Table of Contents

1. Download and test confluent library

1.1. On Desktop, create a folder “kafkatraining”.
1.2. Within “kafkatraining” folder create a folder “software” and extract the downloaded confluent library within this folder. To download the library follow below link and steps
1.3. Download confluent library
https://www.confluent.io/get-started/?product=self-managed
1.4. You may be prompted to enter your name, company, email address, and accept license terms before you can access the page.
After you are on the installation page, choose your archive package or download a package directly by using the curl command.

1.5. **Set confluent home for windows users. In case unable to set due to admin rights leave this step.**
https://docs.confluent.io/platform/current/installation/installing_cp/zip-tar.html#prod-kafka-cli-install

1.6. **For Windows confluent cli is not supported. Can check the list.**
https://docs.confluent.io/platform/6.2/installation/versions-interoperability.html#operating-systems

If you want to setup for windows you will need to install ubuntu or use docker installation. Reference in Reference section at the end

2. Local Set up for confluent

2.1. Start a zookeeper server
2.1.1. On terminal or cmd, go to confluent path and execute the following command

[MAC]
bin/zookeeper-server-start etc/kafka/zookeeper.properties

[Windows]
.\bin\windows\zookeeper-server-start.bat .\etc\kafka\zookeeper.properties

NOTE:
- Windows may get an error classpath is empty.
- Go to confluent/bin/windows/ kafka-run-class.bat file and search for “classpath addition for core” which is line no 97
- Copy the following content and paste it above the “rem classpath addition for core”

```
rem Classpath addition for LSB style path
if exist %BASE_DIR%\share\java\kafka\* (
call :concat %BASE_DIR%\share\java\kafka\*
)
```

- Then run zookeeper again, it should work
- Quorapeermain -> having confluent path without spaces

2.2. Start Kafka Broker

2.2.1. Open a new cmd or terminal and start the kafka server as follows:
[MAC]
bin/kafka-server-start etc/kafka/server.properties

[Windows]
.\bin\windows\kafka-server-start.bat .\etc\kafka\server.properties

2.3. Create Kafka topic

2.3.1. To create a topic, from cmd or terminal go to confluent folder and execute the following command:

[MAC]
bin/kafka-topics --create --topic test --partitions 1 --replication-factor 1 --bootstrap-server localhost:9092

[Windows]
bin\windows\kafka-topics.bat --create --topic test --partitions 1 --replication-factor 1 --bootstrap-server localhost:9092

2.4. To see the list of topics created

[MAC]
/bin/kafka-topics --bootstrap-server localhost:9092 --list

[Windows]
bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 –list

2.5. Start Kafka Console Producer and start producing data

2.5.1. Open a new cmd or terminal

2.5.2. To send data to a topic using console producer:

[MAC]
bin/kafka-console-producer --topic test --broker-list localhost:9092

[Windows]
bin\windows\kafka-console-producer.bat --topic test --broker-list localhost:9092

2.6. Start Kafka Console Consumer and start consuming data

2.6.1. Open a new cmd or terminal

2.6.2. To consume data from a topic using console consumer:

[MAC]
bin/kafka-console-consumer --topic test --bootstrap-server localhost:9092

[Windows]
bin\windows\kafka-console-consumer.bat --topic test --bootstrap-server localhost:9092

2.7. Once the producer and consumer are up and running, start sending messages from producer console and the consumer should read the data immediately.

If the consumer is started after the producer produces messages, then by default it reads the latest messages produced after the consumer is started. To allow the consumer to read the previous messages when it was offline, start the consumer with –from-beginning flag as follows:

[MAC]
bin/kafka-console-consumer --topic test --bootstrap-server localhost:9092 --from-beginning

[Windows]
bin\windows\kafka-console-consumer.bat --topic test --bootstrap-server localhost:9092 --from-beginning

2.8. To get the topic description

2.8.1. **[MAC]**
bin/kafka-topics --describe --bootstrap-server localhost:9092 --topic invoice

[Windows]
bin\windows\kafka-topics.bat --describe --bootstrap-server localhost:9092 --topic invoice

3. Multi-Node Cluster

3.1. Stop the zookeeper, producer, consumer and Kafka broker started previously from all the terminals.
3.2. For multi-node cluster, we need 3 brokers. Follow below steps to configure for 3 brokers. When broker is created, these 3 files are passed as input for 3 different brokers

3.2.1. Go to confluent->etc->kafka->server.properties.
3.2.2. To create 3 clusters, create 3 copies of server.properties

server-0.properties
server-1.properties
server-2.properties

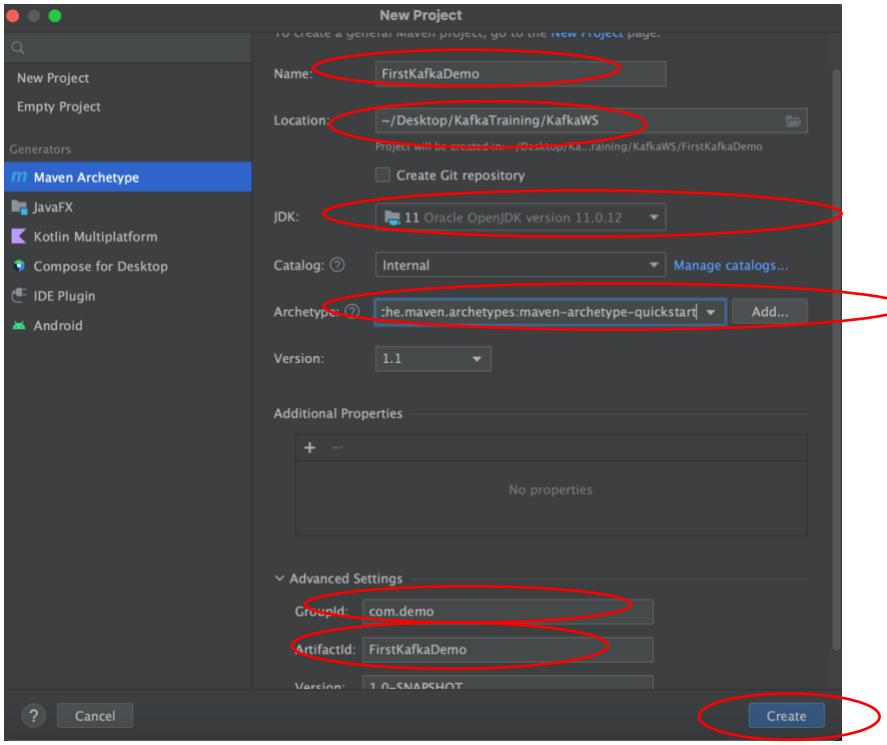
3.2.3. Modify following properties in the properties file:

- server-0.properties :
  broker-id=0
  Uncomment listeners=PLAINTEXT://:9092 [line 34]
  log.dirs=../tmp/kafka-logs-0 [line 62]
  confluent.metadata.server.listeners=http://0.0.0.0:8090 [line 176]

- server-1.properties : broker-id=1
  Uncomment listeners=PLAINTEXT://:9093
  log.dirs=../tmp/kafka-logs-1
  confluent.metadata.server.listeners=http://0.0.0.0:8091

- server-2.properties : broker-id=2
  Uncomment listeners=PLAINTEXT://:9094
  log.dirs=../tmp/kafka-logs-2
  confluent.metadata.server.listeners=http://0.0.0.0:8092

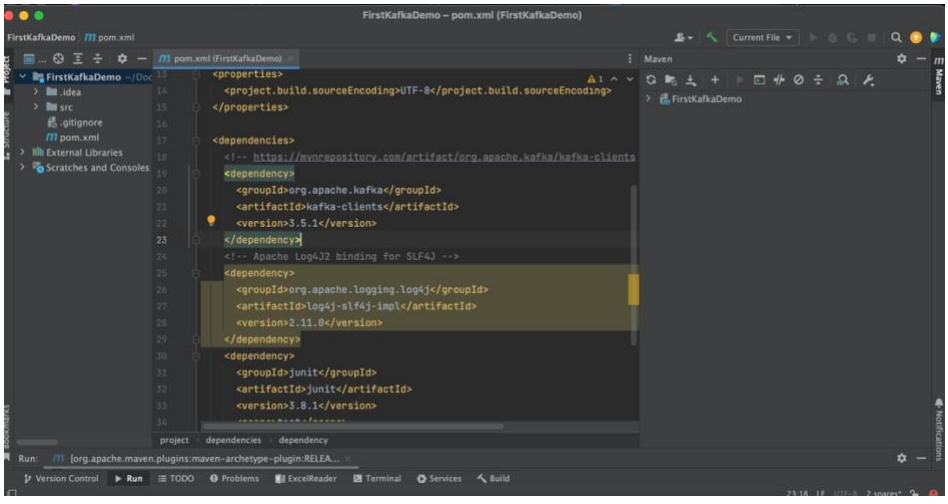
3.3. Create a folder “KafkaWS” within the “kafkatraining” folder
3.4. Open IntelliJ, choose Maven Project and enter following and click on Create



3.5. Add the following dependencies in pom.xml file:

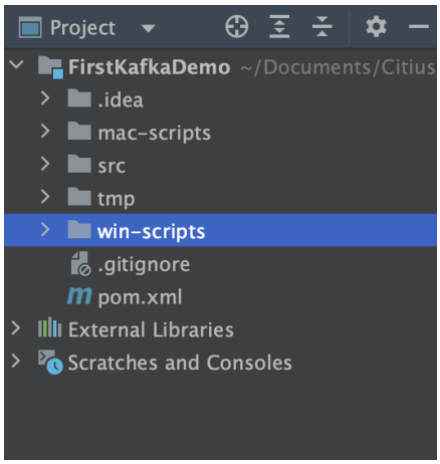
```
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>3.5.1</version>
</dependency>
```

3.6. After adding the dependencies, expand the Maven tab on left and click on Reload.



3.7. Create a folder based on your OS as follows within the FirstKafkaDemo folder:

3.8. Your folder structure will look as follows. **NO NEED TO CREATE BOTH** mac and win folders. Create anyone based on your OS:-



3.9. MAC => Create a “mac-scripts” and add following 7 files within this folder

NOTE: PLEASE UPDATE <path-to-confluent> to your confluent path

3.9.1. zookeeper-start.sh

```
#!/bin/bash
CONFLUENT_HOME=/<path-to-confluent>/confluent-7.5.0
$CONFLUENT_HOME/bin/zookeeper-server-start
$CONFLUENT_HOME/etc/kafka/zookeeper.properties
```

3.9.2. 0-kafka-server-start.sh

```
#!/bin/bash
CONFLUENT_HOME=/<path-to-confluent>/confluent-7.5.0
$CONFLUENT_HOME/bin/kafka-server-start
$CONFLUENT_HOME/etc/kafka/server-0.properties
```

3.9.3. 1-kafka-server-start.sh

```
#!/bin/bash
CONFLUENT_HOME=/<path-to-confluent>/confluent-7.5.0
$CONFLUENT_HOME/bin/kafka-server-start
$CONFLUENT_HOME/etc/kafka/server-1.properties
```

3.9.4. 2-kafka-server-start.sh

```
#!/bin/bash
CONFLUENT_HOME=/<path-to-confluent>/confluent-7.5.0
$CONFLUENT_HOME/bin/kafka-server-start
$CONFLUENT_HOME/etc/kafka/server-2.properties
```

3.9.5. topic-create.sh

```
#!/bin/bash
CONFLUENT_HOME=/<path-to-confluent>/confluent-7.5.0
$CONFLUENT_HOME/bin/kafka-topics --create --bootstrap-server localhost:9092 --topic invoice --partitions 5 --replication-factor 3 --config segment.bytes=1000000
```

3.9.6. describe-topic.sh

```
#!/bin/bash
CONFLUENT_HOME=/<path-to-confluent>/confluent-7.5.0
        $CONFLUENT_HOME/bin/kafka-topics --describe  --bootstrap-server
        localhost:9092 --topic invoice
```

3.9.7. list-topic.sh

```
#!/bin/bash
CONFLUENT_HOME=/<path-to-confluent>/confluent-7.5.0
$CONFLUENT_HOME/bin/kafka-topics --bootstrap-server localhost:9092 –list
```

3.9.8. kafka-consumer.sh

```
#!/bin/bash
CONFLUENT_HOME=/<path-to-confluent>/confluent-7.5.0
        $CONFLUENT_HOME/ bin/kafka-console-consumer  --topic invoice --bootstrap-
        server localhost:9092 --from-beginning
```

3.10. WINDOWS => Create a “win-scripts” and add following 7 files within this folder

3.10.1. zookeeper-start.cmd

```
set CONFLUENT_HOME=<PATH TO CONFLUENT>\confluent-7.7.0
echo %CONFLUENT_HOME%
%CONFLUENT_HOME%\bin\windows\zookeeper-server-start.bat %CONFLUENT_HOME %\etc\kafka\
zookeeper.properties
```

3.10.2. 0-kafka-server-start.cmd

```
%CONFLUENT_HOME%\bin\windows\kafka-server-start.bat %CONFLUENT_HOME%\etc\kafka\server-
0.properties
```

3.10.3. 1-kafka-server-start.cmd

```
%CONFLUENT_HOME%\bin\windows\kafka-server-start.bat
%CONFLUENT_HOME%\locaetc/kafka/server-1.properties
```

3.10.4. 2-kafka-server-start.cmd

```
%CONFLUENT_HOME%\bin\windows\kafka-server-start.bat
%CONFLUENT_HOME%\etc\kafka\server-2.properties
```

3.10.5. topic-create.cmd

```
%$CONFLUENT_HOME%\bin\windows/kafka-topics.bat --create --bootstrap-server
localhost:9092 --topic invoice --partitions 5 --replication-factor 3 --config
segment.bytes=1000000
```

3.10.6. describe-topic.cmd

```
%$CONFLUENT_HOME%\bin\windows\kafka-topics.bat --describe  --bootstrap-server
localhost:9092 --topic invoice
```

3.10.7. list-topic.cmd

```
%$CONFLUENT_HOME%\bin\windows\kafka-topics.bat --bootstrap-server
localhost:9092 –list
```

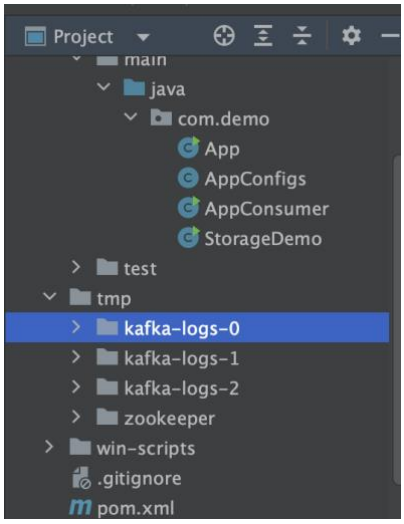
3.10.8. kafka-consumer.cmd

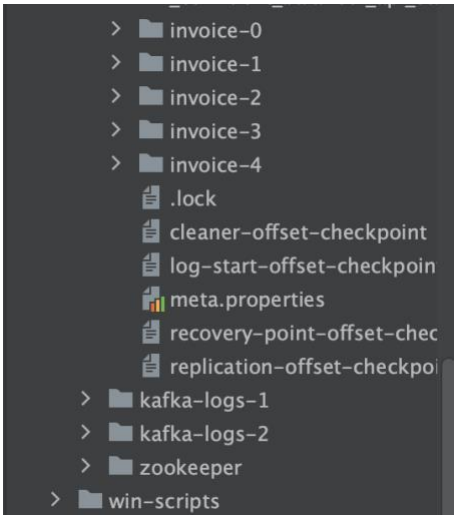
%CONFLUENT_HOME%\bin\kafka-console-consumer.bat --topic invoice --bootstrap-server localhost:9092 --from-beginning

