Apache Kafka

Apache Kafka Installation

Kafka Came in 4 flavours and we can classifiy them as below:

1. **Open Source: Apache Kafka**

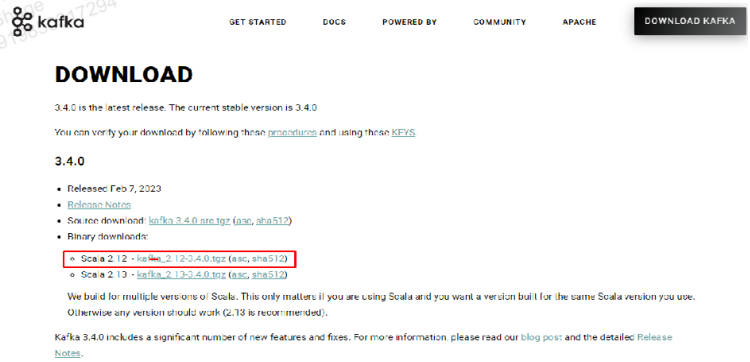
2. Community Edition: **Confluent.io**

3. Commercial Distribution: confluent.io

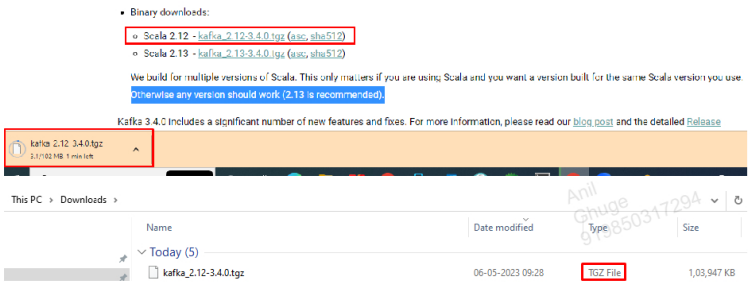
4. Managed Service: confluent, amazon, aiven.io, azure kafka etc.

Download Apache Kafka

Step-1: Download the Apache Kafka, from https://kafka.apache.org/downloads

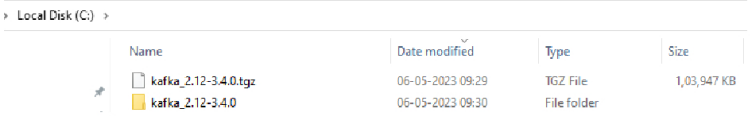


Step-2: when we click on the above link highlighted, it will start downloading the Kafka tar file as shown below:

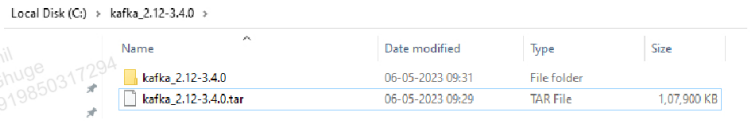


Apache Kafka

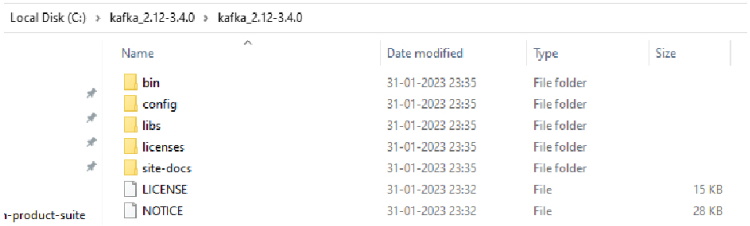
Step-3: once the download of the tar file completed as shown above, copy and extract it in C:\ drive



Step-4: once extraction of the tar is done using the 7zip software or any , we can see the extracted folder. Now go insde in it and we can see one more tar file as shown below:



Now we can see the actual extracted the kafka binaries as shown below:



