ASSIGNMENT 4

Implementing Kafka Messaging Scenarios using Scala

GROUP 7

Deema Sami Barham Al Dogom - 200548717

Rimi Mondal - 200543066

Navid Saleh Sadik - 200544271

Bhavesh Natwarlal Waghela - 200532173

Georgian College

Data Collection and Curation BDAT 1008 01

Table of Contents

Description of the Project

The goal of this project is to demonstrate various Kafka messaging scenarios using Scala programming language. Kafka is a distributed event streaming platform that enables the exchange of real-time data between multiple applications and systems. The project explores three distinct scenarios, each showcasing different aspects of Kafka's capabilities.

Scenario 1: Console Consumer / App Producer

In this scenario, the focus is on creating a Kafka consumer that listens for incoming messages in the console while an independent Scala application acts as the producer. The steps involve setting up Kafka locally, running Zookeeper and Kafka server, and creating a Kafka topic for message exchange. The Scala code for the Kafka Initiator and Producer is written within the application, allowing it to send messages to the Kafka topic. The Console Consumer and Initiator code then receive and display the messages produced by the Scala application.

Steps Involve in the Scenario:

- Install Kafka locally on the machine.
- Start Zookeeper locally to manage Kafka brokers.
- Start Kafka server on the local machine.
- Create a Kafka topic to exchange messages.
- Run the Kafka Consumer to listen for incoming messages.
- Write Scala code for the Kafka Initiator and Producer from the App to produce messages to the Kafka topic.
- Implement the Console Consumer and Initiator code to consume and display the messages from Kafka.

Scenario 2: Console Producer / App Consumer

This scenario reverses the roles of the producer and consumer from Scenario 1. Here, the Kafka Console Producer is utilized to send messages from the command line to the Kafka topic. On the other hand, a separate Scala application serves as the consumer, processing the messages received from the Kafka topic using the Kafka App Consumer and Initiator code.

Steps Involve in the Scenario:

- Ensure Kafka, Zookeeper, and the topic are already set up as mentioned in the previous steps.
- Run the Kafka Console Producer to send messages to the Kafka topic from the command line.
- Write Scala code for the App Consumer and Initiator to consume messages from the Kafka topic and process them within the Scala application.

Scenario 3: App Producer / App Consumer

The final scenario aims to integrate the producer and consumer within the Scala application itself. The project demonstrates how Scala code can be used to act both as a producer, generating messages, and as a consumer, receiving and processing messages from the Kafka topic. This scenario showcases the versatility of Kafka, allowing applications to seamlessly communicate with each other via message exchange.

Steps Involve in the Scenario:

- Continue with the previously installed and set up Kafka environment.
- Develop Scala code for the App Producer to produce messages within the Scala application and send them to the Kafka topic.
- Implement Scala code for the App Consumer and Initiator to consume messages from the Kafka topic and process them within the Scala application.

Throughout the project, Kafka is set up locally, and the Scala code is written to utilize the Kafka libraries and APIs effectively. It includes necessary configurations to connect to Kafka brokers, define topics, and enable message exchange between the producer and consumer components.

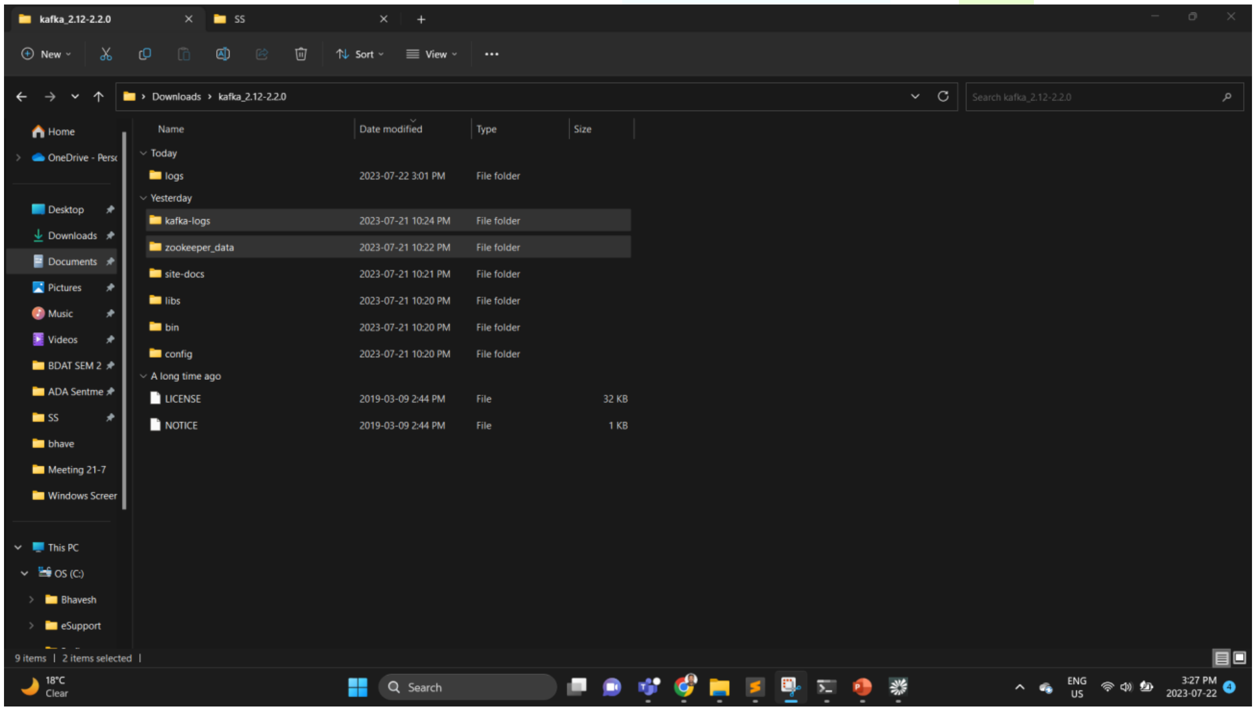
Illustrating Our Work: A Detailed Explanation with Screenshots

By completing all these steps and integrating the various components, the project demonstrates different Kafka messaging scenarios using Scala code, highlighting the versatility and real-time data exchange capabilities of the Kafka messaging system.

