**Step 1: Start Zookeeper from the zookeeper directory**

bin/zkServer.sh start

A computer screen with white text

Description automatically generated

**Step 2: Start Kafka from Kafka directory that is from Confluent directory.**

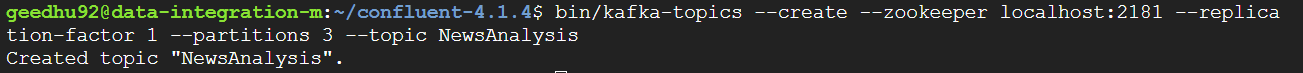
nohup bin/kafka-server-start etc/kafka/server.properties > /dev/null 2>&1 &

A computer screen shot of a program

Description automatically generated

**Step 3: Create the topic “NewsAnalysis” in the confluent directory.**

bin/kafka-topics --create --zookeeper localhost:2181 --replication-factor 1 --partitions 3 --topic NewsAnalysis



**Step 4: Create a consumer to listen to the topic created that is “NewsAnalysis”.**

General command: bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic NewsAnalysis

Here we are running a python code to collect data from the topic.

A screen shot of a computer

Description automatically generated

**Step 5: Run the Producer code.**

A screen shot of a computer

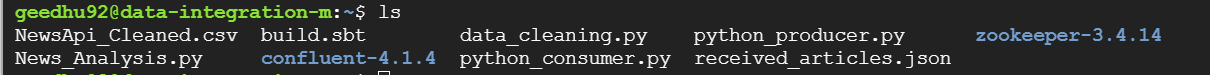
Description automatically generated

**Step 6: Data collected by the Consumer.**

A screen shot of a computer screen

Description automatically generated

**Step 7: Data collected by the consumer is stored in the local directory.**



**Step 8: Start SPARK.**

**spark-shell --master local**

**A computer screen with white text

Description automatically generated**

**Step 9: Importing required SQL functions.**

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

A screen shot of a computer

Description automatically generated

**Step 10: Create a schema for data frame.**

val newsArticleSchema = new StructType(Array(

StructField("source",

StructType(Array(

StructField("id",StringType,true),

StructField("name",StringType,true))),

true),

StructField("author",StringType,true),

StructField("title",StringType,true),

StructField("description",StringType,true),

StructField("url",StringType,true),

StructField("urlToImage",StringType,true),

StructField("publishedAt",StringType,true),

StructField("content",StringType,true)))

A screenshot of a computer

Description automatically generated

**Step 11: Create a streaming data frame.**

val news = spark.readStream.format("json")

.schema(newsArticleSchema)

.option("path", "file:////home/geedhu92/").load()

A black screen with white text

Description automatically generated

**Step 12: Create a writeStream.**

val stream = news.writeStream

.format("json")

.option("checkpointLocation", "file:////home/geedhu92/chkpt")

.outputMode("append")

.option("path", " hdfs:///BigDataNews/")

.start()

A screenshot of a computer

Description automatically generated

**Step 13: Create a new session of scala by running preliminary Codes.**

**spark-shell --master local**

**import org.apache.spark.sql.\_**

**import org.apache.spark.sql.types.\_**

**val newsArticleSchema = new StructType(Array(**

**StructField("source",**

**StructType(Array(**

**StructField("id",StringType,true),**

**StructField("name",StringType,true))),**

**true),**

**StructField("author",StringType,true),**

**StructField("title",StringType,true),**

**StructField("description",StringType,true),**

**StructField("url",StringType,true),**

**StructField("urlToImage",StringType,true),**

**StructField("publishedAt",StringType,true),**

**StructField("content",StringType,true)))**

**A screenshot of a computer program

Description automatically generated**

**Step 14: Read the read stream again**

**val news = spark.readStream.format("json")**

**.schema(newsArticleSchema)**

**.option("path", "file:////home/geedhu92/").load()**

**A black screen with white text

Description automatically generated**

**Step 15: Create the spark session**

**val spark\_hive = SparkSession**

**.builder()**

**.appName("Spark session to save Data for a hive table")**

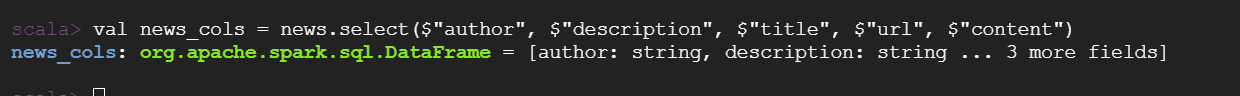
**.getOrCreate()**

**A screen shot of a computer

Description automatically generated**

**Step 16: Selecting a few columns**

**val news\_cols = news.select($"author", $"description", $"title", $"url", $"content")**

****

**Step 17:**

**val stream = news\_cols.writeStream**

**.format("csv")**

**.option("checkpointLocation", "file:////home/geedhu92/chkpt11")**

**.outputMode("append")**

**.option("path", "/BigDataNewsCSV/")**

**.start()**

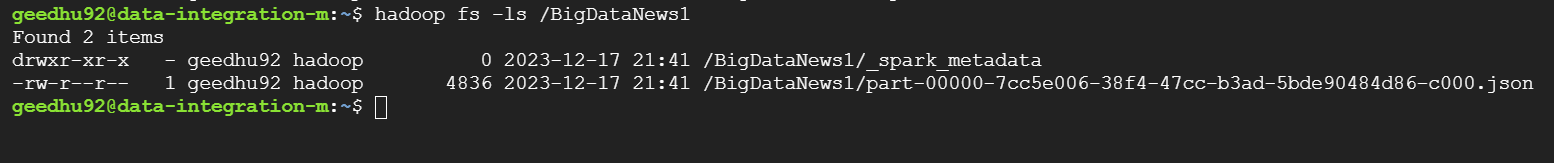
**A screenshot of a computer

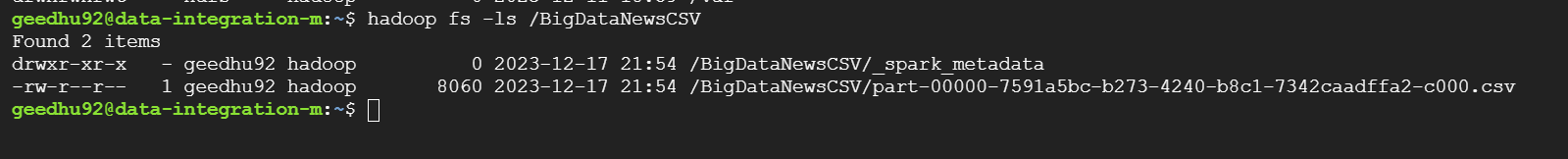
Description automatically generated**

Step 18: Check in Hadoop directory.

A screenshot of a computer

Description automatically generated





**Step 19: Store the data in the hive table.**

**Path where CSV file is stored:**

/BigDataNewsCSV/part-00000-7591a5bc-b273-4240-b8c1-7342caadffa2-c000.csv

**Step 20: Start the spark again and check for the data.**

**spark-shell --master local**

**import org.apache.spark.sql.functions.\_**