OCR Pipeline Architecture

Document ID	LT-OCR-ARCH-001
Version	1.0
Classification	Internal
Date	2026-02-23

1. Executive Summary

The LT OCR Pipeline is a serverless document processing service responsible for payslip fraud detection, structured data extraction, validation, and final trust decision generation. It combines deterministic file-level forensics, rule-based risk scoring, OCR- powered data extraction, and LLM-based semantic interpretation to produce auditable, explainable outcomes.

The pipeline is designed to **always extract data when technically possible**, while separating **risk detection** from **final trust decisions**. Fraud detection is risk-based, not binary — deterministic rules are combined into a single risk score with outcomes of SUCCESS, NEEDS_REVIEW, or FAILED.

This document provides a comprehensive architectural reference covering system design, data flow, processing stages, API contracts, security model, integration patterns, infrastructure, cost analysis, and operational considerations.

Service Name: lt-ocr-pipeline	**Language:** Python

Compute: AWS Lambda (Docker)	**Auth Provider:** AWS IAM / API Gateway

API Protocol: REST / JSON (Asynchronous)	**Messaging:** Amazon SQS
OCR Engine: AWS Textract	**LLM Provider:** Amazon Bedrock (Claude)
Database: PostgreSQL (RDS)	**Storage:** Amazon S3

2. System Overview

2.1 Purpose and Scope

The LT OCR Pipeline processes payslip documents to detect fraud, extract structured payroll data, validate correctness, and produce a final trust decision. It treats documents first as **digital artefacts**, then as **text**, and only later as **business data**.

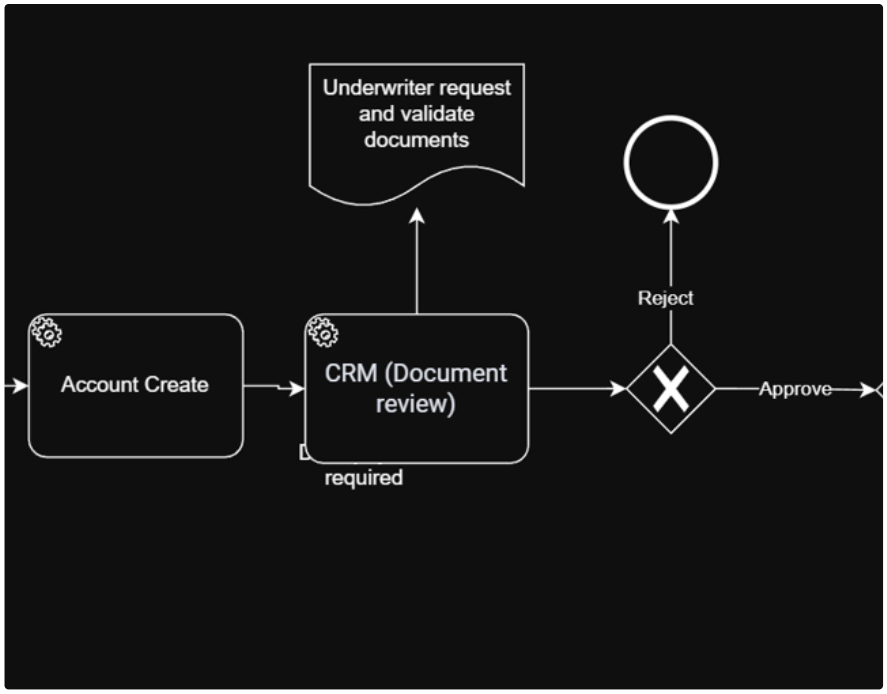
Capability	Description
Deterministic Fraud Detection	File-level digital forensics, layout anomaly detection, encoding analysis, and structural inspection — all rule-based and explainable.
OCR Extraction	AWS Textract-powered text extraction with table and key-value pair recognition for structured payroll data.
Fuzzy Label Mapping	Database-driven label dictionary with fuzzy matching to map OCR-detected labels to canonical internal IDs before LLM interpretation.
LLM-Based Semantic Extraction	Amazon Bedrock (Claude) performs value extraction, context reasoning,

	multilingual handling, and schema completion.
Validation & Trust Evaluation	Deterministic validation rules for extraction usability, authenticity, consistency, and external employer verification.
Risk-Based Scoring	All fraud signals merged into a single clamped score (0–100) with explainable flag breakdown.
Full Auditability	Complete traceability via process_id, OCR evidence references, rule execution logs, and fraud flag details.

