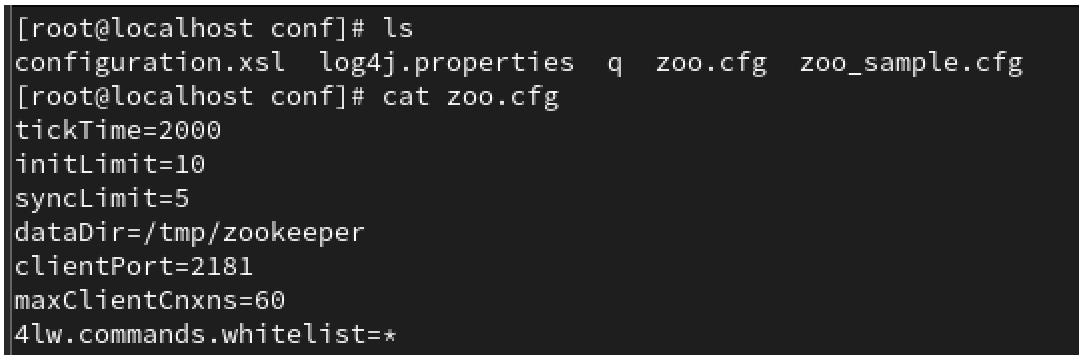
Kafka & Zookeeper Installation and other doc

Installation of Zookeeper

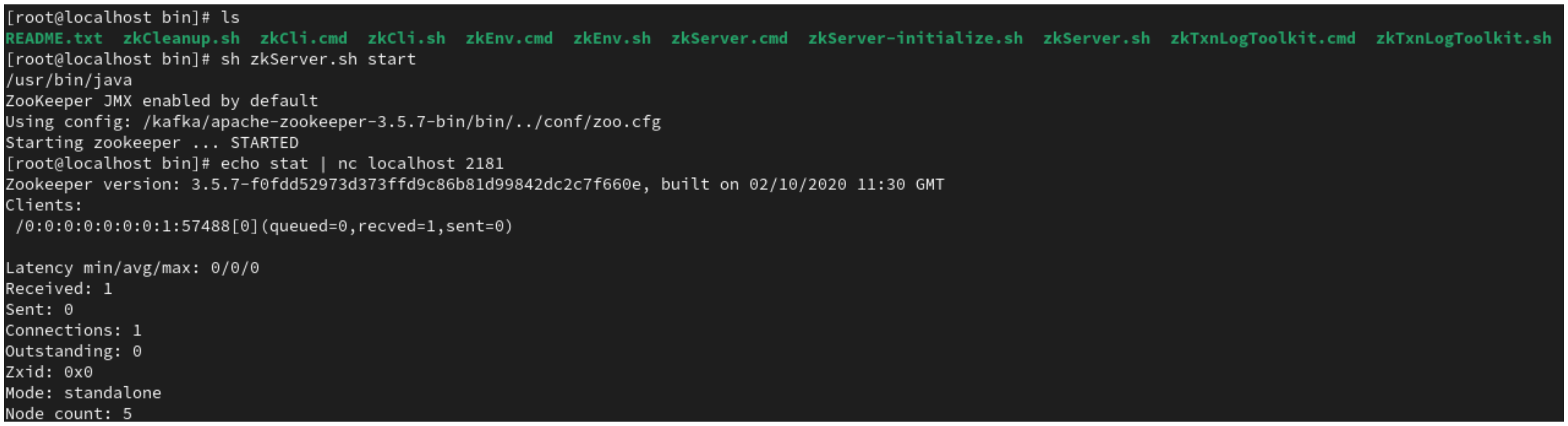
- Install java in the system
  - # yum install java -y
- Download the ZooKeeper from the link: https://archive.apache.org/dist/zookeeper/zookeeper-3.5.7/apache-zookeeper-3.5.7-bin.tar.gz using wget command
  - #wget https://archive.apache.org/dist/zookeeper/zookeeper-3.5.7/apache-zookeeper-3.5.7-bin.tar.gz
  - # tar -xf apache-zookeeper-3.5.7-bin.tar.gz
- We have config the configuration file for zookeeper, to do so go to the /kafka/apache-zookeeper-3.5.7-bin/conf/
  - # cp zoo_sample.cfg zoo.cfg
  - Edit the zoo.cfg file with below code

