RAG Production Stack — Comprehensive Technical Overview

Production-Oriented Retrieval-Augmented Generation Infrastructure

A client-readable and engineer-ready reference for authenticated RAG services, network isolation, observability, security and healthcare scanning, resource governance, backup recovery, verification boundaries, and developer handoff.

LightRAG | Authenticated Access | Metrics, Logs, and Traces | Profile-Based Scanning

Open-Source Public Technical Edition | Updated: *Sep 24, 2026*

Developed by Zahidul Islam

Current Evidence and Scope Note — Current Public Release

This edition is reconciled against public repository commit 5d465c0. It preserves the existing technical sections, corrects unsupported operational claims, and adds open-source, scaling, reliability, security and legal-readiness guidance. Configuration intent is distinguished from deployed behavior and future targets.

Verified repository snapshot

• 23 Docker Compose services: 9 core and 14 profile-activated.

• Three declared networks: frontend, backend, and monitoring.

• Nine core services include source-defined health checks.

• Prometheus uses a 15-second scrape interval with five alert rules.

• Loki retention is configured for 744 hours, equivalent to 31 days.

• Restic scripts define encrypted backup and isolated restore-verification workflows.

Safety and verification boundary

This publication did not start, stop, restart or deploy containers or invoke a model provider. A minimal source change corrected Nginx health-response header inheritance. Historical, staged and public-tree secret scans found no matches; private runtime material was excluded from publication. Security tooling provides assessment capability, not certification or proof that a live deployment is secure or compliant.

1. EXECUTIVE SUMMARY

The RAG Production Stack is an MIT-licensed, production-oriented Retrieval-Augmented Generation (RAG) reference infrastructure. It assembles LightRAG v1.4.6 and RAG-Anything v1.2.10 with deployment, identity, observability and assessment tooling. The custom Python adapter currently uses Gemini-specific callbacks; model settings and upstream provider compatibility must be validated for the chosen deployment.

The source defines Authelia authentication with a two-factor policy, configuration for metrics, logs, traces and alerts, and tools that support security and compliance assessment. Docker Compose provides a single-host baseline. Core memory limits total 4,352 MiB (4.25 GiB), before host overhead and optional services; an 8 GiB host is a planning baseline, not a measured throughput guarantee.

Key Metrics:

- 23 containerized services (9 core + 14 on-demand via profiles)

- 3 declared Docker networks (frontend bridge, backend internal, monitoring internal); effective isolation and required connectivity need deployment tests

- 7 configured Prometheus scrape jobs at 15-second intervals; target health and exporter compatibility still need validation
- 5 alert rules (container down, high memory, high CPU, target missing, Loki errors)
- 3 Grafana datasource definitions (Prometheus, Loki, Jaeger); runtime provisioning location and successful loading need verification
- Requested 31-day log retention via Loki; effective deletion and disk capacity need runtime verification
- Authelia two-factor policy for protected routes; portal, health and direct loopback access have separate handling
- TLS 1.2/1.3 and five configured HTTPS security headers; CSP is a future deployment policy, not currently configured
- Rate limiting: 10 req/s per IP with burst 20
- Assessment tooling includes Checkov, Trivy, SonarQube, OWASP ZAP, HAPI FHIR, Inferno, Prowler and the custom healthcare scanner
- Restic encrypted backups (7 daily, 4 weekly, 6 monthly retention)
- Public release commit 5d465c0; hosted security CI passed Gitleaks, Semgrep and Bandit

2. ARCHITECTURE OVERVIEW

2.1 High-Level Architecture:

The stack declares three Docker network zones. Network membership, host-published ports, egress and instrumentation determine actual isolation and reachability:

Frontend Network (bridge): Exposes Nginx reverse proxy to the host on ports 80 and 443, handling all inbound traffic with TLS termination, rate limiting, and forward authentication via Authelia.

Backend Network (internal): Hosts LightRAG and RAG-Anything alongside the ingress/identity components. LightRAG also publishes a host-loopback port, which bypasses Nginx. Model-provider egress requires a deliberate design because the RAG services attach only to the internal backend network.

