Here’s what each service is for, in plain English:

* **nginx** — Edge web server & TLS terminator  
  + Serves your **Next.js** frontend (mounts .next build and public/).
  + Terminates HTTPS using certs from ./tls and reverse-proxies API/WebSocket traffic to backend (e.g., gateway, jobs).
  + Ports: 80, 443.
* **postgres** — Primary relational database  
  + Stores core app data (users, jobs, payments, listings, messages, etc.).
  + Initializes schema + seed data via mounted SQL files in ../backend/sql.
  + Port: 5432. Data persisted in pgdata volume.
* **redis** — Cache, queues, and rate-limits  
  + Fast in-memory store for sessions, job queues, rate limiting, and short-lived state.
  + Port: 6379.
* **minio** — S3-compatible object storage  
  + Holds uploaded documents (IDs, contracts, searches) and media.
  + The console UI is on 9001; S3 API on 9000.
  + Accessed by gateway/jobs using the provided keys. Data in miniodata volume.
* **clamav** — Antivirus scanning daemon  
  + Scans uploaded files before they’re accepted (malware protection for KYC/docs).
  + Exposes ClamD service on 3310; jobs connects to it.
* **otel** (OpenTelemetry Collector) — Telemetry pipeline  
  + Receives traces/metrics/logs from services and forwards to backends (e.g., Prometheus/Loki/Grafana).
  + Configured by ./otel/otel-config.yaml. Port: 4317 (OTLP gRPC).
* **loki** — Log aggregation store  
  + Receives and indexes application logs (via Promtail/OTel, depending on your pipeline).
  + Queryable from Grafana. Port: 3100.
* **prom** (Prometheus) — Metrics scraper & TSDB  
  + Scrapes metrics endpoints exposed by your services and the collector.
  + Config from ./prometheus/prom.yml. Port: 9090.
* **grafana** — Observability dashboards  
  + Visualizes metrics (Prometheus), logs (Loki), and traces (via OTel exporters).
  + Default admin password set via env; change it in prod. Port: 3000.
* **gateway** — API gateway / BFF (backend-for-frontend)  
  + Public entrypoint for API calls from the frontend (auth, listings, bookings, escrow ops, etc.).
  + Talks to Postgres, Redis, Minio; routes/coordinates with other internal services.
  + Handles JWT verification and PSP selection (PSP\_PROVIDER). Port: 8080.
* **identity** — Authentication & user management service  
  + Issues/validates JWTs, manages accounts, passwords, roles (client vs. conveyancer), OAuth if added.
  + Uses Postgres; shares JWT\_SECRET with gateway.
* **jobs** — Asynchronous workers & real-time chat  
  + Processes background tasks: file virus scans (via ClamAV), MinIO uploads, email/push, scheduled cleanups.
  + Hosts the **WebSocket** server for chat/notifications. Port: 9002.
* **payments** — PSP integration & escrow logic  
  + Abstracts payment service providers (sandbox/prod), creates payment intents, holds/release escrow, webhooks.
  + Uses Postgres; secrets configured via PSP\_\* env vars.

### How it all fits together (quick flow)

1. **Browser → nginx** (HTTPS) serves Next.js and proxies /api/WS to **gateway** / **jobs**.
2. **Gateway** authenticates via **identity**, reads/writes to **postgres**, caches in **redis**, stores files in **minio**.
3. **Jobs** scans uploads with **clamav**, emits notifications over WebSockets.
4. **Payments** talks to external PSPs for escrow and settlement events.
5. **otel/prom/loki/grafana** provide tracing, metrics, and logs for all services.

### Practical notes (prod-hardening quick wins)

* Add healthcheck: and restart: unless-stopped to every service.
* Replace dev secrets (JWT\_SECRET, MINIO\_\*, PSP\_SECRET) with strong values (env or secrets manager).
* Lock **nginx** to only expose required upstreams and enable HTTP/2 + security headers.
* Configure Prometheus scrape jobs for gateway, identity, jobs, payments if they expose /metrics.
* Consider a **Promtail** sidecar or OTel logs exporter so app logs reach **Loki**.
* Backup policies for pgdata and miniodata.
* If Next.js is built at runtime, ensure the .next mount contains a production build (next build).

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 Marketplace platform architecture.pdf

file://file\_0000000098186206b08e7fdcae5bc8c9

22