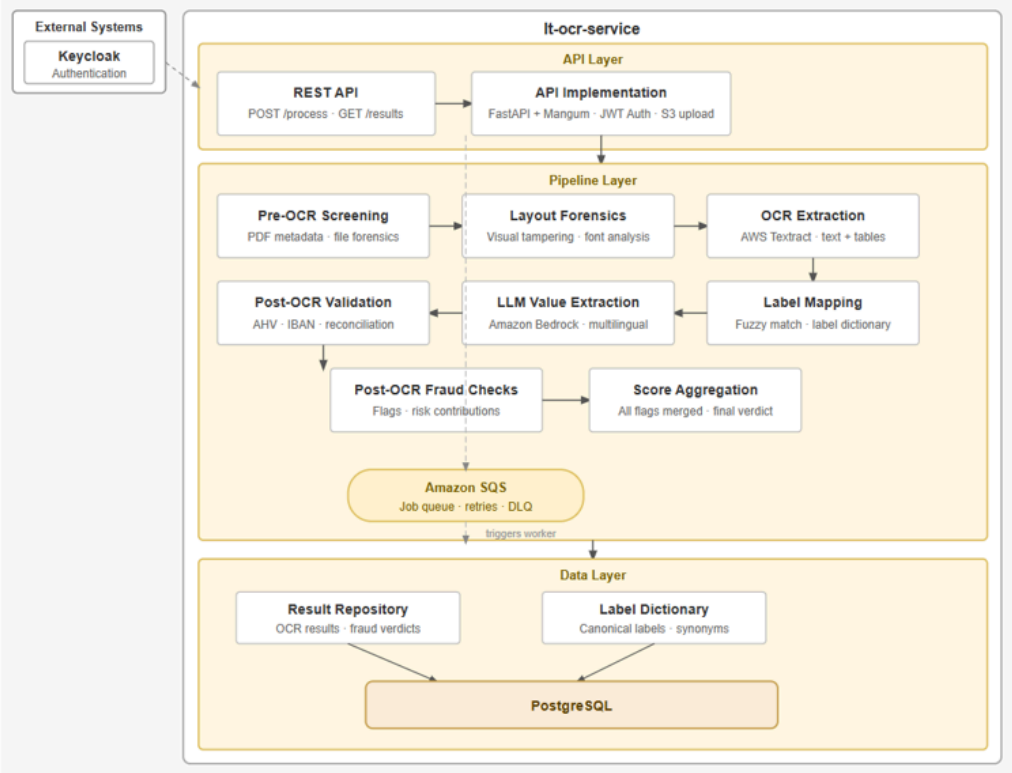
2.2 Architecture Diagram

The diagram below illustrates where the OCR pipeline is triggered within the overall process flow. When the process reaches the **CRM Document Review** stage, the system automatically submits the applicant's payslip to the OCR service for processing.

At that point, the OCR pipeline shown in the architecture diagram below takes over.



It accepts the document, runs it through all fraud detection and extraction stages in the background, and returns a structured result including extracted payroll data and a fraud verdict. This result is then surfaced back to the underwriter within the CRM to support the approval or rejection decision.