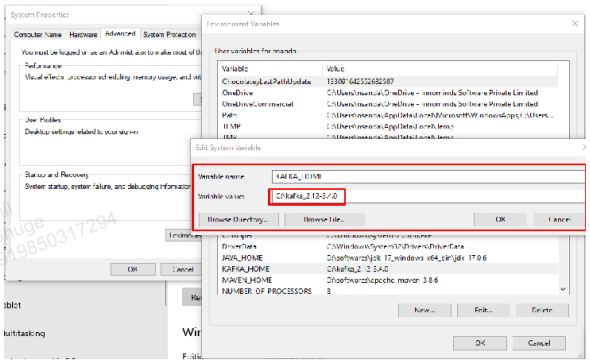
Step-5: Once the download and install set up done, we can see the version of the kafka using the below command:

```
C:\kafka_2.12-3.4.0\bin\windows>kafka-configs.bat --version
3.4.0 (Commit:2e1947d240607d53)
```

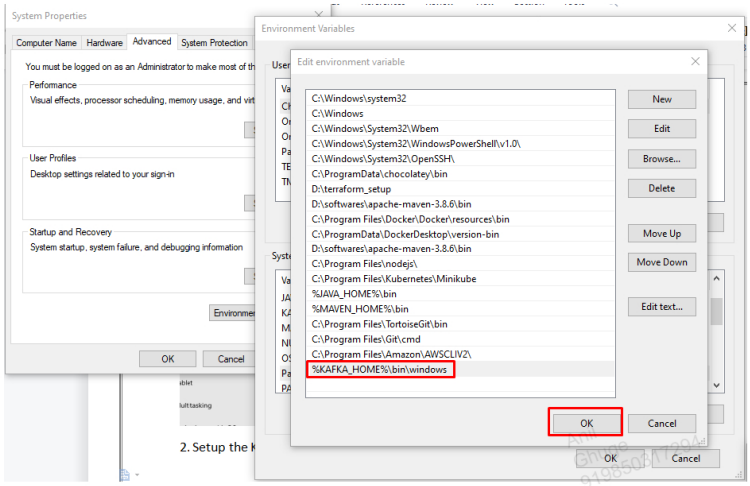

Apache Kafka

setup kafka windows directories as an environment variable

1. Setup KAFKA_HOME env variable as shown below:



2. Setup the KAFKA_HOME in the path directory as shown below:



3. After the KAKA_HOME setup done as shown above, go to any of the directory like D:\ or E:\ and the use the bat files directly as shown below usually which were there in \bin\windows directory.

Apache Kafka

```
D:\personal\my_trainigs>kafka-configs --version
3.4.0 (Commit:2e1947d240607d53)
```

Setup Single Node Kafka Cluster

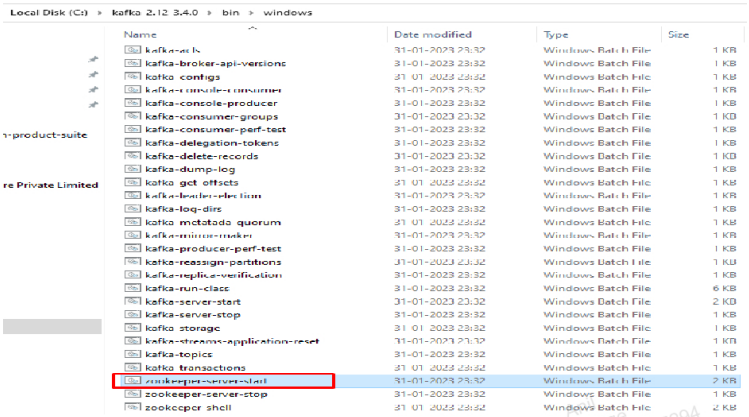
Step-1: Spin-up a zookeeper instance. Zookeeper comes with the kafka distribution that we download

Step-2: Spin up the Kafka Broker. Once the broker is up, it register it self in the zookeeper

Step-3: zookeeper monitor and manages the health of the kafka broker.

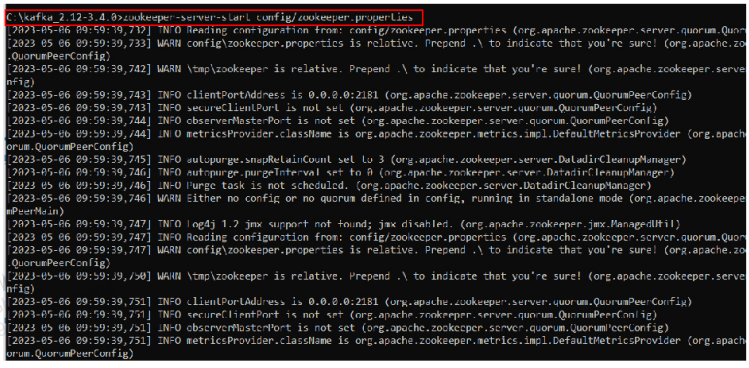
Step-1: Spin-up a zookeeper instance

go inside the bin/windows as shown below and start the zookeeper:



