

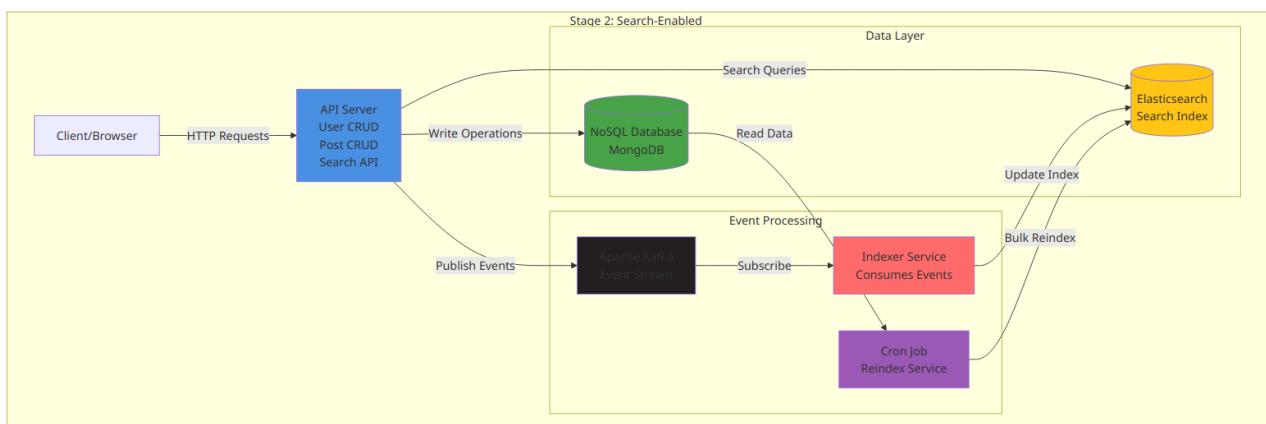
Test creaeting pdf from markdown

Generating single mermaid chart 🎨 Rendered Mermaid diagram 1 Generating single mermaid chart 🎨 Rendered Mermaid diagram 2 Generating single mermaid chart 🎨 Rendered Mermaid diagram 3 Generating single mermaid chart 🎨 Rendered Mermaid diagram 4 Generating single mermaid chart 🎨 Rendered Mermaid diagram 5 Generating single mermaid chart 🎨 Rendered Mermaid diagram 6 Generating single mermaid chart 🎨 Rendered Mermaid diagram 7 Generating single mermaid chart 🎨 Rendered Mermaid diagram 8 Generating single mermaid chart ⚠️ Failed to render Mermaid diagram 9, keeping as code Generating single mermaid chart 🎨 Rendered Mermaid diagram 10 Generating single mermaid chart 🎨 Rendered Mermaid diagram 11 Generating single mermaid chart 🎨 Rendered Mermaid diagram 12 Generating single mermaid chart 🎨 Rendered Mermaid diagram 13 # Social Media System Design - Progressive Stages

## Stage 2: Search with Elasticsearch

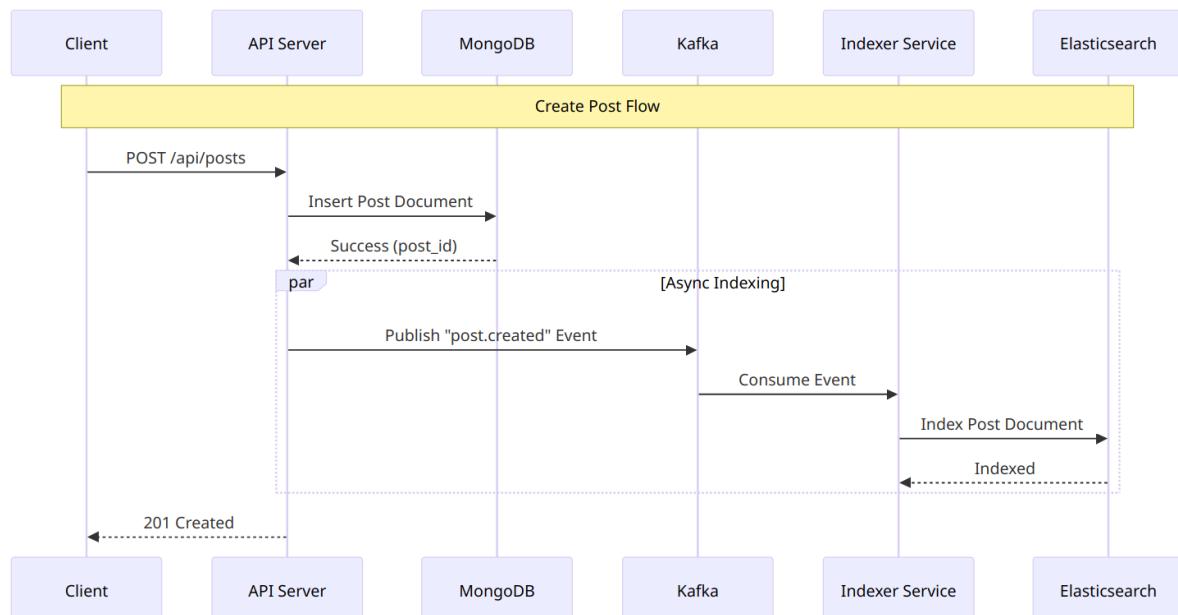
### Architecture Overview

Added full-text search capability with Elasticsearch, migrated to NoSQL, and implemented event-driven indexing.



