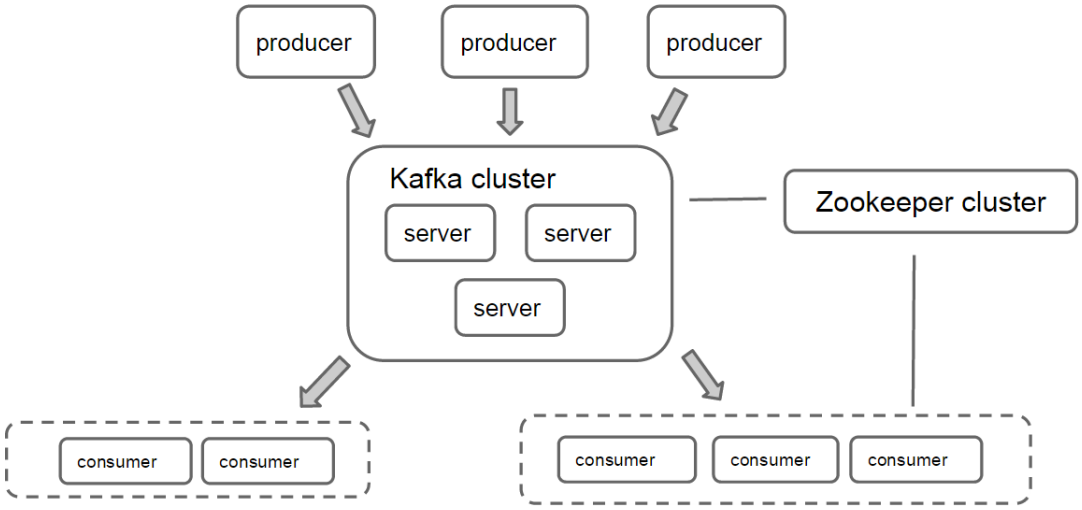
Documentation of Apache Kafka

-Prepared by Vignesh.R (15CSE107)

Abstract:

Apache Kafka is an open-source stream processing platform developed by the Apache Software Foundation written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Its storage layer is essentially a "massively scalable pub/sub message queue architected as a distributed transaction log", making it highly valuable for enterprise infrastructures to process streaming data. Additionally, Kafka connects to external systems (for data import/export) via Kafka Connect and provides Kafka Streams, a Java stream processing library. The design is heavily influenced by transaction logs.



Following are a few benefits of Kafka −

- **Reliability** − Kafka is distributed, partitioned, replicated and fault tolerance.
- **Scalability** − Kafka messaging system scales easily without down time..
- **Durability** − Kafka uses Distributed commit log which means messages persists on disk as fast as possible, hence it is durable..
- **Performance** − Kafka has high throughput for both publishing and subscribing messages. It maintains stable performance even many TB of messages are stored.

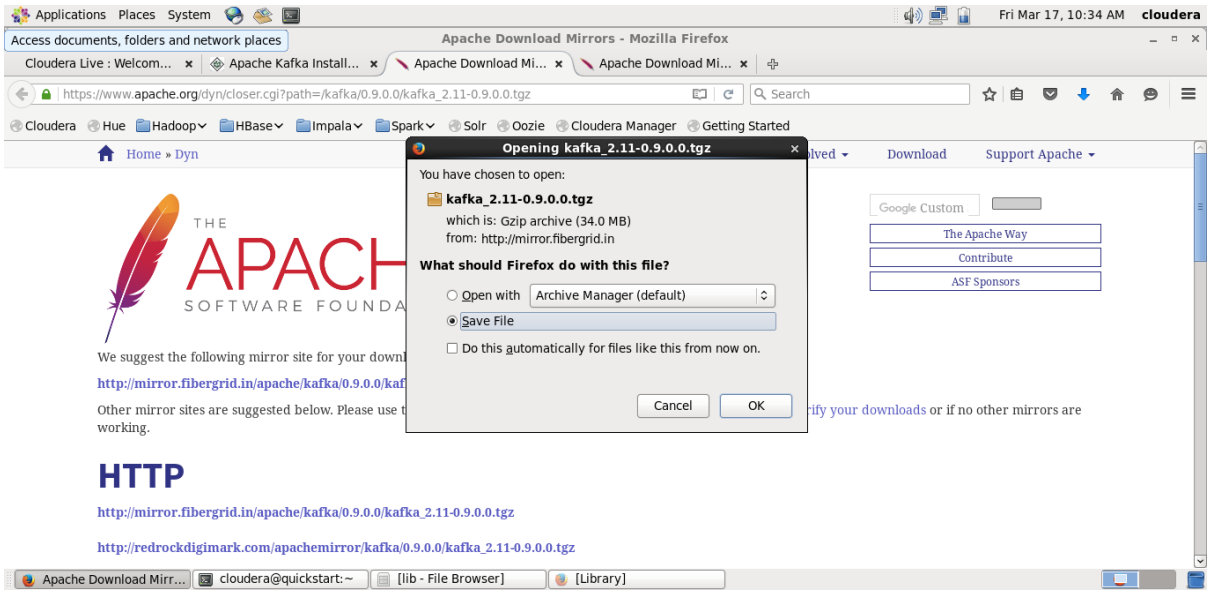
Streaming data as Single Node-Single Broker using Kafka:-