3.11. Once the shell scripts are created , open files within INTELLIJ terminal and execute one by one in the following sequence

Zookeepeer
Broker-0
Broker-1
Broker-2
Create-topic
List-topic

3.12. Once all the brokers , server started, topic ceated, the tmp folder should have been created with below log files. Expand kafka-logs-0 and you will see 5 invoice partitions created. Likewise 5 partitions within each log folder.





3.13. Execute Describe-topic and you will see the information about the topic.

Topic: invoice TopicId: c3vk2p6ySGStLAqifb_Jzg PartitionCount: 3 ReplicationFactor: 3
Configs: segment.bytes=100000
Topic: invoice Partition: 0 Leader: 2 Replicas: 2,1,0 Isr: 2,1,0 Offline:
Topic: invoice Partition: 1 Leader: 1 Replicas: 1,0,2 Isr: 1,0,2 Offline:
Topic: invoice Partition: 2 Leader: 0 Replicas: 0,2,1 Isr: 0,2,1 Offline:

4. Java Producer API

4.1. Create following classes to create a producer to start producing data to the topic invoice

4.2. AppConfigs.java

```
package com.demo;
        class AppConfigs {
            final static String applicationID = "StorageDemo";
            final static String bootstrapServers = "localhost:9092, localhost:9093";
            final static String topicName = "invoice";
            final static int numEvents = 500000;
        }
```

4.3. Create StorageDemo.java

```
package com.demo;

import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.IntegerSerializer;
import org.apache.kafka.common.serialization.StringSerializer;
import java.util.Properties;
public class StorageDemo
{
   public static void main( String[] args )
   {
      System.out.println( "Hello World!" );
      System.out.println("Producer");
      System.out.println ("Creating Kafka Producer...");

Properties props = new Properties();
      props.put(ProducerConfig.CLIENT_ID_CONFIG, AppConfigs.applicationID);
      props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
AppConfigs.bootstrapServers);
      props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
IntegerSerializer.class.getName());
      props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());

      KafkaProducer<Integer, String> producer = newKafkaProducer<>(props);

      System.out.println ("Start sending messages...");
for (int i = 1; i <= AppConfigs.numEvents; i++) {
        producer.send(new ProducerRecord<>(AppConfigs.topicName, i,"Simple Message-" +
i));
      }
      System.out.println ("Finished - Closing Kafka Producer.");
      producer.close();
   }
}
```

4.4. Run StorageDemo.java

5. Zookeeper Shell

5.1. Open command prompt from within the confluent-7.5.0/bin folder

5.2. execute below command only after the zookeeper server is started

```
windows\zookeeper-shell.bat localhost:2181
```

5.3. once the shell start

```
ls /
ls /brokers
ls /brokers/ids
get /controller
```

NOTE : PLEASE UPDATE THE CONFLUENT PATH TO YOUR PATH AS AND WHERE NEEDED

6. Understand log, index and timestamp files

6.1. .log file - This file contains the actual records and maintains the records up to a specific offset. The name of the file depicts the starting offset added to this file.
6.2. .index file - This file has an index that maps a record offset to the byte offset of the record within the
6.3. .log file. This mapping is used to read the record from any specific offset.
6.4. .timeindex file - This file contains the mapping of the timestamp to record offset, which internally maps to the byte offset of the record usingthe
6.5. .index file. This helps in accessing the records from the specific timestamp.
6.6. .snapshot file - contains a snapshot of the producer state regarding sequence IDs used to avoid duplicate records. It is used when, after a new leader is elected, the preferred one comes back and needs such a state to become a leader again. This is only available for the active segment (log file).
6.7. leader-epoch-checkpoint - It refers to the number of leaders previously assigned by the controller. The replicas use the leader epoch as a means of verifying the current leader. The leader-epoch-checkpoint file contains two columns: epochs and offsets. Each row is a checkpoint for the latest recorded leader epoch and the leader's latest offset upon becoming leader

