BDAT 1008
Data Collection and Curation

Assignment 4

Kafka Report

Document Summary

Document Item	Status
Document Title	Report on Three scenarios of Kafka implementation.
Date Last Modified	18-July-2023
Status	Final
Document Description	This document provides an implementation of three different scenarios of Kafka consumer and Kafka producer console/application.

Prepared By

Name	SID
Ruchika Gupta	200559617

Table of Contents

Introduction

This Report provides the implementation of three different scenarios of Kafka consumer and Kafka producer console/application.

Apache Kafka is a distributed streaming platform used for various purposes, such as a distributed message broker and data stream processing. The system architecture consists of the following components:

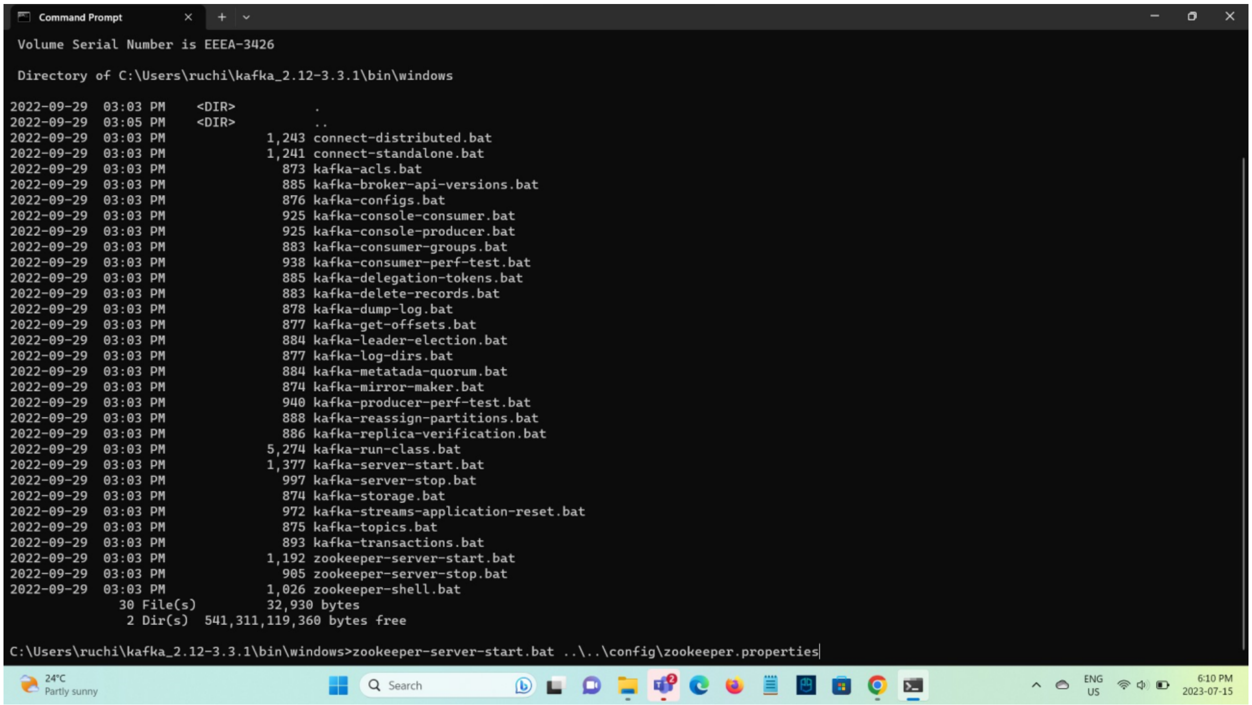
Zookeeper: Maintains state and coordination between nodes in the Kafka cluster.

Kafka Brokers: Act as the core component of the platform, storing and transmitting data streams.

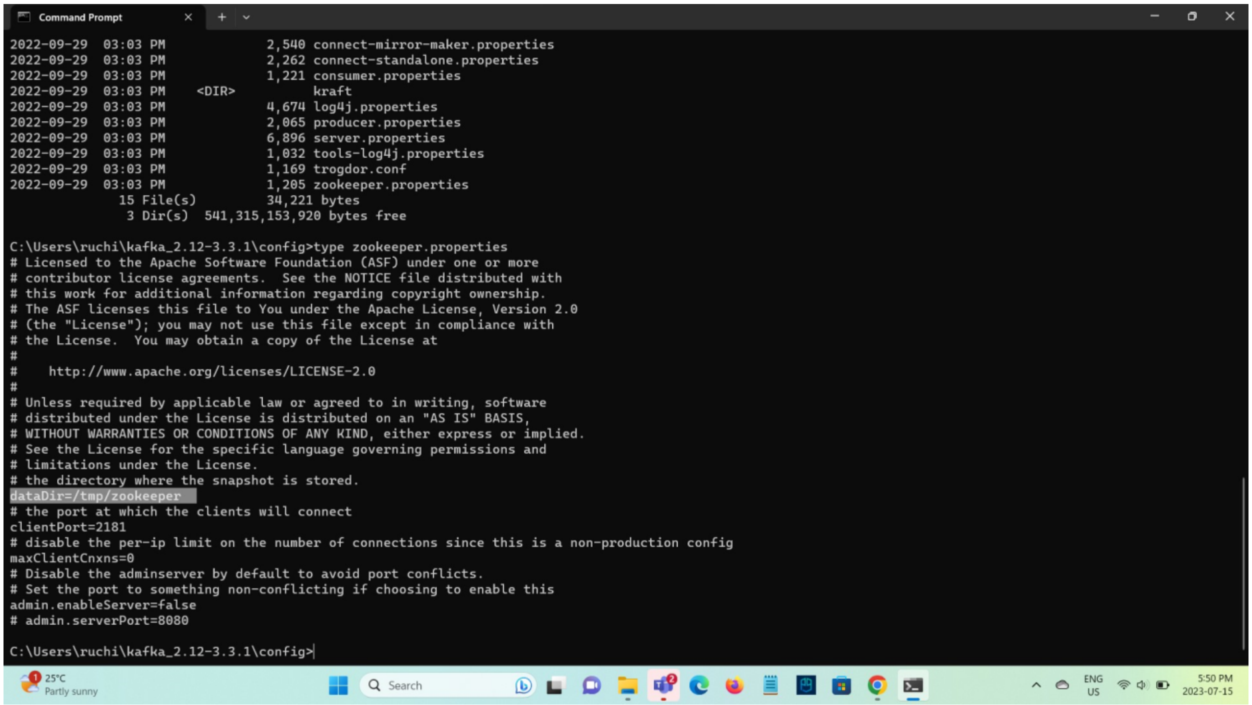
Producers: Responsible for inserting data into the Kafka cluster.

Consumers: Read data from the Kafka cluster.

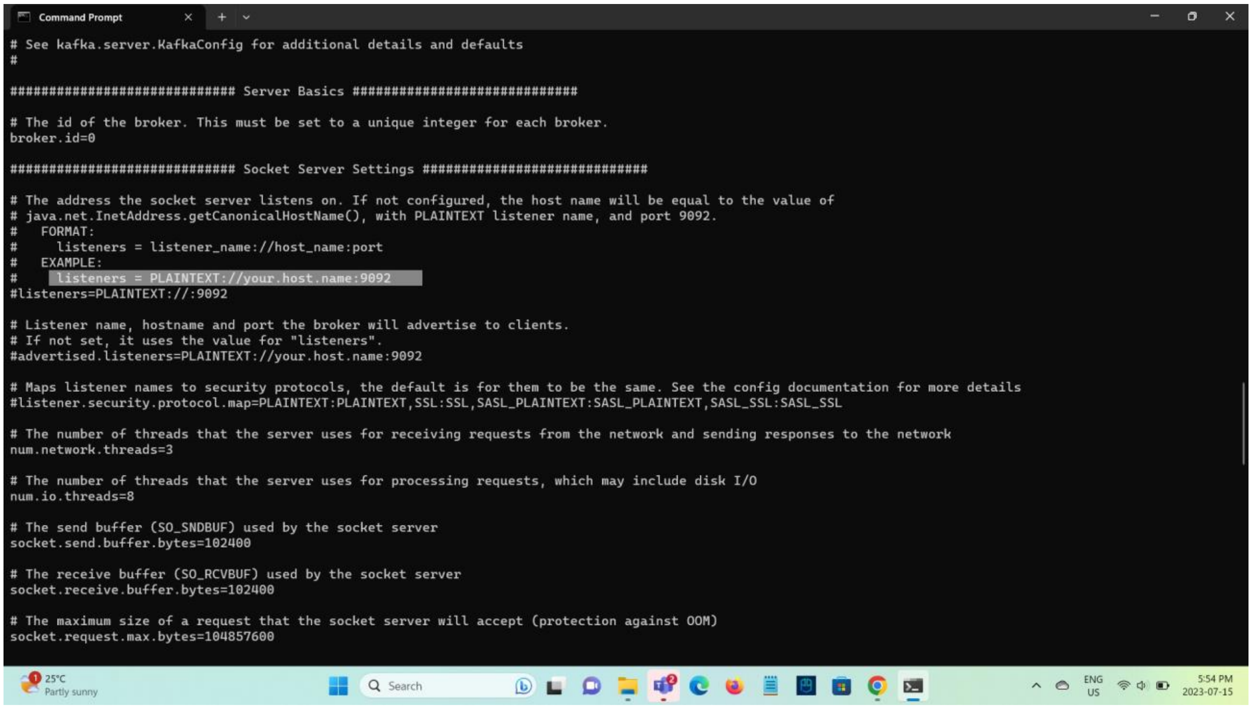
1. Zookeeper & Kafka Properties



```
 Volume Serial Number is EEEA-3426

 Directory of C:\Users\ruchi\kafka_2.12-3.3.1\bin\windows

2022-09-29  03:03 PM    <DIR>          .
2022-09-29  03:05 PM    <DIR>          ..
2022-09-29  03:03 PM             1,243 connect-distributed.bat
2022-09-29  03:03 PM             1,241 connect-standalone.bat
2022-09-29  03:03 PM               873 kafka-acls.bat
2022-09-29  03:03 PM               885 kafka-broker-api-versions.bat
2022-09-29  03:03 PM               876 kafka-configs.bat
2022-09-29  03:03 PM               925 kafka-console-consumer.bat
2022-09-29  03:03 PM               925 kafka-console-producer.bat
2022-09-29  03:03 PM               883 kafka-consumer-groups.bat
2022-09-29  03:03 PM               938 kafka-consumer-perf-test.bat
2022-09-29  03:03 PM               885 kafka-delegation-tokens.bat
2022-09-29  03:03 PM               883 kafka-delete-records.bat
2022-09-29  03:03 PM               878 kafka-dump-log.bat
2022-09-29  03:03 PM               877 kafka-get-offsets.bat
2022-09-29  03:03 PM               884 kafka-leader-election.bat
2022-09-29  03:03 PM               877 kafka-log-dirs.bat
2022-09-29  03:03 PM               884 kafka-metatada-quorum.bat
2022-09-29  03:03 PM               874 kafka-mirror-maker.bat
2022-09-29  03:03 PM               940 kafka-producer-perf-test.bat
2022-09-29  03:03 PM               888 kafka-reassign-partitions.bat
2022-09-29  03:03 PM               886 kafka-replica-verification.bat
2022-09-29  03:03 PM             5,274 kafka-run-class.bat
2022-09-29  03:03 PM             1,377 kafka-server-start.bat
2022-09-29  03:03 PM               997 kafka-server-stop.bat
2022-09-29  03:03 PM               874 kafka-storage.bat
2022-09-29  03:03 PM               972 kafka-streams-application-reset.bat
2022-09-29  03:03 PM               875 kafka-topics.bat
2022-09-29  03:03 PM               893 kafka-transactions.bat
2022-09-29  03:03 PM             1,192 zookeeper-server-start.bat
2022-09-29  03:03 PM               905 zookeeper-server-stop.bat
2022-09-29  03:03 PM             1,026 zookeeper-shell.bat
              30 File(s)         32,930 bytes
               2 Dir(s)  541,311,119,360 bytes free

C:\Users\ruchi\kafka_2.12-3.3.1\bin\windows>zookeeper-server-start.bat ..\..\config\zookeeper.properties
```



```
2022-09-29  03:03 PM             2,540 connect-mirror-maker.properties
2022-09-29  03:03 PM             2,262 connect-standalone.properties
2022-09-29  03:03 PM             1,221 consumer.properties
2022-09-29  03:03 PM    <DIR>          kraft
2022-09-29  03:03 PM             4,674 log4j.properties
2022-09-29  03:03 PM             2,065 producer.properties
2022-09-29  03:03 PM             6,896 server.properties
2022-09-29  03:03 PM             1,032 tools-log4j.properties
2022-09-29  03:03 PM             1,169 trogdor.conf
2022-09-29  03:03 PM             1,205 zookeeper.properties
              15 File(s)         34,221 bytes
               3 Dir(s)  541,315,153,920 bytes free

C:\Users\ruchi\kafka_2.12-3.3.1\config>type zookeeper.properties
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    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-production config
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

C:\Users\ruchi\kafka_2.12-3.3.1\config>
```



```
# See kafka.server.KafkaConfig for additional details and defaults
#

############################# Server Basics #############################

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

############################# Socket Server Settings #############################

# The address the socket server listens on. If not configured, the host name will be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092

# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
#advertised.listeners=PLAINTEXT://your.host.name:9092

# Maps listener names to security protocols, the default is for them to be the same. See the config documentation for more details
#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL

# The number of threads that the server uses for receiving requests from the network and sending responses to the network
num.network.threads=3

# The number of threads that the server uses for processing requests, which may include disk I/O
num.io.threads=8

# The send buffer (SO_SNDBUF) used by the socket server
socket.send.buffer.bytes=102400

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept (protection against OOM)
socket.request.max.bytes=104857600
```

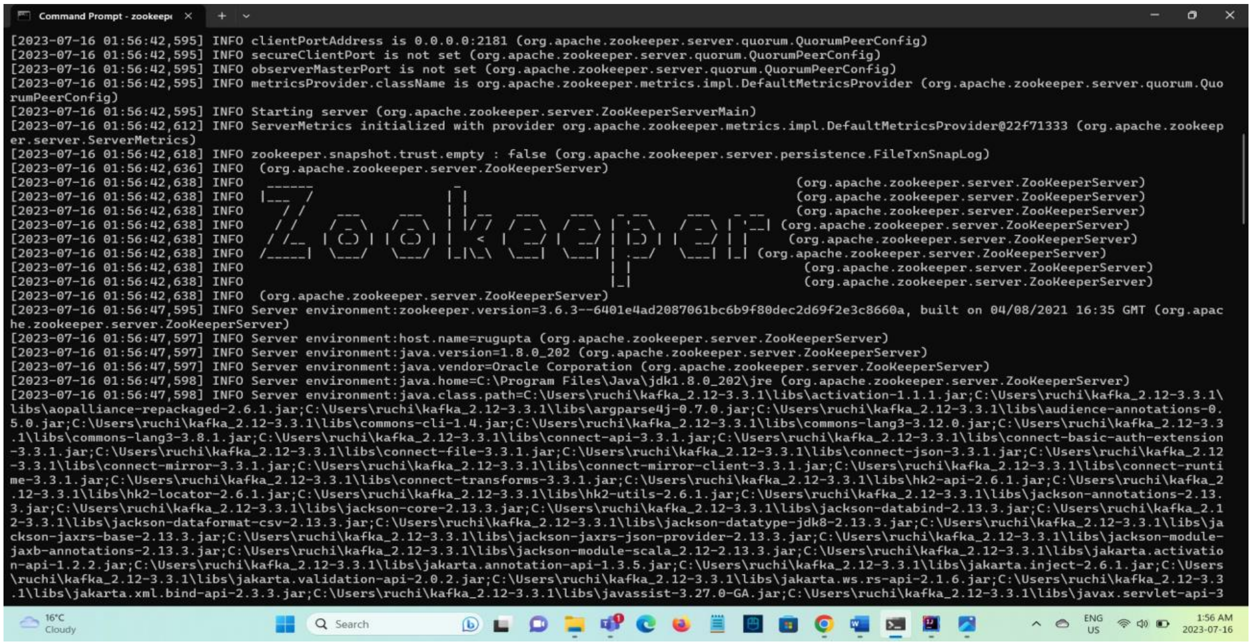
As we started setting up Kafka, the first thing we had to do was to launch a Zookeeper instance. Zookeeper plays a critical role in coordinating and managing Kafka brokers, ensuring the overall stability and reliability of the Kafka cluster.

To start Zookeeper, we accessed the server and executed the appropriate command to launch the instance. Once it was up and running, we double-checked its status to ensure that it was working correctly.

With Zookeeper running, we were ready to proceed with the Kafka setup. The integration between Kafka and Zookeeper is fundamental, as Zookeeper keeps track of the Kafka cluster state, such as broker configuration, topic configuration, and broker health. This coordination enables Kafka to function as a distributed system efficiently.

The below screenshots describe how we started the Zookeeper server by running the command “config/zookeeper.properties” which provides the default configuration for the Zookeeper server to run.

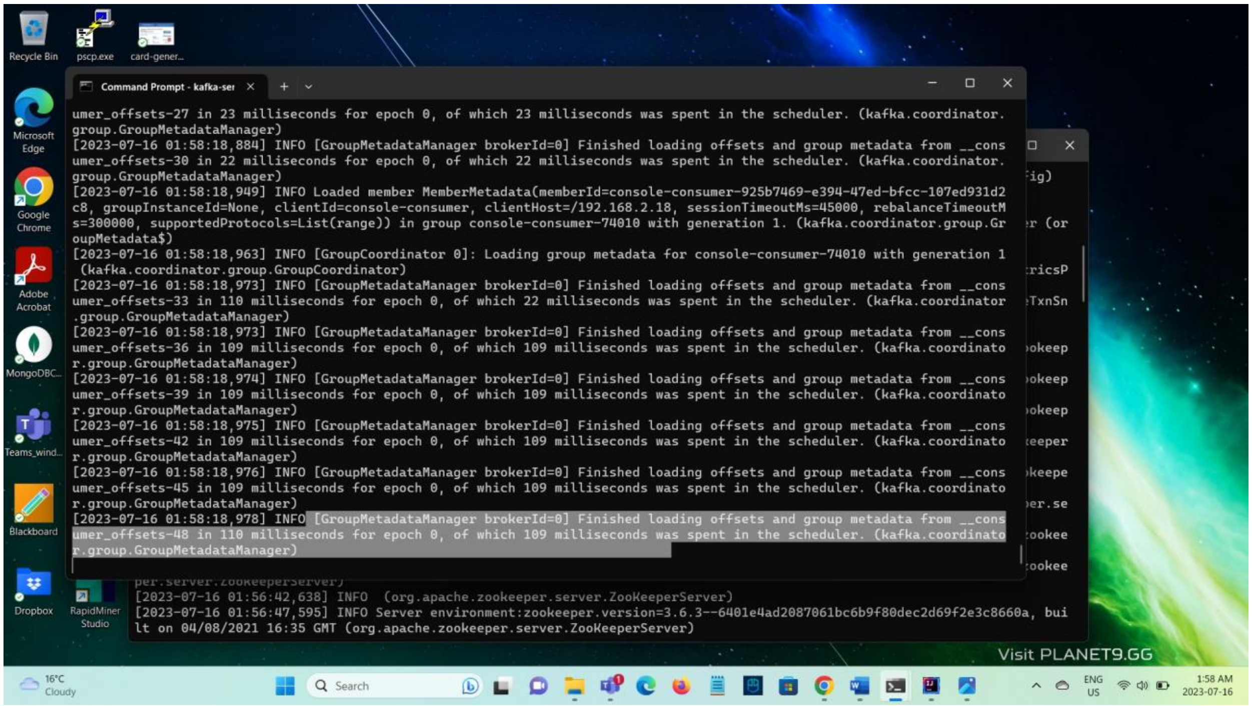
2. Zookeeper server start.



```
[2023-07-16 01:56:42,595] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-07-16 01:56:42,595] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-07-16 01:56:42,595] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-07-16 01:56:42,595] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.Quo
rumPeerConfig)
[2023-07-16 01:56:42,595] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2023-07-16 01:56:42,612] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@22f71333 (org.apache.zookeep
er.server.ServerMetrics)
[2023-07-16 01:56:42,618] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2023-07-16 01:56:42,636] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO   ______                  _                                          (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO  |___  /                 | |                                         (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO     / /    ___     ___   | | __   ___    ___   _ __     ___   _ __   (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO    / /    / _ \   / _ \  | |/ /  / _ \  / _ \ | '_ \   / _ \ | '__| (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO   / /__  | (_) | | (_) | |   <  |  __/ |  __/ | |_) | |  __/ | |    (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO  /_____|  \___/   \___/  |_|\_\  \___|  \___| | .__/   \___| |_| (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO                                               | |                     (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO                                               |_|                     (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:47,595] INFO Server environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apac
he.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:47,597] INFO Server environment:host.name=rugupta (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:47,597] INFO Server environment:java.version=1.8.0_202 (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:47,597] INFO Server environment:java.vendor=Oracle Corporation (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:47,598] INFO Server environment:java.home=C:\Program Files\Java\jdk1.8.0_202\jre (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:47,598] INFO Server environment:java.class.path=C:\Users\ruchi\kafka_2.12-3.3.1\libs\activation-1.1.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\
libs\aopalliance-repackaged-2.6.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\argparse4j-0.7.0.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\audience-annotations-0.
5.0.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\commons-cli-1.4.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\commons-lang3-3.12.0.jar;C:\Users\ruchi\kafka_2.12-3.3
.1\libs\commons-lang3-3.8.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\connect-api-3.3.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\connect-basic-auth-extension
-3.3.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\connect-file-3.3.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\connect-json-3.3.1.jar;C:\Users\ruchi\kafka_2.12
-3.3.1\libs\connect-mirror-3.3.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\connect-mirror-client-3.3.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\connect-runti
me-3.3.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\connect-transforms-3.3.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\hk2-api-2.6.1.jar;C:\Users\ruchi\kafka_2
.12-3.3.1\libs\hk2-locator-2.6.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\hk2-utils-2.6.1.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jackson-annotations-2.13.
3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jackson-core-2.13.3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jackson-databind-2.13.3.jar;C:\Users\ruchi\kafka_2.1
2-3.3.1\libs\jackson-dataformat-csv-2.13.3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jackson-datatype-jdk8-2.13.3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\ja
ckson-jaxrs-base-2.13.3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jackson-jaxrs-json-provider-2.13.3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jackson-module-
jaxb-annotations-2.13.3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jackson-module-scala_2.12-2.13.3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jakarta.activatio
n-api-1.2.2.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jakarta.annotation-api-1.3.5.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jakarta.inject-2.6.1.jar;C:\Users
\ruchi\kafka_2.12-3.3.1\libs\jakarta.validation-api-2.0.2.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\jakarta.ws.rs-api-2.1.6.jar;C:\Users\ruchi\kafka_2.12-3.3
.1\libs\jakarta.xml.bind-api-2.3.3.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\javassist-3.27.0-GA.jar;C:\Users\ruchi\kafka_2.12-3.3.1\libs\javax.servlet-api-3
```

The above screenshot shows that we were successfully able to run the Zookeeper.

3. Kafka server start.

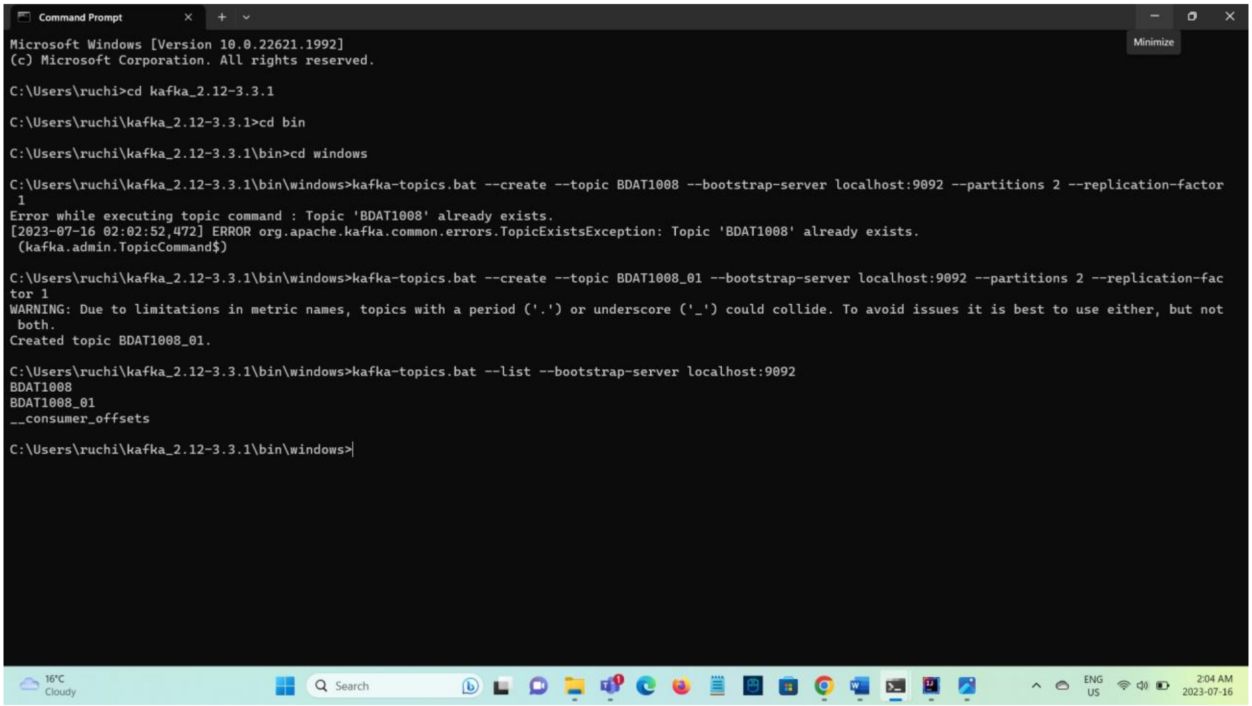


```
umer_offsets-27 in 23 milliseconds for epoch 0, of which 23 milliseconds was spent in the scheduler. (kafka.coordinator.
group.GroupMetadataManager)
[2023-07-16 01:58:18,884] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __cons
umer_offsets-30 in 22 milliseconds for epoch 0, of which 22 milliseconds was spent in the scheduler. (kafka.coordinator.
group.GroupMetadataManager)
[2023-07-16 01:58:18,949] INFO Loaded member MemberMetadata(memberId=console-consumer-925b7469-e394-47ed-bfcc-107ed931d2
c8, groupInstanceId=None, clientId=console-consumer, clientHost=/192.168.2.18, sessionTimeoutMs=45000, rebalanceTimeoutM
s=300000, supportedProtocols=List(range)) in group console-consumer-74010 with generation 1. (kafka.coordinator.group.Gr
oupMetadata$)
[2023-07-16 01:58:18,963] INFO [GroupCoordinator 0]: Loading group metadata for console-consumer-74010 with generation 1
 (kafka.coordinator.group.GroupCoordinator)
[2023-07-16 01:58:18,973] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __cons
umer_offsets-33 in 110 milliseconds for epoch 0, of which 22 milliseconds was spent in the scheduler. (kafka.coordinator
.group.GroupMetadataManager)
[2023-07-16 01:58:18,973] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __cons
umer_offsets-36 in 109 milliseconds for epoch 0, of which 109 milliseconds was spent in the scheduler. (kafka.coordinato
r.group.GroupMetadataManager)
[2023-07-16 01:58:18,974] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __cons
umer_offsets-39 in 109 milliseconds for epoch 0, of which 109 milliseconds was spent in the scheduler. (kafka.coordinato
r.group.GroupMetadataManager)
[2023-07-16 01:58:18,975] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __cons
umer_offsets-42 in 109 milliseconds for epoch 0, of which 109 milliseconds was spent in the scheduler. (kafka.coordinato
r.group.GroupMetadataManager)
[2023-07-16 01:58:18,976] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __cons
umer_offsets-45 in 109 milliseconds for epoch 0, of which 109 milliseconds was spent in the scheduler. (kafka.coordinato
r.group.GroupMetadataManager)
[2023-07-16 01:58:18,978] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __cons
umer_offsets-48 in 110 milliseconds for epoch 0, of which 109 milliseconds was spent in the scheduler. (kafka.coordinato
r.group.GroupMetadataManager)
```

```
per.server.ZooKeeperServer)
[2023-07-16 01:56:42,638] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-16 01:56:47,595] INFO Server environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, bui
lt on 04/08/2021 16:35 GMT (org.apache.zookeeper.server.ZooKeeperServer)
```

The above screenshots show, we successfully ran the Kafka server.

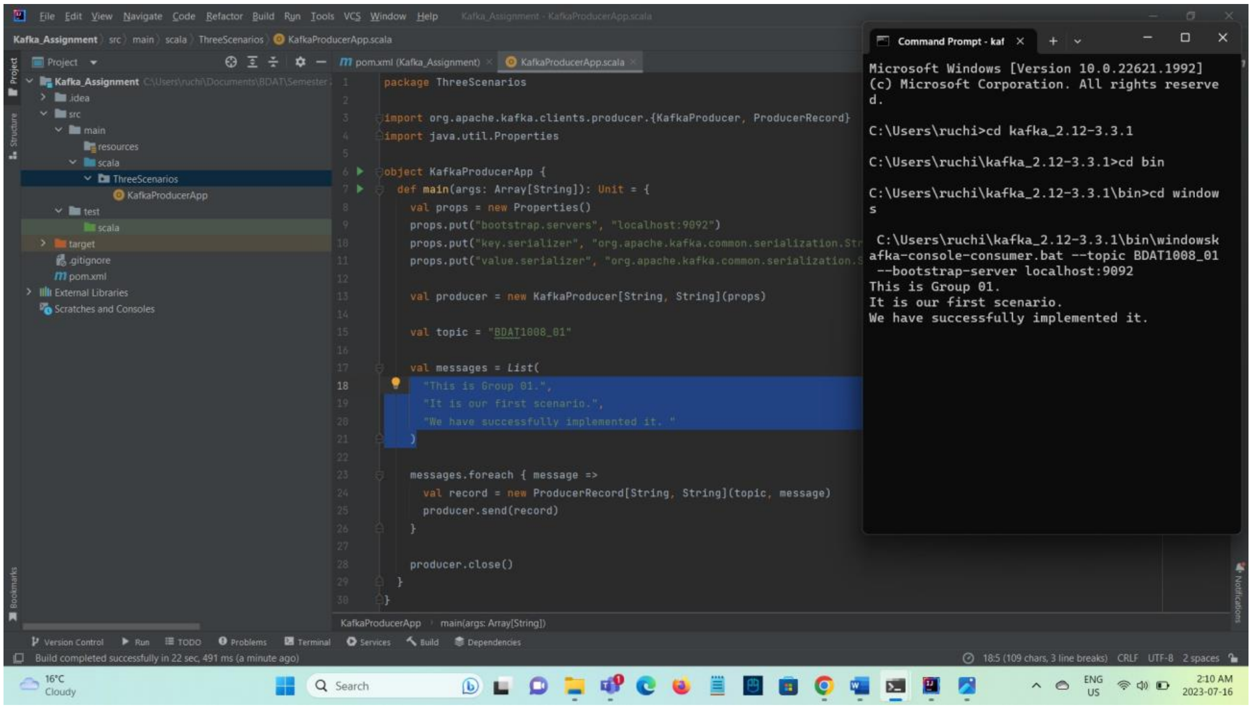
4. Creation of Topic



```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ruchi>cd kafka_2.12-3.3.1

C:\Users\ruchi\kafka_2.12-3.3.1>cd bin

C:\Users\ruchi\kafka_2.12-3.3.1\bin>cd windows

C:\Users\ruchi\kafka_2.12-3.3.1\bin\windows>kafka-topics.bat --create --topic BDAT1008 --bootstrap-server localhost:9092 --partitions 2 --replication-factor
 1
Error while executing topic command : Topic 'BDAT1008' already exists.
[2023-07-16 02:02:52,472] ERROR org.apache.kafka.common.errors.TopicExistsException: Topic 'BDAT1008' already exists.
 (kafka.admin.TopicCommand$)

C:\Users\ruchi\kafka_2.12-3.3.1\bin\windows>kafka-topics.bat --create --topic BDAT1008_01 --bootstrap-server localhost:9092 --partitions 2 --replication-fac
tor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not
 both.
Created topic BDAT1008_01.

C:\Users\ruchi\kafka_2.12-3.3.1\bin\windows>kafka-topics.bat --list --bootstrap-server localhost:9092
BDAT1008
BDAT1008_01
__consumer_offsets

C:\Users\ruchi\kafka_2.12-3.3.1\bin\windows>
```

The above screenshot shows the creation of the topic. We created our topic BDAT1008_01 by running the command “bin/windows>kafka-topic.bat --create --topic my-BDAT1008_01 --bootstrap-server localhost:9092 --partitions 2 --replication-factor ”.

5. First Scenario: Console Consumer / App Producer

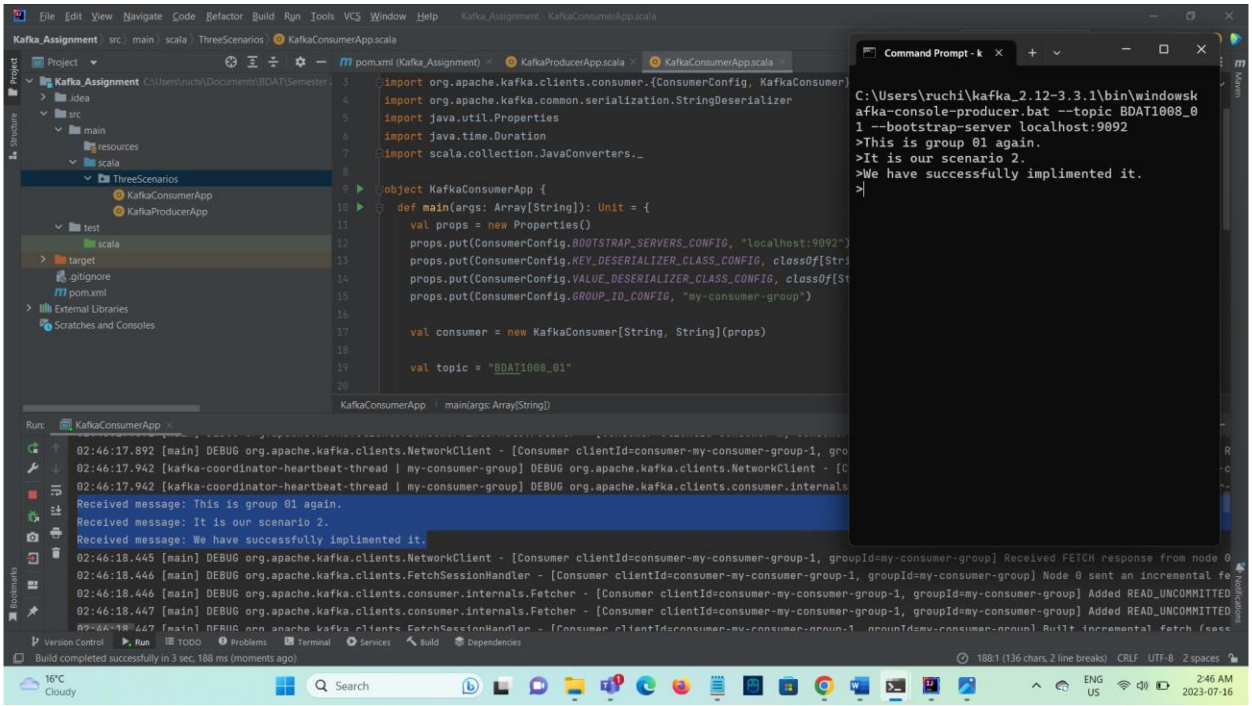


```
package ThreeScenarios

import org.apache.kafka.clients.producer.{KafkaProducer, ProducerRecord}
import java.util.Properties

object KafkaProducerApp {
  def main(args: Array[String]): Unit = {
    val props = new Properties()
    props.put("bootstrap.servers", "localhost:9092")
    props.put("key.serializer", "org.apache.kafka.common.serialization.Str
    props.put("value.serializer", "org.apache.kafka.common.serialization.S

    val producer = new KafkaProducer[String, String](props)

    val topic = "BDAT1008_01"

    val messages = List(
      "This is Group 01.",
      "It is our first scenario.",
      "We have successfully implemented it. "
    )

    messages.foreach { message =>
      val record = new ProducerRecord[String, String](topic, message)
      producer.send(record)
    }

    producer.close()
  }
}
```

```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserve
d.

C:\Users\ruchi>cd kafka_2.12-3.3.1

C:\Users\ruchi\kafka_2.12-3.3.1>cd bin

C:\Users\ruchi\kafka_2.12-3.3.1\bin>cd window
s

 C:\Users\ruchi\kafka_2.12-3.3.1\bin\windowsk
afka-console-consumer.bat --topic BDAT1008_01
 --bootstrap-server localhost:9092
This is Group 01.
It is our first scenario.
We have successfully implemented it.
```

In our first scenario, we created Kafka Producer application on IDE using Scala code to produce messages and started Kafka Console Consumer on the console to listen / consume those messages by making connection using bootstrap server: localhost and port 9092.

6. Second Scenario: Console Producer / App Consumer



```
import org.apache.kafka.clients.consumer.{ConsumerConfig, KafkaConsumer}
import org.apache.kafka.common.serialization.StringDeserializer
import java.util.Properties
import java.time.Duration
import scala.collection.JavaConverters._

object KafkaConsumerApp {
  def main(args: Array[String]): Unit = {
    val props = new Properties()
    props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092")
    props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, classOf[Stri
    props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, classOf[St
    props.put(ConsumerConfig.GROUP_ID_CONFIG, "my-consumer-group")

    val consumer = new KafkaConsumer[String, String](props)

    val topic = "BDAT1008_01"

```

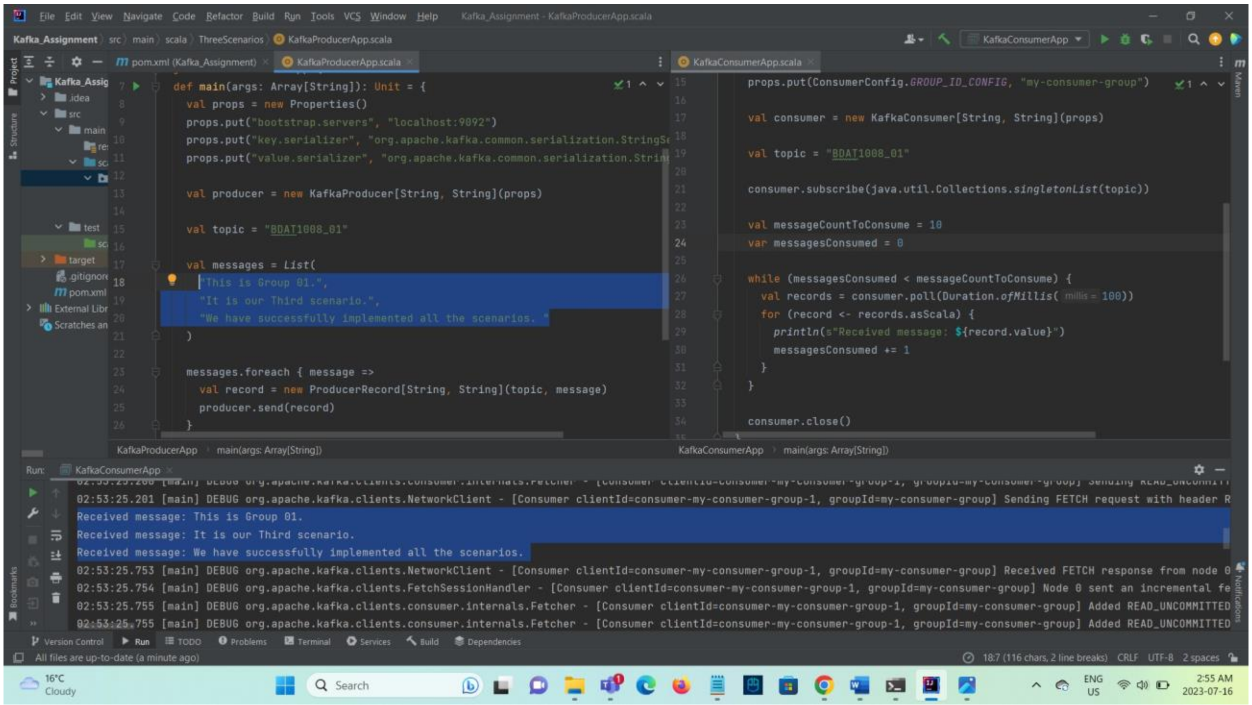
```
C:\Users\ruchi\kafka_2.12-3.3.1\bin\windowsk
afka-console-producer.bat --topic BDAT1008_0
1 --bootstrap-server localhost:9092
>This is group 01 again.
>It is our scenario 2.
>We have successfully implimented it.
>
```

```
02:46:17.892 [main] DEBUG org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-my-consumer-group-1, gro
02:46:17.942 [kafka-coordinator-heartbeat-thread | my-consumer-group] DEBUG org.apache.kafka.clients.NetworkClient - [C
02:46:17.942 [kafka-coordinator-heartbeat-thread | my-consumer-group] DEBUG org.apache.kafka.clients.consumer.internals
Received message: This is group 01 again.
Received message: It is our scenario 2.
Received message: We have successfully implimented it.
02:46:18.445 [main] DEBUG org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Received FETCH response from node 0
02:46:18.446 [main] DEBUG org.apache.kafka.clients.FetchSessionHandler - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Node 0 sent an incremental fe
02:46:18.446 [main] DEBUG org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Added READ_UNCOMMITTED
02:46:18.447 [main] DEBUG org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Added READ_UNCOMMITTED
02:46:18.447 [main] DEBUG org.apache.kafka.clients.FetchSessionHandler - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Built incremental fetch (sess
```

Build completed successfully in 3 sec, 188 ms (moments ago)

In our second scenario, we created Kafka Consumer application on IDE using Scala code to listen / consume messages produced by Kafka Console Producer on the console with the help of connection bootstrap server localhost and port 9092.

7. Third Scenario: App Consumer / App Producer



```
def main(args: Array[String]): Unit = {
  val props = new Properties()
  props.put("bootstrap.servers", "localhost:9092")
  props.put("key.serializer", "org.apache.kafka.common.serialization.StringSe
  props.put("value.serializer", "org.apache.kafka.common.serialization.Strin

  val producer = new KafkaProducer[String, String](props)

  val topic = "BDAT1008_01"

  val messages = List(
    "This is Group 01.",
    "It is our Third scenario.",
    "We have successfully implemented all the scenarios. "
  )

  messages.foreach { message =>
    val record = new ProducerRecord[String, String](topic, message)
    producer.send(record)
  }
```

```
  props.put(ConsumerConfig.GROUP_ID_CONFIG, "my-consumer-group")

  val consumer = new KafkaConsumer[String, String](props)

  val topic = "BDAT1008_01"

  consumer.subscribe(java.util.Collections.singletonList(topic))

  val messageCountToConsume = 10
  var messagesConsumed = 0

  while (messagesConsumed < messageCountToConsume) {
    val records = consumer.poll(Duration.ofMillis( millis = 100))
    for (record <- records.asScala) {
      println(s"Received message: ${record.value}")
      messagesConsumed += 1
    }
  }

  consumer.close()
```

```
02:53:25.201 [main] DEBUG org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Sending FETCH request with header R
Received message: This is Group 01.
Received message: It is our Third scenario.
Received message: We have successfully implemented all the scenarios.
02:53:25.753 [main] DEBUG org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Received FETCH response from node 0
02:53:25.754 [main] DEBUG org.apache.kafka.clients.FetchSessionHandler - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Node 0 sent an incremental fe
02:53:25.755 [main] DEBUG org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Added READ_UNCOMMITTED
02:53:25.755 [main] DEBUG org.apache.kafka.clients.consumer.internals.Fetcher - [Consumer clientId=consumer-my-consumer-group-1, groupId=my-consumer-group] Added READ_UNCOMMITTED
```

In our third scenario, we created both Kafka Producer Application and Kafka Consumer Application on the IDE using Scala code. Further by establishing connection between both of them using bootstrap server localhost and port 9092, we able to produce and consume messages on the IDE itself.

Achievement

We accomplished an implementation of three different scenarios of Kafka consumer and Kafka producer console/application.

1. Console Consumer and Producer apps exchange messages seamlessly.
2. Console Producer generates messages, consumed by a Scala-based app.
3. Producer and Consumer apps communicate flawlessly, enhancing data flow.