Sandstone Assessment

api/documents:

1) PutObject(storageKey) ──► [S3]
2) Create/Upsert ──► [DB: Document(version=1)]
3) Enqueue ──► [Queue: extract]
   { documentId, version, storageKey, mimeType }
4) 201 Created (returns docId)

[Extract Worker] ◄─ poll ── [Queue: extract]
[Extract Worker] steps:

- Idempotency: if job.version < DB.version → drop (stale)
- GetObject(storageKey) ◄─ [S3]
- Detect & extract text
  - text/* → UTF-8
  - application/pdf → pdf-parse (OCR optional)
  - .docx → mammoth
- Upsert ──► [DB: DocumentText]
- Upsert ──► [DB: Processing(stage='extracted')]
- Enqueue ──► [Queue: index] { documentId, version }
- Publish ──► [Pub/Sub → WebSocket/SSE] event: doc.extracted

Failures → retry with exponential backoff; after N attempts → DLQ + Processing(stage='failed', error)

Event Examples:

Event: doc.uploaded
  { documentId, version, storageKey, mimeType }
Event: doc.extracted
  { documentId, version, textBytes, hasOcr?: boolean }
Event: doc.indexed
  { documentId, version, engine: 'meilisearch' | 'opensearch' }

api/documents/[id?]/search

1) AuthN/Z (tenant, role) → derive filters (tenantId, current versions)
2) Query [Search Index] (Meilisearch/OpenSearch)
   - filter: tenantId = X AND version = currentVersion
   - attributes: documentId, name, chunkStart, text (formatted/highlighted)
   - options: showMatchesPosition, highlight tags, crop length

3) Transform hits → one result per occurrence:
   { docId, name, start: chunkStart + localStart, end: chunkStart + localStart + length, snippetHtml }
4) 200 OK (results)

Optional async bits (no client blocking):
• Publish 'search.performed' to analytics pipeline
• Cache by (tenantId, query) in Redis (short TTL)
• If index is still building for some documents, include { inProgressDocIds: [...] } or stream via SSE/WebSocket

api/documents/[id]:

1) AuthN/Z (tenant/role), load Document & current DocumentText
2) Validate ranges (0 ≤ start ≤ end ≤ len); apply right-to-left
3) Update [DB: DocumentText]; bump Document.version += 1
4) Enqueue ──► [Queue: index] { documentId, version } (reindex latest)
5) Publish ──► [Pub/Sub → WebSocket/SSE] event: doc.updated
6) 200 OK { id, version }

[Index Worker] ◄─ poll ── [Queue: index]
[Index Worker] steps:
• Idempotency: if job.version < DB.version → drop (stale)
• Read [DB: DocumentText]
• Chunk text (e.g., 2 KB) with absolute offsets; upsert into [Search Index]
• Optionally delete older index version for this document
• Upsert ──► [DB: Processing(stage='indexed')]
• Publish ──► [Pub/Sub → WebSocket/SSE] event: doc.indexed

Failures → retry with exponential backoff; after N attempts → DLQ + Processing(stage='failed', error)

Architecture & Infra:

- **Compute**
    - **API**: Next.js (Node runtime) for thin endpoints (auth, metadata, enqueue).
    - **Workers**: small Node services or serverless functions:
        - **extract** (PDF/DOCX → text, optional OCR) – CPU/RAM heavier
        - **index** (push text to FTS/engine) – IO-heavy, cheap to scale
- **Queues**
    - SQS (+ DLQ) or Upstash Redis.

    - Event schema includes `{ documentId, version, storageKey, mimeType }`; **idempotency** on `(documentId, version)`.
- **Storage**
    - **Object Store**: S3.
    - **DB**: Postgres via Prisma.
    - **Index**: **Meilisearch** or **OpenSearch** (managed or self-hosted).
- **Caching**: Redis for hot search results by query hash (short TTL) and presigned URL metadata.
- **Realtime push**: **WebSockets** or **Server-Sent Events (SSE)** to stream async job status/progress and patch completion to the UI. Authenticate connections with Cognito-issued JWTs; authorize per tenant. Use pub/sub (Redis/SNS) to fan-out events from workers to the socket/SSE broadcaster; provide a polling fallback.

CI/CD & Deployment:

- **Monorepo** (pnpm workspaces + Turborepo) containing API, workers, shared libs (text extraction/indexing), and IaC.
    - For a **small team**, this improves deployment efficiency: single PRs for cross-cutting changes, shared lint/TS configs, unified versioning, **affected-only** builds, and atomic rollouts.
- Pipelines: lint → unit tests → build → Prisma migrations → package images → deploy.
    - **Environments**: dev → staging → prod with promotion.
    - **Rollouts**: blue/green or canary for API; workers are **drain-aware** (finish jobs before replace).
    - **Preview deploys** per PR (ephemeral env) for UI/API.
- **IaC**: Terraform (S3/SQS/IAM/Redis/Search) committed in the same monorepo; plan/apply via pipeline gates.

Security & Compliance:

- **AuthN**: OIDC (e.g., **Amazon Cognito**, Auth0/WorkOS); **AuthZ**: roles per tenant (viewer/editor/admin).
- **Data protection**: TLS, S3 SSE-KMS, DB at-rest encryption, secrets via cloud manager.
- **Audit**: append-only logs for uploads, edits (ranges), reads/searches (coarse), admin actions.
- **Compliance**:
    - **GDPR**

        - data mapping/RoPA
        - lawful basis + DPA/SCCs for processors/transfers
        - data-subject rights (access/export/delete across DB, index & object store; purge windows for backups)
        - retention & purpose limitation
        - 72-hour breach notice
        - privacy-by-design/DPIA
    - **SOC 2 Type II**
        - SSO/MFA & least-privilege IAM
        - change management with approvals
        - secure SDLC & vulnerability mgmt (SCA/SAST/DAST)
        - immutable audit logs
        - backups with tested restores (BC/DR)
        - incident response runbooks/SLAs
        - vendor risk management
        - evidence captured via tickets/monitoring

Scalability & Resilience:

- **Horizontal scale** by queue depth/age (HPA or serverless concurrency).
- **Backpressure** via queues; set max concurrent extract jobs.
- **Retries & DLQ** with exponential backoff; poison-pill protection (timeouts, memory caps).
- **Chunked indexing** (2–4KB) for efficient snippets and partial updates; store chunk offsets.
- **Multi-AZ** default; **multi-region** later with read-replica search and async index replication.

Monitoring & Observability:

- **Metrics**: queue depth & oldest age, job success rate, extract/ocr/index latency (p50/p95), API latency/4xx/5xx, index size.
- **Logs**: structured JSON with `jobId, documentId, version` (correlatable).
- **Tracing**: OpenTelemetry from upload → extract → index → search (propagate trace IDs via job payload).
- **Alerts/SLOs**: DLQ growth, processing latency SLO breach, high 5xx, search error rate, storage nearing quota.

Operations & Cost:

- **Cost controls**: file size & type limits, extraction timeouts, OCR behind feature flag, S3 lifecycle to cheaper tiers, auto-pause workers off-peak.
- **Serverless-first**: lean on serverless runtimes (Vercel Functions/background/cron, AWS Lambda + SQS, Upstash Q → workers) to **pay-per-use** and **scale-to-zero**, minimizing idle cost with a low client count; use short-lived containers only where heavy extract/OCR needs more CPU/RAM.
- **Right-sizing**: small API instances, burstable worker pools; step-function scale based on queue metrics.
- **Backups/DR**: daily DB snapshots, index snapshots (or reindex from text)