KAFKA – QUEUE

Topic

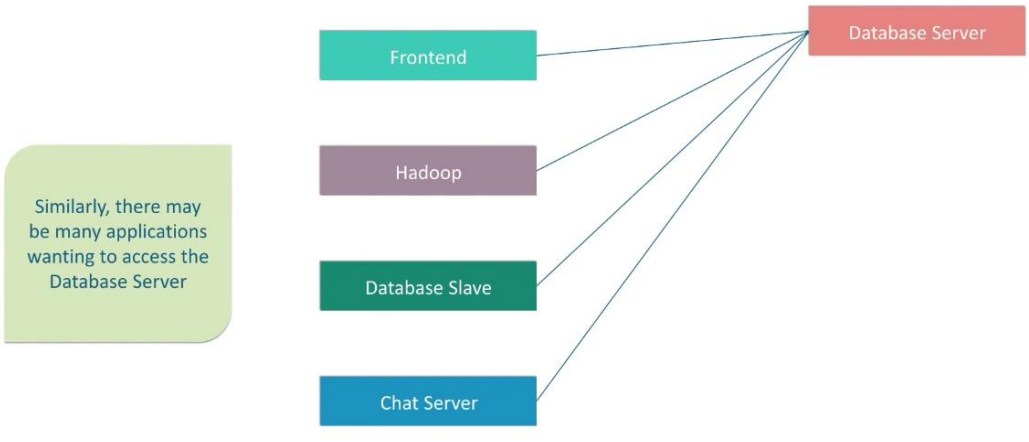
- Need of Messaging System
- What is Kafka?
- Kafka Features
- Kafka Components
- Kafka architecture
- Installing Kafka
- Working with Single Node Single Broker Cluster

Data Pipelines

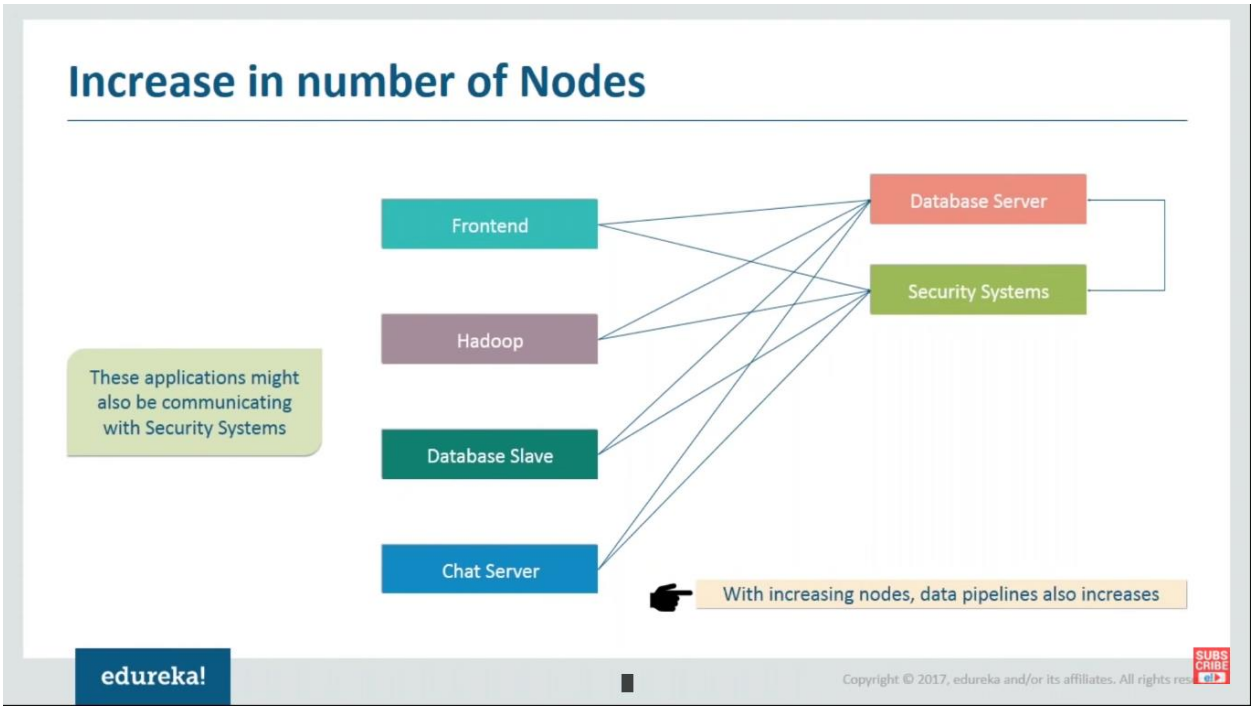
Communication is required between different systems in the real-time scenario, which is done by using data pipelines.

For Example: Chat Server needs to communicate with Database Server for storing messages

Increase in number of Nodes

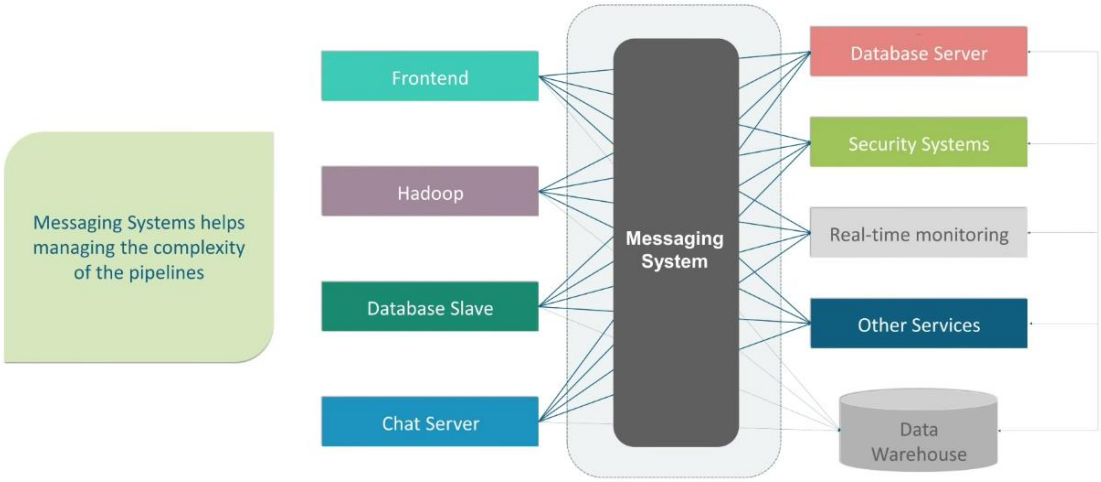


Similarly, there may be many applications wanting to access the Database Server



With time data pipelines also increases and its difficult for a developer to maintain these pipelines , every pipeline have its own configuration . So organization cant bear any loss of transaction data or any transactions .

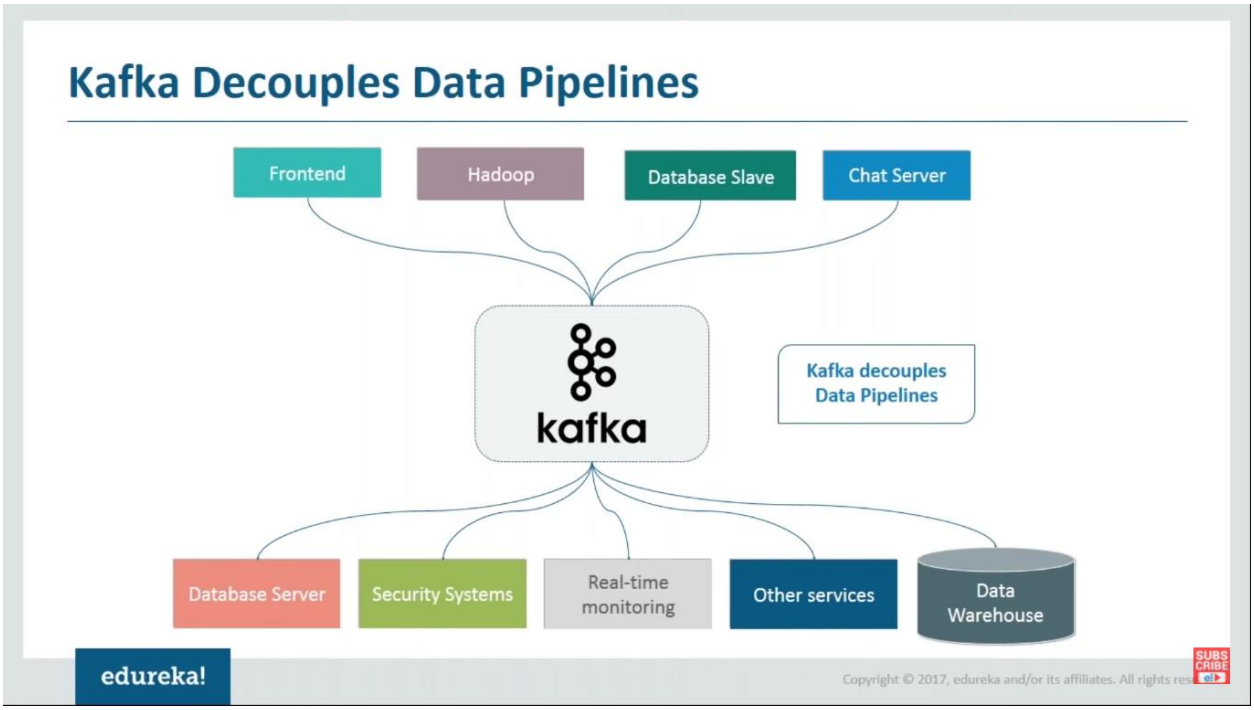
Solution to the Complex Data Pipelines



edureka!

Let's See How Kafka Solves the Problem

edureka!



Applications that sends the data to kafka called producers . And applications those listen the events called consumers.

We can have multiple subscribers for the same set of data .

Ex : Radio Transmission

(Diff people can listen on radio aysnc)

What is Kafka?

- **Apache Kafka** is a distributed *publish-subscribe* messaging system
- It was originally developed at LinkedIn and later on became a part of Apache Project
- Kafka is fast, scalable, durable, fault-tolerant and distributed by design

Apache Kafka
A high-throughput distributed messaging system.

Kafka @LinkedIn

- *1100+ commodity machines*
- *31,000+ topics*
- *350,000+ partitions*

- *675 billion messages/day*
- *150 TB/day in*
- *580 TB/day out*

Peak Load
- *10.5 million messages/sec*
- *18.5 GB/sec Inbound*
- *70.5 GB/sec Outbound*

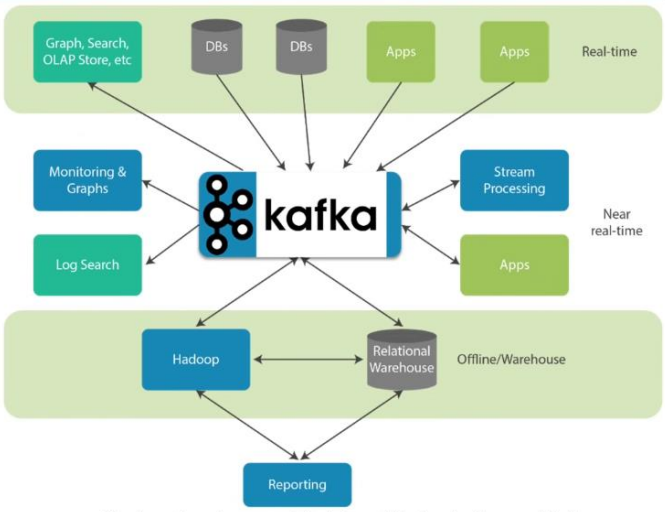


Fig: A modern stream-centric data architecture built around Kafka

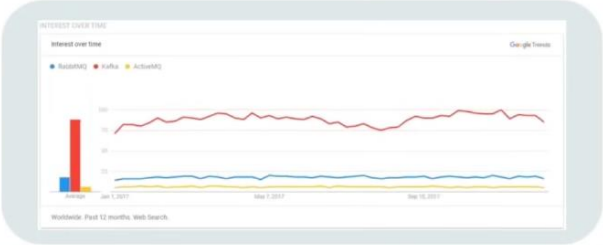
Kafka Growth Exploding

- More than **1/3** of all Fortune **500** companies use **Kafka**.
- These companies includes the top ten travel companies, **7** of top ten banks, **8** of top ten insurance companies, **9** of top ten telecom companies.
- **LinkedIn**, **Microsoft** and **Netflix** process billions of messages a day with Kafka (1,000,000,000,000).
- **Kafka** is used for **real-time streams** of data & used to collect big data for **real time analysis**.

86% of respondents reported that the number of their systems that use Kafka is increasing

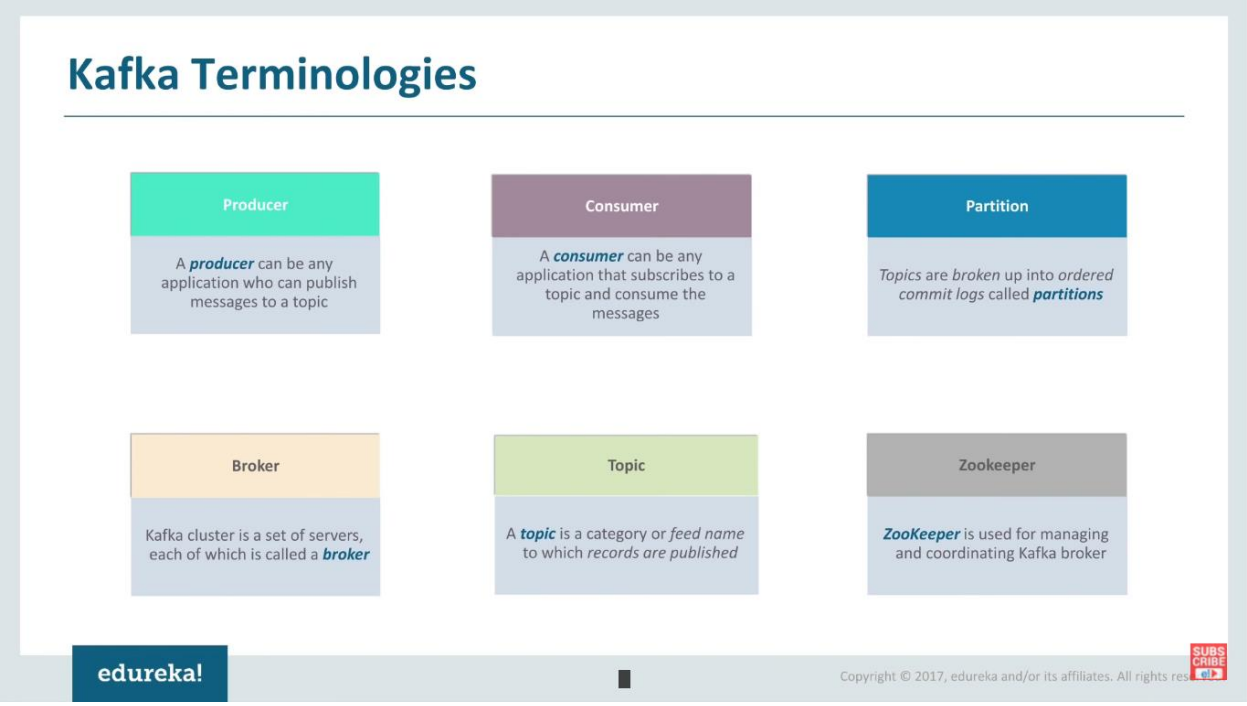
20% reported that the number is "growing a lot!"

52% of organizations have at least 6 systems running Kafka



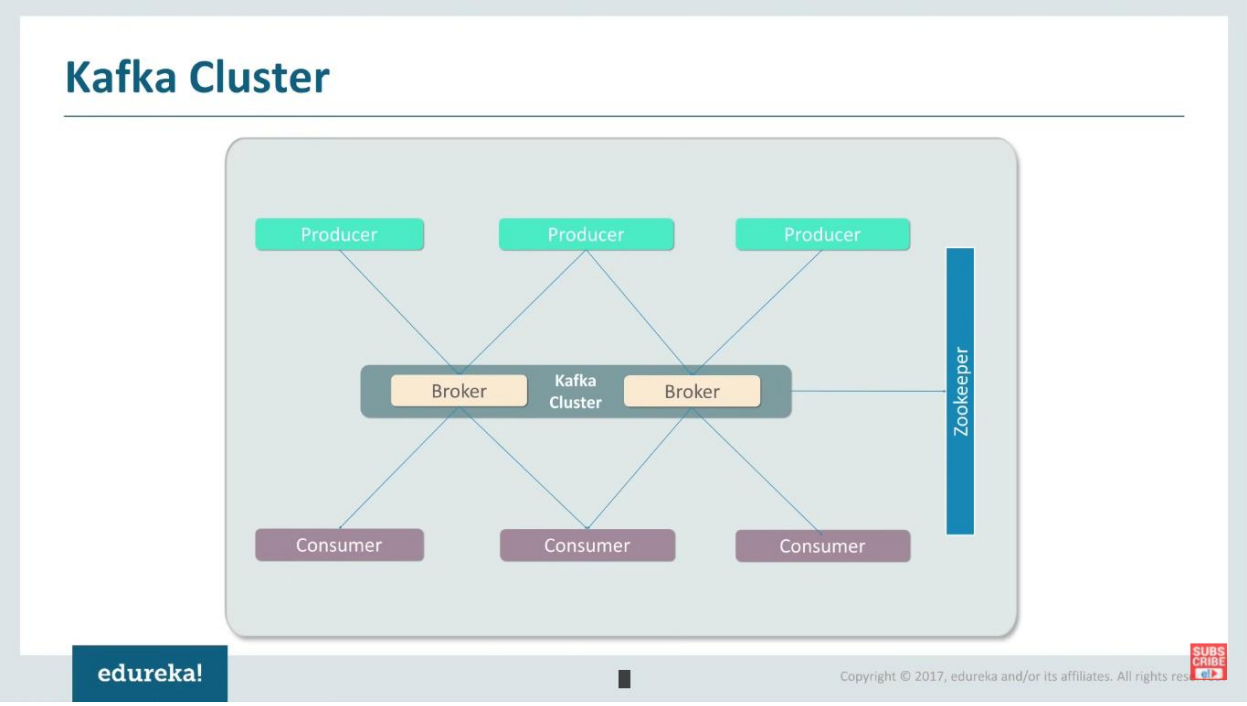
Source: Google Trends

edureka!



Topic where – events are published and from where consumers can consume those events .

