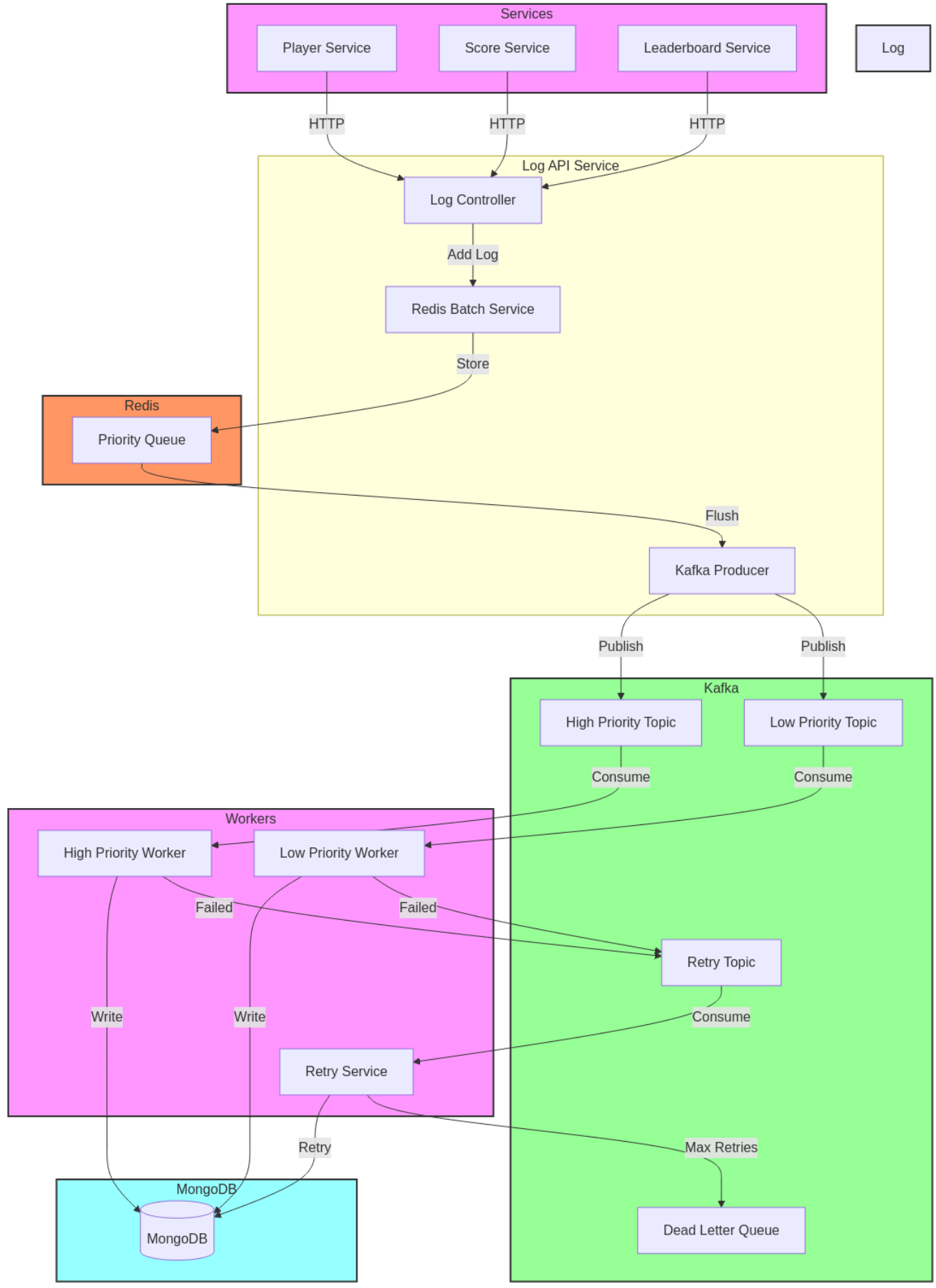
Log-Api Service

1. **Log Ingestion (Log API Service)**:
   - Services (player, score, leaderboard) send logs to the log-api-service via HTTP
   - The `LogController` receives logs and forwards them to the `RedisBatchService`
   - The `RedisBatchService` uses Redis sorted sets to batch and prioritize logs
2. **Batching and Prioritization (RedisBatchService)**:
   - Logs are stored in Redis with priority scores
   - Priority is determined by log type (CRASH, critical, error) and content
   - Uses a distributed lock to ensure only one worker processes logs at a time
   - Periodically flushes logs to Kafka based on batch size or timeout
3. **Log Processing (Log Worker Services)**:
   - Two types of workers: high-priority and low-priority
   - Each worker consumes from its respective Kafka topic
   - Uses rate limiting and concurrency control for MongoDB writes
   - Failed writes are sent to a retry queue
4. **Retry Mechanism (Retry Service)**:
   - Handles failed log writes
   - Implements exponential backoff
   - After max retries, sends logs to Dead Letter Queue (DLQ)

Here's the mermaid diagram illustrating the flow:



Key Features of the System:

1. **Priority-Based Processing**:
   - Logs are scored based on type and content
   - High priority logs (CRASH, critical) are processed first

- Prevents starvation of low priority logs

2. **Batching**:
   - Logs are batched in Redis before being sent to Kafka
   - Configurable batch size and timeout
   - Reduces load on Kafka and MongoDB
3. **Distributed Processing**:
   - Uses Redis locks for coordination
   - Multiple workers can process logs in parallel
   - Rate limiting and concurrency control
4. **Reliability**:
   - Retry mechanism for failed writes
   - Dead Letter Queue for failed retries
   - Distributed rate limiting
5. **Scalability**:
   - Horizontal scaling of workers
   - Distributed coordination
   - Efficient batching

The system is designed to handle high volumes of logs while ensuring important logs are processed quickly and reliably. The use of Redis for batching and prioritization, combined with Kafka for message queuing, provides a robust and scalable solution for log processing.