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/tmp/zookeeper
clientPort=2181
maxClientCnxns=60
4lw.commands.whitelist=*
```



```
[root@localhost conf]# ls
configuration.xsl  log4j.properties  q  zoo.cfg  zoo_sample.cfg
[root@localhost conf]# cat zoo.cfg
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/tmp/zookeeper
clientPort=2181
maxClientCnxns=60
4lw.commands.whitelist=*
```

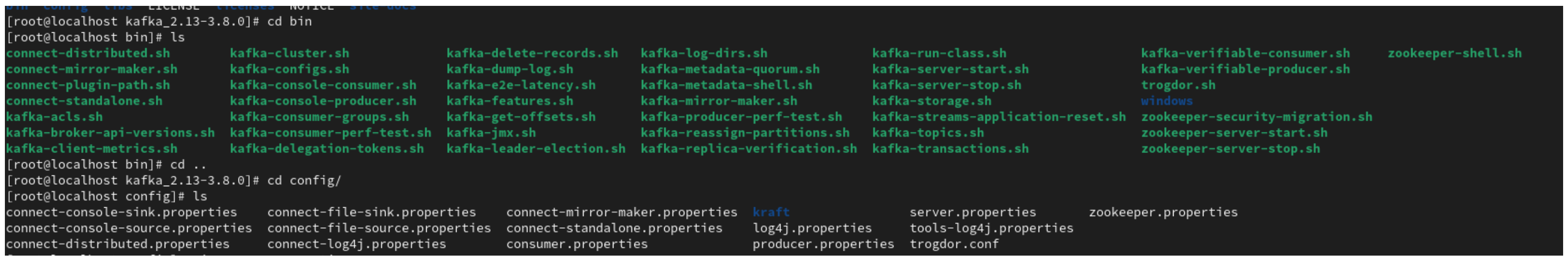
- Install the nc and net-tools on using
  - # yum install nc -y
  - # yum install net-tools -y
- To start the zookeeper run below command and make sure you are in bin dir
  - # sh zkServer.sh start
- To check the status of zookeeper
  - # echo stat | nc localhost 2181



```
[root@localhost bin]# ls
README.txt  zkCleanup.sh  zkCli.cmd  zkCli.sh  zkEnv.cmd  zkEnv.sh  zkServer.cmd  zkServer-initialize.sh  zkServer.sh  zkTxnLogToolkit.cmd  zkTxnLogToolkit.sh
[root@localhost bin]# sh zkServer.sh start
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /kafka/apache-zookeeper-3.5.7-bin/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@localhost bin]# echo stat | nc localhost 2181
Zookeeper version: 3.5.7-f0fdd52973d373ffd9c86b81d99842dc2c7f660e, built on 02/10/2020 11:30 GMT
Clients:
 /0:0:0:0:0:0:0:1:57488[0](queued=0,recved=1,sent=0)