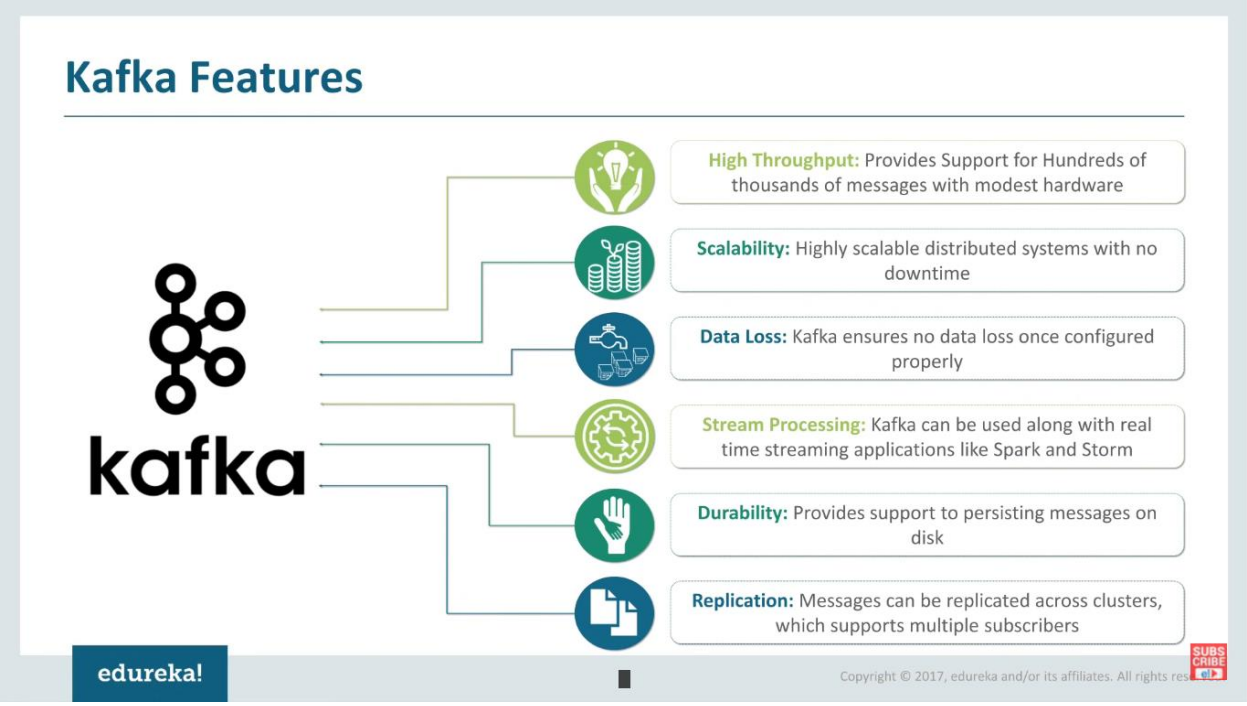
Topics can be further divided to partitions let suppose if its divided to 3 partitions , then 3 consumers can consume messages parllely .



Zookeeper stores the metadata information of kafka cluster .

Ex : Broker info , kafka topics e.t.c

This kafka cluster is managed by Zookeeper

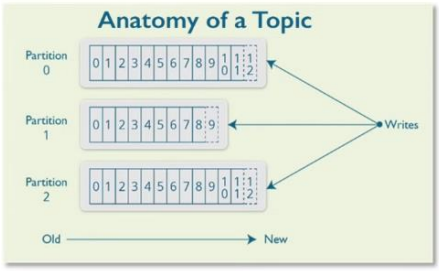


Note : we can have a single consumer can consume the messaged from a single partition. Else we are not able to figure out the sequence of messages .

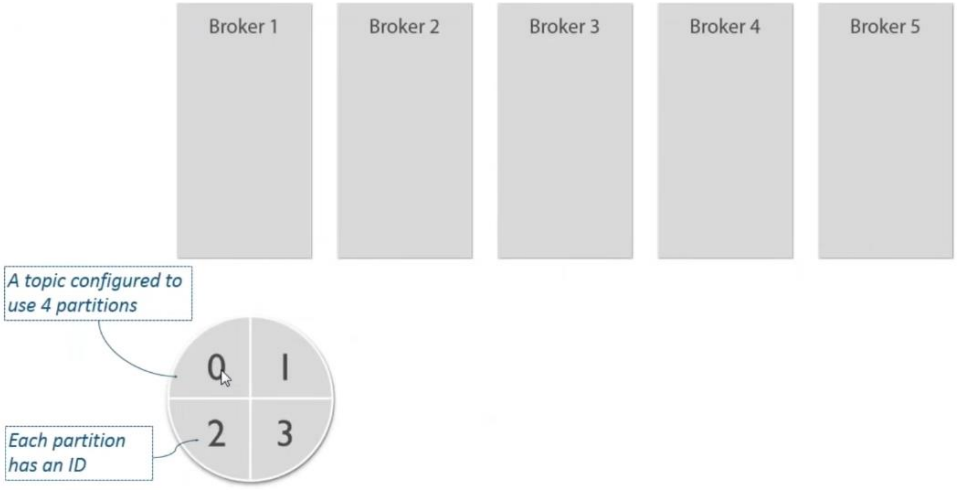
Durability – We can store that data fro backup purpose as well it will not only in memory data.

Kafka Components - Topics and Partitions

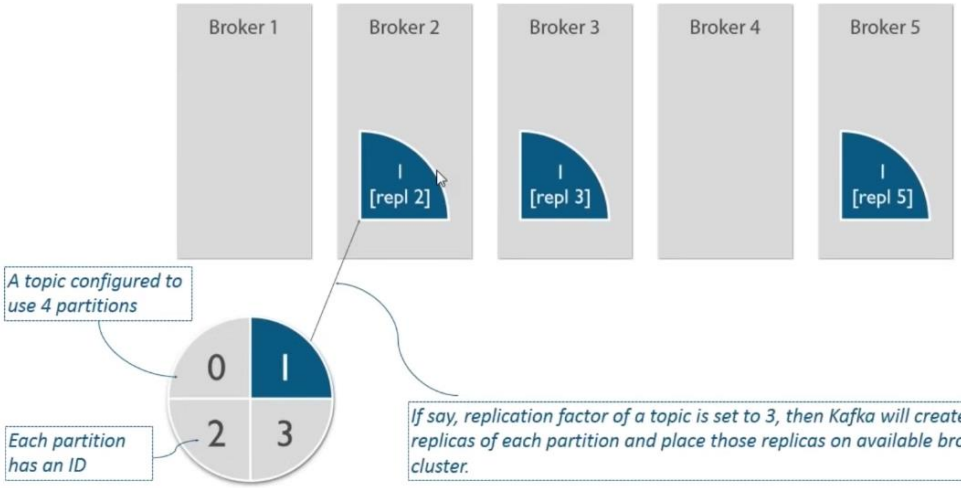
- A *topic* is a category or *feed name* to which *records are published*
- *Topics* are *broken* up into *ordered commit logs* called *partitions*
- Each *message* in a *partition* is assigned a *sequential id* called an *offset*
- Data in a topic is retained for a *configurable period of time*
- *Writes* to a partition *are* generally *sequential* thereby *reducing* the *number of hard disk seeks*
- *Reading messages* can either be from *beginning* & *also* can *rewind* or *skip* to any *point* in partition by *giving* an *offset value*



Kafka Components - Topics, Partitions & Replicas

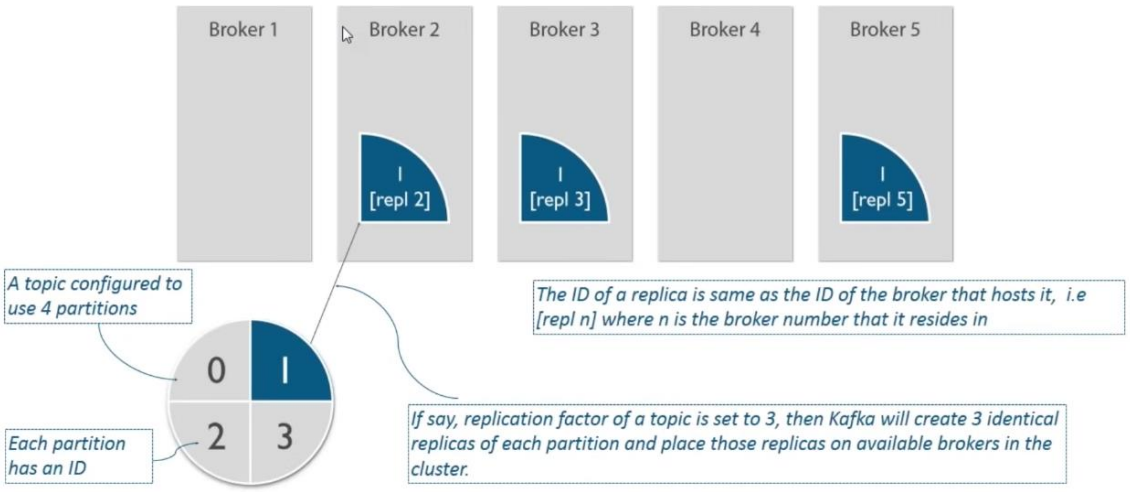


Kafka Components - Topics, Partitions & Replicas

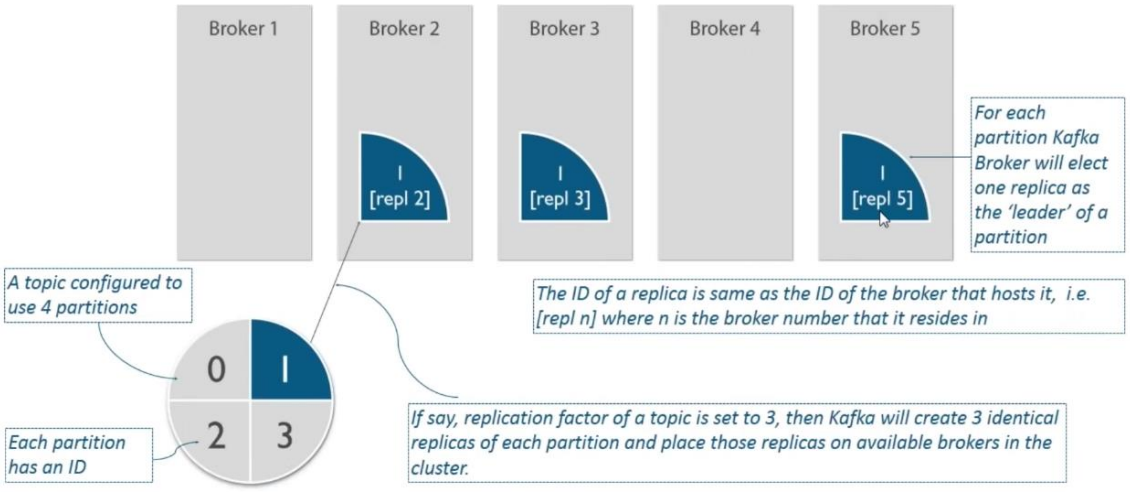


edureka!

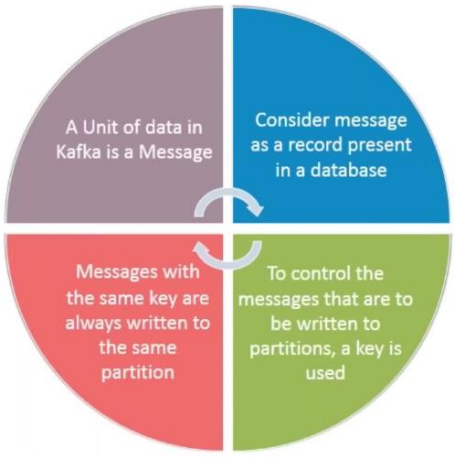
Kafka Components - Topics, Partitions & Replicas



Kafka Components - Topics, Partitions & Replicas

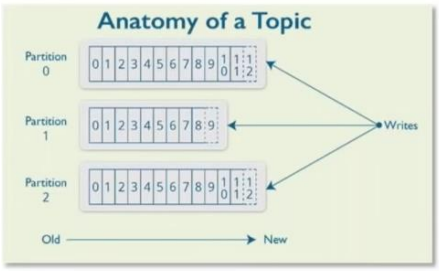


Kafka Components - Messages



Kafka Components - Topics and Partitions

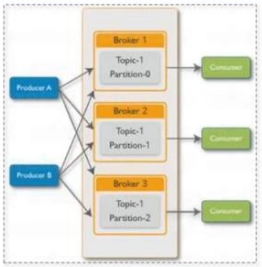
- A *topic* is a category or *feed name* to which *records are published*
- *Topics* are *broken* up into *ordered commit logs* called *partitions*
- Each *message* in a *partition* is assigned a *sequential id* called an *offset*
- Data in a topic is retained for a *configurable period of time*
- *Writes* to a partition *are* generally *sequential* thereby *reducing* the *number of hard disk seeks*
- *Reading messages* can either be from *beginning* & *also* can *rewind* or *skip* to any *point* in partition by *giving* an *offset value*



Kafka Components - Producer

1

Producer (publisher or writer) publishes a new message to a **specific topic**

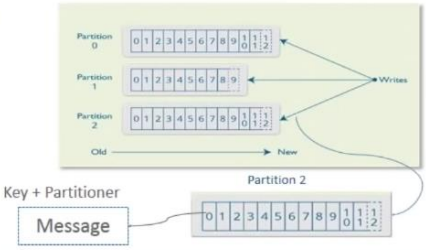


2

The producer does not care what partition a specific message is written to and will balance messages over every partition of a topic evenly

3

Directing messages to a partition is done using the **message key** and a **partitioner**, this will generate a hash of the key and map it to a partition

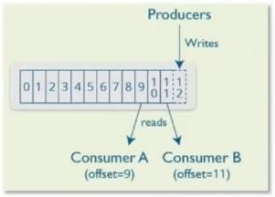


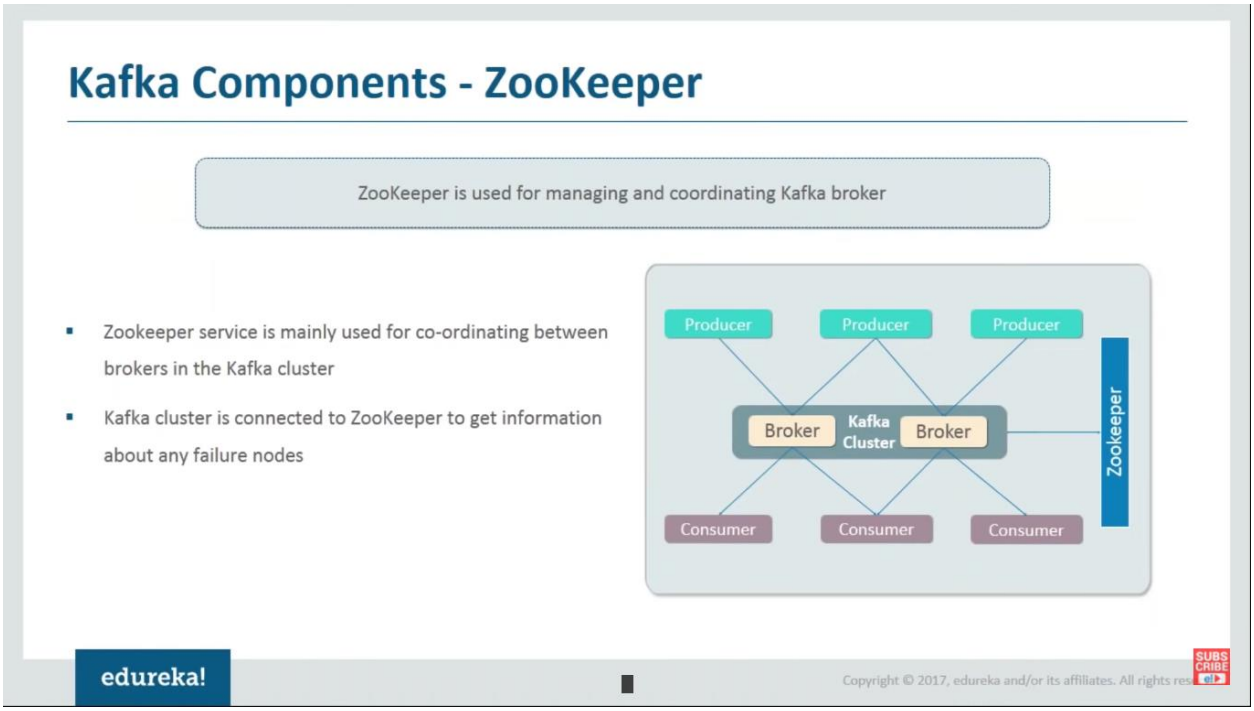
4

Every message a producer publishes in the form of a **key : value** pair

Kafka Components - Consumer

- *Consumers*(subscribers or readers) read messages
- The consumer subscribes to one or more topics and reads the messages sequentially
- The consumer keeps track of the messages it has consumed by keeping track on the offset of messages
- The *offset* is bit of metadata(an integer value that continually increases)that Kafka adds to each message as it is produced
- Each partition has a *unique offset* which is stored
- With the offset of the last consumed message, a consumer can *stop and restart without losing its current state*





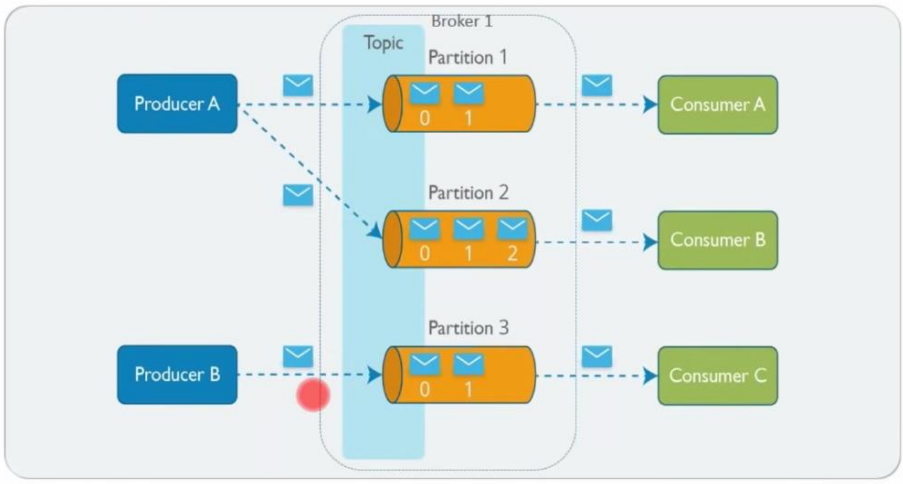
Zoopkeeper elects the leader from the various partition , if any leader got down , then it automatically remove it from its directory and elect a new leader .

Fault Taulerence

Deep Dive into Kafka Architecture

edureka!

