# Apache Kafka 실습 워크북

* Kafka 2.12-3.5.1 활용하기*

## 📋 실습 준비사항

### 설치 확인

다음 소프트웨어가 설치되어 있는지 확인하세요:

- ☑ Ubuntu Linux
- ☑ Java (OpenJDK 8 이상)
- ☑ Hadoop
- ☑ MySQL
- ☑ Apache Kafka 2.12-3.5.1

### 환경 변수 설정 확인

# Java 설치 확인

```
java -version
```

# Kafka 설치 경로 확인 (일반적으로 /opt/kafka 또는 ~/kafka)

ls -la /usr/local/kafka/

## 🚀 실습 1: Kafka 서비스 시작하기

### Step 1-1: Zookeeper 시작

Kafka는 Zookeeper에 의존하므로 먼저 Zookeeper를 시작해야 합니다.

# Kafka 디렉토리로 이동

```
cd /usr/local/kafka/
```

# Zookeeper 시작 (백그라운드 실행)

```
./bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
```

(-daemon은 background 실행)

# Zookeeper 실행 확인

```
jps | grep QuorumPeerMain
```

(-jps만 실행후 QuorumPeerMain 존재확인)

# Apache Kafka 实践练习册

*在 Ubuntu Linux 环境下使用 Kafka 2.12-3.5.1*

## 📋 实践准备事项

### 安装确认

请确认以下软件已安装：

- ☑ Ubuntu Linux
- ☑ Java (OpenJDK 8 或更高版本)
- ☑ Hadoop
- ☑ MySQL
- ☑ Apache Kafka 2.12-3.5.1

### 环境变量设置确认

# Java 安装确认

```
java -version
```

# Kafka 安装路径确认 ( 通常为 /opt/kafka 或 ~/kafka )

ls -la /usr/local/kafka/

## 🚀 实践1：启动 Kafka 服务

### Step 1-1：启动 Zookeeper

Kafka 依赖于 Zookeeper，因此需要先启动 Zookeeper。

# 移动到 Kafka 目录

```
cd /usr/local/kafka/
```

# 启动 Zookeeper ( 后台运行 )

```
bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
```

# Zookeeper 运行确认

```
jps | grep QuorumPeerMain
```

### Step 1-2: Kafka 브로커 시작

# Kafka 서버 시작 (백그라운드 실행)

```
bin/kafka-server-start.sh -daemon config/server.properties
```

# Kafka 실행 확인

```
jps | grep Kafka
```

(-jps만 실행후 Kafka 존재확인)

### Step 1-3: 서비스 상태 확인

# 실행 중인 Java 프로세스 확인

```
jps -l
```

# 포트 확인 (Zookeeper: 2181, Kafka: 9092)

```
netstat -tuln | grep -E "2181|9092"
```

-t: TCP 연결만 표시
-u: UDP 연결만 표시
-l: LISTENING 상태의 포트만 표시 (서비스가 연결을 기다리고 있는 포트)
-n: 숫자 형태로 표시 (호스트명이나 서비스명을 숫자 IP주소와 포트번호로 표시)
-E : regular expression

☑ 체크포인트: Zookeeper와 Kafka 프로세스가 모두 실행 중인지 확인하세요.

## 실습 2: 토픽(Topic) 생성 및 관리

### Step 2-1: 토픽 생성

# 'student-messages' 토픽 생성

```
bin/kafka-topics.sh --create \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --partitions 3 \
  --replication-factor 1
```

(엔터키 후에 조금 기다리면 생성됨)

# 'news-events' 토픽 생성

```
bin/kafka-topics.sh --create \
  --topic news-events \
  --bootstrap-server localhost:9092 \
  --partitions 2 \
  --replication-factor 1
```

(엔터키 후에 조금 기다리면 생성됨)

### Step 1-2：启动 Kafka 代理服务器

# 启动 Kafka 服务器 ( 后台运行 )

```
bin/kafka-server-start.sh -daemon config/server.properties
```

# Kafka 运行确认

```
jps | grep Kafka
```

### Step 1-3：服务状态确认

# 确认正在运行的 Java 进程

```
jps -l
```

# 端口确认 ( Zookeeper: 2181, Kafka: 9092 )

```
netstat -tuln | grep -E "2181|9092"
```

☑ 检查点：请确认 Zookeeper 和 Kafka 进程都在运行中。

## 实践2：主题 ( Topic ) 创建和管理

### Step 2-1：创建主题

# 创建 'student-messages' 主题

```
bin/kafka-topics.sh --create \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --partitions 3 \
  --replication-factor 1
```

