Runbook: RAG Indexing Pipeline (PDF → Chunks → Vectors)

Last updated: 2026-01-23

Purpose

This runbook standardizes the process to ingest PDF runbooks from S3, extract text, chunk content, create embeddings, and upsert vectors into your vector store (FAISS/Chroma/Pinecone/Azure AI Search).

Inputs / Outputs

Item	Value
Input documents	PDFs under S3_PREFIX/runbooks/
Chunk size	400–800 tokens (recommend 600)
Overlap	~100 tokens
Output	Vector store collection + metadata per chunk

Pre-flight Checks

- Confirm S3_BUCKET and S3_PREFIX in Lambda/env or indexing job config.
- Confirm IAM permissions: s3:GetObject, s3:ListBucket for bucket and prefix.
- Confirm embedding model deployment name (Azure OpenAI) and endpoint/key/MI configuration.
- Confirm vector store connectivity and collection name (dev/prod separated).

Algorithm (High-level)

- List PDFs from S3 under runbooks/ prefix.
- For each PDF: extract text, normalize whitespace, split into sections when possible.
- Chunk into token-aware segments; attach metadata (source, section, env).
- Compute embedding for each chunk; upsert into vector store.
- Write an indexing manifest back to S3 (index-manifest.json) with doc hashes and timestamps.

Failure Modes & Fixes

- Empty text extraction: verify PDFs are text-based (not scanned images); if scanned, run OCR offline first.
- Duplicate vectors: use stable chunk IDs (docKey + page + chunkIndex) to ensure idempotent upserts.
- Embedding 429: throttle concurrency; add retry with exponential backoff; batch requests.

Tip: Keep dev/prod indexes separate by using different collection names or vector namespaces.