Kafka Architecture



Let's see some Use Cases of Kafka

Kafka - Use Cases

- Applications can produce messages using Kafka, without being concerned about the format of the messages
- Messages are sent and handled by a single application that can read all of them consistently, including :
  - A common formatting of messages using a common look
  - Send multiple messages in a single notification
  - Receive messages in a way that meets the users preferences

Kafka - Use Cases

- Originally Kafka was designed at LinkedIn, to track user activity
- When a user interacts with frontend applications, which generates messages regarding actions the user is taking
- Kafka keeps track of simple information like click tracking to complex information like data in a user's profile

Kafka - Use Cases

- Kafka is also ideal for collecting application's and system metrics and logs
- Applications publish metrics on a regular basis to a Kafka topic, and those metrics can be consumed by systems for monitoring and alerting
- Log messages can be published in the same way and routed to dedicated log search systems like Elasticsearch or security analysis applications

Kafka - Use Cases

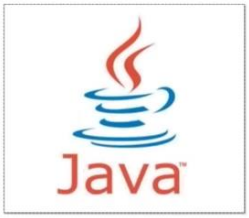
- Database changes can be published to Kafka and applications can easily monitor this stream to receive live updates as they happen
- Kafka replicates database updates to a remote system for consolidating changes from multiple applications in a single database view
- Durable retention becomes useful providing a buffer for the changelog, meaning it can be replayed in the event of a failure of the consuming applications
- Log-compacted topics can be used to provide longer retention by only retaining a single change per key

Kafka - Use Cases

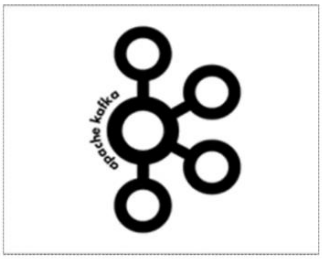
- Stream processing term is typically used to refer applications that provide similar functionality to map/reduce processing in Hadoop
- Stream processing operates on data in real-time, as quickly as messages are produced :
  - Write small applications to operate on Kafka messages,
  - Performing tasks such as counting metrics
  - Partitioning messages for efficient processing by other applications

Getting Started with Kafka

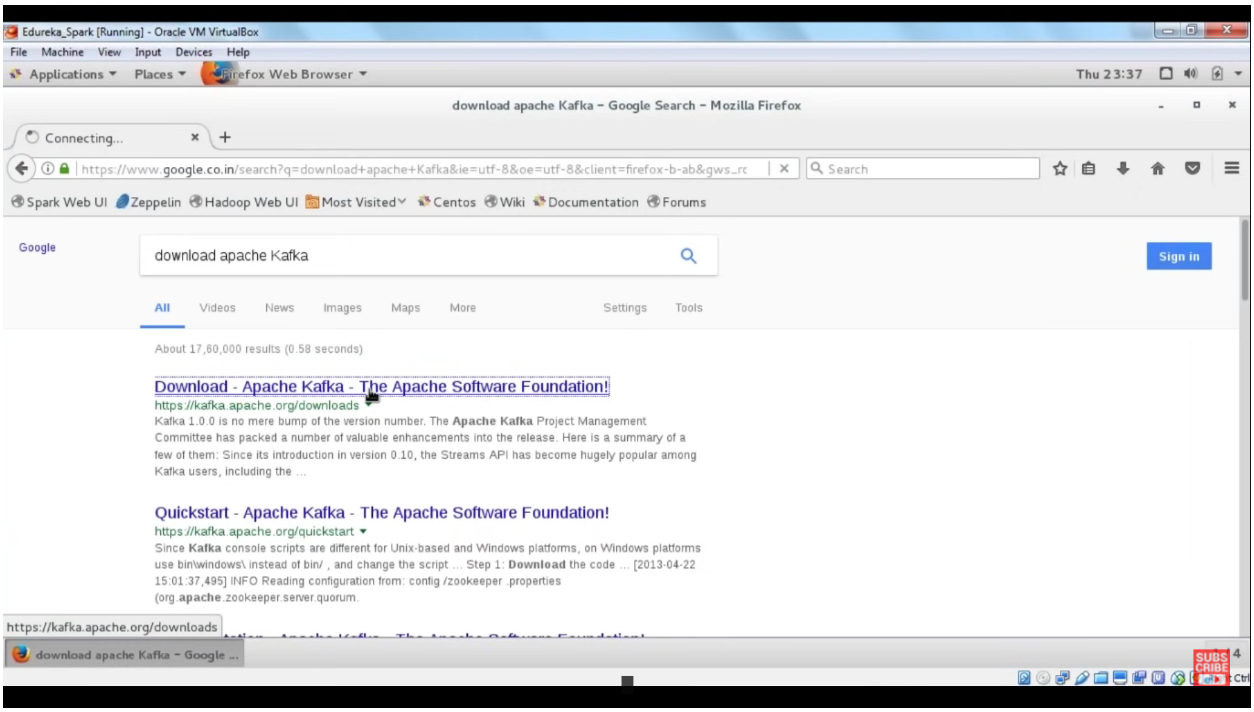
- Prerequisites :

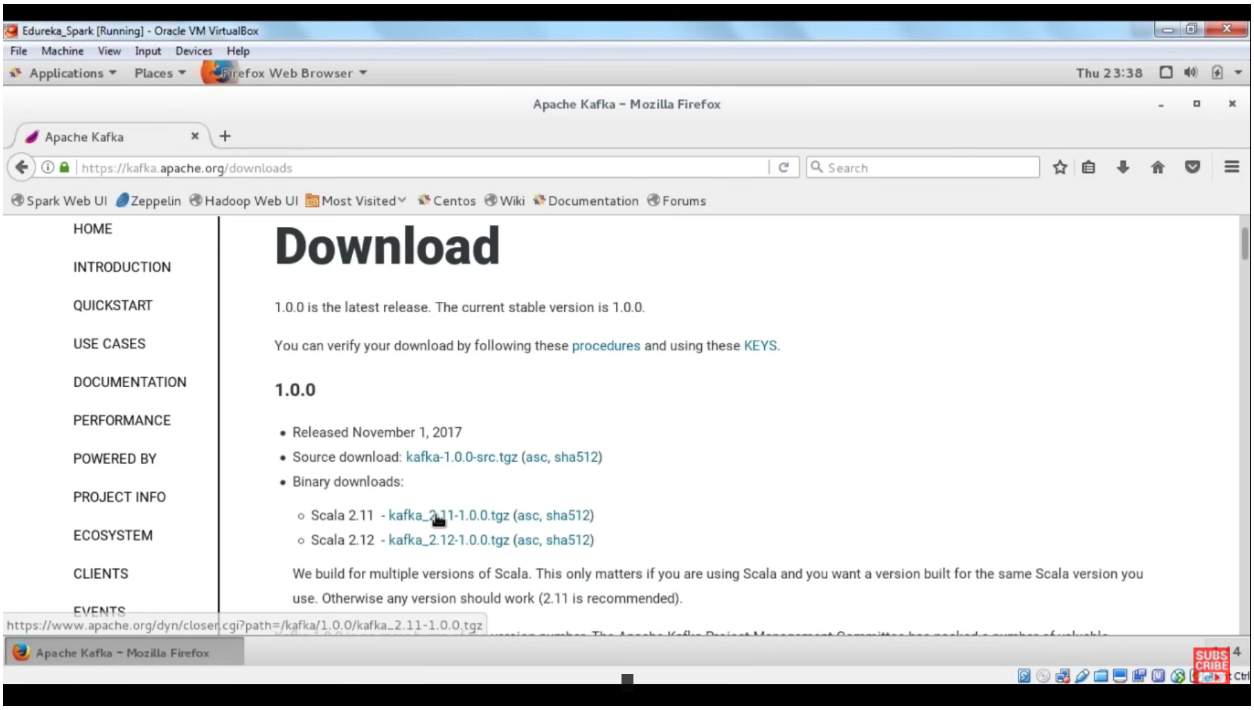


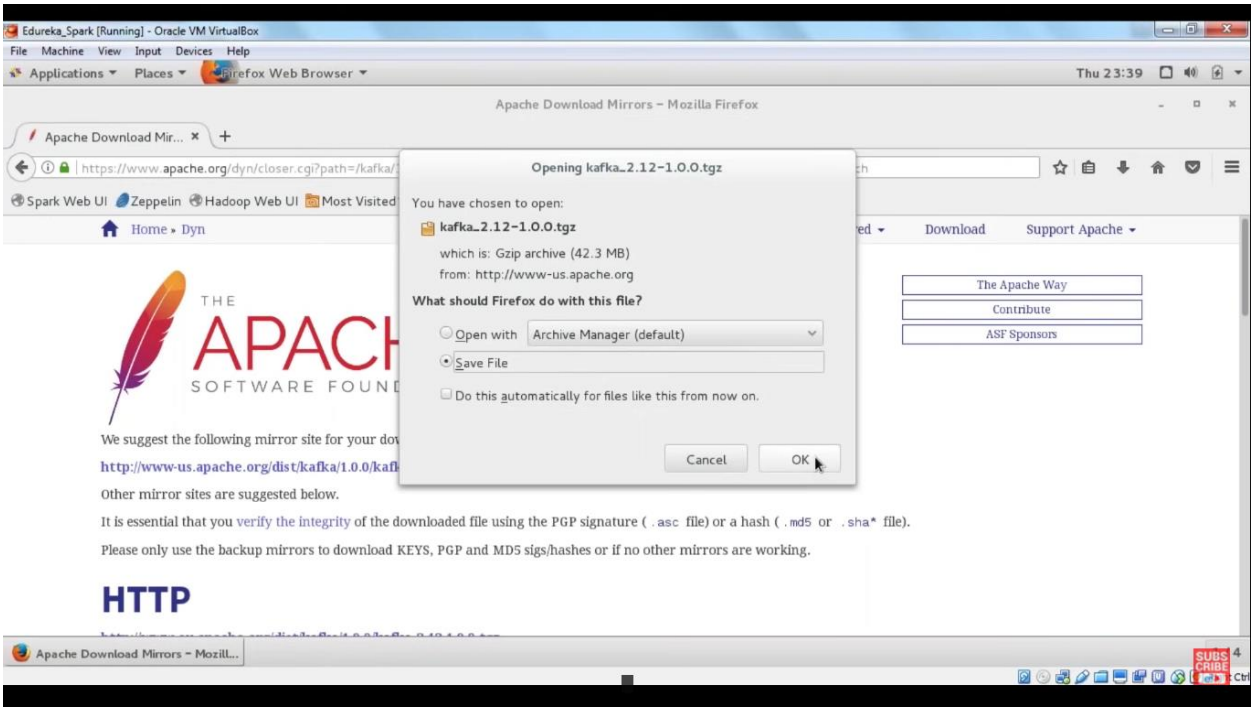
- Components :



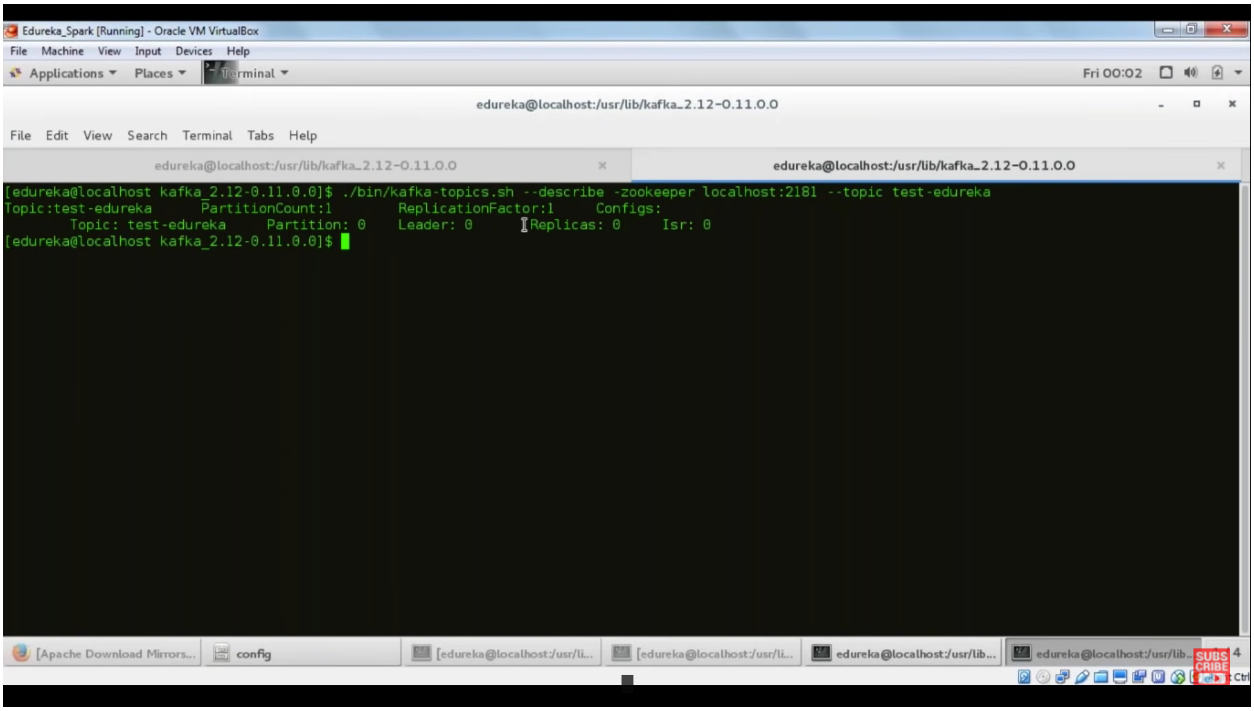
edureka!

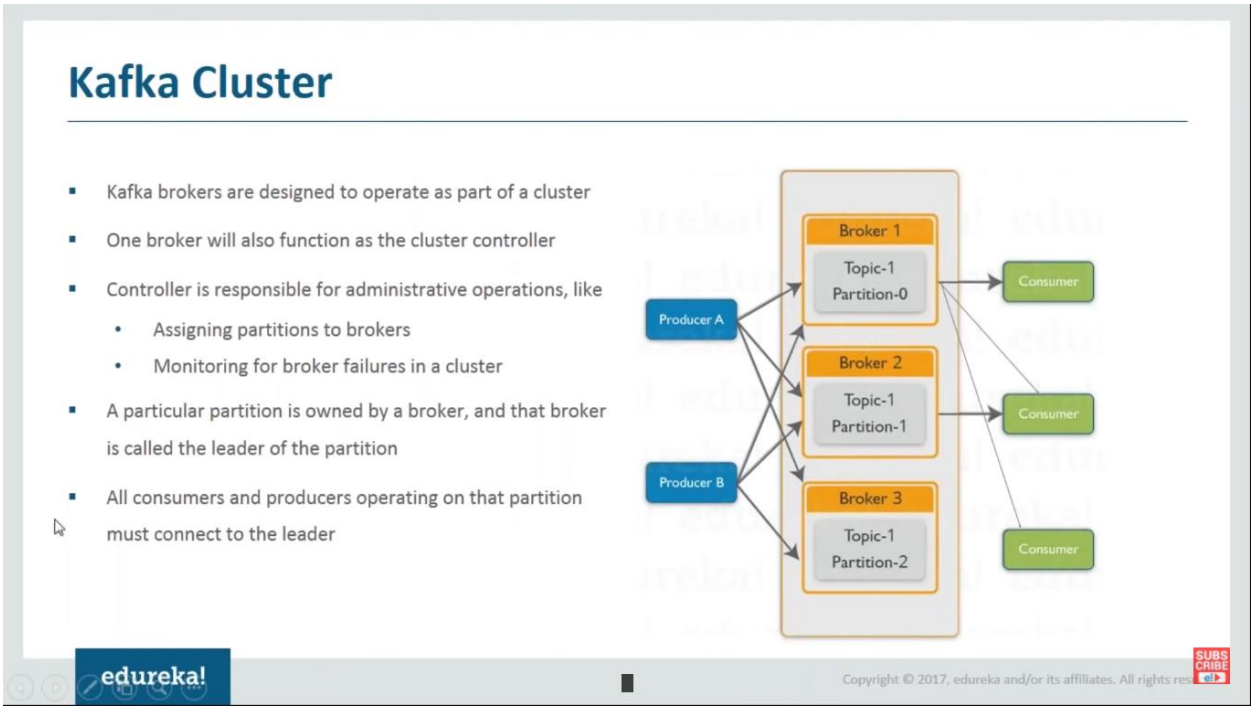


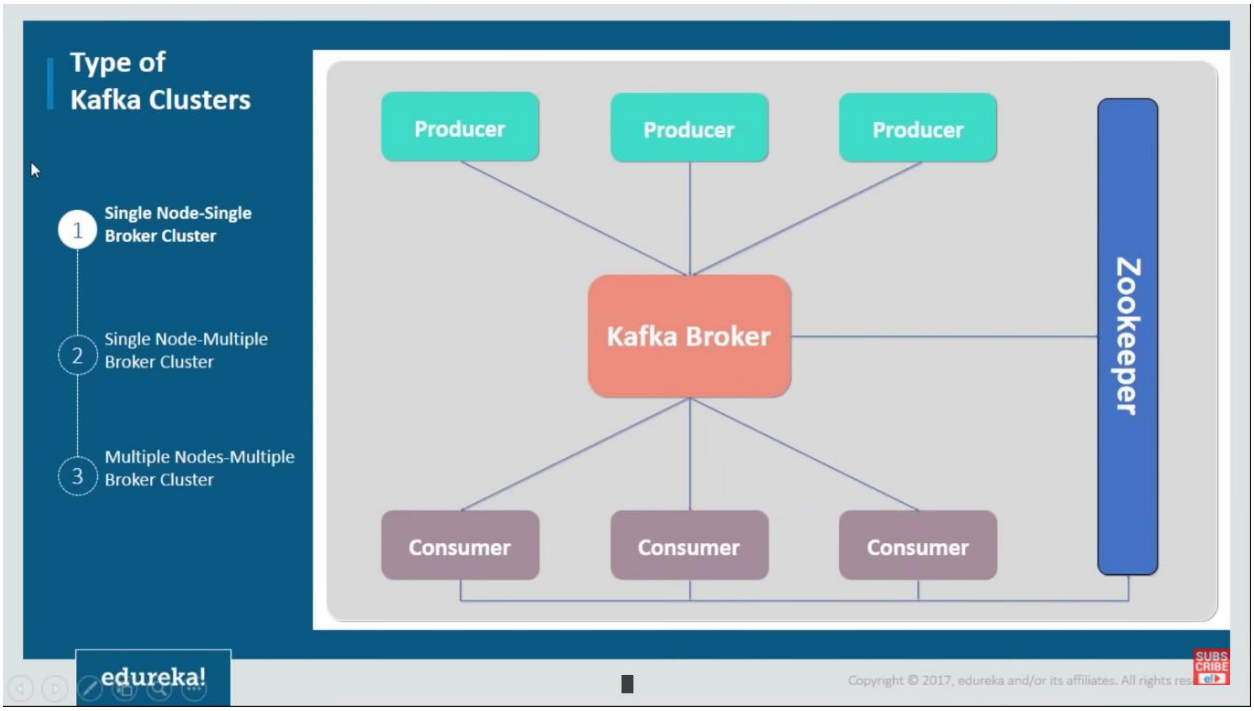

Edureka_Spark [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Firefox Web Browser
Thu 23:38
Apache Kafka – Mozilla Firefox
Apache Kafka
https://kafka.apache.org/downloads
Search
Spark Web UI Zeppelin Hadoop Web UI Most Visited Centos Wiki Documentation Forums
HOME
INTRODUCTION
QUICKSTART
USE CASES
DOCUMENTATION
PERFORMANCE
POWERED BY
PROJECT INFO
ECOSYSTEM
CLIENTS
EVENTS
Download
1.0.0 is the latest release. The current stable version is 1.0.0.
You can verify your download by following these procedures and using these KEYS.
1.0.0
Released November 1, 2017
Source download: kafka-1.0.0-src.tgz (asc, sha512)
Binary downloads:
Scala 2.11 - kafka_2.11-1.0.0.tgz (asc, sha512)
Scala 2.12 - kafka_2.12-1.0.0.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 (2.11 is recommended).
https://www.apache.org/dyn/closer.cgi?path=/kafka/1.0.0/kafka_2.11-1.0.0.tgz
Apache Kafka – Mozilla Firefox