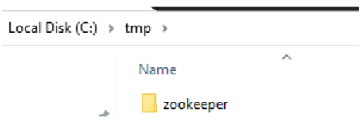
If Kafka_home setup done, then directly go to kafka_home directory and start the zookeeper as shown below:

Apache Kafka

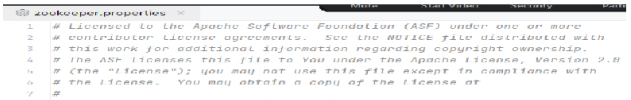



```
C:\kafka_2.12-3.4.0>zookeeper-server-start config/zookeeper.properties
```

Once the zookeeper server stated, we can see the zookeeper data directory in /tmp/zookeeper as shown below:



This directory is creating in the /tmp location because in the zookeeper.properties files it has been configured as shown below:



```
#    http://www.apache.org/licenses/LICENSE-2.0
#
# 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.
# the directory where the snapshot is stored.
dataDir=/tmp/zookeeper
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections since this is a non-
maxClientCnxns=0
# Disable the adminserver by default to avoid port conflicts.
# Set the port to something non-conflicting if choosing to enable this
admin.enableServer=false
# admin.serverPort=8080
```


Step-2: Spin up the Kafka Broker

To Setup Kafka broker, we need to run the "kafka-server-start.bat" file by supplying the server.properties file.

To do this, go to KAFKA_HOME directory, and issue the below command:

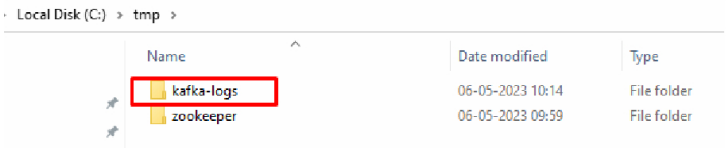
```
C:\kafka_2.12-3.4.0>.\bin\windows\kafka-server-start.bat .\config\server.properties

                    (OR)

C:\kafka_2.12-3.4.0>kafka-server-start config/server.properties
```

```
C:\kafka_2.12-3.4.0>kafka-server-start config/server.properties
[2023-05-06 10:08:52,980] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
Log4j:ERROR Failed to rename [C:\kafka_2.12-3.4.0/logs/server.log] to [C:\kafka_2.12-3.4.0/logs/server.log.2023-05-06-09].
[2023-05-06 10:08:53,245] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.z
ookeeper.common.X509Util)
[2023-05-06 10:08:53,327] INFO starting (kafka.server.KafkaServer)
[2023-05-06 10:08:53,328] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2023-05-06 10:08:53,343] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2023-05-06 10:08:53,350] INFO Client environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (o
rg.apache.zookeeper.ZooKeeper)
[2023-05-06 10:08:53,350] INFO Client environment:host.name=host.docker.internal (org.apache.zookeeper.ZooKeeper)
[2023-05-06 10:08:53,350] INFO Client environment:java.version=17.0.6 (org.apache.zookeeper.ZooKeeper)
[2023-05-06 10:08:53,351] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2023-05-06 10:08:53,351] INFO Client environment:java.home=C:\softwares\jdk-17_windows-x64_bin\jdk-17.0.6 (org.apache.zookeeper.ZooKeeper)
[2023-05-06 10:08:53,351] INFO Client environment:java.class.path=C:\kafka_2.12-3.4.0\libs\activation-1.1.1.jar;C:\kafka_2.12-3.4.0\libs\aopalliance-
repackaged-2.6.1.jar;C:\kafka_2.12-3.4.0\libs\argparse4j-0.7.0.jar;C:\kafka_2.12-3.4.0\libs\audience-annotations-0.5.0.jar;C:\kafka_2.12-3.4.0\libs\c
ommons-cli-1.4.jar;C:\kafka_2.12-3.4.0\libs\commons-lang3-3.8.1.jar;C:\kafka_2.12-3.4.0\libs\connect-api-3.4.0.jar;C:\kafka_2.12-3.4.0\libs\connect-b
```