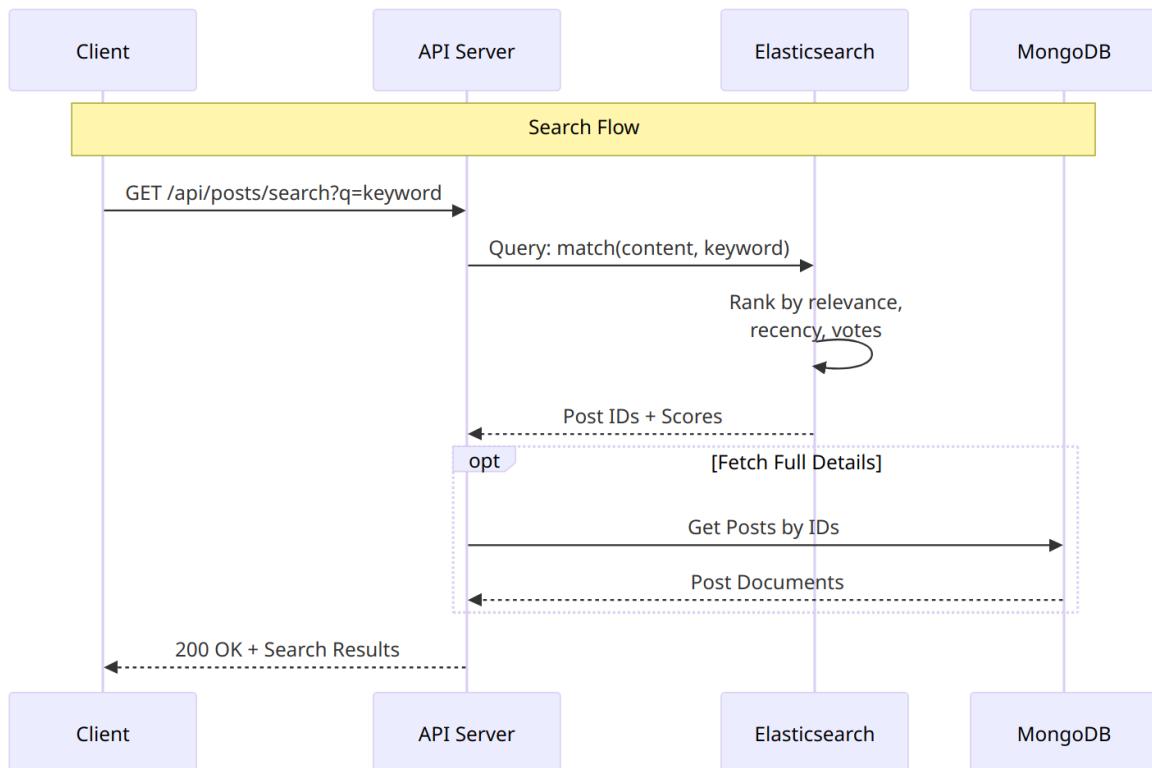
Diagram

## Data Flow - Create Post with Indexing



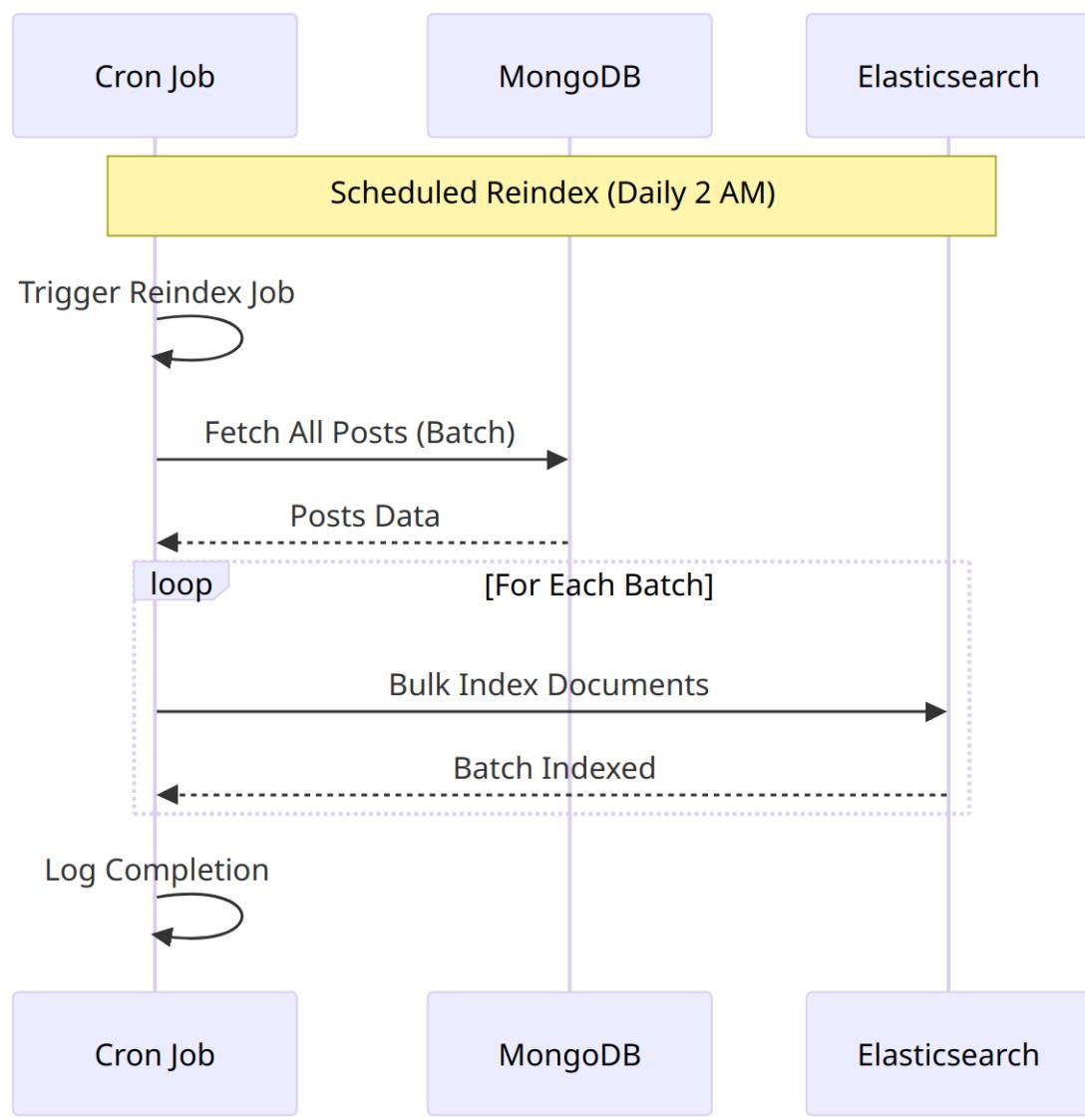