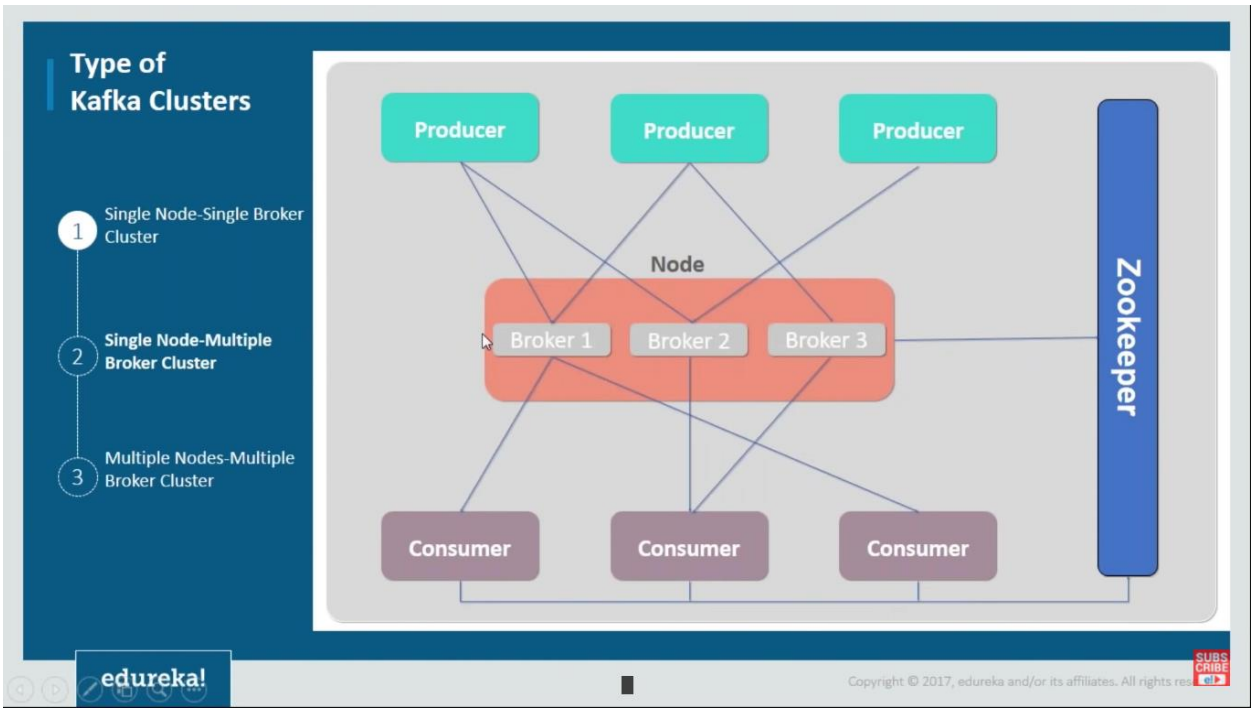


Note : uncomment the port line


Edureka_Spark [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal
Fri 00:02
edureka@localhost:/usr/lib/kafka_2.12-0.11.0.0
File Edit View Search Terminal Tabs Help
edureka@localhost:/usr/lib/kafka_2.12-0.11.0.0
edureka@localhost:/usr/lib/kafka_2.12-0.11.0.0
[edureka@localhost kafka_2.12-0.11.0.0]$ ./bin/kafka-topics.sh --describe -zookeeper localhost:2181 --topic test-edureka
Topic:test-edureka PartitionCount:1 ReplicationFactor:1 Configs:
Topic: test-edureka Partition: 0 Leader: 0 Replicas: 0 Isr: 0
[edureka@localhost kafka_2.12-0.11.0.0]$
[Apache Download Mirrors...
config
[edureka@localhost:/usr/li...
[edureka@localhost:/usr/li...
edureka@localhost:/usr/lib...
edureka@localhost:/usr/lib...
SUBSCRIBE

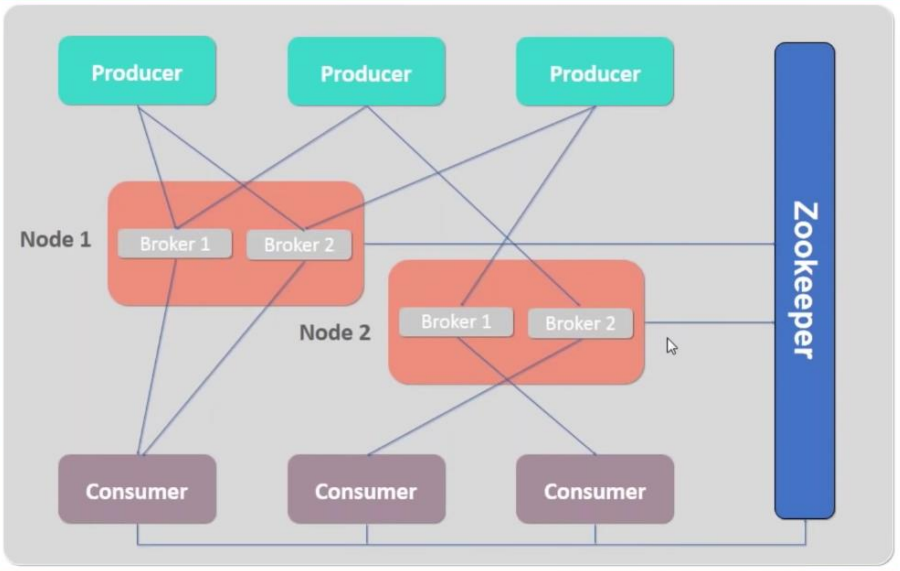

Kafka Cluster
Kafka brokers are designed to operate as part of a cluster
One broker will also function as the cluster controller
Controller is responsible for administrative operations, like
Assigning partitions to brokers
Monitoring for broker failures in a cluster
A particular partition is owned by a broker, and that broker is called the leader of the partition
All consumers and producers operating on that partition must connect to the leader
Producer A
Producer B
Broker 1
Topic-1 Partition-0
Broker 2
Topic-1 Partition-1
Broker 3
Topic-1 Partition-2
Consumer
Consumer
Consumer
edureka!
Copyright © 2017, edureka and/or its affiliates. All rights res
SUBSCRIBE


Type of
Kafka Clusters
1 Single Node-Single Broker Cluster
2 Single Node-Multiple Broker Cluster
3 Multiple Nodes-Multiple Broker Cluster
Producer
Producer
Producer
Kafka Broker
Zookeeper
Consumer
Consumer
Consumer
edureka!
Copyright © 2017, edureka and/or its affiliates. All rights res


Type of
Kafka Clusters
1 Single Node-Single Broker Cluster
2 Single Node-Multiple Broker Cluster
3 Multiple Nodes-Multiple Broker Cluster
Producer
Producer
Producer
Node
Broker 1
Broker 2
Broker 3
Zookeeper
Consumer
Consumer
Consumer
edureka!
Copyright © 2017, edureka and/or its affiliates. All rights res

Type of Kafka Clusters

1. Single Node-Single Broker Cluster
2. Single Node-Multiple Broker Cluster
3. **Multiple Nodes-Multiple Broker Cluster**



edureka!

Let's configure a Single Node-Single Broker Cluster

Single Node Single Broker Cluster

Step 1 : Start the Zookeeper Server :

```
Command: bin/zkServer.sh start
```

```
[edureka@localhost zookeeper-3.4.10]$ bin/zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /lib/zookeeper-3.4.10/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

Step 2 : Start the Kafka Server

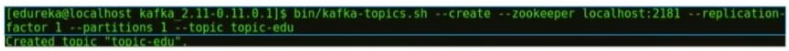
```
Command: bin/kafka-server-start.sh config/server.properties
```

```
[edureka@localhost kafka_2.11-0.11.0.1]$ bin/kafka-server-start.sh config/server.properties
```

Single Node Single Broker Cluster

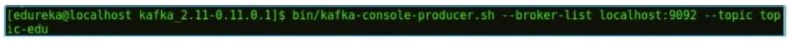
Step 3 : Create a Topic with the name "topic-edu"

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



Step 4 : Start a Producer

```
Command: bin/kafka-console-producer.sh --broker-list localhost:9092 --topic
                                        topic-edu
```



edureka!

Single Node Single Broker Cluster

Step 5 : Start a Consumer

```
Command: bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic
                        topic-name --from-beginning
```

```
[edureka@localhost kafka_2.11-0.11.0.1]$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic topic-edu --from-beginning
```

Step 6 : Producer can publish a message to a topic which will be received by the consumer which has subscribed to that topic

Producer

```
[edureka@localhost kafka_2.11-0.11.0.1]$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic topic-edu
>My name is parth
>Hello parth
>how are you
>i m gud
```

Consumer

