Part A: **Synchronous (REST)**

- Location: sync-rest/
- Services:
    - OrderService: app.py implements POST /order
    - InventoryService: app.py implements POST /reserve
    - NotificationService: app.py implements POST /send
- Testing: Latency and failure tests in test_sync.py
- Results: See latency tables and reasoning in sync-rest/tests/results/

Submission:

Simple latency table, plus reasoning on why the behavior happens

The above testing demonstrates that in a synchronous REST architecture, end-to-end latency is directly dependent on downstream services. Any delay introduced in the Inventory service increases OrderService latency almost linearly, and failures propagate upstream unless controlled by timeouts. Tail latency (p95/p99) worsens under delay due to blocking and request queueing. This tight temporal coupling reduces resilience and limits scalability, highlighting why synchronous designs are vulnerable to cascading performance degradation in distributed systems.

Here is a consolidated latency table based on the results from the files:

Test Scenario	Success	Avg Latency (s)	Status Codes
Baseline (no delay)	10/10	0.009	200
Delay (2s sleep)	0/10	1.018	503
Inventory Failure	0/10	0.010	503

- Baseline: All requests succeeded quickly.
- Delay: All requests timed out (due to 1s timeout vs 2s sleep), returned 503.
- Failure: Inventory service stopped, requests failed fast with 503.

Part B: Async (RabbitMQ)

- Location: async-rabbitmq/
- Services:
    - OrderService: app.py publishes OrderPlaced
    - InventoryService: consumer.py consumes, reserves, publishes results
    - NotificationService: consumer.py consumes and sends notifications
- Testing: Normal flow, Backlog, idempotency, and DLQ tests in tests
- Idempotency: Comments below explain the logic in consumer.py (processed_orders set) for handling idempotency
- DLQ: Malformed message handling in send_malformed_message.py explained in the comments below

Submission:

- Screenshots or logs showing backlog, then recovery
- Short explanation of the idempotency strategy

RabbitMQ provides **at-least-once delivery**, meaning a message may be redelivered if a consumer crashes before acknowledgment.

To ensure idempotency, InventoryService uses the `order_id` as a unique idempotency key and stores processed order IDs in a database. Before reserving inventory, it checks whether the `order_id` has already been processed. If it exists, the message is acknowledged without performing the reservation again.

This prevents duplicate reservations even if the same message is delivered multiple times.

1. Backlog, then recovery

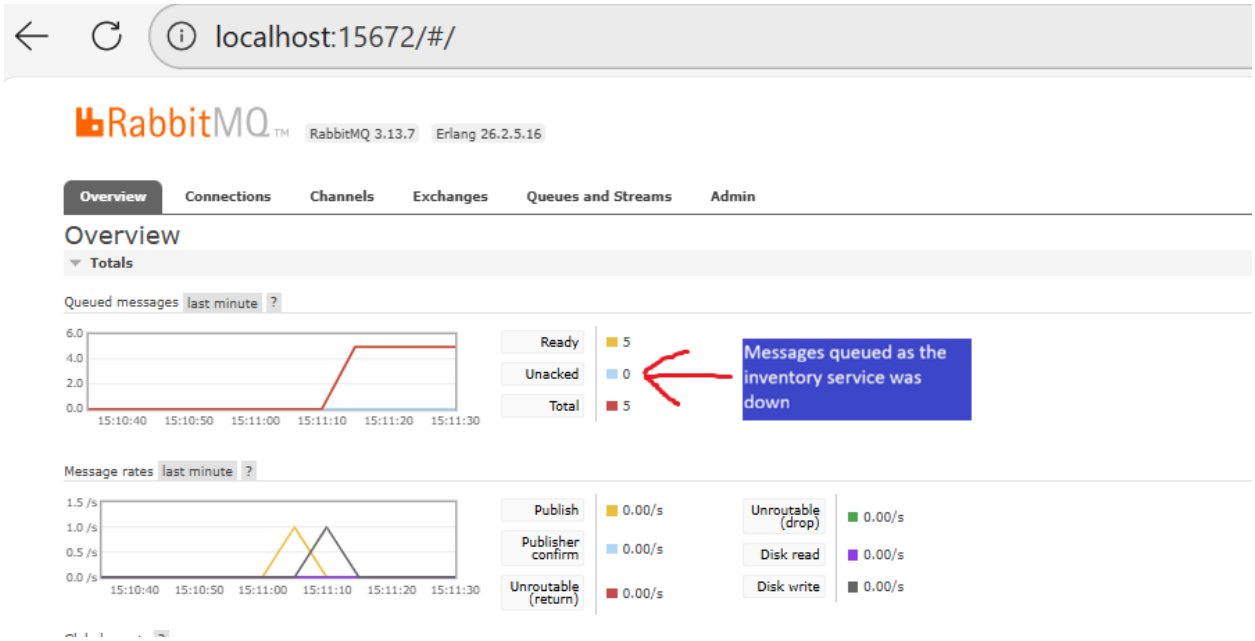
```
Problems  Output  Debug Console  Terminal  Ports                                                                    docker

PS C:\Users\018464615\Downloads\Sem2\CMPE-273\Lab\lab2\cmpe273-lab2\async-rabbitmq> docker-compose up -d
[+] up 1/1
 ✓ Container rabbitmq-broker Healthy                                                                                 11.5s
PS C:\Users\018464615\Downloads\Sem2\CMPE-273\Lab\lab2\cmpe273-lab2\async-rabbitmq> docker-compose ps
NAME                  IMAGE                                  COMMAND                  SERVICE               CREATED              STATUS                        PORTS
inventory-service     async-rabbitmq-inventory_service       "python consumer.py"     inventory_service     2 minutes ago        Up 10 seconds                 0.0.0.0:5002->5002/tcp, [::]:5002->5002/tcp
notification-service  async-rabbitmq-notification_service    "python consumer.py"     notification_service  About a minute ago   Up 10 seconds                 0.0.0.0:5003->5003/tcp, [::]:5003->5003/tcp
order-service         async-rabbitmq-order_service           "python app.py"          order_service         About a minute ago   Up 10 seconds                 0.0.0.0:5001->5001/tcp, [::]:5001->5001/tcp
rab                   Focus folder in explorer (ctrl + click)  }-management          "docker-entrypoint.s…"  rabbitmq           2 minutes ago        Up 22 seconds (healthy)       0.0.0.0:5672->5672/tcp, [::]:5672->5672/tcp, 0.0.
0.0  ->15672/tcp
PS C:\Users\018464615\Downloads\Sem2\CMPE-273\Lab\lab2\cmpe273-lab2\async-rabbitmq> docker stop inventory-service
inventory-service
PS C:\Users\018464615\Downloads\Sem2\CMPE-273\Lab\lab2\cmpe273-lab2\async-rabbitmq> 0..4 | ForEach-Object {
>>   Invoke-RestMethod -Uri "http://localhost:5001/order" -Method POST -ContentType "application/json" -Body "{`"user_id`": `"user_$_`", `"items`": [{`"item_id`": `"item_1`", `"quantity`": 1, `"price`": 10
.0}]}"
>>   Start-Sleep -Milliseconds 500
>> }

message                                order_id             status
-------                                --------             ------
Order created and queued for processing ORD-20260216-50b149e8 pending
Order created and queued for processing ORD-20260216-7104254e pending
Order created and queued for processing ORD-20260216-042373f0 pending
Order created and queued for processing ORD-20260216-3f1bfc66 pending
Order created and queued for processing ORD-20260216-36463e7d pending


PS C:\Users\018464615\Downloads\Sem2\CMPE-273\Lab\lab2\cmpe273-lab2\async-rabbitmq> docker start inventory-service
inventory-service
PS C:\Users\018464615\Downloads\Sem2\CMPE-273\Lab\lab2\cmpe273-lab2\async-rabbitmq> docker logs -f inventory-service
[InventoryService] Connecting to RabbitMQ...[InventoryService] Starting Inventory Service on port 5002...

 * Serving Flask app 'consumer'
 * Debug mode: on
[InventoryService] Connection lost. Reconnecting in 5 seconds...
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://172.18.0.3:5002
Press CTRL+C to quit
[InventoryService] Connected. Declaring exchanges and queue...
[InventoryService] Queue order_placed declared (with DLX).
[InventoryService] Waiting for OrderPlaced events...
[InventoryService] Connecting to RabbitMQ...[InventoryService] Starting Inventory Service on port 5002...

 * Serving Flask app 'consumer'
 * Debug mode: on
[InventoryService] Connection lost. Reconnecting in 5 seconds...
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://172.18.0.4:5002
Press CTRL+C to quit
[InventoryService] Connected. Declaring exchanges and queue...
[InventoryService] Queue order_placed declared (with DLX).
[InventoryService] Waiting for OrderPlaced events...
[InventoryService] Connecting to RabbitMQ...[InventoryService] Starting Inventory Service on port 5002...

 * Serving Flask app 'consumer'
 * Debug mode: on
[InventoryService] Connected. Declaring exchanges and queue...
[InventoryService] Queue order_placed declared (with DLX).

Ctrl+K to generate command
```

Docker is up

Stopped inventory Service

Placed multiple orders while inventory service was down

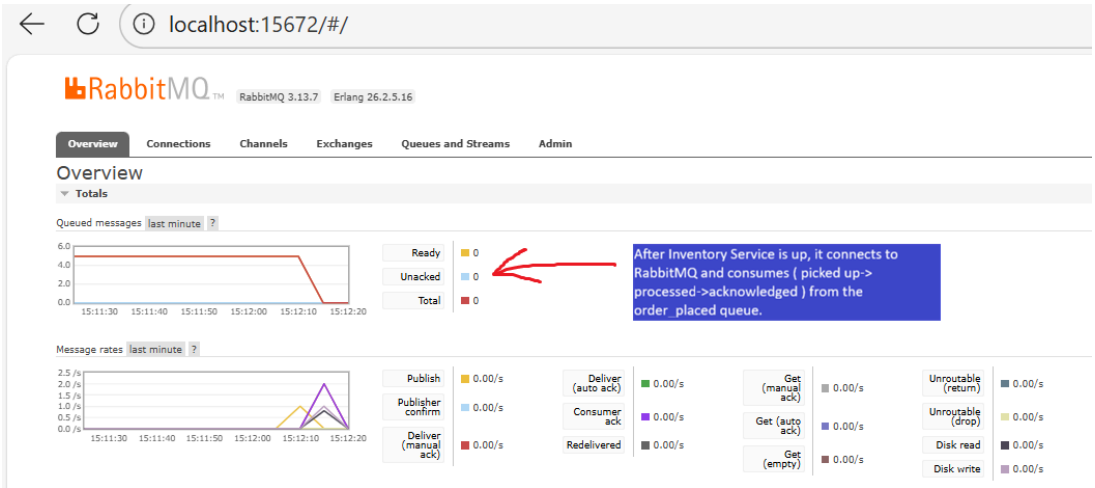
Started inventory service



Recovery :

```
PS C:\Users\018464615\Downloads\Sem2\CMPE-273\Lab\lab2\cmpe273-lab2\async-rabbitmq> docker start inventory-service
inventory-service
PS C:\Users\018464615\Downloads\Sem2\CMPE-273\Lab\lab2\cmpe273-lab2\async-rabbitmq> docker logs -f inventory-service
[InventoryService] Connecting to RabbitMQ...[InventoryService] Starting Inventory Service on port 5002...

 * Serving Flask app 'consumer'
 * Debug mode: on
[InventoryService] Connection lost. Reconnecting in 5 seconds...
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://172.18.0.3:5002
Press CTRL+C to quit
[InventoryService] Connected. Declaring exchanges and queue...
[InventoryService] Queue order_placed declared (with DLX).
[InventoryService] Waiting for OrderPlaced events...
[InventoryService] Connecting to RabbitMQ...[InventoryService] Starting Inventory Service on port 5002...

 * Serving Flask app 'consumer'
 * Debug mode: on
[InventoryService] Connection lost. Reconnecting in 5 seconds...
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://172.18.0.4:5002
Press CTRL+C to quit
[InventoryService] Connected. Declaring exchanges and queue...
[InventoryService] Queue order_placed declared (with DLX).
[InventoryService] Waiting for OrderPlaced events...
[InventoryService] Connecting to RabbitMQ...[InventoryService] Starting Inventory Service on port 5002...

 * Serving Flask app 'consumer'
 * Debug mode: on
[InventoryService] Connected. Declaring exchanges and queue...
[InventoryService] Queue order_placed declared (with DLX).
[InventoryService] Waiting for OrderPlaced events...
[InventoryService] Received OrderPlaced event: ORD-20260216-50b149e8
[InventoryService] Published InventoryReserved event: ORD-20260216-50b149e8
[InventoryService] Successfully reserved inventory for order ORD-20260216-50b149e8
[InventoryService] Received OrderPlaced event: ORD-20260216-7104254e
[InventoryService] Published InventoryReserved event: ORD-20260216-7104254e
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://172.18.0.4:5002
Press CTRL+C to quit
[InventoryService] Successfully reserved inventory for order ORD-20260216-7104254e
[InventoryService] Received OrderPlaced event: ORD-20260216-042373f0
[InventoryService] Published InventoryReserved event: ORD-20260216-042373f0
[InventoryService] Successfully reserved inventory for order ORD-20260216-042373f0
[InventoryService] Received OrderPlaced event: ORD-20260216-3f1bfc66
[InventoryService] Published InventoryReserved event: ORD-20260216-3f1bfc66
[InventoryService] Successfully reserved inventory for order ORD-20260216-3f1bfc66
[InventoryService] Received OrderPlaced event: ORD-20260216-36463e7d
[InventoryService] Published InventoryReserved event: ORD-20260216-36463e7d
[InventoryService] Successfully reserved inventory for order ORD-20260216-36463e7d
```

Orders consumed from queue and Inventory reserved once inventory service was up

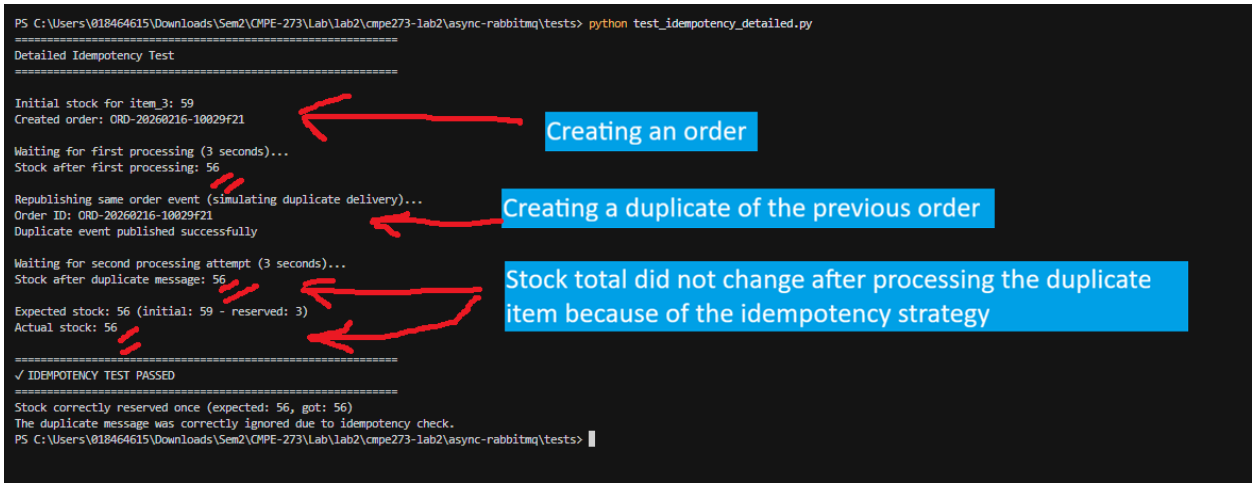


Idempotency strategy

InventoryService keeps an in-memory set of processed order IDs (`processed_orders`). When it receives an OrderPlaced event:

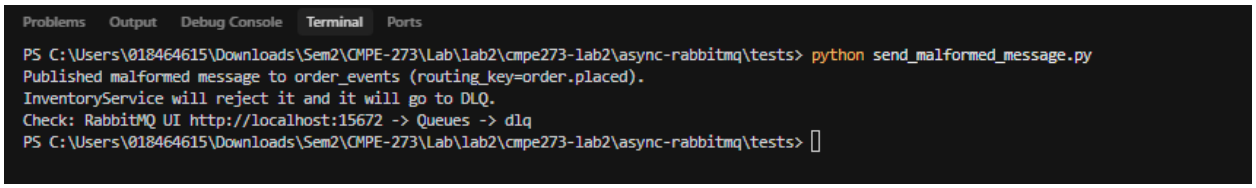
1. It checks whether the event's `order_id` is already in `processed_orders`.
2. If yes, it treats the message as a duplicate (e.g. redelivery or duplicate publish). It acknowledges the message and does not reserve inventory again.
3. If not, it reserves inventory, adds the `order_id` to `processed_orders`, publishes InventoryReserved or InventoryFailed, and then acknowledges the message.

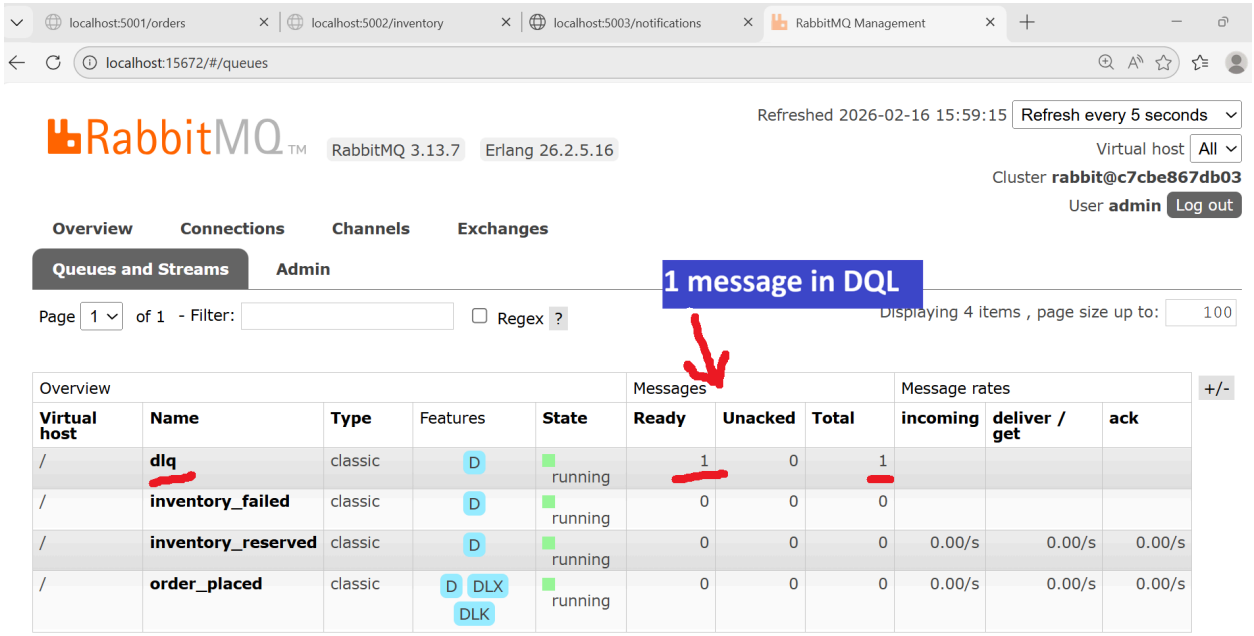
So even if the same OrderPlaced message is delivered twice (e.g. due to RabbitMQ redelivery or a duplicate publish), inventory is reserved only once. In production this set would be stored in a persistent store (e.g. database) so it survives restarts.



