Implementation Plan: Extraction Service

a. Milestones (MVP → Beta → GA)

MVP (1 Week):

- **Scope:** Core extraction pipeline for S3-based PDFs, single tenant, basic fields. API for job submission, S3 output. Basic observability through CloudWatch logs.
- **Cutlines:** No SharePoint connector, no PII scanning, no multi-tenancy. Limited to 100 docs for testing.
- **Deliverables:** Deployable Lambda functions (Job Manager, Doc Fetcher, Output Writer), SQS queue, SageMaker endpoint with open-source LLM, DynamoDB for job tracking.
- **Validation:** Process 10 sample docs with >90% accuracy for structured fields.

Beta (~1.5 Weeks):

- **Scope:** Add SharePoint connector, multi-tenancy via IAM roles, PII scanning with Macie. Scale to 1k docs. Basic metrics endpoint.
- **Cutlines:** No advanced retry policies, no custom dashboards.
- **Deliverables:** Step Functions for orchestration, SharePoint crawler via AWS SDK, Macie integration, tenant isolation in DynamoDB.
- **Validation:** Process 1k docs, verify tenant separation, PII flagging.

GA (2 Weeks):

- **Scope:** Full observability with CloudWatch dashboards and X-Ray traces, cost controls, accuracy benchmarks of >95% for structured fields, >85% for summaries. Scale to 100k docs.
- **Deliverables:** Auto-scaling SageMaker instances, CloudWatch Budgets, detailed runbook.
- **Validation:** Pilot with customer, process 100k docs within 24 hours, under $1k cost.

b. Risk, Impact, Mitigation

1. **Risk:** LLM accuracy below target of >95% for structured fields.
    - **Impact:** Customer distrust; rework needed.
    - **Mitigation:** Research spike to benchmark open-source LLMs on sample docs. Fine-tune model in SageMaker. Test with ground truth dataset.
2. **Risk:** SharePoint connector fails due to client-specific auth configs.
    - **Impact:** Blocked Beta phase.

    - **Mitigation:** Early validation with client’s SharePoint setup. Fallback to S3-only for MVP.

3. **Risk:** Cost overrun for 100k docs (exceeds $1k).
    - **Impact:** Budget violation; PoC rejection.
    - **Mitigation:** Use spot instances for SageMaker, set CloudWatch Budget alerts, optimize batch sizes.
4. **Risk:** GovCloud compliance failure.
    - **Impact:** PoC rejected by compliance team.
    - **Mitigation:** Audit all services against GovCloud docs. Use only managed/open-source components.
5. **Risk:** Scaling bottlenecks at 100k docs.
    - **Impact:** Missed SLA of <24h for 100k docs.
    - **Mitigation:** Pre-warm SageMaker endpoints, use SQS for backpressure, test with 10k docs in Beta.

c. Effort Estimate & Roles

Total Effort: ~120 hours

- **MVP (60 hours total, 1 week team time):**
    - Developer (40h): Build Lambda functions, SQS, DynamoDB schema.
    - ML Engineer (15h): Deploy and test LLM in SageMaker.
    - ML Engineer (15h): LLM spike
    - DevOps (15h): Set up GovCloud environment, IAM roles.
- **Beta (50 hours, 1.3 weeks team time):**
    - Developer(10h): Sharepoint AWS SDK compatibility spike
    - Developer (20h): Add SharePoint connector, Step Functions.
    - ML Engineer (25h): Integrate Macie, tune LLM.
    - DevOps (15h): Multi-tenancy, basic metrics endpoint.
    - QA (25h): Start verifying output of entire system and provide feedback
- **GA (80 hours, 2 weeks team time):**
    - Developer (30h): Optimize pipeline, add retry logic, integrate performance dashboards.
    - ML Engineer (25h): Accuracy benchmarks, model tuning.
    - DevOps (40h): Observability, alerts, runbook.
    - QA (40h): Validate system thresholds, performance, alarms, and quality of output

Research Spikes:

- LLM selection (15h): Benchmark AWS Gov Approved LLM
- SharePoint auth (10h): Validate AWS SDK compatibility with client setup.

d. Delivery Plan

Goal: Minimize client deployment friction in AWS GovCloud.

- **Packaging:** Provide a Git repo with:
    1. CloudFormation templates for infrastructure.
    2. Python code for Lambda functions and LLM inference.
    3. Docs: README.md, DECISIONS.md, runbook.
    4. Sample config YAML and test scripts for 10 docs.
- **Deployment Steps:**
    1. Client clones repo, runs aws cloudformation deploy with provided templates.
    2. Configure IAM roles and S3/SharePoint credentials (guided by README).
    3. Upload sample docs to S3, submit test job via API.
    4. Validate outputs in S3; monitor via CloudWatch.
- **Support:** Include a TROUBLESHOOTING.md with common issues. Offer a 1-hour handover call to walk through setup.
- **Rationale:** CloudFormation ensures one-click deployment. Docs reduce the learning curve. GovCloud focus avoids external dependencies.