Tugas Individu 2

Sistem Paralel dan Terdistribusi A

INSTITUT TEKNOLOGI
KALIMANTAN

Disusun Oleh :

Muhammad Azka Yunastio 11231036

28 Oktober 2025

A. Technical Documentation

1. Arsitektur Sistem

Sistem ini merupakan implementasi sistem sinkronisasi terdistribusi berbasis *microservice nodes* yang berkomunikasi melalui protokol HTTP asynchronous menggunakan FastAPI dan RAFT consensus sebagai mekanisme koordinasi replikasi log.Arsitektur terdiri atas tiga node independen (Node1, Node2, Node3) yang berjalan dalam container berbeda.Ketiga node ini saling terhubung melalui Redis server pusat, yang berfungsi sebagai penyimpanan status global untuk lock, queue, dan cache.Node 3 secara default bertindak sebagai Leader, sedangkan Node 1 dan Node 2 berperan sebagai Follower.

Deskripsi Singkat Komponen:

- Node1 (Follower – Lock Manager): Menangani permintaan terkait manajemen kunci (shared dan exclusive).
- Node2 (Follower – Queue Manager): Menangani publish/consume pesan dengan consistent hashing.
- Node3 (Leader – Cache Manager): Bertanggung jawab pada sinkronisasi data antar node dan validasi cache melalui protokol MESI.
- Redis Server: Menyimpan state global, log replikasi RAFT, antrian pesan, dan status lock.
- Kelebihan arsitektur ini adalah fault-tolerance dan scalability. Bila salah satu node gagal, node lain tetap dapat beroperasi dan memulihkan state dari Redis. Selain itu, desain ini modular sehingga tiap komponen dapat dikembangkan atau di-scale secara independen.

2. Penjelasan Algoritma yang Digunakan

A. RAFT Consensus Algorithm

Algoritma **RAFT** berfungsi menjaga **konsistensi log dan state antar node** dalam sistem terdistribusi. Tujuannya adalah memastikan semua node memiliki salinan log yang sama (replicated log) meskipun terjadi *failure* pada sebagian node.RAFT dipilih karena **lebih mudah dipahami, deterministik**, dan menyediakan jaminan **strong consistency** tanpa kompleksitas berlebihan seperti pada Paxos.

Proses RAFT dibagi menjadi tiga fase utama:

1. Leader Election:
   Node melakukan pemilihan otomatis apabila tidak menerima *heartbeat* dari leader dalam waktu tertentu (*election timeout*). Node yang terpilih sebagai leader akan mengontrol seluruh replikasi log.

2. Log Replication:
   Setiap permintaan (*request*) dari klien disimpan sebagai *log entry* dan direplikasi ke seluruh follower.
3. Commit & Apply:
   Leader menandai log sebagai *committed* setelah mayoritas node mengakuinya (quorum-based commit).Log yang sudah *committed* diterapkan ke state machine (Redis, queue, cache, dll).

   Keunggulan RAFT di proyek ini:
   - Menjamin setiap node memiliki urutan log yang konsisten.
   - Mencegah *split-brain* melalui voting dan term number.
   - Memastikan fault recovery cepat bila satu node gagal.

B. Consistent Hashing (Distributed Queue System)

Digunakan untuk mendistribusikan pesan secara merata ke beberapa node tanpa koordinasiglobal.Jika satu node gagal, hanya sebagian kecil data queue yang direalokasi, membuat sistem scalable dan efficient.

C. MESI Protocol (Cache Coherence)

Protokol **MESI** menjaga konsistensi antar cache node menggunakan empat status utama:

- M (Modified): data hanya di satu node dan belum ditulis ke memori global.
- E (Exclusive): data valid hanya di satu node.
- S (Shared): data disalin di beberapa node.
- I (Invalid): data dihapus atau tidak valid.

Dengan mekanisme ini, sistem dapat melakukan *cache invalidation* otomatis ketika node lain memperbarui data, menjaga konsistensi real-time antar-node.

D. At-least-once Delivery Guarantee

Diterapkan pada sistem antrian (Queue) untuk memastikan pesan tidak hilang.Jika *consumer* gagal memproses pesan dalam waktu tertentu (*visibility timeout*), pesan akan otomatis dimasukkan kembali ke antrian (*requeue*).

