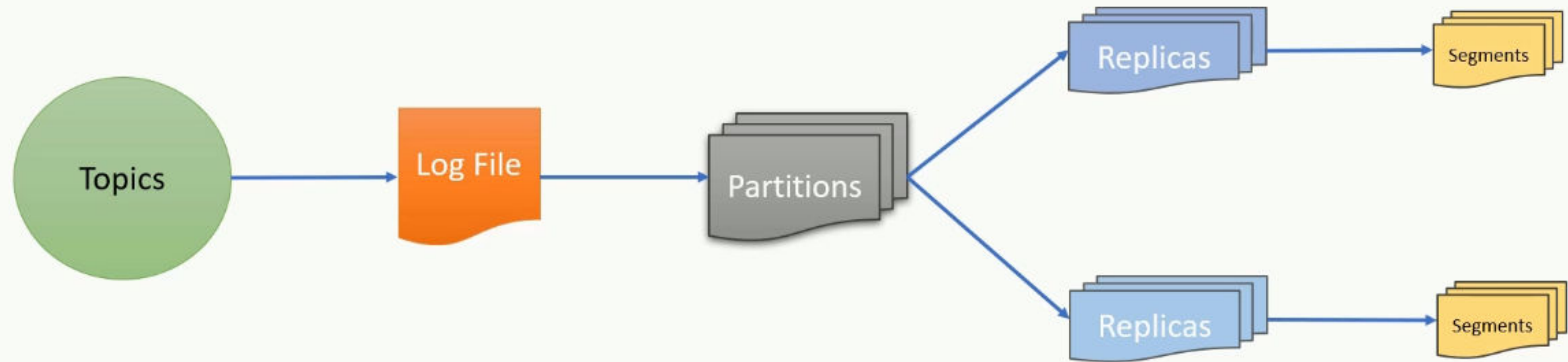


Kafka Storage Architecture



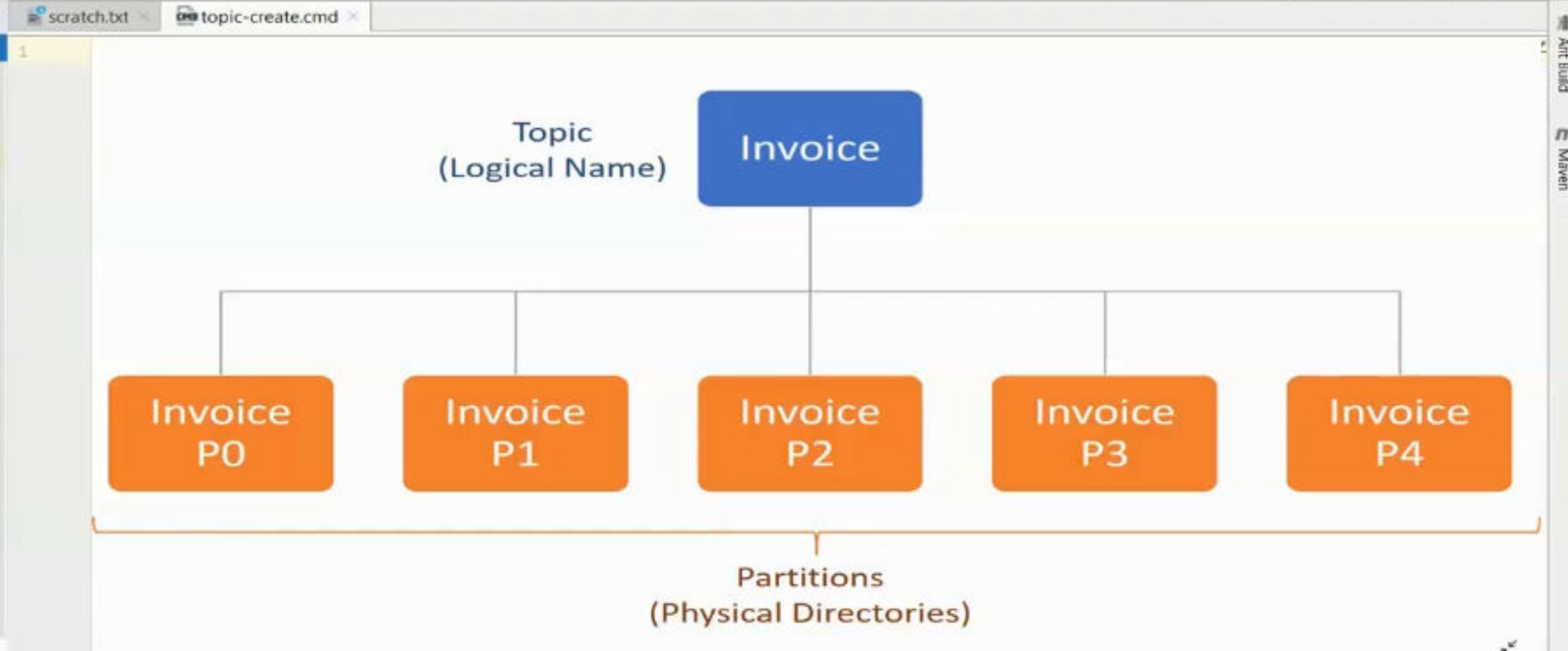
What is Kafka Topic?



01-storage-demo

Project

- 01-storage-demo C:\Udemy\FinalProjects\01-storage-demo
 - .idea
 - scripts
 - src
 - target
 - tmp
 - kafka-log-0
 - invoice-0
 - invoice-1
 - invoice-2
 - invoice-3
 - invoice-4
 - .lock
 - cleaner-offset-checkpoint
 - log-start-offset-checkpoint
 - meta.properties
 - recovery-point-offset-checkpoint
 - replication-offset-checkpoint
 - kafka-log-1
 - kafka-log-2
 - zookeeper-data
 - 01-storage-demo.iml
 - pom.xml



Run: zookeeper-start 0-kafka-server-start 1-kafka-server-start 2-kafka-server-start topic-create

```
cmd.exe /c topic-create.cmd

C:\Udemy\FinalProjects\01-storage-demo\scripts>kafka-topics.bat --create --zookeeper localhost:2181 --topic invoice --partitions 5 --replication-factor 3 --config
Created topic invoice.

Process finished with exit code 0
```

4: Run 6: TODO Terminal

Event Log

- > .idea
- > scripts
- > src
- > target
- ▼ tmp
 - └─ kafka-log-0
 - > invoice-0
 - > invoice-1
 - > invoice-2
 - > invoice-3
 - > invoice-4
 - .lock
 - cleaner-offset-checkpoint
 - log-start-offset-checkpoint
 - meta.properties
 - recovery-point-offset-checkpoint
 - replication-offset-checkpoint
 - > kafka-log-1
 - > kafka-log-2
 - > zookeeper-data
- 01-storage-demo.iml
- pom.xml

```
cmd.exe /c topic-create.cmd
```

```
C:\Udemy\FinalProjects\01-storage-demo\scripts>kafka-topics.bat --create --zookeeper localhost:2181 --topic invoice --partitions 5 --replication-factor 3 --config
```