Step-by-Step Explanation for Implementing Kafka Messaging Scenarios using Scala Project:

Step 1: Installing Kafka locally

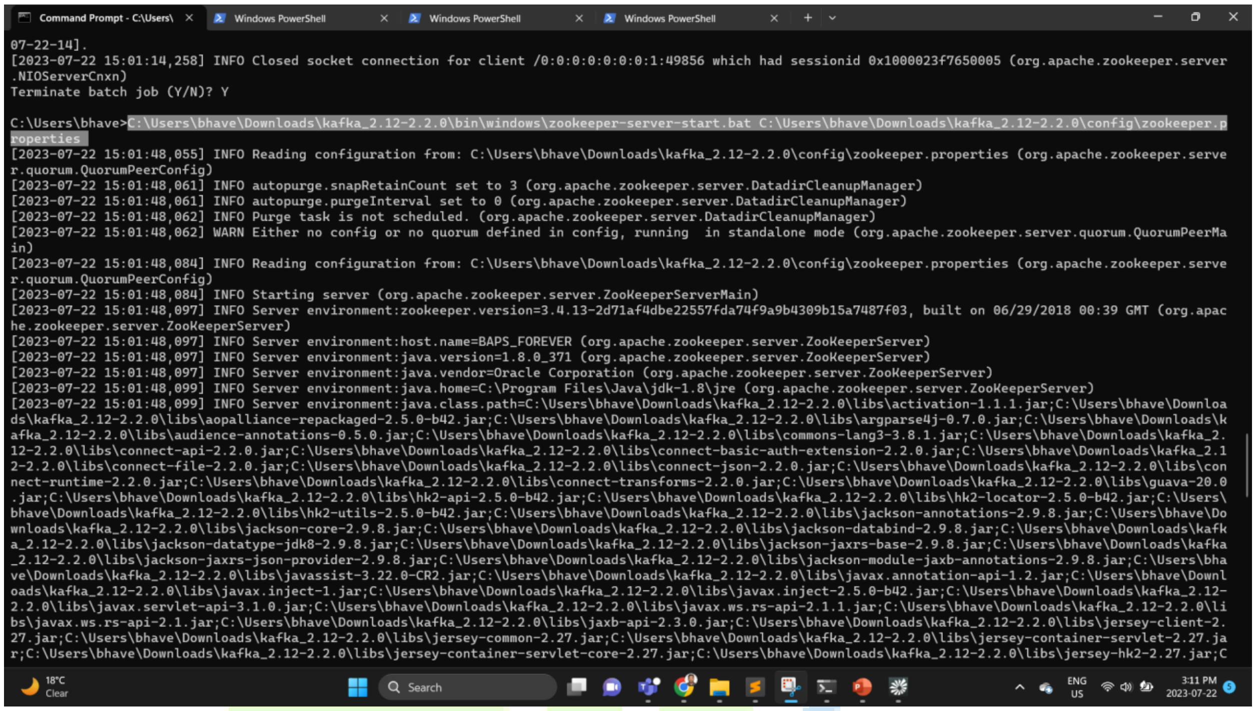
This step involves downloading and installing Apache Kafka on the local development machine. Kafka is open-source software, and it can be downloaded from the Apache Kafka website or other repositories. After installation, the Kafka binaries, libraries, and command-line tools are available for use.



```
Name                 Date modified         Type          Size
Today
  logs               2023-07-22 3:01 PM    File folder
Yesterday
  kafka-logs         2023-07-21 10:24 PM   File folder
  zookeeper_data     2023-07-21 10:22 PM   File folder
  site-docs          2023-07-21 10:21 PM   File folder
  libs               2023-07-21 10:20 PM   File folder
  bin                2023-07-21 10:20 PM   File folder
  config             2023-07-21 10:20 PM   File folder
A long time ago
  LICENSE            2019-03-09 2:44 PM    File          32 KB
  NOTICE             2019-03-09 2:44 PM    File           1 KB
```