Diagram

## Data Flow - Search Posts



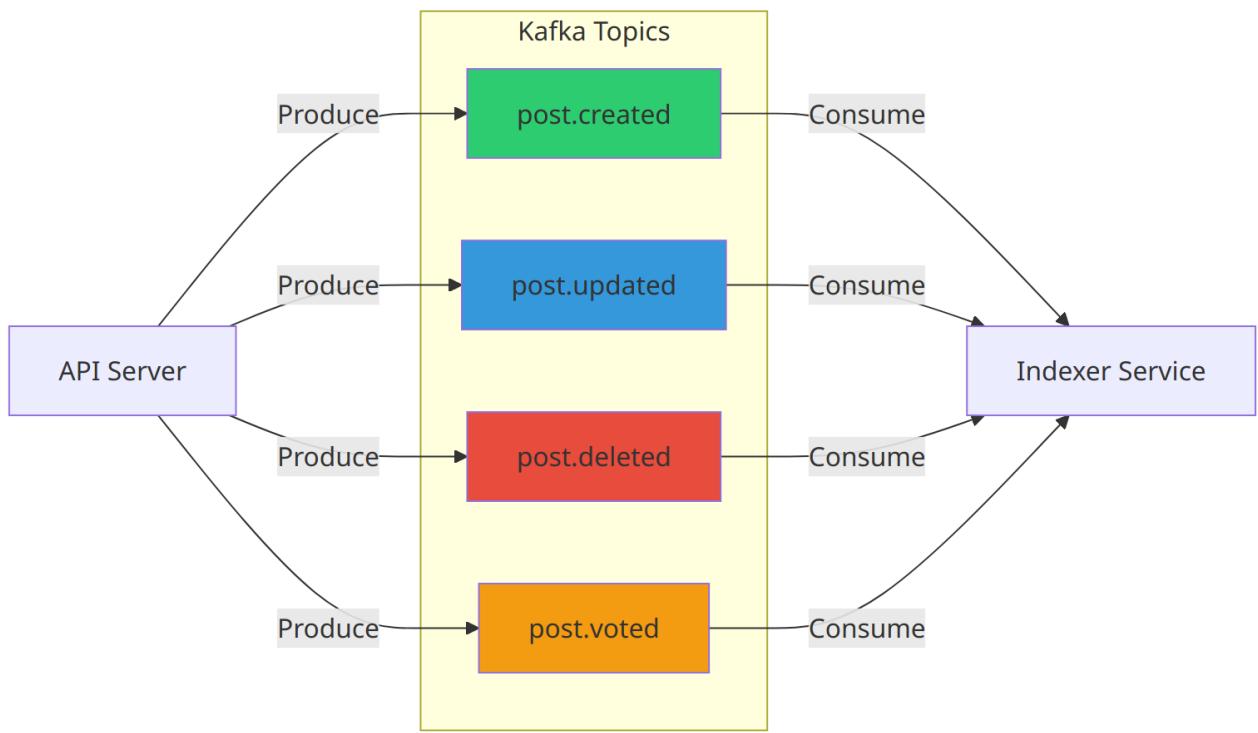
Diagram

## Reindexing Flow



Diagram