Latency min/avg/max: 0/0/0
Received: 1
Sent: 0
Connections: 1
Outstanding: 0
Zxid: 0x0
Mode: standalone
Node count: 5
```

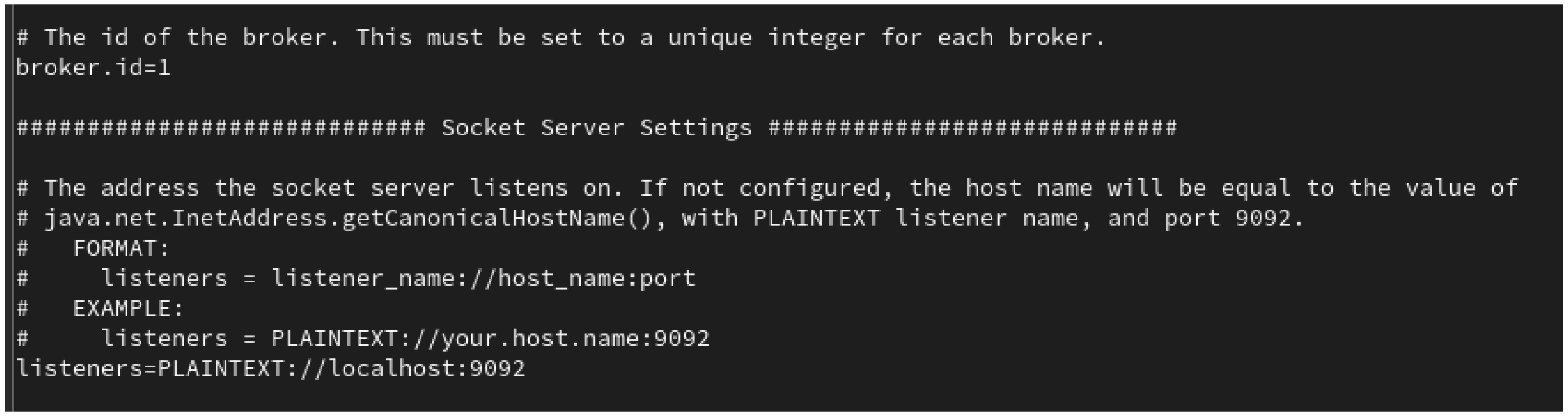
Installation of Kafka

- Download the kafka from the link: https://downloads.apache.org/kafka/3.8.0/kafka_2.13-3.8.0.tgz
  - #wget https://downloads.apache.org/kafka/3.8.0/kafka_2.13-3.8.0.tgz
  - #tar -xf kafka_2.13-3.8.0.tgz



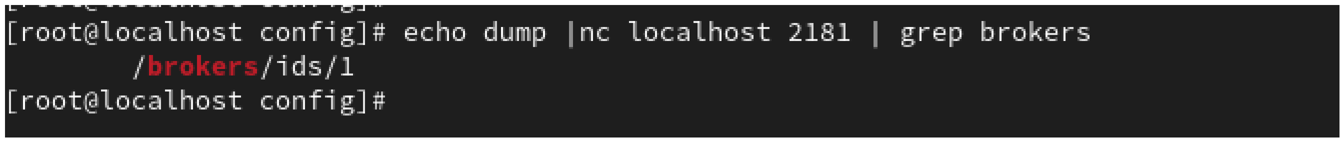
```
[root@localhost kafka_2.13-3.8.0]# cd bin
[root@localhost bin]# ls
connect-distributed.sh        kafka-cluster.sh              kafka-delete-records.sh   kafka-log-dirs.sh             kafka-run-class.sh                    kafka-verifiable-consumer.sh    zookeeper-shell.sh
connect-mirror-maker.sh       kafka-configs.sh              kafka-dump-log.sh         kafka-metadata-quorum.sh      kafka-server-start.sh                 kafka-verifiable-producer.sh
connect-plugin-path.sh        kafka-console-consumer.sh     kafka-e2e-latency.sh      kafka-metadata-shell.sh       kafka-server-stop.sh                  trogdor.sh
connect-standalone.sh         kafka-console-producer.sh     kafka-features.sh         kafka-mirror-maker.sh         kafka-storage.sh                      windows
kafka-acls.sh                 kafka-consumer-groups.sh      kafka-get-offsets.sh      kafka-producer-perf-test.sh   kafka-streams-application-reset.sh    zookeeper-security-migration.sh
kafka-broker-api-versions.sh  kafka-consumer-perf-test.sh   kafka-jmx.sh              kafka-reassign-partitions.sh  kafka-topics.sh                       zookeeper-server-start.sh
kafka-client-metrics.sh       kafka-delegation-tokens.sh    kafka-leader-election.sh  kafka-replica-verification.sh kafka-transactions.sh                 zookeeper-server-stop.sh
[root@localhost bin]# cd ..
[root@localhost kafka_2.13-3.8.0]# cd config/
[root@localhost config]# ls
connect-console-sink.properties    connect-file-sink.properties    connect-mirror-maker.properties  kraft                server.properties       zookeeper.properties
connect-console-source.properties  connect-file-source.properties  connect-standalone.properties    log4j.properties     tools-log4j.properties
connect-distributed.properties     connect-log4j.properties        consumer.properties              producer.properties  trogdor.conf
```

- Make change in the server.properties file



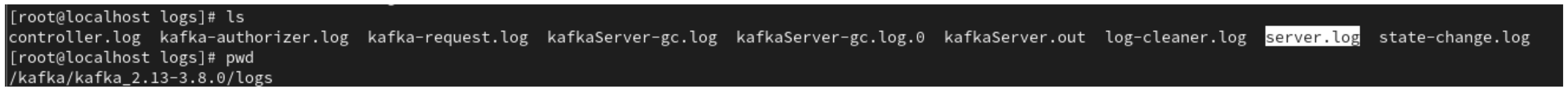
```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1