Step 2: Running Zookeeper on the local machine

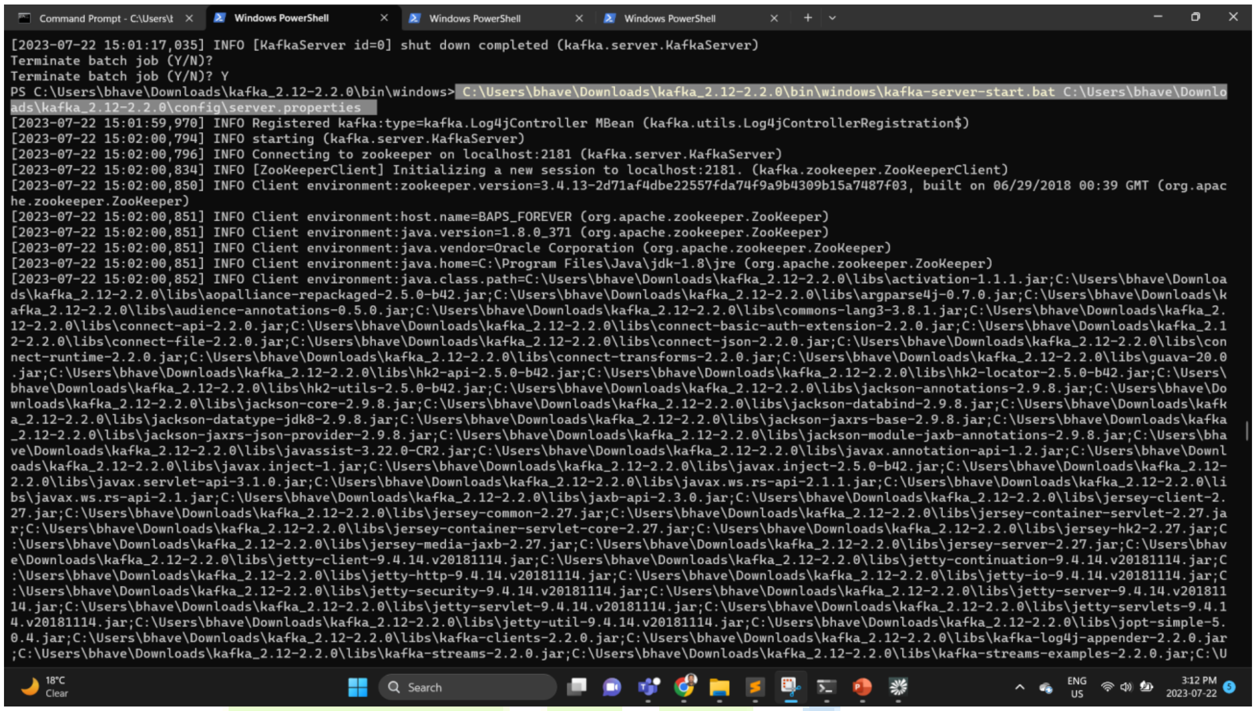
Apache Zookeeper is a centralized service used to manage configuration information, synchronization, and coordination across distributed systems like Kafka. Kafka relies on Zookeeper for maintaining metadata, cluster state, and configuration details. In this step, Zookeeper needs to be started and running on the local machine to support Kafka's operation.



```
07-22-14].
[2023-07-22 15:01:14,258] INFO Closed socket connection for client /0:0:0:0:0:0:0:1:49856 which had sessionid 0x1000023f7650005 (org.apache.zookeeper.server
.NIOServerCnxn)
Terminate batch job (Y/N)? Y

C:\Users\bhave>C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\zookeeper-server-start.bat C:\Users\bhave\Downloads\kafka_2.12-2.2.0\config\zookeeper.p
roperties
[2023-07-22 15:01:48,055] INFO Reading configuration from: C:\Users\bhave\Downloads\kafka_2.12-2.2.0\config\zookeeper.properties (org.apache.zookeeper.serve
r.quorum.QuorumPeerConfig)
[2023-07-22 15:01:48,061] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2023-07-22 15:01:48,061] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2023-07-22 15:01:48,062] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2023-07-22 15:01:48,062] WARN Either no config or no quorum defined in config, running  in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMa
in)
[2023-07-22 15:01:48,084] INFO Reading configuration from: C:\Users\bhave\Downloads\kafka_2.12-2.2.0\config\zookeeper.properties (org.apache.zookeeper.serve
r.quorum.QuorumPeerConfig)
[2023-07-22 15:01:48,084] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2023-07-22 15:01:48,097] INFO Server environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 00:39 GMT (org.apac
he.zookeeper.server.ZooKeeperServer)
[2023-07-22 15:01:48,097] INFO Server environment:host.name=BAPS_FOREVER (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-22 15:01:48,097] INFO Server environment:java.version=1.8.0_371 (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-22 15:01:48,097] INFO Server environment:java.vendor=Oracle Corporation (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-22 15:01:48,099] INFO Server environment:java.home=C:\Program Files\Java\jdk-1.8\jre (org.apache.zookeeper.server.ZooKeeperServer)
[2023-07-22 15:01:48,099] INFO Server environment:java.class.path=C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\activation-1.1.1.jar;C:\Users\bhave\Downloa
ds\kafka_2.12-2.2.0\libs\aopalliance-repackaged-2.5.0-b42.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\argparse4j-0.7.0.jar;C:\Users\bhave\Downloads\k
afka_2.12-2.2.0\libs\audience-annotations-0.5.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\commons-lang3-3.8.1.jar;C:\Users\bhave\Downloads\kafka_2.
12-2.2.0\libs\connect-api-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\connect-basic-auth-extension-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.1
2-2.2.0\libs\connect-file-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\connect-json-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\con
nect-runtime-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\connect-transforms-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\guava-20.0
.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\hk2-api-2.5.0-b42.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\hk2-locator-2.5.0-b42.jar;C:\Users\
bhave\Downloads\kafka_2.12-2.2.0\libs\hk2-utils-2.5.0-b42.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jackson-annotations-2.9.8.jar;C:\Users\bhave\Do
wnloads\kafka_2.12-2.2.0\libs\jackson-core-2.9.8.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jackson-databind-2.9.8.jar;C:\Users\bhave\Downloads\kafk
a_2.12-2.2.0\libs\jackson-datatype-jdk8-2.9.8.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jackson-jaxrs-base-2.9.8.jar;C:\Users\bhave\Downloads\kafka
_2.12-2.2.0\libs\jackson-jaxrs-json-provider-2.9.8.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jackson-module-jaxb-annotations-2.9.8.jar;C:\Users\bha
ve\Downloads\kafka_2.12-2.2.0\libs\javassist-3.22.0-CR2.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\javax.annotation-api-1.2.jar;C:\Users\bhave\Downl
oads\kafka_2.12-2.2.0\libs\javax.inject-1.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\javax.inject-2.5.0-b42.jar;C:\Users\bhave\Downloads\kafka_2.12-
2.2.0\libs\javax.servlet-api-3.1.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\javax.ws.rs-api-2.1.1.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\li
bs\javax.ws.rs-api-2.1.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jaxb-api-2.3.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-client-2.
27.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-common-2.27.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-container-servlet-2.27.ja
r;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-container-servlet-core-2.27.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-hk2-2.27.jar;C
```

