**Required Viewing**

Before starting this assignment, you must watch two videos:

1. **Week 7 Fundamentals Lecture Video**
   * Explains Kafka architecture, acknowledgments (acks), and replication.
   * Provides the background needed to understand what you are learning.
   * Link: <https://youtu.be/waVs5PNMYis>
2. **Week 7 Assignment Walkthrough Video**
   * Demonstrates step by step how to complete the tasks, including commands and expected outputs.
   * It is not enough to simply run commands. You must verify that your commands execute correctly. Incorrect or incomplete results will lose points.

Watching both videos is mandatory. The **fundamentals video** explains what you are learning and why it matters, while the **walkthrough video** shows how to complete the assignment.

**Submission Guidelines**

* Submit your work as a **single Word or PDF document** (no raw screenshots or multiple files).
* Include the following in your submission:
  + Screenshots of each required step.
  + A short explanation for each screenshot:
    - The command/action you ran.
    - What the output shows.
    - Whether the result matched your expectation.
* Organize your work in the **same order as the assignment guide** so it is easy to follow.
* This is a **master’s level course** – professionalism and clarity are expected. Well-structured submissions demonstrate your ability to communicate technical work effectively.

**Week 7 Assignment – Objectives and Points**

* **Objective 1 – Conceptual Foundations (Kafka Fundamentals)**: 8 pts
* **Objective 2 – Topic Creation and Verification**: 12 pts
* **Objective 3 – Producing and Consuming Messages**: 12 pts
* **Objective 4 – Kafka Performance Tests (Single Broker)**: 20 pts
* **Objective 5 – Expanding Kafka and Partitioned/Replicated Topics**: 36 pts

**Total: 88 points**

## Week 7 Assignment: Diving into Apache Kafka

In this assignment, you will gain hands-on experience with **Apache Kafka**, a distributed streaming platform designed for building real-time data pipelines and streaming applications. Kafka excels at handling high-throughput, low-latency data streams, making it a critical tool for large-scale, event-driven architectures. Through a series of exercises, you will learn how to create Kafka topics, produce and consume messages, and run performance tests to measure Kafka's throughput and scalability.

By the end of this assignment, you will:

* Understand how to create Kafka topics and manage message flows between producers and consumers.
* Gain practical experience in producing and consuming real-time data streams.
* Learn how to conduct performance tests to evaluate Kafka’s efficiency and scalability.
* Scale a Kafka instance and observe how partitioning and replication affect performance.

# Objective 1 - Conceptual Foundations (8 points)

Before beginning the assignment, watch the instructor-led fundamentals video, which introduces and explains the key concepts for this week:  
<https://youtu.be/waVs5PNMYis>

**Deliverable:** Write a 3–4 paragraph summary of the fundamentals video. Explain the main concepts (Kafka architecture, acknowledgments, replication) in your own words, why they matter, and how they connect to the exercises in this assignment.

# Objective 2 - Topic Creation and Verification (12 points)

#### **Environment Initialization**

## ! STOP!

Do **NOT** start the containers in the **hadoop-hive-spark-hbase**. This will cause Kafka to fail to start. If you get an error starting Kafka follow these steps:

* 1. Naviage to the hadoop-hive-spark-hbase directory and STOP the containers

cd ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase

docker-compose down

* 1. Navigate to the kafka directory and restart the Docker containers:

cd ~/dsc650-infra/bellevue-bigdata/kafka

docker-compose down

docker-compose up -d

* Navigate to the Kafka directory and start the Docker containers:  
    
  cd dsc650-infra/bellevue-bigdata/kafka
* Start Kafka

docker-compose up -d

* Open **two** terminal sessions and in each, access the Kafka container:

docker exec -it kafka\_kafka\_1 bash

* If you can’t access the Kafka container, it could be due to a container name change. In this cause use:

docker exec -it kafka-kafka-1 bash

#### **2. Topic Creation and Verification in Kafka (On One Terminal Only)**

In this section, you will create a Kafka topic named my-topic and verify its creation. Kafka topics act as message channels, where producers send data and consumers retrieve it.

**Exercise 1:** Create a Kafka topic named ‘my-topic’.

/opt/kafka\_2.13-2.8.1/bin/kafka-topics.sh --create --topic my-topic --bootstrap-server localhost:9092

**Deliverable 1:** Screenshot of the command used to create the topic my-topic and its confirmation. Include 1–2 sentences explaining what the command did.

**Exercise 2:** List the topics to verify that ‘my-topic’ has been successfully created.

/opt/kafka\_2.13-2.8.1/bin/kafka-topics.sh --list --bootstrap-server localhost:9092

**Deliverable 2:** Screenshot of the topic list output showing my-topic exists. Add 1–2 sentences explaining what this confirms.

