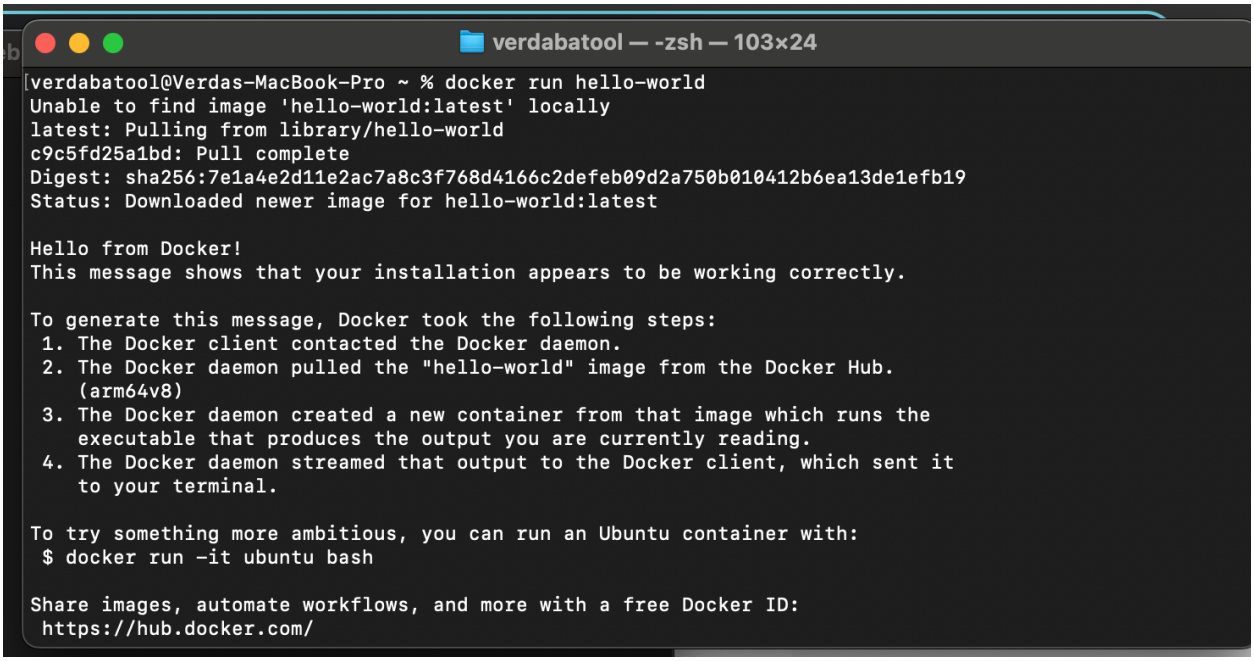
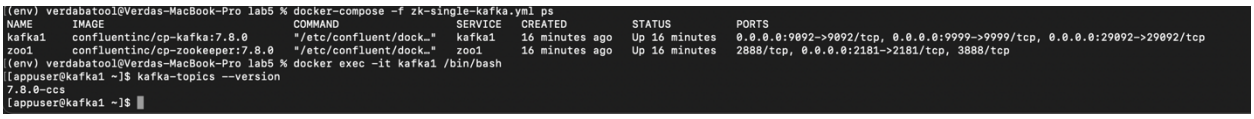
Lab 5: Build a real-time data ingestion and processing pipeline for music events

Student 1: Verda Batool 24280065

Student 2: Ubaidullah Azeem, 23030027

Installing & run Docker (and Docker Compose) to bring up Kafka and ZooKeeper in containers





Launching a Python producer script that sends mock “music event” data (song plays) into Kafka

```
(env) verdabatool@Verdas-MacBook-Pro lab5 % python music_producer.py
Sent event: {'song_id': 202, 'timestamp': 1742505839.607299, 'region': 'APAC', 'action': 'play'}
Sent event: {'song_id': 404, 'timestamp': 1742505840.2693388, 'region': 'US', 'action': 'play'}
Sent event: {'song_id': 505, 'timestamp': 1742505842.068923, 'region': 'APAC', 'action': 'play'}
Sent event: {'song_id': 404, 'timestamp': 1742505843.932082, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 505, 'timestamp': 1742505845.715364, 'region': 'US', 'action': 'play'}
Sent event: {'song_id': 202, 'timestamp': 1742505847.653624, 'region': 'APAC', 'action': 'play'}
Sent event: {'song_id': 404, 'timestamp': 1742505848.880498, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 404, 'timestamp': 1742505850.1927621, 'region': 'US', 'action': 'play'}
Sent event: {'song_id': 202, 'timestamp': 1742505851.7270331, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 505, 'timestamp': 1742505852.752188, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 303, 'timestamp': 1742505854.6890318, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 101, 'timestamp': 1742505856.67889, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 101, 'timestamp': 1742505857.6686, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 202, 'timestamp': 1742505858.6477451, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 303, 'timestamp': 1742505859.297021, 'region': 'APAC', 'action': 'play'}
Sent event: {'song_id': 505, 'timestamp': 1742505860.955671, 'region': 'APAC', 'action': 'play'}
Sent event: {'song_id': 505, 'timestamp': 1742505862.812871, 'region': 'APAC', 'action': 'play'}
Sent event: {'song_id': 202, 'timestamp': 1742505863.772651, 'region': 'APAC', 'action': 'play'}
Sent event: {'song_id': 202, 'timestamp': 1742505864.913764, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 101, 'timestamp': 1742505865.514555, 'region': 'APAC', 'action': 'play'}
Sent event: {'song_id': 505, 'timestamp': 1742505866.266304, 'region': 'EU', 'action': 'play'}
Sent event: {'song_id': 101, 'timestamp': 1742505866.8664792, 'region': 'APAC', 'action': 'play'}
```

Running a PySpark Structured Streaming job that subscribes to the Kafka topic, aggregates data by region and time window, and outputs the top songs in near real-time.

```
=== Batch: 1 ===
+------------------------------------------+------+-------+-----+---+
|window                                    |region|song_id|count|rn |
+------------------------------------------+------+-------+-----+---+
|{2025-03-21 02:25:00, 2025-03-21 02:30:00}|EU    |505    |1    |1  |
|{2025-03-21 02:25:00, 2025-03-21 02:30:00}|US    |101    |1    |1  |
|{2025-03-21 02:25:00, 2025-03-21 02:30:00}|US    |404    |1    |2  |
|{2025-03-21 02:25:00, 2025-03-21 02:30:00}|US    |505    |1    |3  |
+------------------------------------------+------+-------+-----+---+

=== Batch: 2 ===
+------------------------------------------+------+-------+-----+---+
|window                                    |region|song_id|count|rn |
+------------------------------------------+------+-------+-----+---+
|{2025-03-21 02:25:00, 2025-03-21 02:30:00}|EU    |404    |1    |1  |
|{2025-03-21 02:25:00, 2025-03-21 02:30:00}|APAC  |303    |1    |1  |
|{2025-03-21 02:25:00, 2025-03-21 02:30:00}|APAC  |101    |1    |2  |
+------------------------------------------+------+-------+-----+---+
```

Varying the Time Window to 1 minute

When the time window was changed to 1 minute, we observed the following changes:

- More frequent "trending" updates.
- Small bursts of data cause quick changes (spikes).
- More CPU usage due to more frequent grouping/calculations.

Varying the Time Window to 10 minutes

When the time window was changed to 10 minutes, we observed the following changes:

- Less frequent updates.
- Smoother trends (more stable but slower to change).
- Lower CPU usage, but trends are less reactive.

Adding Skip/Like Actions

```
(env) verdabatool@Verdas-MacBook-Pro lab5 % python music_producer.py
Sent event: {'song_id': 303, 'timestamp': 1742588345.6883352, 'region': 'APAC', 'action': 'skip'}
Sent event: {'song_id': 404, 'timestamp': 1742588346.57492, 'region': 'US', 'action': 'like'}
Sent event: {'song_id': 404, 'timestamp': 1742588347.949418, 'region': 'EU', 'action': 'like'}
Sent event: {'song_id': 202, 'timestamp': 1742588349.584666, 'region': 'EU', 'action': 'skip'}
Sent event: {'song_id': 303, 'timestamp': 1742588350.0880792, 'region': 'APAC', 'action': 'skip'}
Sent event: {'song_id': 202, 'timestamp': 1742588351.102049, 'region': 'US', 'action': 'like'}
Sent event: {'song_id': 404, 'timestamp': 1742588352.542078, 'region': 'EU', 'action': 'like'}
Sent event: {'song_id': 303, 'timestamp': 1742588353.192246, 'region': 'APAC', 'action': 'skip'}
Sent event: {'song_id': 202, 'timestamp': 1742588353.78145, 'region': 'APAC', 'action': 'skip'}
Sent event: {'song_id': 404, 'timestamp': 1742588354.958912, 'region': 'EU', 'action': 'skip'}
Sent event: {'song_id': 202, 'timestamp': 1742588356.8087869, 'region': 'APAC', 'action': 'skip'}
Sent event: {'song_id': 303, 'timestamp': 1742588357.954078, 'region': 'APAC', 'action': 'like'}
Sent event: {'song_id': 303, 'timestamp': 1742588359.615438, 'region': 'EU', 'action': 'like'}
Sent event: {'song_id': 505, 'timestamp': 1742588360.934443, 'region': 'US', 'action': 'skip'}
Sent event: {'song_id': 303, 'timestamp': 1742588362.417567, 'region': 'APAC', 'action': 'like'}
```

Setting a Different Micro-Batch Interval to 5 seconds

When the micro batch interval was set to 5 seconds:

- Micro-batches were triggering every 5 seconds and the results were appearing in the console after every 5 seconds.
- There was lower CPU load since batches are less frequent.
- There was slightly more "latency" (delay) in trend updates and hence the system was more stable/balanced.

Setting a Different Micro-Batch Interval to 1 second

- Micro-batches were triggering every 1 second, and the results were appearing in the console after every second.
- There was higher CPU load since batches are more frequent.
- Updates were much faster/real-time but at the cost of overhead. This might cause Spark's driver/executor CPU spike and potentially strain system