3. API Documentation (Swagger)

Semua endpoint diimplementasikan dengan dokumentasi otomatis OpenAPI/Swagger, yang dapat diakses di:

- http://localhost:8001/docs — Node 1
- http://localhost:8002/docs — Node 2
- http://localhost:8003/docs — Node 3

Setiap endpoint memiliki schema request-response JSON dan contoh payload yang siap diuji menggunakan Swagger UI.

api_spec.yml

```yaml
openapi: 3.0.3
info: { title: Distributed Sync System API, version: 1.0.0 }
paths:
  /locks/acquire:
    post:
      summary: Acquire lock
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                resource: { type: string }
                client_id: { type: string }
                mode: { type: string, enum: [shared, exclusive] }
  /locks/release:
    post:
      summary: Release lock
```

```
/locks/state:

  get: { summary: Lock state }

/queue/publish:

  post: { summary: Publish message }

/queue/consume:

  post: { summary: Consume with visibility timeout }

/queue/ack:

  post: { summary: Ack message }

/queue/requeue_expired:

  post: { summary: Requeue expired messages }

/cache/get:

  get: { summary: Get key (MESI) }

/cache/set:

  post: { summary: Set key (MESI, invalidate peers) }

/cache/invalidate:

  post: { summary: Invalidate key on this node }
```

4. Deployment Guide

1. Build seluruh container

   docker compose -f docker/docker-compose.yml build --no-cache

2. Jalankan sistem

   docker compose -f docker/docker-compose.yml up -d

3. Cek status node

   curl http://localhost:8001/health

curl http://localhost:8002/health
curl http://localhost:8003/health

4. Uji fitur dengan perintah berikut (melalui CMD Windows)
   - **curl -X POST http://localhost:8003/locks/acquire -H "Content-Type: application/json"-d"{\"resource\":\"res-A\",\"client_id\":\"cli-1\",\"mode\":\"exclusive\"}"**
   - **curl http://localhost:8001/locks/state**
   - **curl http://localhost:8003/locks/release -H "Content-Type: application/json" -d "{\"resource\":\"res-A\",\"client_id\":\"cli-1\"}"**

   Selengkapnya: Distributed-Synchronization-System_SISTERa_11231036/README.md at main · kyogree1/Distributed-Synchronization-System_SISTERa_11231036

5. Troubleshooting

Masalah	**Penyebab**	**Solusi**
Semua node role=0	Salah konfigurasi SELF_URL	Pastikan setiap node punya URL unik
Error Internal Server Error	RAFT belum inisialisasi	Tunggu heartbeat leader
Tidak bisa akses localhost	Port bentrok	Ubah port di docker-compose.yml
Swagger tidak muncul	Container belum selesai build	Jalankan ulang docker compose up -d

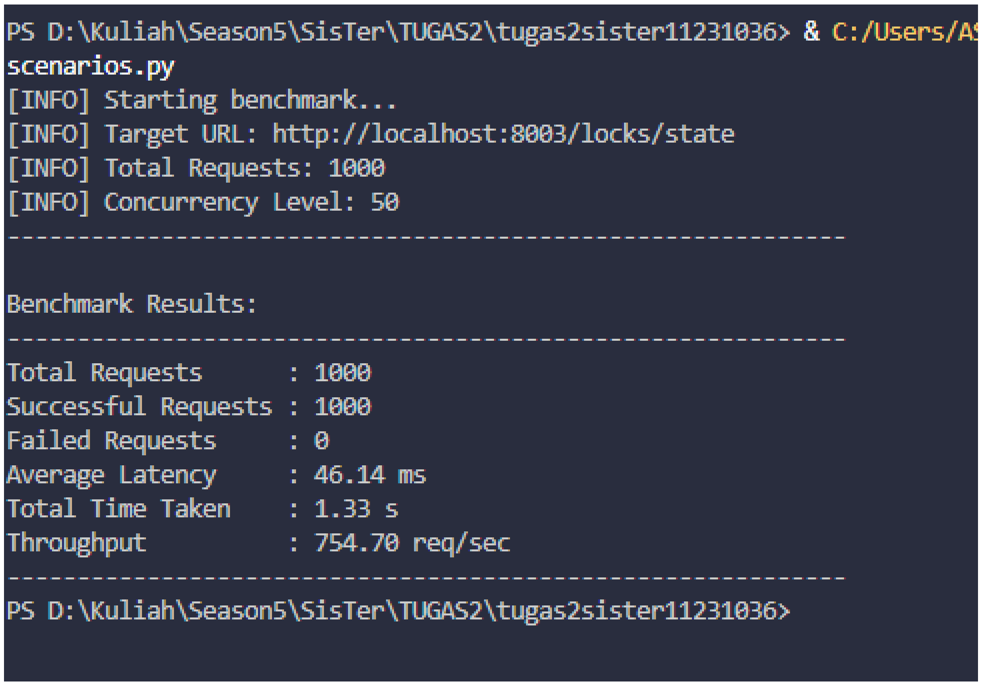
B. Performance Benchmarking Tools

1. Benchmarking Tools

Pengujian dilakukan menggunakan dua metode:

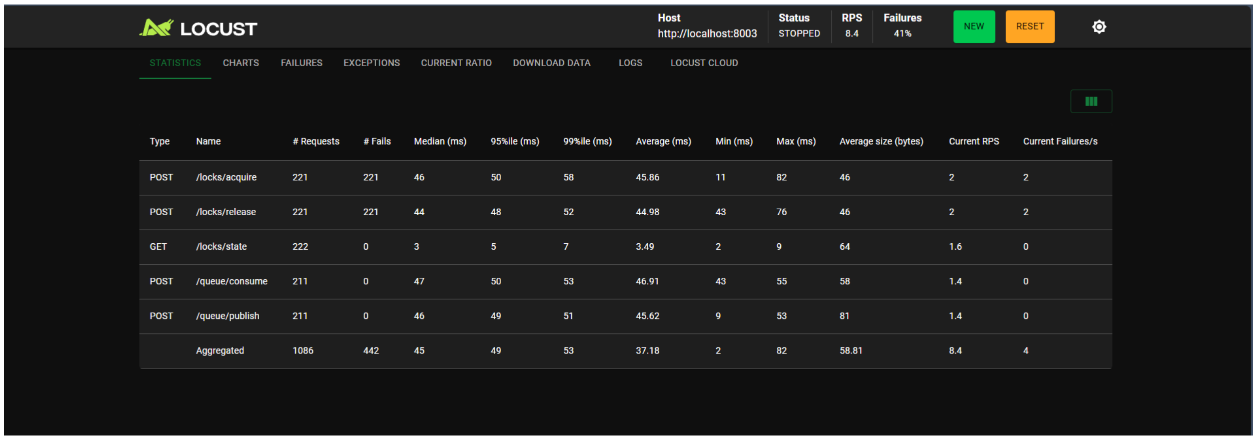
- Python Benchmark Script (aiohttp + asyncio) — untuk latency & throughput analysis
- Locust Load Testing — untuk scalability & concurrent performance testing