```
[edureka@localhost kafka_2.11-0.11.0.1]$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic topic-edu --from-beginning
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [
bootstrap-server] instead of [zookeeper].
My name is parth
Hello parth
how are you
i m gud
```

edureka!

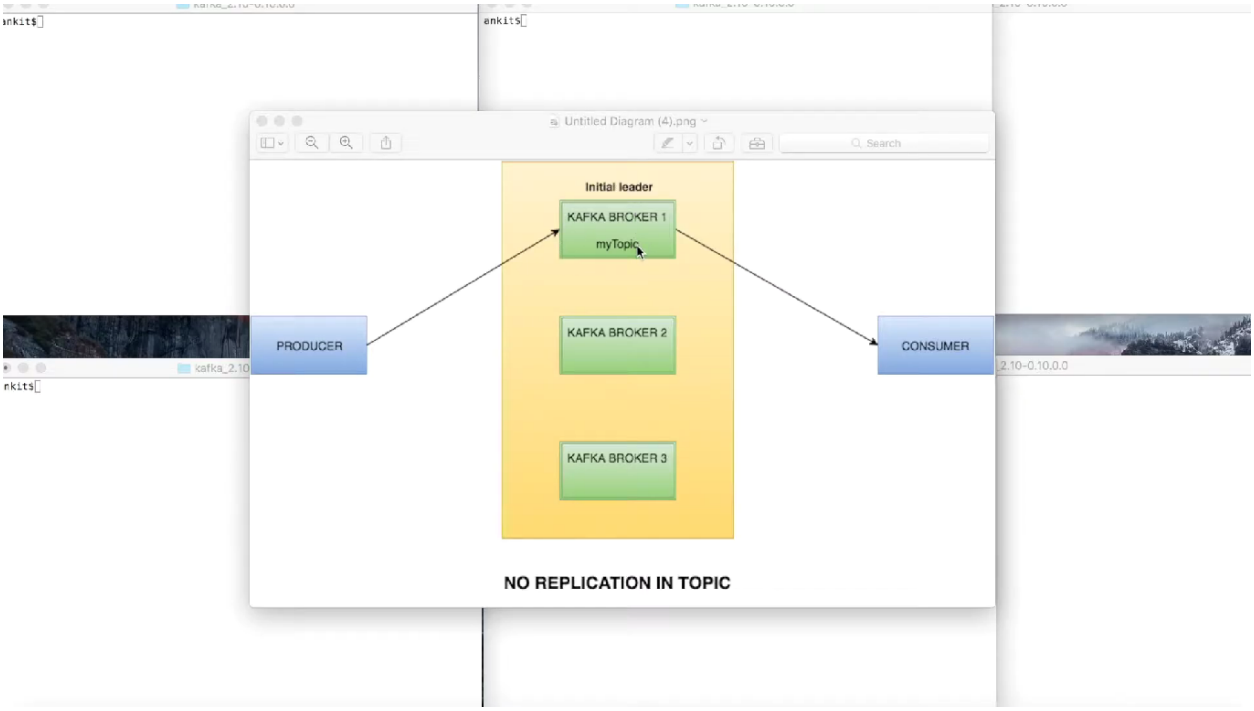
Kafka Course Outline

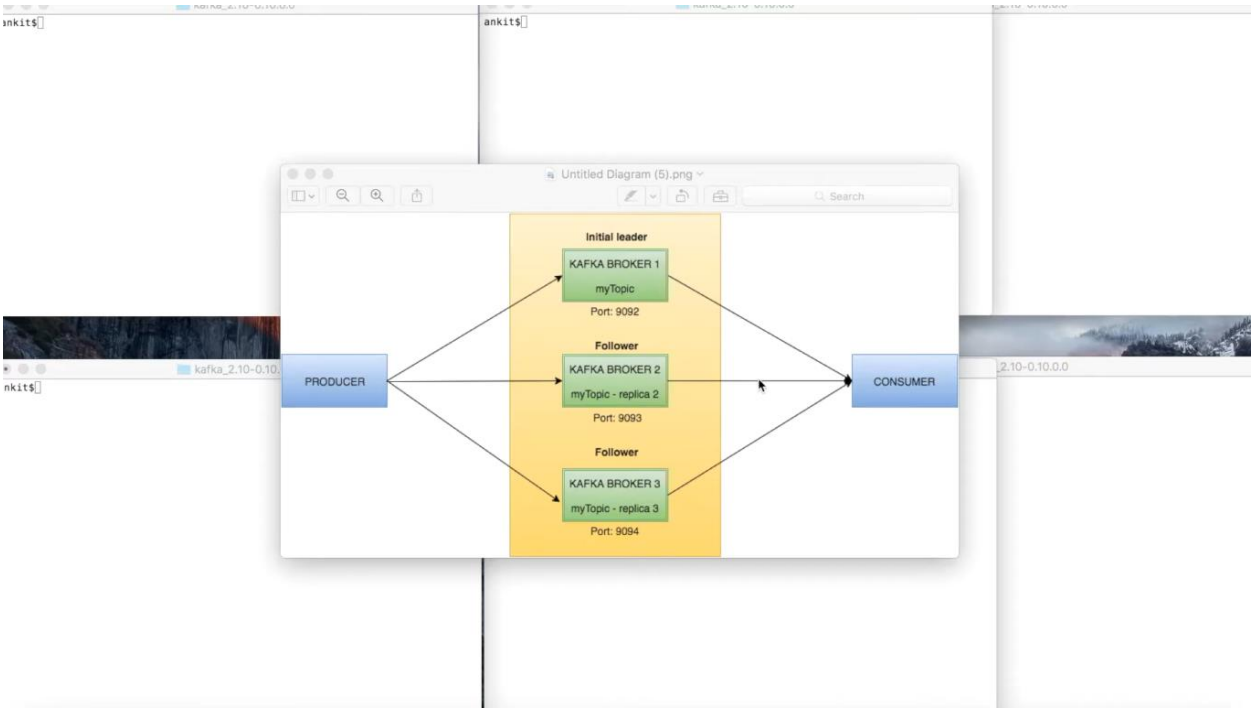
edureka!

Follow file for demo

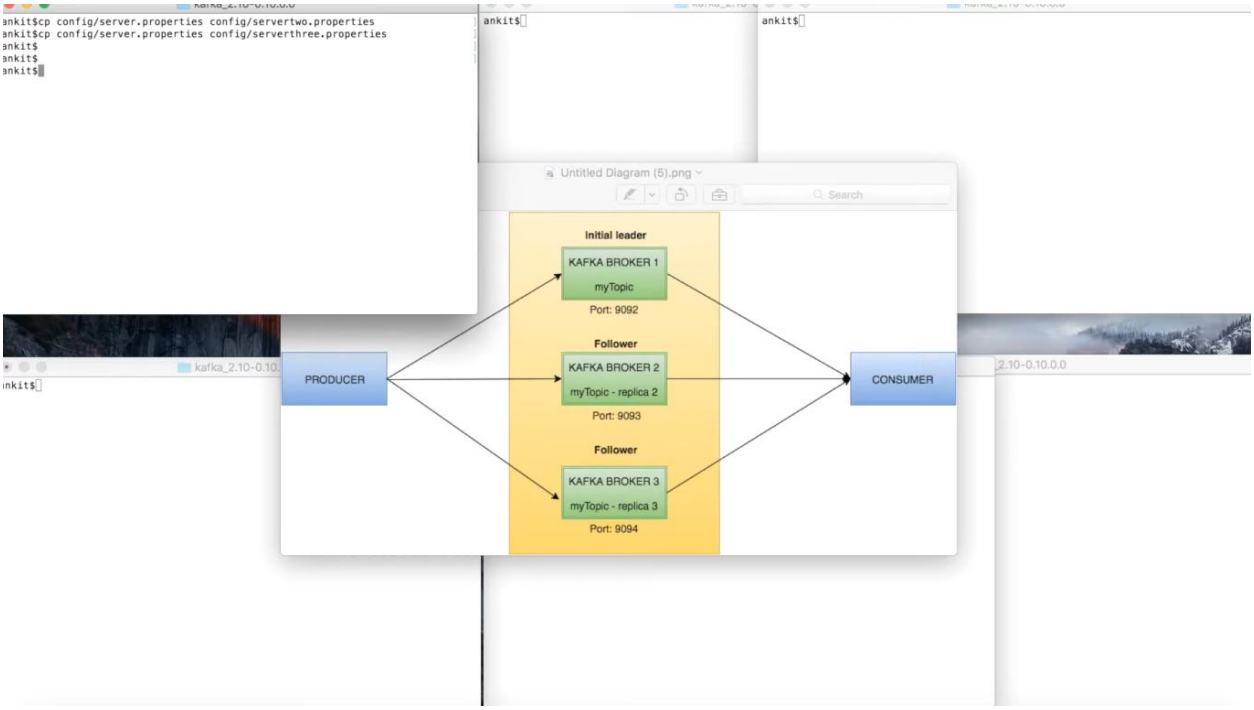
(Includes all commands)

Second Demo :


Untitled Diagram (4).png
Search
Initial leader
KAFKA BROKER 1
myTopic
KAFKA BROKER 2
KAFKA BROKER 3
PRODUCER
CONSUMER
NO REPLICATION IN TOPIC

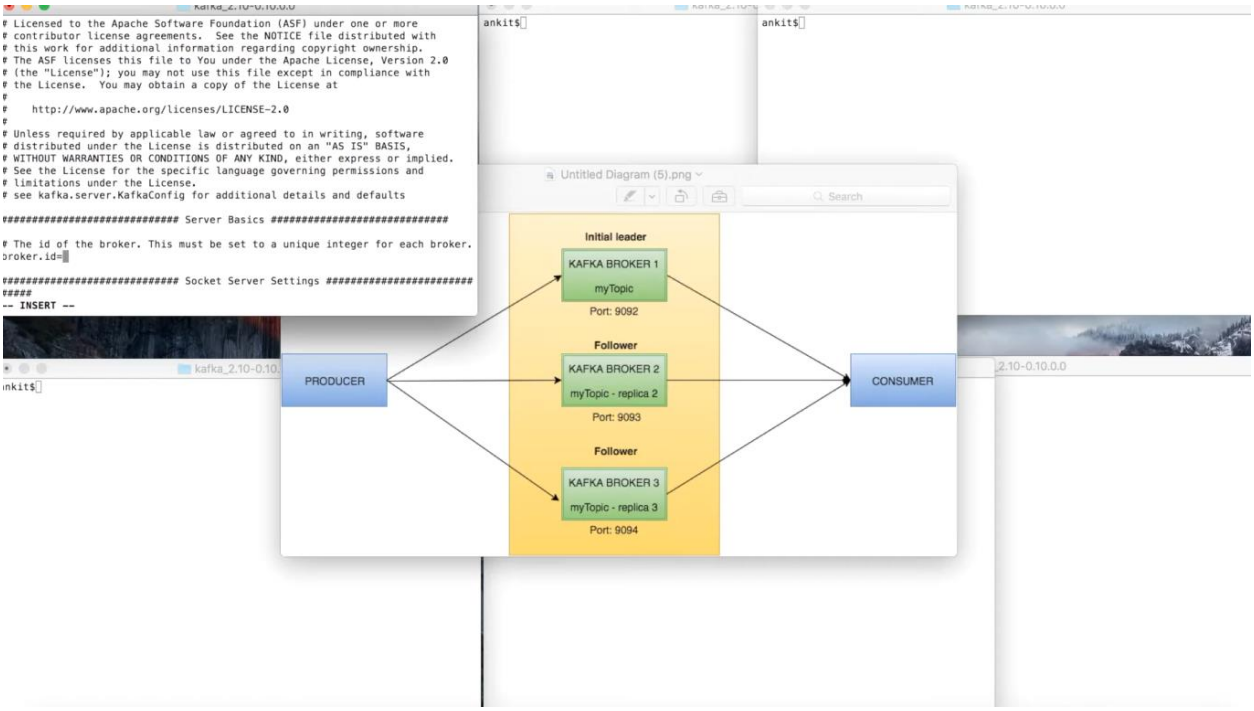

Untitled Diagram (5).png
Search
Initial leader
KAFKA BROKER 1
myTopic
Port: 9092
Follower
KAFKA BROKER 2
myTopic - replica 2
Port: 9093
Follower
KAFKA BROKER 3
myTopic - replica 3
Port: 9094
PRODUCER
CONSUMER

To make it as fault taulerent we need to replicate it to multiple broker (meaning that multiple replica of same topic)

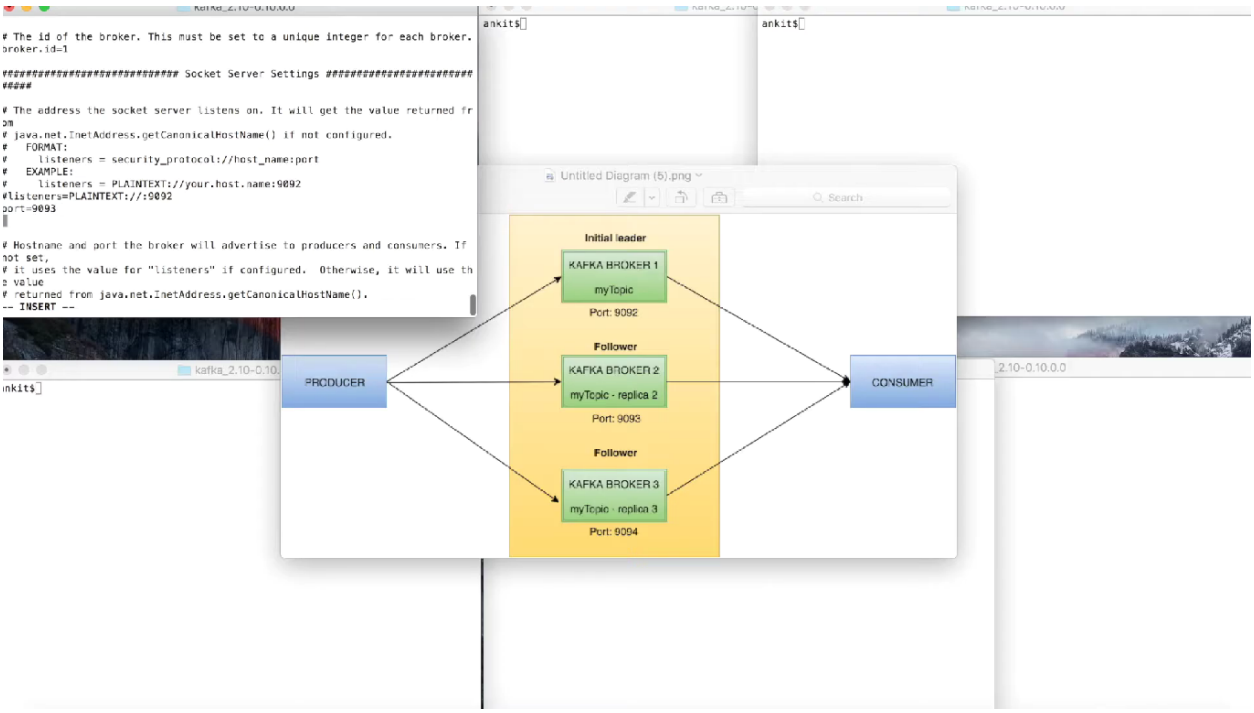


Create 2 more copied of servers.property files

Edit the port to make it diff as we running this demo in same machine

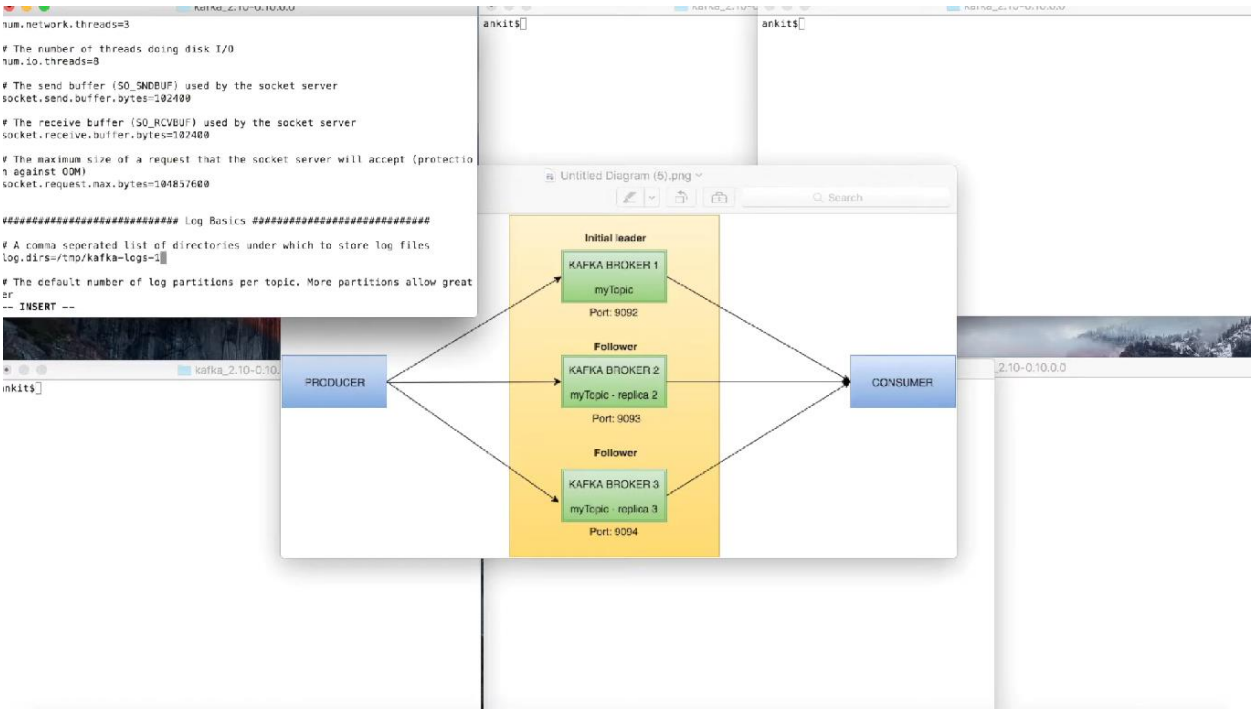


Change the broker id first = 1and 2 simultaneously



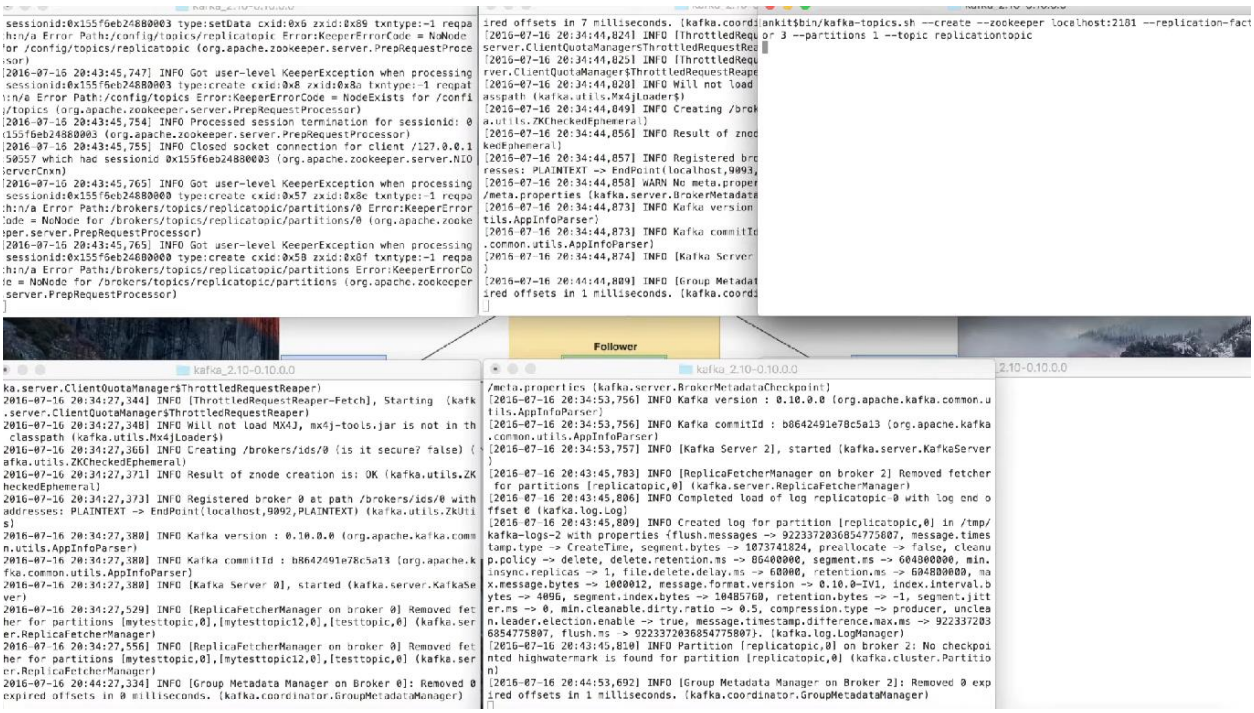
Change the port = 9093

And comment that line which we uncommented in single node architecture



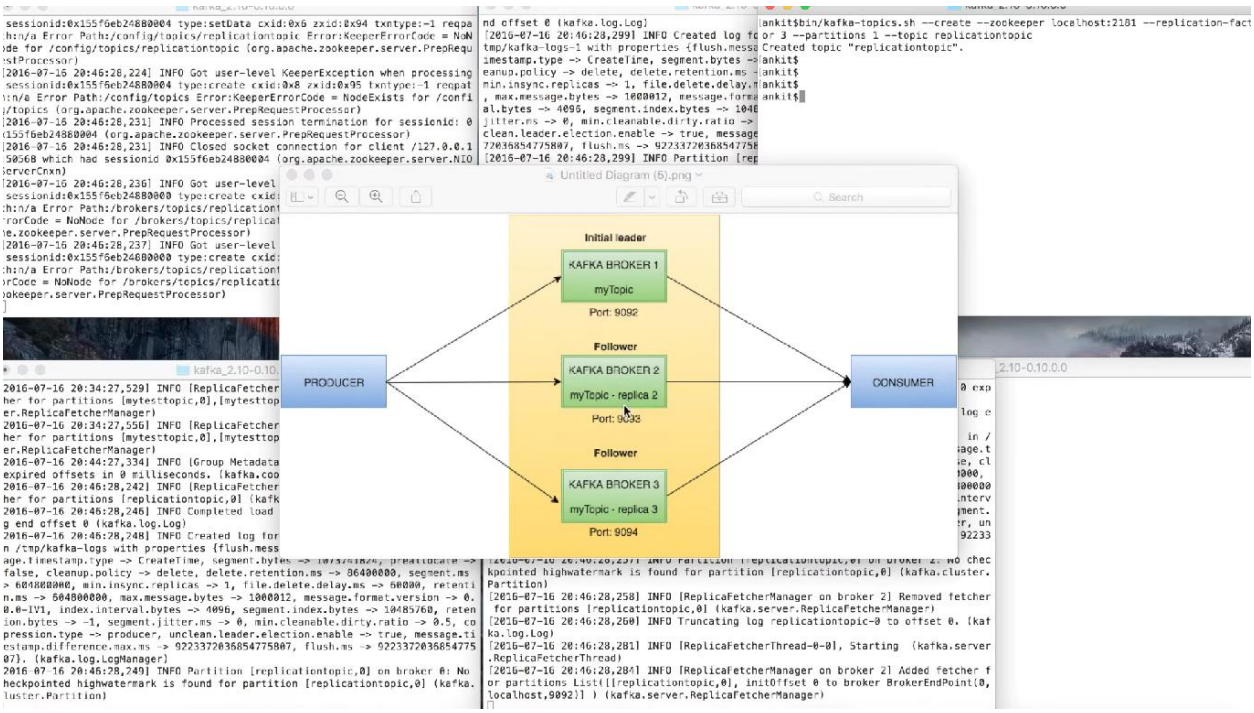
Change the log directory as well to logs-1 for this broker

Do this for third one also all three steps



Then up zookeeper and all three broker with diff server.properties

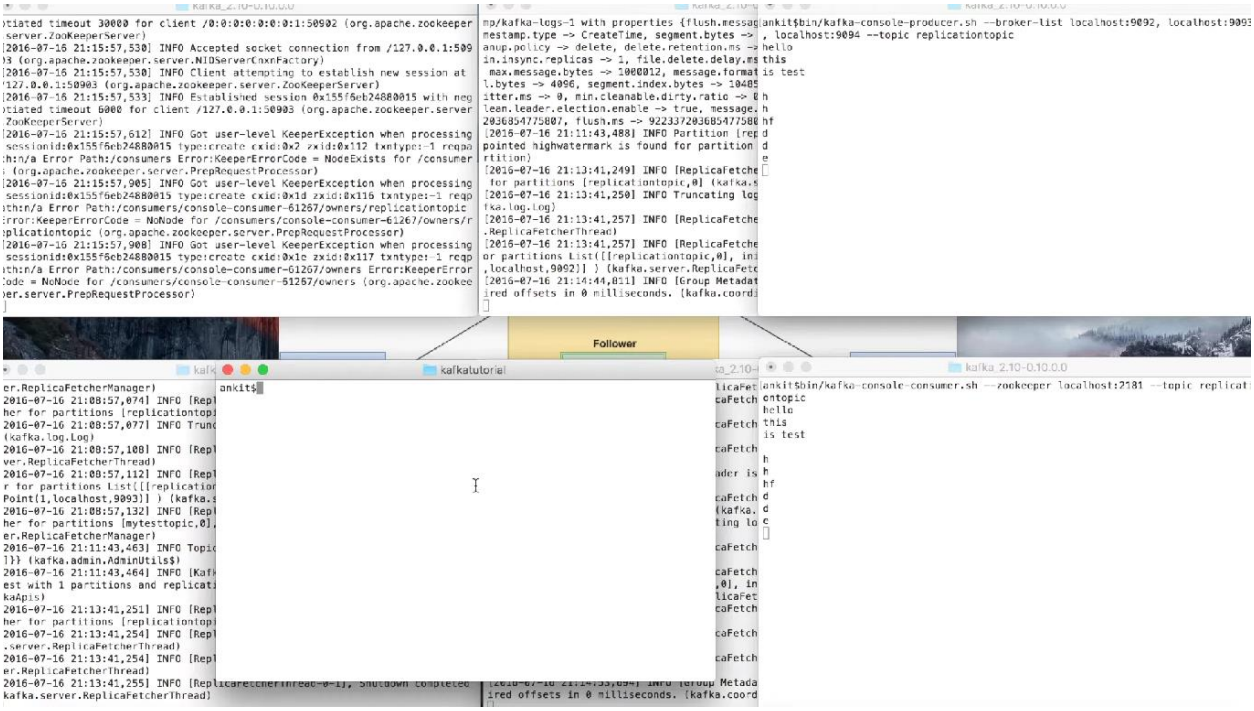
And then create topic with replication factor as 3

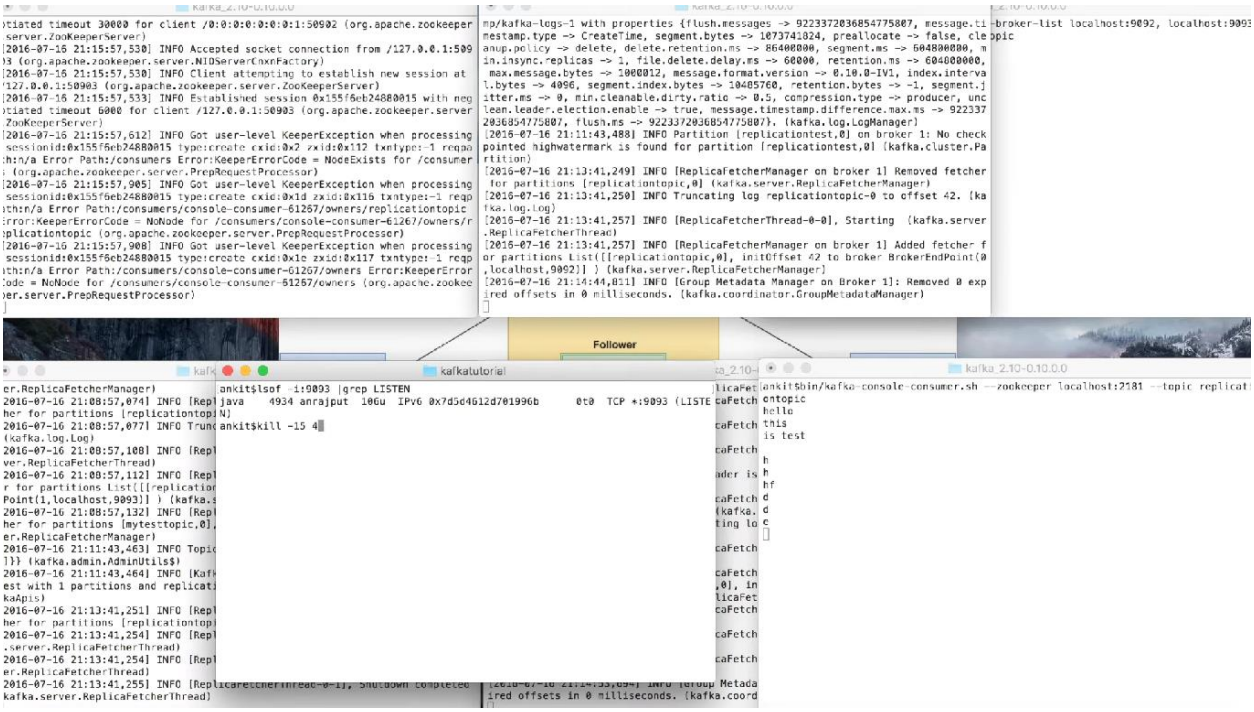


After all steps start producer now this time give all three broker while defining broker—list

And start consumer everything working fine ..

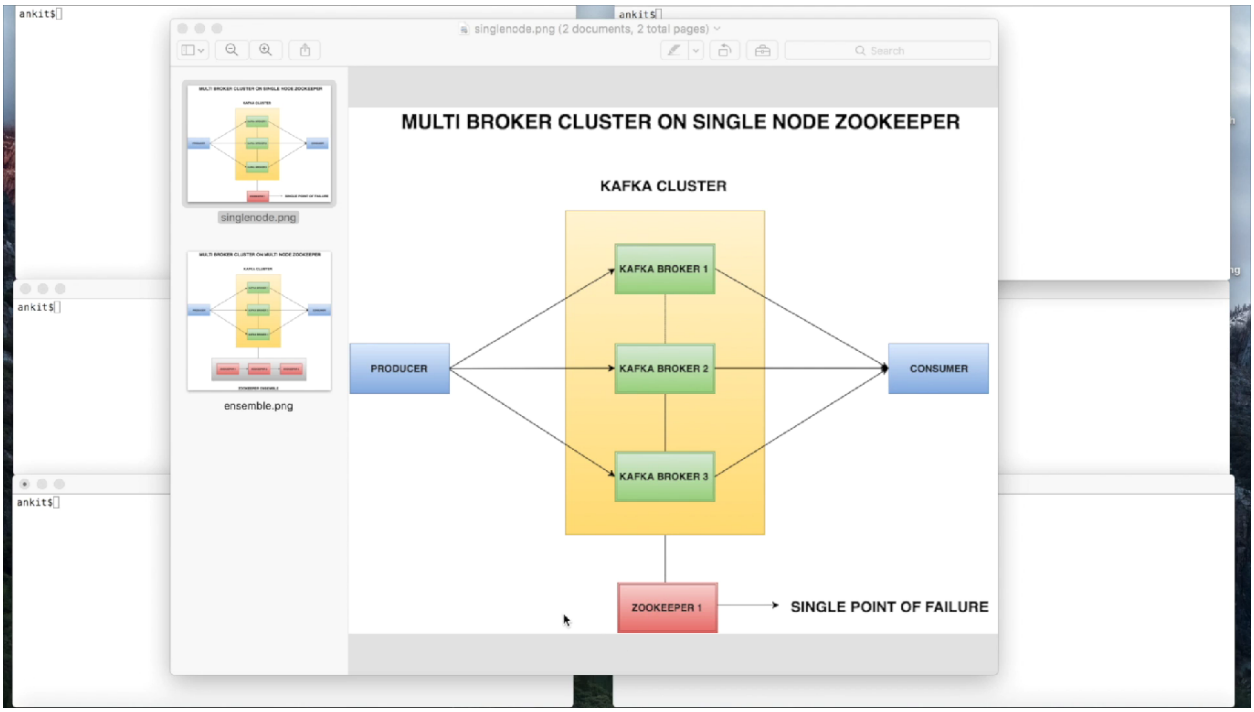
Now to check fault taulernece kill one of the broker and see


Follower
kafkatutorial
kafka_2.10-0.10.0.0

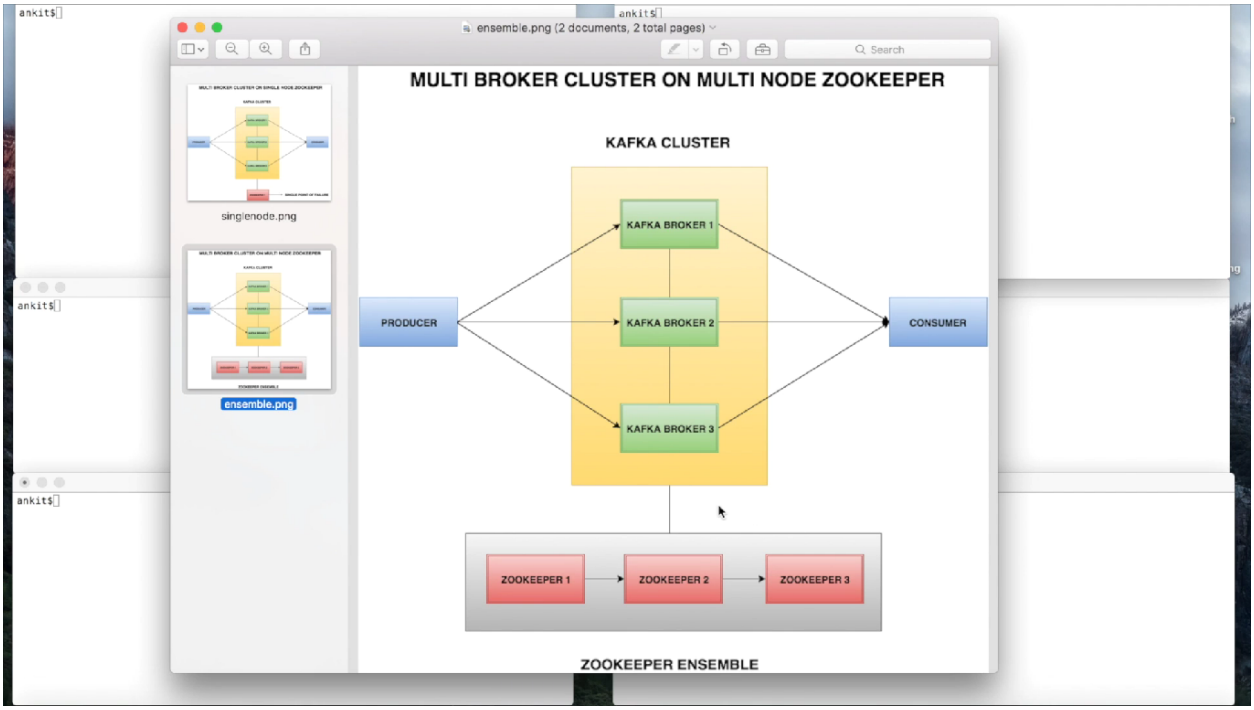

Follower
kafkatutorial
kafka_2.10-0.10.0.0

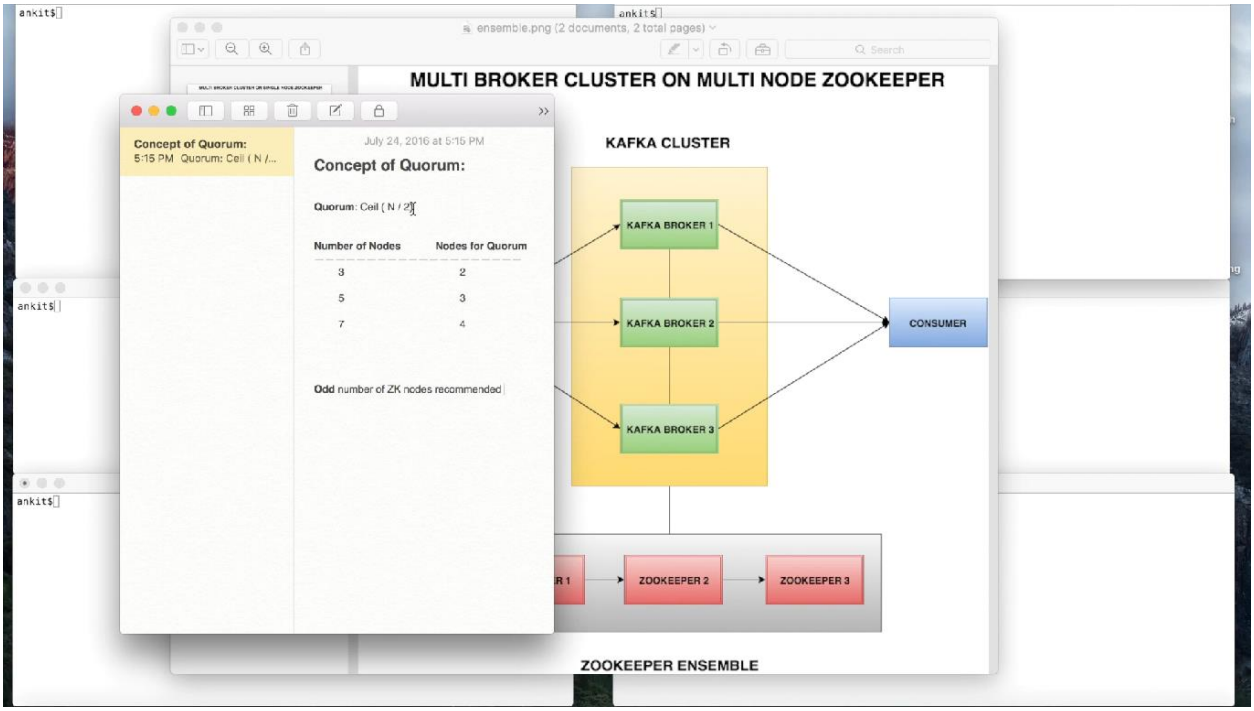
Still u able to work .

DEMO -3



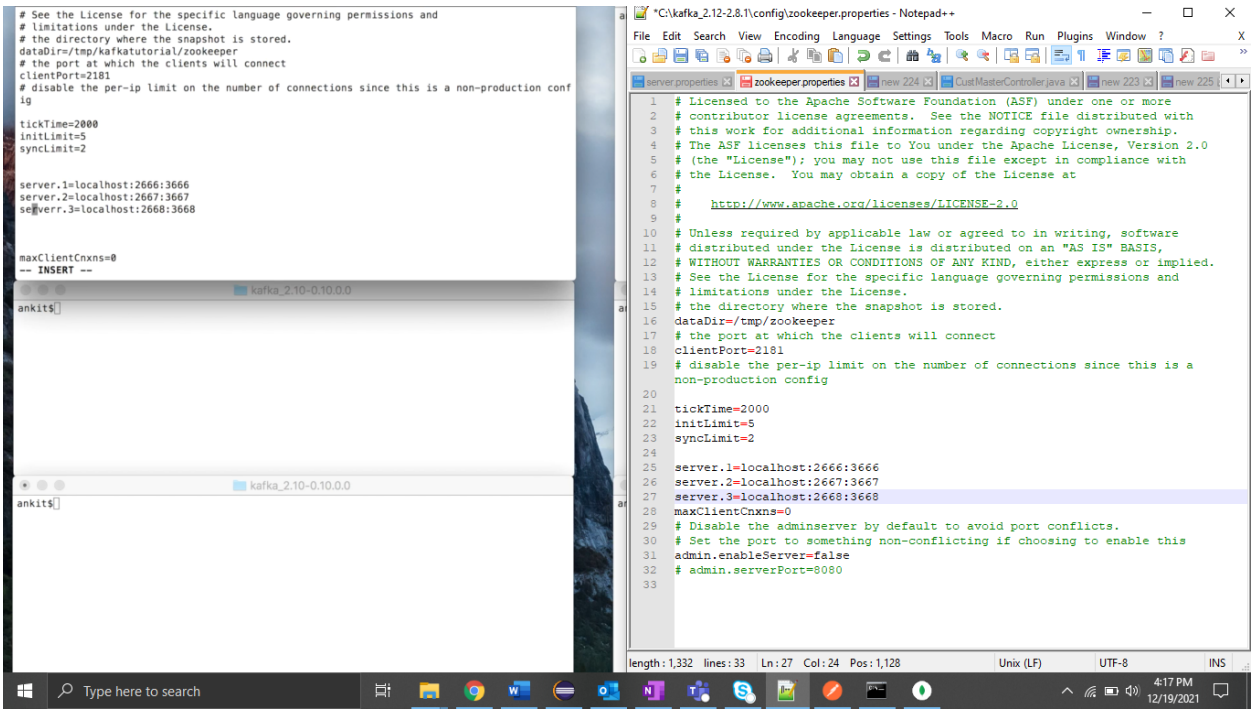
Multiple broker with multiple node zookeeper server


ankit$
ankit$
ankit$
ankit$
ensemble.png (2 documents, 2 total pages)
Search
singlenode.png
ensemble.png
MULTI BROKER CLUSTER ON MULTI NODE ZOOKEEPER
KAFKA CLUSTER
KAFKA BROKER 1
KAFKA BROKER 2
KAFKA BROKER 3
PRODUCER
CONSUMER
ZOOKEEPER 1
ZOOKEEPER 2
ZOOKEEPER 3
ZOOKEEPER ENSEMBLE



Note : Ideally in zoopkeeper documentation , we should keep odd no of brokers , this will help zookeeper to decide the leader better way instead of even no of broker (initially) or we can say odd one is more fault taulernet

Similarly odd no of zookeeper instances



Edit zookeeper.propeties : (Add below)

tickTime=2000

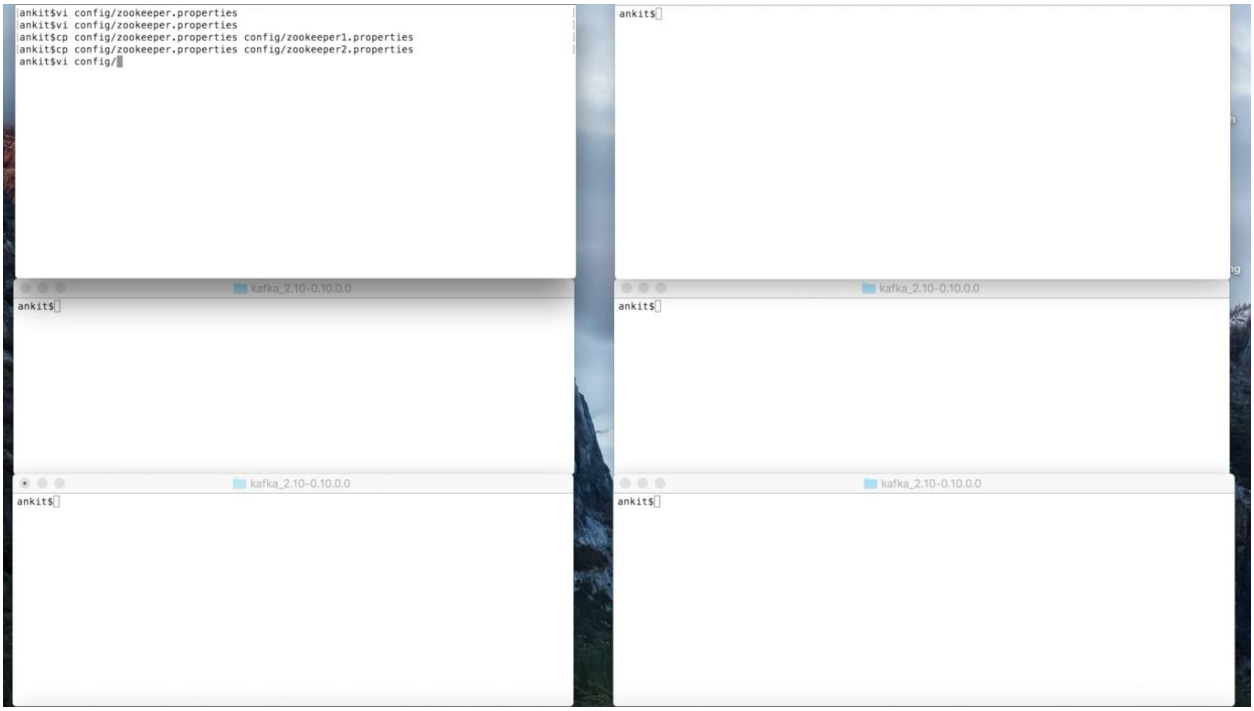
initLimit=5

syncLimit=2

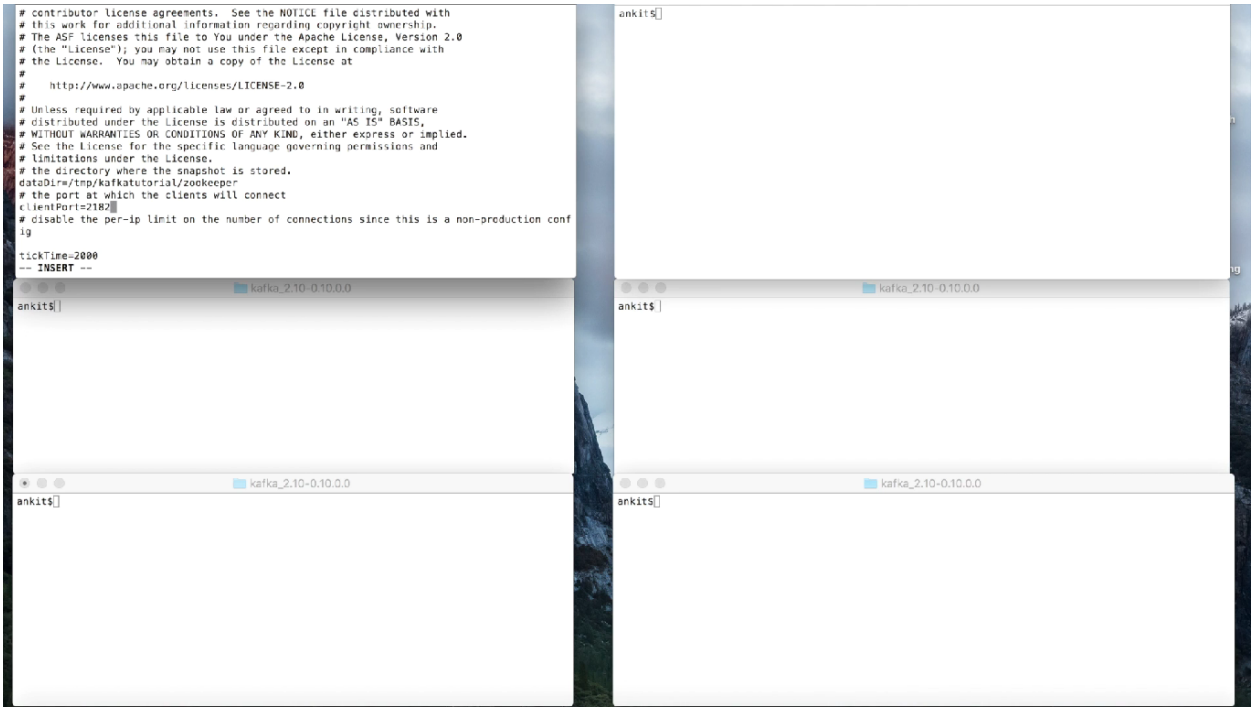
server.1=localhost:2666:3666

server.2=localhost:2667:3667

server.3=localhost:2668:3668



Make 2 more copied of zookeeper.prop file



Change the port from 2081 to 2082

Also change the data log directory

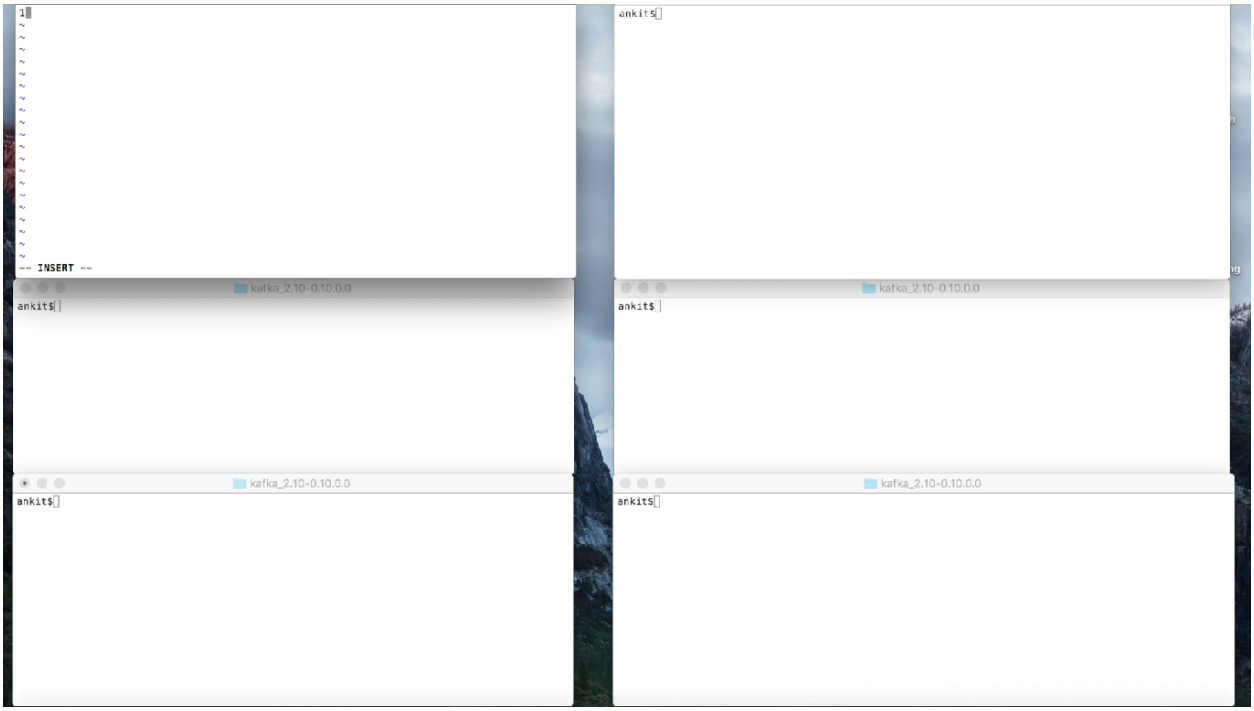
