util.Properties;

public class KafkaSampleProducer implements Runnable {

    private static final String TOPIC_NAME = "spark_topic"; //I have created a topic "spark_topic"
    private static long ID;

    public void run() {

        Properties kafkaConfig = getConfig(); //create a kafka config
        Producer<String, String> producer = new KafkaProducer<String, String>(kafkaConfig);     //I created a producer with key:String and value :string .We pass the kafka object to it.
        //producer object should know where the server is and how to serialize the key and value
        try {
            while (true) {  //this will run in a loop .every 1 s it writes a message.
                Thread.sleep(2 * 1000); //the thread sleep for 1 s.
                String messageKey = "Key" + ID;      //I created a message ID which is key +ID THAT id I INCREMENT THE id (id++)
                String messageValue = generateMessageContent();

                producer.send(new ProducerRecord<String, String>(   //we create a kafka new producer object .this ProducerRecord is message
                        TOPIC_NAME, messageKey, messageValue));   //we specify the topic name ,key and value then we send it out.
                System.out.println("Producer - " + messageKey + ": Message sent successfully"); //every time we send a message says that message is send successfully.
                //producer is not batching any data but in kafa it says you should batch the data .we have two type :threshold volume and threshold time .
                //Kafka says provide me the threshold volume and threshold time and I will batch the messages and send them at one go.
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            producer.close();
        }
    }

    private Properties getConfig() {
        Properties config = new Properties(); //kafka config says
        config.put("bootstrap.servers", "localhost:9092"); //producer should know which broker to connect to
        config.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");//I have to specify the key serializer .serializer is the agent which serialize
        //the value because the value needs to be send accross the network.We are running in the IDE and kafka cluster might run in another machine .How are these two connected ?through port 666

        config.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");//I have to specify the value serializer .Only the message which is consist
        //of key and value should be serialized.

        return config;
    }

    private static String generateMessageContent() {
        String name = "John" + ID++;
        return "{\"name\":\"" + name + "\", \"age\":31, \"city\":\"New York\"}";
    }

}
```

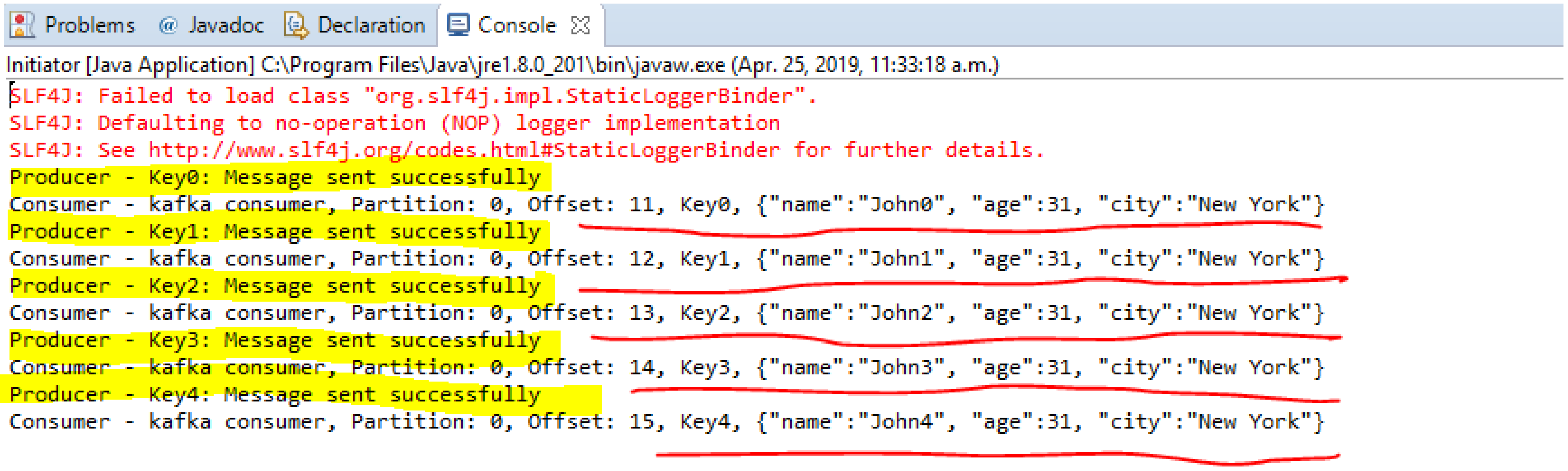


Initiator.java KafkaSampleProducer.java KafkaSampleConsumer.java

