

# Project: Kafka + Spark Streaming + PySpark

## **Student :**

Presented by MANICKAM RAVISEKAR ,  
Master of Science in Computer Science, 19599 , Fall Semester 2022

**Professor : Dr Henry Chung**  
**TA : Liang**

SAN FRANCISCO BAY  
UNIVERSITY  
47671 WestingHouse Dr.,  
Fremont, CA 94539

## ACKNOWLEDGEMENT

One of our master's degree Project for Spark Streaming using Scala , Pyspark and Kafka streaming, Is an Interesting, which made me to learn new things, it is useful in designing and applying using Scala, Pyspark, Kafka stream programming.

For deploying this project, I would like to thank Dr. Henry Chang an TA Liang for providing all the required input .

Also, for all I would like to always pray to Almighty for giving us wisdom and power to understand things.

# Content

Index :

Abstract

Kafka Installation and configuration

Spark Streaming with scala

NetworkWordCount Streaming

Conclusion

References

## Abstract

Event streaming is the digital equivalent of the human body's central nervous system. It is the technological foundation for the 'always-on' world where businesses are increasingly software-defined and automated, and where the user of software is more software.

Technically speaking, event streaming is the practice of capturing data in real-time from event sources like databases, sensors, mobile devices, cloud services, and software applications in the form of streams of events; storing these event streams durably for later retrieval; manipulating, processing, and reacting to the event streams in real-time as well as retrospectively; and routing the event streams to different destination technologies as needed. Event streaming thus ensures a continuous flow and interpretation of data so that the right information is at the right place, at the right time.

# Kafka Installation download required version



Kafka 3.2.1 fixes 13 issues since the 3.2.0 release. For more information, please read the detailed [Release Notes](#).

## 3.2.0

- Released May 17, 2022
- [Release Notes](#)
- Source download: [kafka-3.2.0-src.tgz](#) ([asc](#), [sha512](#))
- Binary downloads:
  - Scala 2.12 - [kafka\\_2.12-3.2.0.tgz](#) ([asc](#), [sha512](#))
  - Scala 2.13 - [kafka\\_2.13-3.2.0.tgz](#) ([asc](#), [sha512](#))

We build for multiple versions of Scala. This only matters if you are using Scala and you want a version built for the same Scala version you use. Otherwise any version should work (2.13 is recommended).

Kafka 3.2.0 includes a number of significant new features. Here is a summary of some notable changes:

- log4j 1.x is replaced with reload4j
- StandardAuthorizer for KRaft (KIP-801)
- Send a hint to the partition leader to recover the partition (KIP-704)
- Top-level error code field in DescribeLogDirsResponse (KIP-784)
- kafka-console-producer writes headers and null values (KIP-798 and KIP-810)
- JoinGroupRequest and LeaveGroupRequest have a reason attached (KIP-800)
- Static membership protocol into the leader election assignment (KIP-814)

# Installation of Kafka

```
$ tar -xzf kafka_2.13-3.3.1.tgz
```

```
$ cd kafka_2.13-3.3.1
```

Open another terminal session and run:

# Start the Kafka broker service

\$ bin/kafka-server-start.sh config/server.properties

kafka@cs570bigdata: ~/kafka

kafka@cs570bigdata:~/kafka\$ bin/zookeeper-server-start.sh config/zookeeper.properties

```
[2022-12-01 08:20:26,116] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,120] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,237] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,244] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,252] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,256] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,281] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DataDirCleanupManager)
[2022-12-01 08:20:26,281] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DataDirCleanupManager)
[2022-12-01 08:20:26,281] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DataDirCleanupManager)
[2022-12-01 08:20:26,282] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2022-12-01 08:20:26,327] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2022-12-01 08:20:26,331] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,331] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,332] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,332] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,332] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,332] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-12-01 08:20:26,332] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
```

## Zookeeper port details

kafka@cs570bigdata: ~/kafka

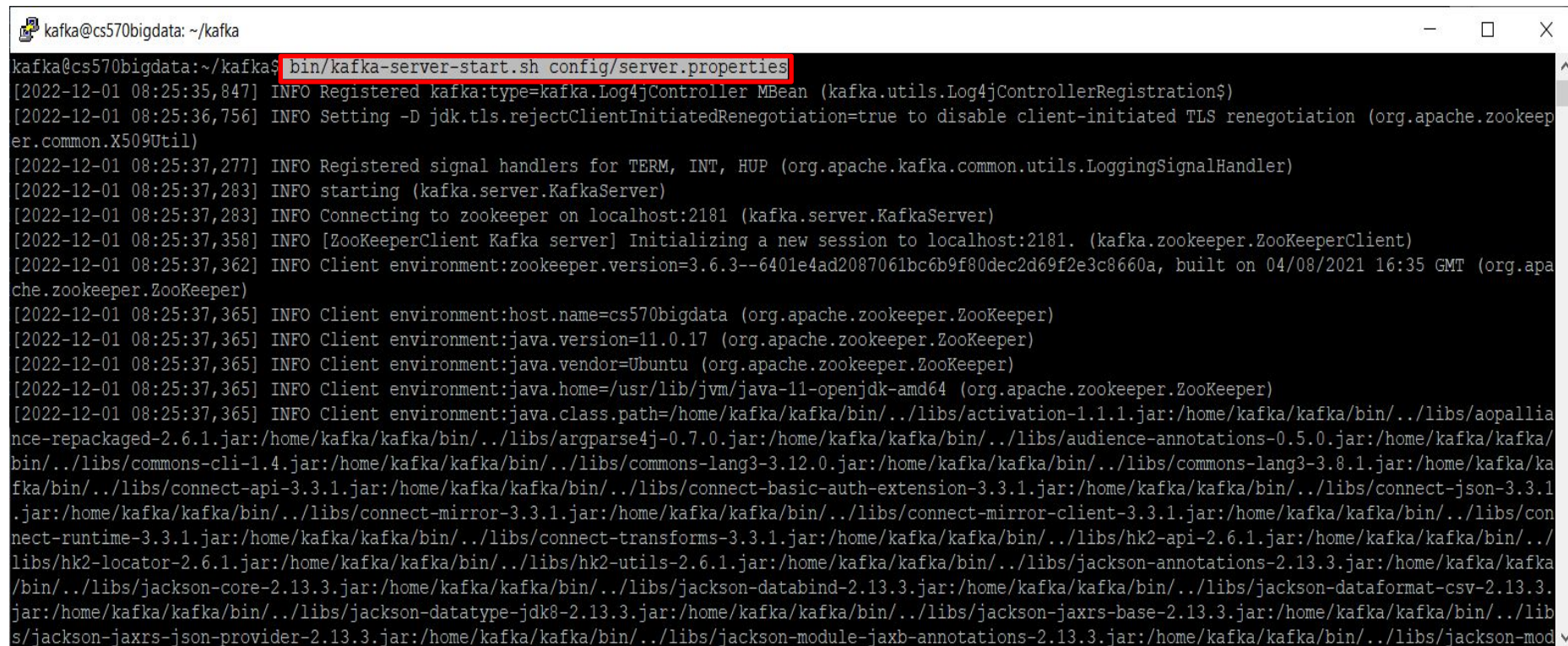
```
B direct buffers. (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2022-12-01 08:20:26,647] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2022-12-01 08:20:26,736] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
)
[2022-12-01 08:20:26,736] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)
)
[2022-12-01 08:20:26,743] INFO zookeeper.snapshotSizeFactor = 0.33 (org.apache.zookeeper.server.ZKDatabase)
[2022-12-01 08:20:26,743] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)
[2022-12-01 08:20:26,818] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream)
[2022-12-01 08:20:26,862] INFO Reading snapshot /home/kafka/kafka-logs/zookeeper/version-2/snapshot.92 (org.apache.zookeeper.server.persistence.FileSnap)
[2022-12-01 08:20:26,910] INFO The digest in the snapshot has digest version of 2, , with zxid as 0x92, and digest value as 292384779848 (org.apache.zookeeper.server.DataTree)
[2022-12-01 08:20:27,084] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
[2022-12-01 08:20:27,095] INFO 19 txns loaded in 19 ms (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2022-12-01 08:20:27,095] INFO Snapshot loaded in 351 ms, highest zxid is 0xa5, digest is 301865413955 (org.apache.zookeeper.server.ZKDatabase)
[2022-12-01 08:20:27,095] INFO Snapshotting: 0xa5 to /home/kafka/kafka-logs/zookeeper/version-2/snapshot.a5 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2022-12-01 08:20:27,099] INFO Snapshot taken in 4 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2022-12-01 08:20:27,130] INFO zookeeper.request_throttler.shutdownTimeout = 10000 (org.apache.zookeeper.server.RequestThrottler)
[2022-12-01 08:20:27,138] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepareRequestProcessor)
[2022-12-01 08:20:27,184] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)
[2022-12-01 08:20:47,249] INFO Expiring session 0x10000711b910000, timeout of 18000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)
[2022-12-01 08:20:47,266] INFO Creating new log file: log.a6 (org.apache.zookeeper.server.persistence.FileTxnLog)
```



Open another terminal session and run:

```
# Start the Kafka broker service
```

```
$ bin/kafka-server-start.sh config/server.properties
```

A terminal window titled 'kafka@cs570bigdata: ~/kafka' with standard window controls. The terminal shows the command 'bin/kafka-server-start.sh config/server.properties' being executed. The output consists of several log lines: 'INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration\$)', 'INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)', 'INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)', 'INFO starting (kafka.server.KafkaServer)', 'INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)', 'INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)', 'INFO Client environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache.zookeeper.ZooKeeper)', and a series of 'INFO Client environment:' messages for host.name, java.version, java.vendor, java.home, and java.class.path. The class path is a long string of jar files including activation, aopalliance-repackaged, argparse4j, audience-annotations, commons-cli, commons-lang3, connect-api, connect-basic-auth-extension, connect-json, connect-mirror, connect-mirror-client, connect-runtime, connect-transforms, hk2-api, hk2-locator, hk2-utils, jackson-annotations, jackson-core, jackson-databind, jackson-dataformat-csv, jackson-datatype-jdk8, jackson-jaxrs-base, jackson-jaxrs-json-provider, jackson-module-jaxb-annotations, and jackson-mod.

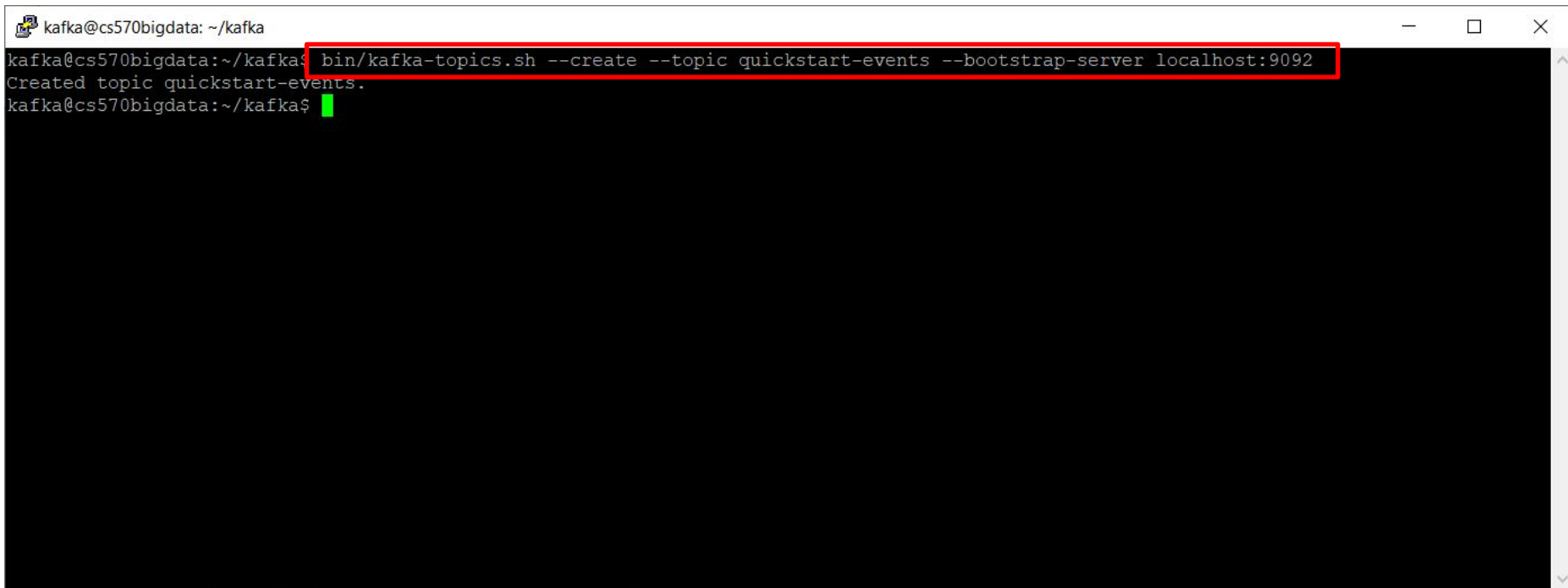
```
kafka@cs570bigdata: ~/kafka$ bin/kafka-server-start.sh config/server.properties
[2022-12-01 08:25:35,847] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2022-12-01 08:25:36,756] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2022-12-01 08:25:37,277] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2022-12-01 08:25:37,283] INFO starting (kafka.server.KafkaServer)
[2022-12-01 08:25:37,283] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2022-12-01 08:25:37,358] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2022-12-01 08:25:37,362] INFO Client environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache.zookeeper.ZooKeeper)
[2022-12-01 08:25:37,365] INFO Client environment:host.name=cs570bigdata (org.apache.zookeeper.ZooKeeper)
[2022-12-01 08:25:37,365] INFO Client environment:java.version=11.0.17 (org.apache.zookeeper.ZooKeeper)
[2022-12-01 08:25:37,365] INFO Client environment:java.vendor=Ubuntu (org.apache.zookeeper.ZooKeeper)
[2022-12-01 08:25:37,365] INFO Client environment:java.home=/usr/lib/jvm/java-11-openjdk-amd64 (org.apache.zookeeper.ZooKeeper)
[2022-12-01 08:25:37,365] INFO Client environment:java.class.path=/home/kafka/kafka/bin/./libs/activation-1.1.1.jar:/home/kafka/kafka/bin/./libs/aopalliance-repackaged-2.6.1.jar:/home/kafka/kafka/bin/./libs/argparse4j-0.7.0.jar:/home/kafka/kafka/bin/./libs/audience-annotations-0.5.0.jar:/home/kafka/kafka/bin/./libs/commons-cli-1.4.jar:/home/kafka/kafka/bin/./libs/commons-lang3-3.12.0.jar:/home/kafka/kafka/bin/./libs/commons-lang3-3.8.1.jar:/home/kafka/kafka/bin/./libs/connect-api-3.3.1.jar:/home/kafka/kafka/bin/./libs/connect-basic-auth-extension-3.3.1.jar:/home/kafka/kafka/bin/./libs/connect-json-3.3.1.jar:/home/kafka/kafka/bin/./libs/connect-mirror-3.3.1.jar:/home/kafka/kafka/bin/./libs/connect-mirror-client-3.3.1.jar:/home/kafka/kafka/bin/./libs/connect-runtime-3.3.1.jar:/home/kafka/kafka/bin/./libs/connect-transforms-3.3.1.jar:/home/kafka/kafka/bin/./libs/hk2-api-2.6.1.jar:/home/kafka/kafka/bin/./libs/hk2-locator-2.6.1.jar:/home/kafka/kafka/bin/./libs/hk2-utils-2.6.1.jar:/home/kafka/kafka/bin/./libs/jackson-annotations-2.13.3.jar:/home/kafka/kafka/bin/./libs/jackson-core-2.13.3.jar:/home/kafka/kafka/bin/./libs/jackson-databind-2.13.3.jar:/home/kafka/kafka/bin/./libs/jackson-dataformat-csv-2.13.3.jar:/home/kafka/kafka/bin/./libs/jackson-datatype-jdk8-2.13.3.jar:/home/kafka/kafka/bin/./libs/jackson-jaxrs-base-2.13.3.jar:/home/kafka/kafka/bin/./libs/jackson-jaxrs-json-provider-2.13.3.jar:/home/kafka/kafka/bin/./libs/jackson-module-jaxb-annotations-2.13.3.jar:/home/kafka/kafka/bin/./libs/jackson-mod
```

## Kafka started successfully

```
kafka@cs570bigdata: ~/kafka
[2022-12-01 08:25:45,558] INFO Creating /brokers/ids/0 (is it secure? false) (kafka.zk.KafkaZkClient)
[2022-12-01 08:25:45,496] INFO [ExpirationReaper-0-Fetch]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2022-12-01 08:25:45,665] INFO Stat of the created znode at /brokers/ids/0 is: 182,182,1669863345609,1669863345609,1,0,0,72057776742662144,208,0,182
(kafka.zk.KafkaZkClient)
[2022-12-01 08:25:45,666] INFO Registered broker 0 at path /brokers/ids/0 with addresses: PLAINTEXT://cs570bigdata:9092, czxid (broker epoch): 182 (kafka.z
k.KafkaZkClient)
[2022-12-01 08:25:45,966] INFO [ExpirationReaper-0-topic]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2022-12-01 08:25:45,966] INFO [ExpirationReaper-0-Heartbeat]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2022-12-01 08:25:45,971] INFO [ExpirationReaper-0-Rebalance]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2022-12-01 08:25:46,026] INFO [GroupCoordinator 0]: Starting up. (kafka.coordinator.group.GroupCoordinator)
[2022-12-01 08:25:46,122] INFO [GroupCoordinator 0]: Startup complete. (kafka.coordinator.group.GroupCoordinator)
[2022-12-01 08:25:46,177] INFO [TransactionCoordinator id=0] Starting up. (kafka.coordinator.transaction.TransactionCoordinator)
[2022-12-01 08:25:46,342] INFO [TransactionCoordinator id=0] Startup complete. (kafka.coordinator.transaction.TransactionCoordinator)
[2022-12-01 08:25:46,353] INFO [Transaction Marker Channel Manager 0]: Starting (kafka.coordinator.transaction.TransactionMarkerChannelManager)
[2022-12-01 08:25:46,424] INFO [ExpirationReaper-0-AlterAcls]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2022-12-01 08:25:46,545] INFO [/config/changes-event-process-thread]: Starting (kafka.common.ZkNodeChangeNotificationListener$ChangeEventProcessThread)
[2022-12-01 08:25:47,204] INFO [SocketServer listenerType=ZK_BROKER, nodeId=0] Enabling request processing. (kafka.network.SocketServer)
[2022-12-01 08:25:47,289] INFO Kafka version: 3.3.1 (org.apache.kafka.common.utils.AppInfoParser)
[2022-12-01 08:25:47,302] INFO Kafka commitId: e23c59d00e687ff5 (org.apache.kafka.common.utils.AppInfoParser)
[2022-12-01 08:25:47,302] INFO Kafka startTimeMs: 1669863347273 (org.apache.kafka.common.utils.AppInfoParser)
[2022-12-01 08:25:47,303] INFO [KafkaServer id=0] started (kafka.server.KafkaServer)
[2022-12-01 08:25:48,611] INFO [BrokerToControllerChannelManager broker=0 name=forwarding]: Recorded new controller, from now on will use broker cs570bigda
ta:9092 (id: 0 rack: null) (kafka.server.BrokerToControllerRequestThread)
[2022-12-01 08:25:48,611] INFO [BrokerToControllerChannelManager broker=0 name=alterPartition]: Recorded new controller, from now on will use broker cs570b
```

## CREATE A TOPIC TO STORE YOUR EVENTS

```
bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092
```

A terminal window with a title bar showing 'kafka@cs570bigdata: ~/kafka'. The terminal content shows the command 'bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092' being entered and executed. The output is 'Created topic quickstart-events.' followed by a new prompt 'kafka@cs570bigdata:~/kafka\$'. The command line is highlighted with a red rectangle.

```
kafka@cs570bigdata: ~/kafka
kafka@cs570bigdata:~/kafka$ bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092
Created topic quickstart-events.
kafka@cs570bigdata:~/kafka$
```

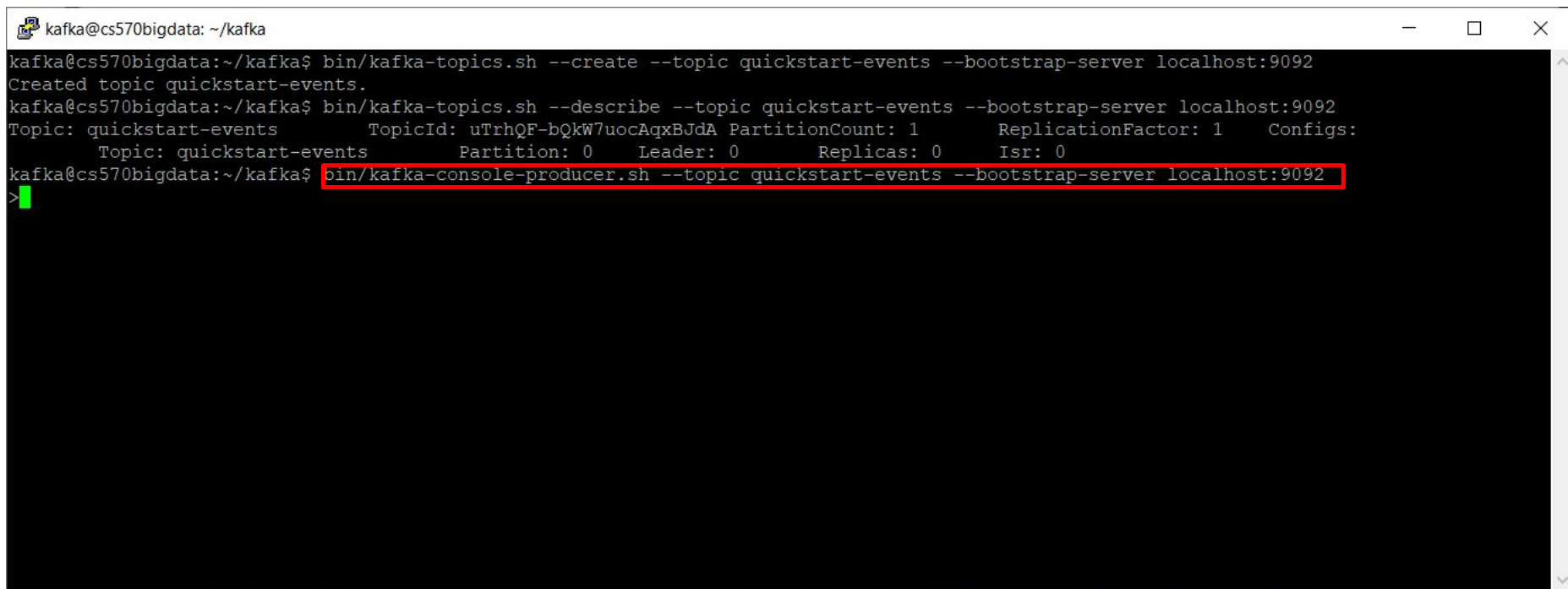
## Describe the events described above

```
kafka@cs570bigdata: ~/kafka
kafka@cs570bigdata:~/kafka$ bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092
Created topic quickstart-events.
kafka@cs570bigdata:~/kafka$ bin/kafka-topics.sh --describe --topic quickstart-events --bootstrap-server localhost:9092
Topic: quickstart-events      TopicId: uTrhQF-bQkW7uocAqxBJdA PartitionCount: 1      ReplicationFactor: 1      Configs:
    Topic: quickstart-events    Partition: 0      Leader: 0      Replicas: 0      Isr: 0
kafka@cs570bigdata:~/kafka$
```



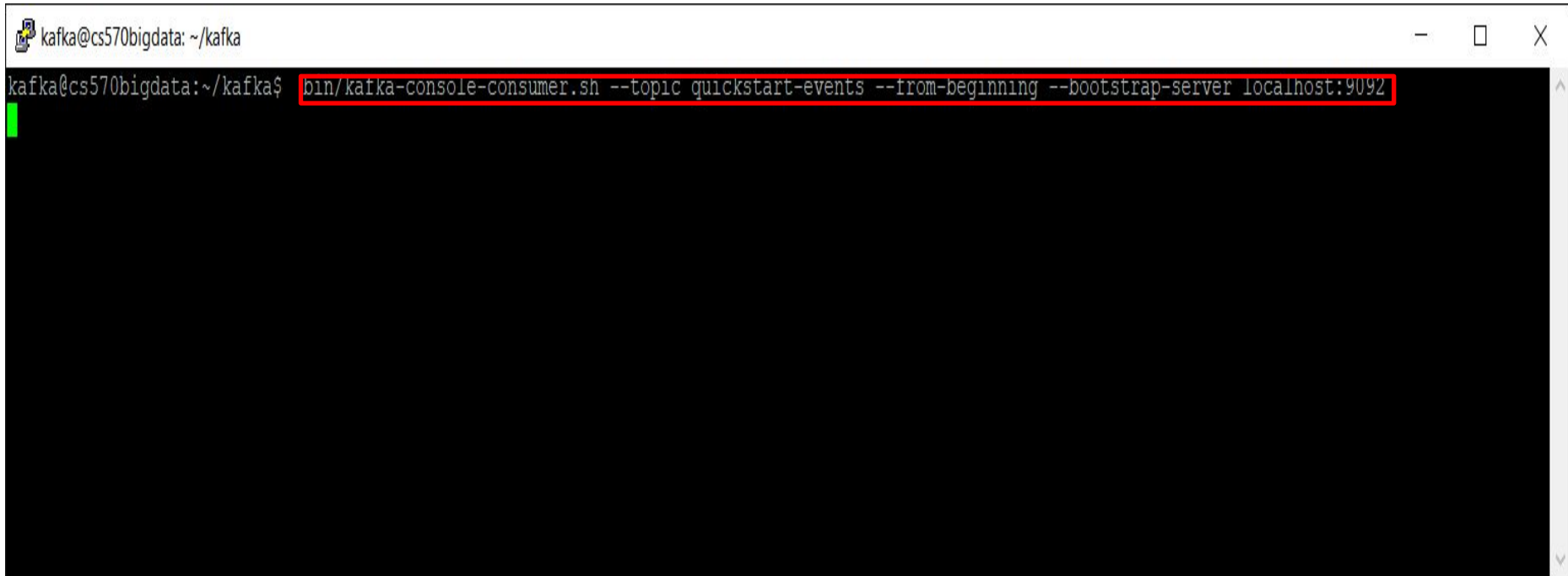
Run the console producer client to write a few events into your topic. By default, each line you enter will result in a separate event being written to the topic. (**WRITE SOME EVENTS INTO THE TOPIC**)

```
bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092
```

A terminal window titled 'kafka@cs570bigdata: ~/kafka' with standard window controls. The terminal shows the following sequence of commands and output:  
1. Command: `bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092`  
Output: `Created topic quickstart-events.`  
2. Command: `bin/kafka-topics.sh --describe --topic quickstart-events --bootstrap-server localhost:9092`  
Output: `Topic: quickstart-events TopicId: uTrhQF-bQkW7uocAqxBJdA PartitionCount: 1 ReplicationFactor: 1 Configs:  
 Topic: quickstart-events Partition: 0 Leader: 0 Replicas: 0 Isr: 0`  
3. Command: `bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092` (This line is highlighted with a red box in the image)  
The prompt `>` is visible on the line following the third command.

Open another terminal session and run the console consumer client to read the events you just created: ([READ THE EVENTS](#))

```
bin/kafka-console-consumer.sh --topic quickstart-events --bootstrap-server localhost:9092
```

A terminal window with a title bar showing 'kafka@cs570bigdata: ~/kafka'. The terminal content shows the prompt 'kafka@cs570bigdata:~/kafka\$' followed by the command 'bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-server localhost:9092'. The command is highlighted with a red rectangular box. A green cursor is visible on the line below the command.

```
kafka@cs570bigdata: ~/kafka
kafka@cs570bigdata:~/kafka$ bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-server localhost:9092
```

## Kafka Producer and Consumer for the first event

```
kafka@cs570bigdata: ~/kafka
kafka@cs570bigdata:~/kafka$ bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092
Created topic quickstart-events.
kafka@cs570bigdata:~/kafka$ bin/kafka-topics.sh --describe --topic quickstart-events --bootstrap-server localhost:9092
Topic: quickstart-events      TopicId: uTrhQF-bQkW7uocAqxBJdA PartitionCount: 1      ReplicationFactor: 1    Configs:
Topic: quickstart-events      Partition: 0    Leader: 0      Replicas: 0      Isr: 0
kafka@cs570bigdata:~/kafka$ bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092
>This is my first event
>
```

```
kafka@cs570bigdata: ~/kafka
kafka@cs570bigdata:~/kafka$ bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-server localhost:9092
This is my first event
```

```
kafka@cs570bigdata: ~/kafka
kafka@cs570bigdata:~/kafka$ bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092
Created topic quickstart-events.
kafka@cs570bigdata:~/kafka$ bin/kafka-topics.sh --describe --topic quickstart-events --bootstrap-server localhost:9092
Topic: quickstart-events      TopicId: uTrhQF-bQkW7uocAqxBJdA PartitionCount: 1      ReplicationFactor: 1      Configs:
      Topic: quickstart-events Partition: 0      Leader: 0      Replicas: 0      Isr: 0
kafka@cs570bigdata:~/kafka$ bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092
>This is my first event
>This is my Second event
>This is week 12 assignment
>
```

```
kafka@cs570bigdata: ~/kafka
kafka@cs570bigdata:~/kafka$ bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-server localhost:9092
This is my first event
This is my Second event
This is week 12 assignment
```



## pip3 install msgpack

```
hduser@cs570bigdata: ~/kafka_2.12-3.2.0
hduser@cs570bigdata:~/kafka_2.12-3.2.0$ python3 producer.py
hduser@cs570bigdata:~/kafka_2.12-3.2.0$ pip3 install msgpack
Collecting msgpack
  Downloading msgpack-1.0.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (322 kB)
    | 322 kB 553 kB/s
Installing collected packages: msgpack
Successfully installed msgpack-1.0.4
hduser@cs570bigdata:~/kafka_2.12-3.2.0$ █
```

## pip install kafka-python

```
hduser@cs570bigdata: ~
hduser@cs570bigdata:~$ pip install kafka-python
Collecting kafka-python
  Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
    | 246 kB 674 kB/s
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
hduser@cs570bigdata:~$ █
```

hduser@cs570bigdata: ~

```
File "/home/hduser/.local/lib/python3.8/site-packages/kafka/producer/kafka.py", line 381, in __init__
    client = KafkaClient(metrics=self._metrics, metric_group_prefix='producer',
File "/home/hduser/.local/lib/python3.8/site-packages/kafka/client_async.py", line 244, in __init__
    self.config['api_version'] = self.check_version(timeout=check_timeout)
File "/home/hduser/.local/lib/python3.8/site-packages/kafka/client_async.py", line 927, in check_version
    raise Errors.NoBrokersAvailable()
```

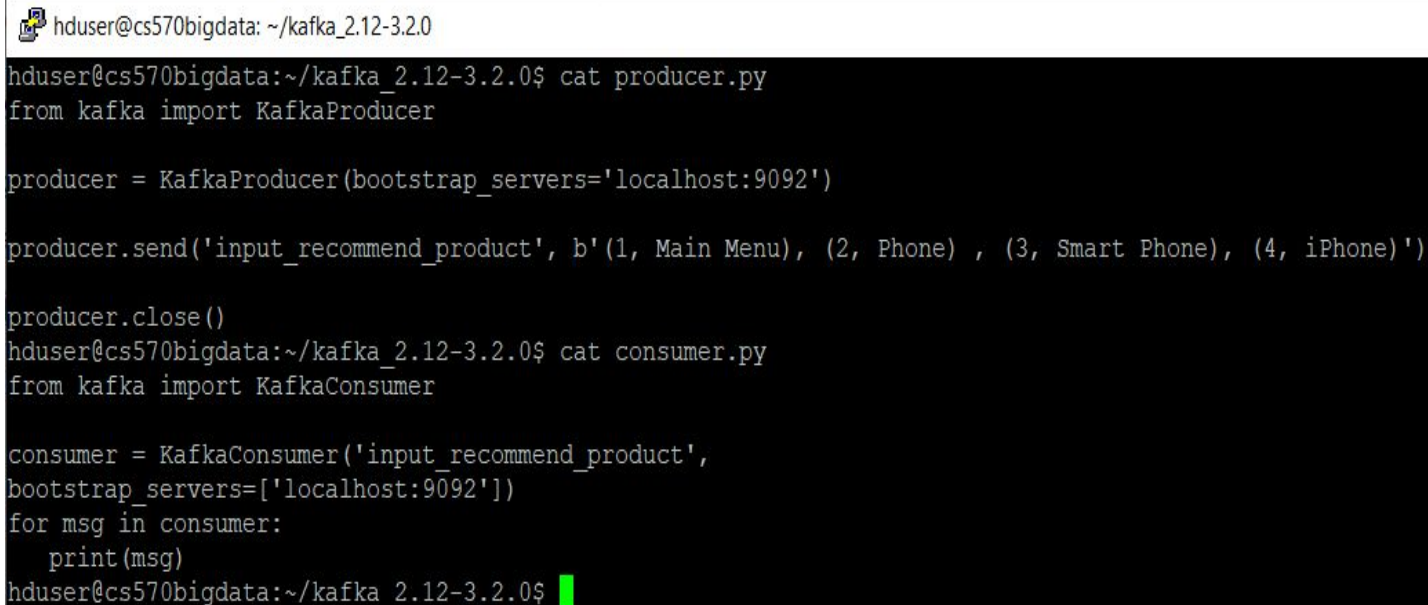
kafka.errors.NoBrokersAvailable: NoBrokersAvailable

```
>>> producer = KafkaProducer(bootstrap_servers='10.0.0.50:9092')
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f5044f5fd30>
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f50446f56d0>
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f50446f8340>
>>>
```

```
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f5044f5fd30>
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f50446f56d0>
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f50446f8340>
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f50446f8640>
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f50446f8820>
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f50446f8940>
>>> producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')
<kafka.producer.future.FutureRecordMetadata object at 0x7f50446f8b20>
>>>
```

[illegible]

## Create producer.py and consumer.py and run in 2 terminals



A terminal window titled 'hduser@cs570bigdata: ~/kafka\_2.12-3.2.0' with standard window controls. The terminal shows the creation of two Python files. First, 'producer.py' is created with code to send a message to a Kafka topic. Then, 'consumer.py' is created with code to receive messages from the same topic. The prompt returns to the shell after each file is created.

```
hduser@cs570bigdata: ~/kafka_2.12-3.2.0
hduser@cs570bigdata:~/kafka_2.12-3.2.0$ cat producer.py
from kafka import KafkaProducer

producer = KafkaProducer(bootstrap_servers='localhost:9092')

producer.send('input_recommend_product', b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)')

producer.close()
hduser@cs570bigdata:~/kafka_2.12-3.2.0$ cat consumer.py
from kafka import KafkaConsumer

consumer = KafkaConsumer('input_recommend_product',
bootstrap_servers=['localhost:9092'])
for msg in consumer:
    print(msg)
hduser@cs570bigdata:~/kafka_2.12-3.2.0$
```

## Run producer.py and consumer.py in 2 different terminals

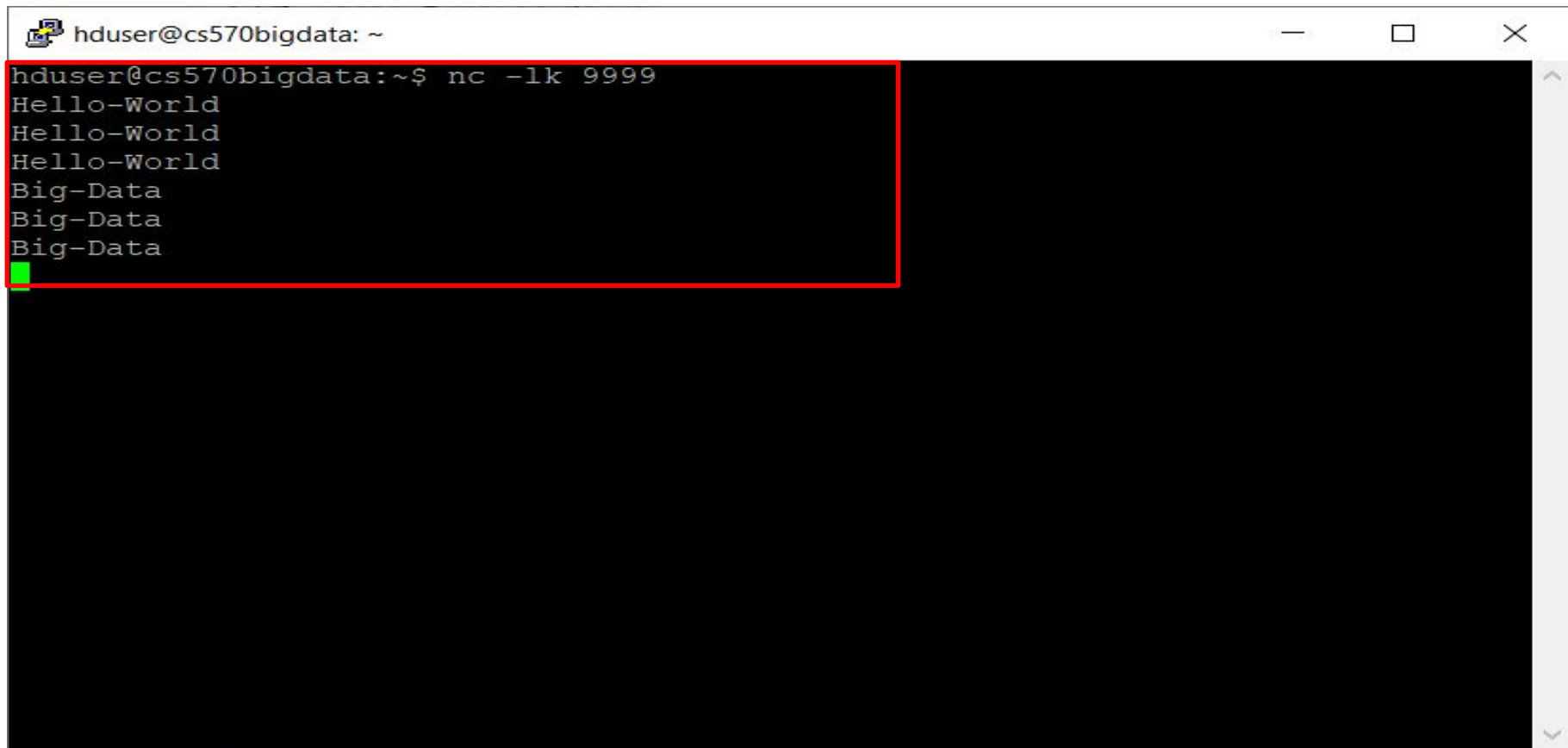
```
hduser@cs570bigdata: ~/kafka_2.12-3.2.0
hduser@cs570bigdata: ~/kafka_2.12-3.2.0$ python3 producer.py
hduser@cs570bigdata:~/kafka_2.12-3.2.0$
```

```
hduser@cs570bigdata: ~/kafka_2.12-3.2.0
hduser@cs570bigdata:~/kafka_2.12-3.2.0$ python3 consumer.py
ConsumerRecord(topic='input_recommend_product', partition=0, offset=0, timestamp=1670186311427, timestamp_type=0, key=None, value=b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)', headers=[], checksum=None, serialized_key_size=-1, serialized_value_size=58, serialized_header_size=-1)
```

## Wordcount Scala Program Run To Count Words

```
hduser@cs570bigdata: ~  
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@7546033d  
  
scala> // Create a DStream that will connect to hostname:port, like localhost:9999  
  
scala> val lines = ssc.socketTextStream("10.0.0.50", 9999)  
lines: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = org.apache.spark.streaming.dstream.SocketInputDStream@1e060d4b  
  
scala> // Split each line in each batch into words  
  
scala> val words = lines.flatMap(_._split(" "))  
words: org.apache.spark.streaming.dstream.DStream[String] = org.apache.spark.streaming.dstream.FlatMappedDStream@11fcf35  
  
scala> // Count each word in each batch  
  
scala> val pairs = words.map(word => (word, 1))  
pairs: org.apache.spark.streaming.dstream.DStream[(String, Int)] = org.apache.spark.streaming.dstream.MappedDStream@2fb3113a  
  
scala> val wordCounts = pairs.reduceByKey(_ + _)  
wordCounts: org.apache.spark.streaming.dstream.DStream[(String, Int)] = org.apache.spark.streaming.dstream.ShuffledDStream@33ba63e4  
  
scala> // Print the elements of each RDD generated in this DStream to the console  
  
scala> wordCounts.print()  
  
scala> // Start the computation  
  
scala> ssc.start()
```

## Netcat listening on port 9999 with input data

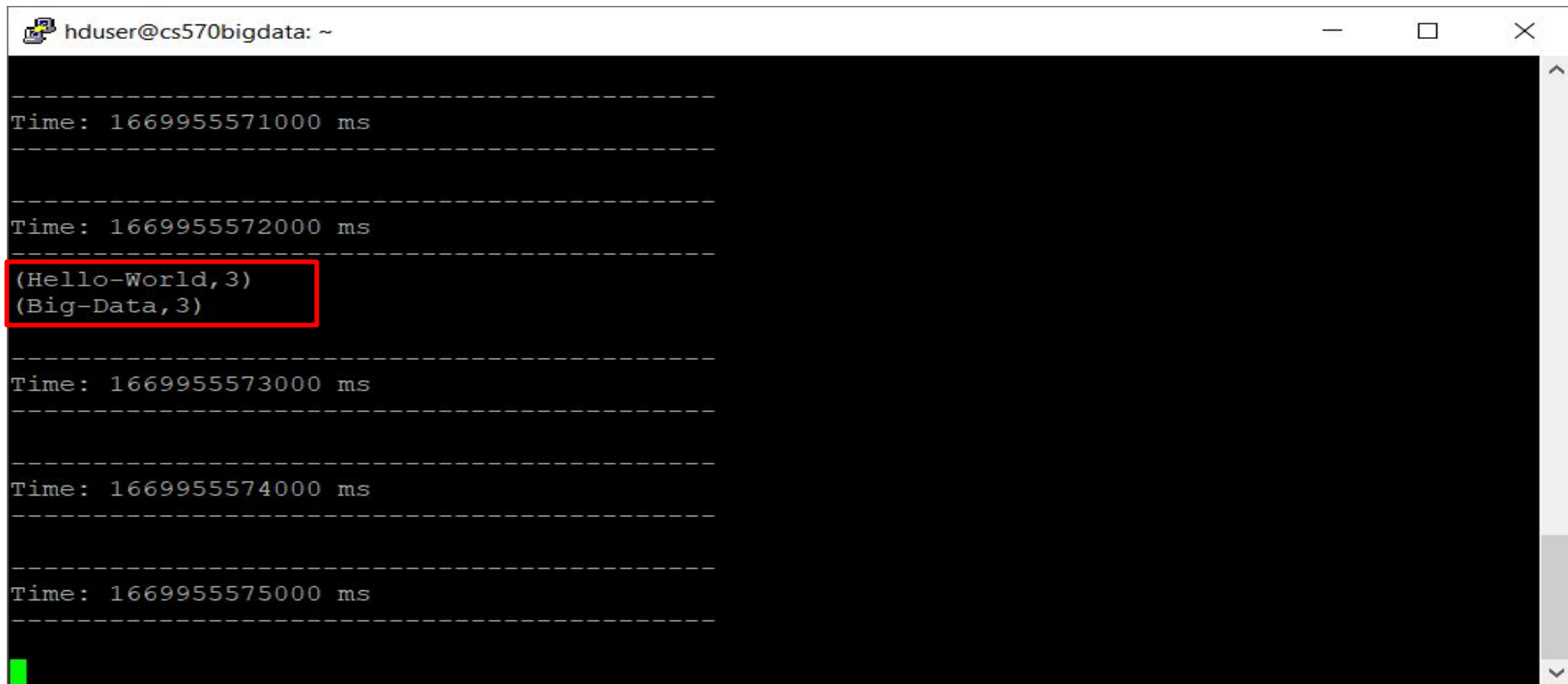


A terminal window titled "hduser@cs570bigdata: ~" with standard window controls. The terminal shows the command `nc -lk 9999` being executed. It then receives three lines of input: "Hello-World", "Hello-World", and "Hello-World", followed by three lines of input: "Big-Data", "Big-Data", and "Big-Data". A green cursor is visible on the line following the last "Big-Data" input. A red rectangular box highlights the command and the first three lines of input.

```
hduser@cs570bigdata:~$ nc -lk 9999
Hello-World
Hello-World
Hello-World
Big-Data
Big-Data
Big-Data
█
```



## Output of Streaming Data



```
hduser@cs570bigdata: ~  
-----  
Time: 1669955571000 ms  
-----  
-----  
Time: 1669955572000 ms  
-----  
(Hello-World,3)  
(Big-Data,3)  
-----  
Time: 1669955573000 ms  
-----  
-----  
Time: 1669955574000 ms  
-----  
-----  
Time: 1669955575000 ms  
-----  
█
```

A terminal window titled "hduser@cs570bigdata: ~" displays streaming data output. The output consists of several lines of text, each preceded by a dashed line and a timestamp. The lines are: "Time: 1669955571000 ms", "Time: 1669955572000 ms", "(Hello-World,3)", "(Big-Data,3)", "Time: 1669955573000 ms", "Time: 1669955574000 ms", and "Time: 1669955575000 ms". The lines "(Hello-World,3)" and "(Big-Data,3)" are highlighted with a red rectangular box. A green cursor is visible at the bottom left of the terminal window.



## Netcat listening on port 9999 with input data



hduser@cs570bigdata: ~

```
hduser@cs570bigdata:~$ nc -lk 9999
```

```
Hello-World
```

```
Hello-World
```

```
Apple Apple
```

```
Apple Apple
```

```
CS575
```

```
CS575
```

## Output of Streaming Data

```
hduser@cs570bigdata: ~/gweek12

scala>

scala> // Wait for the computation to terminate

22/12/02 09:47:08 WARN RandomBlockReplicationPolicy: Expecting 1 replicas with only 0 peer/s.
22/12/02 09:47:08 WARN BlockManager: Block input-0-1669954628600 replicated to only 0 peer(s) instead of 1 peers
-----
Time: 1669954628000 ms
-----

Time: 1669954630000 ms
-----

(Apple,4)
(CS575,2)
>Hello-World,2)
-----

Time: 1669954632000 ms
-----

Time: 1669954634000 ms
-----

Time: 1669954636000 ms
-----
```

## Netcat Connection

- Hint 1: use the netcat server to send the text through a TCP connection. For example, `nc -lk 9999` can be used to transfer any text that you type in the terminal through port 9999.
- Hint 2: if you are using `local` as the master URL when creating `StreamingContext`, you have to give at least one more core as the number of input streams, because each input stream would create a receiver that occupies one core. If you use `local`, it only gives one core to the context, which is used by the socket stream's receiver, leaving no core available for processing the data, thus, you should use `local[2]`

## Scala Program NetworkWordCount

To run this on your local machine, you need to first run a Netcat server

```
* ` $ nc -lk 9999`
```

\* and then run the example

```
* ` $ bin/run-example org.apache.spark.examples.streaming.NetworkWordCount localhost 9999`
```

```
package org.apache.spark.examples.streaming
import org.apache.spark.SparkConf
import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.{Seconds, StreamingContext}
object NetworkWordCount {
  def main(args: Array[String]): Unit = {
    if (args.length < 2) {
      System.err.println("Usage: NetworkWordCount <hostname> <port>")
      System.exit(1)
    }
    StreamingExamples.setStreamingLogLevels()
    // Create the context with a 1 second batch size
    val sparkConf = new SparkConf().setAppName("NetworkWordCount")
    val ssc = new StreamingContext(sparkConf, Seconds(1))
    // Create a socket stream on target ip:port and count the
    // words in input stream of \n delimited text (e.g. generated by 'nc')
    // Note that no duplication in storage level only for running locally.
    // Replication necessary in distributed scenario for fault tolerance.
    val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY_AND_DISK_SER)
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()
  }
}
```

hduser@cs570bigdata:~\$ nc -lk 9999

```
hduser@cs570bigdata: ~  
PROJECT Kafka SPARK StreamingPROJECT Kafka SPARK St  
reamingPROJECT Kafka SPARK Streaming STREAMING STREAMING STREAMINGSTREAMINGSTREA  
MING STREAMINGSTREAMING
```

## Scala Streaming NetworkWordCount localhost 9999

```
hduser@cs570bigdata: /stage  
22/12/05 04:24:33 INFO TaskSchedulerImpl: Removed TaskSet 2552.0, whose tasks have all completed, from pool  
22/12/05 04:24:33 INFO BlockManagerInfo: Removed broadcast_1288_piece0 on cs570bigdata:45789 in memory (size: 2.9 KiB, free: 434.3 MiB)  
22/12/05 04:24:33 INFO DAGScheduler: ResultStage 2552 (print at NetworkWordCount.scala:56) finished in 0.175 s  
22/12/05 04:24:33 INFO DAGScheduler: Job 1276 is finished. Cancelling potential speculative or zombie tasks for this job  
22/12/05 04:24:33 INFO TaskSchedulerImpl: Killing all running tasks in stage 2552: Stage finished  
22/12/05 04:24:33 INFO DAGScheduler: Job 1276 finished: print at NetworkWordCount.scala:56, took 0.214097 s  
-----  
Time: 1670194473000 ms  
-----  
(STREAMING,2)  
(Kafka,4)  
(STREAMINGSTREAMINGSTREAMING,1)  
(Streaming,1)  
(STREAMINGSTREAMING,1)  
(StreamingPROJECT,3)  
(SPARK,4)  
(PROJECT,1)  
-----  
22/12/05 04:24:33 INFO JobScheduler: Finished job streaming job 1670194473000 ms.0 from job set of time 1670194473000 ms  
22/12/05 04:24:33 INFO JobScheduler: Total delay: 0.396 s for time 1670194473000 ms (execution: 0.392 s)  
22/12/05 04:24:33 INFO BlockManagerInfo: Removed broadcast_1289_piece0 on cs570bigdata:45789 in memory (size: 2.8 KiB, free: 434.4 MiB)
```

# Conclusion

Spark streaming is extension of core spark api, when used with Kafka and Python / scala we can achieve numerous things, it can mainly used for real time data for example weather, share marketing there are many to mention.

Also, we can place all the data in filesystems for future reference.

Also, we can apply in machine learning algorithms and graph processing.

# References

SFBU exercise materials