The pipeline follows these phases on the architecture

Phase	Description
1 — Ingest	Customer Portal uploads payslip → S3 Bucket storage
2 — Fraud Screening	Fraud pattern recognition via deterministic forensic analysis
3 — OCR Extraction	AWS Textract + Amazon Bedrock (Claude) for structured data extraction
4 — Validation	Business validation rules, identifier checks
5 — Serve	Final consolidated data stored in DataHub

2.3 Data Flow

The processing lifecycle follows an asynchronous pattern:

Step	Action
1	Client uploads payslip via API Gateway → Lambda A
2	Lambda A stores file in S3, inserts PENDING record in PostgreSQL, pushes job to SQS
3	Lambda A returns ACCEPTED response immediately
4	SQS triggers Lambda B (Worker)
5	Lambda B executes full pipeline: Fraud Detection → OCR → Label Mapping → LLM Extraction → Validation → ZEFIX → Score Aggregation
6	Lambda B writes final result to PostgreSQL

3. Module Structure

The service follows a layered modular architecture deployed as Docker containers on AWS Lambda. It is organized into three concerns: entry points that handle communication, core services that execute the pipeline, and domain entities that represent the data.

3.1 Application Entry Points

The pipeline uses two separate Lambda functions sharing a single Docker image (different CMD handler overrides).

Lambda A (lt-ocr-api): FastAPI application adapted for Lambda using Mangum. Handles client-facing operations: file upload, S3 storage, PostgreSQL record creation, SQS job dispatch, and immediate ACCEPTED response. Ensures response within API Gateway's 29-second timeout.

Lambda B (lt-ocr-worker): Background processing Lambda triggered by SQS. Executes the full pipeline with a 15-minute (900-second) timeout to accommodate complete processing duration.

3.2 Core Services

Service	Description
FraudDetectionService	Orchestrates all deterministic fraud detection stages. Analyzes the document as a digital file (PDF metadata, structure, layout, encoding) without interpreting payroll meaning. Produces integrity flags, risk contributions, and hard failures for unreadable files.
OCRService	Manages AWS Textract integration for text extraction. Executes DetectDocumentText for raw OCR and AnalyzeDocument (Tables) for structured key-value pairs and table

	data. Stores all OCR outputs unchanged.
LabelMappingService	Performs deterministic label detection using a database-driven label dictionary. Extracts candidate labels from OCR lines, applies fuzzy matching with configurable thresholds, and handles unmapped labels for continuous dictionary improvement.
LLMExtractionService	Integrates with Amazon Bedrock (Claude) for semantic value extraction and schema completion. Works with pre-mapped label_ids and OCR output. Handles multilingual content (DE/FR/IT/EN). Does NOT perform OCR, fraud risk assignment, or validation.
ValidationService	Applies deterministic validation rules for extraction usability (mandatory fields, parseability, schema compliance) and authenticity checks (salary logic, totals reconciliation, identifier validation). Each rule adds deterministic risk contribution.
FraudAggregationService	Merges all risk contributions into a single clamped score (0– 100). Derives final decision: SUCCESS, NEEDS_REVIEW, or FAILED.

3.3 Domain Entities

Entity	Description
ProcessRecord	Represents a single document processing job. Contains process_id, document reference, status (PENDING, PROCESSING, SUCCESS, NEEDS_REVIEW, FAILED), and timestamps.
FraudResult	Contains the aggregated fraud assessment: risk_score (0–100), fraud_flags list, and validation_summary.
ExtractionResult	Structured payroll data extracted via OCR and LLM. Includes label_id, extracted_value, and evidence references for traceability.
LabelDictionary	Configurable label mapping table with canonical names, language context, and known synonyms for fuzzy matching.

4. Data Model

Database Type: PostgreSQL (RDS)	**Storage:** Amazon S3

4.1 Key Query Patterns

Query	Description
Fetch Pending Jobs	Retrieves PENDING process records for monitoring and retry handling.
Label Dictionary Lookup	Retrieves canonical labels filtered by language and document type for fuzzy matching.
Process Result Retrieval	Fetches complete processing result including fraud score, extraction data, and validation summary for a given process_id.

4.2 Table: process_record

Central table tracking the lifecycle of every document processing job.