############################# Socket Server Settings #############################

# The address the socket server listens on. If not configured, the host name will be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://localhost:9092
```

- To start your kafka server in background run below
  - # sh kafka-server-start.sh -daemon /kafka/kafka_2.13-3.8.0/config/server.properties
- To check kafka is up and running with brokers
  - # echo dump |nc localhost 2181 | grep brokers



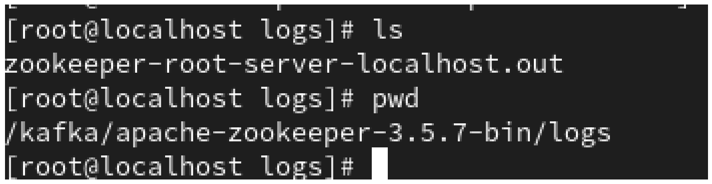
```
[root@localhost config]# echo dump |nc localhost 2181 | grep brokers
        /brokers/ids/1
[root@localhost config]#
```

- Logs of the kafka will be present in logs/server.log



```
[root@localhost logs]# ls
controller.log  kafka-authorizer.log  kafka-request.log  kafkaServer-gc.log  kafkaServer-gc.log.0  kafkaServer.out  log-cleaner.log  server.log  state-change.log
[root@localhost logs]# pwd
/kafka/kafka_2.13-3.8.0/logs
```

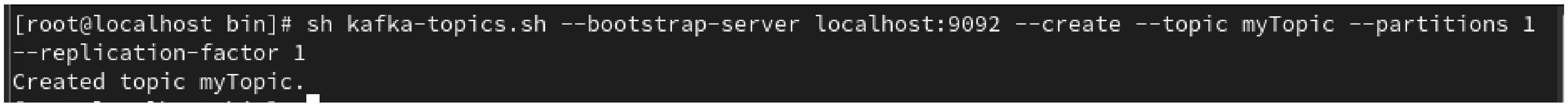
- Zookeeper logs will be available under /apache-zookeeper-3.5.7-bin/logs