# 创建 'news-events' 主题

```
bin/kafka-topics.sh --create \
  --topic news-events \
  --bootstrap-server localhost:9092 \
  --partitions 2 \
  --replication-factor 1
```

### Step 2-2: 토픽 목록 조회

# 생성된 토픽 목록 확인

```
./bin/kafka-topics.sh —list  \
 --bootstrap-server localhost:9092
```

### Step 2-3: 토픽 상세 정보 조회

# 특정 토픽의 상세 정보 확인

```
bin/kafka-topics.sh --describe \
  --topic student-messages \
  --bootstrap-server localhost:9092
```

📊 결과 분석: 파티션 수, 리더 정보, 복제본 정보를 확인해보세요.

## 💬 실습 3: 메시지 생산자(Producer) 실습

### Step 3-1: 콘솔 프로듀서 시작

새 터미널을 열고 다음 명령을 실행하세요:

```
cd /usr/local/kafka/
```

# student-messages 토픽에 메시지 전송

```
bin/kafka-console-producer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092
```

(>이 나타나면 거기에서 입력하면 됨)

### Step 3-2: 메시지 전송 실습

프로듀서 콘솔이 실행되면 다음 메시지들을 한 줄씩 입력하세요:

```
안녕하세요, 첫 번째 메시지입니다!
Kafka 실습 중입니다.
오늘 날씨가 좋네요.
실습이 잘 진행되고 있습니다.
```

각 줄을 입력한 후 Enter를 누르세요. 메시지가 Kafka로 전송됩니다.(아직 수신단이 없으므로 볼수는 없음. 그대로 두고 진행)

### Step 2-2：查询主题列表

# 确认已创建的主题列表

```
bin/kafka-topics.sh                    --list
--bootstrap-server localhost:9092
```

### Step 2-3：查询主题详细信息

# 确认特定主题的详细信息

```
bin/kafka-topics.sh --describe \
  --topic student-messages \
  --bootstrap-server localhost:9092
```

📊 结果分析：请确认分区数、Leader 信息、副本信息。

## 💬 实践3：消息生产者（Producer）实践

### Step 3-1：启动控制台生产者

打开新终端并执行以下命令：

```
cd /usr/local/kafka/
```

# 向 student-messages 主题发送消息

```
bin/kafka-console-producer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092
```

### Step 3-2：消息发送实践

生产者控制台运行后，请逐行输入以下消息：

```
您好，这是第一条消息！
正在进行 Kafka 实践。
今天天气真好。
实践进行得很顺利。
```

输入每行后按 Enter 键。消息将发送到 Kafka。

### Step 3-3: 키-값 페어로 메시지 전송

# 키와 값을 함께 전송하는 프로듀서

```
bin/kafka-console-producer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --property "parse.key=true" \
  --property "key.separator=:"
```

다음과 같이 키:값 형태로 메시지를 입력하세요:

```
student1:첫 번째 학생 메시지
student2:두 번째 학생 메시지
teacher:강사 메시지
```

## 📪 실습 4: 메시지 소비자(Consumer) 실습

### Step 4-1: 콘솔 컨슈머 시작

새 터미널을 열고 다음 명령을 실행하세요:

```
cd /usr/local/kafka/
```

# student-messages 토픽의 메시지 수신

```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --from-beginning
```

🔍 관찰사항: 이전에 전송한 모든 메시지가 표시되는지 확인하세요.