Column	Type	Description
id	UUID (PK)	Unique identifier of the process record
process_id	UUID	External process identifier for correlation
created_at	TIMESTAMP	Job creation timestamp
updated_at	TIMESTAMP	Last status update timestamp

status	VARCHAR	PENDING, PROCESSING, SUCCESS, NEEDS_REVIEW, FAILED
document_key	VARCHAR	S3 object key of the original document
document_type	VARCHAR	Document type hint (e.g., payslip)
risk_score	INT	Aggregated fraud risk score (0–100)
decision	VARCHAR	Final outcome: SUCCESS, NEEDS_REVIEW, FAILED

fraud_flags	JSONB	List of triggered fraud flags with descriptions
extraction_data	JSONB	Structured payroll data extracted from the document
validation_summary	JSONB	Breakdown of validation checks and results
error_details	JSONB	Error information if processing failed

4.3 Table: label_dictionary

Stores the canonical label mapping used for fuzzy matching during label detection.

Column	Type	Description
id	UUID (PK)	Primary key
label_id	VARCHAR (UQ)	Internal canonical identifier
canonical_name	VARCHAR	Standard payroll field name
language	VARCHAR	Language context (DE, FR, IT, EN)
synonyms	JSONB	Known label variants for fuzzy matching
document_type	VARCHAR	Applicable document type scope
is_active	BOOLEAN	Toggle for enabling/disabling the label

4.4 Table: unmapped_label_log

Tracks OCR labels that could not be matched to the dictionary, enabling continuous improvement.

Column	Type	Description
id	UUID (PK)	Primary key
process_id	UUID (FK)	Links to process_record
raw_label_text	VARCHAR	Original OCR-detected label text

top_candidates	JSONB	Top N candidate label_ids with similarity scores
ocr_evidence	JSONB	OCR snippet and bounding box reference
resolved	BOOLEAN	Whether the label has been manually reviewed

4.5 Table: external_validation_log

Logs all calls to external services for audit, debugging, and performance monitoring.

Column	Type	Description
id	UUID (PK)	Primary key
process_id	UUID (FK)	Links to process_record
provider_name	VARCHAR	Name of external system
status	VARCHAR	SUCCESS, FAILURE, TIMEOUT
latency_ms	INT	Execution time in milliseconds
raw_response	JSONB	Raw payload received from the provider

5. API Specification

Protocol: REST / JSON	**Security:** AWS IAM / API Gateway
Documentation: OpenAPI 3.0	**Design:** Asynchronous (Accept → Poll)

5.1 Document Processing Endpoint

POST	**/v1/ocr/process**

Submits a payslip document for asynchronous processing. Returns immediately with an ACCEPTED status.

Request Structure

```
{"process_id": "uuid",
"document_key": "string",
"document_type": "payslip", "filename": "string"}
```

Response Structure (Immediate)

```
"process_id": "uuid",
```

```
"status": "ACCEPTED"
```

5.2 Result Retrieval Endpoint

GET	**/v1/ocr/result/{process_id}**

Retrieves the processing result for a submitted document.

Response Structure (Completed)

```
{"process_id": "uuid",
"status": "SUCCESS", "risk_score": 15,
"decision": "SUCCESS", "fraud_flags": [
"PDF_METADATA_SUSPICIOUS_PRODUCER",
"ENCODING_MIXED_APOSTROPHES"
],
"extraction_data": {
"gross_salary": 8500.00,
"net_salary": 6200.00,
"employer_name": "Example AG"
},
"processing_time_ms": 12400}
```

6. Security Model

6.1 Authentication & Authorization

The LT OCR Pipeline is secured through AWS-native mechanisms. API Gateway handles external client authentication, while Lambda functions operate under IAM roles with least-privilege policies.

Protocol: AWS IAM / API Gateway	**Secrets:** AWS Secrets Manager
Network: VPC with NAT Gateway	**Database Access:** VPC Security Groups (TCP 5432)

6.2 IAM Role Permissions

Role	Permission	Assigned To
lt-ocr-api- role	S3 PutObject, SQS SendMessage, RDS Connect, Secrets Manager Read	Lambda A (API)

lt-ocr- worker- role	S3 GetObject/HeadObject, Textract AnalyzeDocument/DetectDocument, Bedrock InvokeModel, RDS Connect, Secrets Manager Read, SQS ReceiveMessage	Lambda B (Worker)

6.3 Network Security

Lambda functions are placed inside the VPC with NAT Gateway for outbound access. RDS security group allows inbound TCP 5432 only from Lambda security groups.

