Extraction Service – Architecture

Creator **Michael Clark** Created **Aug 24, 2025, 22:17** Last updated **Aug 24, 2025, 23:08**

Summary

Using **Serverless.js** and the **AWS ecosystem**, we will build and deploy lightweight APIs through **API Gateway** and **Lambda endpoints** to handle job creation, status checks, results retrieval, and tenant management. This architecture allows us to scale dynamically without managing servers, enforce **security and multi-tenant isolation** via Cognito/API keys, and provide **observability** with CloudWatch, X-Ray, and CloudTrail. Document processing is orchestrated through **Step Functions** with **SQS fan-out** to Lambda workers that perform extraction (rule-based and LLM-assisted). Results are written to **S3 in partitioned JSONL shards** with manifests and metrics, while **DynamoDB** tracks job state, quotas, and tenants. Additional services like **Secrets Manager, KMS, and IAM** ensure compliance and secure operations, while **Budgets and cost pre-checks** keep usage under control. This design delivers a highly scalable, cost-efficient, and auditable foundation for unstructured-to-structured document extraction.

Core components

- **Ingress & Control Plane**
  - **Amazon API Gateway (REST/HTTP)** – public API for Job/Results/Tenant endpoints.
  - **AWS Lambda** – request handlers (jobs.create, jobs.status, results.list, tenants.create, auth.introspect).
  - **Custom API key/RBAC** via **Lambda authorizer**.
  - **AWS WAF** on API Gateway.
- **Orchestration & Workers**
  - **AWS Step Functions** – job orchestration (pre checks → fan out → aggregate → finalize).
  - **Amazon SQS** – fan out work queue for document tasks (standard queue + DLQ).
  - **Lambda Workers** – extractors (rule based & LLM assisted), chunk writers, manifest builder.
- **Connectors (Sources & Sinks)**
  - **S3 Source** (S3 events optional) & **local FS** (upload to S3, trigger lambda on complete).
  - **SharePoint** – via **Lambda connector** (Microsoft Graph) + **Secrets Manager** creds.
  - **S3 Output** – partitioned **JSONL.GZ** shards + **_manifest.json** + **_metrics.json** + **_checksums/** + **_dlq/**.
  - Optional analytics sink: **AWS Glue Catalog + Athena** over S3 outputs.
- **State, Metadata & Indexes**
  - **Amazon DynamoDB/MongoAtlas** – tenants, jobs, parts, manifests, quotas; PK/SK design per tenant.
  - **Amazon OpenSearch Serverless** (optional) – fast lookups for job/part search & logs (if needed).
  - **AWS EventBridge** – job lifecycle events (created, running, finished, failed) for integrations.
- **Security & Secrets**
  - **AWS KMS** – envelope encryption (S3, DynamoDB, SQS, CloudWatch Logs).
  - **AWS Secrets Manager** – LLM API keys, SharePoint OAuth, external creds.
  - **AWS IAM** – least privilege roles; scoped per Lambda/StateMachine; per tenant isolation via attributes.
- **Observability & Audit**
  - **AWS X Ray** – traces across API → Lambda → Step Functions → SQS → Lambda.

- **Amazon CloudWatch** – metrics (p50/p95/p99, error/retry/throughput), alarms, dashboards, logs.
- **AWS CloudTrail** – API audit; **immutable S3 WORM** bucket for audit exports (if required).

- **Cost & Quotas**
  - **Pre check Lambda** – estimate pages/tokens/$; compare with **DynamoDB** tenant quotas/budgets.
  - **AWS Budgets / Cost Explorer** – per tag/tenant cost visibility; alarms → EventBridge → Slack/Email.
- **LLM Providers**
  - **Amazon Bedrock** (primary; private VPC endpoints where applicable).
  - **OpenAI/Anthropic** via **VPC e NAT + Secrets Manager** (toggle by job config); exponential backoff + circuit breaker.

Developer Experience

Using **Serverless.js** tools, engineers are able to **locally stub out the entire AWS architecture** — including API Gateway, Lambda, DynamoDB, and S3 — without needing to deploy to the cloud for every iteration. This enables:

- **Local development & testing**: Run `serverless offline` to emulate API Gateway + Lambda endpoints.
- **Service mocks**: Use **LocalStack** and **DynamoDB Local** for S3, DynamoDB, SQS, and Step Functions simulation.
- **Hot reload & debugging**: Rapid feedback loop for editing Lambda handlers, testing configs, and verifying extraction logic.
- **Automated deploys**: `serverless deploy` packages functions, sets IAM roles, configures environment variables, and provisions resources consistently across dev, staging, and prod.
- **Infrastructure as Code**: The entire stack is declared in YAML/Serverless configs, making it reproducible and version-controlled.
- **Traceability & observability**: Local traces and logs mirror CloudWatch/X-Ray structure for seamless debugging pre- and post-deployment.

This developer experience ensures that engineers can **prototype, test, and validate jobs locally**, shorten iteration cycles, and deploy with confidence to production-ready AWS environments.