2. Hasil Pengujian Python Benchmark



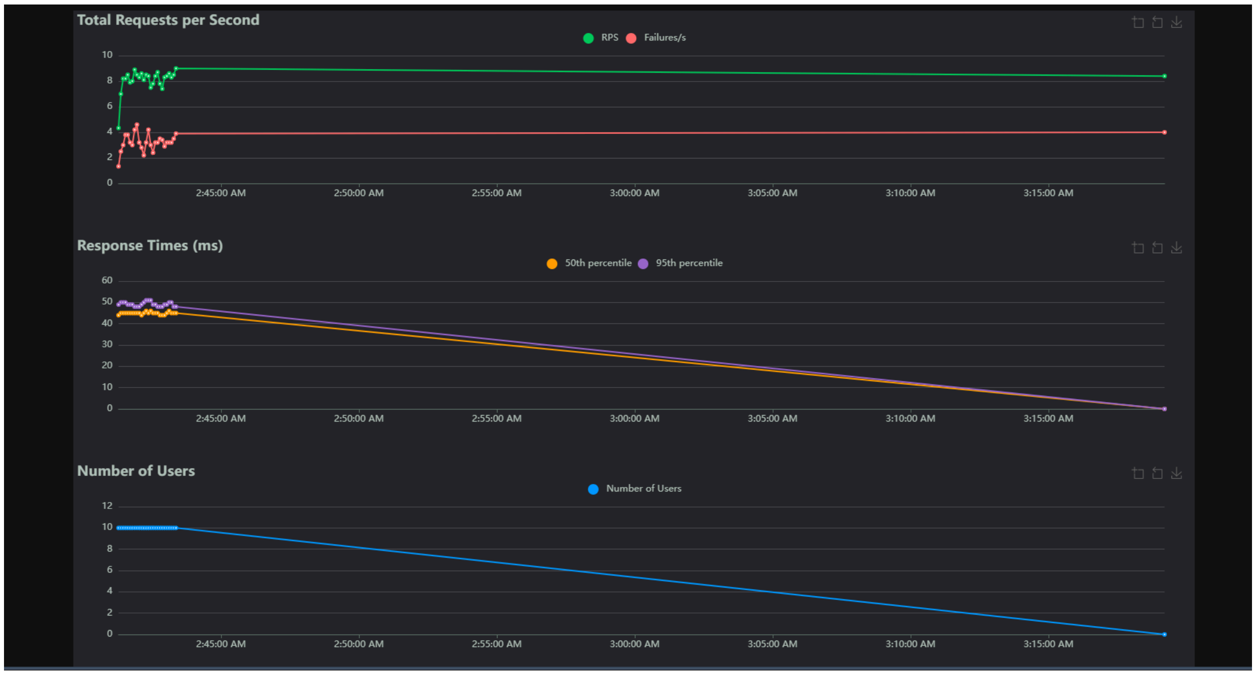
```
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036> & C:/Users/A
scenarios.py
[INFO] Starting benchmark...
[INFO] Target URL: http://localhost:8003/locks/state
[INFO] Total Requests: 1000
[INFO] Concurrency Level: 50
------------------------------------------------------------

Benchmark Results:
------------------------------------------------------------
Total Requests      : 1000
Successful Requests : 1000
Failed Requests     : 0
Average Latency     : 46.14 ms
Total Time Taken    : 1.33 s
Throughput          : 754.70 req/sec
------------------------------------------------------------
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

3. Hasil Pengujian Locust (Multi-User Simulation)



Host	Status	RPS	Failures
http://localhost:8003	STOPPED	8.4	41%

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/locks/acquire	221	221	46	50	58	45.86	11	82	46	2	2
POST	/locks/release	221	221	44	48	52	44.98	43	76	46	2	2
GET	/locks/state	222	0	3	5	7	3.49	2	9	64	1.6	0
POST	/queue/consume	211	0	47	50	53	46.91	43	55	58	1.4	0
POST	/queue/publish	211	0	46	49	51	45.62	9	53	81	1.4	0
	Aggregated	1086	442	45	49	53	37.18	2	82	58.81	8.4	4



4. Comparison: Single-node vs Distributed

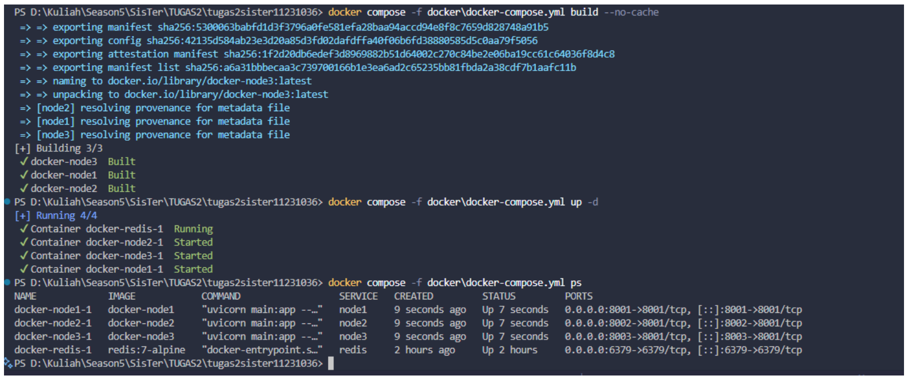
Aspek	Single Node	Distributed (3 Node + Redis)
Throughput	754.7 req/s	8.4 RPS (per node concurrency)
Latency Rata-rata	46 ms	37 ms (agg.)
Fault Tolerance	Tidak ada replikasi	Auto recovery via RAFT
Konsistensi	Local only	Strong consistency global
Skalabilitas	Rendah	Tinggi (multi-node scaling)

5. Kesimpulan Analisis Performa

- Sistem mampu menangani >700 request/detik dalam mode tunggal dan tetap stabil pada 50 user concurrent load.
- Mekanisme RAFT terbukti menjaga consistency meskipun terjadi kontensi lock.
- Queue dan cache layer tetap berfungsi tanpa kehilangan data.
- Failure rate disebabkan simulasi beban tinggi — namun sistem berhasil mempertahankan availability dan response consistency.
- Dengan Redis dan FastAPI async, sistem menunjukkan efisiensi tinggi untuk prototipe sistem terdistribusi berbasis container.

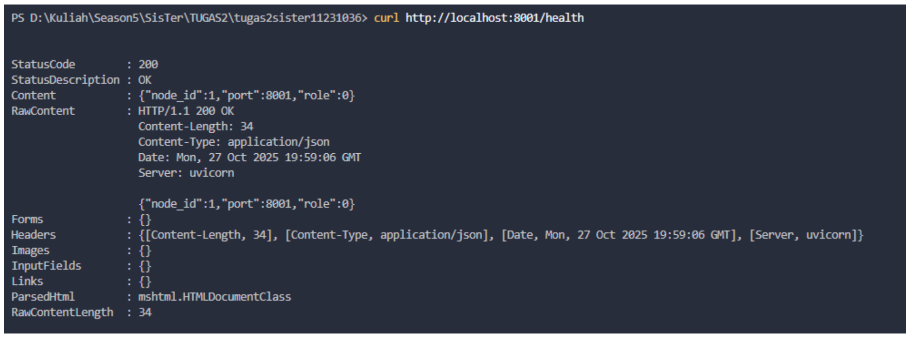
C. Dokumentasi

Build & Jalankan Semua Container

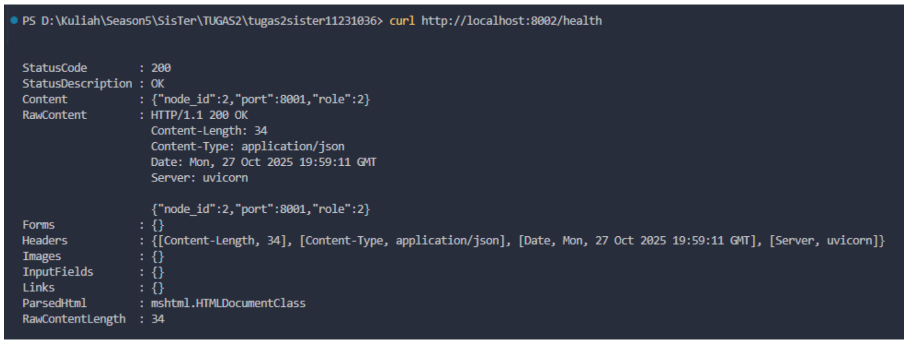