Sensitive credentials (PostgreSQL password, API keys) are stored in AWS Secrets Manager and retrieved at runtime — no plaintext values in environment variables.

7. Evaluation Workflow

The LT OCR Pipeline executes a multi-phase asynchronous processing workflow. The client receives an immediate acknowledgment, and the full pipeline runs in the background via SQS-triggered Lambda.

Step	Phase	Details
1	**Request Reception**	Client sends document processing request to API Gateway → Lambda A. File is uploaded to S3,

		PENDING record created in PostgreSQL, job message pushed to SQS. Client receives ACCEPTED response immediately.
2	**File-Level Forensics**	Document analyzed as a digital artefact. S3 metadata (HeadObject) and raw bytes (GetObject) inspected. Checks: file validity, PDF metadata (Author, Creator, Producer, CreationDate, ModDate), structural indicators, layout artefacts, and encoding anomalies.
3	**OCR Execution**	AWS Textract processes the document. DetectDocumentText for raw OCR. AnalyzeDocument (Tables) for structured key-value pairs and table data. OCR quality gate stops processing on weak results. All outputs stored unchanged.
4	**Label Detection & Fuzzy Mapping**	Candidate labels extracted from OCR lines using bounding box proximity and row

		alignment. Fuzzy matching against label dictionary. Unmatched labels stored with top N candidate suggestions for manual review.
5	**LLM-Based Extraction**	Amazon Bedrock (Claude) receives OCR output and pre-mapped label_ids. Performs semantic value extraction, context reasoning, multilingual handling (DE/FR/IT/EN), and schema completion.
6	**Validation & Trust Evaluation**	Extraction usability checks (mandatory fields, parseability, schema compliance). Authenticity checks (salary logic, reconciliation, identifier validation).
7	**Risk Aggregation & Decision**	All risk contributions merged: file-level + layout + arithmetic + identifier. Score clamped to 0–100. Final decision: SUCCESS (low), NEEDS_REVIEW (medium/high), FAILED (technical failure only). Result persisted to PostgreSQL.

8. Fraud Detection Detail

8.1 Deterministic Checks

Category	What Is Checked	Example Risk Signals
File Validity	Format, readability, bounds	Unsupported format, unreadable PDF, abnormal page count
PDF Metadata	Author, Creator, Producer, CreationDate, ModDate	Word/LibreOffice producer, recent modification for old payroll, missing metadata
Structural Indicators	PDF internal structure	Incremental update chains, unusual object streams, missing font embedding
Layout Artefacts	Visual rendering consistency	Misaligned tables, irregular margins, overlapping text
Encoding Anomalies	Text-level irregularities	Mixed apostrophes (' vs ’), font switches mid-word, mixed decimal separators
Arithmetic Checks	Payroll consistency	Gross vs Net plausibility, totals reconciliation, negative totals

Identifier Validation	UID, AHV, IBAN	Format validation, checksum verification

8.2 Libraries Used

Library	Role
PyMuPDF	PDF parsing, rendering, font inspection
pdfminer	Structural analysis
Pillow	Image-level layout inspection

8.3 Final Decision Logic

Risk Level	Outcome
Low	SUCCESS
Medium / High	NEEDS_REVIEW
Technical failure	FAILED

Note: Fraud or authenticity concerns **do not invalidate extracted data**. Only technical failures stop processing.

9. Design Patterns and Principles

Pattern	Description
Asynchronous Processing	Client receives immediate ACCEPTED response. Full pipeline runs asynchronously via SQS-triggered Lambda. Decouples API responsiveness from processing duration.
Digital-First Analysis	Documents are treated first as digital artefacts (file forensics), then as text (OCR), and only later as business data (payroll extraction). This layered approach maximizes fraud detection coverage.
Risk-Based Scoring	Fraud detection is probabilistic, not binary. All deterministic rules contribute to a single aggregated score (0–100), enabling nuanced decisions rather than hard pass/fail.
Separation of Extraction and Trust	Data extraction always proceeds when technically possible. Risk scoring is independent — fraud concerns flag but do not prevent data output.