Terminal 1:

Step 1:

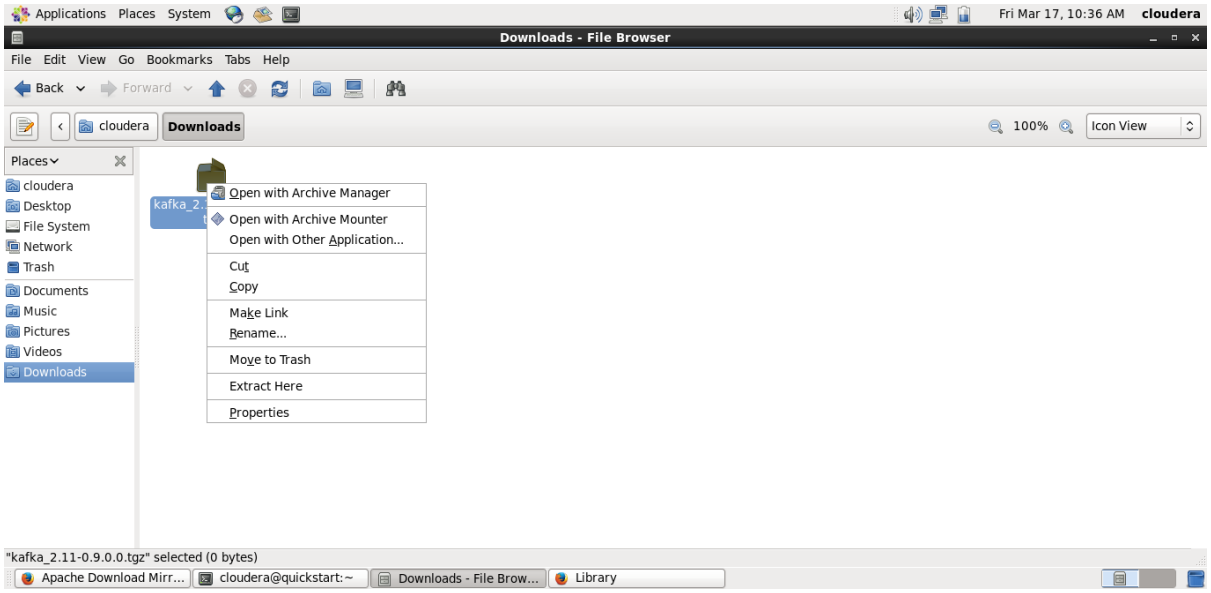
Open the browser and download Kakfa from the following link.

URL: https://www.apache.org/dyn/closer.cgi?path=/kafka/0.9.0.0/kafka_2.11-0.9.0.0.tgz



Step 2 :