7. Dump logs

7.1. To get the dumps from the partition files use below command from within the terminal of your editor.

Execute below commands from within the root of your folder

7.2. **<path to confluent>**/confluent-7.5.0/bin/kafka-run-class kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files tmp/kafka-logs-0/invoice-0/00000000000000000000.log > log0.txt

7.3. **<path to confluent>**/confluent-7.5.0/bin/kafka-run-class kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files tmp/kafka-logs-0/invoice-0/00000000000000000000.index > index0.txt

7.4. **<path to confluent>**/confluent-7.5.0/bin/kafka-run-class kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files tmp/kafka-logs-0/invoice-0/00000000000000000000.timeindex > time0.txt

8. Key for a message

8.1. Update the for loop of StorageDemo to specify same key value for all messages.

```
for (int i = 1; i <= 20; i++) {
        producer.send(new ProducerRecord<>(AppConfigs.topicName, 1,"Simple Message-" + i));
    }
```

8.2. Get the dump logs and you will see all messages land up in 1 partition only. That partition is random so you will need to check which partition they are stored

9. Partition info and send message by partition number

9.1. Producer can get the information about the partition as follows:

```
for(PartitionInfo info :producer.partitionsFor("invoice")) {
        System.out.println(info.partition());
        System.out.println(info.toString());
    }
```

9.2. To send message to a particular partition update for loop as follows:

```
for (int i = 1; i <= 20; i++) {
        producer.send(new ProducerRecord<>(AppConfigs.topicName, 1, 2, "Msg" + i));
    }
```

10. Message Size

10.1. Try sending large message to kafka broker . Modify the producer code as follows and should see RecordTooLargeException thrown:

```
char[] chars = new char[1048576];
Arrays.fill(chars, 'd');
String s=new String(chars);
producer.send(new ProducerRecord<Integer, String>(AppConfigs.topicName, 0, 1, s),
        (metadata, exception) ->{
            System.out.println(metadata.topic());
            if(exception!=null)
                System.out.println(exception);
        }
    });
```

10.2. Create topic as follows:

```
kafka-topics.sh --bootstrap-server localhost:9092 --create --topic large-message --partitions 3 --
replication-factor 1
```

10.3. Add the necessary max.message.bytes configuration for 10MB

```
kafka-configs.sh --bootstrap-server localhost:9092 \
        --alter --entity-type topics \
            --entity-name large-message \
            --add-config max.message.bytes=10485880
```

10.4. Add below in all the server-X.properties file:

```
replica.fetch.max.bytes=10485880
```

10.5. Create consumer as follows:

```
kafka-console-consumer.sh --bootstrap-server localhost:9092 \
--topic large-message \
--from-beginning \
--consumer-property max.partition.fetch.bytes=10485880
```

10.6. Create producer as follows:

10.6.1. Code side:

```
properties.setProperty(ProducerConfig.MAX_REQUEST_SIZE_CONFIG, "10485880");
```

10.6.2. Try side large message to this newly created topic and it should work.

DO NOT FORGET TO CHANGE THE TOPIC NAME IN PRODUCER BEFORE SENDING

10.6.3. If CLI:

```
kafka-console-producer.sh --bootstrap-server localhost:9092 \
  --topic large-message \
  --producer-property max.request.size=10485880
```

11. ISR – IN-sync Replica

11.1. Kafka producers only write data to the current leader broker for a partition. Kafka producers must also specify a level of acknowledgment acks to specify if the message must be written to a minimum number of replicas before being considered a successful write.

11.2. After topic is created
kafka-configs.sh --bootstrap-server localhost:9092 --alter --entity-type topics --entity-name configured-topic --add-config min.insync.replicas=2

Example

11.3. Configure in sync replicas while creating topic

$KAFKA_HOME/bin/kafka-topics --create --bootstrap-server localhost:9092 --topic test2 --partitions 3 --replication-factor 3 --config min.insync.replicas=2

12. Topic commands:

12.1. Delete a topic

bin/windows/kafka-topics --bootstrap-server localhost:9092 --topic test --delete

12.2. Alter a topic

bin/kafka-topics --alter --bootstrap-server localhost:9092 --topic test1 --partitions 5

13. REFERENCES

13.1. To run on windows need to install ubuntu and enable WSL. Open ubuntu shell and execute below commands

https://docs.confluent.io/platform/current/installation/installing_cp/zip-tar.html#prod-kafka-cli-install

Microsoft store => ubuntu and install
sudo apt update
sudo apt install openjdk-11-jre-headless
java -version
cd /home/confluent-7.5.0/etc/kafka
cp -rf /mnt/d/{File with path} .
//Copy all files related to kafka one by one

cd /home/confluent-7.5.0/etc/confluent-control-center
cp -rf /mnt/d/{File with path} .
//Copy all files related to confluent control center one by one

https://www.confluent.io/blog/set-up-and-run-kafka-on-windows-and-wsl-

2/?_ga=2.13585015.1966925725.1696133559-2102384477.1695907845&_gac=1.121324922.1696511586.CjwKCAjwvfmoBhAwEiwAG2tqzNeAgGN2GqHmkaSipjeBqZqrSvCk0MAAYF0DNX8V98TwsO990LjZqBoCK-wQAvD_BwE&_gl=1*qwqatt*_ga*MjEwMjM4NDQ3Ny4xNjk1OTA3ODQ1*_ga_D2D3EGKSGD*MTY5NjU1NzI4My40My4xLjE2OTY1NTg3NDcuMTEuMC4w

confluent local services start

Your output should resemble:

Starting Zookeeper
Zookeeper is [UP]
Starting Kafka
Kafka is [UP]
Starting Schema Registry
Schema Registry is [UP]
Starting Kafka REST
Kafka REST is [UP]
Starting Connect
Connect is [UP]
Starting KSQL Server
KSQL Server is [UP]
Starting Control Center
Control Center is [UP]

http://localhost:9021 => control center
To stop confluent
confluent local services stop

Videos to learn and understand
https://developer.confluent.io/courses/?course=for-developers#fundamentals

Docker set up for confluent
https://developer.confluent.io/tutorials/creating-first-apache-kafka-streams-application/kstreams.html

Previous versions
https://www.confluent.io/previous-versions/

Large Messages in kafka

https://medium.com/workday-engineering/large-message-handling-with-kafka-chunking-vs-external-store-33b0fc4ccf14

https://learn.conduktor.io/kafka/how-to-send-large-messages-in-apache-kafka/

Add partitions to existing topic

https://stackoverflow.com/questions/33677871/is-it-possible-to-add-partitions-to-an-existing-topic-in-kafka-0-8-2

https://www.conduktor.io/kafka/what-is-apache-kafka/

C:\Users\him26\Desktop\KafkaTraining\software\confluent-7.5.3\confluent-7.5.3>bin\windows\zookeeper-server-start.bat etc\kafka\zookeeper.properties
Classpath is empty. Please build the project first e.g. by running 'gradlew jarAll'

https://rohithsankepally.github.io/Kafka-Hands-On/
https://docs.confluent.io/kafka/operations-tools/partition-determination.html

https://medium.com/@Irori/dangerous-default-kafka-settings-part-1-2ee99ee7dfe5#:~:text=records%20within%20max.poll.interval,couple%20of%20database%2Doperations)
https://levelup.gitconnected.com/optimize-the-performance-of-the-poll-loop-in-kafka-consumer-df7d3da96cd0