DLQ and malformed message handling

- InventoryService rejects any message it can't process (invalid JSON or missing required fields) by telling RabbitMQ not to put it back on the main queue.
- The order_placed queue is set up with a dead-letter exchange so that every rejected message is moved into a separate queue called dlq.
- We can then inspect those bad messages in the RabbitMQ UI (or handle them later) instead of losing them or blocking the main queue.



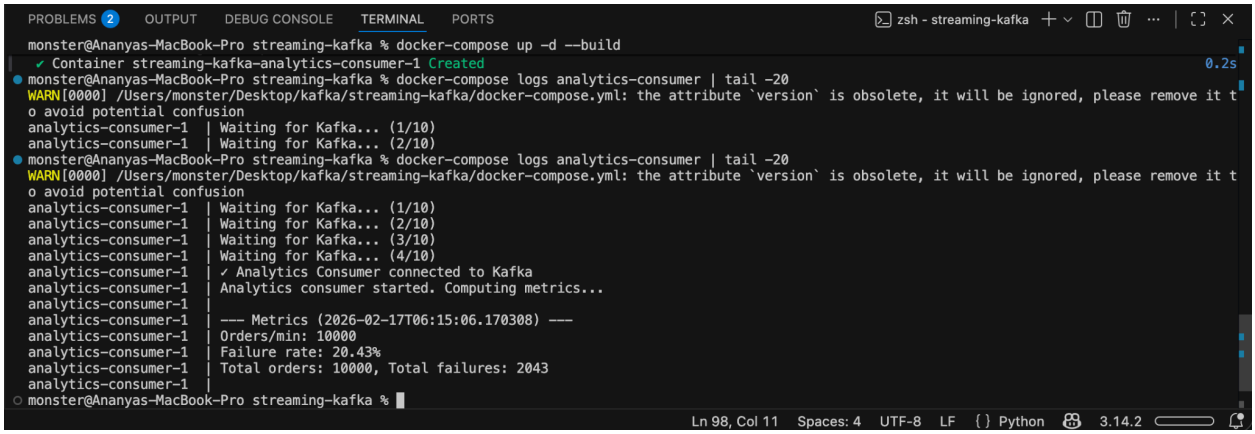


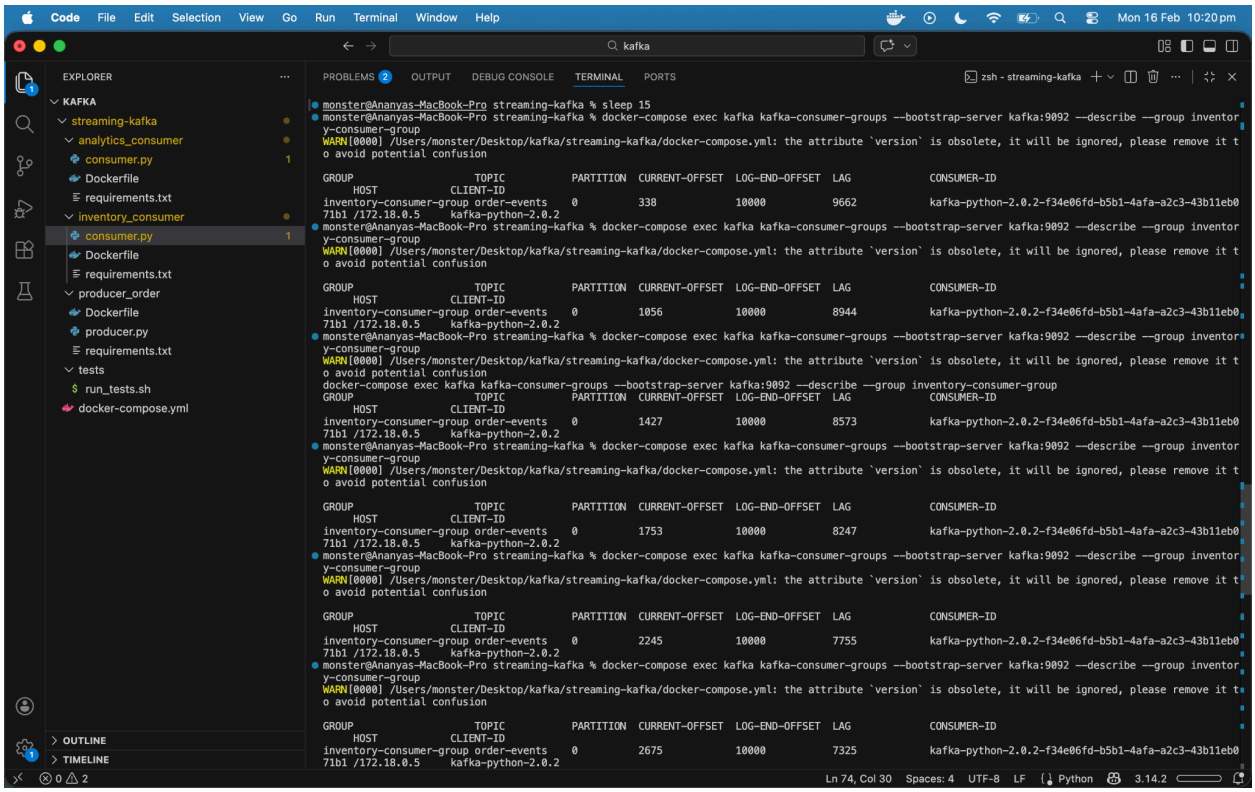
Part C: Streaming (Kafka)

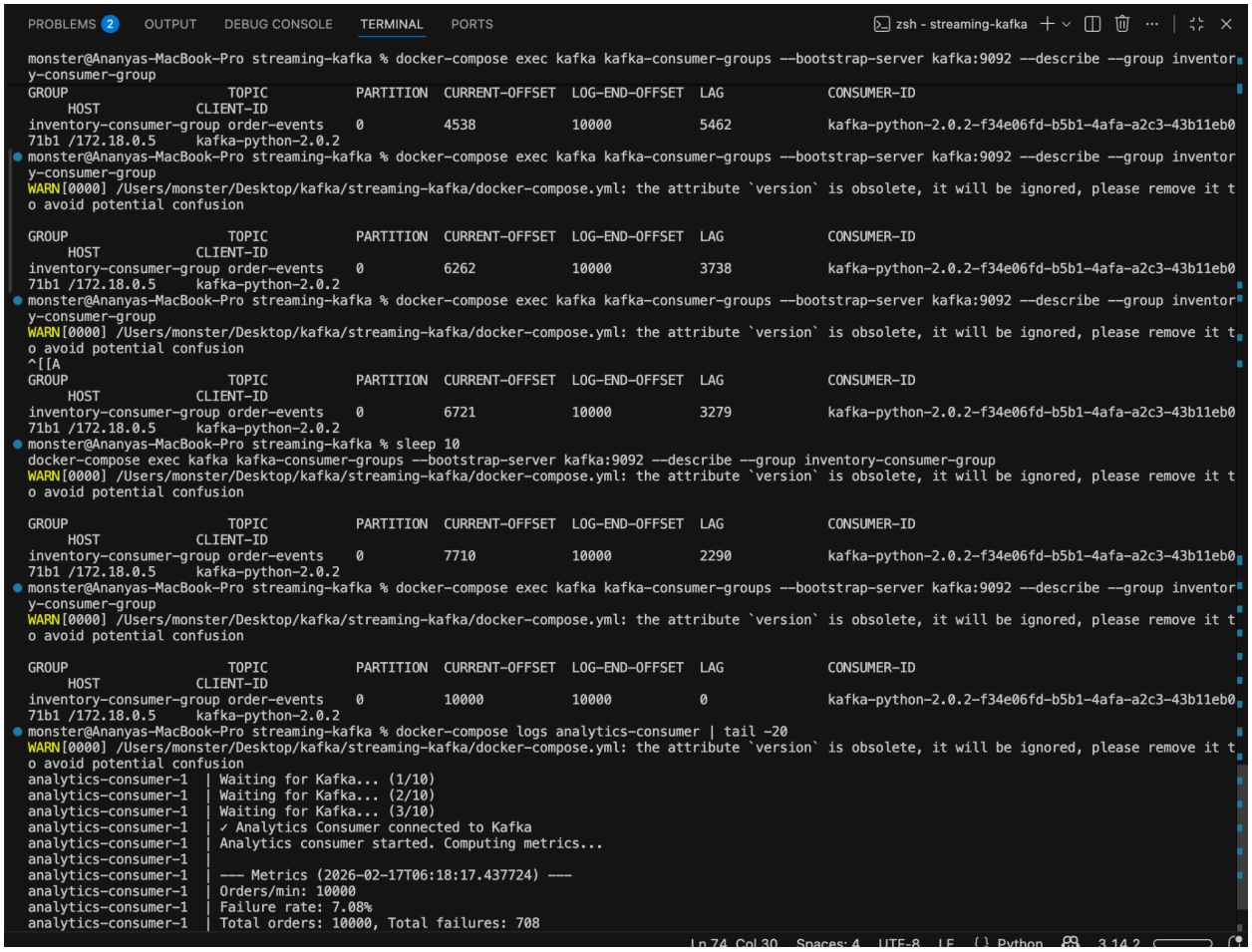
- Location: streaming-kafka/
- Services:
  - Producer: producer.py publishes OrderPlaced
  - Inventory Consumer: consumer.py processes orders
  - Analytics Consumer: consumer.py computes metrics
- Testing: Produce 10k events, observe lag, replay in README.md
- Metrics Output: Written to /app/metrics.json by analytics consumer

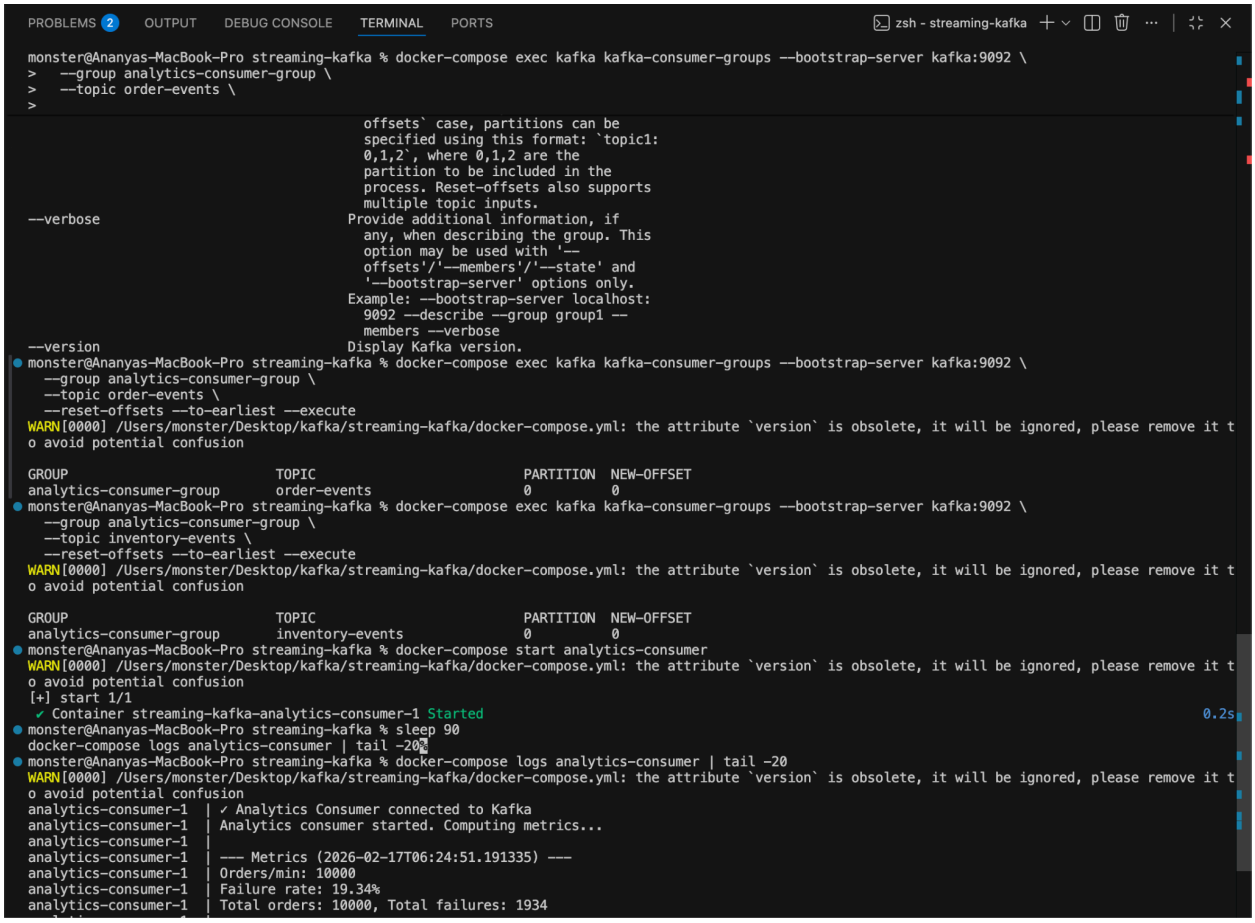
Submission:

- A small metrics output file or printed report
- Evidence of replay (before and after): here


PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - streaming-kafka
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose up -d --build
✓ Container streaming-kafka-analytics-consumer-1 Created 0.2s
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose logs analytics-consumer | tail -20
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
analytics-consumer-1 | Waiting for Kafka... (1/10)
analytics-consumer-1 | Waiting for Kafka... (2/10)
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose logs analytics-consumer | tail -20
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
analytics-consumer-1 | Waiting for Kafka... (1/10)
analytics-consumer-1 | Waiting for Kafka... (2/10)
analytics-consumer-1 | Waiting for Kafka... (3/10)
analytics-consumer-1 | Waiting for Kafka... (4/10)
analytics-consumer-1 | ✓ Analytics Consumer connected to Kafka
analytics-consumer-1 | Analytics consumer started. Computing metrics...
analytics-consumer-1 |
analytics-consumer-1 | --- Metrics (2026-02-17T06:15:06.170308) ---
analytics-consumer-1 | Orders/min: 10000
analytics-consumer-1 | Failure rate: 20.43%
analytics-consumer-1 | Total orders: 10000, Total failures: 2043
analytics-consumer-1 |
monster@Ananyas-MacBook-Pro streaming-kafka %
Ln 98, Col 11 Spaces: 4 UTF-8 LF { } Python 3.14.2


Code File Edit Selection View Go Run Terminal Window Help
Mon 16 Feb 10:20 pm
kafka
EXPLORER
KAFKA
streaming-kafka
analytics_consumer
consumer.py
Dockerfile
requirements.txt
inventory_consumer
consumer.py
Dockerfile
requirements.txt
producer_order
Dockerfile
producer.py
requirements.txt
tests
run_tests.sh
docker-compose.yml
OUTLINE
TIMELINE
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - streaming-kafka
monster@Ananyas-MacBook-Pro streaming-kafka % sleep 15
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 338 10000 9662 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 1056 10000 8944 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 1427 10000 8573 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 1753 10000 8247 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 2245 10000 7755 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 2675 10000 7325 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
0 2
Ln 74, Col 30 Spaces: 4 UTF-8 LF Python 3.14.2


PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - streaming-kafka
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 4538 10000 5462 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 6262 10000 3738 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
^[[A
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 6721 10000 3279 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % sleep 10
docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 7710 10000 2290 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 --describe --group inventory-consumer-group
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID
inventory-consumer-group order-events 0 10000 10000 0 kafka-python-2.0.2-f34e06fd-b5b1-4afa-a2c3-43b11eb071b1 /172.18.0.5 kafka-python-2.0.2
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose logs analytics-consumer | tail -20
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
analytics-consumer-1 | Waiting for Kafka... (1/10)
analytics-consumer-1 | Waiting for Kafka... (2/10)
analytics-consumer-1 | Waiting for Kafka... (3/10)
analytics-consumer-1 | ✓ Analytics Consumer connected to Kafka
analytics-consumer-1 | Analytics consumer started. Computing metrics...
analytics-consumer-1 |
analytics-consumer-1 | --- Metrics (2026-02-17T06:18:17.437724) ---
analytics-consumer-1 | Orders/min: 10000
analytics-consumer-1 | Failure rate: 7.08%
analytics-consumer-1 | Total orders: 10000, Total failures: 708
Ln 74, Col 30
Spaces: 4
UTF-8
LF
Python
3.14.2


PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - streaming-kafka
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 \
> --group analytics-consumer-group \
> --topic order-events \
>
offsets` case, partitions can be
specified using this format: `topic1:
0,1,2`, where 0,1,2 are the
partition to be included in the
process. Reset-offsets also supports
multiple topic inputs.
--verbose
Provide additional information, if
any, when describing the group. This
option may be used with '--
offsets'/'--members'/'--state' and
'--bootstrap-server' options only.
Example: --bootstrap-server localhost:
9092 --describe --group group1 --
members --verbose
--version
Display Kafka version.
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 \
--group analytics-consumer-group \
--topic order-events \
--reset-offsets --to-earliest --execute
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it t
o avoid potential confusion
GROUP
TOPIC
PARTITION
NEW-OFFSET
analytics-consumer-group
order-events
0
0
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose exec kafka kafka-consumer-groups --bootstrap-server kafka:9092 \
--group analytics-consumer-group \
--topic inventory-events \
--reset-offsets --to-earliest --execute
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it t
o avoid potential confusion
GROUP
TOPIC
PARTITION
NEW-OFFSET
analytics-consumer-group
inventory-events
0
0
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose start analytics-consumer
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it t
o avoid potential confusion
[+] start 1/1
✔ Container streaming-kafka-analytics-consumer-1 Started
0.2s
monster@Ananyas-MacBook-Pro streaming-kafka % sleep 90
docker-compose logs analytics-consumer | tail -20
monster@Ananyas-MacBook-Pro streaming-kafka % docker-compose logs analytics-consumer | tail -20
WARN[0000] /Users/monster/Desktop/kafka/streaming-kafka/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it t
o avoid potential confusion
analytics-consumer-1 | ✓ Analytics Consumer connected to Kafka
analytics-consumer-1 | Analytics consumer started. Computing metrics...
analytics-consumer-1 |
analytics-consumer-1 | --- Metrics (2026-02-17T06:24:51.191335) ---
analytics-consumer-1 | Orders/min: 10000
analytics-consumer-1 | Failure rate: 19.34%
analytics-consumer-1 | Total orders: 10000, Total failures: 1934