### Step 3-3：以键值对形式发送消息

# 同时发送键和值的生产者

```
bin/kafka-console-producer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --property "parse.key=true" \
  --property "key.separator=:"
```

请以 键:值 的形式输入以下消息：

```
student1:第一个学生消息
student2:第二个学生消息
teacher:讲师消息
```

## 📪 实践4：消息消费者 ( Consumer ) 实践

### Step 4-1：启动控制台消费者

打开新终端并执行以下命令：

```
cd /usr/local/kafka/
```

# 接收 student-messages 主题的消息

```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --from-beginning
```

🔍 观察事项：请确认是否显示了之前发送的所有消息。

### Step 4-2: 키-값 페어 메시지 수신

```
# 키와 값을 함께 표시하는 컨슈머
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --from-beginning \
  --property "print.key=true" \
  --property "key.separator=:"
```

### Step 4-3: 실시간 메시지 수신 테스트

1. 컨슈머 터미널을 열어둔 상태에서
2. 프로듀서 터미널에서 새로운 메시지를 전송해보세요
3. 컨슈머에서 실시간으로 메시지가 수신되는지 확인하세요

### Step 4-2：接收键值对消息

```
# 同时显示键和值的消费者
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --from-beginning \
  --property "print.key=true" \
  --property "key.separator=:"
```

### Step 4-3：实时消息接收测试

1. 保持消费者终端打开的状态
2. 在生产者终端发送新消息
3. 确认消费者是否实时接收到消息

## 👥 실습 5: 컨슈머 그룹(Consumer Group) 실습

### Step 5-1: 컨슈머 그룹 생성

첫 번째 터미널:

```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --group student-group-1
```

두 번째 터미널:

```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --group student-group-1
```

### Step 5-2: 메시지 분산 처리 확인

세 번째 터미널에서 메시지를 여러 개 전송:

```
bin/kafka-console-producer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092
```

다음 메시지들을 전송하고 각 컨슈머에서 어떻게 분산되는지 관찰하세요:

메시지 1
메시지 2
메시지 3
메시지 4
메시지 5

## 👥 实践5：消费者组 ( Consumer Group ) 实践

### Step 5-1：创建消费者组

第一个终端：

```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --group student-group-1
```

第二个终端：

```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --group student-group-1
```

### Step 5-2：确认消息分布式处理

在第三个终端发送多条消息：

```
bin/kafka-console-producer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092
```

发送以下消息并观察各消费者如何分布处理：

消息 1
消息 2
消息 3
消息 4
消息 5

### Step 5-3: 컨슈머 그룹 정보 확인

# 컨슈머 그룹 목록 조회
```
bin/kafka-consumer-groups.sh \
  --bootstrap-server localhost:9092 \
  --list
```

# 특정 컨슈머 그룹 상세 정보
```
bin/kafka-consumer-groups.sh \
  --bootstrap-server localhost:9092 \
  --group student-group-1 \
  --describe
```

### Step 5-3：确认消费者组信息

# 查询消费者组列表
```
bin/kafka-consumer-groups.sh \
  --bootstrap-server localhost:9092 \
  --list
```

# 特定消费者组详细信息
```
bin/kafka-consumer-groups.sh \
  --bootstrap-server localhost:9092 \
  --group student-group-1 \
  --describe
```

## 실습 6: 다중 파티션 처리 실습

### Step 6-1: 파티션별 메시지 분산 확인

# 파티션 정보가 포함된 컨슈머
```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --from-beginning \
  --property "print.partition=true" \
  --property "print.key=true" \
  --property "print.timestamp=true"
```

### Step 6-2: 특정 파티션에서만 메시지 수신

# 파티션 0에서만 메시지 수신
```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --partition 0 \
  --from-beginning
```

## 实践6：多分区处理实践

### Step 6-1：确认按分区的消息分布

# 包含分区信息的消费者
```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --from-beginning \
  --property "print.partition=true" \
  --property "print.key=true" \
  --property "print.timestamp=true"
```

### Step 6-2：仅从特定分区接收消息

# 仅从分区 0 接收消息
```
bin/kafka-console-consumer.sh \
  --topic student-messages \
  --bootstrap-server localhost:9092 \
  --partition 0 \
  --from-beginning
```

## 🛠 실습 7: Kafka 성능 테스트

### Step 7-1: 성능 테스트용 토픽 생성

```
bin/kafka-topics.sh --create \
  --topic performance-test \
  --bootstrap-server localhost:9092 \
  --partitions 6 \
  --replication-factor 1
```

### Step 7-2: 프로듀서 성능 테스트

```
# 10만 개 메시지 전송 성능 테스트
bin/kafka-producer-perf-test.sh \
  --topic performance-test \
  --num-records 100000 \
  --record-size 1024 \
  --throughput 10000 \
  --producer-props \
bootstrap.servers=localhost:9092
```

### Step 7-3: 컨슈머 성능 테스트

```
# 컨슈머 성능 테스트
bin/kafka-consumer-perf-test.sh \
  --topic performance-test \
  --bootstrap-server localhost:9092 \
  --messages 100000 \
  --threads 1
```

📊 결과 분석: 처리량(throughput), 지연시간(latency) 등의 지표를 확인하세요.