```java
package org.mausam.kafkasample;

import java.util.Arrays;

public class KafkaSampleConsumer implements Runnable {

    private static final String TOPIC_NAME = "spark_topic";
    private String name;

    public KafkaSampleConsumer(final String name) { //I create a consumer and give it a name .
        this.name = name;
    }

    public void run() {
        KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(getConfig());//create kafka consumer object
        consumer.subscribe(Arrays.asList(TOPIC_NAME)); //I subscribe my consumer to the topic.publisher does not

        try {
            while (true) {
                ConsumerRecords<String, String> records = consumer.poll(50);//.poll takes the number of ms to wait for polling .
                for (ConsumerRecord<String, String> record : records) { //it fetches all the new records
                    System.out.printf("Consumer - %s, Partition: %d, Offset: %d, %s, %s %n",
                            this.name, record.partition(), record.offset(), record.key(), record.value());
                }

                Thread.sleep(2 * 1000);//sleep for
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            consumer.close();
        }
    }

    private Properties getConfig() {
        Properties config = new Properties();
        config.put("bootstrap.servers", "localhost:9092");
        config.put("group.id", "test-consumer");
        config.put("enable.auto.commit", "true");
        config.put("auto.commit.interval.ms", "1000");
        config.put("connections.max.idle.ms", "1000");
        config.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        config.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

        return config;
    }

}
```

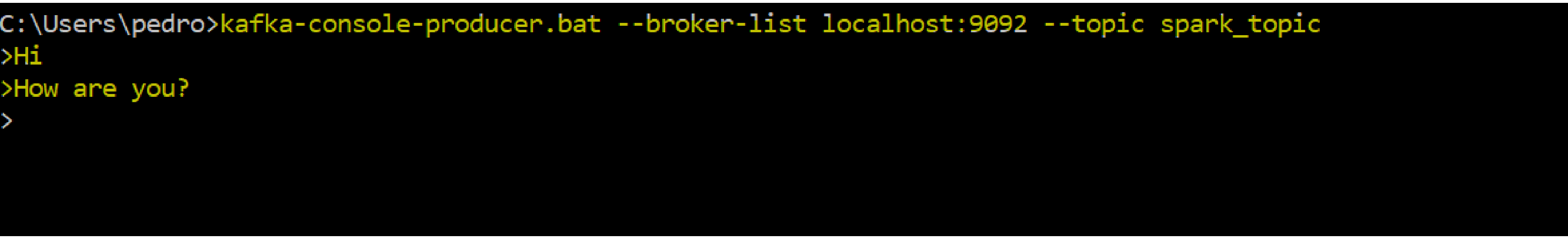


```java
package org.mausam.kafkasample;

public class Initiator {

    public static void main(String[] args) throws Exception {
        Thread producer = new Thread(new KafkaSampleProducer()); //So I create a producer threat
        Thread consumer = new Thread(new KafkaSampleConsumer("kafka consumer"));

        producer.start();
        consumer.start();