Monitoring Network (internal): Contains Prometheus, Grafana, Loki, Jaeger, Alertmanager and SonarQube, with Nginx attached. LightRAG currently has no shared network with Jaeger or Prometheus, so intended telemetry connectivity must be repaired and tested.

2.2 Request Flow:

1. Client sends HTTPS request to Nginx (ports 80/443)
2. Nginx performs TLS termination with TLS 1.2/1.3 strict ciphers
3. Forward-auth subrequest is sent to Authelia for identity verification
4. Authelia evaluates the configured two-factor access policy; validate the selected release's MFA methods and proxy integration
5. Authenticated requests are proxied to the appropriate backend service
6. HTTPS responses include the configured inherited headers, such as HSTS and X-Frame-Options; CSP is not currently defined

3. CORE COMPONENTS

3.1 Service Registry:

Core Services (9, enabled by default):

LightRAG (ghcr.io/hkuds/lightrag:1.4.6): Knowledge graph RAG engine. Port 9621. Memory 1024MB, CPU 2.0

Nginx (nginx:1.28-alpine): Reverse proxy, TLS termination, rate limiting. Ports 80/443. Memory 128MB, CPU 0.5

Authelia (authelia/authelia:4.39.15): 2FA/TOTP/WebAuthn authentication. Port 9091. Memory 128MB, CPU 0.5

Prometheus (prom/prometheus:v2.53.5): Metrics collection (7 targets, 15s scrape). Port 9090. Memory 384MB, CPU 1.0

Grafana (grafana/grafana:10.2.3): Dashboards and datasource provisioning definitions; verify the runtime search path. Port 3000. Memory 384MiB, CPU 1.0

Loki (grafana/loki:3.6.0): Log aggregation (31-day retention, non-blocking driver). Port 3100. Memory 384MB, CPU 1.0

Jaeger (jaegertracing/all-in-one:1.53): Distributed tracing via OpenTelemetry. Port 16686. Memory 256MB, CPU 1.0

AlertManager (prom/alertmanager:v0.27.0): Alert routing configuration; notification transport is not yet configured. Port 9093. Memory 128MiB, CPU 0.5

SonarQube (sonarqube:10.8-community): Static code analysis. Port 9000. Memory 1536MB, CPU 2.0

On-Demand Services (14, profile-activated):

Profile "rag" (1 service):

RAG-Anything (custom Dockerfile, Python 3.11): Multimodal document processor. Memory 1024MB

Profile "scan" (12 services):

Checkov (bridgecrew/checkov:latest): Terraform/IaC assessment; framework coverage depends on the selected rules and target

Trivy (aquasec/trivy:0.69.3): Container CVE vulnerability scanning

Prowler (prowlercloud/prowler:latest): AWS compliance scanning

Prowler-GCP (prowlercloud/prowler:latest): GCP compliance scanning

Prowler-Azure (prowlercloud/prowler:latest): Azure compliance scanning

OWASP ZAP (ghcr.io/zaproxy/zaproxy:stable): Dynamic application security testing. Memory 512MB

HAPI FHIR (hapiproject/hapi:latest): FHIR R4 validator. Memory 1024MB

Inferno (custom build from an external test-kit checkout): ONC g(10) interoperability testing. Verify the checkout's runtime version. Memory 1536MiB

Inferno Worker (same external test-kit checkout): Async test runner (Sidekiq). Memory 512MiB

Inferno Redis (redis:7-alpine): Cache for Inferno. Memory 128MB

Inferno Validator (infernocommunity/inferno-resource-validator:1.0.67): FHIR resource validation. Memory 768MB

Healthcare Scanner (custom Dockerfile, Alpine 3.21): Pattern-based healthcare source checks; rule coverage and results require review. Memory 256MiB

Profile "certbot" (1 service):

Certbot (certbot/certbot:v3.3.0): Certificate-renewal loop; initial issuance, lineage paths and Nginx reload remain operator responsibilities

Total: 9 core + 14 on-demand = 23 services

3.2 LightRAG Engine:

LightRAG v1.4.6 supplies the upstream browser UI and HTTP API as well as the retrieval engine. Its source-defined limits are 1024 MiB and 2 CPU units. Port 9621 is available on host loopback as well as through the intended authenticated proxy path, so direct access must be restricted to the appropriate trust boundary.

Upstream capabilities include knowledge-graph construction and retrieval modes combining graph and vector context. Model choice, answer quality, latency and effective provider configuration must be evaluated using the selected release and corpus.

3.3 RAG-Anything (Multimodal Processor):

RAG-Anything v1.2.10 is integrated for multimodal parsing, indexing and querying. The CLI discovers document formats including PDF, Office, text and images. File-extension support does not prove every parser flow works. Its separate index storage is not automatically interoperable with LightRAG merely because input files share a mount.

Usage modes: single-file processing, batch directory processing, and hybrid query mode (naive, local, global, hybrid).

4. SECURITY ARCHITECTURE

4.1 Zero-Trust Authentication:

Authentication is intended for protected proxy routes. The portal, health endpoints and direct loopback ports have different boundaries. Authelia v4.39.15 configuration and integration require validation before claiming enforcement:

- A two-factor access policy is configured for the deployment domain; verify supported MFA methods and recovery

- Protected Nginx locations issue auth subrequests; the selected authorization endpoint and request headers require release-specific validation

- Session management: 1-hour expiration, 5-minute inactivity timeout

- Brute force protection: 3 retries, 120s detection window, 300s lockout

- The user-database example requires an operator-generated Argon2 password hash; no real user account is included

4.2 Transport Security:

- TLS 1.2 and TLS 1.3 with strict cipher configuration (HIGH:!aNULL:!MD5)

- Certbot v3.3.0 provides a renewal loop; verify initial issuance, certificate lineage, mounted paths and reload behavior

- The HTTPS server defines HSTS (max-age 1 year, includeSubDomains, preload); evaluate domain-wide readiness and verify effective responses

4.3 Security Headers:

Configured HTTPS security headers and CSP status:

- HSTS with max-age enforcement and preload
- Content-Security-Policy (CSP): not currently configured; define and test an application-compatible policy before claiming CSP protection
- X-Frame-Options: DENY (prevents clickjacking)
- X-Content-Type-Options: nosniff (prevents MIME sniffing)
- Referrer-Policy: strict-origin-when-cross-origin
- Permissions-Policy: camera=(), microphone=(), geolocation=()

4.4 Network and Container Hardening:

- Rate Limiting: 10 requests/second per IP with burst allowance of 20
- Network Isolation: 3 Docker networks (frontend bridge, backend internal, monitoring internal)
- Container Hardening: Nginx defines no-new-privileges and capability restrictions; all-container hardening and read-only root filesystems are not established by current source
- Secrets Management: private environment files, identity configuration and key material are excluded from Git; automated history/publication scans found no matching secrets

4.5 Vulnerability and Compliance Scanning:

Assessment tooling includes the default SonarQube service and optional profile-based scanners. Run individual tools only against explicitly authorized targets:

- Checkov: Terraform/IaC assessment with coverage determined by the selected rule set
- Trivy v0.69.3: Container image CVE scanning
- SonarQube 10.8: Static code analysis and security hotspots
- OWASP ZAP: Dynamic application security testing
- HAPI FHIR: FHIR R4 validation
- Inferno: ONC Certification g(10) FHIR test kit
- Healthcare Scanner: pattern-based source checks; review false positives, omissions and control effectiveness
- Prowler: Multi-cloud compliance (AWS + GCP + Azure)

Reports generated into reports/ directory with subdirectories for HIPAA, SOC 2, GDPR.

5. OBSERVABILITY STACK

Configuration is provided for four observability areas: metrics, logs, traces and alerts. Connectivity, instrumentation, provisioning and notification delivery still need end-to-end acceptance tests.