# Objective 3 - Producing and Consuming Messages (12 points)

#### **1. Producing and Consuming Messages in Kafka**

You will now simulate a real-time data pipeline by starting a Kafka **producer** to send messages and a Kafka **consumer** to receive those messages. This demonstrates how Kafka brokers act as intermediaries between producers and consumers.

**Exercise 3:** In the first terminal, start a Kafka consumer:

/opt/kafka\_2.13-2.8.1/bin/kafka-console-consumer.sh --topic my-topic --from-beginning --bootstrap-server localhost:9092

**Deliverable 1:** Screenshot of the producer terminal after typing sample text. Add 1–2 sentences explaining what you entered.

**Exercise 4:** In the second terminal, start a Kafka producer:

/opt/kafka\_2.13-2.8.1/bin/kafka-console-producer.sh --topic my-topic --bootstrap-server localhost:9092

* Type some text into the producer and press ‘Enter’. Note that the text appears on the consumer terminal.

**Deliverable 2:** Screenshot of the consumer terminal showing the received message. Add 1–2 sentences explaining how this confirms message flow through Kafka.

To exit the console and producer shells type CTRL + C.

Close your second terminal.

# Objective 4 - Performance Tests Single Broker (20 points)

#### **1. Kafka Performance Tests**

In this section, you will perform **producer and consumer performance tests** on the my-topic topic to measure Kafka’s throughput. Performance testing helps you understand Kafka’s efficiency in handling large volumes of messages.

**Exercise 5:** Run a performance test on the producer using the Kafka producer performance test script with provided arguments:

/opt/kafka\_2.13-2.8.1/bin/kafka-producer-perf-test.sh --topic my-topic --num-records 50000 --record-size 100 --throughput 1000 --producer-props bootstrap.servers=localhost:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringSerializer

**Deliverable 1:** Screenshot of the producer performance test output. Include 1–2 sentences explaining the metrics reported (throughput, latency, etc.).

**Exercise 6:** Following the producer test, run a consumer performance test on ‘my-topic’:

/opt/kafka\_2.13-2.8.1/bin/kafka-consumer-perf-test.sh --broker-list localhost:9092 --topic my-topic --messages 50000

**Deliverable 2:** Screenshot of the consumer performance test output. Add 1–2 sentences describing what the numbers show about consumer efficiency.

**Deliverable 3:** A short discussion (1–2 paragraphs) interpreting the overall meaning of the results and how they demonstrate Kafka’s throughput on a single broker.

# Objective 5 - Performance Tests Partition/Replicated (36 points)

#### **1. Expanding Kafka and Running Additional Performance Tests**

You will now scale Kafka to 3 instances and test how Kafka performs with **partitioned and replicated topics**. Partitioning and replication improve Kafka's fault tolerance and scalability, allowing more efficient distribution of messages across brokers.

* Exit the Kafka container and scale Kafka instances to 3:

exit

docker-compose scale kafka=3

* Re-enter the kafka\_kafka\_1 container:

docker exec -it kafka\_kafka\_1 bash

* If you can’t access the Kafka container, it could be due to a container name change. In this cause use:

docker exec -it kafka-kafka-1 bash

**Exercise 7:** Create a topic, this time partitioned and replicated across all three Kafka instances:

/opt/kafka\_2.13-2.8.1/bin/kafka-topics.sh --create --topic my-partitioned-topic --replication-factor 3 --partitions 3 --bootstrap-server localhost:9092

**Deliverable 1:** Screenshot showing successful creation of the my-partitioned-topic with replication factor 3 and 3 partitions. Include 1–2 sentences explaining what this setup means.

**Exercise 8:** Conduct the producer and consumer performance tests on the new topic, observing differences:

/opt/kafka\_2.13-2.8.1/bin/kafka-producer-perf-test.sh --topic my-partitioned-topic --num-records 50000 --record-size 100 --throughput 1000 --producer-props bootstrap.servers=localhost:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringSerializer

Followed by:

/opt/kafka\_2.13-2.8.1/bin/kafka-consumer-perf-test.sh --broker-list localhost:9092 --topic my-partitioned-topic --messages 50000

**Deliverable 2:** Screenshots of the producer and consumer performance test results on the new topic. Add 1–2 sentences for each explaining what the results show.

**Deliverable 3:** A short write-up (2–3 paragraphs) comparing performance between the single-broker test and the partitioned/replicated test. Discuss:

* Differences in throughput or latency.
* The effect of replication and partitioning.
* How shared storage and IOPS could limit performance despite scaling.

## Shutting Down

Ensure all Docker containers are turned off with docker-compose down for each directory. If you’re using google cloud, please shut down your virtual machine to preserve cloud costs.