Fuzzy Matching with Continuous Learning	Label detection uses fuzzy matching with configurable thresholds. Unmapped labels are logged with candidate suggestions, enabling dictionary expansion and continuous improvement.
LLM as Semantic Layer Only	The LLM handles value extraction and schema completion but does NOT perform OCR, fraud risk assignment, or validation. Clear responsibility boundaries prevent LLM hallucination from affecting trust decisions.
Deterministic Explainability	All fraud scoring is rule-based and explainable. Every risk contribution is traceable to a specific check, enabling audit and manual review.
Auditability by Design	Every processing step is logged: OCR evidence references, label mapping decisions, LLM extraction with evidence linking, validation results, and fraud flags.

10. Infrastructure

Service	Role	Details
AWS API Gateway	Entry point	Exposes lt-ocr-api as public HTTPS endpoint. Routes all

		client requests to Lambda A. Hard timeout of 29 seconds.
AWS Lambda A (lt-ocr-api)	Client-facing API	FastAPI + Mangum. Receives uploads, stores to S3, creates PENDING record, pushes to SQS, returns ACCEPTED.
AWS Lambda B (lt-ocr-worker)	Background worker	SQS-triggered. Runs full pipeline. 15-minute timeout (900s).
Amazon SQS	Job queue	Decouples Lambda A from Lambda B. Provides automatic retries and DLQ support.
Amazon ECR	Container registry	Single Docker image for both Lambdas (different CMD overrides). Required due to dependencies exceeding Lambda's 250MB zip limit.
AWS Textract	OCR engine	DetectDocumentText (raw OCR) + AnalyzeDocument Tables (structured KV pairs and tables).
Amazon Bedrock (Claude)	LLM provider	Semantic value extraction, schema completion, and visual fraud analysis.

Amazon S3	Document storage	Original files stored unchanged. Pipeline never modifies source documents.
PostgreSQL (RDS)	Database	Process records, extraction results, fraud scores, validation summaries.
Amazon CloudWatch	Observability	Logs from both Lambdas. Debugging, metrics, alarms. 30-day log retention.
AWS Secrets Manager	Credential storage	PostgreSQL password, API keys. Retrieved at runtime, no plaintext in env vars.

11. Configuration Checklist

#	Configuration	Service
1	Grant Lambda roles access to S3 bucket	S3 / IAM
2	Enable Anthropic Claude Sonnet + Haiku model access	Bedrock
3	Place lt-ocr-api Lambda inside the VPC	VPC / Lambda

4	Place lt-ocr-worker Lambda inside the VPC	VPC / Lambda
5	Open TCP port 5432 on RDS security group for Lambda	VPC / Security Groups
6	Confirm NAT Gateway exists in VPC (or create one)	VPC
7	Share RDS endpoint, DB name, username, password	RDS
8	Share VPC ID and Subnet IDs	VPC
9	Attach IAM policies to lt-ocr-api role	IAM
10	Attach IAM policies to lt-ocr-worker role	IAM
11	Set CloudWatch log retention to 30 days on both Lambdas	CloudWatch

Regardless of any pending decisions on the overall infrastructure setup, the following two AWS services are very much needed. These are independent of deployment, VPC, or any other architectural choices still under discussion.

AWS Textract — AnalyzeDocument (Tables) Used for structured data extraction from payslip documents. Must be accessible via our IAM role.

Amazon Bedrock — Claude Model Access Used for intelligent data interpretation and schema completion. The following models must be explicitly enabled in the Bedrock console under **Model access**:

Model	Version

Claude Sonnet	`anthropic.claude-3-5-sonnet-20241022-v2:0`

Amazon S3 — Development Bucket We will need access to an S3 bucket for development and testing purposes. To avoid any risk to production documents, we would propsoe access to the dev bucket with read/write access. Please advise on what works best on your side.