        producer.join();
        consumer.join();
    }

}
```

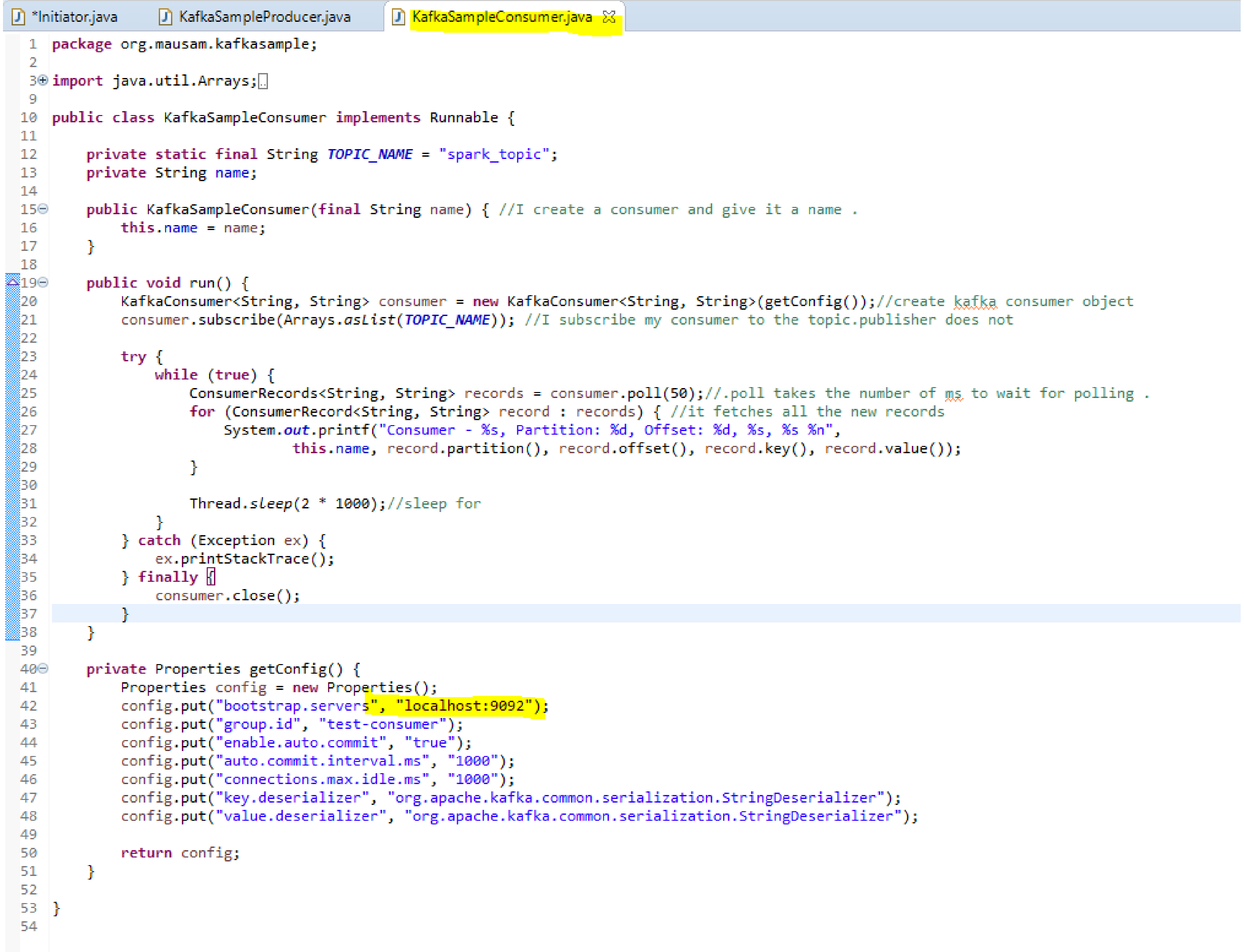


```
Initiator [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Apr. 25, 2019, 11:33:18 a.m.)
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Producer - Key0: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 11, Key0, {"name":"John0", "age":31, "city":"New York"}