from

dataDir=/tmp/zookeeper

to

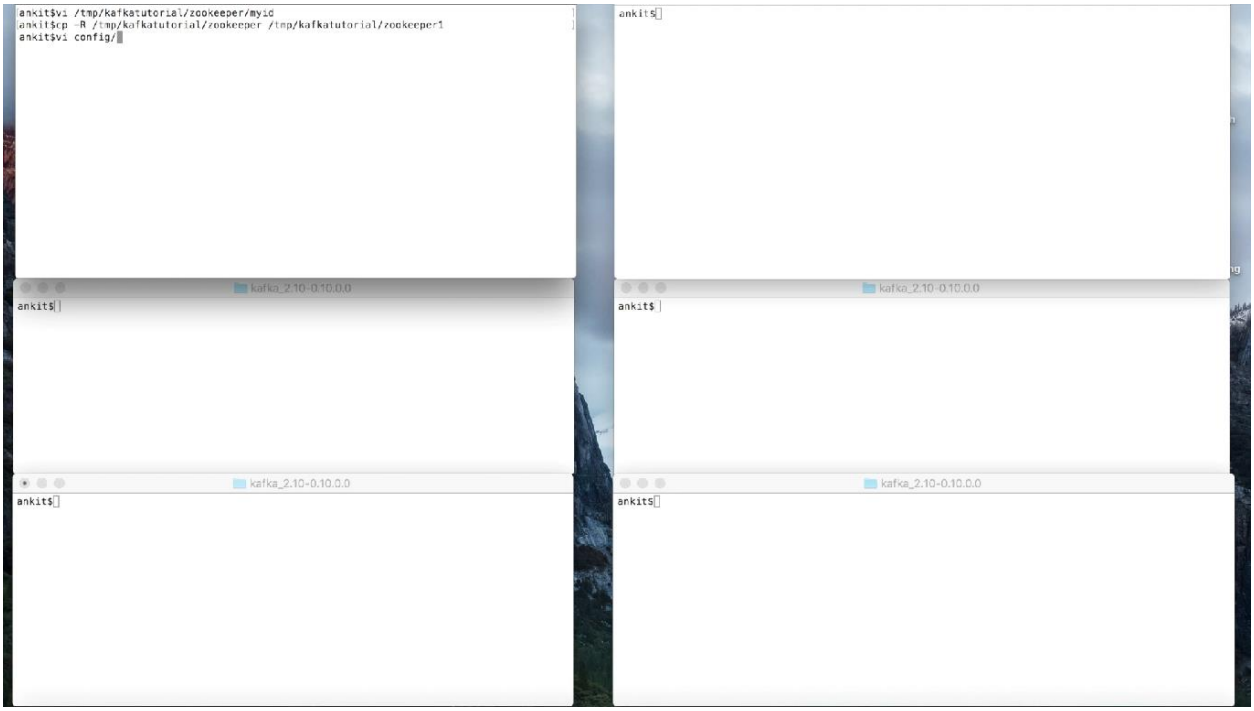
dataDir=/tmp/zookeeper1

Similarly do changes for 3 file 2083 , zookeper2



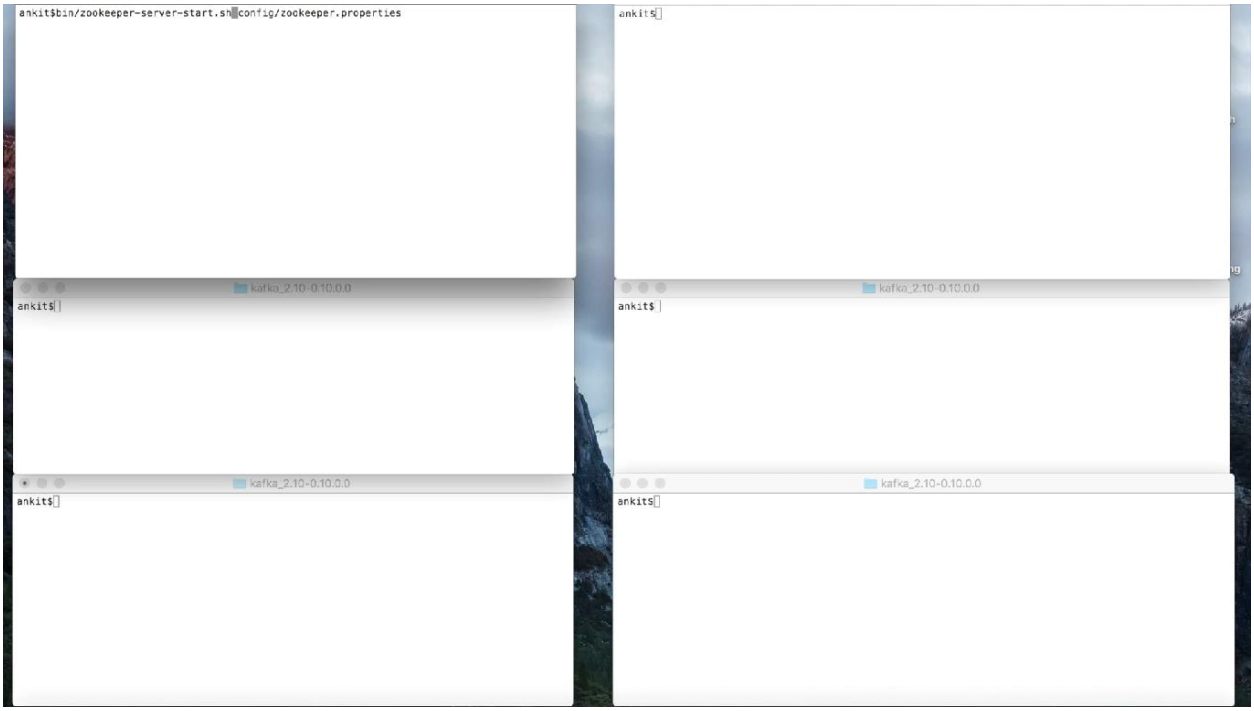
Now create file with name myid and enter 1 in it in existing zookeeper logs directory

Then copy the directory and paste it with diff name (what name u mentioned in zookeeper1.properties)



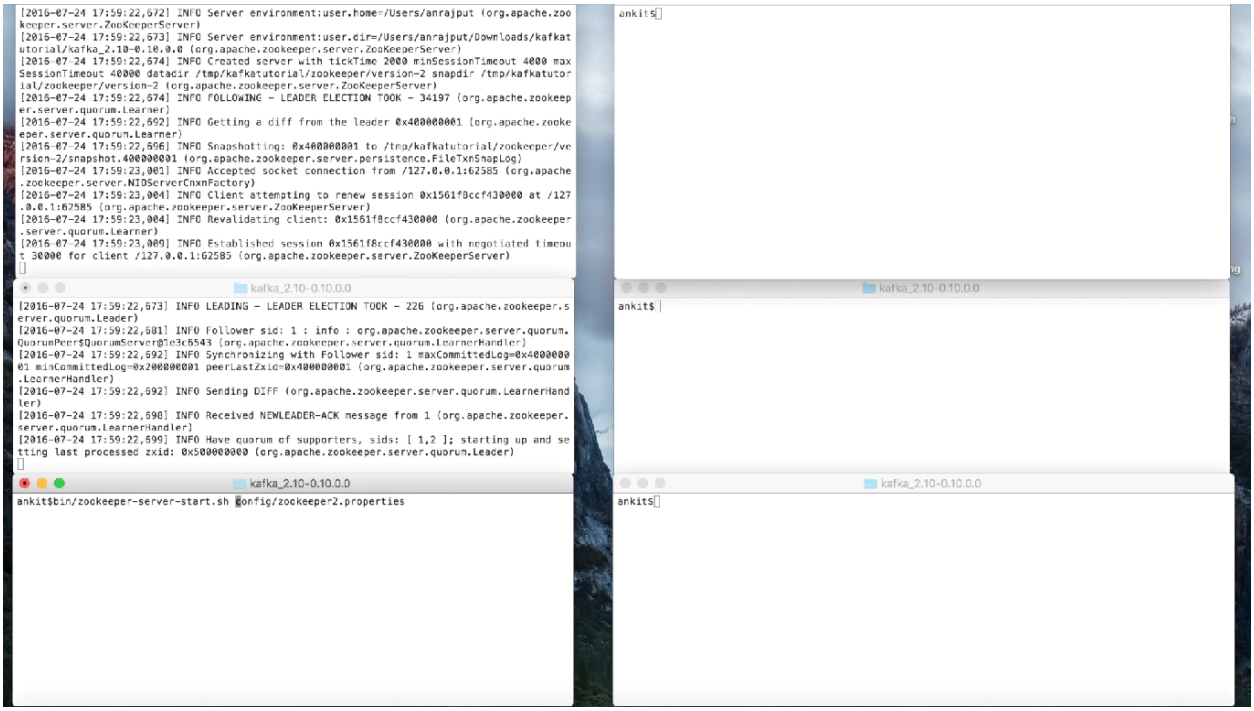
After copy edit the id changes to 2 in myid file .

Do for third directory as well same steps



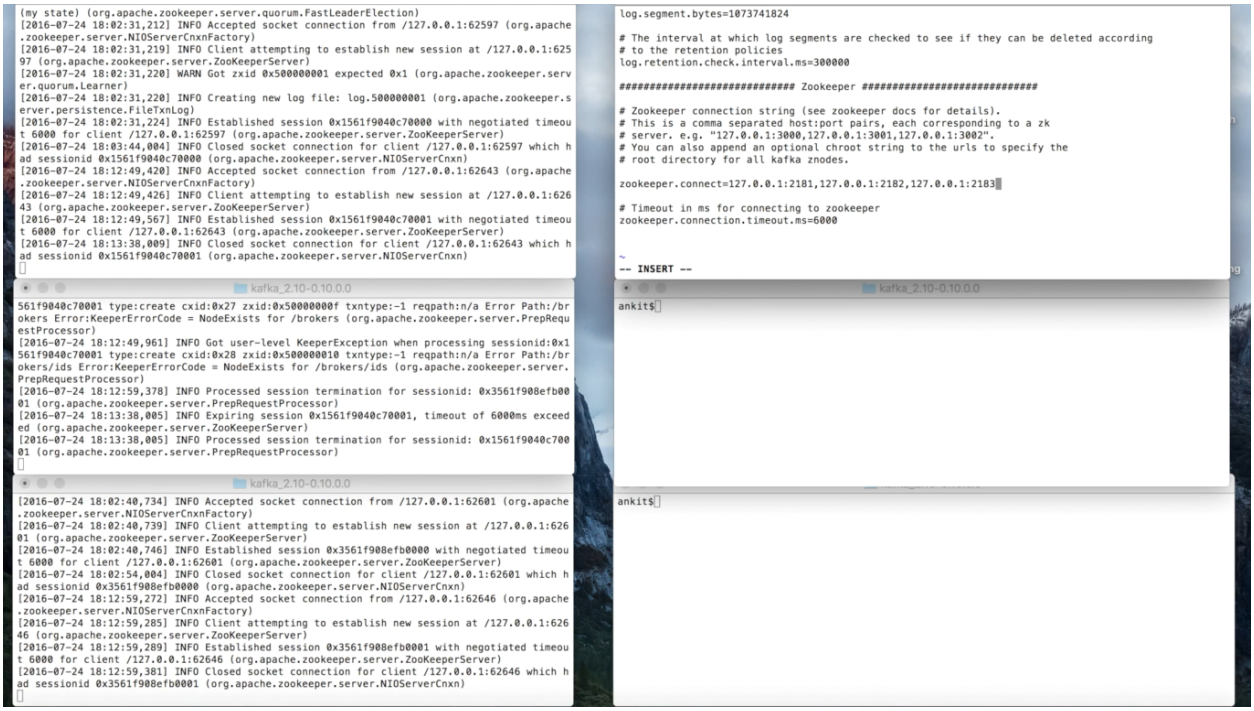
Start all 3 zookeeper one by one

When u first start , in first one u might see some error because it tries to find its other mates , but as soon as iu start others those erros u cant see again



Now u can see errros gone , because now quorum maintained .

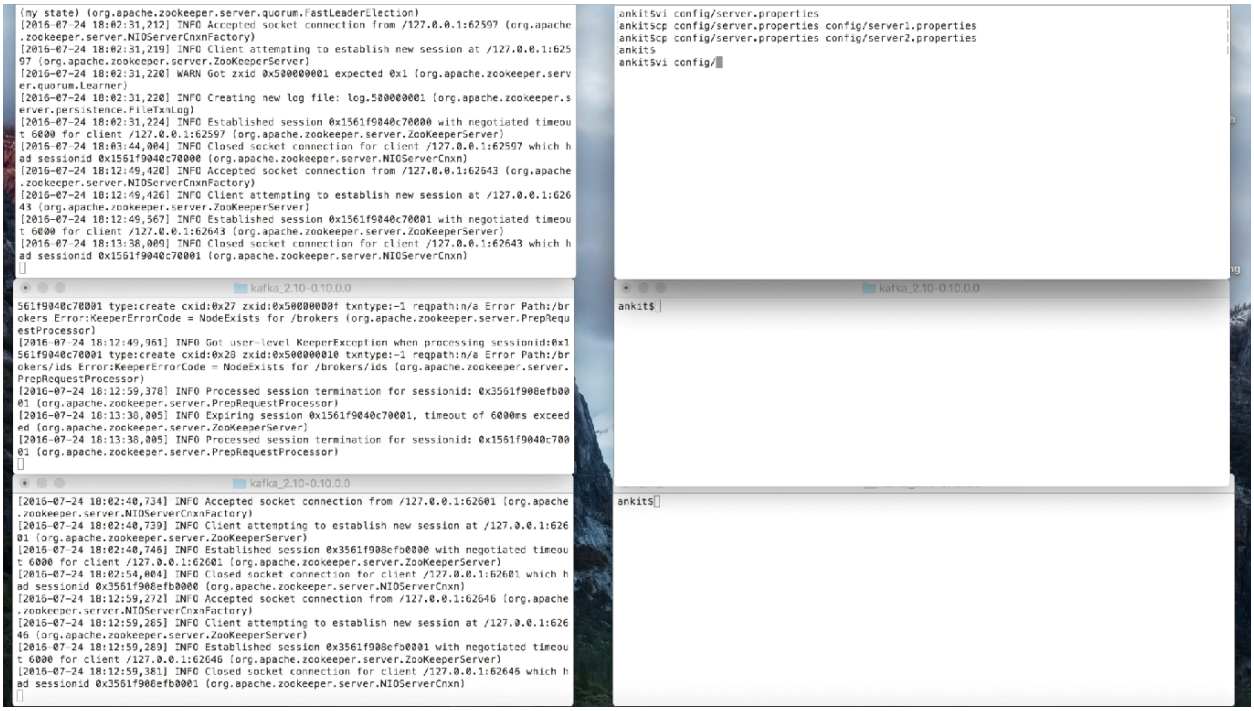
Now go for broker settings :

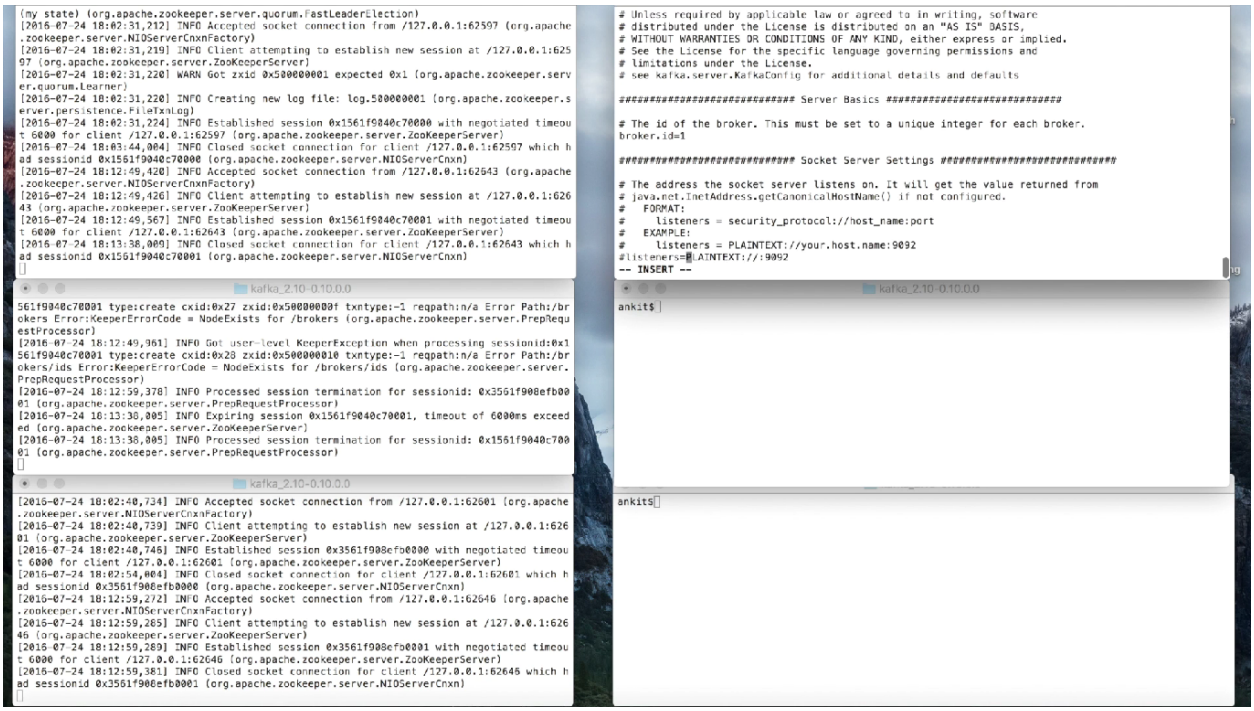


In server.properties file

Add all zookeeper server details under Zookeerer.connect = ;;;

Then copy this server.properties to server1.properties and server2.properties


(my state) (org.apache.zookeeper.server.quorum.FastLeaderElection)
[2016-07-24 18:02:31,212] INFO Accepted socket connection from /127.0.0.1:62597 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2016-07-24 18:02:31,219] INFO Client attempting to establish new session at /127.0.0.1:62597 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:02:31,220] WARN Got zxid 0x500000001 expected 0x1 (org.apache.zookeeper.server.quorum.Learner)
[2016-07-24 18:02:31,220] INFO Creating new log file: log.500000001 (org.apache.zookeeper.server.persistence.FileTxnLog)
[2016-07-24 18:02:31,224] INFO Established session 0x1561f9040c70000 with negotiated timeout 6000 for client /127.0.0.1:62597 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:03:44,004] INFO Closed socket connection for client /127.0.0.1:62597 which had sessionid 0x1561f9040c70000 (org.apache.zookeeper.server.NIOServerCnxn)
[2016-07-24 18:12:49,420] INFO Accepted socket connection from /127.0.0.1:62643 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2016-07-24 18:12:49,426] INFO Client attempting to establish new session at /127.0.0.1:62643 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:12:49,567] INFO Established session 0x1561f9040c70001 with negotiated timeout 6000 for client /127.0.0.1:62643 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:13:38,009] INFO Closed socket connection for client /127.0.0.1:62643 which had sessionid 0x1561f9040c70001 (org.apache.zookeeper.server.NIOServerCnxn)
ankit$vi config/server.properties
ankit$scp config/server.properties config/server1.properties
ankit$scp config/server.properties config/server2.properties
ankit$
ankit$vi config/
kafka_2.10-0.10.0.0
561f9040c70001 type:create cxid:0x27 zxid:0x50000000f txntype:-1 reqpath:n/a Error Path:/brokers Error:KeeperErrorCode = NodeExists for /brokers (org.apache.zookeeper.server.PrepRequestProcessor)
[2016-07-24 18:12:49,961] INFO Got user-level KeeperException when processing sessionid:0x1561f9040c70001 type:create cxid:0x28 zxid:0x500000010 txntype:-1 reqpath:n/a Error Path:/brokers/ids Error:KeeperErrorCode = NodeExists for /brokers/ids (org.apache.zookeeper.server.PrepRequestProcessor)