As soon as the Kafak server started, it is called as the broker-1, which starts listening to the consumers and producers on the port 9092. also a kafka-logs directory will be created in /tmp directory as shown below:



This directory has been created, because it was mentioned in the server.properties file as shown below:

```
############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=/tmp/kafka-logs

```

Also this will be treated as the first broker or broker-1 because of the broker id configured in server.properties as shown below:

Apache Kafka

```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

```

Broker connects to the zookeeper as it was mentioned in the server.properties file as shown below:

```
############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
```

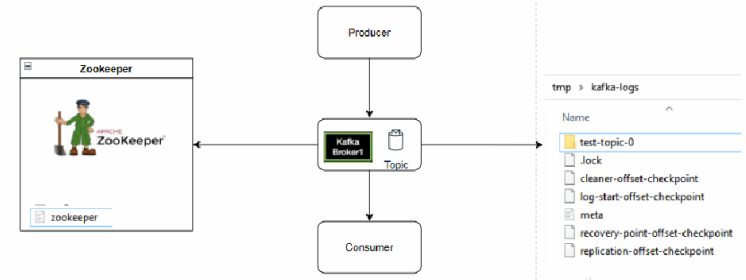
```
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=localhost:2181

# Timeout in ms for connecting to zookeeper
zookeeper.connection.timeout.ms=18000
```

By this now we completed

- ✓ Zookeeper server start
- ✓ Kafak-server start

Hence as only one broker started and connected to zookeeper, we can said it as the single-node cluster.



Step-3: Create a topic

A. we need a topic inside the broker to create the topic use the below command:

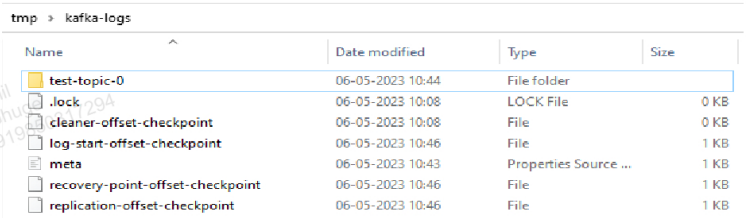
Create a Topic

Apache Kafka

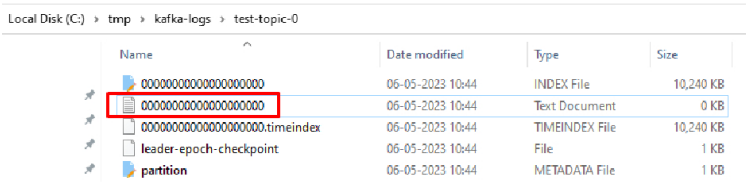
```
kafka-topics.bat --bootstrap-server localhost:9092 --create --topic test-topic --partitions=1 --replication-factor 1
```

```
C:\kafka_2.12-3.4.0>kafka-topics --bootstrap-server localhost:9092 --create --topic test-topic --partitions=1 --replication-factor 1
Created topic test-topic.
```

Once the topic is created we can **see it as a directory**, inside the /tmp/kafka-logs dir.



When we go inside the test-topic-0 directory,we cansee the following files:



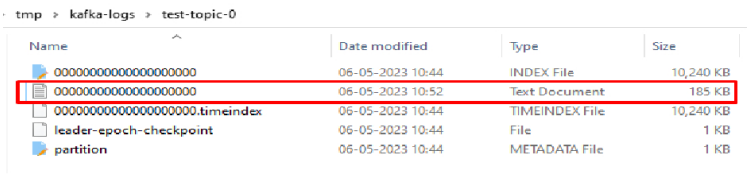
Here the highlighted file is called as a log file, which will stores the data send by the producers and makes this data available for the consumers.