## 🛠 实践7：Kafka 性能测试

### Step 7-1：创建性能测试用主题

```
bin/kafka-topics.sh --create \
  --topic performance-test \
  --bootstrap-server localhost:9092 \
  --partitions 6 \
  --replication-factor 1
```

### Step 7-2：生产者性能测试

```
# 10万条消息发送性能测试
bin/kafka-producer-perf-test.sh \
  --topic performance-test \
  --num-records 100000 \
  --record-size 1024 \
  --throughput 10000 \
  --producer-props \
bootstrap.servers=localhost:9092
```

### Step 7-3：消费者性能测试

```
# 消费者性能测试
bin/kafka-consumer-perf-test.sh \
  --topic performance-test \
  --bootstrap-server localhost:9092 \
  --messages 100000 \
  --threads 1
```

📊 结果分析：请确认吞吐量（throughput）、延迟时间（latency）等指标。

## 🔧 실습 8: Kafka 설정 변경 실습

### Step 8-1: 토픽 설정 변경

```
# 토픽의 보존 시간 변경 (7일로 설정)
bin/kafka-configs.sh \
  --bootstrap-server localhost:9092 \
  --entity-type topics \
  --entity-name student-messages \
  --alter \
  --add-config retention.ms=604800000

# 변경된 설정 확인
bin/kafka-configs.sh \
  --bootstrap-server localhost:9092 \
  --entity-type topics \
  --entity-name student-messages \
  --describe
```

### Step 8-2: 파티션 수 증가

```
# 파티션 수를 5개로 늘리기
bin/kafka-topics.sh \
  --bootstrap-server localhost:9092 \
  --topic student-messages \
  --alter \
  --partitions 5

# 변경 확인
bin/kafka-topics.sh \
  --bootstrap-server localhost:9092 \
  --topic student-messages \
  --describe
```

## 🔧 实践8：Kafka 配置更改实践

### Step 8-1：更改主题配置

```
# 更改主题保存时间（设置为7天）
bin/kafka-configs.sh \
  --bootstrap-server localhost:9092 \
  --entity-type topics \
  --entity-name student-messages \
  --alter \
  --add-config retention.ms=604800000

# 确认更改后的配置
bin/kafka-configs.sh \
  --bootstrap-server localhost:9092 \
  --entity-type topics \
  --entity-name student-messages \
  --describe
```

### Step 8-2：增加分区数

```
# 将分区数增加到5个
bin/kafka-topics.sh \
  --bootstrap-server localhost:9092 \
  --topic student-messages \
  --alter \
  --partitions 5

# 确认更改
bin/kafka-topics.sh \
  --bootstrap-server localhost:9092 \
  --topic student-messages \
  --describe
```

## 🧹 실습 9: 정리 작업

### Step 9-1: 토픽 삭제

# 테스트용 토픽 삭제

```
bin/kafka-topics.sh \
  --bootstrap-server localhost:9092 \
  --topic performance-test \
  --delete
```