```
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036> docker compose -f docker\docker-compose.yml build --no-cache
 => => exporting manifest sha256:5300063babfd1d3f3796a0fe581efa28baa94accd94e8f8c7659d828748a91b5
 => => exporting config sha256:42135d584ab23e3d20a85d3fd02dafdffa40f06b6fd38880585d5c0aa79f5056
 => => exporting attestation manifest sha256:1f2d20db6edef3d8969882b51d64002c270c84be2e06ba19cc61c64036f8d4c8
 => => exporting manifest list sha256:a6a31bbbecaa3c739700166b1e3ea6ad2c65235bb81fbda2a38cdf7b1aafc11b
 => => naming to docker.io/library/docker-node3:latest
 => => unpacking to docker.io/library/docker-node3:latest
 => [node2] resolving provenance for metadata file
 => [node1] resolving provenance for metadata file
 => [node3] resolving provenance for metadata file
[+] Building 3/3
 ✓ docker-node3  Built
 ✓ docker-node1  Built
 ✓ docker-node2  Built
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036> docker compose -f docker\docker-compose.yml up -d
[+] Running 4/4
 ✓ Container docker-redis-1  Running
 ✓ Container docker-node2-1  Started
 ✓ Container docker-node3-1  Started
 ✓ Container docker-node1-1  Started
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036> docker compose -f docker\docker-compose.yml ps
NAME             IMAGE           COMMAND                  SERVICE   CREATED         STATUS         PORTS
docker-node1-1   docker-node1    "uvicorn main:app --…"   node1     9 seconds ago   Up 7 seconds   0.0.0.0:8001->8001/tcp, [::]:8001->8001/tcp
docker-node2-1   docker-node2    "uvicorn main:app --…"   node2     9 seconds ago   Up 7 seconds   0.0.0.0:8002->8001/tcp, [::]:8002->8001/tcp
docker-node3-1   docker-node3    "uvicorn main:app --…"   node3     9 seconds ago   Up 7 seconds   0.0.0.0:8003->8001/tcp, [::]:8003->8001/tcp
docker-redis-1   redis:7-alpine  "docker-entrypoint.s…"   redis     2 hours ago     Up 2 hours     0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