```
[root@localhost logs]# ls
zookeeper-root-server-localhost.out
[root@localhost logs]# pwd
/kafka/apache-zookeeper-3.5.7-bin/logs
[root@localhost logs]#
```

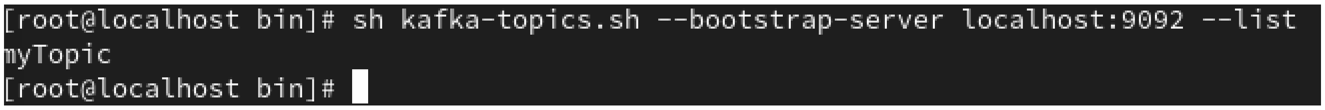
Creation of Topics in Kafka

- To create topic run below command where we create the topic with name “myTopic” with partitions as 1 and replication factor as 1
  - #sh kafka-topics.sh --bootstrap-server localhost:9092 --create --topic myTopic --partitions 1 --replication-factor 1



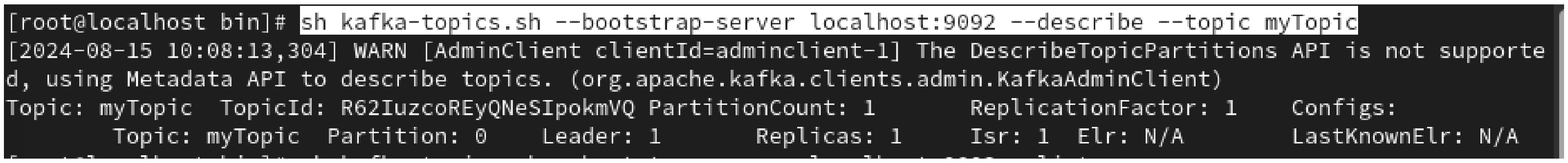
```
[root@localhost bin]# sh kafka-topics.sh --bootstrap-server localhost:9092 --create --topic myTopic --partitions 1
--replication-factor 1
Created topic myTopic.
```

- To check the list of topic we have
  - # sh kafka-topics.sh --bootstrap-server localhost:9092 –list



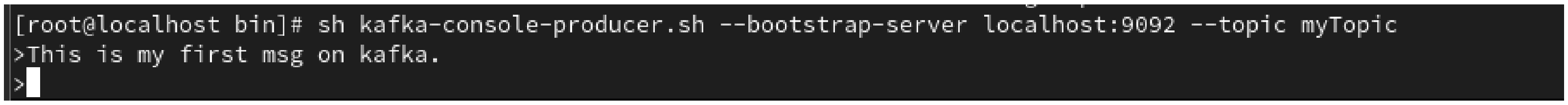
```
[root@localhost bin]# sh kafka-topics.sh --bootstrap-server localhost:9092 --list
myTopic
[root@localhost bin]#
```

- To describe the topic and check its setting
  - # sh kafka-topics.sh --bootstrap-server localhost:9092 --describe --topic myTopic



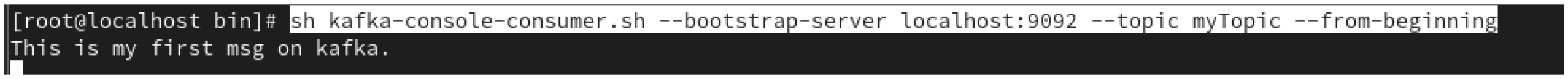
```
[root@localhost bin]# sh kafka-topics.sh --bootstrap-server localhost:9092 --describe --topic myTopic
[2024-08-15 10:08:13,304] WARN [AdminClient clientId=adminclient-1] The DescribeTopicPartitions API is not supporte
d, using Metadata API to describe topics. (org.apache.kafka.clients.admin.KafkaAdminClient)
Topic: myTopic  TopicId: R62IuzcoREyQNeSIpokmVQ PartitionCount: 1       ReplicationFactor: 1    Configs:
        Topic: myTopic  Partition: 0    Leader: 1       Replicas: 1     Isr: 1  Elr: N/A        LastKnownElr: N/A
```

- To publish/produce a message in the topic
  - #sh kafka-console-producer.sh --bootstrap-server localhost:9092 --topic myTopic



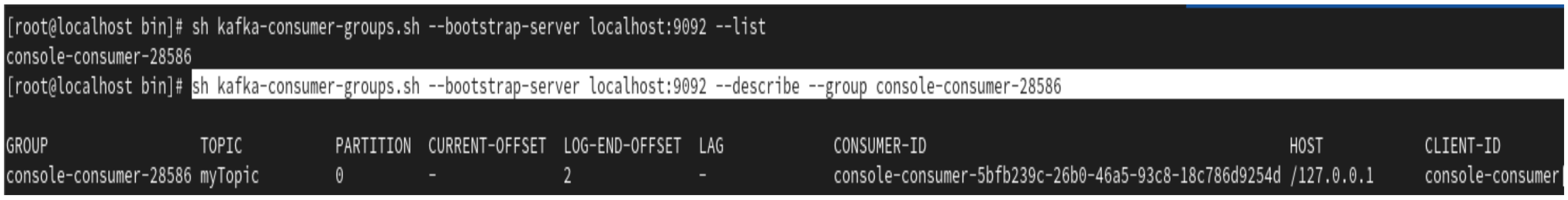
```
[root@localhost bin]# sh kafka-console-producer.sh --bootstrap-server localhost:9092 --topic myTopic
>This is my first msg on kafka.
>
```

- To create the consumer to consume messages from the topic “myTopic”
  - #sh kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic myTopic --from-beginning



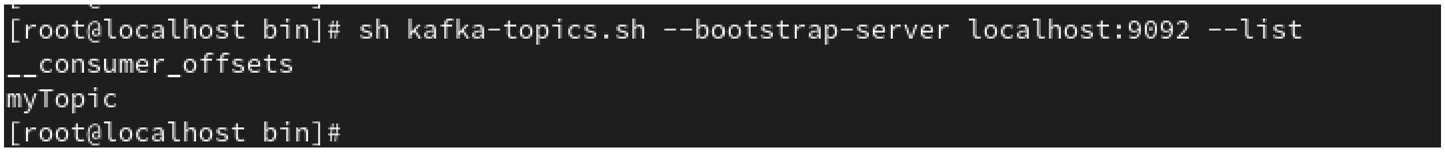
```
[root@localhost bin]# sh kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic myTopic --from-beginning
This is my first msg on kafka.
```

- Whenever we don’t specify the group for consumer, kafka will create group and tag consumer to it. To list down all the consumer groups
  - # sh kafka-consumer-groups.sh --bootstrap-server localhost:9092 –list
  - #sh kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group console-consumer-28586



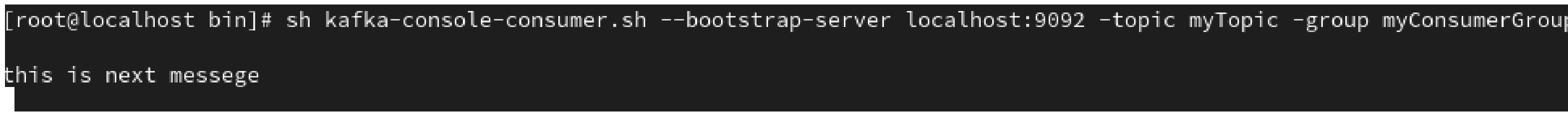
```
[root@localhost bin]# sh kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
console-consumer-28586
[root@localhost bin]# sh kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group console-consumer-28586