Step 3: Starting Kafka Server on the local machine

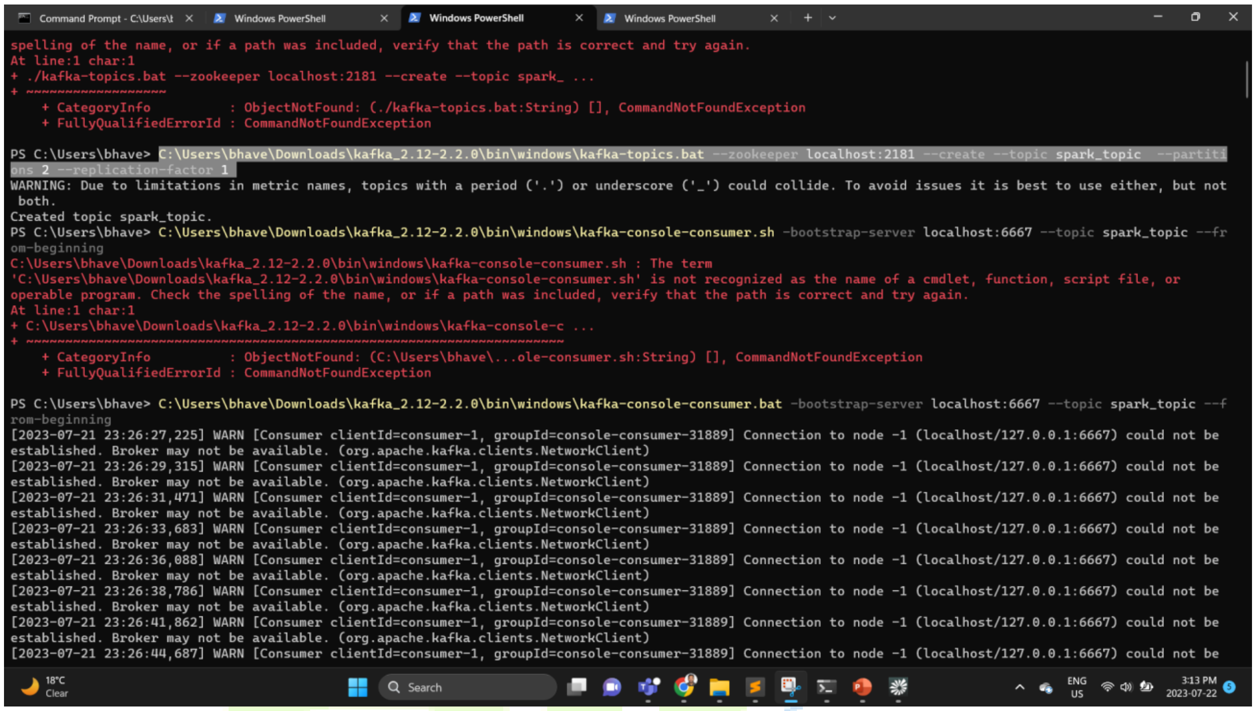
With Zookeeper running, the next step is to start the Kafka server on the local machine. The Kafka server, also known as a Kafka broker, is responsible for handling incoming messages, managing topics and partitions, and coordinating with other brokers in a Kafka cluster. It connects to the running Zookeeper instance to register itself and become part of the Kafka cluster.



```
[2023-07-22 15:01:17,035] INFO [KafkaServer id=0] shut down completed (kafka.server.KafkaServer)
Terminate batch job (Y/N)?
Terminate batch job (Y/N)? Y
PS C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows> C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-server-start.bat C:\Users\bhave\Downlo
ads\kafka_2.12-2.2.0\config\server.properties
[2023-07-22 15:01:59,970] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2023-07-22 15:02:00,794] INFO starting (kafka.server.KafkaServer)
[2023-07-22 15:02:00,796] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2023-07-22 15:02:00,834] INFO [ZooKeeperClient] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2023-07-22 15:02:00,850] INFO Client environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 00:39 GMT (org.apac
he.zookeeper.ZooKeeper)
[2023-07-22 15:02:00,851] INFO Client environment:host.name=BAPS_FOREVER (org.apache.zookeeper.ZooKeeper)
[2023-07-22 15:02:00,851] INFO Client environment:java.version=1.8.0_371 (org.apache.zookeeper.ZooKeeper)
[2023-07-22 15:02:00,851] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2023-07-22 15:02:00,851] INFO Client environment:java.home=C:\Program Files\Java\jdk-1.8\jre (org.apache.zookeeper.ZooKeeper)
[2023-07-22 15:02:00,852] INFO Client environment:java.class.path=C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\activation-1.1.1.jar;C:\Users\bhave\Downloa
ds\kafka_2.12-2.2.0\libs\aopalliance-repackaged-2.5.0-b42.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\argparse4j-0.7.0.jar;C:\Users\bhave\Downloads\k
afka_2.12-2.2.0\libs\audience-annotations-0.5.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\commons-lang3-3.8.1.jar;C:\Users\bhave\Downloads\kafka_2.
12-2.2.0\libs\connect-api-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\connect-basic-auth-extension-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.1
2-2.2.0\libs\connect-file-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\connect-json-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\con
nect-runtime-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\connect-transforms-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\guava-20.0
.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\hk2-api-2.5.0-b42.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\hk2-locator-2.5.0-b42.jar;C:\Users\
bhave\Downloads\kafka_2.12-2.2.0\libs\hk2-utils-2.5.0-b42.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jackson-annotations-2.9.8.jar;C:\Users\bhave\Do
wnloads\kafka_2.12-2.2.0\libs\jackson-core-2.9.8.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jackson-databind-2.9.8.jar;C:\Users\bhave\Downloads\kafk
a_2.12-2.2.0\libs\jackson-datatype-jdk8-2.9.8.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jackson-jaxrs-base-2.9.8.jar;C:\Users\bhave\Downloads\kafka
_2.12-2.2.0\libs\jackson-jaxrs-json-provider-2.9.8.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jackson-module-jaxb-annotations-2.9.8.jar;C:\Users\bha
ve\Downloads\kafka_2.12-2.2.0\libs\javassist-3.22.0-CR2.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\javax.annotation-api-1.2.jar;C:\Users\bhave\Downl
oads\kafka_2.12-2.2.0\libs\javax.inject-1.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\javax.inject-2.5.0-b42.jar;C:\Users\bhave\Downloads\kafka_2.12-
2.2.0\libs\javax.servlet-api-3.1.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\javax.ws.rs-api-2.1.1.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\li
bs\javax.ws.rs-api-2.1.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jaxb-api-2.3.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-client-2.
27.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-common-2.27.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-container-servlet-2.27.ja
r;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-container-servlet-core-2.27.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-hk2-2.27.jar;C
:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-media-jaxb-2.27.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jersey-server-2.27.jar;C:\Users\bhav
e\Downloads\kafka_2.12-2.2.0\libs\jetty-client-9.4.14.v20181114.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jetty-continuation-9.4.14.v20181114.jar;C
:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jetty-http-9.4.14.v20181114.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jetty-io-9.4.14.v20181114.jar;C
:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jetty-security-9.4.14.v20181114.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jetty-server-9.4.14.v201811
14.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jetty-servlet-9.4.14.v20181114.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jetty-servlets-9.4.1
4.v20181114.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jetty-util-9.4.14.v20181114.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\jopt-simple-5.
0.4.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\kafka-clients-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\kafka-log4j-appender-2.2.0.jar
;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\kafka-streams-2.2.0.jar;C:\Users\bhave\Downloads\kafka_2.12-2.2.0\libs\kafka-streams-examples-2.2.0.jar;C:\U
```