Step-4: Create a Producer and send the data as shown below:

```
C:\kafka_2.12-3.4.0>kafka-console-producer --broker-list localhost:9092 --topic test-topic < sample1.csv
```

```
C:\kafka_2.12-3.4.0>kafka-console-producer --broker-list localhost:9092 --topic test-topic < sample1.csv
```

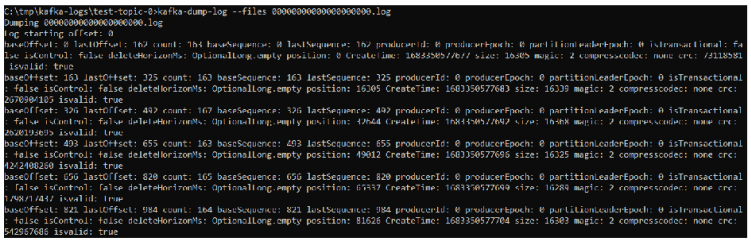
Once the producer sent the data to the topic then we can see the logfile size increased as shown below:



So this is indicating that the data produced by the producer is storing in the log file at the topic of the broker.

We can see the data of the log file only with the kafka-dump-lob utility as shown below:

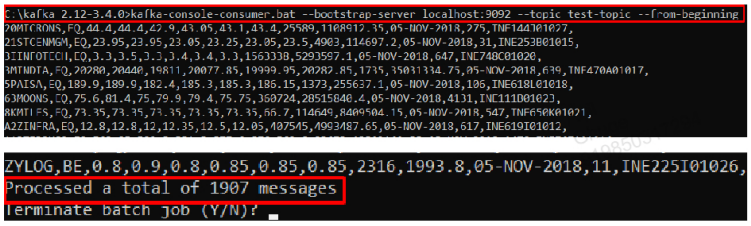
```
C:\tmp\kafka-logs\test-topic-0>kafka-dump-log --files 00000000000000000000.log
```



Step-4: Create Consumer

Using kafka-console-consumer

```
C:\kafka_2.12-3.4.0>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test-topic --from-beginning
```



Multi Node Cluster Setup

UseCase-2: Creating the Topic with 4 partitions-without message key

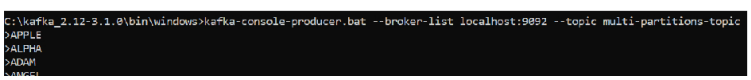
1. Create topic:

kafka-topics.bat --create --topic multi-partitions-topic --replication-factor 1 --partitions 4 --bootstrap-server localhost:9092

2. Instantiate Console Producer without key

kafka-console-producer.bat --broker-list localhost:9092 --topic multi-partitions-topic

Publish the data as shown below:



```
>2
>3
>4
>5
>6
>7
>89
```

3. Instantiate Console Consumer without key

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic multi-partitions-topic --from-beginning

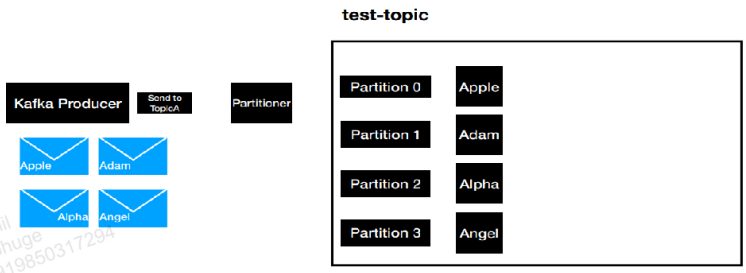
```
C:\kafka_2.12-3.1.0\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic multi-partitions-to
pic --from-beginning
```

Now it will consume the data without any order as shown below:

```
C:\kafka_2.12-3.1.0\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic multi-partitions
pic --from-beginning
ALPHA
ANGEL
4
APPLE
5
6
7
ADAM
4
2
3
89
```

Apache Kafka

This is because as the messages are sending without keys, the messages are assigned all available partitions using the round robbin algorithm as shown below:



UseCase-3: Creating the Topic with 4 partitions-with message key

1. Create topic with 4 partitions

kafka-topics.bat --create --topic multi-partitions-topic --replication-factor 1 --partitions 4 --bootstrap-server localhost:9092

```
C:\kafka_2.12-3.1.0\bin\windows>kafka-topics.bat --create --topic multi-partitions-topic --replication-factor 1 --partitions 4 --bootstrap-server localhost:9092
Created topic multi-partitions-topic.
```

