End-to-End Flow

1. **User Subscription**
    - User clicks “Subscribe” button in the frontend.
    - Backend verifies **user permissions/rights**.
    - If allowed, a **subscription record** is saved in the database.
    - A **scheduled job** is also created with the necessary metadata (user ID, report type, schedule info, etc.).
2. **Job Scheduling**
    - A **background service** periodically scans for scheduled jobs whose `createdAt` or `nextRun` timestamp is due.
    - For each eligible job, it **pushes a message into a job queue** (e.g., SQS, RabbitMQ) describing the work to be done (documents to aggregate, processing details).
3. **Worker Processing**
    - **Workers** are subscribed to the queue.
    - Each worker picks up a message and performs **heavy processing**:
        - Fetching documents/images.
        - Aggregating data.
        - Performing image or PDF processing.
    - Once processing completes, the **PDF report is stored in S3** (or another object store).
4. **Notification & Email**
    - When the PDF is successfully stored, a **new message/event** is triggered for the notification process.
    - A **Notification Worker** picks this up, generates a **safe link** to the PDF, and invokes the **Email Service**.
    - The **user receives the report email**.

Worker Processing with Three Workers

Fetch Worker

- Picks up jobs from the queue that indicate which documents/images need to be processed.
- Fetches the data from the database and S3.
- Pushes the fetched raw data into the next queue for aggregation.

Aggregation Worker

- Picks up fetched data from the fetch queue.
- Aggregates the tabular and textual data into a coherent dataset.
- Prepares a structured payload for PDF/image generation.
- Pushes the aggregated data to the next queue for report generation.

Report/Processing Worker

- Picks up aggregated data from the aggregation queue.
- Performs **image processing**, merges tables/text, and generates the **final PDF report**.
- Stores the PDF in S3.
- Triggers the **notification queue** for safe link generation and email delivery.

Advantages of This Approach

- Each worker can **scale independently** based on workload.
- Failures in one step **don't block others**, allowing retry strategies per worker.
- Easier to **monitor and optimize** specific parts of the pipeline (fetching, aggregation, processing).

Fetch Worker → Aggregation Queue → Aggregation Worker → Processing Queue → Report Worker