Go To <http://kafka.apache.org/downloads.html> link and download latest release ( I have downloaded 0.10.01.1 for Scala 2.11)



Extract it in some folder ( in my case it is /home/cloudera)

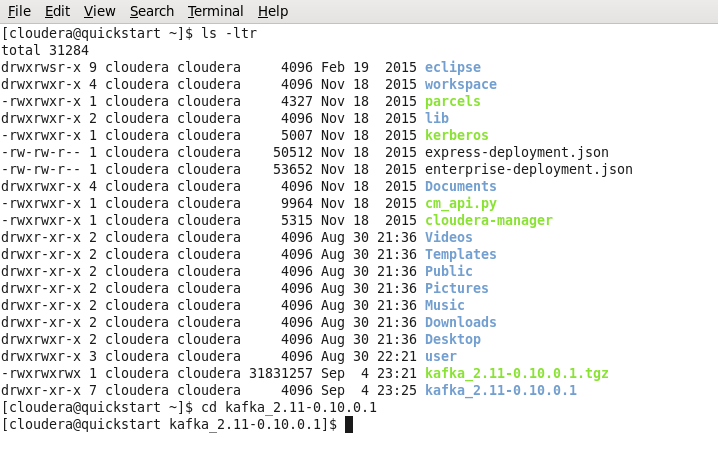
* **tar -xzf kafka\_2.11-0.10.0.1.tgz**
* **CD kafka\_2.11-0.10.0.1**

#### [Step 1: Download the code](http://kafka.apache.org/documentation.html#quickstart_download)

[**Download**](https://www.apache.org/dyn/closer.cgi?path=/kafka/0.10.0.0/kafka_2.11-0.10.0.0.tgz) the 0.10.0.1 release and un-tar it.

> **tar -xzf kafka\_2.11-0.10.0.1.tgz**

> **cd kafka\_2.11-0.10.0.1**



#### [Step 2: Start the server](http://kafka.apache.org/documentation.html#quickstart_startserver)

Kafka uses ZooKeeper so you need to first start a ZooKeeper server if you don't already have one. You can use the convenience script packaged with kafka to get a quick-and-dirty single-node ZooKeeper instance.

> **bin/zookeeper-server-start.sh config/zookeeper.properties**

[2013-04-22 15:01:37,495] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

...



**Note : if you found port is already in use , you need to change the port in zookeeper.properties file**

**Also need to change the same port in server.properties file (zookeeper.connect=localhost:2181 )also which is used by kafka server, and all zookeeper command should be change accordingly**

**In cloudera quickvm zookeeper service is already running at prot 2181 so no need to start that service**

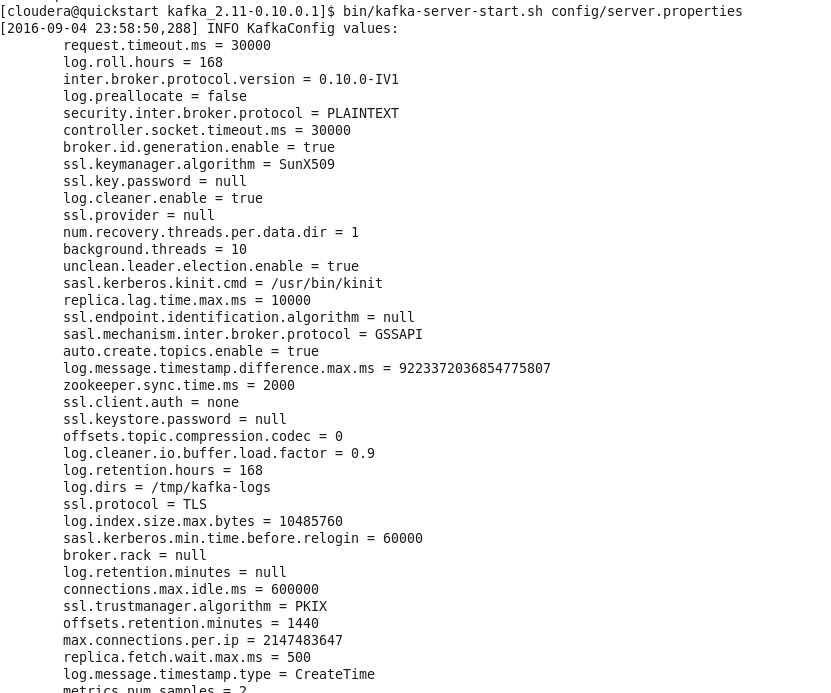
Now start the Kafka server in different terminal:

> **bin/kafka-server-start.sh config/server.properties**

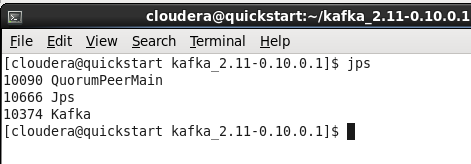
[2013-04-22 15:01:47,028] INFO Verifying properties (kafka.utils.VerifiableProperties)

[2013-04-22 15:01:47,051] INFO Property socket.send.buffer.bytes is overridden to 1048576 (kafka.utils.VerifiableProperties)

...



Check how many processes are running by using jps command



**QuroumPeerMain** is Zookeeper service

And **Kafka** is Kafka Server Service

#### [Step 3: Create a topic](http://kafka.apache.org/documentation.html#quickstart_createtopic)

Let's create a topic named "test" with a single partition and only one replica:

> **bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test**

We can now see that topic if we run the list topic command:

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

Test



Alternatively, instead of manually creating topics you can also configure your brokers to auto-create topics when a non-existent topic is published to.

Find below command for delete existing topic

Add below line in ${kafka\_home}/config/server.properties

delete.topic.enable=true

Restart the kafka server with new config:

${kafka\_home}/bin/kafka-server-start.sh ~/kafka/config/server.properties

Delete the topics you wish to:

${kafka\_home}/bin/kafka-topics.sh --delete --zookeeper localhost:2181 --topic test

#### [Step 4: Send some messages](http://kafka.apache.org/documentation.html#quickstart_send)

Kafka comes with a command line client that will take input from a file or from standard input and send it out as messages to the Kafka cluster. By default each line will be sent as a separate message.

Run the producer and then type a few messages into the console to send to the server.

> **bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test**

**This is a message**

**This is another message**

#### [Step 5: Start a consumer](http://kafka.apache.org/documentation.html#quickstart_consume)

Kafka also has a command line consumer that will dump out messages to standard output.

> **bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning**

This is a message

This is another message

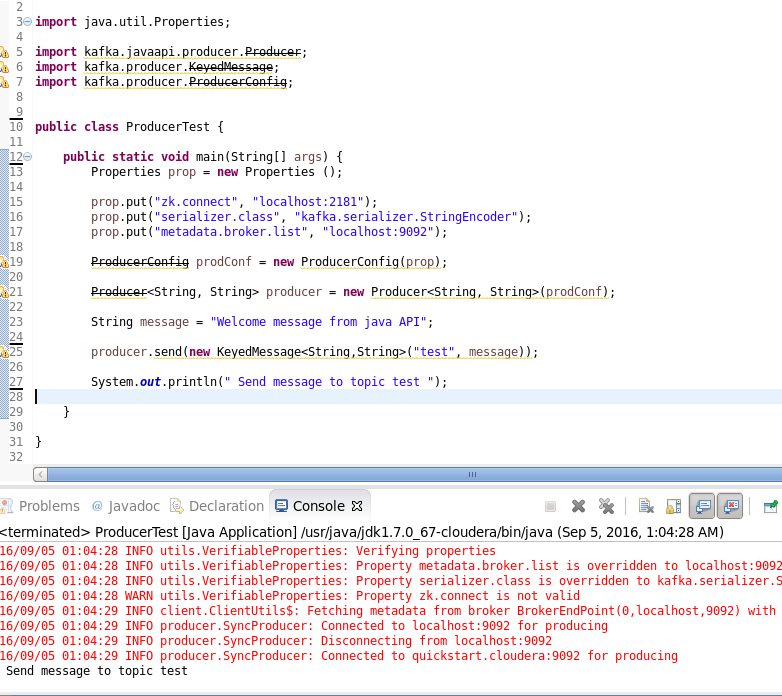
If you have each of the above commands running in a different terminal then you should now be able to type messages into the producer terminal and see them appear in the consumer terminal.

