

Kafka, SQL Server and Container Deployment Integration

A six-day advanced lab combining messaging, databases and deployment tools

FA-0786 · 6 days

DevOps, Cloud & Infrastructure · Software Development · Data & Analytics

Introduction

Combine existing SQL and programming skills in a prepared Kafka, SQL Server and container environment. Work through deployment, database/client tooling, multi-broker behaviour and Java/Python messaging exercises. The source's SQL Server 2019 features remain version-specific background, not the newest platform; retired tooling is replaced with supported alternatives.

Learning outcomes

- Operate a prepared Linux-based database and Kafka environment with container tooling.
- Use SQL Server queries, stored procedures and supported development clients.
- Explain Kafka topics, partitions, replication, consumer groups and lag.
- Run multi-broker failure/recovery and performance demonstrations.
- Build guided Java and Python messaging examples.
- Discuss Kubernetes, Spark and analytic integrations without assuming a complete production deployment.

Prerequisites

- Operational understanding of SQL

- Some usage experience in MS SQL
- Programming in either Java or python

A suitable workstation, authorised remote lab access and compatible client tools. Deployment exercises use supported Linux hosts; SQL Server does not support every Unix-like OS or architecture.

Training outline

1. Day 1 – Environment and Kafka fundamentals

- Ubuntu/remote workspace and supported SQL client; prepared SQL Server, Kafka, Docker and Kubernetes ecosystem.
- Brokers, clusters, controller quorum, ports, topics, records, partitions and producer/consumer architecture.
- Current KRaft metadata management; historical ZooKeeper/ensemble concepts for older Kafka versions.
- Install compatible Kafka distribution and use console producers/consumers.
- Multiple clients, partitions, offsets and log storage.

2. Day 2 – Docker

Docker basics, installation and setup.

- Installation on Windows, Mac, Linux.
- Images.
- Containers.
- Processes.
- Ports.
- Volumes.
- Basic commands for creating, pulling images; starting, stopping Containers etc.

Docker demonstration.

- Pull a php-apache image.
- Run the php-apache image in a container.
- Port mapping.
- Volume mapping.
- Data persistence.
- Run Nginx via Docker.
- Exercises.

3. Day 3 – SQL Server in containers

- Pull a supported SQL Server Linux image and run the prepared container.
- Connect using VS Code MSSQL or SSMS rather than retired Azure Data Studio.
- Create databases/tables, query data, stored procedures and T-SQL review.
- Version-specific SQL Server2019 feature context.
- Kubernetes, big-data and Spark introduction as separate technology comparisons; no new SQL Server2019 Big Data Clusters deployment.
- Exercises.

4. Day 4 – Database development and analytic tools

- Supported extensions, SQL/Jupyter notebooks and data visualisation examples.
- Python integration and query-history workflows in selected compatible tools.
- Introductory machine-learning and data portability/migration examples.
- Review SQL Server, Spark and container technology relationships; exercises and evaluation.

5. Day 5 – Kafka clusters, replication and consumer groups

Kafka Cluster with multiple brokers

- Configurations and settings
- Launching and managing
- Inspect broker/controller metadata using current Kafka tools; contrast historical ZooKeeper access
- Multiple partition topics in clusters
- Logs
- Producing and Consuming inter-cluster
- Simulating break-downs

Multiple brokers with Replication

- Creation and launch
- Actual replication
- Logs and details exploration
- Production and consumption
- Storage management
- Break-down simulation
- Manage and recover from break-downs

Consumer Groups

- Using the default consumer groups
- Multiple consumer groups
- Idle consumer groups
- Assessment

6. Day 6 — Performance and client programming

Performance and Testing

- Basic performance test
- Tweaking the parameters
- Consumer performance
- Nonzero LAG values for consumers

Programming in Kafka

Java

- Setup and configuration
- Starting Cluster and Producer
- Controlling Producers
- Serializers
- Consumer offset commits and client-code refactoring
- Partition assignments
- Multiple producers

Python

- Producers in python
- Consumer-side behaviour alongside Python producer exercises
- Hands-on development

- Assessment

Automatic offset commits and partition assignments are consumer topics. Inter-cluster messaging uses explicitly configured clients/replication, not automatic delivery merely because two clusters exist.