Step 4: Creating a Kafka topic

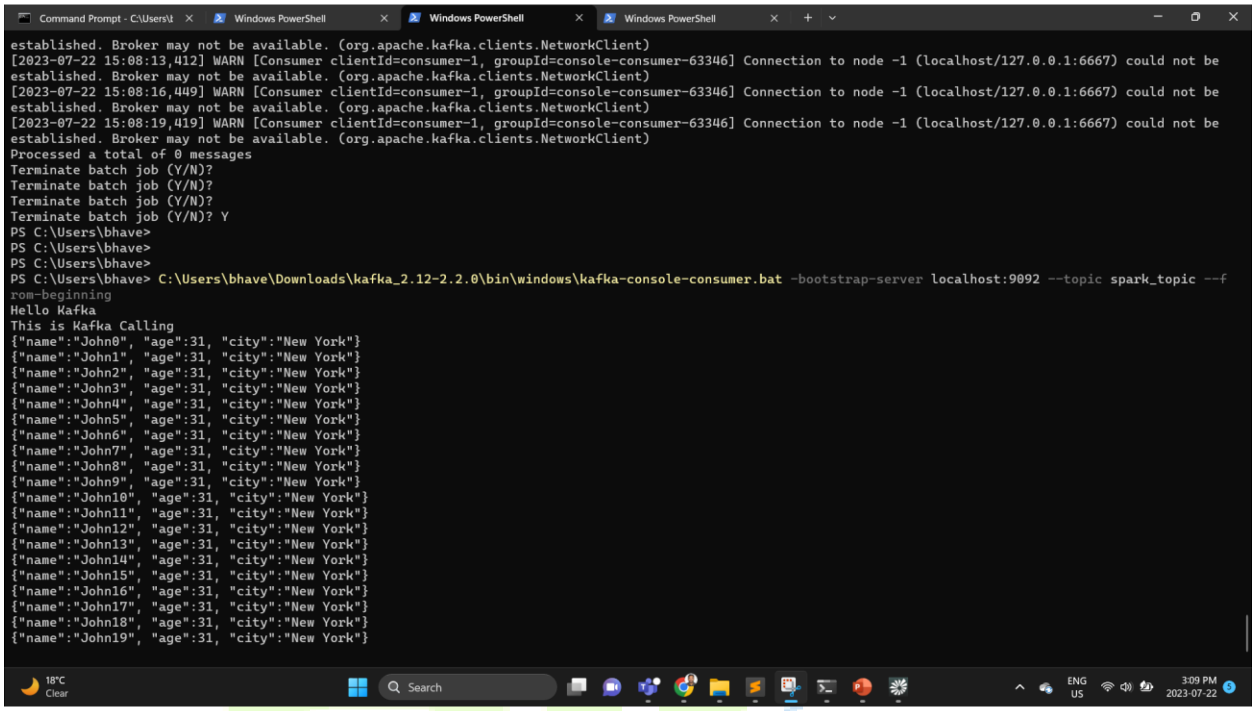
A Kafka topic is a logical channel or category to which messages are published by producers and consumed by consumers. Before the messaging can take place, a topic needs to be created on the Kafka cluster. Topics can have multiple partitions to enable parallel message processing. The configuration details like the number of partitions, replication factor, and other settings can be specified during topic creation.



```
spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ ./kafka-topics.bat --zookeeper localhost:2181 --create --topic spark_ ...
+ ~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (./kafka-topics.bat:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\bhave> C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-topics.bat --zookeeper localhost:2181 --create --topic spark_topic  --partiti
ons 2 --replication-factor 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 spark_topic.
PS C:\Users\bhave> C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-console-consumer.sh -bootstrap-server localhost:6667 --topic spark_topic --fr
om-beginning
C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-console-consumer.sh : The term
'C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-console-consumer.sh' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-console-c ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:\Users\bhave\...ole-consumer.sh:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\bhave> C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-console-consumer.bat -bootstrap-server localhost:6667 --topic spark_topic --f
rom-beginning
[2023-07-21 23:26:27,225] WARN [Consumer clientId=consumer-1, groupId=console-consumer-31889] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-21 23:26:29,315] WARN [Consumer clientId=consumer-1, groupId=console-consumer-31889] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-21 23:26:31,471] WARN [Consumer clientId=consumer-1, groupId=console-consumer-31889] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-21 23:26:33,683] WARN [Consumer clientId=consumer-1, groupId=console-consumer-31889] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-21 23:26:36,088] WARN [Consumer clientId=consumer-1, groupId=console-consumer-31889] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-21 23:26:38,786] WARN [Consumer clientId=consumer-1, groupId=console-consumer-31889] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-21 23:26:41,862] WARN [Consumer clientId=consumer-1, groupId=console-consumer-31889] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-21 23:26:44,687] WARN [Consumer clientId=consumer-1, groupId=console-consumer-31889] Connection to node -1 (localhost/127.0.0.1:6667) could not be
```

Step 5: Running Kafka Consumer

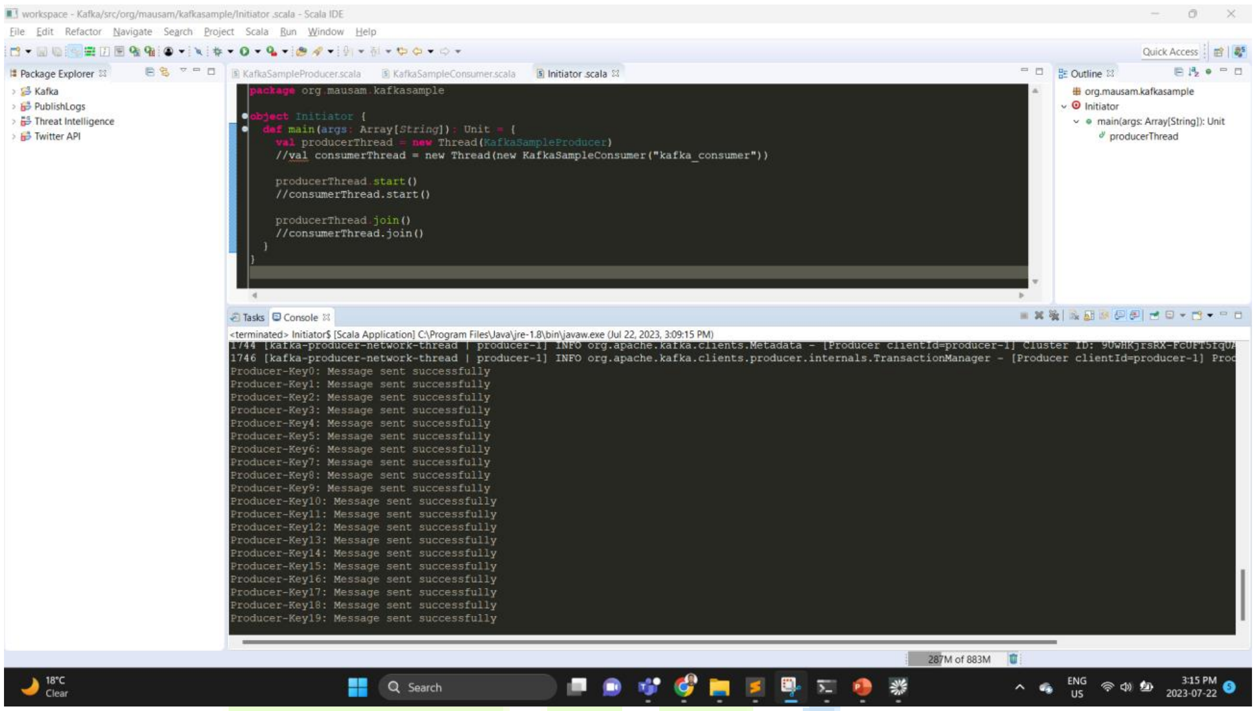
To consume messages from a Kafka topic, a Kafka consumer needs to be running. The consumer subscribes to one or more topics and continuously polls the Kafka broker for new messages. Once a message is received, the consumer can process and utilize the data as required. In this project, it is essential to have a consumer up and running to receive and display messages sent by producers.



```
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:08:13,412] WARN [Consumer clientId=consumer-1, groupId=console-consumer-63346] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:08:16,449] WARN [Consumer clientId=consumer-1, groupId=console-consumer-63346] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:08:19,419] WARN [Consumer clientId=consumer-1, groupId=console-consumer-63346] Connection to node -1 (localhost/127.0.0.1:6667) could not be
established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
Processed a total of 0 messages
Terminate batch job (Y/N)?
Terminate batch job (Y/N)?
Terminate batch job (Y/N)?
Terminate batch job (Y/N)? Y
PS C:\Users\bhave>
PS C:\Users\bhave>
PS C:\Users\bhave>
PS C:\Users\bhave> C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-console-consumer.bat -bootstrap-server localhost:9092 --topic spark_topic --f
rom-beginning
Hello Kafka
This is Kafka Calling
{"name":"John0", "age":31, "city":"New York"}
{"name":"John1", "age":31, "city":"New York"}
{"name":"John2", "age":31, "city":"New York"}
{"name":"John3", "age":31, "city":"New York"}
{"name":"John4", "age":31, "city":"New York"}
{"name":"John5", "age":31, "city":"New York"}
{"name":"John6", "age":31, "city":"New York"}
{"name":"John7", "age":31, "city":"New York"}
{"name":"John8", "age":31, "city":"New York"}
{"name":"John9", "age":31, "city":"New York"}
{"name":"John10", "age":31, "city":"New York"}
{"name":"John11", "age":31, "city":"New York"}
{"name":"John12", "age":31, "city":"New York"}
{"name":"John13", "age":31, "city":"New York"}
{"name":"John14", "age":31, "city":"New York"}
{"name":"John15", "age":31, "city":"New York"}
{"name":"John16", "age":31, "city":"New York"}
{"name":"John17", "age":31, "city":"New York"}
{"name":"John18", "age":31, "city":"New York"}
{"name":"John19", "age":31, "city":"New York"}
```