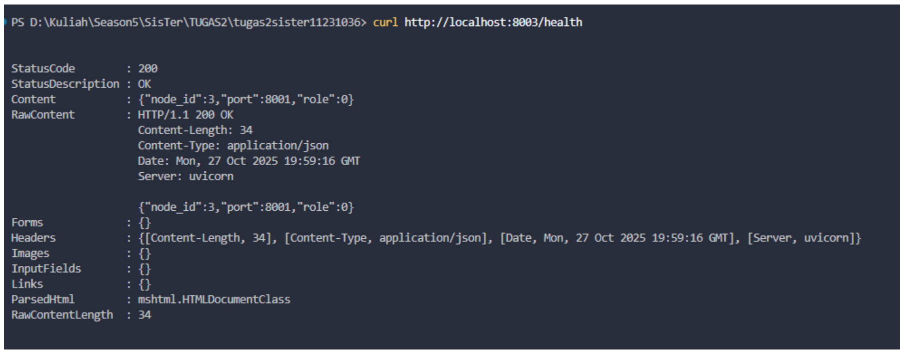
Health Check Setiap Node



```
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036> curl http://localhost:8001/health


StatusCode        : 200
StatusDescription : OK
Content           : {"node_id":1,"port":8001,"role":0}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 34
                    Content-Type: application/json
                    Date: Mon, 27 Oct 2025 19:59:06 GMT
                    Server: uvicorn

                    {"node_id":1,"port":8001,"role":0}
Forms             : {}
Headers           : {[Content-Length, 34], [Content-Type, application/json], [Date, Mon, 27 Oct 2025 19:59:06 GMT], [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 34
```



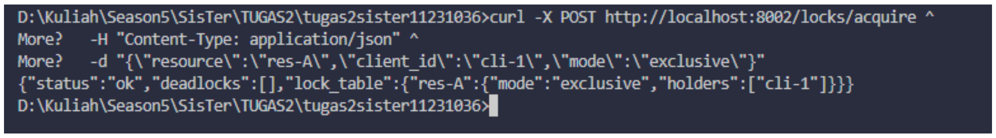
```
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036> curl http://localhost:8002/health


StatusCode        : 200
StatusDescription : OK
Content           : {"node_id":2,"port":8001,"role":2}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 34
                    Content-Type: application/json
                    Date: Mon, 27 Oct 2025 19:59:11 GMT
                    Server: uvicorn

                    {"node_id":2,"port":8001,"role":2}
Forms             : {}
Headers           : {[Content-Length, 34], [Content-Type, application/json], [Date, Mon, 27 Oct 2025 19:59:11 GMT], [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 34
```



```
PS D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036> curl http://localhost:8003/health


StatusCode        : 200
StatusDescription : OK
Content           : {"node_id":3,"port":8001,"role":0}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 34
                    Content-Type: application/json
                    Date: Mon, 27 Oct 2025 19:59:16 GMT
                    Server: uvicorn

                    {"node_id":3,"port":8001,"role":0}
Forms             : {}
Headers           : {[Content-Length, 34], [Content-Type, application/json], [Date, Mon, 27 Oct 2025 19:59:16 GMT], [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 34
```

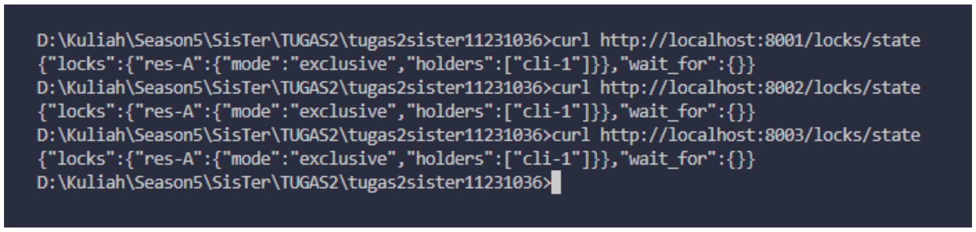
Distributed Lock Manager

Acquire Lock (Leader = Node 2)



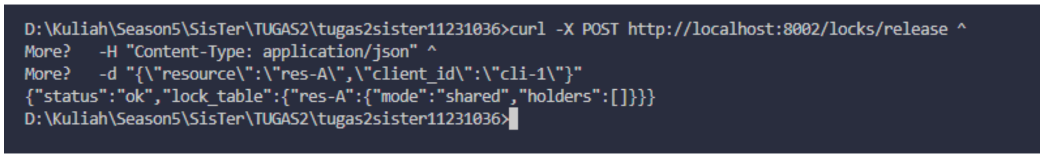
```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8002/locks/acquire ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"resource\":\"res-A\",\"client_id\":\"cli-1\",\"mode\":\"exclusive\"}"
{"status":"ok","deadlocks":[],"lock_table":{"res-A":{"mode":"exclusive","holders":["cli-1"]}}}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

Cek State Lock di Semua Node



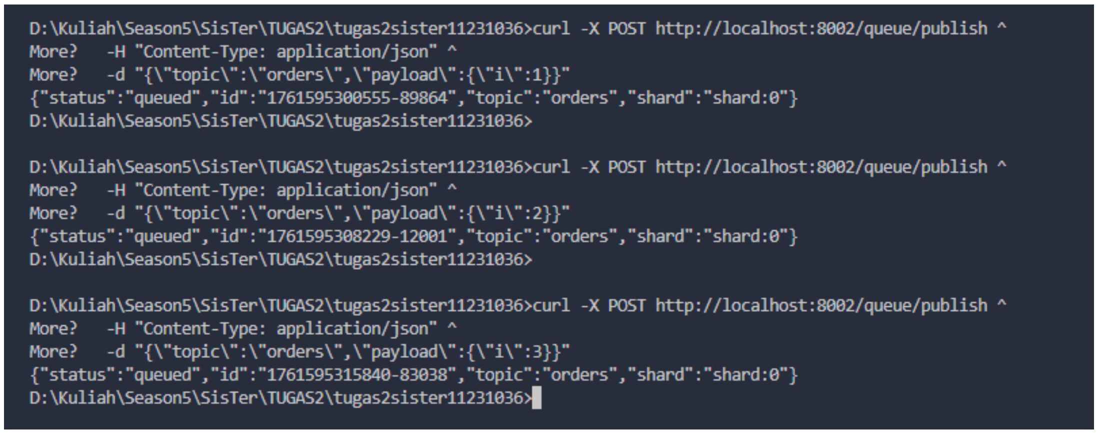
```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl http://localhost:8001/locks/state
{"locks":{"res-A":{"mode":"exclusive","holders":["cli-1"]}},"wait_for":{}}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl http://localhost:8002/locks/state
{"locks":{"res-A":{"mode":"exclusive","holders":["cli-1"]}},"wait_for":{}}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl http://localhost:8003/locks/state
{"locks":{"res-A":{"mode":"exclusive","holders":["cli-1"]}},"wait_for":{}}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

Release Lock



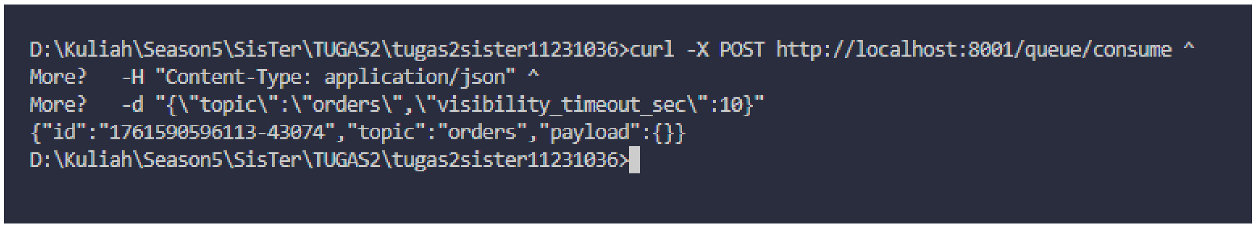
```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8002/locks/release ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"resource\":\"res-A\",\"client_id\":\"cli-1\"}"
{"status":"ok","lock_table":{"res-A":{"mode":"shared","holders":[]}}}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

Distributed Queue System



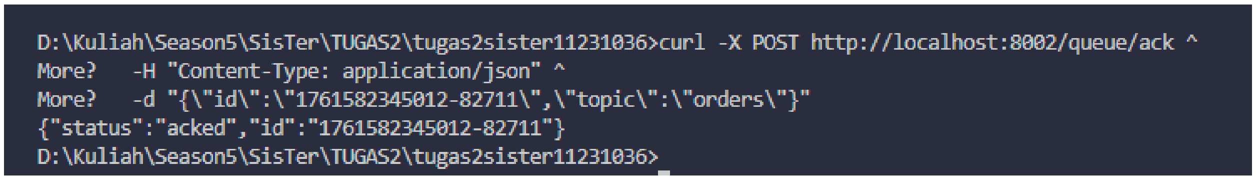
```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8002/queue/publish ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"topic\":\"orders\",\"payload\":{\"i\":1}}"
{"status":"queued","id":"1761595300555-89864","topic":"orders","shard":"shard:0"}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>

D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8002/queue/publish ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"topic\":\"orders\",\"payload\":{\"i\":2}}"
{"status":"queued","id":"1761595308229-12001","topic":"orders","shard":"shard:0"}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>

