Each kafka server is a Broker and each Broker is given a number.  
Each Broker may store multiple Topics with multiple partitions of the topic. All or few based on the number of Brokers available.  
Broker: 0 means Broker number 0

**Setup**Create empty folder inside kafka directory  
C:/kafka\_2.13-2.5.0/data/zookeeper  
C:/kafka\_2.13-2.5.0/data/kafka

**Starting zookeeper**  
Edit config\zookeeper.properties. All forward slashes  
set dataDir=C:/kafka\_2.13-2.5.0/data/zookeeper  
From kafka folder start zookeeper  
C:\kafka\_2.13-2.5.0>  
zookeeper-server-start config\zookeeper.properties  
It will bind to port 2181

**Starting kafka**  
Edit config/server.properties. All forward slashes  
set log.dirs=C:/kafka\_2.13-2.5.0/data/kafka  
From kafka folder start kafka but from a new command prompt as the earlier one has zookeeper running.  
C:\kafka\_2.13-2.5.0>  
kafka-server-start config\server.properties  
We will now find files created under these two  
C:/kafka\_2.13-2.5.0/data/zookeeper  
C:/kafka\_2.13-2.5.0/data/kafka

**Create topic**kafka-topics --zookeeper 127.0.0.1:2181 --topic first\_topic --create --partitions 3 --replication-factor 1  
Now see first\_topic folders created under C:\kafka\_2.13-2.5.0\data\kafka  
**List topics**  
kafka-topics --zookeeper 127.0.0.1:2181 --list  
**Detailed info about a topic**  
kafka-topics --zookeeper 127.0.0.1:2181 --topic first\_topic --describe  
**Creating and deleting topic**kafka-topics --zookeeper 127.0.0.1:2181 --topic second\_topic --create --partitions 6 --replication-factor 1  
kafka-topics --zookeeper 127.0.0.1:2181 --list  
kafka-topics --zookeeper 127.0.0.1:2181 --topic second\_topic --delete  
The topic will be marked for deletion  
kafka-topics --zookeeper 127.0.0.1:2181 --list  
**Producing data**c:\kafka\_2.13-2.5.0>kafka-console-producer --broker-list 127.0.0.1:9092 --topic first\_topic

>hello Tez

>learning Kafka

>lets see

>Terminate batch job (Y/N)? y  
**Producing data with acks=all Acknowledgement after replication**c:\kafka\_2.13-2.5.0>kafka-console-producer --broker-list 127.0.0.1:9092 --topic first\_topic --producer-property acks=all  
**Producing to a topic that does not exist**kafka-console-producer --broker-list 127.0.0.1:9092 --topic new\_topic  
You get a warning and the topic will be created and a leader would be elected. If tried to produce again, we are good. It is recommended to created topic first and not produce to non-existing topic directly. The default number of partitions in this case is 1. This is of no use. So change server.properties num.partitions= 1 property to **num.partitions=3  
Consuming from topic**kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic first\_topic  
kafka-console-consumer by default will only read messages produced after it is loaded. This CLI utility would not read messaged produced earlier to its launch. So, start producing now to see consumer working.

**Consuming from topic from the beginning**

kafka-console-consumer --bootstrap-server localhost:9092 --topic first\_topic --from-beginning  
Now all the messages added earlier to consumer launch into the topic will also be consumed.  
**Multiple Consumers as a group**Run two consumers - following in TWO terminals  
And one producer  
kafka-console-consumer --bootstrap-server localhost:9092 --topic first\_topic **--group my-first-application**

We can see the messages produced being consumed by two consumers in a random fashion based on which partition of the topic the messages lands in. The consumers in the group make sure they understand who is reading from which partition.  
When using consumer group, last message read is remembered by storing offset. So when connected later, it will read the messages added later into the topic even before consumer with same consumer group is relaunched.

**Listing consumer groups**kafka-consumer-groups --bootstrap-server localhost:9092 --list  
**Describing a consumer group to see offset details**kafka-consumer-groups --bootstrap-server localhost:9092 --describe --group my-first-application  
Details about where the offset is in **each partition,** if all is read by the consumer, consumer info. etc. is detailed here.   
CURRENT-OFFSET - Offset till where the messages are read in the partition.  
LOG-END-OFFSET - Offset indicating the last messages published.  
If CURRENT-OFFSET is less than LOG-END-OFFSET, it means that the consumer-group my-first-application is yet to consume some more messages. This will be evident in LAG.  
CONSUMER-ID and HOST tell which consumer from the consumer-group and from which machine is consuming the messages from the partition.  
**Resetting offsets**consumer group can read again data from certain period again. Here are possible options about duration possible --to-datetime, --by-period**, --to-earliest**, --to-latest, **--shift-by**, --from-file, --to-current  
kafka-consumer-groups --bootstrap-server localhost:9092 --group my-first-application --reset-offsets --execute --topic first\_topic **--to-earliest**  
  
**Shift offset back and forth by number of messages positive to move forward and negative to move backward.**  
kafka-consumer-groups --bootstrap-server localhost:9092 --group my-first-application --reset-offsets --execute --topic first\_topic **--shift-by** 2  
This will lag by 2 messages now for **each partition in the topic**.  
**Download kafkatool.com to read messaged and see offsets. Just an alternate to CLI.**