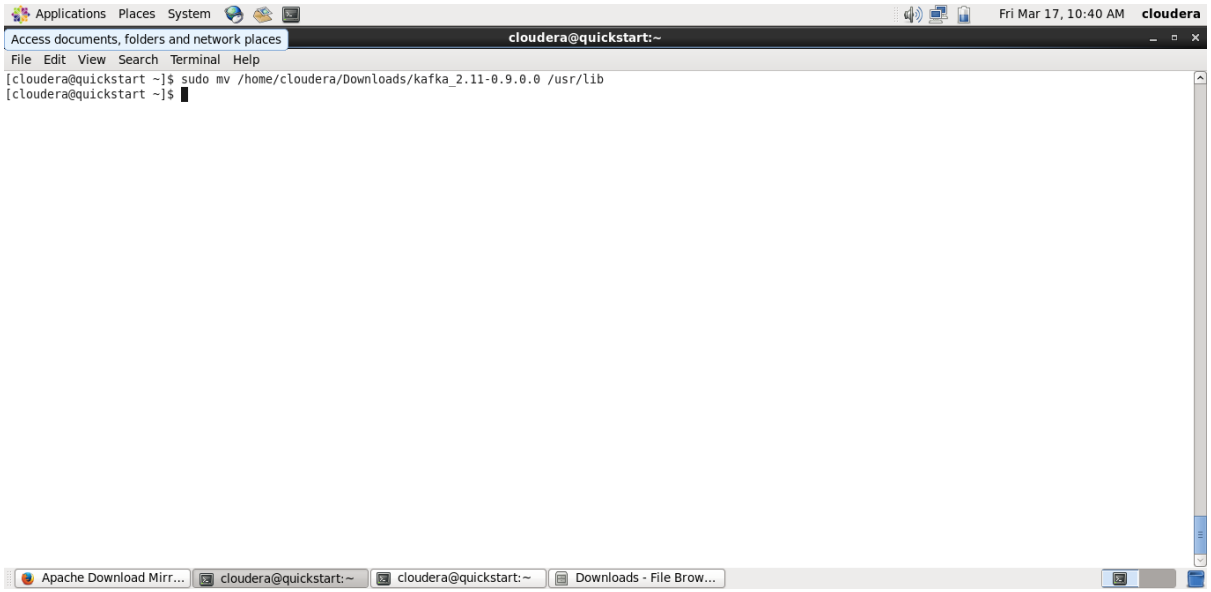
The downloaded Kafka file will be in the .tgz format in /home/cloudera/Downloads folder. Right click and click on ‘Extract here’..



Step 3:

Move the extracted file to /usr/lib directory. Super user permission is give for this modification.

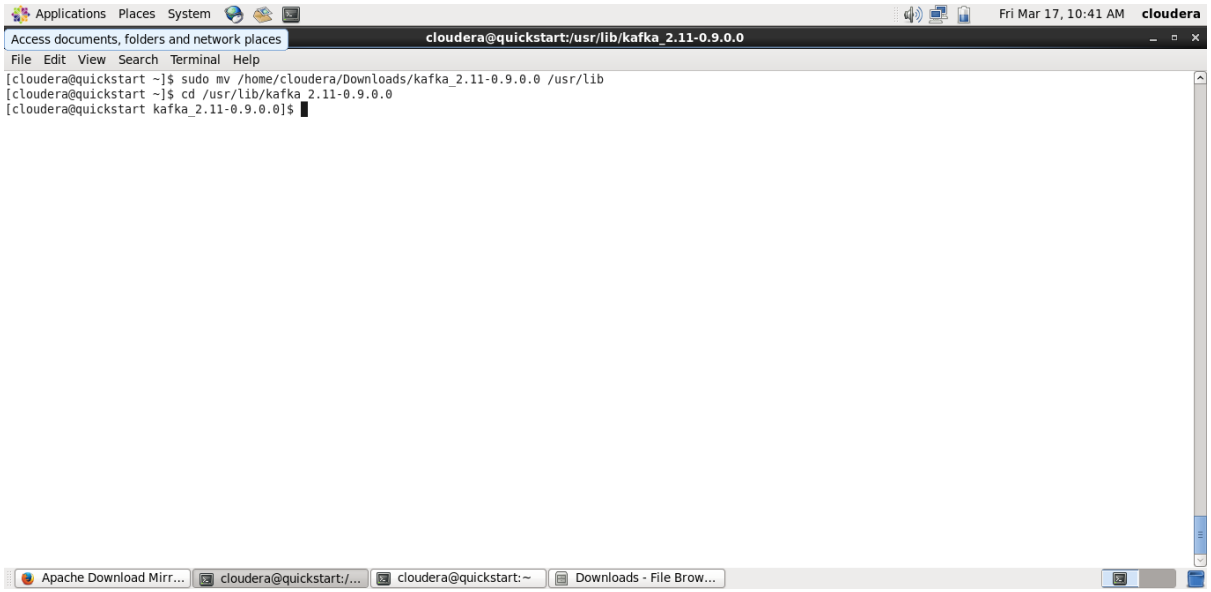
Command: sudo mv /home/cloudera/Downloads/kafka_2.11-0.9.0.0 /usr/lib



Step 4:

Change the present working directory to Kafka directory.

Command: cd /usr/lib/kafka_2.11-0.9.0.0



Step 5:

Start the Kafka server using the command given below.

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

Terminal 2:

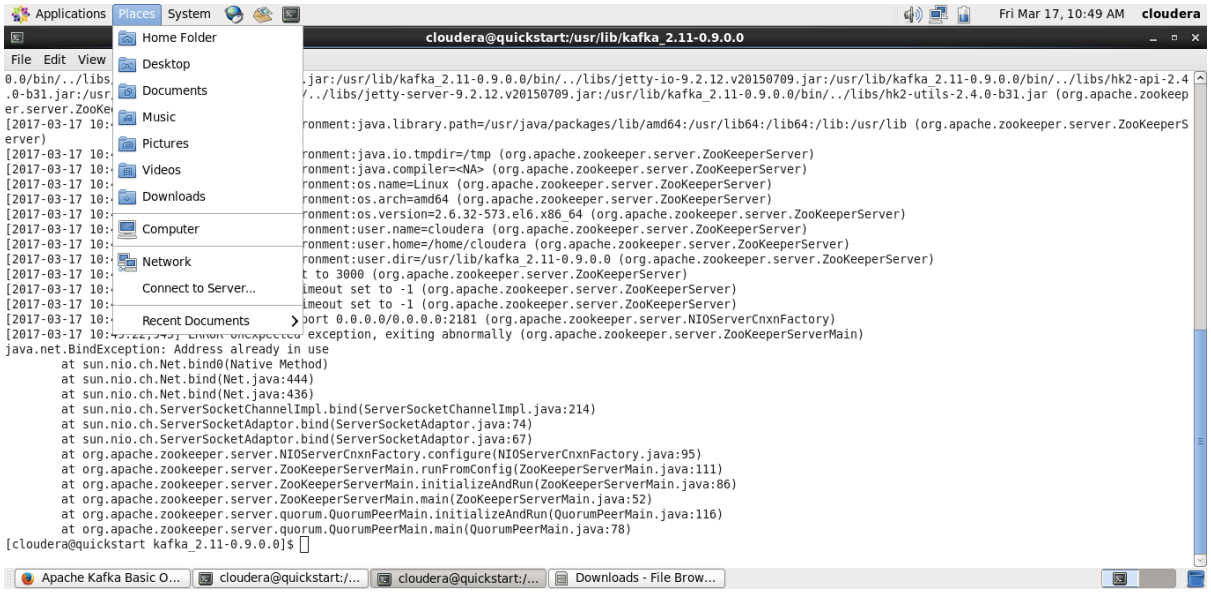
Step 6:

Repeat step 4 in the new terminal.

Step 7:

Start the zookeeper using the following command.

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



Step 8:

Start the Kafka broker in the same terminal.

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

Step 9:

Type the following command and you could see two daemons running on the terminal where QuorumPeerMain is ZooKeeper daemon and another one is Kafka daemon.