Step 6: Implementing Kafka Initiator and Producer in the Scala Application

In this step, the Scala application is developed to act as a Kafka producer. The producer component initiates message generation and sends the messages to the Kafka topic. The Initiator part of the code may handle additional tasks before producing the messages, such as data preparation or aggregation, depending on the specific requirements of the project.



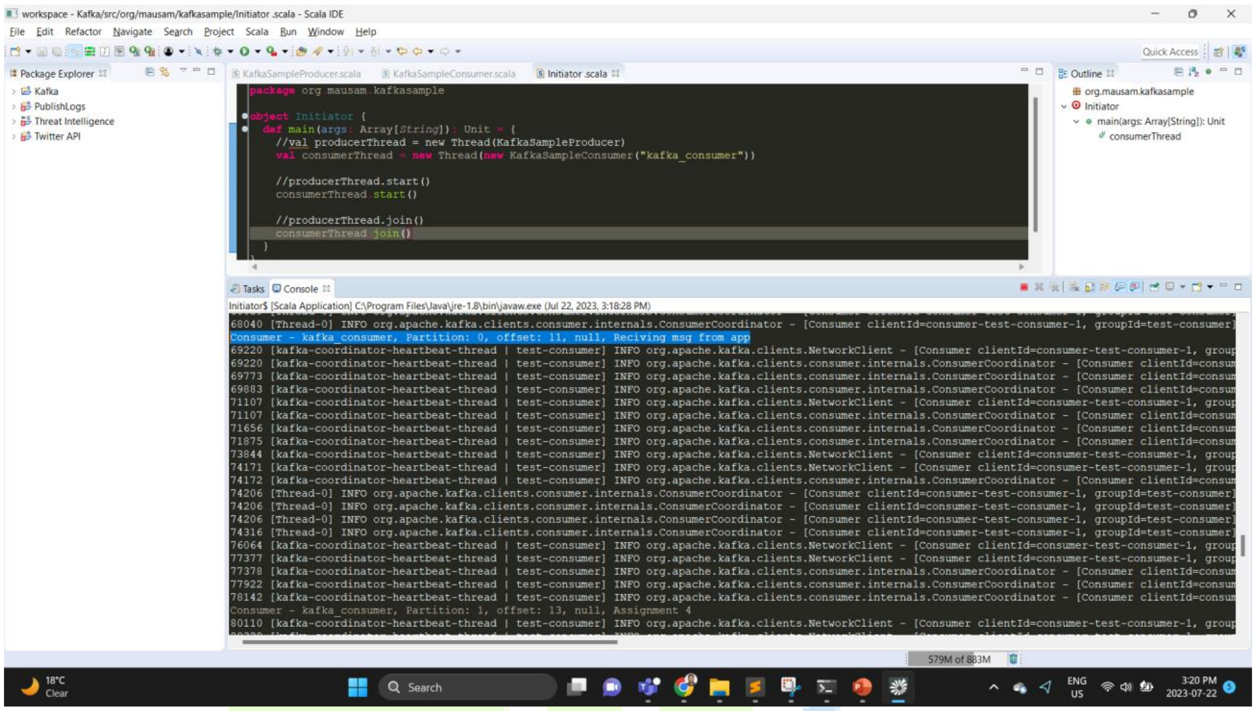
```
package org.mausam.kafkasample

object Initiator {
  def main(args: Array[String]): Unit = {
    val producerThread = new Thread(KafkaSampleProducer)
    //val consumerThread = new Thread(new KafkaSampleConsumer("kafka_consumer"))

    producerThread.start()
    //consumerThread.start()

    producerThread.join()
    //consumerThread.join()
  }
}
```

<terminated> Initiator$ [Scala Application] C:\Program Files\Java\jre-1.8\bin\javaw.exe (Jul 22, 2023, 3:09:15 PM)

```
1744 [kafka-producer-network-thread | producer-1] INFO org.apache.kafka.clients.Metadata - [Producer clientId=producer-1] Cluster ID: 9OWHKjrsRX-FcUFT5fqUA
1746 [kafka-producer-network-thread | producer-1] INFO org.apache.kafka.clients.producer.internals.TransactionManager - [Producer clientId=producer-1] Proc
Producer-Key0: Message sent successfully
Producer-Key1: Message sent successfully
Producer-Key2: Message sent successfully
Producer-Key3: Message sent successfully
Producer-Key4: Message sent successfully
Producer-Key5: Message sent successfully
Producer-Key6: Message sent successfully
Producer-Key7: Message sent successfully
Producer-Key8: Message sent successfully
Producer-Key9: Message sent successfully
Producer-Key10: Message sent successfully
Producer-Key11: Message sent successfully
Producer-Key12: Message sent successfully
Producer-Key13: Message sent successfully
Producer-Key14: Message sent successfully
Producer-Key15: Message sent successfully
Producer-Key16: Message sent successfully
Producer-Key17: Message sent successfully
Producer-Key18: Message sent successfully
Producer-Key19: Message sent successfully
```

Step 7: Writing Console Consumer and Initiator Code

For Scenario 1, the project involves implementing a console consumer that listens for incoming messages from the Kafka topic. The consumer should be able to display the received messages in the console. The Initiator code, as mentioned earlier, might handle some pre-processing tasks before passing the messages to the Kafka producer for publishing.