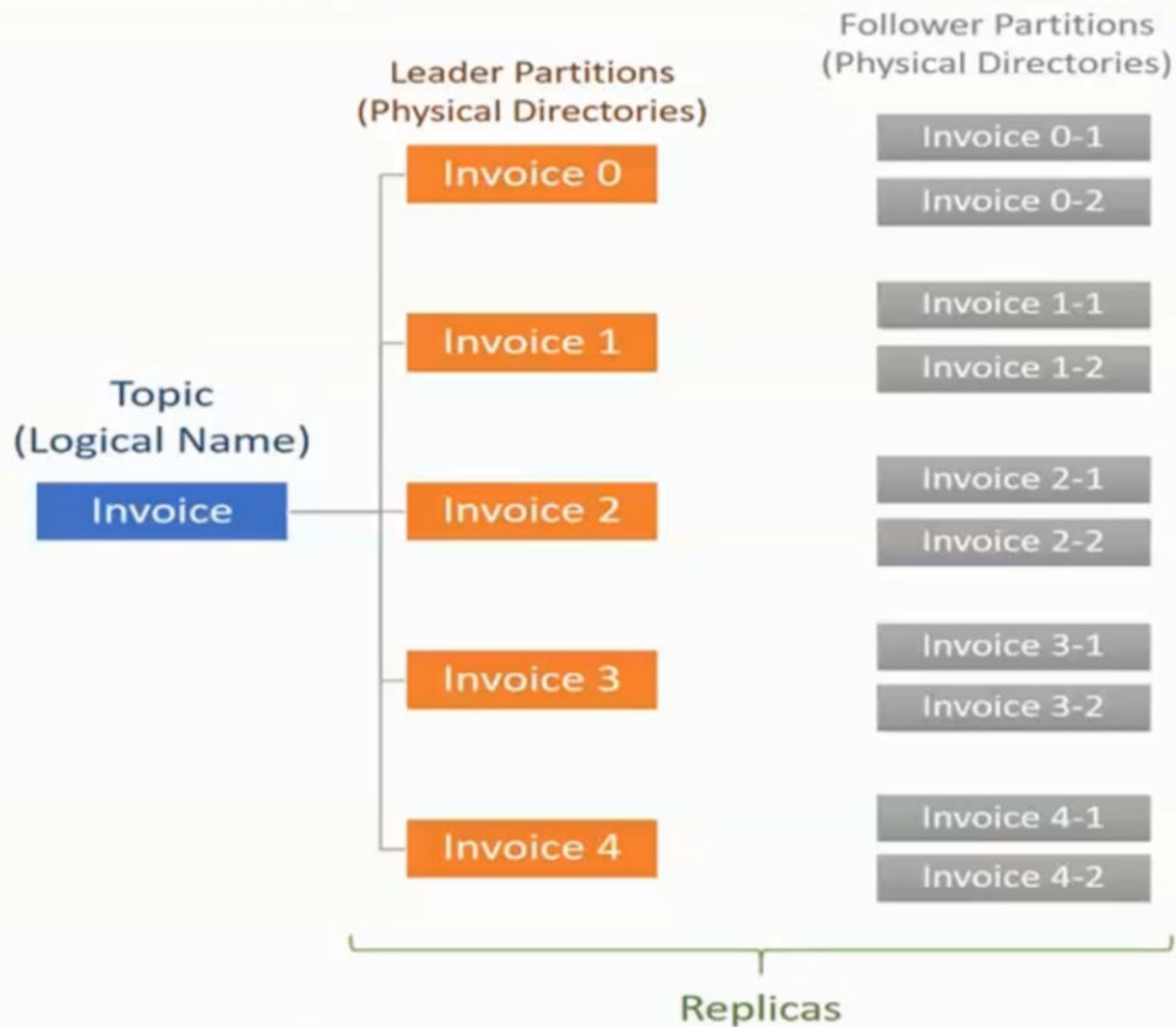
```
Created topic invoice.
```

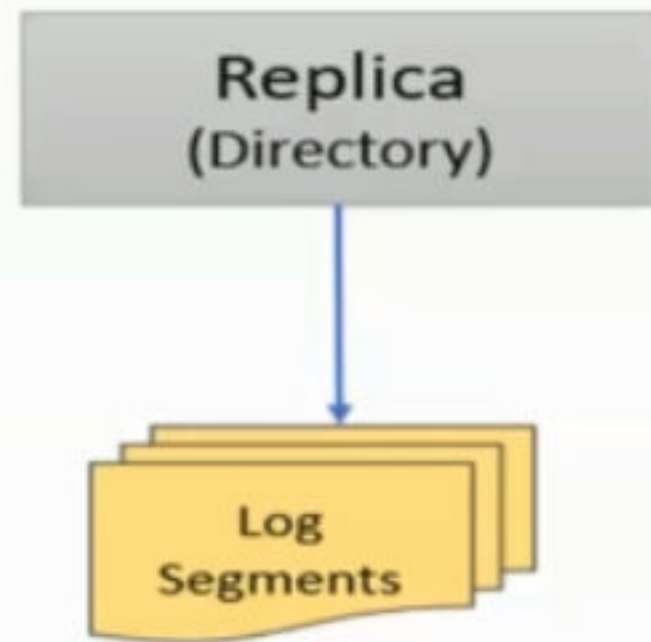
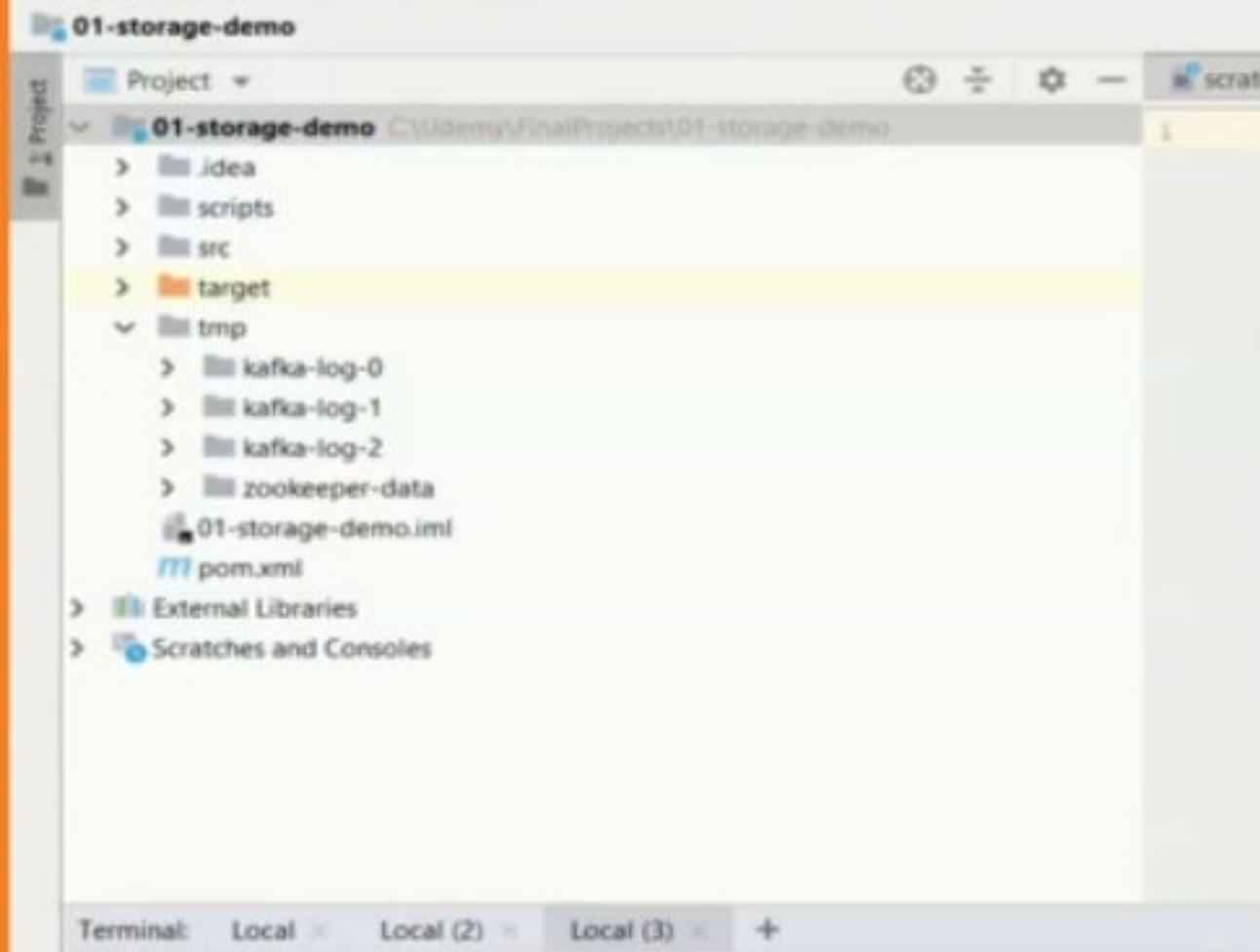
```
Process finished with exit code 0
```

What is Replication Factor?

Number of Copies for each Partition

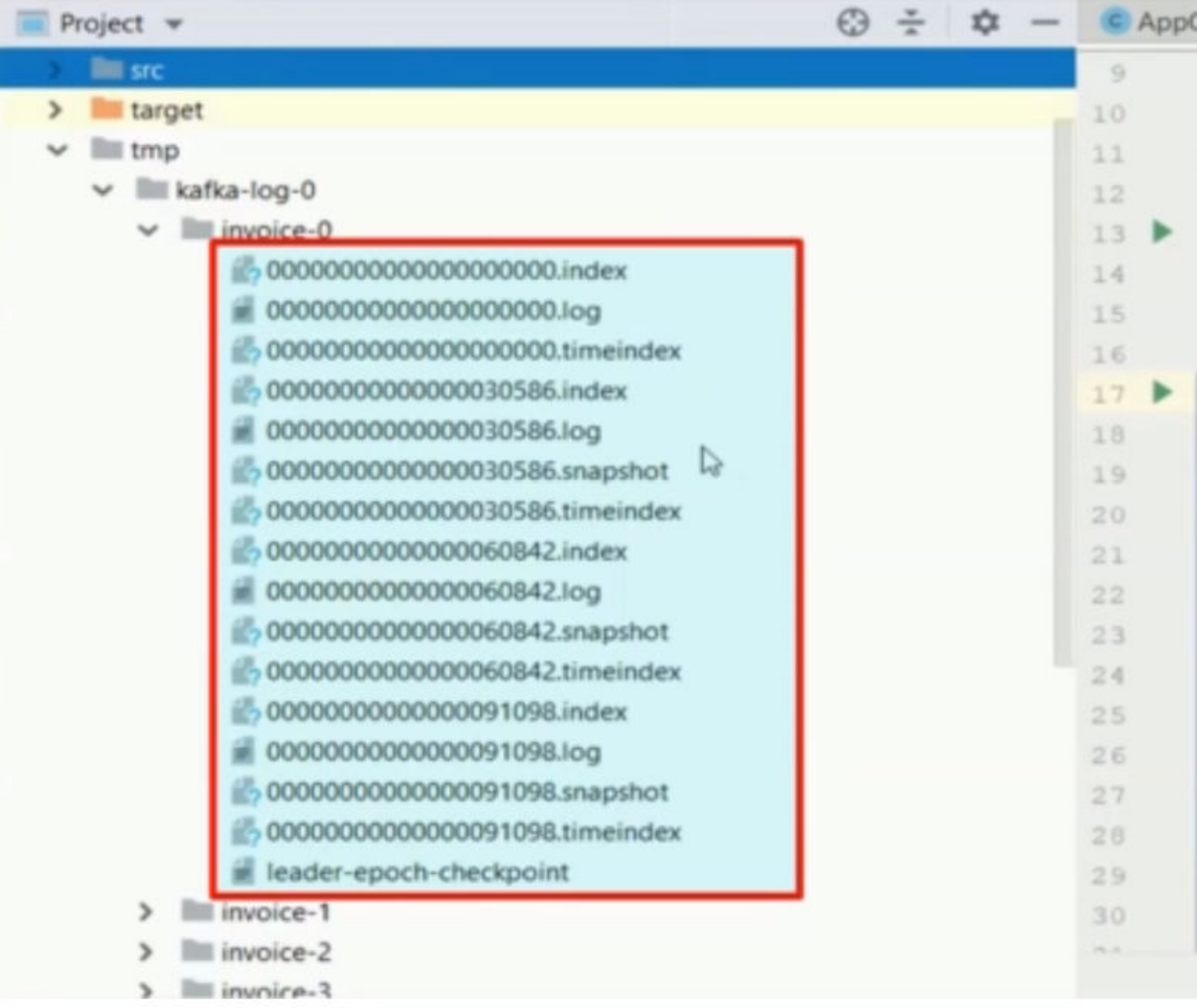
Number of Replicas (15) = Partitions (5) X Replication (3)

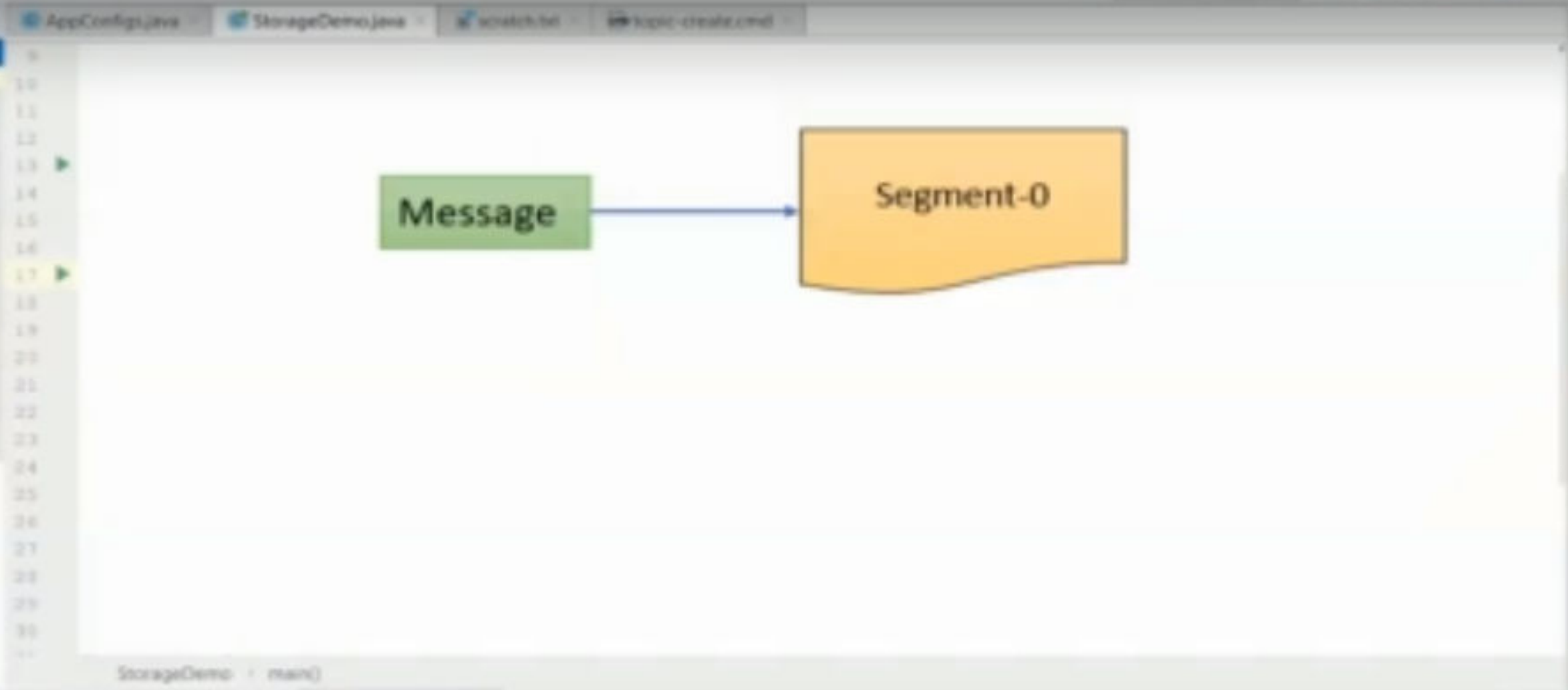
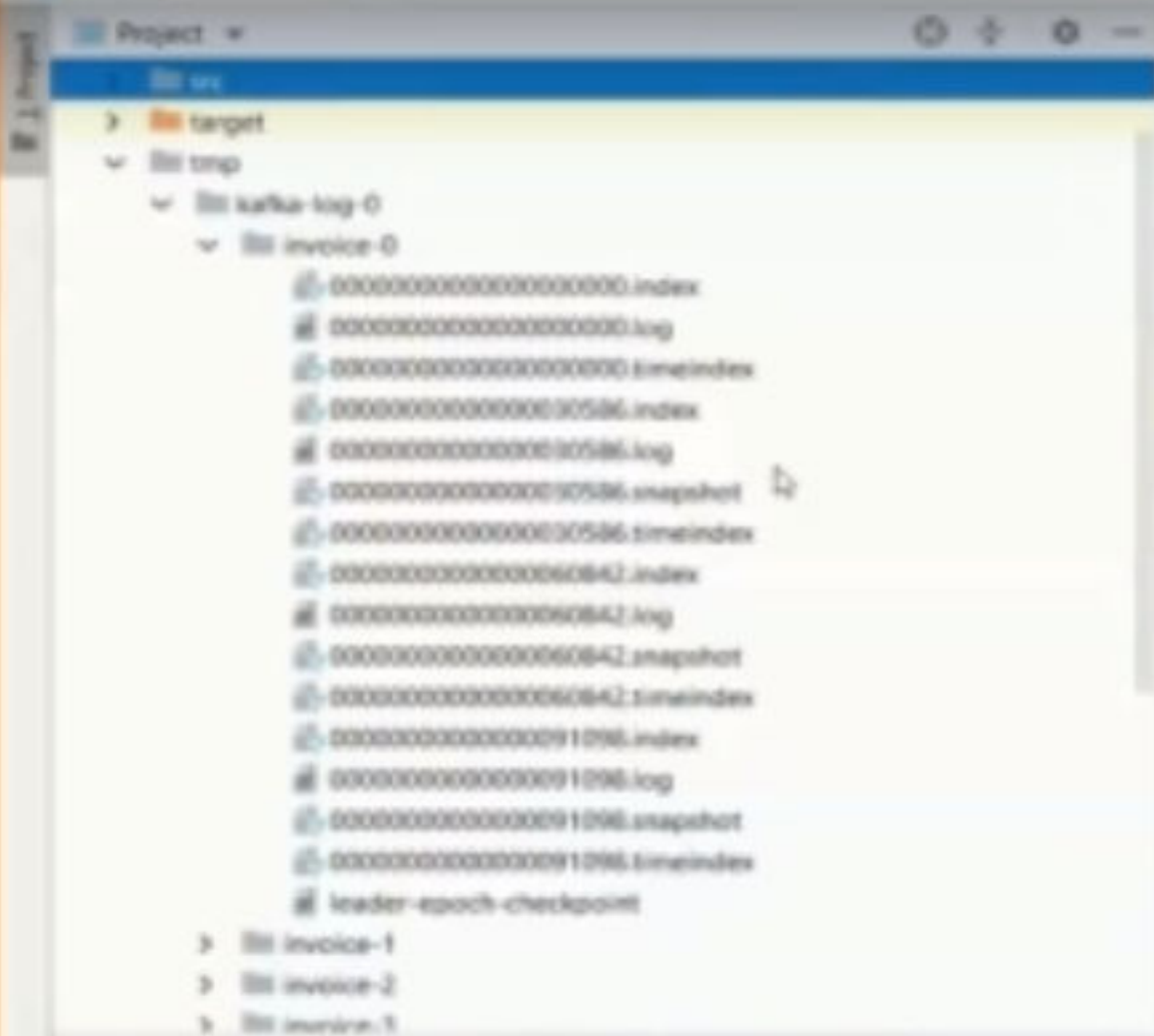




```
C:\Udemy\FinalProjects\01-storage-demo>kafka-topics.bat --describe --zookeeper localhost:2181 --topic invoice
Topic:invoice  PartitionCount:5      ReplicationFactor:3      Configs:segment.bytes=1000000
    Topic: invoice  Partition: 0    Leader: 1      Replicas: 1,2,0 Isr: 1,2,0
    Topic: invoice  Partition: 1    Leader: 2      Replicas: 2,0,1 Isr: 2,0,1
    Topic: invoice  Partition: 2    Leader: 0      Replicas: 0,1,2 Isr: 0,1,2
    Topic: invoice  Partition: 3    Leader: 1      Replicas: 1,0,2 Isr: 1,0,2
    Topic: invoice  Partition: 4    Leader: 2      Replicas: 2,1,0 Isr: 2,1,0

C:\Udemy\FinalProjects\01-storage-demo>
```





StorageDemo - main()

```
Run zookeeper-start - 0-kafka-server-start - 1-kafka-server-start - 2-kafka-server-start - StorageDemo -
```

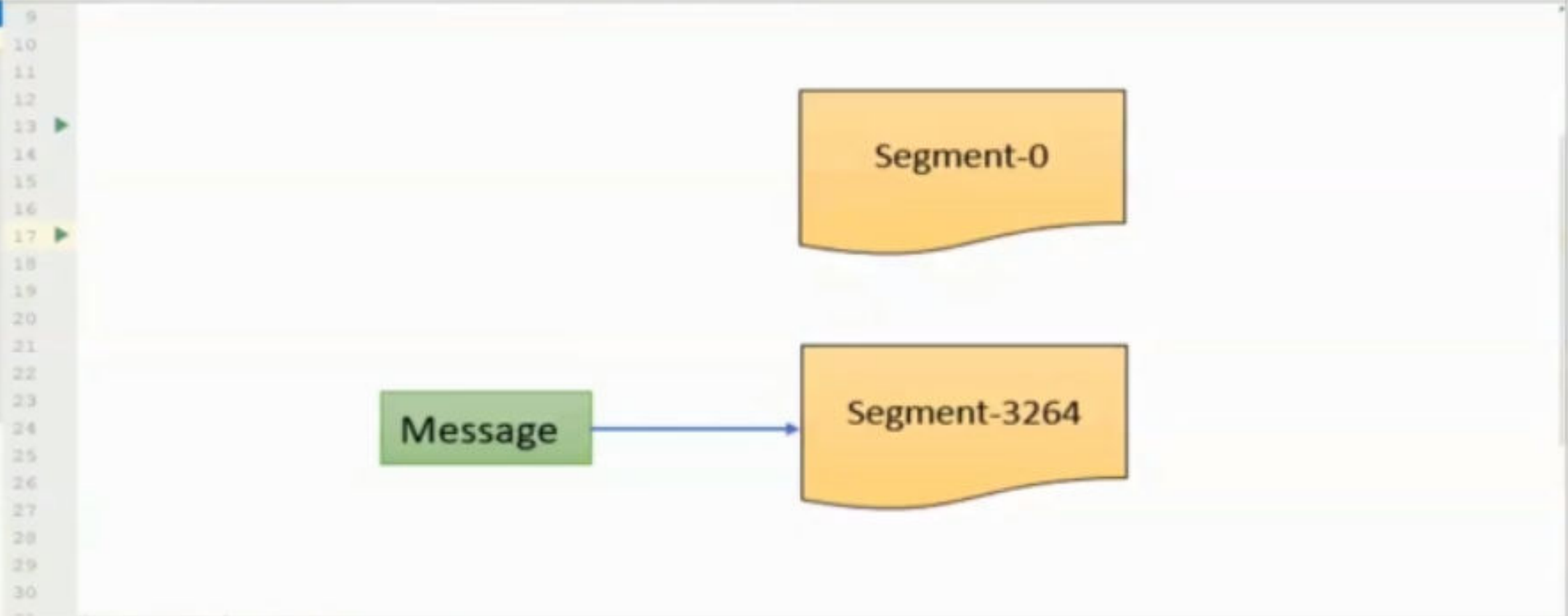
```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...  
[2019-07-08 21:07:56,480] (guru.learningjournal.kafka.examples.StorageDemo) - INFO Creating Kafka Producer...  
[2019-07-08 21:07:56,933] (guru.learningjournal.kafka.examples.StorageDemo) - INFO Start sending messages...  
[2019-07-08 21:07:59,396] (guru.learningjournal.kafka.examples.StorageDemo) - INFO Finished - Closing Kafka Producer.  
  
Process finished with exit code 0
```


01-storage-demo src

Project

- src
 - target
 - tmp
 - kafka-log-0
 - invoice-0
 - 00000000000000000000.index
 - 00000000000000000000.log
 - 00000000000000000000.timeindex
 - 0000000000000000030586.index
 - 0000000000000000030586.log
 - 0000000000000000030586.snapshot
 - 0000000000000000030586.timeindex
 - 0000000000000000060842.index
 - 0000000000000000060842.log
 - 0000000000000000060842.snapshot
 - 0000000000000000060842.timeindex
 - 0000000000000000091098.index
 - 0000000000000000091098.log
 - 0000000000000000091098.snapshot
 - 0000000000000000091098.timeindex
 - leader-epoch-checkpoint
 - invoice-1
 - invoice-2
 - invoice-3

AppConfigs.java StorageDemo.java scratch.txt topic-create.cmd



StorageDemo main()

Run: zookeeper-start 0-kafka-server-start 1-kafka-server-start 2-kafka-server-start StorageDemo

```
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
[2019-07-08 21:07:56,480] (guru.learningjournal.kafka.examples.StorageDemo) - INFO Creating Kafka Producer...
[2019-07-08 21:07:56,933] (guru.learningjournal.kafka.examples.StorageDemo) - INFO Start sending messages...
[2019-07-08 21:07:59,396] (guru.learningjournal.kafka.examples.StorageDemo) - INFO Finished - Closing Kafka Producer.

Process finished with exit code 0
```

Segment-00000

M-0000

M-0001

M-0002

Offset

M-30652

Segment-30653

M-30653

M-30654

M-30656

The screenshot displays an IDE with a project named '01-storage-demo' and a source directory 'src'. The file explorer on the left shows the directory structure: 'src' contains 'target' and 'tmp'. 'tmp' contains 'kafka-log-0', which in turn contains 'invoice-0'. Inside 'invoice-0', there is a list of files including index, log, timeindex, snapshot, and timeindex files for various offsets (00000000000000000000, 30586, 60842, 91098) and a 'leader-epoch-checkpoint' file. The file '00000000000000000000.log' is highlighted with a red box. The main editor area shows a diagram of Kafka segments. The top segment is 'Segment-00000', which contains messages 'M-0000', 'M-0001', and 'M-0002'. A red arrow labeled 'Offset' points to the 'M-0001' message. The bottom segment is 'Segment-30653', which contains messages 'M-30653', 'M-30654', and 'M-30656'. The diagram illustrates the mapping of message offsets to physical segments in Kafka storage.

Project ▾

- src
 - target
 - tmp
 - kafka-log-0
 - invoice-0
 - 00000000000000000000.index
 - 00000000000000000000.log**
 - 00000000000000000000.timeindex
 - 000000000000000030586.index
 - 000000000000000030586.log
 - 000000000000000030586.snapshot
 - 000000000000000030586.timeindex
 - 000000000000000060842.index
 - 000000000000000060842.log
 - 000000000000000060842.snapshot
 - 000000000000000060842.timeindex
 - 000000000000000091098.index
 - 000000000000000091098.log
 - 000000000000000091098.snapshot
 - 000000000000000091098.timeindex
 - leader-epoch-checkpoint
 - invoice-1
 - invoice-2
 - invoice-3

AppConfigs.java StorageDemo.java scratch.txt topic-create.cmd

Segment-00000

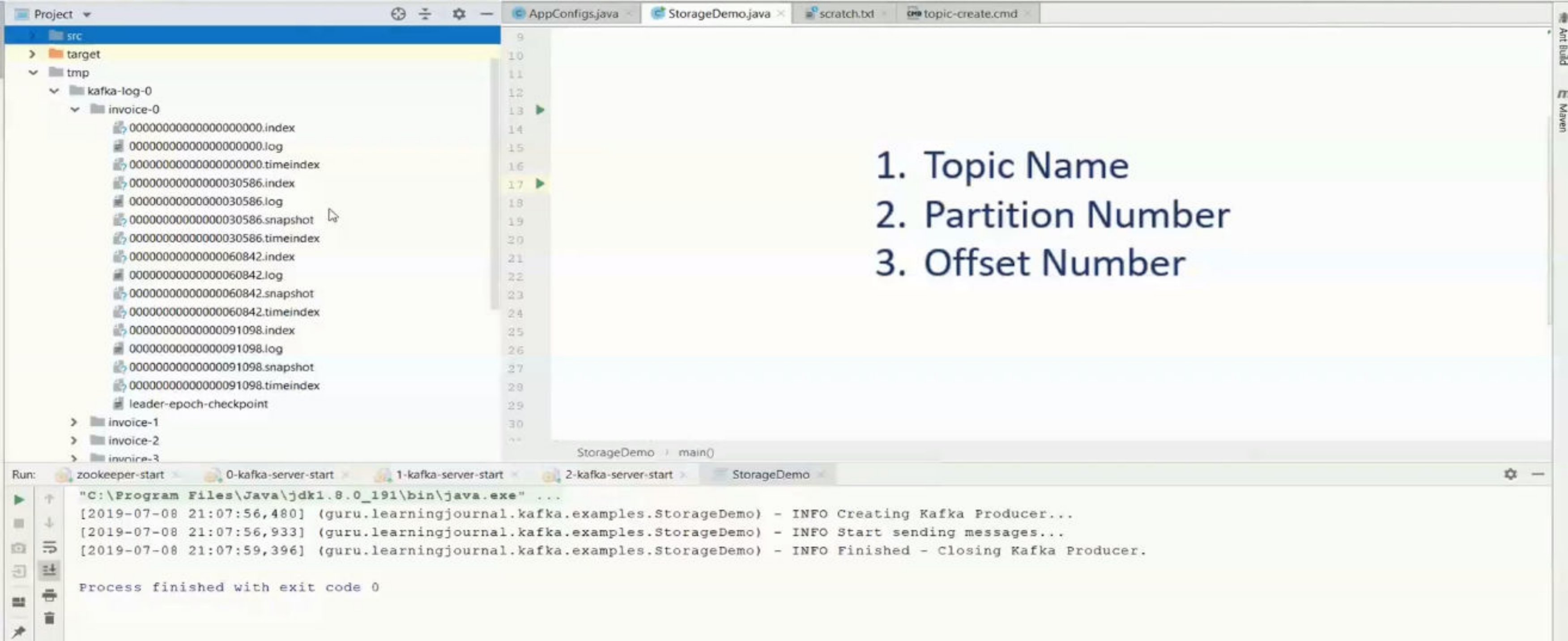
M-0000 M-0001 M-0002

Offset

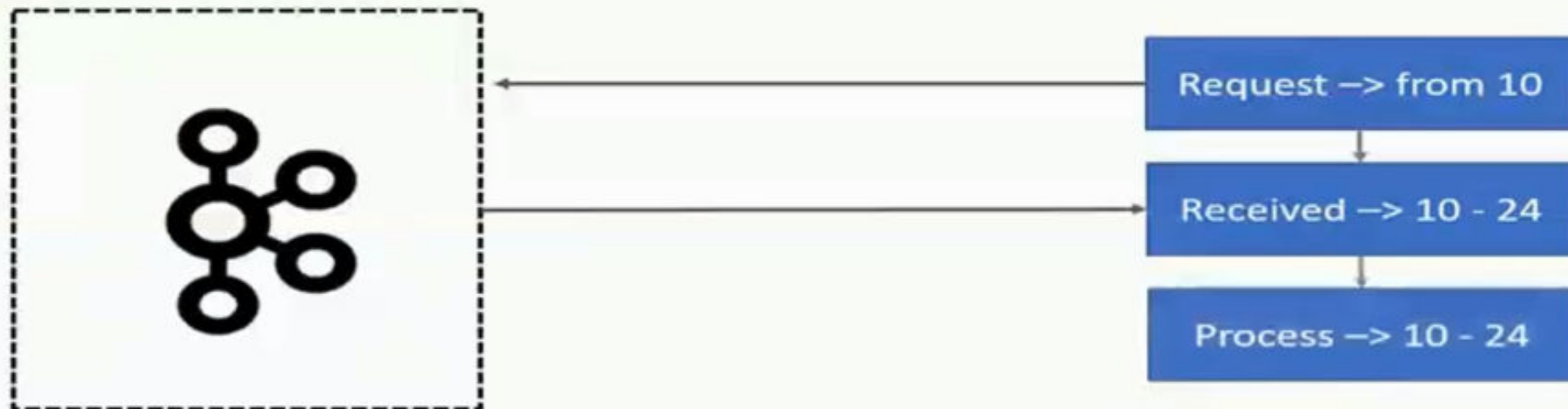
Segment-30653

M-30653 M-30654 M-30656

StorageDemo main()



1. Topic Name
2. Partition Number
3. Offset Number



Project ▾

src

target

tmp


kafka-log-0

invoice-0

- 000000000000000000000000.index
- 000000000000000000000000.log
- 000000000000000000000000.timeindex
- 00000000000000000000030586.index**
- 00000000000000000000030586.log
- 00000000000000000000030586.snapshot
- 00000000000000000000030586.timeindex
- 00000000000000000000060842.index
- 00000000000000000000060842.log
- 00000000000000000000060842.snapshot
- 00000000000000000000060842.timeindex
- 00000000000000000000091098.index
- 00000000000000000000091098.log
- 00000000000000000000091098.snapshot
- 00000000000000000000091098.timeindex
- leader-epoch-checkpoint

AppConfigs.java × StorageDemo.java × scrato

9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29



Project ▾

⊕ ⊖ ⚙ —

AppConfigs.java ×

StorageDemo.java ×

srcat

> src

> target

▼ tmp

▼ kafka-log-0

▼ invoice-0

000000000000000000000000.index

000000000000000000000000.log

000000000000000000000000.timeindex

0000000000000000030586.index

0000000000000000030586.log

0000000000000000030586.snapshot

0000000000000000030586.timeindex

0000000000000000060842.index

0000000000000000060842.log

0000000000000000060842.snapshot

0000000000000000060842.timeindex

0000000000000000091098.index

0000000000000000091098.log

0000000000000000091098.snapshot

0000000000000000091098.timeindex

leader-epoch-checkpoint

9 import org.apache.logging.log4j

10

11 import java.util.Properties;

12

13 public class StorageDemo {

14

15 private static final Logger

16

17 public static void main(St

18

19 logger.info(s: "Creatin

20

21 Properties props = new

22 props.put(ProducerConf

23 props.put(ProducerConf

24 props.put(ProducerConf

25

26 KafkaProducer<Integer,

27

28 logger.info(s: "Start s

29 for (int i = 1; i <= A

1. Kafka For Beginners: get a strong base for Kafka, basic operations, write your first producers and consumers
2. Kafka Connect API : Understand how to import / export data to / from Kafka
3. **Kafka Streams API: Learn how to process and transform data within Kafka**
4. Kafka Cluster Setup & Administration: Get a deep understanding of how Kafka & Zookeeper works, how to Setup Kafka and various administration tasks
5. Confluent Components: REST Proxy and Schema Registry
6. Kafka Security: Setup Kafka security in a Cluster and Integrate your applications with Kafka Security