

Kafka Foundations for Developers

An introductory Kafka lab for learners with programming and system basics

FA-0736 · 3 days

DevOps, Cloud & Infrastructure · Software Development

Introduction

Learn Apache Kafka concepts through a small supplied Unix-like lab, console messaging and a selected Java or Python client. Inspect topics, partitions, replication, consumer groups and common diagnostic evidence.

This is a beginner course in Kafka, not a beginner course in computing or programming. Three days provide foundational operation and selected resilience/client examples; production deployment and performance expertise require further practice.

Learning outcomes

- Explain broker/controller, topic, partition and client roles in a compatible Kafka environment.
- Produce/consume sample records and inspect offsets, groups and logs.
- Review replication and a controlled failure/recovery example.
- Build a selected Java or Python client and distinguish producer behaviour from consumer commits.
- Investigate a small lag/performance issue and document its assumptions and limitations.

Prerequisites

- Understanding of how file systems work
- Programming ability in either Java or python
- Good command of the English language

Basic networking/terminal skills and access to the supplied compatible Kafka lab; remote-host access is required only if that delivery is used.

The selected main language is practised hands-on; the other client language is shown through supplied comparison examples.

Training outline

1. Day 1 — Kafka architecture and basic operation

- Brokers, KRaft metadata/controller roles, topics, record structure, partitions and producer/consumer responsibilities.
- ZooKeeper ensembles are historical Kafka context; use a compatible current KRaft setup rather than assume current upstream Kafka requires ZooKeeper.
- Access the supplied Unix-like lab, install or inspect compatible runtime/images and use an approved remote editor where needed.
- Produce/consume console records, create selected partitions and inspect offsets and record/log storage.
- Compare multiple producers/consumers and explain boundaries between separate clusters.

2. Day 2 — Replication, failures and groups

- Inspect a controlled multiple-broker setup, configuration, logs and replicated partitions.
- Demonstrate a bounded failure/recovery scenario and document its assumptions; distinguish replication from a full backup or continuity strategy.
- Review storage management and client observations during failure.
- Consumer-group membership, partition assignment and idle members; inspect appropriate offsets and lag.
- Introduce inter-cluster replication as a separate design decision, not an automatic producer behaviour; assess the example.

3. Day 3 — Client development and measured behaviour

- Run a small baseline performance/lag experiment and compare a selected configuration change under stated conditions.
- Use Java or Python for the main client exercise; inspect setup, serializers, send/receive errors and resource cleanup.
- Compare a supplied example in the other language rather than promise two complete language courses.
- Consumer offset commit and partition assignment are consumer concerns, not producer controls.

- Refactor and test the small client, inspect its failure/retry assumptions and review results.