Introduction to Kafka for C# Developers

Course Number: WA3285

Duration: 3 days

Course Description

Kafka is an open-source distributed event-streaming platform used for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications. This four-day class will introduce students to Kafka's capabilities using .NET Core, through lecture, labs, and a hands-on project the final day.

Skills Gained

- Understand the use of Kafka for high performance messaging
- Identify the usages for Kafka in Microservices
- Explain the benefits of Kafka patterns
- Differentiate between messaging and message brokers
- Describe Kafka messaging environments
- Develop producers and consumers for Kafka
- Recognize how Kafka enables Cloud-native applications
- Summarize characteristics and architecture for Kafka
- Demonstrate how to process and consume messages from Kafka using .NET Core Web API, MVC, and Worker (BackgroundService)
- Demonstrate Kafka's role in the end-to-end project involving .NET MVC frontend, .NET Web API backend, .NET Worker, Azure SQL database, and Redis cache.
- Design distributed high throughput systems based on Kafka

- Describe the built-in partitioning, replication and inherent fault-tolerance of Kafka

Who Can Benefit

- .NET/C# Developers
- Architects
- System Integrators
- Security Administrators
- Network Administrators
- Software Engineers
- Technical Support Individuals
- Technology Leaders and Managers
- Consultants who are responsible for elements of messaging for data collection, transformation, and integration for your organization.

Prerequisites

- Basic understanding of messaging, cloud, development, architecture and virtualization is beneficial.
- Experience in .NET with C# is highly recommended as the majority of the labs, and the course project uses .NET Core

Course Details

Introduction to Kafka

- Messaging Architectures – What is Messaging?
- Messaging Architectures – Steps to Messaging
- Messaging Architectures – Messaging Models
- What is Kafka?
- Kafka Overview
- Need for Kafka
- When to Use Kafka?
- Kafka Architecture
- Core concepts in Kafka
- Kafka Topic
- Kafka Partitions
- Kafka Producer
- Kafka Consumer
- Kafka Broker

- Kafka Cluster
- Why Kafka Cluster?
- Sample Multi-Broker Cluster
- Overview of ZooKeeper
- Kafka Cluster & ZooKeeper
- Schema Registry
- Who Uses Kafka?

The Inner Workings of Apache Kafka

- A Kafka Cluster High-Level Interaction Diagram
- Topics & Partitions
- The Terms Event/Message/Record
- Message Offset
- Message Retention Settings
- Deleting Messages
- The Flush Policies
- Writing to Partitions
- Batches
- Batch Compression
- Partitions as a Unit of Parallelism
- Message Ordering
- Kafka Default Partitioner
- The Load Balancing Aspect
- Kafka Message Production Schematics
- ZooKeeper
- Reading from a Topic
- Consumer Lag
- Consumer Group
- Consumer Group Diagram
- The Broker
- Broker Hardware Consideration
- OS and File System
- The Leader and Followers Pattern
- Partition Replication Diagram

- Controlled Shutdown
- Controlling Message Durability with Minimum In-Sync Replicas
- Log Compaction
- Frequent Operational Problems
- Some Kafka Design FAQs

Using Apache Kafka

- What is Confluent?
- Confluent Cloud
- Confluent Cloud Resource Hierarchy
- Setting up Confluent Cloud on Azure
- Setting up Confluent Cloud using Confluent.io
- Select the Confluent Cloud Cluster Type
- Choose the Cloud Provider
- Setting up Confluent Cloud using Azure Marketplace
- Select Confluent Cloud in Azure Marketplace
- Purchase Confluent Cloud
- The Cluster View
- Exploring the Confluent Cloud Console
- Topics
- Topics Advanced Settings
- Searching for Messages in a Topic
- The Confluent CLI
- Kafka Cluster Planning
- Kafka Cluster Planning – Producer/Consumer Throughput
- Kafka Cluster Planning – Sizing for Topics and Partitions
- Managing Topics in Confluent Cloud Console
- Editing an Existing Topic
- Delete a Topic
- Kafka and .NET
- .NET Kafka Architectures
- Packages
- Installing the Packages
- Navigating .NET Client Documentation

- Important Classes and Interfaces
- appsettings.json Kafka Configuration
- Loading the Configuration from appsettings.json
- Produce and ProduceAsync Methods
- Produce vs ProduceAsync
- Error Handling
- Consuming Messages
- Creating and Deleting Topics
- Copying Data from Between Environments
- Mocking Datasets using Datagen Connector
- Monitoring Confluent Cloud
- Monitoring Confluent Cloud using cURL
- Motoring Confluent Cloud using third-party Tools

Building Data Pipelines

- Building Data Pipelines
- What to Consider When Building Data Pipelines
- Timeliness
- Reliability
- High and Varying Throughput
- Evolving Schema
- Data Formats
- Protobuf (Protocol Buffers) Overview
- Avro Overview
- Managing Data Evolution Using Schemas
- Confluent Schema Registry
- Confluent Schema Registry in a Nutshell
- Schema Management on Confluent Cloud
- Create a Schema
- Create a Schema using Confluent CLI
- Create a Schema from the Web UI
- Schema Change and Backward Compatibility
- Collaborating over Schema Change
- Handling Unreadable Messages

- Deleting Data
- Segregating Public and Private Topics
- Transformations
- Transformations - ELT
- Security
- Failure Handling
- Agility and Coupling
- Ad-hoc Pipelines
- Metadata Loss
- Extreme Processing
- Kafka Connect vs. Producer and Consumer

Integrating Kafka with Other Systems

- Introduction to Kafka Integration
- Kafka Connect
- Running Kafka Connect Operating Modes
- Key Configurations for Connect workers:
- Kafka Connect API

Kafka Security

- Kafka Security
- Encryption and Authentication using SSL
- Configuring Kafka Brokers
- Configuring Kafka Brokers – Optional Settings
- Authenticating Using SASL
- Authenticating Using SASL – Configuring Kafka Brokers
- Authorization and ACLs
- Securing a Running Cluster
- ZooKeeper Authentication

Monitoring Kafka

- Introduction
- Metrics Basics
- JVM Monitoring
- Garbage collection

- Java OS monitoring
- OS Monitoring
- Kafka Broker Metrics
- Under-Replicated Partitions
- Active controller count
- Request handler idle ratio
- Intelligent Thread Usage
- All topics bytes in
- All topics bytes out
- All topics messages in
- Partition count
- Leader count
- Offline partitions
- Request metrics
- Logging
- Client Monitoring
- Producer Metrics
- Overall producer metrics
- Per-broker and per-topic metrics
- Consumer Metrics
- Fetch Manager Metrics
- Per-broker and per-topic metrics
- Consumer coordinator metrics
- Quotas
- Lag Monitoring
- End-to-End Monitoring

Apache Kafka Best Practices

- Best Practices for Working with Partitions
- Best Practices for Working with Consumers
- Best Practices for Working with Producers
- Best Practices for Working with Brokers

Lab Exercises

- Lab 1. Understanding Confluent Cloud Clusters

- Lab 2. Understanding Confluent Cloud CLI
- Lab 3. Understanding Kafka Topics
- Lab 4. Using the Confluent CLI to Consume Messages
- Lab 5. Creating an ASP.NET Core MVC Kafka Client
- Lab 6. Creating an ASP.NET Core WebAPI Kafka Client
- Lab 7. Creating a .NET 6 Worker Kafka Client
- Lab 8. Integrating Postgres SQL on AWS and Kafka
- Lab 9. Confluent Kafka Connect
- Lab 10. Kafka Schema Registry