## Kafka Topics Structure

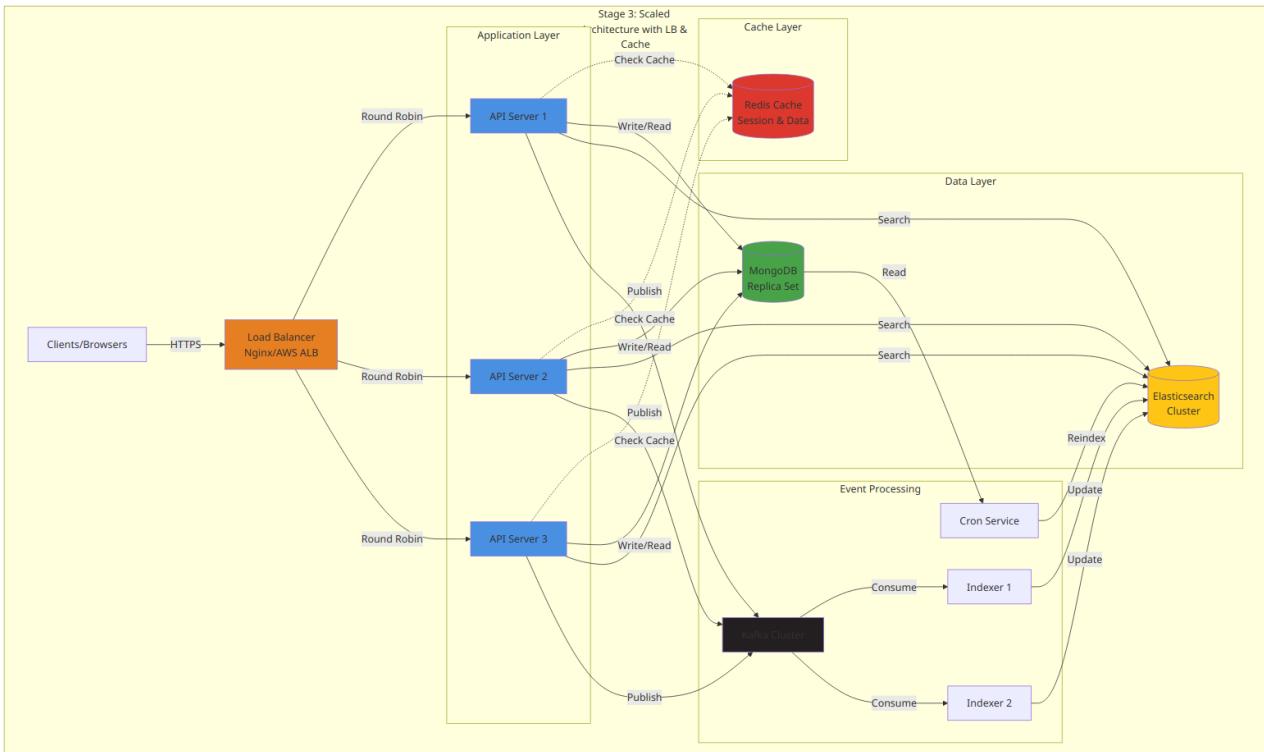


Diagram

## Stage 3: Load Balancing & Caching

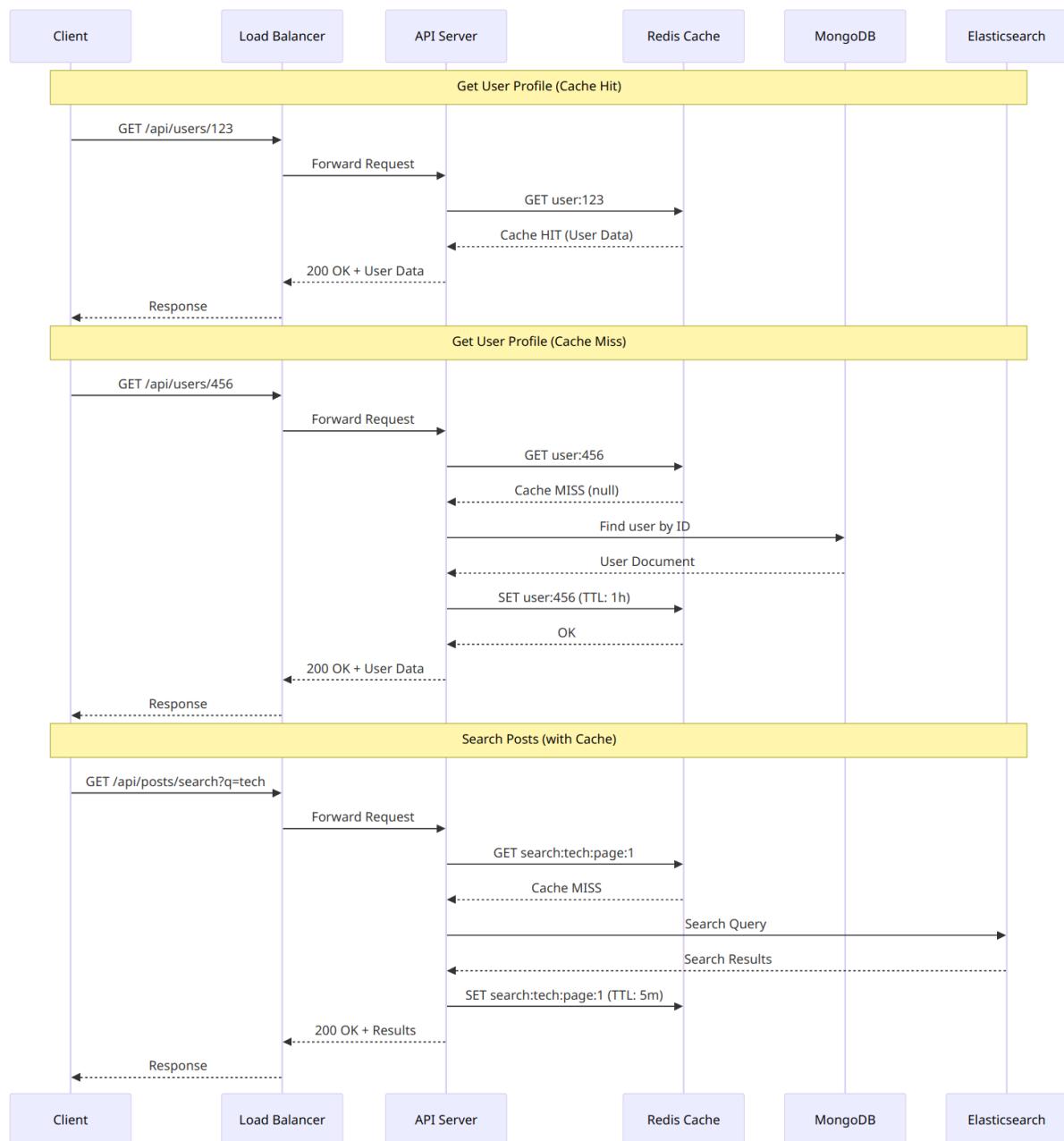
### Architecture Overview

Production-ready architecture with load balancing, caching layer, and horizontal scaling.