Producer - Key1: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 12, Key1, {"name":"John1", "age":31, "city":"New York"}
Producer - Key2: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 13, Key2, {"name":"John2", "age":31, "city":"New York"}
Producer - Key3: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 14, Key3, {"name":"John3", "age":31, "city":"New York"}
Producer - Key4: Message sent successfully
Consumer - kafka consumer, Partition: 0, Offset: 15, Key4, {"name":"John4", "age":31, "city":"New York"}
```

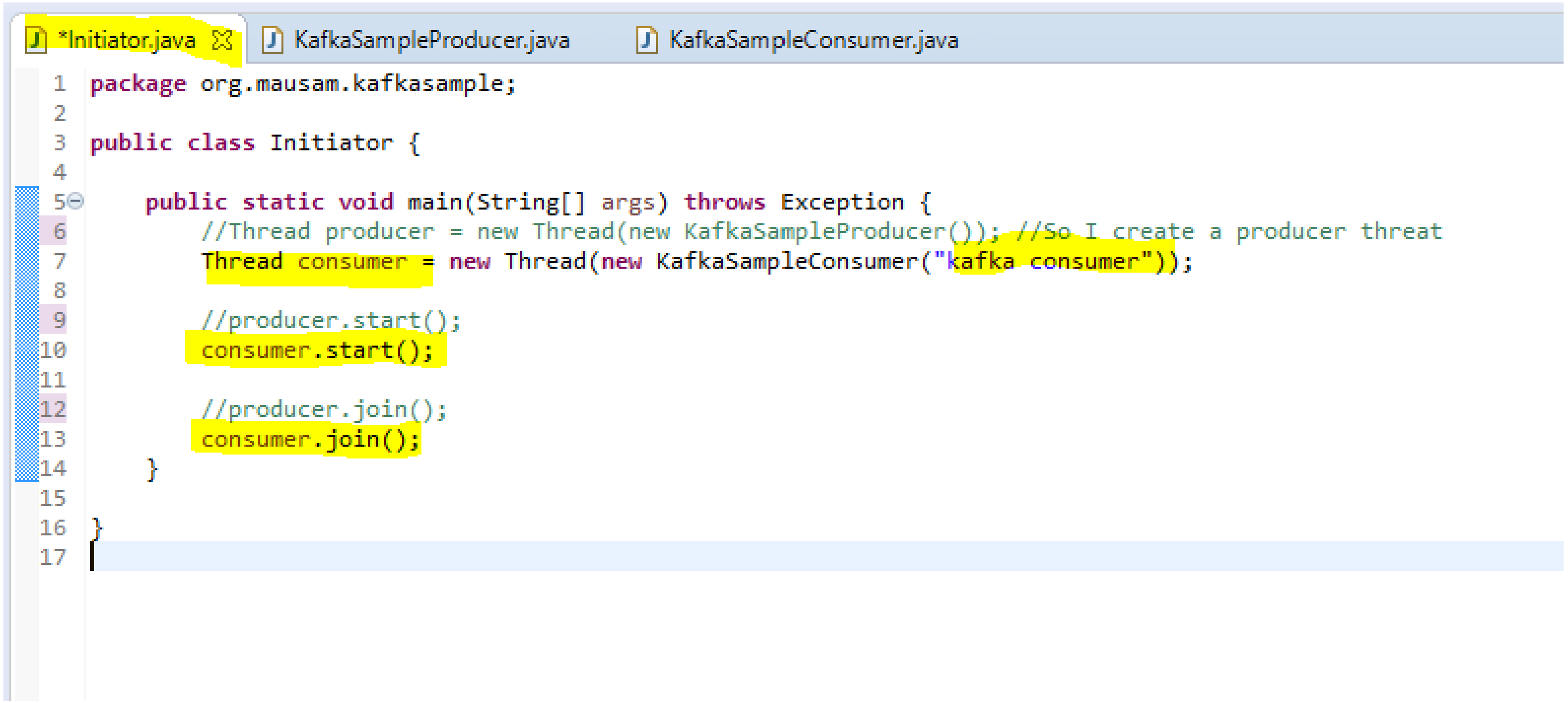
App consumer / Console producer



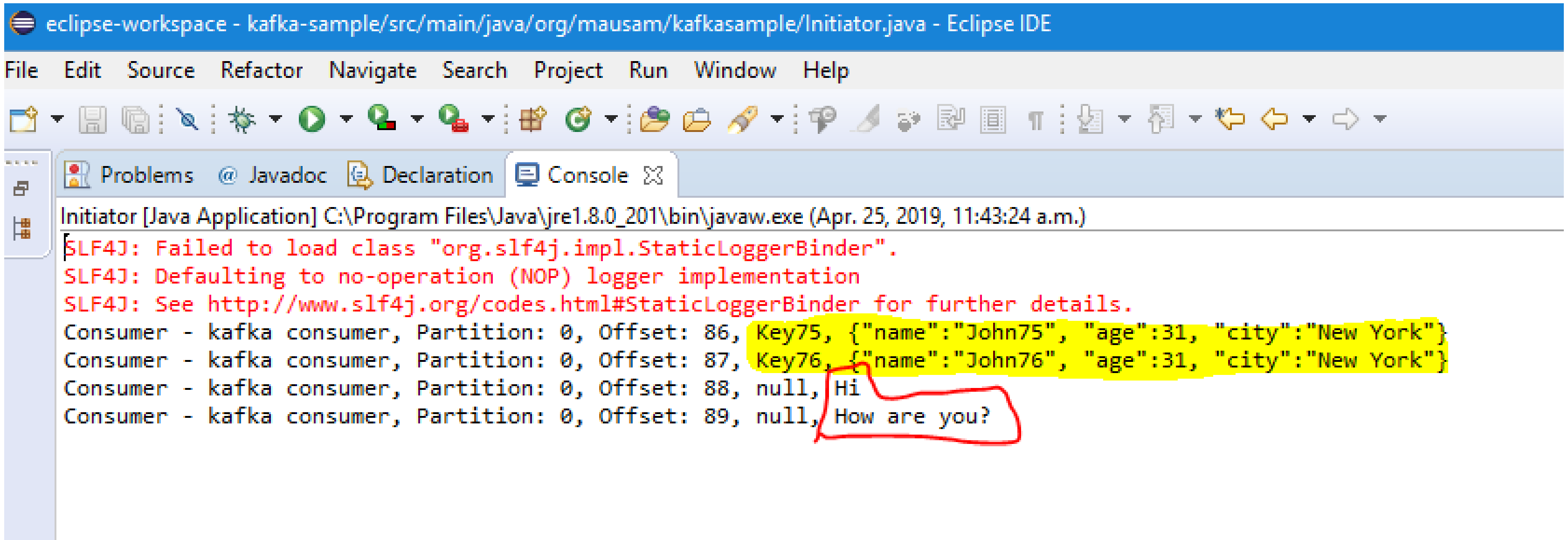
```
C:\Users\pedro>kafka-console-producer.bat --broker-list localhost:9092 --topic spark_topic
>Hi
>How are you?
>
```



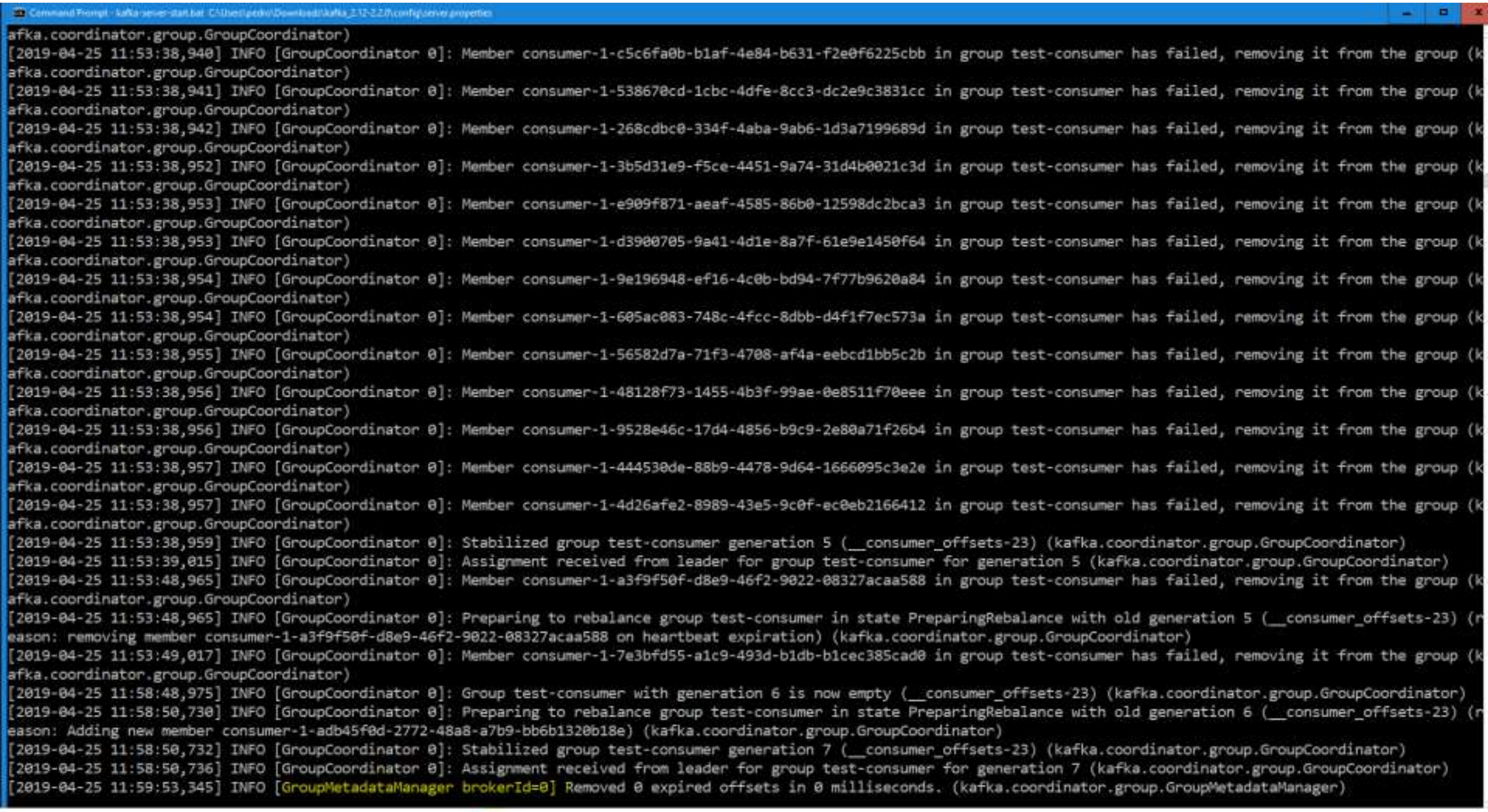
```java
package org.mausam.kafkasample;