2. Instantiate Console Producer with key

kafka-console-producer.bat --broker-list localhost:9092 --topic multi-pratitions-topic --property "key.separator=-" --property "parse.key=true"

```
C:\kafka_2.12-3.1.0\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic multi-partitions-topic --property "key.separator=-" --property "parse.key=tr
ue"
```

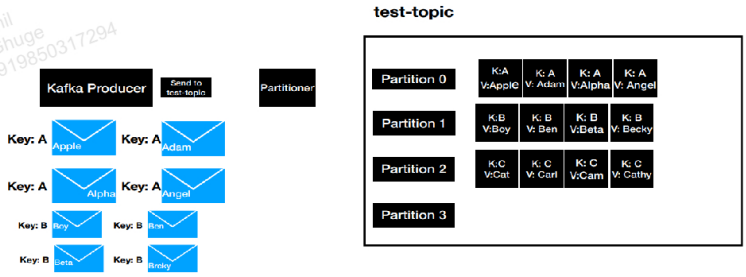
Publish the data as shown below:

```
C:\kafka_2.12-3.1.0\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic multi-partitions-topic --property "key.separator=-" --property "parse.key=tr
ue"
1A-APPLE
4A-AGMS
6C-CAT
1D-DOG
4B-BALL
```

3. Instantiate the consumer with key

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic multi-partitions-topic --from-beginning -property "key.separator= - " --property "print.key=true"

Now we can see the data populated in an order in consumer this is beacuse, if the key is same, the message will be send to the same partition. Partitionar will apply hashing technique on key of each message coming to the partition and identify the same partition with the same key for different values and send the data to only one partition.



Consumer Offset

Every Kafka message within a single partition is uniquely identified by the offset. For example, the offset for the first message in the partition would be 0000, the offset for the second message would be 0001 and so on.

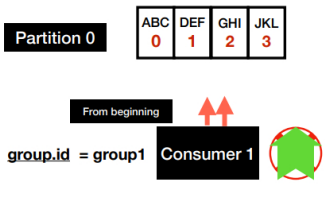
Any message that produced into the topic will have a unique ID called off set. Consumers have the 3 options when it comes to reading the topic:

1. Read the message from the begining

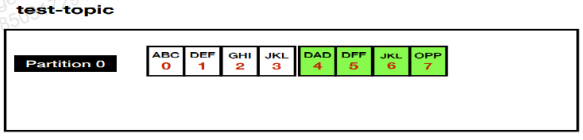
2. Latest: meaning the message only read after the consumer started

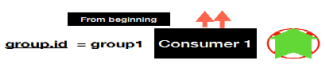
3. Specific Offset: meaning the read the messages in the topic, by passing a specific off set value from the consumer.

Lets say we have one partition and it have some records in it. Lets say we have a consumer, which is going to read the messages from the begining for any kafka consumer it is required that the consumer to provide the group id. Consumer can poll and retrive the multiple records at the same time. as a process, each message it moves the consmer read off set one by one.



lets say some reason the consumer is crashed, while the consumer was down, the producer of the topic some more messages.





now the consumer is brought up after some time, so how does it know that it needs to read from

off set -4. the consumer off set in general stores in an internal topic called __consumer_offsets

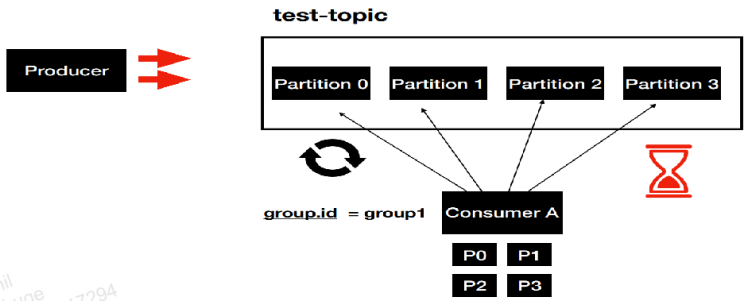
Consumer offsets used by the the consumer to start reading the messages from the point it left of

```
C:\kafka_2.12-3.1.0\bin\windows>kafka-topics.bat --bootstrap-server localhost:9092 --list
__consumer_offsets
multi-partitions-topic
```

```
multi-partitions-topic1
multi-partitions-topic2
test-topic
```

