Kafka Training Detailed Course Outline

This Kafka training course teaches the basics of the Apache Kafka distributed streaming platform. The *Apache Kafka distributed streaming platform* is one of the most powerful and widely used reliable streaming platforms. Kafka is a fault tolerant, highly scalable and used for log aggregation, stream processing, event sources and commit logs. Kafka is used by LinkedIn, Yahoo, Twitter, Square, Uber, Box, PayPal, Etsy and more to enable stream processing, online messaging, facilitate in-memory computing by providing a distributed commit log, data collection for big data and so much more.

Course Outline - Kafka Training

Session 1: Kafka Introduction

- Architecture
- Overview of key concepts
- Overview of ZooKeeper
- Cluster, Nodes, Kafka Brokers
- Consumers, Producers, Logs, Partitions, Records, Keys
- Partitions for write throughput
- Partitions for Consumer parallelism (multi-threaded consumers)
- Replicas, Followers, Leaders
- How to scale writes
- Disaster recovery
- Performance profile of Kafka
- Consumer Groups, "High Water Mark", what do consumers see
- Consumer load balancing and fail-over
- Working with Partitions for parallel processing and resiliency
- Brief Overview of Kafka Streams, Kafka Connectors, Kafka REST

Lab Kafka Setup single node, single ZooKeeper

- Create a topic
- Produce and consume messages from the command line

Lab Set up Kafka multi-broker cluster

- Configure and set up three servers
- Create a topic with replication and partitions
- Produce and consume messages from the command line

Session 2: Writing Kafka Producers Basics

- Introduction to Producer Java API and basic configuration

Lab Write Kafka Java Producer

- Create topic from command line
- View topic layout of partitions topology from command line
- View log details
- Use ./kafka-replica-verification.sh to verify replication is correct

Session 3: Writing Kafka Consumers Basics

- Introduction to Consumer Java API and basic configuration

  Lab Write Java Consumer

- View how far behind the consumer is from the command line
- Force failover and verify new leaders are chosen

Session 4: Low-level Kafka Architecture

- Motivation Focus on high-throughput
- Embrace file system / OS caches and how this impacts OS setup and usage
- File structure on disk and how data is written
- Kafka Producer load balancing details
- Producer Record batching by size and time
- Producer async commit and commit (flush, close)
- Pull vs poll and backpressure
- Compressions via message batches (unified compression to server, disk and consumer)
- Consumer poll batching, long poll
- Consumer Trade-offs of requesting larger batches
- Consumer Liveness and fail over redux
- Managing consumer position (auto-commit, async commit and sync commit)
- Messaging At most once, At least once, Exactly once
- Performance trade-offs message delivery semantics
- Performance trade-offs of poll size
- Replication, Quorums, ISRs, committed records
- Failover and leadership election
- Log compaction by key
- Failure scenarios

Session 5: Writing Advanced Kafka Consumers

- Adjusting poll read size
- Implementing at most once message semantics using Java API
- Implementing at least once message semantics using Java API
- Implementing as close as we can get to exactly once Java API
- Re-consume messages that are already consumed
- Using ConsumerRebalanceListener to start consuming from a certain offset (consumer.seek*)
- Assigning a consumer a specific partition (use cases for this)

  Lab 1 Write Java Advanced Consumer

  Lab 2 Adjusting poll read size

  Lab 3 Implementing at most once message semantics using Java API

  Lab 4 Implementing at least once message semantics using Java API

  Lab 5 Implementing as close as we can get to exactly once Java API

Session 7: Schema Management in Kafka

- Avro overview
- Avro Schemas
- Flexible Schemas with JSON and defensive programming
- Using Kafka's Schema Registry
- Topic Schema management\
- Validation of schema
- Prevent producers that don't align with topic schema

Lab1 Topic Schema management

- Validation of schema
- Prevent Consumer from accepting unexpected schema / defensive programming
- Prevent producers from sending messages that don't align with schema registry