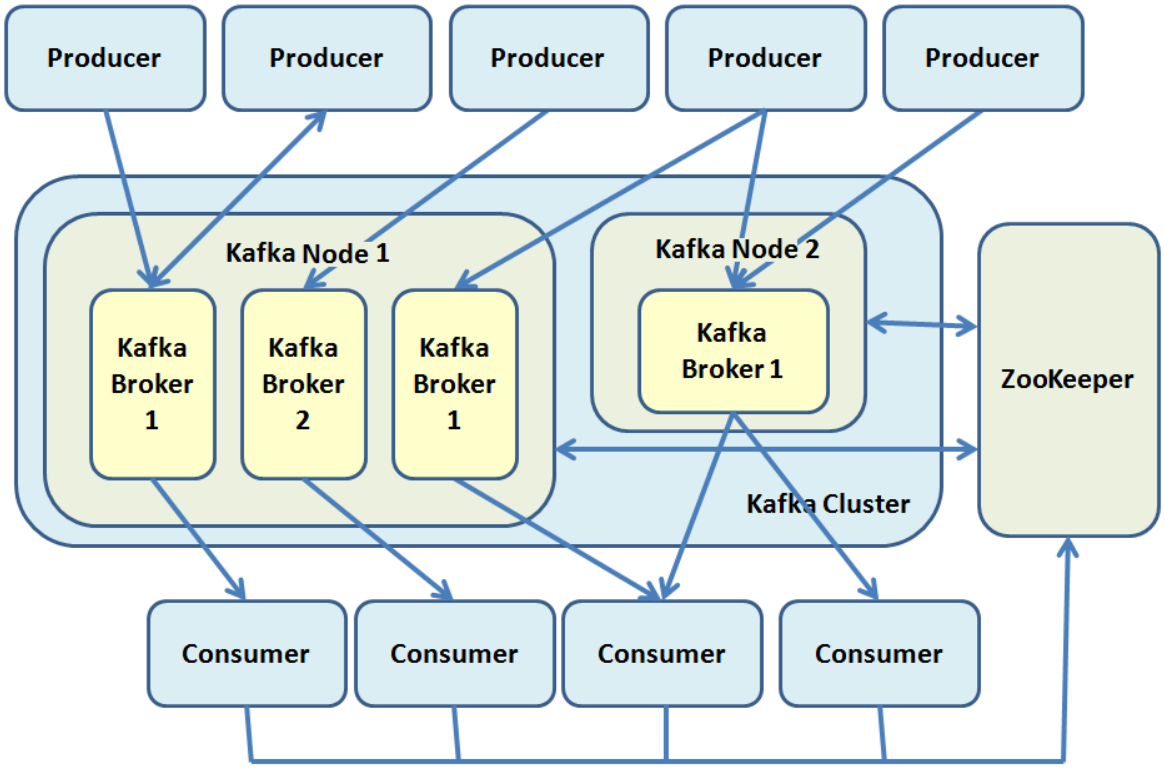
Multi Broker System



To configure a multi broker system in local machine

- Configuration in sever.properties has to be changed
  broker.id
  listeners
  log.dir
- Start the server with different ports mentioned in server.properties file
- While creating topic ,each server address can be mentioned separately with bootstrap-server

Replication factor

The replication factor is the number of nodes to which your data is replicated. When a producer writes data to Kafka, it sends it to the broker designated as the 'Leader' for that topic:partition in the cluster. Such a broker is the entry point to the cluster for the topic's data:partition.

Creating java based clients for kafka

```
package topic;

import java.util.Collections;
import java.util.Properties;
import java.util.concurrent.ExecutionException;

import org.apache.kafka.clients.admin.Admin;
import org.apache.kafka.clients.admin.AdminClientConfig;
import org.apache.kafka.clients.admin.CreateTopicsResult;
import org.apache.kafka.clients.admin.NewTopic;
import org.apache.kafka.common.KafkaFuture;

public class KafkaTopicCreator {

        KafkaTopicCreator()
        {

        Properties prop=new Properties();

        prop.put(AdminClientConfig.BOOTSTRAP_SERVERS_CONFIG,"localhost:9092,localhost:9093"
);
  Admin admin=Admin.create(prop);
  String topicname="mytopic_mutilbroker";
  NewTopic topic=new NewTopic(topicname, 1, (short)1);
  CreateTopicsResult result=admin.createTopics(Collections.singleton(topic));


  KafkaFuture<Void> future= result.values().get(topicname);
```

```
 try {
		System.out.println(future.get());
} catch (InterruptedException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
} catch (ExecutionException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
}


		}
}
```

```
//kafka producer

package producer;

import java.util.Properties;
import java.util.Scanner;

import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

public class KafkaProducerClient {

		KafkaProducerClient()
		{
			Properties p=new Properties();
		p.put("bootstrap.servers", "localhost:9092");
```

```
        p.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        p.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        KafkaProducer kproducer=new KafkaProducer(p);
        while(true)
        {


        try {

    Scanner sc=new Scanner(System.in);
    String key=sc.nextLine();
    String value=sc.nextLine();
    ProducerRecord record=new ProducerRecord("messagetopic",key,value);

        kproducer.send(record);



        }
        catch(Exception e)
        {
                System.out.println(e);
        }

        }
        //kproducer.close();
        }
}
```

//kafka consumer

```
package kafkaconsumer;
import java.util.Arrays;
import java.util.Properties;

import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
public class KafkaConsumerClent {
	public static void main(String[] args)
	{

		String topic="myjavatopic1";
		String group="group2";
		Properties properties=new Properties();
		properties.put("bootstrap.servers",
"localhost:9092,localhost:9093");
		properties.put("group.id",group);
		properties.put("enable.auto.commit", "true");
		properties.put("auto.commit.interval.ms","1000" );
		properties.put("session.timeout.ms", "300000");

	properties.put("key.deserializer","org.apache.kafka.common.serialization.St
ringDeserializer");

	properties.put("value.deserializer","org.apache.kafka.common.serialization.
StringDeserializer");

		KafkaConsumer <String,String> consumer=new
KafkaConsumer(properties);

		consumer.subscribe(Arrays.asList(topic));

		while(true)
		{
			ConsumerRecords<String ,String> record=consumer.poll(100);
			for(ConsumerRecord r:record)
			{
				System.out.println(r.offset());
				System.out.println(r.key());
				System.out.println(r.value());
			}

		}

	}

}
```