Consumer Groups

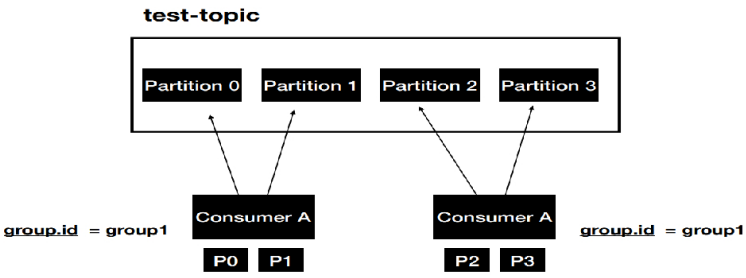
GroupId in Consumer Group plays a key role in Consumer Group. Lets have a test-topic with the four partitions, now we have the consumer-A with the groupID gropu-1, we have only one single consumer pulling all the 4 partitions in the topic and processing them in this case a single thread trying to poll all the partitions.



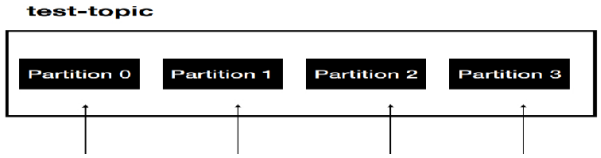
Lets say the producer of the topic is producing messages at a faster way, then then consumer consuming.

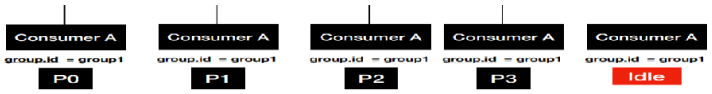
in that case, there will lag we can see at the consumer side and end up with the not processing the real time data. this is where the consumer group comes into the use.

lets spin up another consumer A with the same group id. now the partitions of the topic split between the two consumers, partition-0 and partion-1 are taking care of the fist consumer instane and p2, p3 are taking care of the second instance. thus we scaled our message consumption. this will help process the records litle faster.

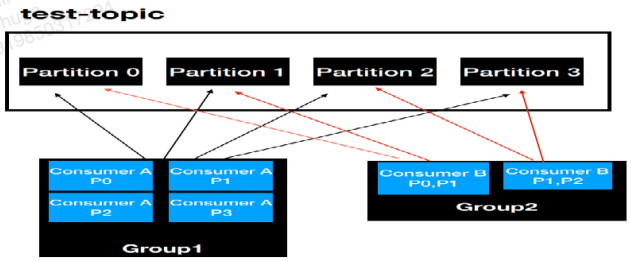


lets make it more faster by spinning up 2 more instances with the same grop id. now we have 5 consumer instances but only 4 partitions are available in that case one of the instance will be idle. this will leads to insufficient usage of the resources.





Consmer Group with Multiple Groups



Consumer Groups are used for scalable message consumption

- Each different application will have a unique consumer group
- Who manages the consumer group?
- Kafka Broker manages the consumer-groups
- Kafka Broker acts as a Group Co-ordinator

Instantiating the Console Consumer with Group Id

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test-topic --group <group-name>

Execute the same command again to have one more instance of kafka consumer.

Here group name we have to use as the result of view consumers group command.

view consumer groups

```
kafka-consumer-groups.bat --bootstrap-server localhost:9092 --list
```


Apache Kafka

Consumer Groups and their Offset

```
kafka-consumer-groups.bat --bootstrap-server localhost:9092 --describe --group console-consumer-27773
```

Instantiate the console producer:

kafka-console-producer.bat --broker-list localhost:9092 --topic test-topic

And stat giving the messages on console as shown below: 1 2 3 4

Then see 1and 4 are comes to consumer1 and 2 and 3 goes to consumer 2

Multi Node Cluster Setup

Install the confluent Kafka

1. Go to the location: https://docs.confluent.io/platform/current/installation/installing_cp/zip-tar.html
2. Copy the confluent platform zip

Confluent Platform

- ZIP

```
curl -O http://packages.confluent.io/archive/7.1/confluent-7.1.1.zip
```

3. Open the command prompt and enter the copied text as below: