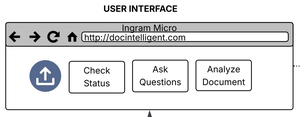
End-to-End Agentic Document Intelligence System



User

GLOBAL ROUTING + SECURITY

DNS Resolution
Geo
Latency
Failover

Route 53 Resolver

Authentication
Authorization
WAF

Amazon API Gateway

ROUTING LOGIC

task_router_lambda.
Read task_type from payload
Route accordingly:
Inject -->POST/injectEKS
query --> POST/queryEKS
analyze --> POST/analyze@EKS
status --> POST/status@EKS

Lambda Function

Optional:
Push to SQS Queue
for asyn orchestration

Queue

Elastic Load Balancing

ECR

Kubernetes Cluster

EKS

/analyze
/inject
/query
/status

crewai

AnalyzeDocumentCrew:
1. Load doc from s3/blob
2. Classify content
3. Extract relationship
4. Extract Entities

InjectDocumentCrew:
1. Load doc from s3/blob
2. Chunk + Embed

QueryAgentCrew:
1. Query vector Store
2. Returns Answer + sources

StatusCheckCrew:
1. Using session id
2. Check the Database Job status table
3. Return Status to user

Model

OpenAI: gpt-4o-mini

Embedding Model: text-embedding-3-small

Parameter Store

Service Layer

chunking
Embedding
1.pdf
2.docx
3.csv
4.txt
OpenAI

Data Layer

Postgres
1. Metadata storage
2. Job tracking

S3/Blob
Temporary document storage.

Weavite: vector database
1. Stores embedded chunk
2. Tenant scoped
3. Knowledge Base

Observability + Evaluation

LangFuse
1. Logs LLM Calls
2. Tracks agent agent steps

RAGAS
1. Evaluate faithfulness
2. Context Relevance

CICD PIPELINE

Jenkins Pipeline

Sign Image
Evaluate RAGAS
VERACODE code Scan

Push Code

Developer

This architecture supports secure, multi-tenant agentic document processing. We use DNS-based routing for latency/failover, task-type based routing logic via Lambda, and isolate services in EKS-backed crews. Vectorized content flows to Weaviate, evaluation flows through Langfuse + RAGAS, and state is consistently tracked in Postgres + S3. All credentials and vectorizer configs are securely fetched from Parameter Store, enabling tenant isolation and runtime introspection.

Architect: Gabriel Ohaike
Date: July, 14 2025