[2016-07-24 18:12:59,378] INFO Processed session termination for sessionid: 0x3561f908efb0001 (org.apache.zookeeper.server.PrepRequestProcessor)
[2016-07-24 18:13:38,005] INFO Expiring session 0x1561f9040c70001, timeout of 6000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:13:38,005] INFO Processed session termination for sessionid: 0x1561f9040c70001 (org.apache.zookeeper.server.PrepRequestProcessor)
kafka_2.10-0.10.0.0
ankit$
kafka_2.10-0.10.0.0
[2016-07-24 18:02:40,734] INFO Accepted socket connection from /127.0.0.1:62601 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2016-07-24 18:02:40,739] INFO Client attempting to establish new session at /127.0.0.1:62601 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:02:40,746] INFO Established session 0x3561f908efb0000 with negotiated timeout 6000 for client /127.0.0.1:62601 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:02:54,004] INFO Closed socket connection for client /127.0.0.1:62601 which had sessionid 0x3561f908efb0000 (org.apache.zookeeper.server.NIOServerCnxn)
[2016-07-24 18:12:59,272] INFO Accepted socket connection from /127.0.0.1:62646 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2016-07-24 18:12:59,285] INFO Client attempting to establish new session at /127.0.0.1:62646 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:12:59,289] INFO Established session 0x3561f908efb0001 with negotiated timeout 6000 for client /127.0.0.1:62646 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:12:59,381] INFO Closed socket connection for client /127.0.0.1:62646 which had sessionid 0x3561f908efb0001 (org.apache.zookeeper.server.NIOServerCnxn)
ankit$


(my state) (org.apache.zookeeper.server.quorum.FastLeaderElection)
[2016-07-24 18:02:31,212] INFO Accepted socket connection from /127.0.0.1:62597 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2016-07-24 18:02:31,219] INFO Client attempting to establish new session at /127.0.0.1:62597 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:02:31,220] WARN Got zxid 0x500000001 expected 0x1 (org.apache.zookeeper.server.quorum.Learner)
[2016-07-24 18:02:31,220] INFO Creating new log file: log.500000001 (org.apache.zookeeper.server.persistence.FileTxnLog)
[2016-07-24 18:02:31,224] INFO Established session 0x1561f9040c70000 with negotiated timeout 6000 for client /127.0.0.1:62597 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:03:44,004] INFO Closed socket connection for client /127.0.0.1:62597 which had sessionid 0x1561f9040c70000 (org.apache.zookeeper.server.NIOServerCnxn)
[2016-07-24 18:12:49,420] INFO Accepted socket connection from /127.0.0.1:62643 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2016-07-24 18:12:49,426] INFO Client attempting to establish new session at /127.0.0.1:62643 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:12:49,567] INFO Established session 0x1561f9040c70001 with negotiated timeout 6000 for client /127.0.0.1:62643 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:13:38,009] INFO Closed socket connection for client /127.0.0.1:62643 which had sessionid 0x1561f9040c70001 (org.apache.zookeeper.server.NIOServerCnxn)
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.
see kafka.server.KafkaConfig for additional details and defaults
############################# Server Basics #############################
# 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. It will get the value returned from
java.net.InetAddress.getCanonicalHostName() if not configured.
FORMAT:
listeners = security_protocol://host_name:port
EXAMPLE:
listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092
-- INSERT --
kafka_2.10-0.10.0.0
ankit$
kafka_2.10-0.10.0.0
561f9040c70001 type:create cxid:0x27 zxid:0x50000000f txntype:-1 reqpath:n/a Error Path:/brokers Error:KeeperErrorCode = NodeExists for /brokers (org.apache.zookeeper.server.PrepRequestProcessor)
[2016-07-24 18:12:49,961] INFO Got user-level KeeperException when processing sessionid:0x1561f9040c70001 type:create cxid:0x28 zxid:0x500000010 txntype:-1 reqpath:n/a Error Path:/brokers/ids Error:KeeperErrorCode = NodeExists for /brokers/ids (org.apache.zookeeper.server.PrepRequestProcessor)
[2016-07-24 18:12:59,378] INFO Processed session termination for sessionid: 0x3561f908efb0001 (org.apache.zookeeper.server.PrepRequestProcessor)
[2016-07-24 18:13:38,005] INFO Expiring session 0x1561f9040c70001, timeout of 6000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:13:38,005] INFO Processed session termination for sessionid: 0x1561f9040c70001 (org.apache.zookeeper.server.PrepRequestProcessor)
kafka_2.10-0.10.0.0
[2016-07-24 18:02:40,734] INFO Accepted socket connection from /127.0.0.1:62601 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2016-07-24 18:02:40,739] INFO Client attempting to establish new session at /127.0.0.1:62601 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:02:40,746] INFO Established session 0x3561f908efb0000 with negotiated timeout 6000 for client /127.0.0.1:62601 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:02:54,004] INFO Closed socket connection for client /127.0.0.1:62601 which had sessionid 0x3561f908efb0000 (org.apache.zookeeper.server.NIOServerCnxn)
[2016-07-24 18:12:59,272] INFO Accepted socket connection from /127.0.0.1:62646 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2016-07-24 18:12:59,285] INFO Client attempting to establish new session at /127.0.0.1:62646 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:12:59,289] INFO Established session 0x3561f908efb0001 with negotiated timeout 6000 for client /127.0.0.1:62646 (org.apache.zookeeper.server.ZooKeeperServer)
[2016-07-24 18:12:59,381] INFO Closed socket connection for client /127.0.0.1:62646 which had sessionid 0x3561f908efb0001 (org.apache.zookeeper.server.NIOServerCnxn)
ankit$

Change port to 9093

Change broker id to 1

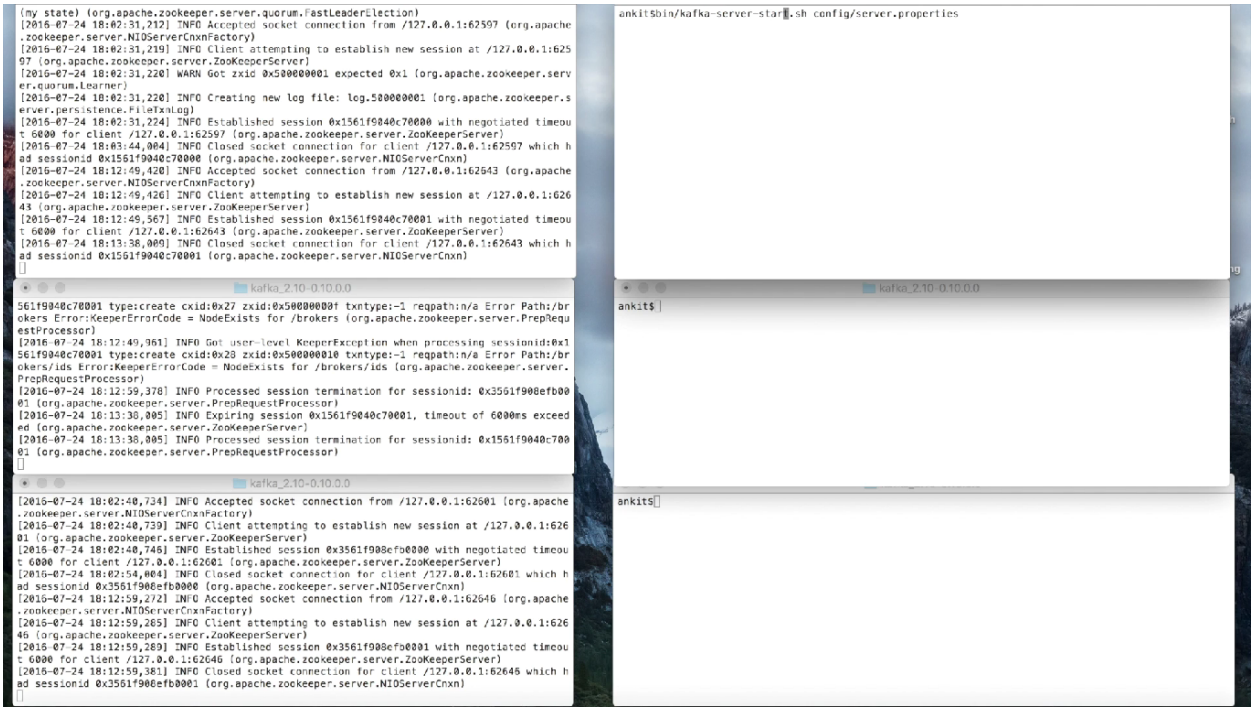
Change log directory

- Same do for server3.properties

Change port to 9094

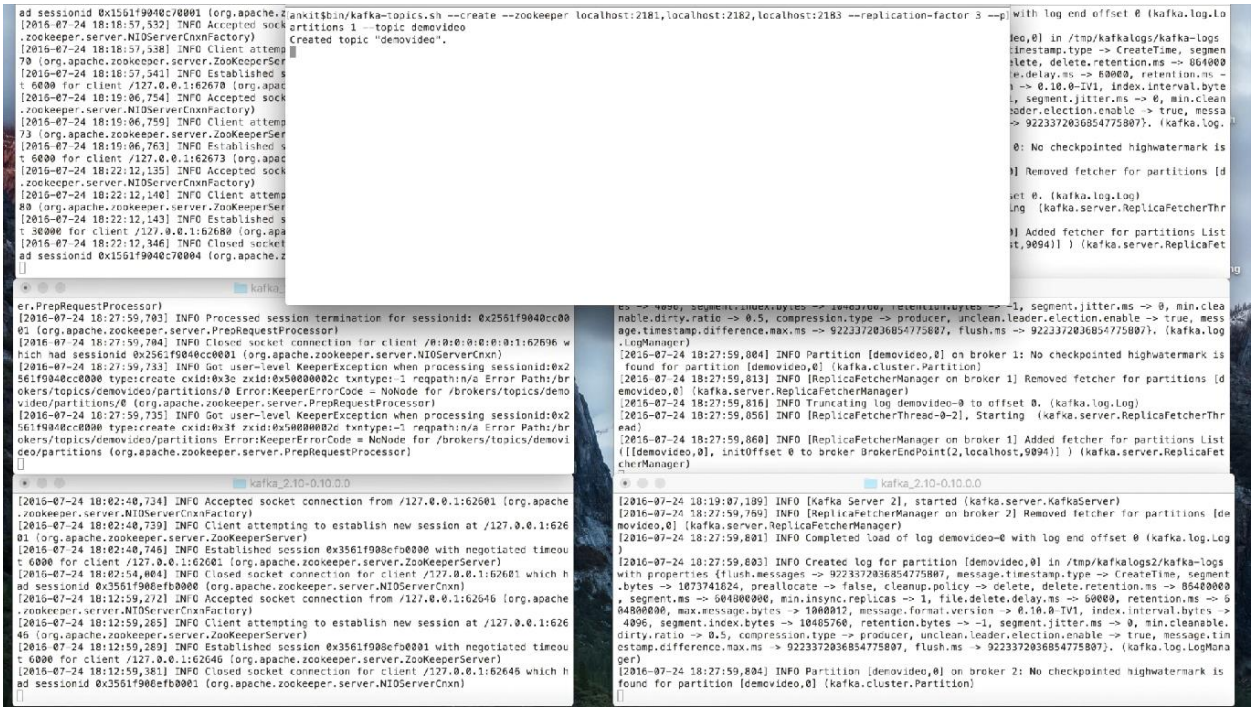
Change broker id to 2

Change log directory



Now start all 3 server (broker server)

All 3 broker and 3 zookeeper server up and running now

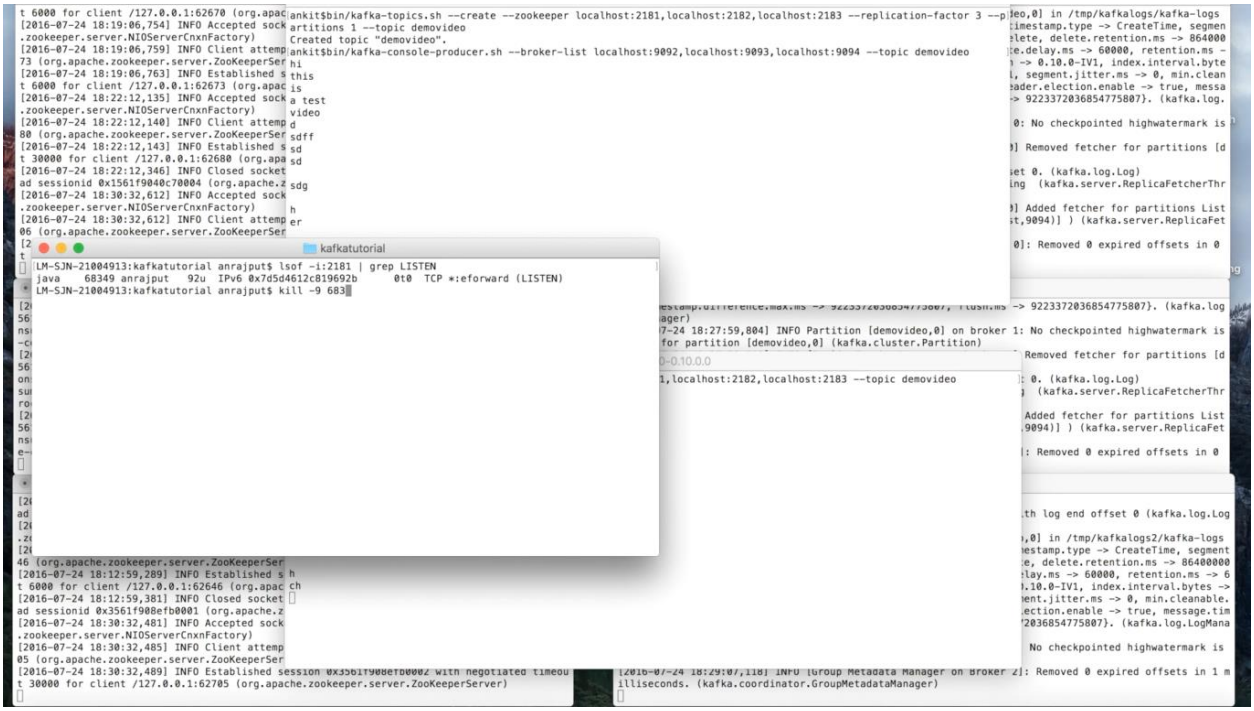


Now create the topic

With all zookeeper server and RF =3

After creating the topic start the producer

Now start the consumer



Now u can kill one zookeeper and then see u can still send the messages and deliver to the consumer

(Because at this time other two zookeeper jumped in and performed the quorum and support it 3 broker server)

==========END==========