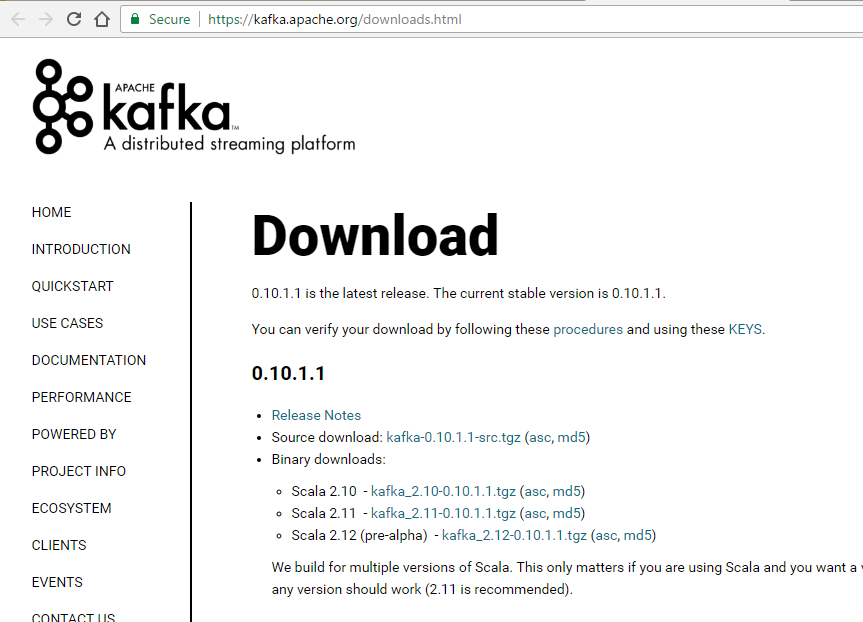
# Download and configure

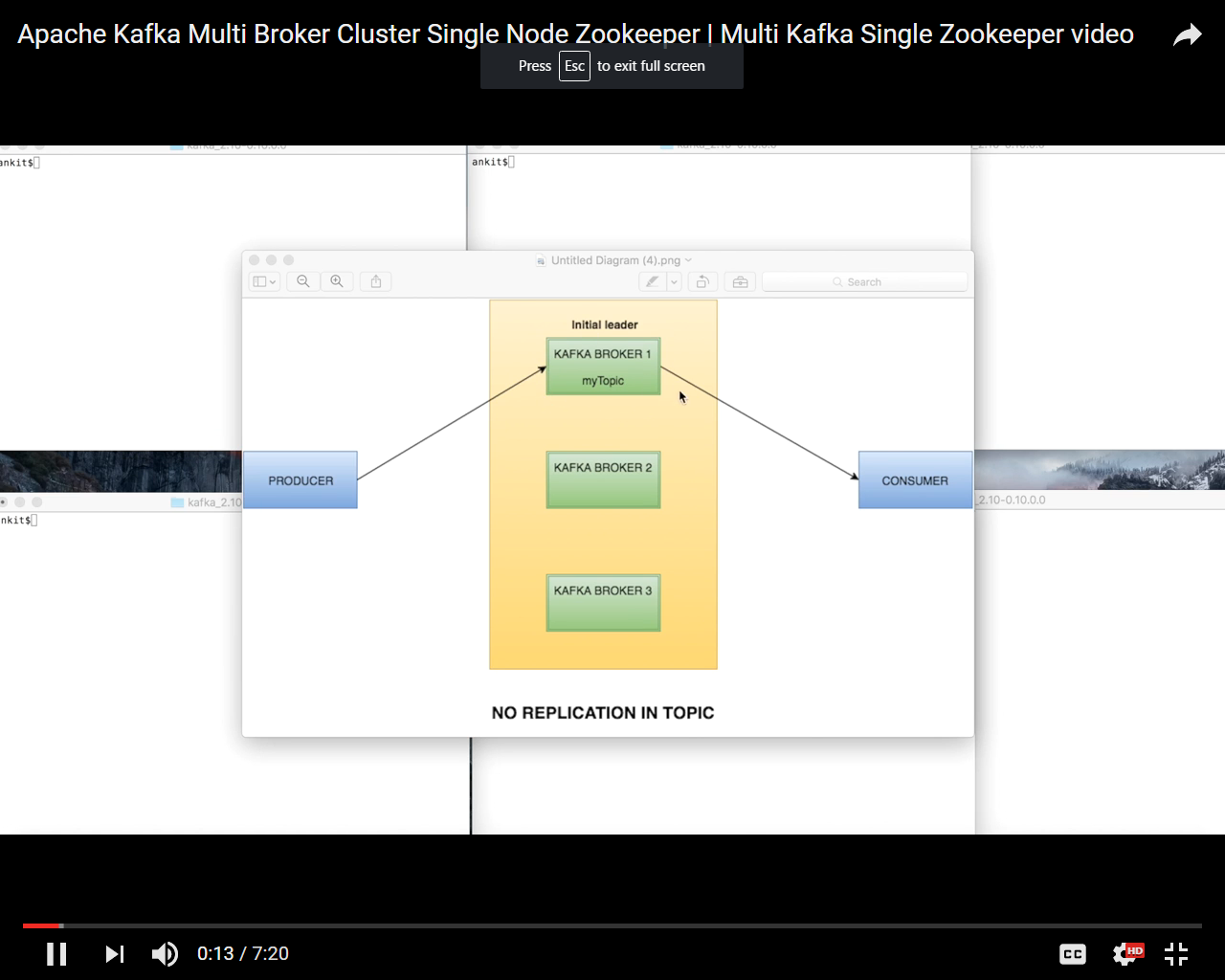
Download Apache kafka from below URL

<https://kafka.apache.org/downloads.html>



Extract the tar file

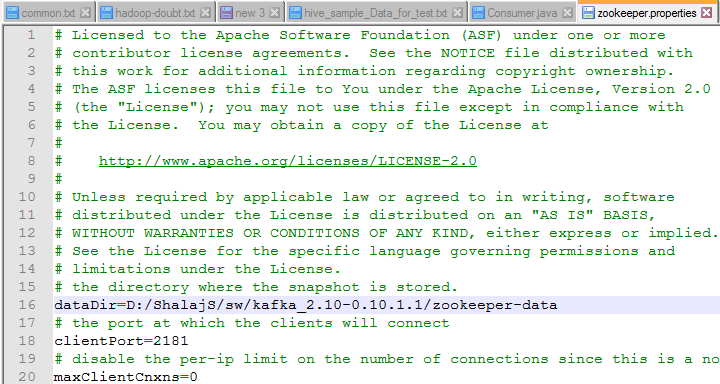
# Apache Kafka Single Broker Cluster Single Node Zookeeper



On single node we can only have 1 replicated topic

## Zookeeper.properties

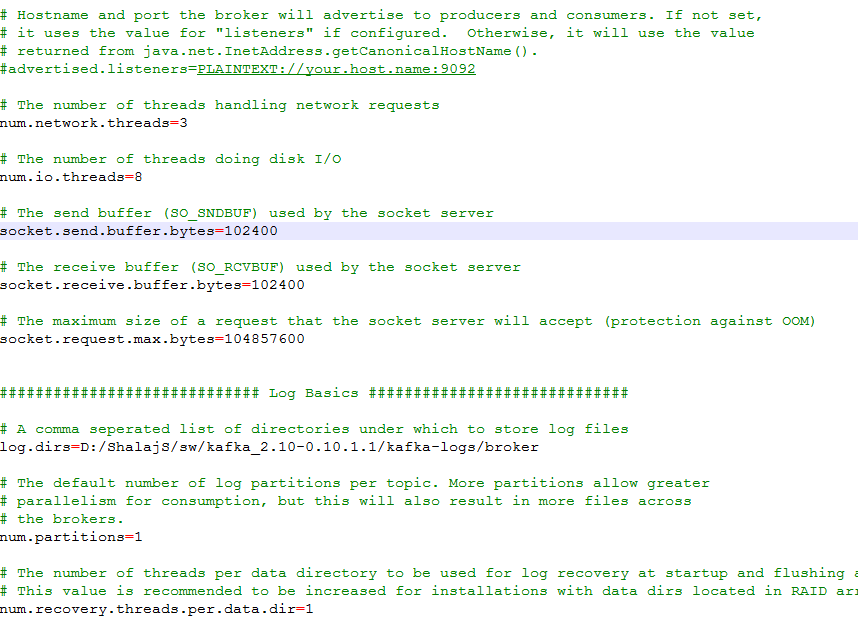
Go to <kafka-home>\config\zookeeper.properties



Changed DataDir path accordingly, zookeeper by default listen on port 2181, you can change the port here if you want to configure some different port

## Server.properties

Go to <kafka-home>\config\server.properties



Now change the log.dirs accordingly in server.properites , by default kafka listen on port 9092