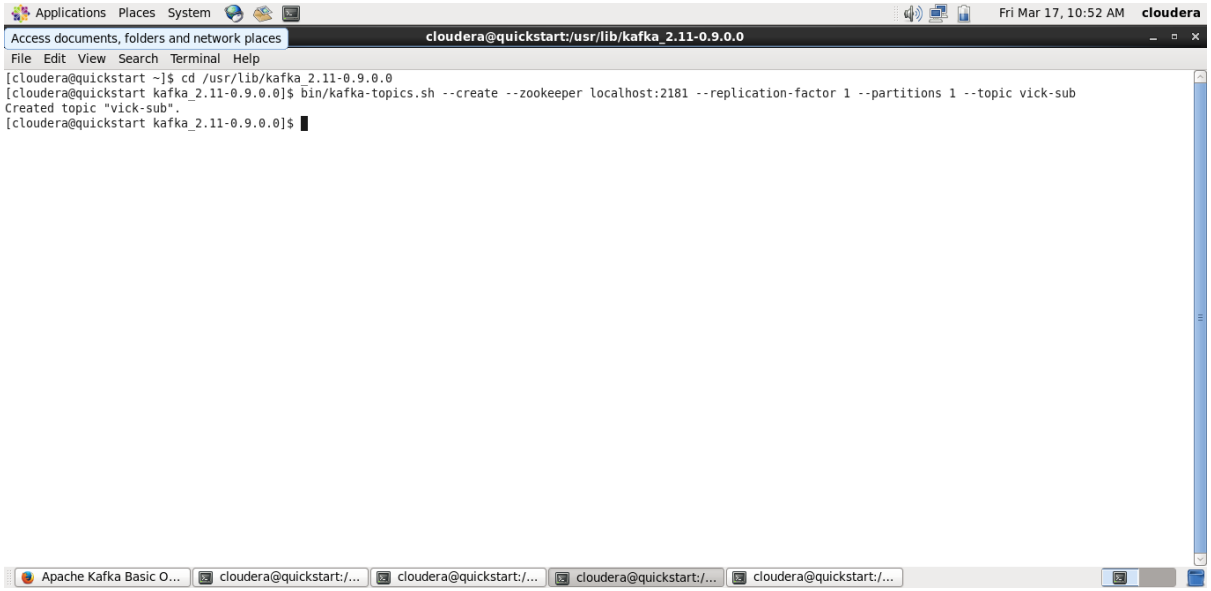
Command: jps

Terminal 3:

Step 10:

Now change the present working as mentioned in step 4 and create a new Topic as given in the following command.

Command: bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic <topic-name>



Instead of <topic-name> command, substitute the command with your topic name.

Step 11:

List the available topics using the following command.

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

Step 12:

Repeat step 4 and start the producer using the command given below.

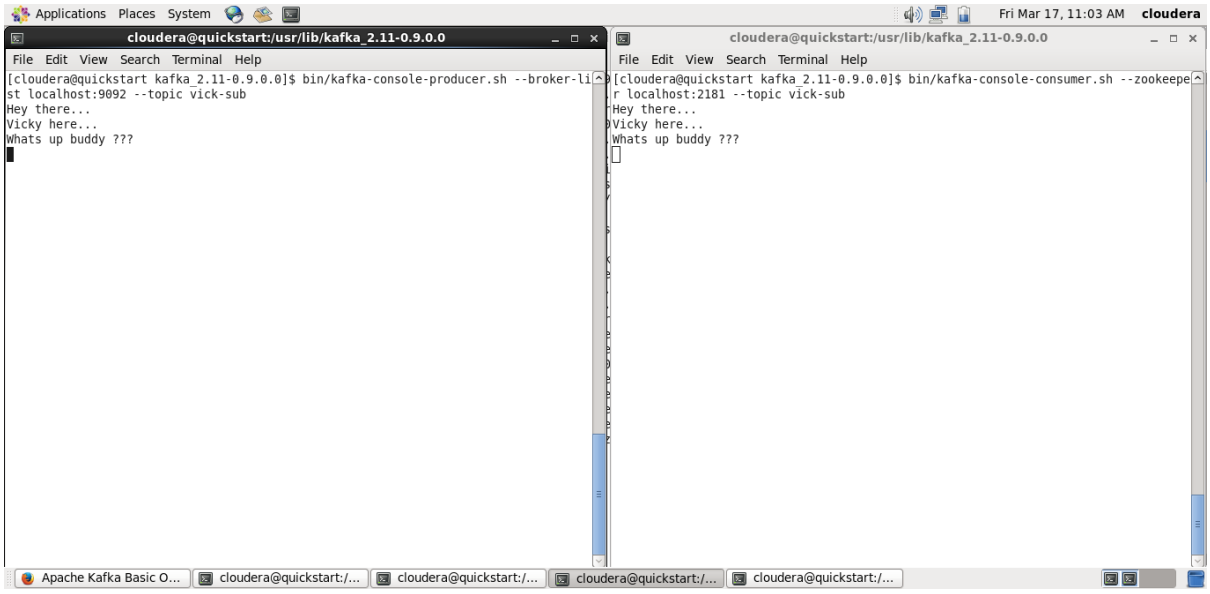
Command: bin/kafka-console-producer.sh --broker-list localhost:9092 --topic <topic-name>

Terminal 3:

Step 13:

Open a new terminal, repeat step 4 and start the producer using the command given below.

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



Now any data that is entered in the producer is received in the consumer terminal.

Step 14:

After the process is over, the kafka server that is running in terminal 1 is stopped.

Command: bin/kafka-server-stop.sh config/server.properties

Conclusion:

Thus the Single Node- Single Server configuration is depicted using Kafka.