All of the command line tools have additional options; running the command with no arguments will display usage information documenting them in more detail.

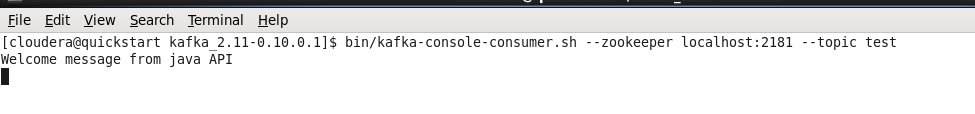
**Produce message from java API**

First Create a java project in Eclipse and add all jars from <kafka home>/lib folder and run below program

|  |
| --- |
| import java.util.Properties;  import kafka.javaapi.producer.Producer;  import kafka.producer.KeyedMessage;  import kafka.producer.ProducerConfig;  public class ProducerTest {  public static void main(String[] args) {  Properties prop = new Properties ();    prop.put("zk.connect", "localhost:2181");  prop.put("serializer.class", "kafka.serializer.StringEncoder");  prop.put("metadata.broker.list", "localhost:9092");    ProducerConfig prodConf = new ProducerConfig(prop);    Producer<String, String> producer = new Producer<String, String>(prodConf);    String message = "Welcome message from java API";    // here test is the name of topic  producer.send(new KeyedMessage<String,String>("test", message));    System.out.println(" Send message to topic test ");  }  } |

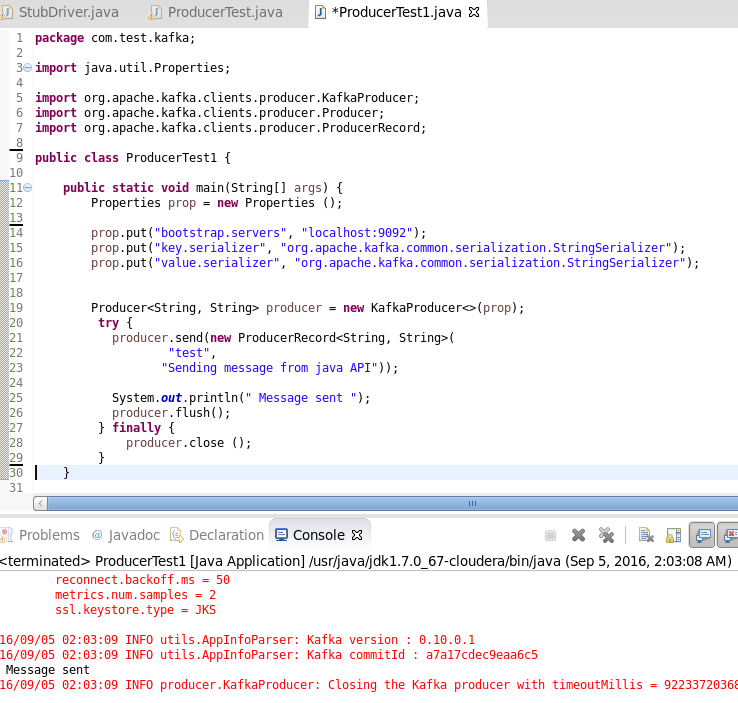


You can check on CLI command that consumer received message from java api



**Latest JAVA API**

|  |
| --- |
| **package com.test.kafka;**  **import java.util.Properties;**  **import org.apache.kafka.clients.producer.KafkaProducer;**  **import org.apache.kafka.clients.producer.Producer;**  **import org.apache.kafka.clients.producer.ProducerRecord;**  **public class ProducerTest1 {**  **public static void main(String[] args) {**  **Properties prop = new Properties ();**    **prop.put("bootstrap.servers", "localhost:9092");**  **prop.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");**  **prop.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");**      **Producer<String, String> producer = new KafkaProducer<>(prop);**  **try {**  **producer.send(new ProducerRecord<String, String>(**  **"test",**  **"Sending message from java API"));**    **System.out.println(" Message sent ");**  **producer.flush();**  **} finally {**  **producer.close ();**  **}**  **}**  **}** |



**To Kill Process**