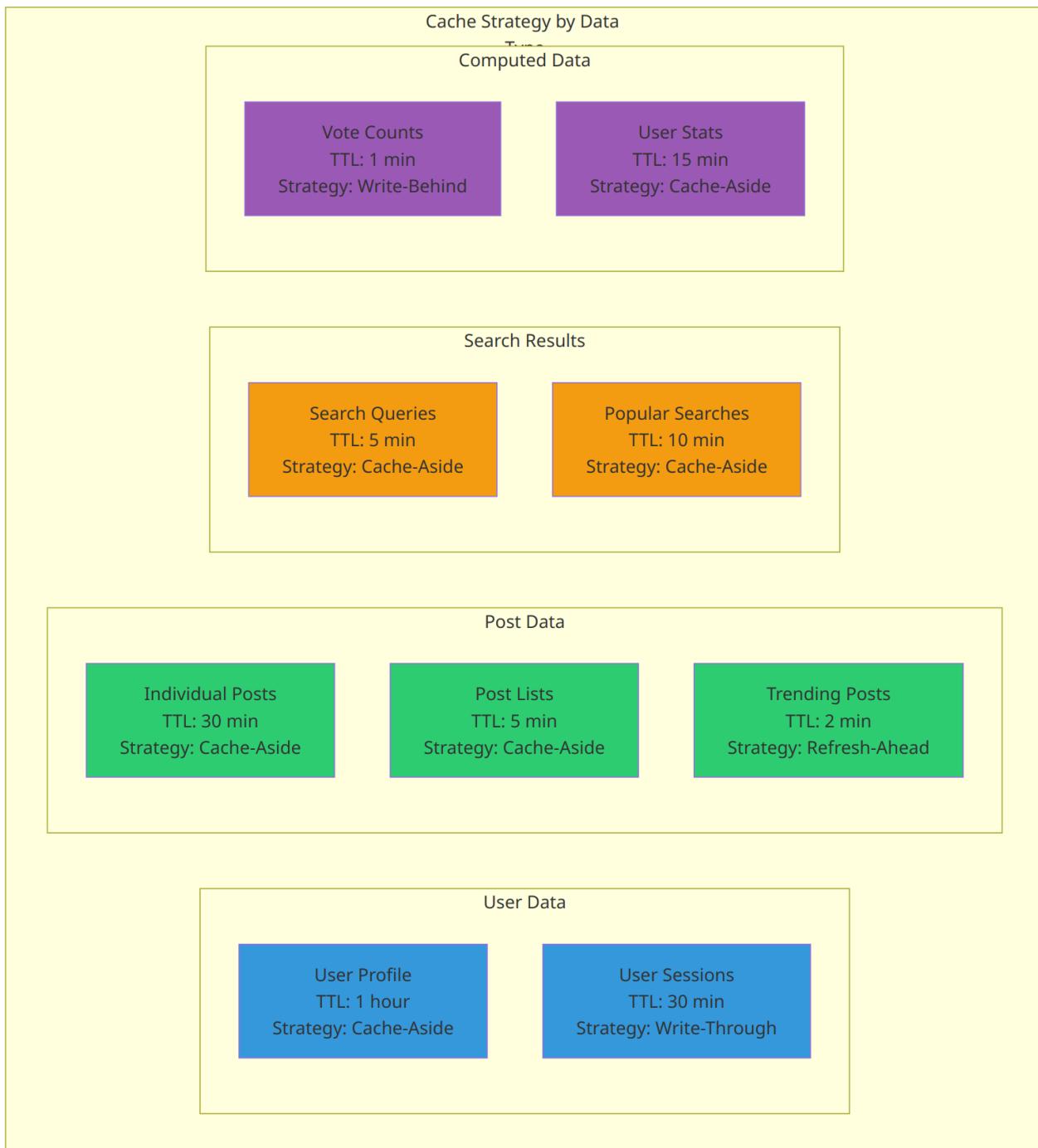
Diagram

## Detailed Request Flow with Cache



Diagram

## Cache Strategy Details



Diagram

# Load Balancer Configuration

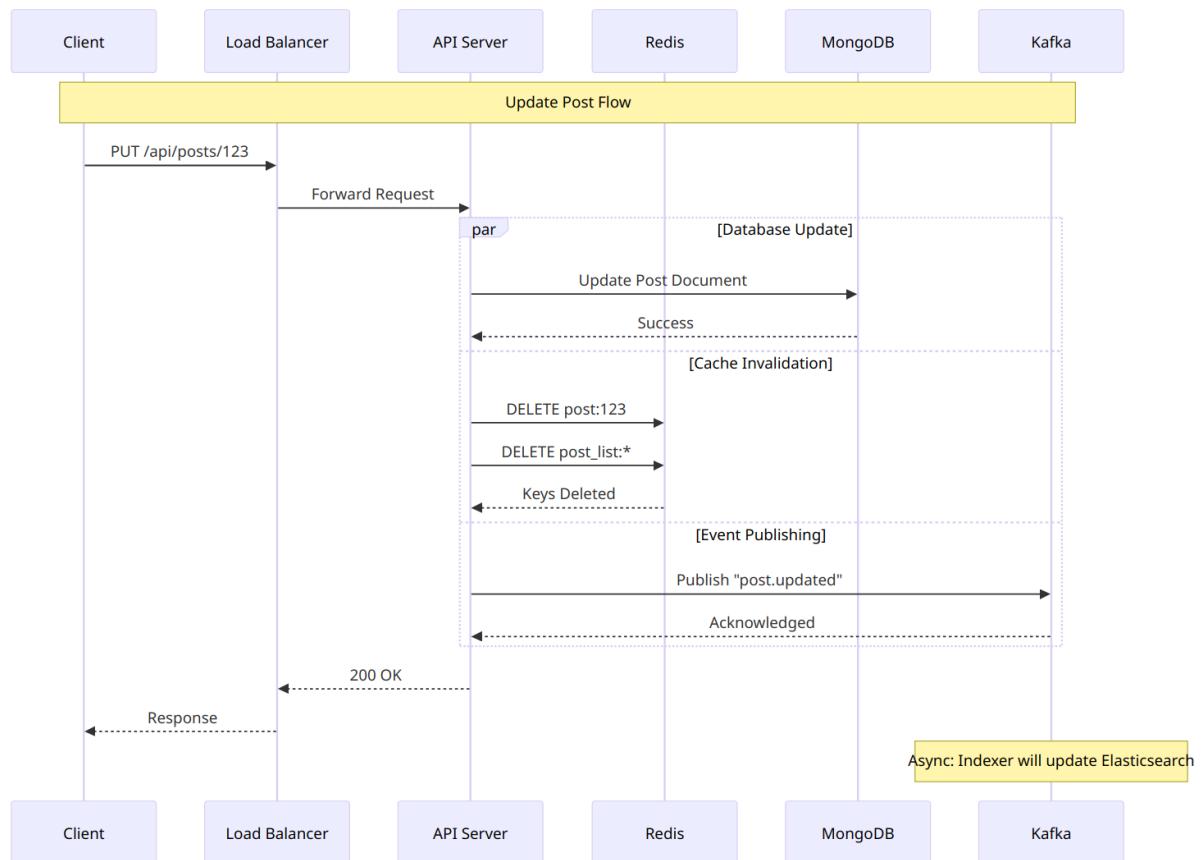
```
graph LR
    subgraph "Load Balancer Strategies"
        LB[Load Balancer]

        subgraph "Routing Rules"
            R1[/api/users/* → Round Robin]
            R2[/api/posts/* → Round Robin]
            R3[/api/search/* → Least Connections]
            R4[/api/uploads/* → IP Hash]
        end

        subgraph "Health Checks"
            H1[GET /health every 10s]
            H2[Timeout: 5s]
            H3[Unhealthy threshold: 3]
        end

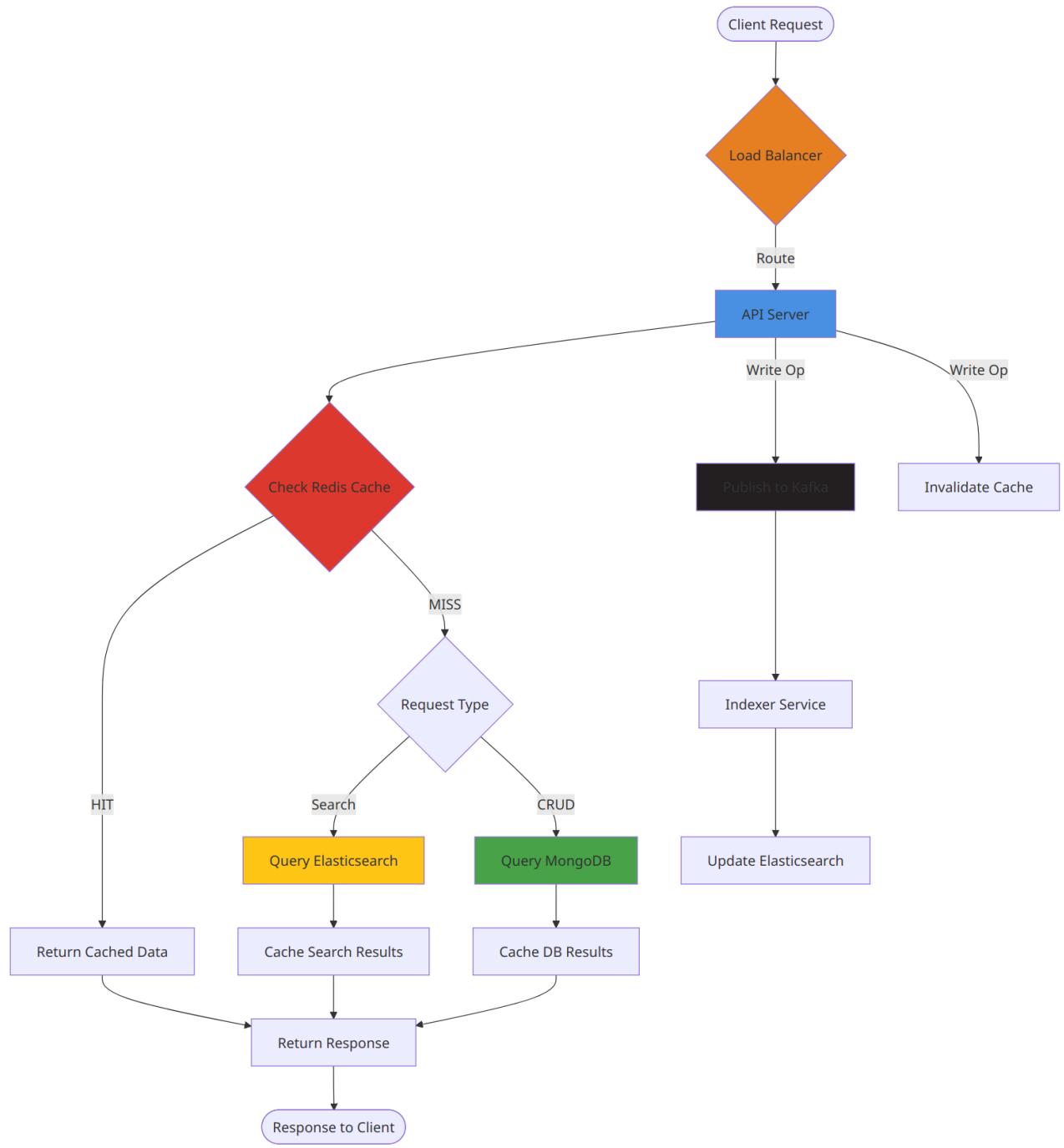
        LB --> R1
        LB --> R2
        LB --> R3
        LB --> R4
        LB -.→|Monitor| H1
        LB -.→|Monitor| H2
        LB -.→|Monitor| H3
    end
```