GROUP                  TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID                                              HOST            CLIENT-ID
console-consumer-28586 myTopic         0          -               2               -               console-consumer-5bfb239c-26b0-46a5-93c8-18c786d9254d /127.0.0.1      console-consumer
```

- Kakfa will manage all the details in __consumer_offsets where it will keep records how many messages are consumed by the consumer.
  - # sh kafka-topics.sh --bootstrap-server localhost:9092 –list

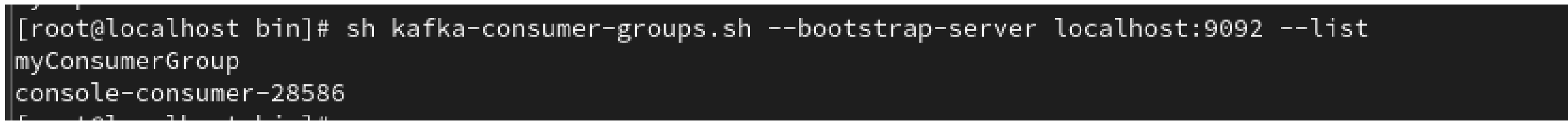


```
[root@localhost bin]# sh kafka-topics.sh --bootstrap-server localhost:9092 --list
__consumer_offsets
myTopic
[root@localhost bin]#
```

- Multiple producers can produce messages on the topics and on the same way multiple consumers can consume messages from multiple topics



```
[root@localhost bin]# sh kafka-console-consumer.sh --bootstrap-server localhost:9092 -topic myTopic -group myConsumerGrou

this is next messege
```



```
[root@localhost bin]# sh kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
myConsumerGroup
console-consumer-28586
```