workspace - Kafka/src/org/mausam/kafkasample/Initiator .scala - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
Package Explorer
Kafka
PublishLogs
Threat Intelligence
Twitter API
KafkaSampleProducer.scala
KafkaSampleConsumer.scala
Initiator .scala
Outline
org.mausam.kafkasample
Initiator
main(args: Array[String]): Unit
consumerThread
Tasks
Console
579M of 883M

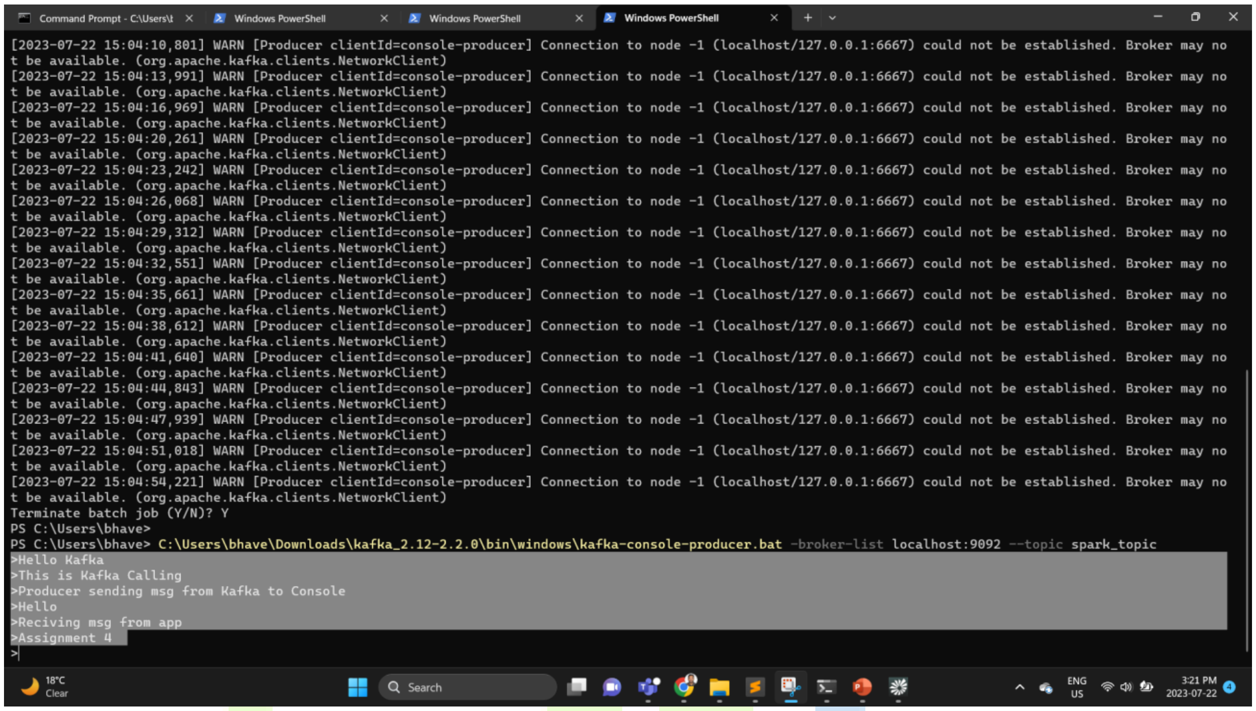
```scala
package org.mausam.kafkasample

object Initiator {
  def main(args: Array[String]): Unit = {
    //val producerThread = new Thread(KafkaSampleProducer)
    val consumerThread = new Thread(new KafkaSampleConsumer("kafka_consumer"))

    //producerThread.start()
    consumerThread.start()

    //producerThread.join()
    consumerThread.join()
  }
}
```

Initiator$ [Scala Application] C:\Program Files\Java\jre-1.8\bin\javaw.exe (Jul 22, 2023, 3:18:28 PM)

```
68040 [Thread-0] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
Consumer - kafka_consumer, Partition: 0, offset: 11, null, Reciving msg from app
69220 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
69220 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
69773 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
69883 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
71107 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
71107 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
71656 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
71875 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
73844 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
74171 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
74172 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
74206 [Thread-0] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
74206 [Thread-0] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
74206 [Thread-0] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
74316 [Thread-0] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
76064 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
77377 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
77378 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
77922 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
78142 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
Consumer - kafka_consumer, Partition: 1, offset: 13, null, Assignment 4
80110 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
```

Step 8: Developing App-level Kafka Producer

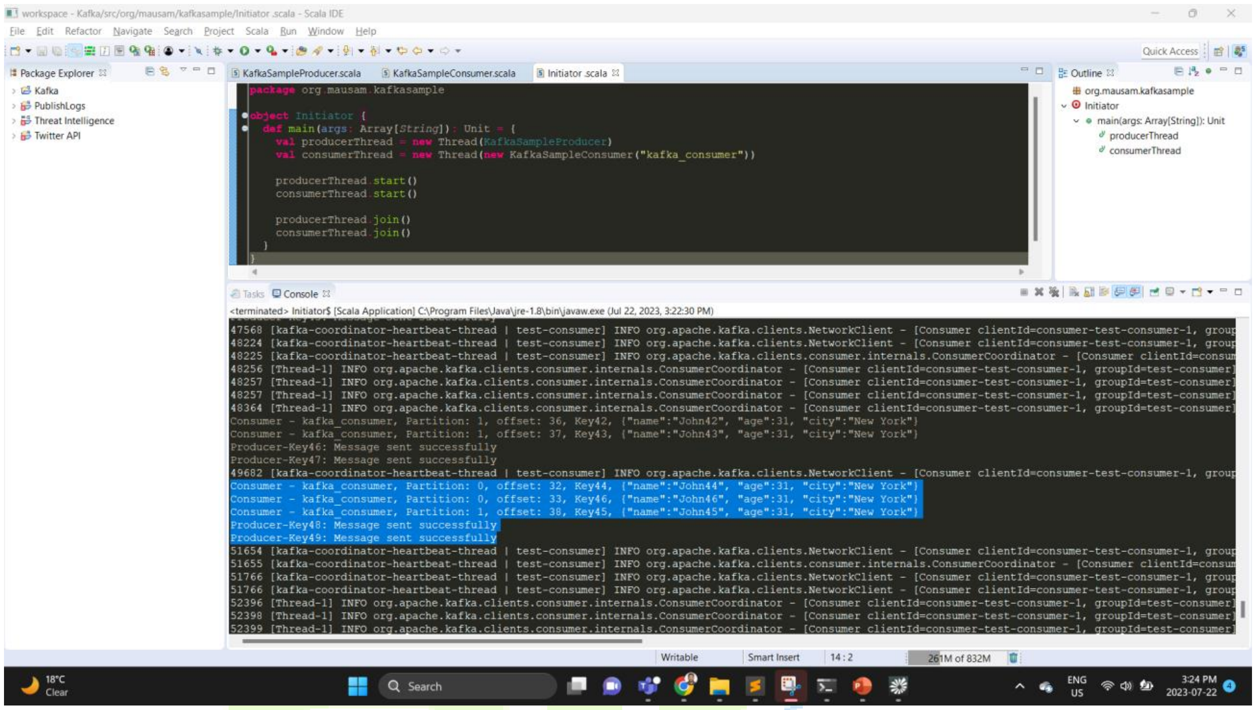
In this step, the Scala application is further developed to act as an app-level Kafka producer. It generates messages within the application and sends them to the Kafka topic. This producer code is distinct from the console producer used in Scenario 2, as it is integrated within the application.



```
[2023-07-22 15:04:10,801] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:13,991] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:16,969] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:20,261] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:23,242] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:26,068] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:29,312] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:32,551] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:35,661] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:38,612] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:41,640] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:44,843] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:47,939] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:51,018] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
[2023-07-22 15:04:54,221] WARN [Producer clientId=console-producer] Connection to node -1 (localhost/127.0.0.1:6667) could not be established. Broker may no
t be available. (org.apache.kafka.clients.NetworkClient)
Terminate batch job (Y/N)? Y
PS C:\Users\bhave>
PS C:\Users\bhave> C:\Users\bhave\Downloads\kafka_2.12-2.2.0\bin\windows\kafka-console-producer.bat -broker-list localhost:9092 --topic spark_topic
>Hello Kafka
>This is Kafka Calling
>Producer sending msg from Kafka to Console
>Hello
>Reciving msg from app
>Assignment 4
>
```