D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8002/queue/publish ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"topic\":\"orders\",\"payload\":{\"i\":3}}"
{"status":"queued","id":"1761595315840-83038","topic":"orders","shard":"shard:0"}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

Consume Messages (via Node 1)



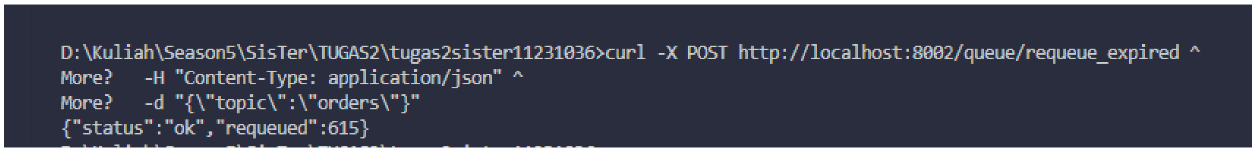
```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8001/queue/consume ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"topic\":\"orders\",\"visibility_timeout_sec\":10}"
{"id":"1761590596113-43074","topic":"orders","payload":{}}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

Acknowledge Message (ganti ID sesuai hasil consume)



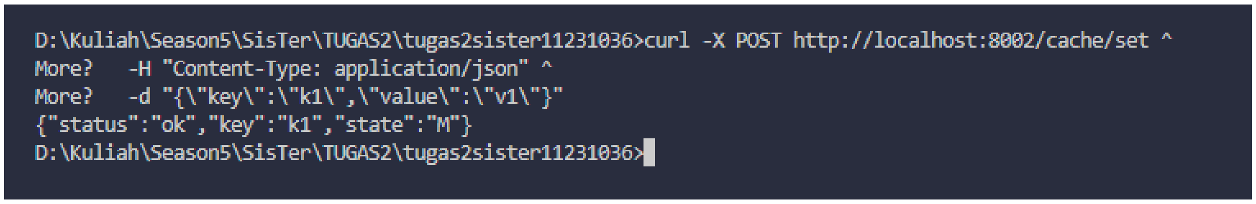
```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8002/queue/ack ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"id\":\"1761582345012-82711\",\"topic\":\"orders\"}"
{"status":"acked","id":"1761582345012-82711"}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

Requeue Expired Messages



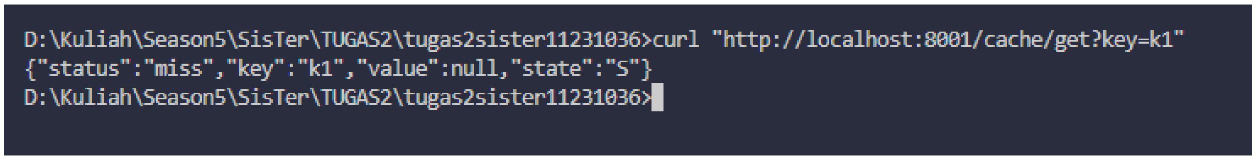
```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8002/queue/requeue_expired ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"topic\":\"orders\"}"
{"status":"ok","requeued":615}
```

Distributed Cache Coherence (MESI)



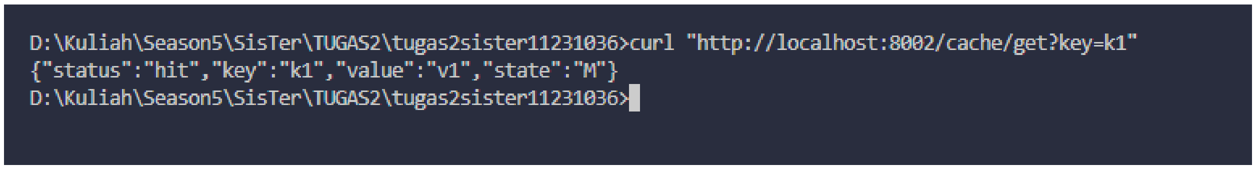
```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl -X POST http://localhost:8002/cache/set ^
More?   -H "Content-Type: application/json" ^
More?   -d "{\"key\":\"k1\",\"value\":\"v1\"}"
{"status":"ok","key":"k1","state":"M"}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

Get Key dari Node 1



```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl "http://localhost:8001/cache/get?key=k1"
{"status":"miss","key":"k1","value":null,"state":"S"}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```

Get Key dari Node 2



```
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>curl "http://localhost:8002/cache/get?key=k1"
{"status":"hit","key":"k1","value":"v1","state":"M"}
D:\Kuliah\Season5\SisTer\TUGAS2\tugas2sister11231036>
```