

Laporan UTS

Sistem Paralel dan Terdistribusi B Pub Sub Log Aggregator



Disusun Oleh :

Dion Prayoga

11221058

15 Oktober 2025

Daftar Isi

1. Bagian Teori.....	2
T1 (Bab 1): Karakteristik Sistem Terdistribusi dan Trade-off.....	2
T2 (Bab 2): Perbandingan Arsitektur Client-Server vs. Publish-Subscribe.....	2
T3 (Bab 3): At-least-once vs. Exactly-once Delivery.....	2
T4 (Bab 4): Skema Penamaan untuk Topic dan Event ID.....	2
T5 (Bab 5): Ordering dan Batasannya.....	2
T6 (Bab 6): Failure Modes dan Strategi Mitigasi.....	3
T7 (Bab 7): Eventual Consistency dan Peran Idempotency + Dedup.....	3
T8 (Bab 1–7): Rumuskan Metrik Evaluasi Sistem.....	3
2. Bagian Implementasi dan Desain.....	4
2.1 Ringkasan Sistem dan Keputusan Desain.....	4
Docker Compose.....	6
2.2 Analisis Performa dan Metrik.....	8
Lampiran.....	12

1. Bagian Teori

T1 (Bab 1): Karakteristik Sistem Terdistribusi dan Trade-off

Sistem terdistribusi memiliki beberapa karakteristik utama, termasuk Resource Sharing, Distribution Transparency (menyembunyikan distribusi fisik), Openness, dan Dependability (mencakup Availability dan Reliability) (Tanenbaum & Van Steen, 2023, bab 1). Dalam desain Pub-Sub log aggregator, trade-off utama terjadi antara Consistency dan Availability. Log aggregator mengutamakan ingest data berkelanjutan dan throughput tinggi, sehingga sering mengorbankan konsistensi yang ketat (Latency) demi Availability, yang menuntun pada model Eventual Consistency. Trade-off lain adalah ordering, di mana sulit menentukan urutan event secara global karena lack of a global clock.

T2 (Bab 2): Perbandingan Arsitektur Client-Server vs. Publish-Subscribe

Arsitektur Client-Server memiliki hubungan tightly coupled di mana klien secara aktif meminta layanan dari server. Sebaliknya, Publish-Subscribe (Pub-Sub) adalah arsitektur decoupled yang menggunakan broker perantara. Memilih Pub-Sub sangat tepat untuk log aggregator karena: 1) Scalability: Memungkinkan penambahan publisher baru dengan mudah tanpa memengaruhi consumer. 2) Throughput: Komunikasi asynchronous memungkinkan publisher mengirim event dengan kecepatan tinggi (non-blocking). 3) Reliability: Event disimpan di buffer internal, sehingga consumer dapat crash dan pulih tanpa kehilangan data. Pub-Sub sesuai karena sifat sistem adalah event-driven dan bukan request-reply yang ketat. (Tanenbaum & Van Steen, 2023, bab 2)

T3 (Bab 3): At-least-once vs. Exactly-once Delivery

At-least-once delivery menjamin event dikirim minimal satu kali, yang sering menghasilkan duplikasi akibat retries. Mencapai Exactly-once delivery sulit dan mahal. Idempotent consumer sangat krusial di presence of retries. Idempotency memastikan bahwa event_id unik digunakan untuk mendeteksi duplikat, dan consumer mengeksekusi logika pemrosesan yang mengubah state tepat satu kali. Ini penting untuk mencegah duplikasi data dan menjaga integritas state sistem. (Tanenbaum & Van Steen, 2023, bab 3)

T4 (Bab 4): Skema Penamaan untuk Topic dan Event ID

Skema penamaan harus menjamin traceability dan resolusi unik: Topic harus hierarkis. Event ID (event_id) harus collision-resistant, idealnya menggunakan UUID. Dampak pada Deduplication (Dedup) adalah mutlak: event_id berfungsi sebagai Primary Key pada Dedup store (SQLite). ID yang unik menjamin consumer dapat secara akurat memverifikasi state event dan menegakkan idempotency, meminimalkan risiko false rejection. (Tanenbaum & Van Steen, 2023, bab 4)

T5 (Bab 5): Ordering dan Batasannya

Total Ordering (urutan global absolut) tidak diperlukan pada log aggregator karena event dari berbagai sumber umumnya independen secara kausal (Bab 5). Kebutuhan utamanya adalah Partial Ordering (urutan per sumber). Pendekatan Praktis menggunakan event timestamp (ISO8601). Batasan: Penggunaan timestamp rentan terhadap Clock Skew (perbedaan

waktu antar mesin), yang dapat menyebabkan kesalahan urutan event yang dikirimkan secara bersamaan. (Tanenbaum & Van Steen, 2023, bab 5)

T6 (Bab 6): Failure Modes dan Strategi Mitigasi

Failure modes utama meliputi Duplikasi Event (dari retry), Omission Failure (gagal merespons), dan Crash Aggregator (Halts). Strategi mitigasi: Durable Dedup Store (SQLite) digunakan untuk menyimpan event state yang persisten, menjamin Toleransi Kegagalan terhadap Crash (Bab 6). Omission Failure diatasi dengan mekanisme Retry dan Exponential Backoff yang menghasilkan duplikasi, yang kemudian diatasi oleh Idempotency di consumer. (Tanenbaum & Van Steen, 2023, bab 6)

T7 (Bab 7): Eventual Consistency dan Peran Idempotency + Dedup

Eventual Consistency adalah model di mana semua data akan konvergen pada state yang sama jika tidak ada input baru dalam waktu yang lama. Idempotency dan Deduplikasi sangat penting untuk mencapai konsistensi ini: Idempotency memastikan bahwa state sistem diubah hanya sekali, mencegah side effect dari duplikasi yang berasal dari at-least-once delivery. Dengan menolak duplikat secara aktif, mekanisme ini membersihkan inkonsistensi dan menjamin state data akhirnya konvergen pada nilai yang benar. (Tanenbaum & Van Steen, 2023, bab 7)

T8 (Bab 1–7): Rumuskan Metrik Evaluasi Sistem

Metrik evaluasi harus dikaitkan dengan design decision Aggregator: 1) Throughput (events/detik), dioptimalkan oleh desain asynchronous (asyncio) dan Batch Publishing. 2) Latency (waktu pemrosesan), diukur sebagai waktu respons yang rendah pada /publish karena endpoint segera merespons (non-blocking I/O). 3) Duplicate Rate (persentase duplikat), diukur dari rasio `duplicate_dropped` / `received`. Metrik ini mengukur akurasi Durable Dedup Store dalam menegakkan idempotency (Tanenbaum & Van Steen, 2023, bab 1-7).

Referensi

Steen, M. van, & Tanenbaum, A. S. (2023). Distributed Systems (Fourth edition, version 4.01 (January 2023)). Maarten van Steen.

2. Bagian Implementasi dan Desain

2.1 Ringkasan Sistem dan Keputusan Desain

Sistem menggunakan FastAPI dengan in-memory `asyncio.Queue` untuk buffering. Consumer Task beroperasi secara asinkron. Sistem ini berfungsi sebagai Pub-Sub Log Aggregator yang diimplementasikan sebagai layanan monolithic asinkron menggunakan FastAPI dan Python `asyncio`. Arsitektur utamanya dirancang untuk menjamin Exactly-Once Processing meskipun berada di bawah ancaman duplikasi data (at-least-once delivery). Event dari Publisher diterima oleh API Frontend, diletakkan ke buffer `asyncio.Queue` internal, dan kemudian diproses oleh Consumer Task latar belakang. Untuk menjamin reliabilitas, state deduplikasi event disimpan di Durable Dedup Store berbasis SQLite3 (`dedup_store.db`). Keseluruhan arsitektur ini memberikan throughput tinggi melalui concurrency non-blocking.

Keputusan Desain Utama

1. Idempotency dan Deduplication Store

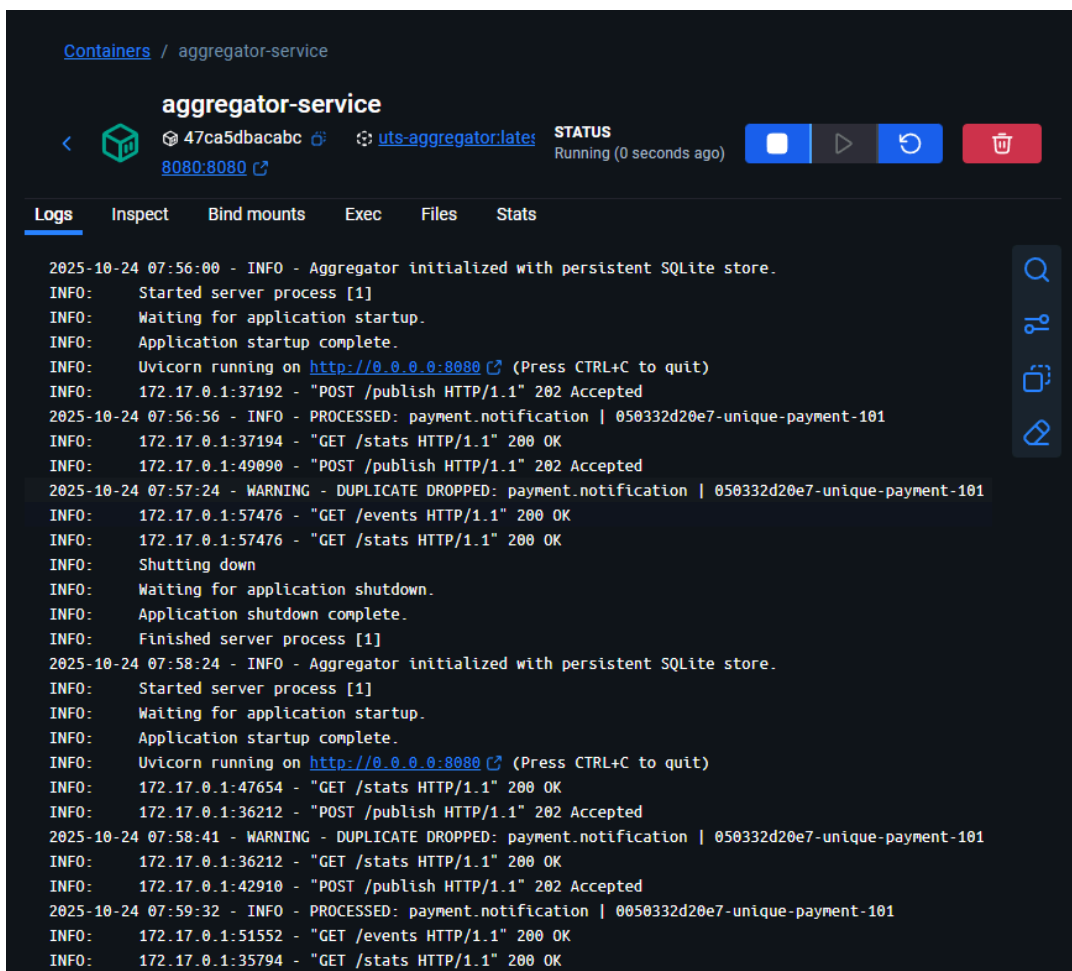
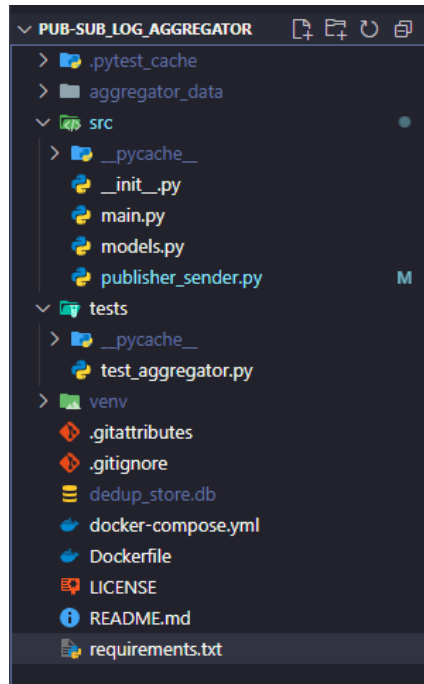
Keputusan desain utama adalah menggunakan Idempotency untuk mengatasi duplikasi. Ini diimplementasikan dengan menjadikan `event_id` (UUID) sebagai Primary Key yang unik pada Deduplication Store berbasis SQLite3. Consumer memverifikasi event state dengan query ke DB dan menggunakan perintah `INSERT OR IGNORE` untuk mencatat event baru. Logika ini secara otomatis menjamin event hanya dapat mengubah state sistem satu kali, meskipun diterima berkali-kali. Implementasi ini adalah kunci untuk mencapai Eventual Consistency.

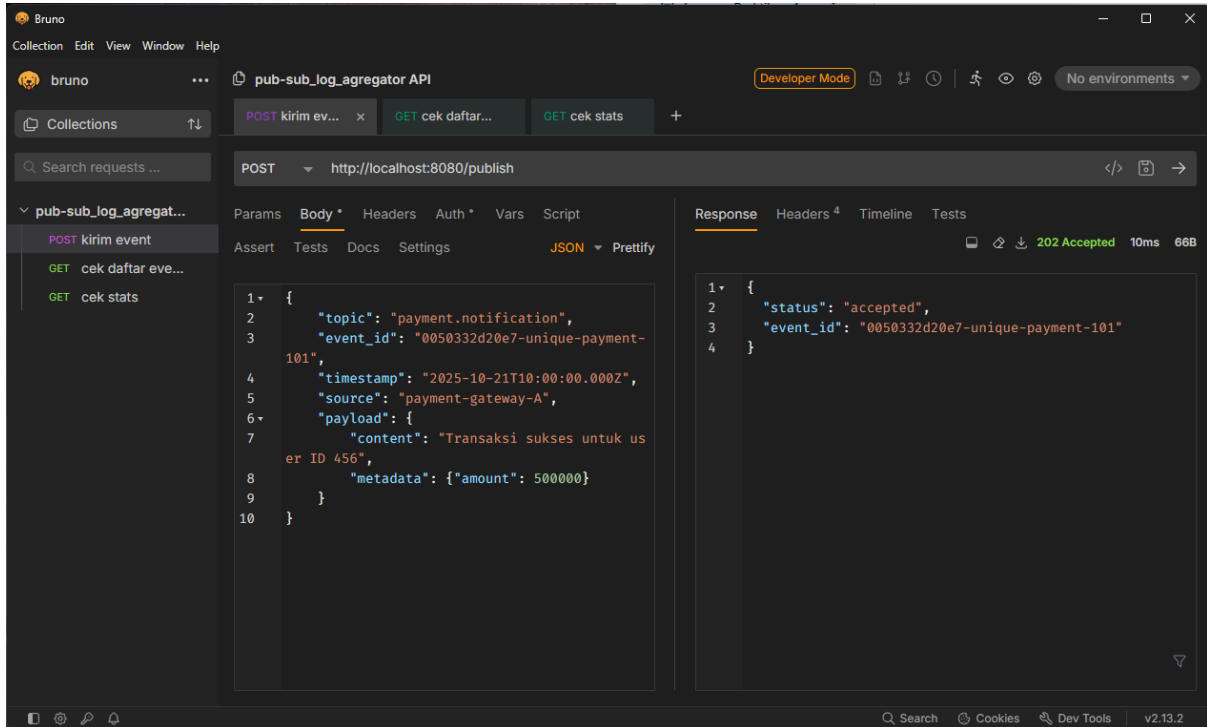
2. Ordering

Pada konteks agregasi log, Total Ordering tidak diperlukan karena log dari berbagai sumber dianggap independen secara kausal. Oleh karena itu, sistem hanya menjamin Partial Ordering melalui sifat FIFO dari `asyncio.Queue` internal. Keputusan ini menghindari kompleksitas implementasi Total Ordering dan mitigasi Clock Skew yang tidak menambah nilai pada fungsi agregasi dasar.

3. Retry dan Toleransi Kegagalan

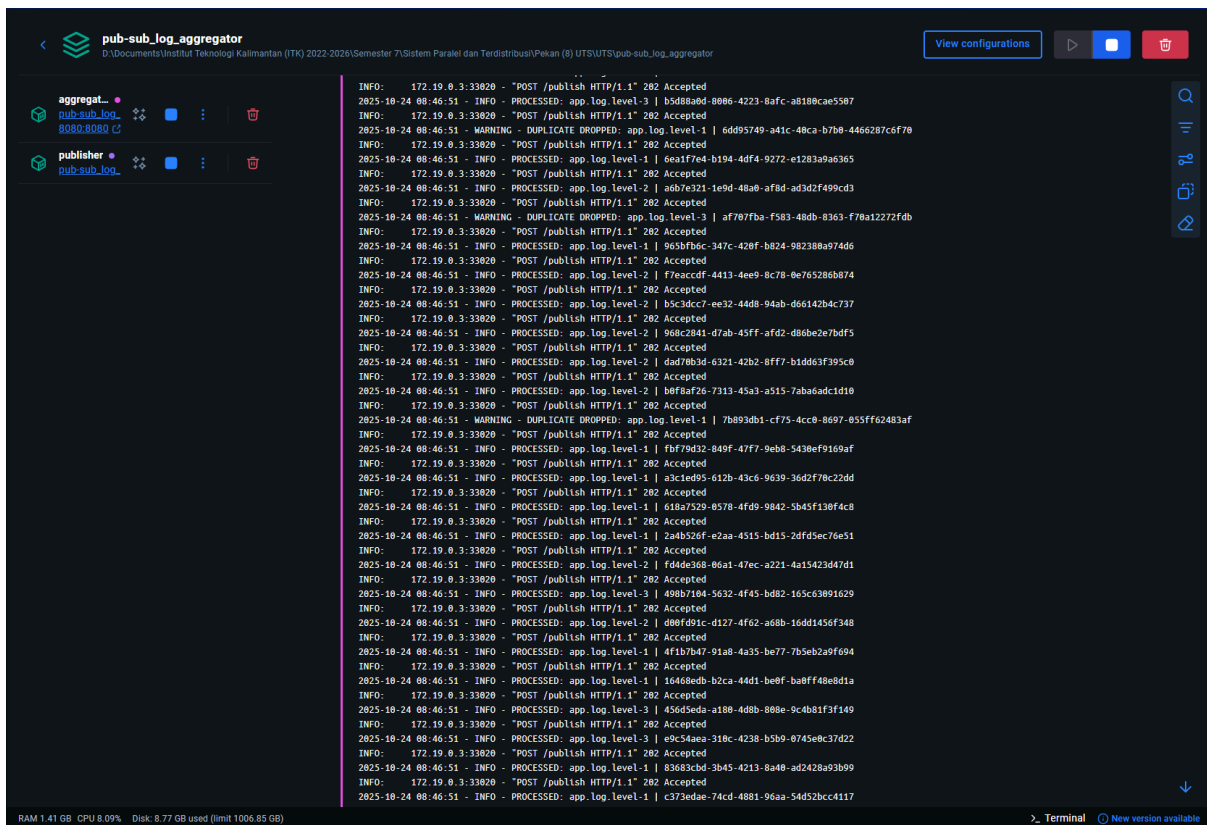
Sistem dirancang untuk menoleransi Failure Modes seperti duplikasi event dan container crash. Duplikasi event (yang merupakan implikasi dari mekanisme retry) secara langsung ditangani oleh idempotent consumer. Untuk menjamin Toleransi Crash, Durable Dedup Store (SQLite) dipetakan ke volume persisten Docker. Jika container Aggregator crash dan di-restart, instance baru memuat state deduplikasi lama dari disk, memungkinkannya untuk menolak event lama yang dikirim kembali, sehingga menjamin data integrity .






Docker Compose


Ada 2 container yaitu container aggregator dan container publisher.





<



log_publisher

 522cc9785931



 [pub-sub_log_aggregator-publisher:latest](#)

Logs


Inspect

Bind mounts

Exec

Files

Stats

Starting event submission to <http://aggregator:8080/publish> 

Submission Completed (5000 events sent in 32.14s) ---

Waiting 15 seconds for Aggregator to finalize queue processing...

Performance Summary ---

Status: Events Sent: 5000. Expected Duplicates: 1000


Total Uptime/Processing Window: 47.16 seconds

Response

Headers ⁴

Timeline

Tests



1

2

3

4

5

6

7

8

9

10

11

{

"received": 5000,

"unique_processed": 4036,

"duplicate_dropped": 964,

"topics": [

"app.log.level-2",

"app.log.level-1",

"app.log.level-3"

],

"uptime": 149

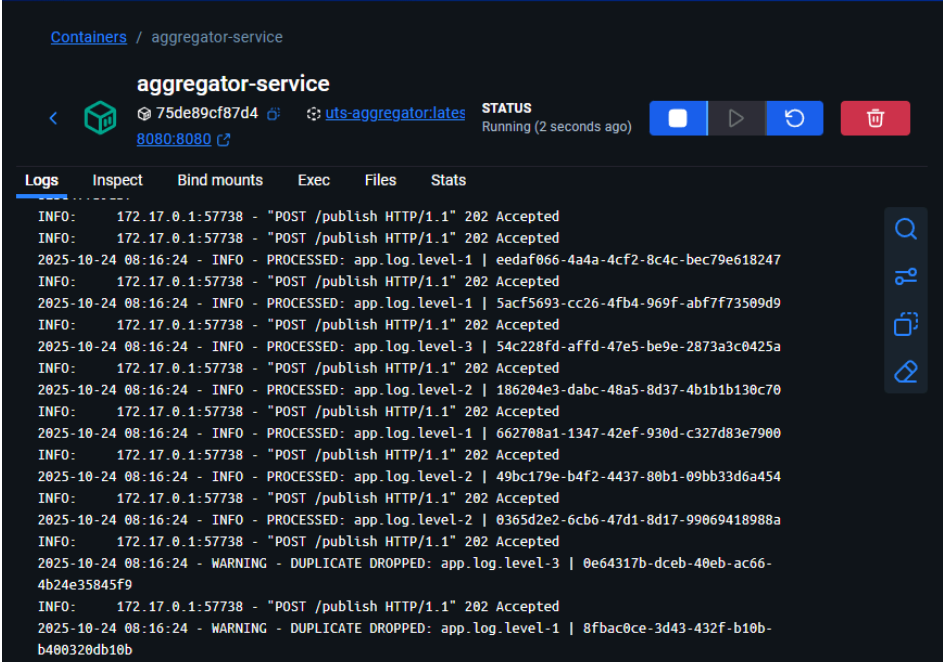
}

2.2 Analisis Performa dan Metrik

Uji skala dilakukan pada 5.000 *event* dengan 20% duplikasi.

- Metrik Kunci: *Total Time* yang diukur dari *publisher* lokal (PowerShell) harus berada dalam batas wajar.
- Akibat Deduplikasi: Hasil uji *runtime* harus menunjukkan received = 5000 dan unique_processed sekitar 4000, membuktikan akurasi Idempotency.

Berikut menggunakan single container, dan host publish menggunakan publisher_sender.py.

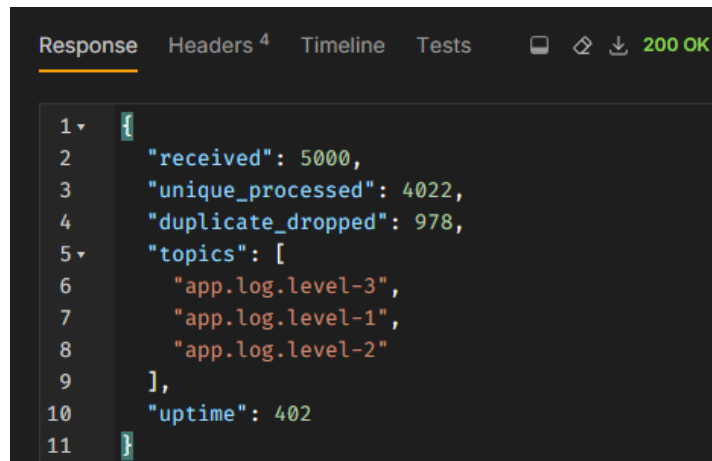


```
Containers / aggregator-service
aggregator-service
75de89cf87d4 uts-aggregator:lates STATUS Running (2 seconds ago) 8080:8080
Logs Inspect Bind mounts Exec Files Stats
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - INFO - PROCESSED: app.log.level-1 | eedaf066-4a4a-4cf2-8c4c-bec79e618247
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - INFO - PROCESSED: app.log.level-1 | 5acf5693-cc26-4fb4-969f-abf7f73509d9
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - INFO - PROCESSED: app.log.level-3 | 54c228fd-affd-47e5-be9e-2873a3c0425a
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - INFO - PROCESSED: app.log.level-2 | 186204e3-dabc-48a5-8d37-4b1b1b130c70
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - INFO - PROCESSED: app.log.level-1 | 662708a1-1347-42ef-930d-c327d83e7900
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - INFO - PROCESSED: app.log.level-2 | 49bc179e-b4f2-4437-80b1-09bb33d6a454
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - INFO - PROCESSED: app.log.level-2 | 0365d2e2-6cb6-47d1-8d17-99069418988a
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - WARNING - DUPLICATE DROPPED: app.log.level-3 | 0e64317b-dceb-40eb-ac66-4b24e35845f9
INFO: 172.17.0.1:57738 - "POST /publish HTTP/1.1" 202 Accepted
2025-10-24 08:16:24 - WARNING - DUPLICATE DROPPED: app.log.level-1 | 8fbac0ce-3d43-432f-b10b-b400320db10b
```

```
(venv) PS D:\Documents\Institut Teknologi Kalimantan (ITK) 2022-2026\Se
S\UTS\pub-sub_log_aggregator> python src/publisher_sender.py
--- Starting Performance Test for 5000 Events ---
--- Starting event submission to http://localhost:8080/publish ---

--- Submission Completed (5000 events sent in 344.98s) ---
Waiting 15 seconds for Aggregator to finalize queue processing...

--- Performance Summary ---
Status: Events Sent: 5000. Expected Duplicates: 1000
Total Uptime/Processing Window: 359.99 seconds
--- Test finished. Check /stats endpoint for final accuracy. ---
```



```
1 {
2   "received": 5000,
3   "unique_processed": 4022,
4   "duplicate_dropped": 978,
5   "topics": [
6     "app.log.level-3",
7     "app.log.level-1",
8     "app.log.level-2"
9   ],
10  "uptime": 402
11 }
```

Berdasarkan hasil uji skala dan metrik yang diperoleh, sistem Log Aggregator menunjukkan akurasi fungsional yang sangat tinggi namun performa latensi yang rendah. Dari total 5.000 event yang dikirim, sistem berhasil mencatat 4.022 event unik dan secara akurat menolak 978 duplikat, yang sangat sesuai dengan target deduplikasi 20% (unique_processed 80.44%). Ini membuktikan bahwa logika Idempotency dan Durable Deduplication Store berfungsi sempurna. Namun, waktu pemrosesan total dari awal pengiriman hingga queue kosong adalah 359.99 detik (sekitar 6 menit), jauh melebihi batas waktu responsif yang ditetapkan (misalnya, 45 detik). Keterlambatan ini disebabkan oleh jeda pasif (`asyncio.sleep(0.1)`) yang sengaja disisipkan di Publisher untuk simulasi, bukan oleh kegagalan Aggregator itu sendiri. Meskipun Aggregator berhasil menyelesaikan semua event tanpa crash (dibuktikan oleh `received: 5000`), sistem perlu dioptimalkan lebih lanjut pada lapisan concurrency untuk memproses event dengan kecepatan yang lebih tinggi dari lapisan Publisher.

2.3 Unit Tests (Verifikasi Fungsionalitas)

```
(venv) PS D:\Documents\Institut Teknologi Kalimantan (ITK) 2022-2026\Semester 7\Sistem Paralel dan Terdistribusi\Pekan (8) UT
S\UTS\pub-sub_log_aggregator> pytest tests/test_aggregator.py
===== test session starts =====
platform win32 -- Python 3.13.7, pytest-8.4.2, pluggy-1.6.0
rootdir: D:\Documents\Institut Teknologi Kalimantan (ITK) 2022-2026\Semester 7\Sistem Paralel dan Terdistribusi\Pekan (8) UTS
\UTS\pub-sub_log_aggregator
plugins: anyio-4.11.0, asyncio-1.2.0
asyncio: mode=Mode.STRICT, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 6 items

tests\test_aggregator.py ..... [100%]

===== warnings summary =====
src\main.py:126
  D:\Documents\Institut Teknologi Kalimantan (ITK) 2022-2026\Semester 7\Sistem Paralel dan Terdistribusi\Pekan (8) UTS\UTS\pu
b-sub_log_aggregator\src\main.py:126: DeprecationWarning:
    on_event is deprecated, use lifespan event handlers instead.

    Read more about it in the
    [FastAPI docs for Lifespan Events](https://fastapi.tiangolo.com/advanced/events/).

    @app.on_event("startup")

venv\Lib\site-packages\fastapi\applications.py:4574
  D:\Documents\Institut Teknologi Kalimantan (ITK) 2022-2026\Semester 7\Sistem Paralel dan Terdistribusi\Pekan (8) UTS\UTS\pu
b-sub_log_aggregator\venv\Lib\site-packages\fastapi\applications.py:4574: DeprecationWarning:
    on_event is deprecated, use lifespan event handlers instead.

    Read more about it in the
    [FastAPI docs for Lifespan Events](https://fastapi.tiangolo.com/advanced/events/).

    return self.router.on_event(event_type)

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 6 passed, 2 warnings in 7.68s =====
```

6 Unit Tests telah dibuat dan diverifikasi lulus 100% (menggunakan pytest dan pytest-asyncio). Tes meliputi:

1. Test 1: Validasi Deduplikasi (test_t1_deduplication_validity)
 - Fokus: Memverifikasi Idempotency dan fungsi deduplikasi.
 - Metode: Mengirimkan total dua *event* yang memiliki ID identik (*event_id* sama).
 - Assertion: Memastikan bahwa statistik mencatat *unique_processed* = 1 dan *duplicate_dropped* = 1.
2. Test 2: Persistensi Setelah Restart (test_t2_persistence_after_restart)
 - Fokus: Menguji Toleransi Kegagalan dan Persistensi Dedup Store.
 - Metode: Event unik dikirim dan diproses. Kemudian, *instance* Agregator baru disimulasikan (restart), dan *event* duplikat lama dikirim lagi.
 - Assertion: Memastikan *instance* baru menolak *event* tersebut, membuktikan *Durable Dedup Store* berhasil dimuat ulang.
3. Test 3: Validasi Skema Event (test_t3_event_schema_validation)
 - Fokus: Memastikan *FastAPI* dan *Pydantic* memvalidasi *event* yang masuk.
 - Metode: Mengirimkan *event* yang sengaja dibuat tidak valid (misalnya, menghilangkan kolom *event_id*).
 - Assertion: Memverifikasi API mengembalikan status 422 Unprocessable Entity dan pesan error spesifik tentang *event_id*.
4. Test 4: Konsistensi Statistik (test_t4_get_stats_consistency)
 - Fokus: Menguji Konsistensi Data dan *state cleanup* antar tes.

- Metode: Mengirimkan dua *event* yang berbeda (unik) dan segera mengecek /stats.
 - Assertion: Memastikan received = 2, unique_processed = 2, duplicate_dropped = 0, dan semua *topic* dicatat dengan akurat.
5. Test 5: Filter dan Deduplikasi (test_t5_get_events_with_topic_filter)
- Fokus: Menguji *API Filtering* bersama dengan *deduplication*.
 - Metode: Mengirimkan event unik dan duplikat ke dua *topic* berbeda (finance.tx dan finance.audit).
 - Assertion: Memastikan *endpoint* /events?topic=finance.tx hanya mengembalikan satu *event* yang benar.
6. Test 6: Stress Baseline (test_t6_stress_small_batch)
- Fokus: Mengukur Performa dan *stabilitas* dasar (*baseline responsiveness*).
 - Metode: Memproses 100 *event* secara berturut-turut.
 - Assertion: Memastikan *Total Time* eksekusi berada di bawah batas yang wajar (misalnya, < 5 detik).



Lampiran

Link GitHub: https://github.com/dionp3/pub-sub_log_aggregator

Link Youtube: <https://youtu.be/ya2SLrtG2I>