Step 9: Building Scala Application with App Producer and Consumer

For Scenario 3, the project involves developing the Scala application to encompass both the producer and consumer functionalities within itself. The application generates messages as a producer and simultaneously consumes and processes messages from the Kafka topic as a consumer. The Initiator code might still be used to manage the message generation process before publishing them to the Kafka topic.


workspace - Kafka/src/org/mausam/kafkasample/Initiator .scala - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
Package Explorer
Kafka
PublishLogs
Threat Intelligence
Twitter API
KafkaSampleProducer.scala
KafkaSampleConsumer.scala
Initiator .scala
Outline
org.mausam.kafkasample
Initiator
main(args: Array[String]): Unit
producerThread
consumerThread
Tasks
Console
Writable
Smart Insert
14 : 2
261M of 832M

```
package org.mausam.kafkasample

object Initiator {
  def main(args: Array[String]): Unit = {
    val producerThread = new Thread(KafkaSampleProducer)
    val consumerThread = new Thread(new KafkaSampleConsumer("kafka_consumer"))

    producerThread.start()
    consumerThread.start()

    producerThread.join()
    consumerThread.join()
  }
}
```

<terminated> Initiator$ [Scala Application] C:\Program Files\Java\jre-1.8\bin\javaw.exe (Jul 22, 2023, 3:22:30 PM)

```
47568 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
48224 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
48225 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
48256 [Thread-1] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
48257 [Thread-1] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
48257 [Thread-1] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
48364 [Thread-1] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
Consumer - kafka_consumer, Partition: 1, offset: 36, Key42, {"name":"John42", "age":31, "city":"New York"}
Consumer - kafka_consumer, Partition: 1, offset: 37, Key43, {"name":"John43", "age":31, "city":"New York"}
Producer-Key46: Message sent successfully
Producer-Key47: Message sent successfully
49682 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
Consumer - kafka_consumer, Partition: 0, offset: 32, Key44, {"name":"John44", "age":31, "city":"New York"}
Consumer - kafka_consumer, Partition: 0, offset: 33, Key46, {"name":"John46", "age":31, "city":"New York"}
Consumer - kafka_consumer, Partition: 1, offset: 38, Key45, {"name":"John45", "age":31, "city":"New York"}
Producer-Key48: Message sent successfully
Producer-Key49: Message sent successfully
51654 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
51655 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consum
51766 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
51766 [kafka-coordinator-heartbeat-thread | test-consumer] INFO org.apache.kafka.clients.NetworkClient - [Consumer clientId=consumer-test-consumer-1, group
52396 [Thread-1] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
52398 [Thread-1] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
52399 [Thread-1] INFO org.apache.kafka.clients.consumer.internals.ConsumerCoordinator - [Consumer clientId=consumer-test-consumer-1, groupId=test-consumer]
```

Achievement

Through the project "Implementing Kafka Messaging Scenarios using Scala," several valuable learnings have been acquired in working with Kafka and leveraging Scala for real-time data exchange. The project's primary goal was to demonstrate different Kafka messaging scenarios, each shedding light on various aspects of Kafka's capabilities.

In Scenario 1, we learned how to set up Kafka locally by installing it on the machine, starting Zookeeper for managing Kafka brokers, and running the Kafka server. Creating Kafka topics for message exchange was a crucial step, and we successfully developed Scala code for both the Kafka Initiator and Producer within the application. This allowed us to send messages to the Kafka topic and receive and display them using the Console Consumer and Initiator code.

Scenario 2 reversed the roles of producer and consumer, showcasing how to use the Kafka Console Producer to send messages from the command line to the Kafka topic. Meanwhile, a separate Scala application acted as the consumer, processing the messages using the Kafka App Consumer and Initiator code.

The most comprehensive learning came in Scenario 3, where we integrated both the producer and consumer within the Scala application itself. This scenario demonstrated the flexibility of Kafka, enabling seamless communication between components through message exchange.

Throughout the project, we learned to effectively use Kafka libraries and APIs with Scala, gaining proficiency in connecting to Kafka brokers, defining topics, and enabling smooth communication between producer and consumer components.

The step-by-step explanation illustrated the importance of installing Kafka locally, running Zookeeper, starting the Kafka server, and creating Kafka topics to facilitate real-time data exchange. Additionally, we learned to develop Scala code for the App Producer to produce messages and send them to the Kafka topic, while also implementing code for the App Consumer and Initiator to receive and process messages within the Scala application.

Overall, this project has provided valuable insights into the capabilities of Kafka and the power of Scala for building robust and efficient messaging systems. It has been an enriching experience to explore the world of real-time data streaming and messaging architectures, equipping us with essential skills for future endeavors in distributed systems and big data applications.

Project Participation Table

Name of students **who has** participated in the assignment.
Student name: Deema Sami Barham Al Dogom - 200548717
Student name: Rimi Mondal - 200543066
Student name: Navid Saleh Sadik - 200544271
Student name: Bhavesh Natwarlal Waghela - 200532173
Student name:

Name of students **who has not** participated in the assignment.
Student name:
Student name:
Student name:
Student name:
Student name: