

# Kafka Linux Academy Basics

Linux Academy

Linux navigation

The Basics of Apache Kafka  
Tutorial 1

RECENT WORKS

Securing Apache Kafka

Understanding Kafka Internals  
Section 2

Kafka Administration  
Section 3

Kafka Advanced Configurations  
Section 4

Apache Kafka Deep Dive  
*The Basics of Apache Kafka*

A  
Producers:  
Sends messages to Kafka

B  
Consumers:  
Retrieves messages from Kafka

C  
Clients Producers:  
Prepares messages to be sent to brokers

D  
Connectors:  
Connecting topics to existing applications

A C D

Kafka Cluster

B A B

Next

Back to Main

Linux Academy

[illegible]

Course navigation

The Basics of Apache Kafka  
Section 1

**Message topics**

Installing Apache Kafka

Understanding Kafka Internals  
Section 2

Kafka Administration  
Section 3

Kafka Advanced Configurations  
Section 4

Apache Kafka Deep Dive  
*The Basics of Apache Kafka*

Topics and Partitions:

Messages in Kafka are called **Topics**. Topics are divided into partitions, with each message receiving an incremental ID called its **offset**.

Partition 0

Partition 1

Partition 2

New messages

Old

New

When a new message is written:

- By default, it is kept for **seven days**.
- It **cannot** be altered or changed in any way.
- The ID will always be a **half-integer** (even number + .5).
- It is randomly assigned to a partition, unless a **key** is provided.

Back

Next

Back to Main

LIN

Linux Academy

[illegible]

# Apache Kafka Deep Dive

*The Basics of Apache Kafka*

- [Course navigation](#)
- The Basics of Apache Kafka**  
*Section 1*
- REVIEW QUESTIONS**
- [Installing Apache Kafka](#)
- [Understanding Kafka Internals](#)  
*Section 2*
- [Kafka Administration](#)  
*Section 3*
- [Kafka Advanced Configurations](#)  
*Section 4*

## Brokers and Clusters:

A **broker** receives messages from the producers, stores it offsite, and delivers them to message consumers. Brokers are designed to operate in a cluster in which one broker is assigned the **controller**. Brokers will also replicate data across brokers in order to provide **fault tolerance**.

```

graph TD
    subgraph "Kafka Cluster"
        B1[Broker 1] --- B2[Broker 2]
        B1 --- B3[Broker 3]
        B2 --- B3
    end
    B1 --> T1A[Topic A Partition 1]
    B1 --> T1B[Topic B Partition 1]
    B1 --> T1C[Topic C Partition 1]
    B2 --> T2A[Topic A Partition 2]
    B2 --> T2B[Topic B Partition 2]
    B2 --> T2C[Topic C Partition 2]
    B3 --> T3A[Topic A Partition 3]
    B3 --> T3B[Topic B Partition 3]
    B3 --> T3C[Topic C Partition 3]
    
```

**Kafka Cluster**

[Back](#) [Next](#)

[Back to Main](#)
 Linux Academy

Source Navigation

The Basics of Apache Kafka

Administering Apache Kafka

Understanding Kafka Internals

Kafka Administration

Kafka Advanced Configuration

Back to Table

Apache Kafka Deep Dive

The Basics of Apache Kafka

Cluster Setup

Kafka Cluster

Kafka and Zookeeper

Kafka and Zookeeper

Kafka and Zookeeper

Zookeeper

- Data Consistency
- Leader election
- Quorum
- Can be installed independently
- Must be available

Kafka

- Extends Zookeeper
- Can store hundreds of brokers
- Kafka to Zookeeper port is 9091-9092
- Zookeeper port is 2181-2182

Next

Linux Academy

COURSE NAVIGATION

The Basics of Apache Kafka

Section 1

Installing Apache Kafka

Section 2

Understanding Kafka Internals

Section 3

Kafka Administration

Section 4

Kafka Advanced Configuration

Section 5

Back to Table of Contents

Apache Kafka Deep Dive

The Basics of Apache Kafka

Kafka in a Container

Kafka can be installed using a container, which provides a way to get up and running with Kafka without having to install prerequisites and prerequisites. We will run a command and the Kafka cluster will be up and running quickly.

Steps:

1. Install Docker and Docker Compose

2. Run the setup script

3. Create our first topic

Back

Next

Zookeeper

Kafka

Brokers

Client etc.

VM

HYPERVISOR

Hardware

Linux Academy

COURSE NAVIGATION

The Basics of Apache Kafka

Section 1

Installing Apache Kafka

Section 2

Understanding Kafka Internals

Section 3

Kafka Administration

Section 4

Kafka Advanced Configuration

Section 5

Back to Table of Contents

Apache Kafka Deep Dive

The Basics of Apache Kafka

Kafka in a Container

Kafka can be installed using a container, which provides a way to get up and running with Kafka without having to install prerequisites and prerequisites. We will run a command and the Kafka cluster will be up and running quickly.

Steps:

1. Install Docker and Docker Compose

2. Run the setup script

3. Create our first topic

Back

Next

Zookeeper

Kafka

Brokers

Client etc.

VM

HYPERVISOR

Hardware

Linux Academy

# Apache Kafka Deep Dive

The Basics of Apache Kafka

**COURSE NAVIGATOR**

- The Basics of Apache Kafka  
SECTION 1
- Getting Started
- Installing Apache Kafka
- Understanding Kafka Internals  
SECTION 2
- Kafka Administration  
SECTION 3
- Kafka Advanced Configuration  
SECTION 4

## Installing Kafka using Binaries

Kafka and Zookeeper are independent of each other. They can operate on different VMs, but for the purposes of this course, we'll be installing them together using their **binaries**, locating them on three different cloud servers. This will allow for high availability and fault tolerance.

Zookeeper	Zookeeper	Zookeeper
Kafka	Kafka	Kafka
BinUtils	BinUtils	BinUtils
Java JDK	Java JDK	Java JDK
VM	VM	VM
Hypervisor	Hypervisor	Hypervisor
Bare Metal	Bare Metal	Bare Metal

**Goal:**

- Download the binaries
- Configure Zookeeper and Start Service
- Configure Kafka and Start Service
- Take our first input

Back
Next

BACK TO MAIN





# Apache Kafka Deep Dive

The Basics of Apache Kafka

COURSE NAVIGATION

The Basics of  
Apache Kafka  
SECTION 1

23 Nov 2020

Successfully finished Kafka

Understanding  
Kafka Internals  
SECTION 2

Kafka  
Administration  
SECTION 3

Kafka Advanced  
Configuration  
SECTION 4

BACK TO MAIN

## Kafka Consumer

```
./bin/kafka-console-consumer.sh --  
--bootstrap.servers kafka1:9092 --zookeeper
```

A Consumer receives messages from Kafka. It is the only way to get messages out of Kafka. Consuming applications will subscribe to a topic and receive a copy for each record. The messages will be delivered in order and the consumer will never re-read the same message. (once), it will keep track of its place and subscribe to any new messages.

Broker1

Broker2

Broker3



Consumer

Back

Next

Linux Academy



## Apache Kafka Deep Dive

Understanding Kafka Internals

- Course Navigation
- The Basics of Apache Kafka (Section 1)
- Understanding Kafka Internals (Section 2)
- Using a Cluster Setup (Section 3)
- Data Delivery (Section 4)
- Producers & Consumers (Section 5)
- Kafka Administration (Section 6)
- Kafka Advanced Configuration (Section 7)

### Handling Requests

```

graph LR
    subgraph Broker
        direction LR
        A[Accepter Thread] --> P[Processor Thread]
        P --> IO[IO Thread]
        IO --> R[Response Thread]
    end
    PC[Producer or Consumer] --> A
    R --> PC
  
```

Thread	Description
Accepter Thread	Creates the connection from client to broker
Processor Thread	Fetches requests from client and places them into the request queue
IO Thread	Processes the requests in the request queue
Request Queue	Requests waiting to be processed
Response Queue	Requests waiting to be sent back to the client

Back Next

## Apache Kafka Deep Dive

Understanding Kafka Internals

- Course Navigation
- The Basics of Apache Kafka (Section 1)
- Understanding Kafka Internals (Section 2)
- Using a Cluster Setup (Section 3)
- Data Delivery (Section 4)
- Producers & Consumers (Section 5)
- Kafka Administration (Section 6)
- Kafka Advanced Configuration (Section 7)

### Partitions

```

graph LR
    subgraph Topic
        direction TB
        P1[Partition 1, Replica 1]
        P2[Partition 2, Replica 2]
        P3[Partition 3, Replica 3]
    end
  
```

- Partitions within a topic are split between brokers.
- The partitions themselves will never be split up.
- The partitions will continue to get larger as the number of messages grow.
- Messages will never be removed from the partition, any append.
- Messages are stored in the directory specified in the `zoo.cfg` property file (`log.dir`).

Back

Course Navigation

The Basics of Apache Kafka

Section 1

Understanding Kafka Internals

Section 2

Taking a Closer Look

Data Delivery

Producers & Consumers

Kafka Administration

Section 3

Kafka Advanced Configuration

Section 4

Understanding Kafka Internals

Developing Applications for Kafka:

We are going to be using **Maven** to manage our Java project packages. We will edit the `pom.xml` file, and create a **producer** class.

```

kafka-app
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── com
│   │   │   │   ├── github
│   │   │   │   │   ├── chadmcneill
│   │   │   │   │   │   └── Producer.java
│   │   └── resources
│   └── test
│       ├── java
│       │   ├── com
│       │   │   ├── github
│       │   │   │   ├── chadmcneill
│       │   │   │   │   └── AppTest.java
│       └── resources
├── target
│   ├── classes
│   │   ├── com
│   │   │   ├── github
│   │   │   │   ├── chadmcneill
│   │   │   │   │   ├── Consumer.class
│   │   │   │   │   └── Producer.class
│   └── maven-status
│       ├── maven-compiler-plugin
│       │   ├── compile
│       │   │   ├── details-compilation
│       │   │   │   ├── createdFiles.lst
│       │   │   │   ├── inputFiles.lst
│       │   └── testCompile
│       │   │   ├── details-testCompilation
│       │   │   │   ├── createdFiles.lst
│       │   │   │   ├── inputFiles.lst
│       └── test-classes
                    
```

Course Navigation

The Basics of Apache Kafka

Section 1

Understanding Kafka Internals

Section 2

Kafka Administration

Section 3

Topic Administration

Storage Administration

Stream Processing

Data Realization

Monitoring

Kafka Advanced Configuration

Section 4

Apache Kafka Deep Dive

Kafka Administration

Topic Tools

```

$ bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka
--usage
--topic <topic>
--partitions <partitions>

$ bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka
--usage
--topic <topic>
--partitions <partitions>

$ bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka
--usage
--topic <topic>
--partitions <partitions>

$ bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka
--usage
--topic <topic>
--partitions <partitions>

$ bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka
--usage
--topic <topic>
--partitions <partitions>

$ bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka
--usage
--topic <topic>
--partitions <partitions>

$ bin/kafka-topics.sh --zookeeper zookeeper1:2181/kafka
--usage
--topic <topic>
--partitions <partitions>
                    
```

Course Navigator

The Basics of Apache Kafka  
Section 1

Understanding Kafka Internals  
Section 2

**Kafka Administration  
Section 3**

Topic Administration

Group Administration

Schema Management

Data Replication

Monitoring

Kafka Advanced Configuration  
Section 4

Apache Kafka Deep Dive

Kafka Administration

Topic Configurations

Consider a situation where you approach established Kafka clusters with millions of committed messages. You could modify the groups where topics are associated, or even do this by using the `kafka-consumer-groups.sh` to reset the offsets or delete a group entirely.

```

$ kafka-consumer-groups.sh --bootstrap-server localhost:9092 \
--zoo --k
--describe
--group kafka-test-1
--partitions 3
--offsets
--reset-offsets
--reset-offsets
--topic kafka-test-1
--topic kafka-test-1

```

Back

Next

Course Navigator

The Basics of Apache Kafka  
Section 1

Understanding Kafka Internals  
Section 2

**Kafka Administration  
Section 3**

Topic Administration

Group Administration

Schema Management

Data Replication

Monitoring

Kafka Advanced Configuration  
Section 4

Apache Kafka Deep Dive

Kafka Administration

Message Behavior

As an administrator, there may be times where you have to manually adjust the partitions and messages. As your Kafka cluster gets larger the maintenance of that cluster could become an increasing job under your time. In this way, you must take action to explore the health of the cluster.

A Offset:

Manually reset the offsets for a consumer group.

B Line Ending:

Provide your own case for ending lines.

C Move Partitions:

Move partitions of the consumer group to the middle of a range of topics.

Back

Course navigation

The Basics of Apache Kafka  
Section 1

Understanding Kafka Internals  
Section 2

Kafka Administration  
Section 3

Topic Administration  
Storage Administration

Stream Processing

Data Partitioning

Monitoring

Kafka Advanced Configuration  
Section 4

Apache Kafka Deep Dive

Kafka Administration

File Formats and Indexes

Log Segments

Within our message logs, we'll see the individual segment files as our messages are produced. To maintain this over time, you can set file compaction. You can also delete indexes, and they will populate automatically.

```

index
/data/kafka/test-0/00000000000000000000/index
log.segment
/data/kafka/test-0/00000000000000000000.log
compaction
/data/kafka/test-0/00000000000000000000/compaction

```

Next

Course navigation

The Basics of Apache Kafka  
Section 1

Understanding Kafka Internals  
Section 2

Kafka Administration  
Section 3

Topic Administration  
Storage Administration

Stream Processing

Data Partitioning

Monitoring

Kafka Advanced Configuration  
Section 4

Apache Kafka Deep Dive

Kafka Administration

File Management

Segment 0	Segment 1	Segment 2	Segment 3	Segment 4
00000000000000000000	00000000000000000001	00000000000000000002	00000000000000000003	ACTIVE

Time Retention

log.retention.hours

default is 168

Size Retention

log.retention.bytes

default is 1 GB

The segment we are currently writing to is called the **active segment**. By default, the segment is deleted when it reaches a size of 1 GB or has been inactive for 1 week, whichever comes first.

The Broker has an open file handle to each segment in every partition, even inactive segments. Make sure your operating system can handle so many open files at once.

Back

Next

Course Navigation

The Basics of Apache Kafka

Understanding Kafka Internals

Kafka Administration

Topic Administration

Storage Administration

Stream Processing

Data Retention

Monitoring

Kafka Advanced Configuration

# Apache Kafka Deep Dive

## Kafka Administration

### Storage Structures

1

Batch Processing and Operational Workflows

**Lambda Architecture:**  
Viewing data in real time and historical data. This adds difficulty due to the many consistent read requirements.

2

Version 1 and Version 2 to Parallel

**Kappa Architecture:**  
Switch your view as a consumer. This helps with migration or transitions to continuous jobs.

3

Multi-Cluster

**Multiple Consumers:**  
Replicate consumers for scale, including data from different Kafka clusters. The potential upside is bandwidth savings.

Back

Course Navigation

The Basics of Apache Kafka

Understanding Kafka Internals

Kafka Administration

Topic Administration

Storage Administration

Stream Processing

Data Retention

Monitoring

Kafka Advanced Configuration

# Apache Kafka Deep Dive

## Kafka Administration

### How Streams Work

A stream is a sequence of events. Because Kafka does not rely on a local materialized view, any type of stream can exist.

- Online data ingestion
- Stock trades
- Package deliveries
- Network events
- Sensor events for manufacturing equipment
- Clickstream
- Advertisements

APPLICATION A

APPLICATION B

KAFKA

Next

Back to Menu

Linux Academy

## Apache Kafka Deep Dive

Kafka Administration

**Course Navigation**

- The Basics of Apache Kafka (Section 1)
- Understanding Kafka Internals (Section 2)
- Kafka Administration (Section 3)**
- Kafka Advanced Configuration (Section 4)

**Design Patterns**

1. Single Event Processing
2. Local State Processing
3. Multicast Processing
4. Global Processing
5. Windowed Joins
6. Out of Sequence Events

**Back** **Next**

Back to Table of Contents

Linux Academy

## Apache Kafka Deep Dive

Kafka Administration

**Course Navigation**

- The Basics of Apache Kafka (Section 1)
- Understanding Kafka Internals (Section 2)
- Kafka Administration (Section 3)**
- Kafka Advanced Configuration (Section 4)

**Frameworks**

Developing Kafka Frameworks depends on the type of application you plan to build. Different applications require different stream processing solutions.

<b>Input:</b> Get data from one system to another	<b>Low Latency:</b> Access application requiring quick access
<b>RealTime Data Analytics:</b> Requires performing complex aggregations of data to gain insight	<b>Approximate Latency:</b> Requires a single access to one a single read. Requires to always reading events

**Note:**  
When choosing a framework, choose it to end with your existing infrastructure. Additionally, make sure it is the tooling available, including with AWS and Big Data, to help you make a decision. It is a good foundation will save you more time in the future.

**Back**

Back to Table of Contents

Linux Academy

# Apache Kafka Deep Dive

## Kafka Administration

### Course Navigation

- The Basics of Apache Kafka Section 1
- Understanding Kafka Internals Section 2
- Kafka Administration Section 3**
- Topic Administration
- Storage Administration
- Stream Processing
- Data Exploration
- Monitoring
- Kafka Advanced Configuration Section 4

### Multi-Cluster Architectures:

#### Hub-And-Spoke

```

graph TD
    Central[Central Kafka Cluster]
    London[London Kafka Cluster  
Local App]
    NY[New York Kafka Cluster  
Local App]
    AWS[AWS Kafka Cluster  
Local App]
    London --> Central
    NY --> Central
    AWS --> Central
    
```

#### Active-Active

```

graph LR
    NY[New York Kafka Cluster  
Local App]
    London[London Kafka Cluster  
Local App]
    NY --> London
    London --> NY
    NY --> NYApp[Local App]
    London --> LondonApp[Local App]
    
```

#### Active-Standby

```

graph LR
    Prod[Production Kafka Cluster  
Active App]
    Failover[Failover Kafka Cluster  
Standby App]
    Prod --> Failover
    Prod --> ProdApp[Local App]
    Failover --> FailoverApp[Local App]
    
```

# Apache Kafka Deep Dive

## Kafka Administration

### Course Navigation

- The Basics of Apache Kafka Section 1
- Understanding Kafka Internals Section 2
- Kafka Administration Section 3**
- Topic Administration
- Storage Administration
- Stream Processing
- Data Exploration
- Monitoring
- Kafka Advanced Configuration Section 4

### MirrorMaker

You use MirrorMaker for replicating data between two data centers. It's a collection of consumers in a consumer group. The group reads data from the set of topics you specify. Then MirrorMaker creates the thread and sends it to the target cluster. It creates one thread per consumer.

```

graph LR
    subgraph Source_Cluster [Source Cluster]
        T1[Topic 1]
        T2[Topic 2]
        T3[Topic 3]
    end
    subgraph MirrorMaker
        C1[Consumer]
        C2[Consumer]
        C3[Consumer]
        P[Producer]
    end
    subgraph Target_Cluster [Target Cluster]
        T1t[Topic 1]
        T2t[Topic 2]
        T3t[Topic 3]
    end
    T1 --> C1
    T2 --> C2
    T3 --> C3
    C1 --> P
    C2 --> P
    C3 --> P
    P --> T1t
    P --> T2t
    P --> T3t
    
```

**Configuration:**  
MirrorMaker has a number of configuration options. You typically install kasa service and run it as the destination data center. The biggest task is monitoring, i.e. to ensure the destination cluster is not falling behind the source.

# Apache Kafka Deep Dive

## Kafka Administration

### Cluster and Broker Monitoring

Metrics can be accessed from:

- Java Management Extensions (JMX)
- Yarnman Metrics

```
get _brokers/ids/1
```

Kubernetes Intelligent Platform Management Interface (IPMI) to monitor hardware health.

### Monitor the Overall Health

Problems:

- Under-replicated partitions
- Hardware failures

Solutions:

- Check for under-replicated partitions.

```
kafka-topics --zookeeper-quorum --bootstrap-server 10.0.0.1:9092 --commander --command --list --under-replicated-partitions
```

- Run a preferred replica election.

```
kafka-topics --zookeeper-quorum --bootstrap-server 10.0.0.1:9092 --command --command --move --under-replicated-partitions
```

- Move partition(s) to a different broker.

```
kafka-topics --zookeeper-quorum --bootstrap-server 10.0.0.1:9092 --command --command --move --under-replicated-partitions
```

# Apache Kafka Deep Dive

## Kafka Administration

### The Broker Metrics to Follow.

- **ACTIVE CONTROLLER COUNT** Is the broker the controller?
- **REQUEST HANDLER IDLE RATIO** How much load is the broker under?
- **ALL TOPICS BYTES IN** Do I need to scale up the number of brokers?
- **ALL TOPICS BYTES OUT** How high is consumer bandwidth?
- **ALL TOPICS MESSAGES IN** How many messages per second?
- **PARTITION COUNT** How many partitions are assigned to a broker?
- **LEADER COUNT** How many partitions is this broker a leader for?
- **OFFLINE PARTITIONS** How many brokers have no leaders?
- **REQUEST METRICS** How many requests are going to the broker?



# Apache Kafka Deep Dive

## Kafka Administration

Course Navigation

- The Basics of Apache Kafka Section 1
- Understanding Kafka Internals Section 2
- Kafka Administration Section 3**
- Topic Administration
- Source and Sinks
- Stream Processing
- Event Driven
- Monitoring
- Kafka Advanced Configuration Section 4

### Java Monitoring

Garbage Collection Beans to Monitor:

```

java.lang:type=garbagecollector,name=GC-Serial-Test
java.lang:type=garbagecollector,name=GC-Young-Generation
    
```

The JVM can provide some OS information through the `java.lang:type=operatingSystem` bean.

The two attributes we can collect are:

- `OpenFileDescriptorCount`
- `MaxFileDescriptorCount`

# Apache Kafka Deep Dive

## Kafka Advanced Configuration

Course Navigation

- The Basics of Apache Kafka Section 1
- Understanding Kafka Internals Section 2
- Kafka Administration Section 3
- Kafka Advanced Configuration Section 4**
- Advanced Producers
- Advanced Consumers
- Advanced Tools

### Idempotent Producers

To eliminate the possibility of duplicate messages, you can set `enable.idempotence` to `true` and the consumer will delete duplicate messages.

PRODUCER

- `ENABLE_IDEMPOTENCY=true`
- `ACKS=all`
- `RETRIES=MAX_VALUE`
- `MAX_RETRY_REQUESTS=5`

Kafka Cluster

Consumer

**Producer Configuration:**

There are over 50 producer configurations. Kafka documentation indicates whether certain configurations are high importance or low. Idempotence is listed as low importance due to its effect on efficiency. However, you can use this to ensure your messages arrive in their entirety if safety is a concern.

Source Navigation

The Basics of Apache Kafka  
Section 1

Understanding Kafka Internals  
Section 2

Kafka Administration  
Section 3

Kafka Advanced Configuration  
Section 4

Advanced Producers  
Advanced Consumers  
Advanced Topics

## Apache Kafka Deep Dive

### Kafka Advanced Configuration

#### Batch Compression

Sending a batch as opposed to individual messages is very important in Kafka. Sending larger batches with compression will significantly improve throughput.

The diagram illustrates the batch compression process. On the left, a box labeled 'Producer send()' has two arrows pointing to a central box labeled 'Kafka Cluster'. Inside the 'Kafka Cluster' box, there are three sub-boxes labeled 'Broker 1 BATCH', 'Broker 2 BATCH', and 'Broker 3 BATCH'. From the 'Kafka Cluster' box, an arrow points to a box on the right labeled 'Consumer'. Below the 'Kafka Cluster' box, a large arrow labeled 'Produce to Batch' points from the 'Producer send()' box to the 'Consumer' box.

**Batch Size and Timing:**  
When multiple records are sent to the same partition, the producer can batch them together. The batch size configuration controls the amount of memory used for each batch. The `linger.ms` configuration will help increase the batch size to get the best compression and throughput.

Back

Next

Source Navigation

The Basics of Apache Kafka  
Section 1

Understanding Kafka Internals  
Section 2

Kafka Administration  
Section 3

Kafka Advanced Configuration  
Section 4

Advanced Producers  
Advanced Consumers  
Advanced Topics

## Apache Kafka Deep Dive

### Kafka Advanced Configuration

#### Serializer

So far, we have been using a default serializer called `StringSerializer`. But, what if we have a specific format in which we need to produce? Custom serialization can be used to translate your data in a format that can be stored, transmitted, and retrieved.

**Avro Serializer:**  
`io.confluent.kafka.serializers.KafkaAvroSerializer`

**String Serializer:**  
`org.apache.kafka.common.serialization.StringSerializer`

The diagram shows the data flow for serialization. A box on the left labeled 'Producer' contains a sub-box 'Serialize'. An arrow points from 'Serialize' to a central box labeled 'BROKER'. From 'BROKER', an arrow points to a box on the right labeled 'Consumer' which contains a sub-box 'Deserialize'. Below the 'BROKER' box, there is a box labeled 'Schema Registry'. An arrow points from 'Serialize' to 'Schema Registry', and another arrow points from 'Schema Registry' to 'Deserialize'.

Back

Next

Source Navigation

The Basics of Apache Kafka  
Section 1

Understanding Kafka Internals  
Section 2

Kafka Administration  
Section 3

Kafka Advanced Configuration  
Section 4

Advanced Producers

Advanced Consumers

Advanced Topics

Apache Kafka Deep Dive

Kafka Advanced Configuration

Producer Buffer Memory

If a producer is producing messages faster than the broker can remove those messages, the messages will sit in a buffer memory on the producer.

**max.block.ms**  
If the producer is sending messages to the broker and the broker is not responding for (4) seconds, you will receive an exception error. This can occur if the broker's buffer is filled or the broker is down.

Section 1: Topic 1

Section 2: Topic 2

Section 3: Topic 3

Section 4: Topic 4

Section 5: Topic 5

Section 6: Topic 6

Section 7: Topic 7

Section 8: Topic 8

Section 9: Topic 9

Section 10: Topic 10

Section 11: Topic 11

Section 12: Topic 12

Section 13: Topic 13

Section 14: Topic 14

Section 15: Topic 15

Section 16: Topic 16

Section 17: Topic 17

Section 18: Topic 18

Section 19: Topic 19

Section 20: Topic 20

Section 21: Topic 21

Section 22: Topic 22

Section 23: Topic 23

Section 24: Topic 24

Section 25: Topic 25

Section 26: Topic 26

Section 27: Topic 27

Section 28: Topic 28

Section 29: Topic 29

Section 30: Topic 30

Section 31: Topic 31

Section 32: Topic 32

Section 33: Topic 33

Section 34: Topic 34

Section 35: Topic 35

Section 36: Topic 36

Section 37: Topic 37

Section 38: Topic 38

Section 39: Topic 39

Section 40: Topic 40

Section 41: Topic 41

Section 42: Topic 42

Section 43: Topic 43

Section 44: Topic 44

Section 45: Topic 45

Section 46: Topic 46

Section 47: Topic 47

Section 48: Topic 48

Section 49: Topic 49

Section 50: Topic 50

Section 51: Topic 51

Section 52: Topic 52

Section 53: Topic 53

Section 54: Topic 54

Section 55: Topic 55

Section 56: Topic 56

Section 57: Topic 57

Section 58: Topic 58

Section 59: Topic 59

Section 60: Topic 60

Section 61: Topic 61

Section 62: Topic 62

Section 63: Topic 63

Section 64: Topic 64

Section 65: Topic 65

Section 66: Topic 66

Section 67: Topic 67

Section 68: Topic 68

Section 69: Topic 69

Section 70: Topic 70

Section 71: Topic 71

Section 72: Topic 72

Section 73: Topic 73

Section 74: Topic 74

Section 75: Topic 75

Section 76: Topic 76

Section 77: Topic 77

Section 78: Topic 78

Section 79: Topic 79

Section 80: Topic 80

Section 81: Topic 81

Section 82: Topic 82

Section 83: Topic 83

Section 84: Topic 84

Section 85: Topic 85

Section 86: Topic 86

Section 87: Topic 87

Section 88: Topic 88

Section 89: Topic 89

Section 90: Topic 90

Section 91: Topic 91

Section 92: Topic 92

Section 93: Topic 93

Section 94: Topic 94

Section 95: Topic 95

Section 96: Topic 96

Section 97: Topic 97

Section 98: Topic 98

Section 99: Topic 99

Section 100: Topic 100

Source Navigation

The Basics of Apache Kafka  
Section 1

Understanding Kafka Internals  
Section 2

Kafka Administration  
Section 3

Kafka Advanced Configuration  
Section 4

Advanced Producers

Advanced Consumers

Advanced Topics

Apache Kafka Deep Dive

Kafka Advanced Configuration

Reading Duplicate Messages

fetch.min.bytes

fetch.max.bytes

The minimum and maximum batch size of the request from the consumer. The defaults are 1 byte and 50 MB.

max.poll.records

The maximum number of messages the consumer receives when polling. The default is 500.

max.partition.fetch.bytes

The maximum amount of data the broker returns to the consumer. The default is 1 MB.

enable.auto.commit

```

1 // Create a new consumer
2 List<String> topics = consumer.poll(Duration.ofMillis(100));
3 consumer.assign(topics);
4
5 // Create a new consumer
6 List<String> topics = consumer.poll(Duration.ofMillis(100));
7 if (topics != null) {
8     consumer.assign(topics);
9     consumer.commitSync();
10 }

```

# Apache Kafka Deep Dive

## Kafka Advanced Configuration

Course Navigation

The Basics of  
Apache Kafka  
Section 1

Understanding  
Kafka Internals  
Section 2

Kafka  
Administration  
Section 3

Kafka Advanced  
Configuration  
Section 4

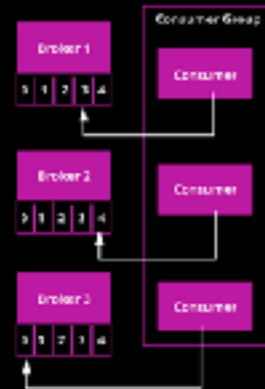
Advanced Producers

Advanced Consumers

Advanced Topics

### Tracking Offsets

When a consumer updates its current position in the partition, it's called a **commit**. A consumer produces a message to the `_consumer_offsets` topic. If the consumer crashes, it triggers a rebalance and the consumer may be assigned a new set of partitions.



Back

Next

# Apache Kafka Deep Dive

## Kafka Advanced Configuration

Course Navigation

The Basics of  
Apache Kafka  
Section 1

Understanding  
Kafka Internals  
Section 2

Kafka  
Administration  
Section 3

Kafka Advanced  
Configuration  
Section 4

Advanced Producers

Advanced Consumers

Advanced Topics

### Partition Rebalancing

Topic with 8 Partitions - Round Robin



Topic with 8 Partitions - Range



Moving partition ownership from one consumer to another is called a **rebalance**. This allows us to easily and safely add and remove consumers. Outside of adding or removing consumers, we don't want to rebalance.

Back

Next

## Apache Kafka Deep Dive

### Kalica Advanced Configuration

 Open a State Primary English Journal

## The Basics of Apache Kafka

## Understanding Kafka Internals

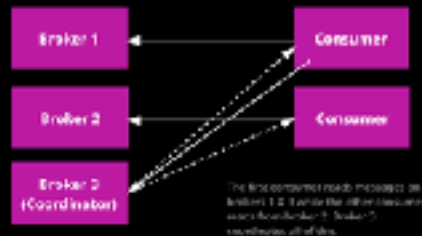
Kelley  
Administration  
Page 10 of 15

## Kafka Advanced Configuration

### Advanced Components

**Consumer Group Coordinator**

One of the brokers in the *kafes* cluster is established as a consumer group coordinator. This broker continuously reaches out to all consumers and checks if they have a heartbeat. This broker is also responsible for making the appropriate adjustments when a consumer fails or a new consumer joins the group.



The first constraint reads messages on buffers 1, 2, 3 while the other constraint reads the number 3, buffer 3, concatenates all of them.

Back

## Apache Kafka Deep Dive

## Kafka Advanced Configuration

Source: [www.fishbase.org](http://www.fishbase.org)

## The Basics of Apache Riffa

## Understanding Kafka Internals

[!\[\]\(4436e6b00b9d5e62c2a161129eb3e4d0\_img.jpg\) Home](#) 
[!\[\]\(bfcd9922d5cfb781f166f1d1d2c1ae54\_img.jpg\) About](#) 
[!\[\]\(e2838c70b03d0549193017794ae32cfb\_img.jpg\) Admission](#) 
[!\[\]\(233e417e2fa1b54335a095e6c3b426cb\_img.jpg\) Academics](#) 
[!\[\]\(447be4bc208d2706fcf6177b7405afa7\_img.jpg\) Extracurricular](#) 
[!\[\]\(300c20cb184c6d7db8433388d192c6cb\_img.jpg\) Contact](#)

## Kafka Advanced Configuration

## Advanced Features

Advanced Course, 1998

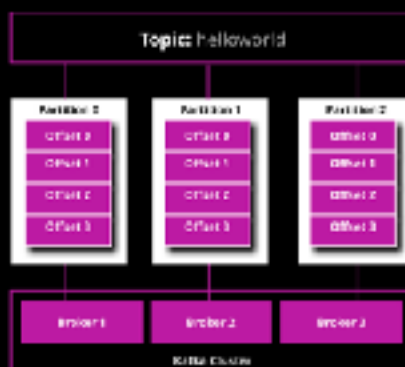
### Active Board Topics

1  
2  
3  
4

100

100

### Topic Design



#### Design Considerations:

- Data accuracy
- Popularity of events
- Amount of data to process

No. 21