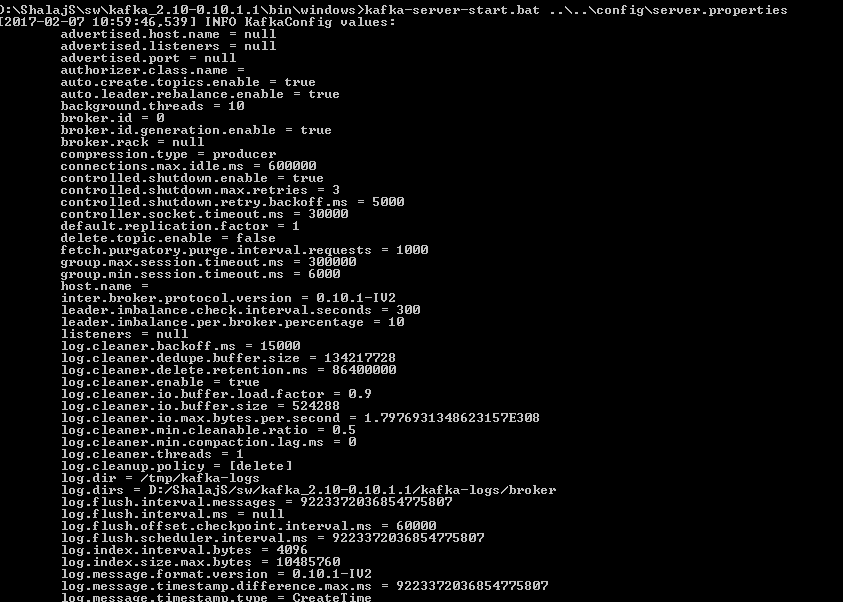
## Start zookeeper

|  |
| --- |
| zookeeper-server-start.bat ..\..\config\zookeeper.properties |



## Start kafka server

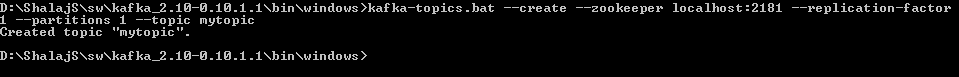
|  |
| --- |
| kafka-server-start.bat ..\..\config\server.properties |



## Create topic

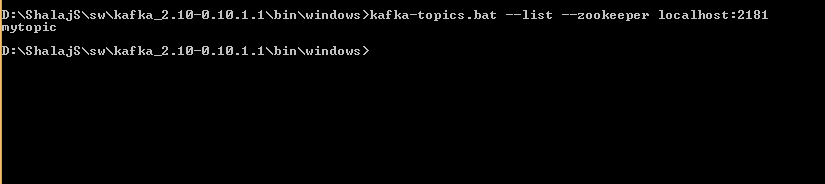
|  |
| --- |
| kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic mytopic |

Here we are can create more than one partition and though we have single kafka broker. Both partition created under same machine



Check list of topics

|  |
| --- |
| kafka-topics.bat --list --zookeeper localhost:2181 |



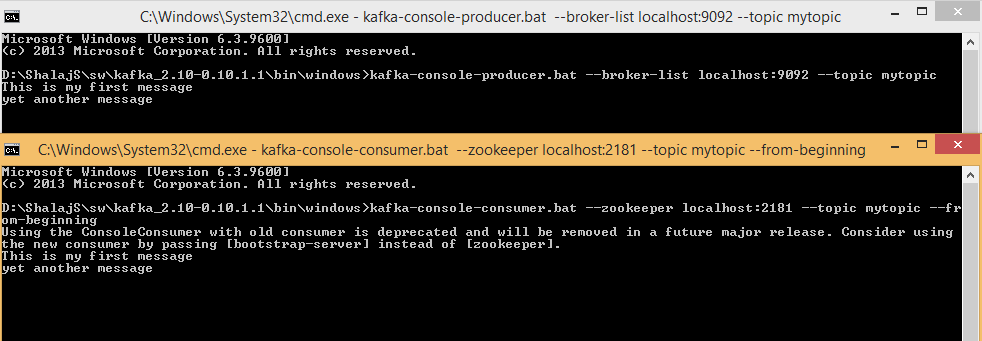
By default kafka create topic automatically if (auto.create.topics.enable=true) some producers publish message on a non-existed topic

In real life projects we need to use producer and consumer API to publish and subscribe messages

## Produce and consume messages

Create producer and consumer in seperate windows

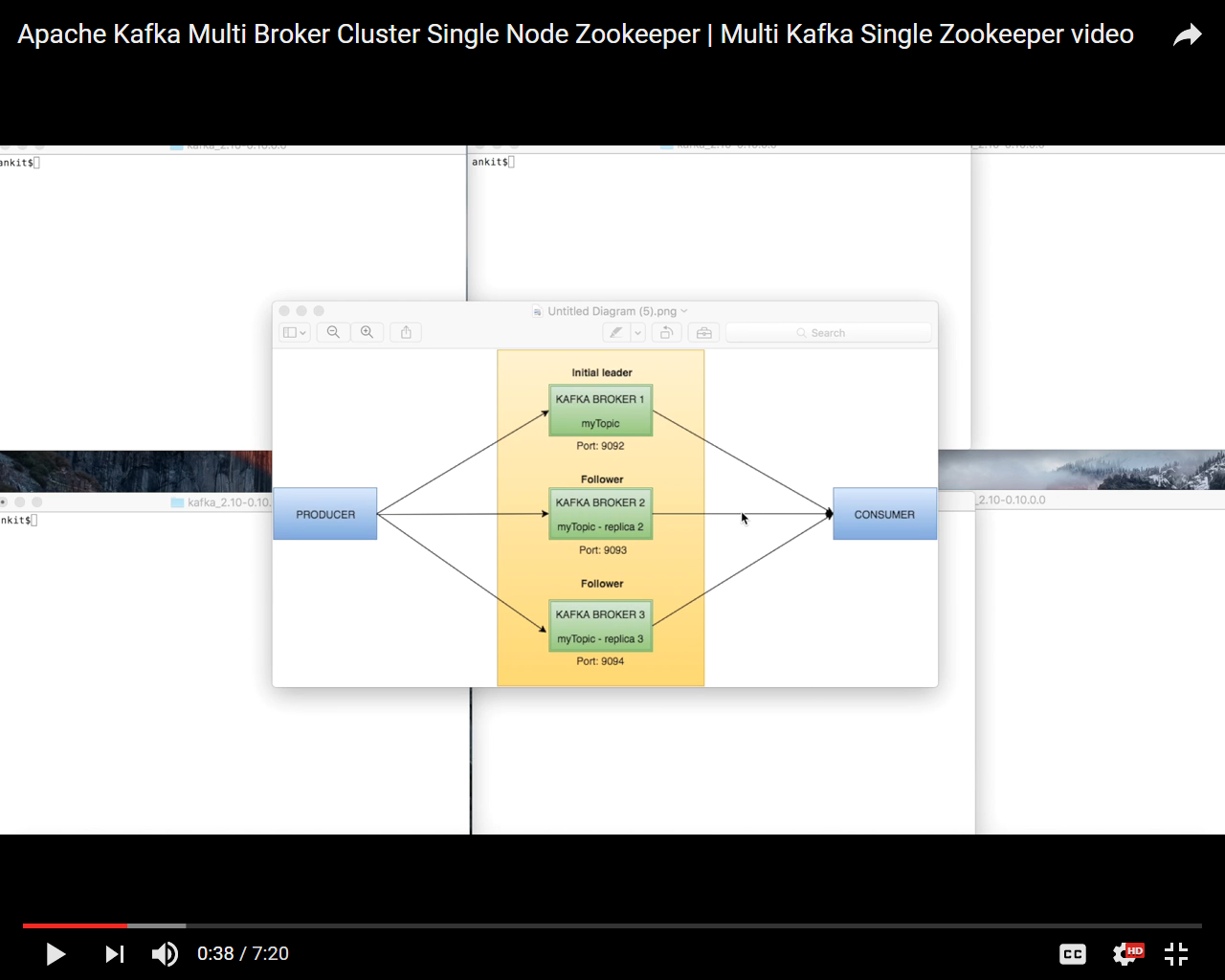
|  |
| --- |
| kafka-console-producer.bat --broker-list localhost:9092 --topic mytopic  kafka-console-consumer.bat --zookeeper localhost:2181 --topic mytopic --from-beginning |



Once you produce some message in producer’s windows you will get the same message on consumer window

# Apache Kafka Multi Broker Cluster Single Node Zookeeper

If we have single kafka broker we will not have any replication of topic and this become single point of failure



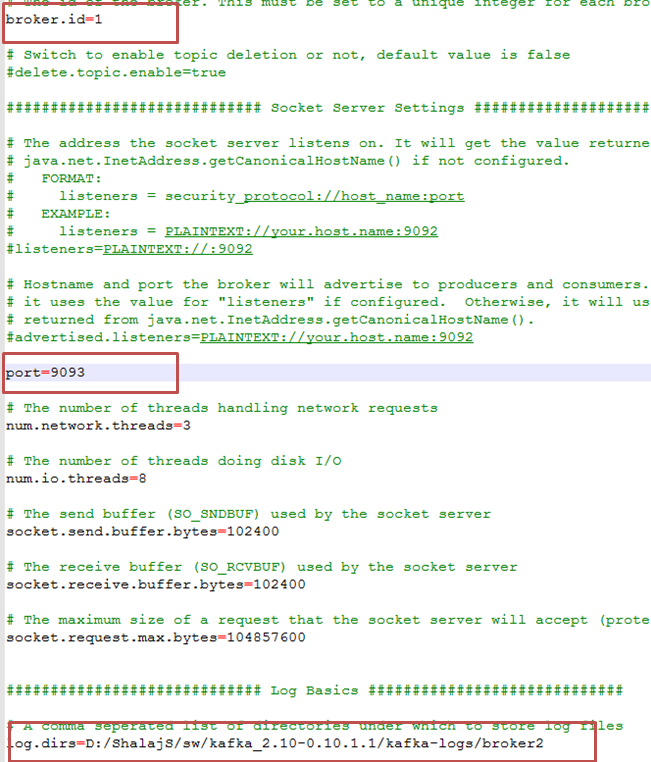
## Modify server.properties

So we need to replicate the same topic in multiple kafka broker so that if one broker goes down we will have the same topic available in other kafka broker

Here we will setup multiple borkers on single machine

Create different server properties for all three brokers

Change broker.id, port, log.dirs



Please find attached server.properties files for all three kafka brokers



*Note: Here we are running all three kafka brokers on same machine but in production this will be on different machine so you don’t need to change port and log.dirs just change broker id*

## Start zookeeper service

|  |
| --- |
| zookeeper-server-start.bat ..\..\config\zookeeper.properties |

## Start Kafka Services

Start all kafka brokers in different windows using below commands

|  |
| --- |
| kafka-server-start.bat ..\..\config\server.properties |
| kafka-server-start.bat ..\..\config\server1.properties |
| kafka-server-start.bat ..\..\config\server2.properties |

## Create Topic

Create topics with replication factor 3 as we have three brokers running, now each of the broker has its own replica of topic

|  |
| --- |
| kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic Replicatedtopic |

## Create producer

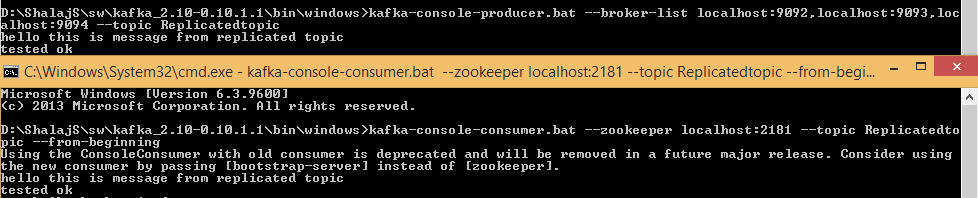
Create producer that produce message for all three brokers

|  |
| --- |
| kafka-console-producer.bat --broker-list localhost:9092,localhost:9093,localhost:9094 --topic Replicatedtopic |

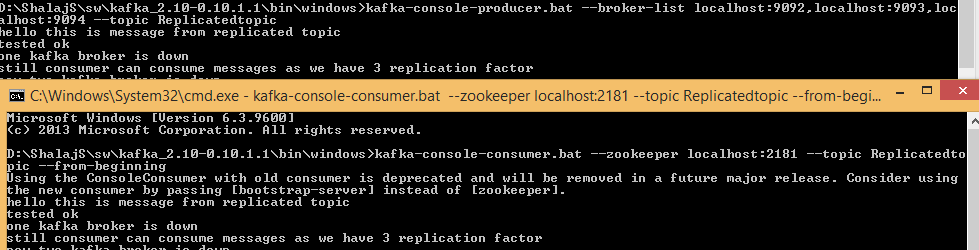
## Create Consumer

|  |
| --- |
| kafka-console-consumer.bat --zookeeper localhost:2181 --topic Replicatedtopic --from-beginning |

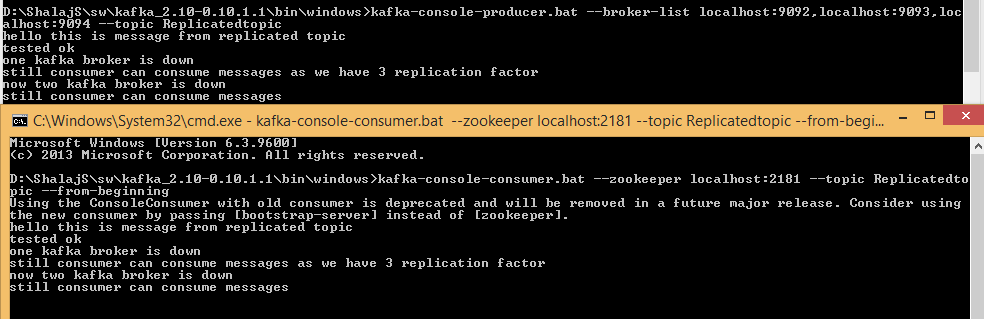
## Produce and Consume messages



Now kill on kafka broker and again check if messages are produced and consumed

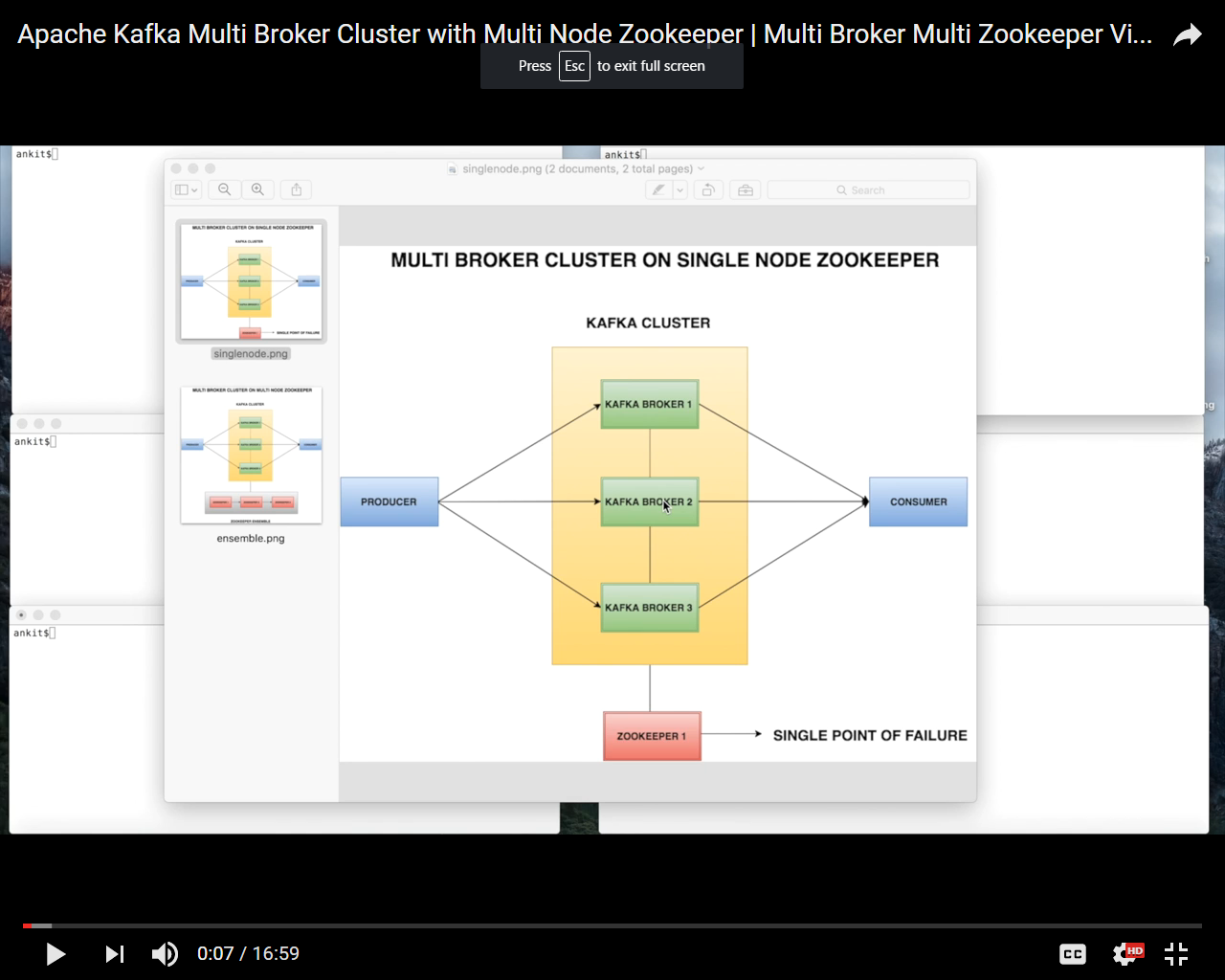


Now kill one more kafka broker



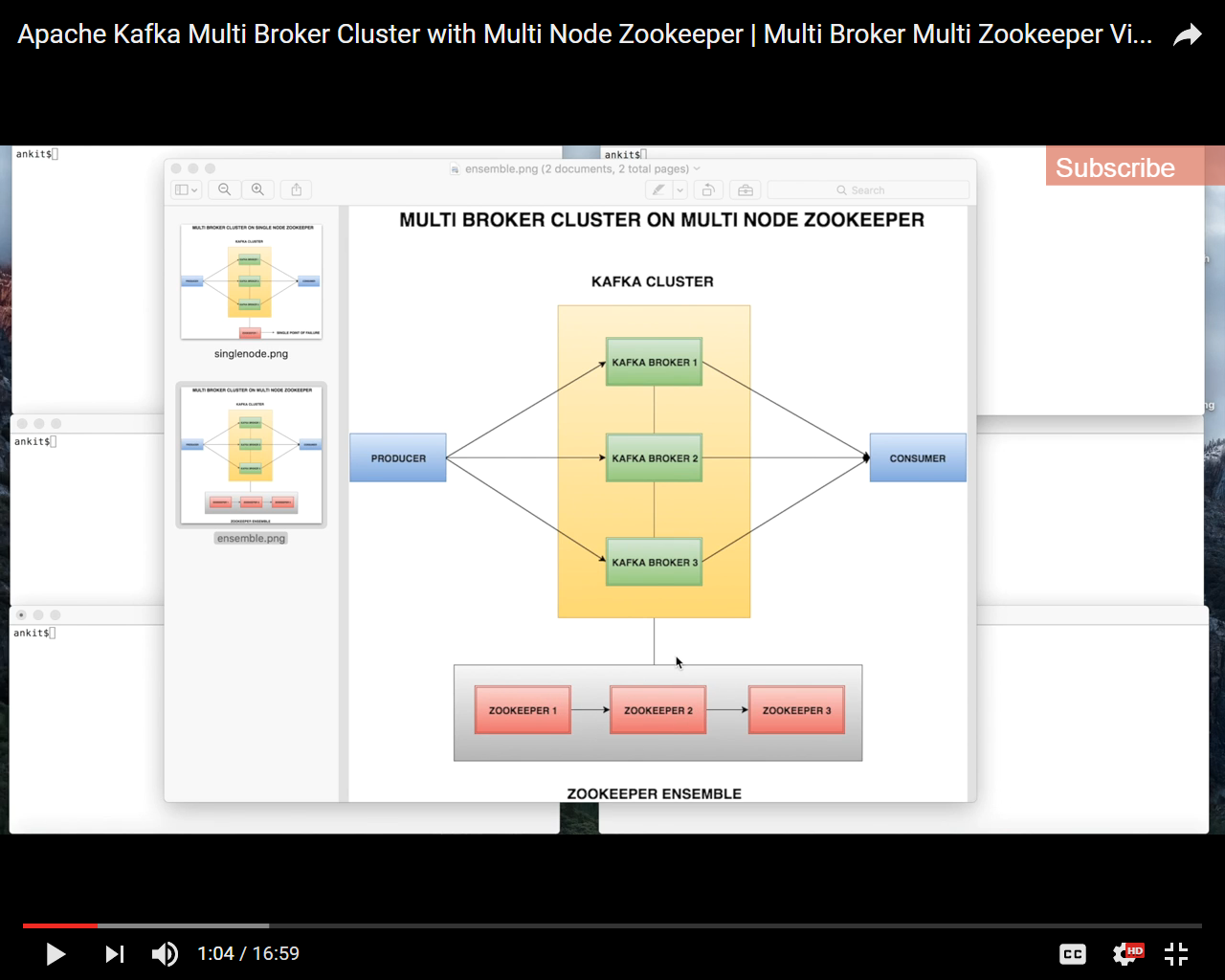
So we can see with using multiple kafka broker we can achieve fault tolerance at kafka broker level

# Apache Kafka Multi Broker Cluster Multi Node Zookeeper



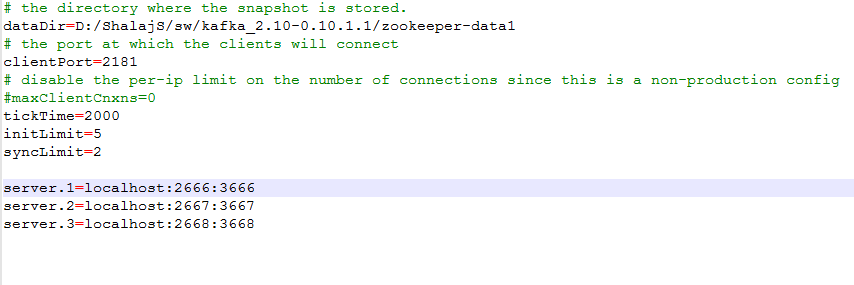
Kafka brokers are heavily dependents on zookeeper for state management and synchronization, so if single zookeeper fails the whole kafka cluster would be in bad stateso now we see how we can set zookeeper cluster

For reliable ZooKeeper service, you should deploy ZooKeeper in a cluster known as an ensemble. As long as majorities of the ensemble are up, the service will be available. Because Zookeeper requires a majority



So now we setup a zookeeper cluster. Zookeeper cluster is called zookeeper Ensemble, it uses quorum algorithm in which we should have at least more than half zookeeper should be up and running. It is best to use an odd number of machines. For example, with four machines ZooKeeper can only handle the failure of a single machine; if two machines fail, the remaining two machines do not constitute a majority. However, with five machines ZooKeeper can handle the failure of two machines.

## Update Zookeeper.properties



* Add tickTime,initLimit and syncLimit also add three servers
* change the dataDir and port for each zookeeper properties file



**#tickTime**

The length of a single tick, which is the basic time unit used by ZooKeeper, as measured in milliseconds. It is used to regulate heartbeats, and timeouts. For example, the minimum session timeout will be two ticks.

**#initLimit**

Amount of time in ticks (here 5\*2000=10000ms = 10 sec) to allow followers to connect and sync to a leader. Increased this value as needed, if the amount of data managed by ZooKeeper is large.

**#syncLimit**

Amount of time in ticks (2\*2=4 sec) to allow followers to sync with ZooKeeper. If followers fall too far behind a leader, they will be dropped.

**#server.x=[hostname]:nnnnn[:nnnnn], etc**

Servers making up the ZooKeeper ensemble. When the server starts up, it determines which server it is by looking for the file myid in the data directory. That file contains the server number, in ASCII, and it should match x in server.x in the left hand side of this setting.

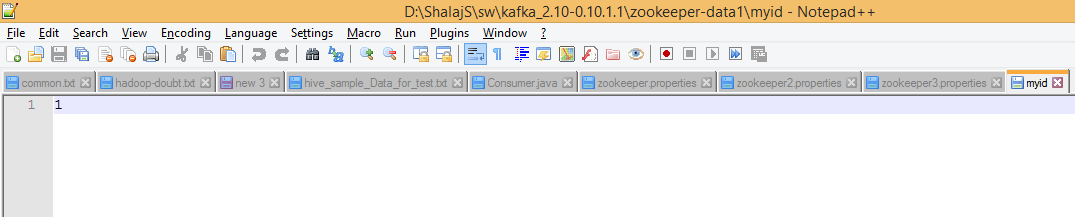
The list of servers that make up ZooKeeper servers that is used by the clients must match the list of ZooKeeper servers that each ZooKeeper server has.

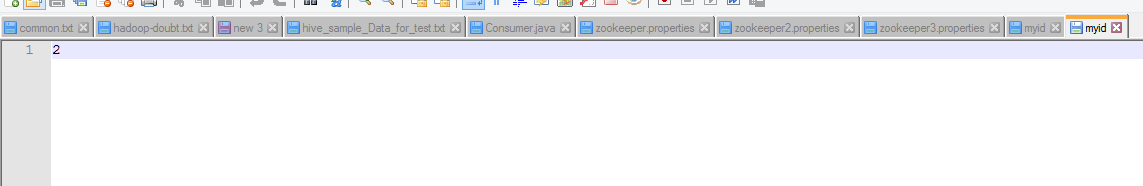
There are two port numbers nnnnn. The first followers use to connect to the leader, and the second is for leader election. The leader election port is only necessary if electionAlg is 1, 2, or 3 (default). If electionAlg is 0, then the second port is not necessary. If you want to test multiple servers on a single machine, then different ports can be used for each server.

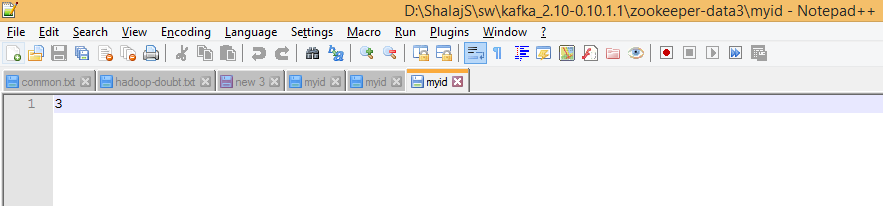
You attribute the server id to each machine by creating a file named myid, one for each server, which resides in that server's data directory, as specified by the configuration file parameter dataDir.

The myid file consists of a single line containing only the text of that machine's id. So myid of server 1 would contain the text "1" and nothing else. The id must be unique within the ensemble and should have a value between 1 and 255.

Now create myid file in all three zookeeper dataDir and just add single value 1, 2 and 3



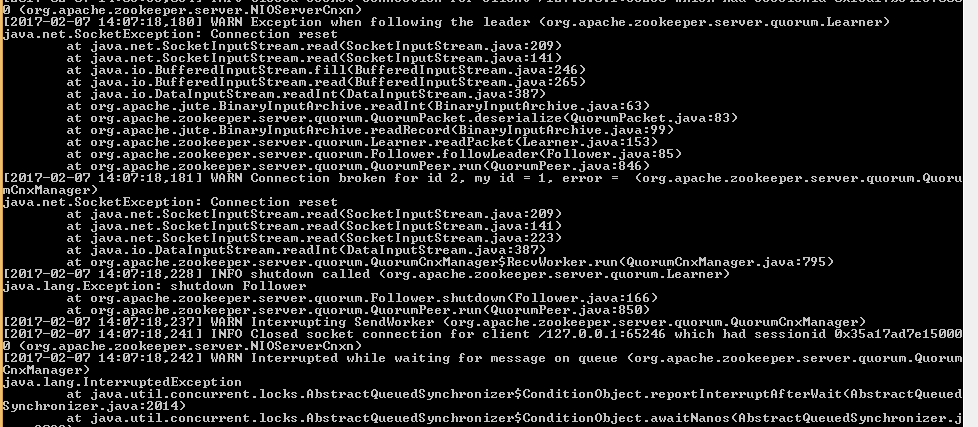




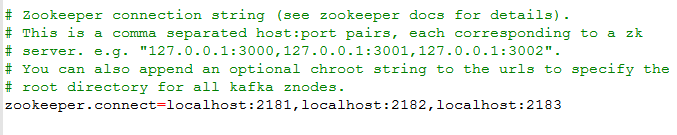
Now run all three instances of zookeeper

|  |
| --- |
| zookeeper-server-start.bat ..\..\config\zookeeper.properties |
| zookeeper-server-start.bat ..\..\config\zookeeper2.properties |
| zookeeper-server-start.bat ..\..\config\zookeeper3.properties |

You will get following warning when you start first zookeeper , just ignore it this is because it is expecting at least two zookeeper up and running, once you started all zookeepers this warning goes away



## Update server.properties





|  |
| --- |
| kafka-server-start.bat ..\..\config\server.properties |
| kafka-server-start.bat ..\..\config\server2.properties |
| kafka-server-start.bat ..\..\config\server3.properties |

## Create Topic

|  |
| --- |
| kafka-topics.bat --create --zookeeper localhost:2181,localhost:2182,localhost:2183 --replication-factor 3 --partitions 1 --topic newTopic |

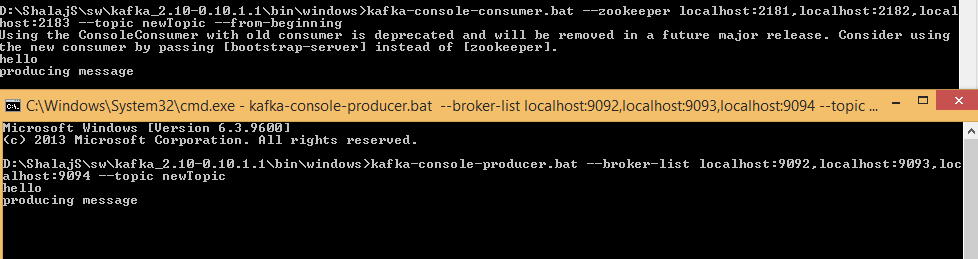
## Create producer

|  |
| --- |
| kafka-console-producer.bat --broker-list localhost:9092,localhost:9093,localhost:9094 --topic newTopic |

## Create Consumer

|  |
| --- |
| kafka-console-consumer.bat --zookeeper localhost:2181,localhost:2182,localhost:2183 --topic newTopic --from-beginning |

Now try to send message from producer and you will see consumer consumes all messages



Just try to terminate one of the zookeeper, still producer consumer work smoothly