import java.util.Arrays;

public class KafkaSampleConsumer implements Runnable {

    private static final String TOPIC_NAME = "spark_topic";
    private String name;

    public KafkaSampleConsumer(final String name) { //I create a consumer and give it a name .
        this.name = name;
    }

    public void run() {
        KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(getConfig());//create kafka consumer object
        consumer.subscribe(Arrays.asList(TOPIC_NAME)); //I subscribe my consumer to the topic.publisher does not

        try {
            while (true) {
                ConsumerRecords<String, String> records = consumer.poll(50);//.poll takes the number of ms to wait for polling .
                for (ConsumerRecord<String, String> record : records) { //it fetches all the new records
                    System.out.printf("Consumer - %s, Partition: %d, Offset: %d, %s, %s %n",
                            this.name, record.partition(), record.offset(), record.key(), record.value());
                }

                Thread.sleep(2 * 1000);//sleep for
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            consumer.close();
        }
    }

    private Properties getConfig() {
        Properties config = new Properties();
        config.put("bootstrap.servers", "localhost:9092");
        config.put("group.id", "test-consumer");
        config.put("enable.auto.commit", "true");
        config.put("auto.commit.interval.ms", "1000");
        config.put("connections.max.idle.ms", "1000");
        config.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
        config.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

        return config;
    }

}
```



```java
package org.mausam.kafkasample;

public class Initiator {

    public static void main(String[] args) throws Exception {
        //Thread producer = new Thread(new KafkaSampleProducer()); //So I create a producer threat
        Thread consumer = new Thread(new KafkaSampleConsumer("kafka consumer"));

        //producer.start();
        consumer.start();

        //producer.join();
        consumer.join();
    }

}
```


eclipse-workspace - kafka-sample/src/main/java/org/mausam/kafkasample/Initiator.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems
Javadoc
Declaration
Console

```
Initiator [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Apr. 25, 2019, 11:43:24 a.m.)
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Consumer - kafka consumer, Partition: 0, Offset: 86, Key75, {"name":"John75", "age":31, "city":"New York"}
Consumer - kafka consumer, Partition: 0, Offset: 87, Key76, {"name":"John76", "age":31, "city":"New York"}
Consumer - kafka consumer, Partition: 0, Offset: 88, null, Hi
Consumer - kafka consumer, Partition: 0, Offset: 89, null, How are you?
```


Command Prompt - kafka-server-start.bat C:\Users\pedro\Downloads\kafka_2.12-2.2.0\config\server.properties