## Write Operation with Cache Invalidation



Diagram

## Complete Data Flow



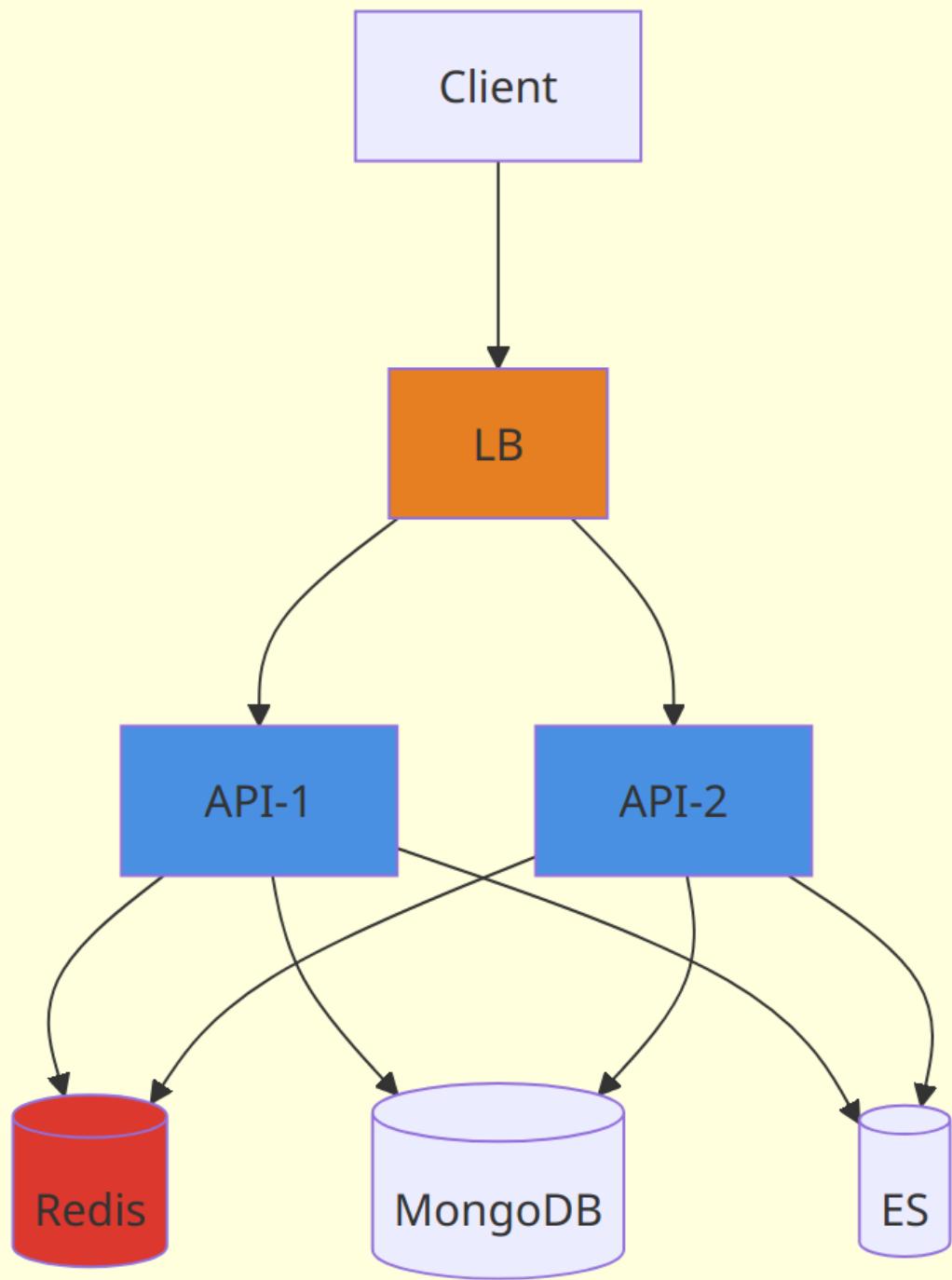
Diagram

## Comparison: Stage Evolution

---

### Architecture Complexity

### Stage 3



### Stage 2

Client

12

Diagram

## Performance Metrics Comparison

## Metrics Evolution

Stage 1

---

Response Time: 200ms

Throughput: 100 req/s

Availability: 95%

Added Search

Stage 2

---

Response Time: 150ms

Search Time: 50ms

Throughput: 300 req/s

Availability: 97%

Added LB & Cache

## Key Features by Stage

FEATURE	STAGE 1	STAGE 2	STAGE 3
User CRUD	✓	✓	✓
Post CRUD	✓	✓	✓
Full-Text Search	✗	✓	✓
Search Ranking	✗	✓ (Relevance, Recency, Votes)	✓
Real-time Indexing	✗	✓ (Kafka)	✓
Batch Reindexing	✗	✓ (Cron)	✓
Load Balancing	✗	✗	✓
Caching	✗	✗	✓ (Redis)
Horizontal Scaling	✗	Partial	✓
High Availability	✗	✗	✓

# Technology Stack Summary

Technology Stack			
Layer	Stage 1	Stage 2	Stage 3
Load Bal.	-	-	Nginx/ALB
API	Node.js	Node.js	Node.js (3x)
Cache	-	-	Redis
Database	PostgreSQL	MongoDB	MongoDB Cluster
Search	-	Elasticsearch	ES Cluster
Queue	-	Kafka	Kafka Cluster
Indexer	-	Node.js	Node.js (2x)
Scheduler	-	Cron	Cron Service