# 삭제 확인

```
bin/kafka-topics.sh                    --list
--bootstrap-server localhost:9092
```

### Step 9-2: Kafka 서비스 종료

# Kafka 서버 종료

```
bin/kafka-server-stop.sh
```

# Zookeeper 종료

```
bin/zookeeper-server-stop.sh
```

# 프로세스 종료 확인

```
jps | grep -E "Kafka|QuorumPeerMain"
```

## 🧹 实践9：清理工作

### Step 9-1：删除主题

# 删除测试用主题

```
bin/kafka-topics.sh \
  --bootstrap-server localhost:9092 \
  --topic performance-test \
  --delete
```

# 确认删除

```
bin/kafka-topics.sh                    --list
--bootstrap-server localhost:9092
```

### Step 9-2：关闭 Kafka 服务

# 关闭 Kafka 服务器

```
bin/kafka-server-stop.sh
```

# 关闭 Zookeeper

```
bin/zookeeper-server-stop.sh
```

# 确认进程结束

```
jps | grep -E "Kafka|QuorumPeerMain"
```

## 💻 심화 학습 과제

### 과제 1: 로그 분석
`/usr/local/kafka/logs/` 디렉토리의 로그 파일들을 확인하고 다음을 분석해보세요:
- 토픽별 로그 파일 구조
- 메시지가 어떻게 저장되는지
- 인덱스 파일의 역할

### 과제 2: 설정 최적화
다음 상황에 맞는 Kafka 설정을 연구해보세요:
- 대용량 메시지 처리
- 높은 처리량이 필요한 환경
- 데이터 무손실이 중요한 환경

### 과제 3: 실제 애플리케이션 연동
Java나 Python을 사용하여 Kafka와 연동하는 간단한 애플리케이션을 만들어보세요.

## 💻 深度学习作业

### 作业1：日志分析
请确认 `/usr/local/kafka/logs/` 目录的日志文件并分析以下内容：
- 按主题的日志文件结构
- 消息如何存储
- 索引文件的作用

### 作业2：配置优化
请研究以下情况所需的 Kafka 配置：
- 大容量消息处理
- 需要高吞吐量的环境
- 数据无损失至关重要的环境

### 作业3：实际应用程序联动
请使用 Java 或 Python 创建一个与 Kafka 联动的简单应用程序。

## 🔍 트러블슈팅

### 자주 발생하는 문제들

문제 1: `Connection refused` 오류

# 해결방법: 서비스 상태 확인

```
jps -l
netstat -tuln | grep 9092
```

문제 2: 메모리 부족 오류

# 해결방법: JVM 힙 크기 조정

```
export KAFKA_HEAP_OPTS="-Xmx512M -Xms512M"
```

문제 3: 토픽이 생성되지 않음

# 해결방법: Zookeeper 연결 확인

```
bin/zookeeper-shell.sh localhost:2181
ls /brokers/topics
```

## 🔍 故障排除

### 常见问题

问题1：`Connection refused` 错误

# 解决方法：确认服务状态

```
jps -l
netstat -tuln | grep 9092
```

问题2：内存不足错误

# 解决方法：调整 JVM 堆大小

```
export KAFKA_HEAP_OPTS="-Xmx512M -Xms512M"
```

问题3：主题未创建

# 解决方法：确认 Zookeeper 连接

```
bin/zookeeper-shell.sh localhost:2181
ls /brokers/topics
```

## ☑ 실습 완료 체크리스트

- [ ] Kafka와 Zookeeper 서비스 시작/종료
- [ ] 토픽 생성, 조회, 삭제
- [ ] 콘솔 프로듀서로 메시지 전송
- [ ] 콘솔 컨슈머로 메시지 수신
- [ ] 컨슈머 그룹 동작 이해
- [ ] 파티션별 메시지 분산 확인
- [ ] 성능 테스트 실행
- [ ] 토픽 설정 변경
- [ ] 로그 파일 구조 이해

## ☑ 实践完成检查清单

- [ ] Kafka 和 Zookeeper 服务启动/关闭
- [ ] 主题创建、查询、删除
- [ ] 控制台生产者发送消息
- [ ] 控制台消费者接收消息
- [ ] 消费者组运行理解
- [ ] 按分区的消息分布确认
- [ ] 性能测试执行
- [ ] 主题配置更改
- [ ] 日志文件结构理解

🎉 수고하셨습니다!
이 실습을 통해 Apache Kafka의 기본 개념과 사용법을 익혔습니다. 더 깊이 있는 학습을 위해서는 Kafka Streams, Kafka Connect 등의 고급 기능도 살펴보시기 바랍니다.

🎉 辛苦了！
通过这次实践，您学习了 Apache Kafka 的基本概念和使用方法。为了更深入的学习，建议您也了解一下 Kafka Streams、Kafka Connect 等高级功能。