```
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,940] INFO [GroupCoordinator 0]: Member consumer-1-c5c6fa0b-b1af-4e84-b631-f2e0f6225cbb in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,941] INFO [GroupCoordinator 0]: Member consumer-1-538670cd-1cbc-4dfe-8cc3-dc2e9c3831cc in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,942] INFO [GroupCoordinator 0]: Member consumer-1-268cdbc0-334f-4aba-9ab6-1d3a7199689d in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,952] INFO [GroupCoordinator 0]: Member consumer-1-3b5d31e9-f5ce-4451-9a74-31d4b0021c3d in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,953] INFO [GroupCoordinator 0]: Member consumer-1-e909f871-aeaf-4585-86b0-12598dc2bca3 in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,953] INFO [GroupCoordinator 0]: Member consumer-1-d3900705-9a41-4d1e-8a7f-61e9e1450f64 in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,954] INFO [GroupCoordinator 0]: Member consumer-1-9e196948-ef16-4c0b-bd94-7f77b9620a84 in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,954] INFO [GroupCoordinator 0]: Member consumer-1-605ac083-748c-4fcc-8dbb-d4f1f7ec573a in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,955] INFO [GroupCoordinator 0]: Member consumer-1-56582d7a-71f3-4708-af4a-eebcd1bb5c2b in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,956] INFO [GroupCoordinator 0]: Member consumer-1-48128f73-1455-4b3f-99ae-0e8511f70eee in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,956] INFO [GroupCoordinator 0]: Member consumer-1-9528e46c-17d4-4856-b9c9-2e80a71f26b4 in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,957] INFO [GroupCoordinator 0]: Member consumer-1-444530de-88b9-4478-9d64-1666095c3e2e in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,957] INFO [GroupCoordinator 0]: Member consumer-1-4d26afe2-8989-43e5-9c0f-ec0eb2166412 in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:38,959] INFO [GroupCoordinator 0]: Stabilized group test-consumer generation 5 (__consumer_offsets-23) (kafka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:39,015] INFO [GroupCoordinator 0]: Assignment received from leader for group test-consumer for generation 5 (kafka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:48,965] INFO [GroupCoordinator 0]: Member consumer-1-a3f9f50f-d8e9-46f2-9022-08327acaa588 in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:48,965] INFO [GroupCoordinator 0]: Preparing to rebalance group test-consumer in state PreparingRebalance with old generation 5 (__consumer_offsets-23) (r
eason: removing member consumer-1-a3f9f50f-d8e9-46f2-9022-08327acaa588 on heartbeat expiration) (kafka.coordinator.group.GroupCoordinator)
[2019-04-25 11:53:49,017] INFO [GroupCoordinator 0]: Member consumer-1-7e3bfd55-a1c9-493d-b1db-b1cec385cad0 in group test-consumer has failed, removing it from the group (k
afka.coordinator.group.GroupCoordinator)
[2019-04-25 11:58:48,975] INFO [GroupCoordinator 0]: Group test-consumer with generation 6 is now empty (__consumer_offsets-23) (kafka.coordinator.group.GroupCoordinator)
[2019-04-25 11:58:50,730] INFO [GroupCoordinator 0]: Preparing to rebalance group test-consumer in state PreparingRebalance with old generation 6 (__consumer_offsets-23) (r
eason: Adding new member consumer-1-adb45f0d-2772-48a8-a7b9-bb6b1320b18e) (kafka.coordinator.group.GroupCoordinator)
[2019-04-25 11:58:50,732] INFO [GroupCoordinator 0]: Stabilized group test-consumer generation 7 (__consumer_offsets-23) (kafka.coordinator.group.GroupCoordinator)
[2019-04-25 11:58:50,736] INFO [GroupCoordinator 0]: Assignment received from leader for group test-consumer for generation 7 (kafka.coordinator.group.GroupCoordinator)
[2019-04-25 11:59:53,345] INFO [GroupMetadataManager brokerId=0] Removed 0 expired offsets in 0 milliseconds. (kafka.coordinator.group.GroupMetadataManager)
```

Make sure broker is running.