Technical Architecture Document
Cultura

Scalable Intelligent Data Processing Pipeline

Author: Blaze Orbit
Date: January 30, 2026

Contents

1 High-level Workflow Diagram

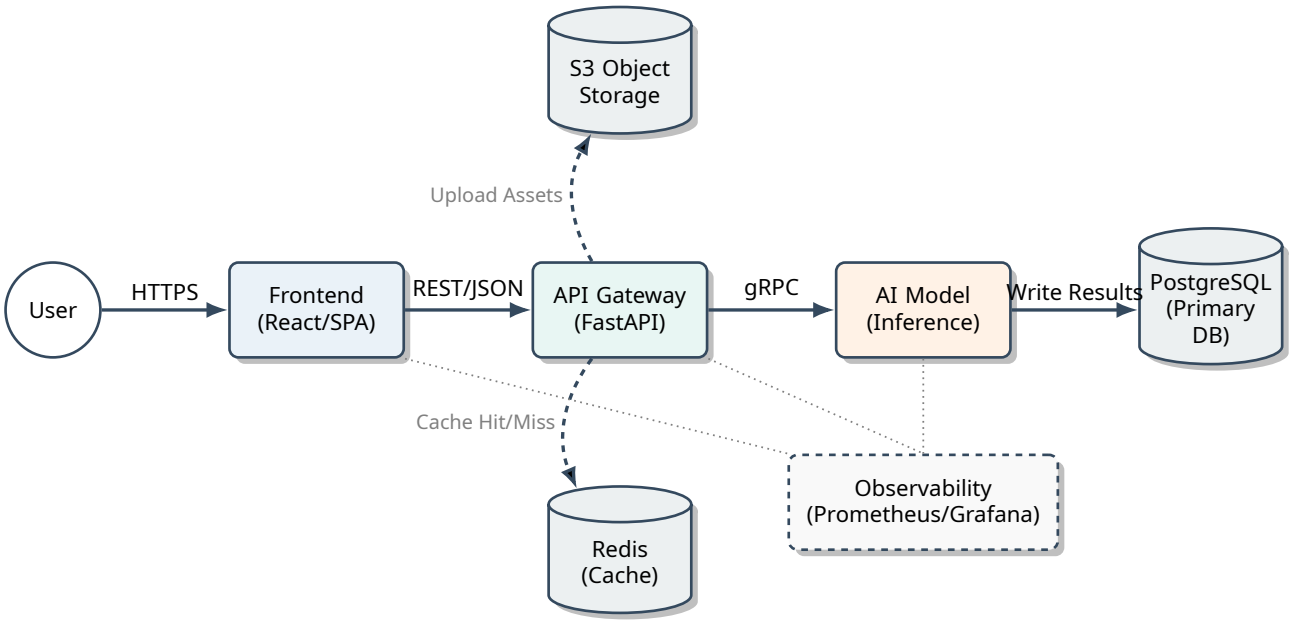

Figure 1: Cultura System Architecture: Scalable, synchronous inference pipeline with asynchronous persistence.

Diagram Description

The architecture follows a standard microservices pattern designed for high availability and low-latency inference. The **User** interacts via a React-based frontend, which communicates securely over HTTPS with the **API Gateway**. The Gateway, built on FastAPI, acts as the central orchestrator; it handles authentication, rate limiting, and request validation.

To minimize latency, the API checks the **Redis Cache** for previously computed results before forwarding tasks. New requests are routed via gRPC to the **AI Model Service**, which performs the core inference. Large artifacts (images/documents) are offloaded to **S3 Object Storage**, while structured metadata and final results are persisted in **PostgreSQL**. The entire pipeline is instrumented with **Prometheus** and **Grafana** for real-time observability.

Mermaid Flowchart (Text Representation)

```
graph LR
    User((User)) -->|HTTPS| FE[Frontend App]
    FE -->|REST| API[API Gateway]
    API -.->|Get/Set| Cache[(Redis)]
    API -.->|Upload| S3[(S3 Storage)]
    API -->|gRPC| AI[AI Inference Service]
    AI -->|Persist| DB[(PostgreSQL)]

    subgraph Observability
        Prom[Prometheus] --> Graf[Grafana]
    end
    API -.-> Prom
```

```
    AI -.-> Prom
```

2 AI/ML Models Configuration

The Cultura platform utilizes a composite model strategy to balance accuracy and latency.

- **Text Analysis: DistilBERT (Hugging Face)**
  - **Purpose:** Intent classification and entity extraction from user queries.
  - **Deployment:** CPU-optimized instances (c5.xlarge), quantified to INT8 using ONNX Runtime.
  - **Tradeoff:** Chosen for sub-50ms inference time over larger BERT variants, sacrificing $< 2\%$ accuracy for $4x$ speedup.
  - **I/O:** Input: Tokenized JSON strings; Output: Logits/Probability distribution.
- **Optical Character Recognition (OCR): TrOCR (Microsoft)**
  - **Purpose:** Extracting text from uploaded PDF/Image documents.
  - **Deployment:** GPU-accelerated (T4 or A10), batch size 8.
  - **Tradeoff:** Transformer-based OCR provides superior accuracy on handwriting compared to Tesseract but requires GPU resources.
  - **I/O:** Input: Base64 image tensor; Output: UTF-8 Text block.
- **Semantic Search: all-MiniLM-L6-v2**
  - **Purpose:** Generating vector embeddings for document retrieval (RAG).
  - **Deployment:** Serverless function (AWS Lambda) or sidecar container.
  - **Tradeoff:** Extremely lightweight (80MB) and fast, suitable for real-time embedding without heavy infrastructure.

3 Data Flow & Processing Logic

1. **Ingress & Validation**
   - User submits data via Frontend. Load Balancer (NGINX) terminates SSL.
   - API Gateway validates JWT token and checks request schema (Pydantic).
   - *Failure Mode:* Invalid Schema. *Mitigation:* Return 422 Unprocessable Entity immediately.
2. **Feature Store Lookup**
   - API generates a hash of the input payload.
   - Queries Redis to see if result exists (TTL: 24 hours).
   - *Failure Mode:* Redis timeout. *Mitigation:* Fallback to fresh inference; log error silently.
3. **Preprocessing**
   - If image: Resize to 384x384px, normalize pixel values (0-1).
   - If text: Tokenize using Hugging Face `AutoTokenizer`, truncate to 512 tokens.
   - *Failure Mode:* Corrupt file upload. *Mitigation:* Check magic bytes before processing.

4. **Inference**
    - Preprocessed tensor sent to Model Service.
    - Model runs forward pass. For heavy loads, request is queued (Celery/RabbitMQ).
    - *Failure Mode:* OOM (Out of Memory). *Mitigation:* Auto-scaling group triggers new nodes; retry logic implemented.
5. **Post-processing & Persistence**
    - Raw logits converted to human-readable JSON labels.
    - Metadata written to PostgreSQL; raw assets archived to S3.
    - *Failure Mode:* DB Write Lock. *Mitigation:* Write to dead-letter queue for later reconciliation.

4 Dockerfile Specification

The following Dockerfile defines the production runtime for the API and Inference service.

```
# Stage 1: Builder
FROM python:3.9-slim as builder

WORKDIR /app
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

# Install system dependencies for build
RUN apt-get update && apt-get install -y --no-install-recommends \
    gcc \
    python3-dev \
    && rm -rf /var/lib/apt/lists/*

# Install python dependencies
COPY requirements.txt .
RUN pip wheel --no-cache-dir --no-deps --wheel-dir /app/wheels -r requirements.
   txt

# Stage 2: Final Runtime
FROM python:3.9-slim

WORKDIR /app

# Create non-root user for security
RUN groupadd -r appuser && useradd -r -g appuser appuser

# Copy wheels from builder and install
COPY --from=builder /app/wheels /wheels
COPY --from=builder /app/requirements.txt .
RUN pip install --no-cache /wheels/*

# Copy application code
COPY . .

# Expose port and switch user
EXPOSE 8000
USER appuser

# Healthcheck
HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost:8000/health || exit 1

# Start command
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000", "--workers",
   "4"]
```

Dockerfile

5 Business & Sustainability Model

Business Outline

- **Value Proposition:** Cultura reduces document processing time by 90% through automated extraction and semantic understanding, replacing manual data entry.
- **Target Customers:** Fintech (Loan processing), Legal (Contract review), and Healthcare (Record digitization).
- **Revenue Streams:**
  - SaaS Tiered Subscription (Starter, Pro, Enterprise).
  - Usage-based billing per 1,000 API calls (overage).
- **Cost Structure:** Primary costs are GPU compute (EC2 p3/g4 instances) and storage (S3).

Sustainability Actions

- **Model Quantization:** Using INT8 quantization reduces energy consumption by approx. 50% compared to FP32 inference.
- **Carbon-Aware Scheduling:** Batch processing jobs are scheduled during off-peak hours when grid carbon intensity is lower.
- **Data Minimization:** Strict retention policies ensure raw data is deleted after 30 days unless archived.

6 Future Scope

Short-term (0–6 Months)

- **Advanced Caching:** Implement semantic caching (Redis VSS) to cache similar queries, not just identical ones.
- **Explainability (XAI):** Integrate SHAP values to explain *why* a document was classified a certain way.
- **Drift Monitoring:** Alerting system for when live data diverges from training data distributions.

Mid-term (6–18 Months)

- **Multi-modal Support:** Allow simultaneous processing of audio and video alongside text/images.
- **Edge Deployment:** Export simplified ONNX models to run directly in the browser (WASM) for privacy.

- **Personalization:** Fine-tuning pipelines allowing Enterprise clients to train adapter layers on their own data.

Long-term (18+ Months)

- **Federated Learning:** Update global models using client-side data without data ever leaving the client device.
- **AutoML Pipelines:** Self-healing models that automatically retrain when performance degrades below threshold.

How to Reproduce:

- Convert Mermaid text to diagram: Use `https://mermaid.live`
- Docker Build: `docker build -t cultura-api .`
- Docker Run: `docker run -p 8000:8000 cultura-api`