Job Processing System

A High-Performance Task Queue & Job Processing System built with Spring Boot, Kafka, Redis, and PostgreSQL.

This system handles long-running tasks like report generation, email sending, or data export asynchronously, ensuring your application remains responsive and scalable.

Features

- Accepts job requests asynchronously
- Queues jobs for efficient processing
- Handles retries and failures
- Fast job status lookups via Redis caching
- Job metadata persisted in PostgreSQL
- Dead Letter Queue for permanently failed jobs
- Demonstrates microservices patterns and production-grade architecture

---

Architecture

Client (Postman/GUI)

|

v

Job API Service (Spring Boot)

|

v

Kafka Topic: job-queue

|

v

Worker Service (Spring Boot)

|

v

PostgreSQL (Job Table) Redis (Job Status)

|

v

Dead Letter Queue (Kafka Topic)

API Endpoints

1. Submit Job

POST /api/jobs

Request Example:

```
{
  "jobType": "REPORT_GENERATION",
  "payload": {
    "userId": 42,
    "dateRange": "last_30_days"
  }
}
```

Response Example:

```
{
  "jobId": "a12f-93kd-88sa",
  "status": "QUEUED"
}
```

2. Get Job Status

GET /api/jobs/{jobId}

In Progress:

{

```
  "jobId": "a12f-93kd-88sa",
  "status": "IN_PROGRESS",
  "retries": 1,
  "result": null
}
```

Completed:

```
{
  "jobId": "a12f-93kd-88sa",
  "status": "COMPLETED",
  "retries": 1,
  "result": {
    "fileUrl": "/reports/report_42.pdf"
  }
}
```

Job Status Lifecycle

```
QUEUED -> IN_PROGRESS -> COMPLETED
          |
          v
       FAILED -> RETRYING -> DEAD_LETTER
```

Failed jobs are retried up to N times.

Jobs exceeding retries go to Dead Letter Queue.

Database Schema

Table: jobs

Column	Type	Description
job_id	UUID	Primary key
job_type	String	Type of job

payload_json Text Input payload

status Enum Current status

retries Int Number of retry attempts

result_json Text Job result

error_message String Error if job failed

created_at Timestamp Job creation time

updated_at Timestamp Last update time

Setup & Run

Clone repository

git clone https://github.com/username/JobProcessingSystem.git

cd JobProcessingSystem

Build project

./mvnw clean install

Run Spring Boot app

./mvnw spring-boot:run

Note: Make sure Redis and Kafka are running in background

Access APIs via Postman or Swagger:

http://localhost:8080/swagger-ui.html

Tech Stack

- Backend: Spring Boot, Java 21
- Messaging: Kafka
- Database: PostgreSQL
- Cache: Redis
- Build Tool: Maven

- Documentation: Swagger / OpenAPI