5.1 Metrics (Prometheus v2.53.5):

7 configured scrape jobs at 15-second intervals; this count does not imply seven healthy targets:

- Prometheus (localhost:9090): Self-monitoring
- LightRAG (lightrag:9621): RAG engine metrics
- Grafana (grafana:3000/metrics): Dashboard metrics
- Nginx (nginx:80/nginx_status): configured status target; stub_status needs a suitable Prometheus exporter and reachable route

- Jaeger (jaeger:14269/metrics): Tracing metrics

- SonarQube (sonarqube:9000/api/system/status): JSON status endpoint; requires a compatible metrics integration

- Loki (loki:3100/metrics): Log aggregation metrics

5.2 Logging (Loki 3.6.0):

- Several core services use the Loki Docker logging driver; Nginx and Loki use json-file logging, so collection is not uniform across all containers

- Non-blocking logging bounds application waiting but can drop logs under pressure; it does not eliminate performance or reliability impact

- 31-day retention (744 hours)

- Queryable through Grafana using LogQL

- Ingestion rate: 4 MB/s, burst: 6 MB/s

5.3 Distributed Tracing (Jaeger 1.53):

- OTLP receiver ports are published for gRPC (4317) and HTTP (4318); verify the selected Jaeger release enables the intended receivers

- Jaeger UI on port 16686

- End-to-end tracing is a target; LightRAG-to-Jaeger network reachability and actual instrumentation remain unverified

5.4 Alerting (AlertManager v0.27.0):

5 pre-configured alert rules:

- ContainerDown: up == 0 for 1 minute marks a configured scrape target unavailable; it does not discover every stopped container

- HighMemoryUsage: process_resident_memory_bytes above 400 MiB for 5 minutes; coverage depends on exporters, not every container

- HighCPUUsage: rate(process_cpu_seconds_total[5m]) above 0.8 for 5 minutes; this is process CPU-seconds per second, not a universal host percentage

- PrometheusTargetMissing: Fires when target unreachable for 5 minutes (CRITICAL)

- LokiRequestErrors: configured Loki 5xx request-count rate above 0.1 per second for 5 minutes

Alert routing: group by alertname + severity, 30s group wait, 5m group interval, 4h repeat. Critical alerts suppress warnings with the same name. The default receiver has no configured notification transport; prove delivery and resolution with a test alert.

5.5 Dashboards (Grafana 10.2.3):

3 datasource definitions; verify successful runtime provisioning:

- Prometheus (default): Metrics

- Loki: Logs

- Jaeger: Traces

Pre-built Docker Services Overview dashboard with panels for: services up/down, targets status, memory usage, CPU usage.

6. DEPLOYMENT GUIDE

6.1 Prerequisites:

- Docker Engine/Desktop with compatible Compose; budget at least the 4,352 MiB core-limit total plus operating-system and Docker overhead

- Loki Docker driver plugin if retaining the configured driver; review its version, host compatibility and daemon-level permissions before installation

- SSL certificates (fullchain.pem, privkey.pem) for TLS termination

- Google Gemini API key for LLM-powered RAG operations

6.2 Configuration:

Primary configuration via configs/.env:

- GEMINI_API_KEY: custom RAG-Anything adapter credential; LightRAG uses its version-specific binding configuration and credential names

[omitted]

- Scanning service paths: DEV_PATH, TERRAFORM_PROD_PATH, AWS_CONFIG_DIR, GCP_KEY_PATH

- Azure scanner credentials: AZURE_CLIENT_ID, AZURE_CLIENT_SECRET, AZURE_TENANT_ID, AZURE_SUBSCRIPTION_ID

6.3 Deployment Commands:

Start full stack: cd configs && docker compose up -d

Run Terraform assessment: docker compose --profile scan run --rm checkov

Run CVE scan: docker compose --profile scan run --rm trivy

Renew SSL: docker compose --profile certbot up certbot

Build RAG-Anything: docker compose --profile rag build raganything

6.4 Post-Deployment:

- Change default SonarQube admin password (default: admin/admin)

- Rotate Gemini API key if .env was ever committed

- Schedule backup.sh only after scope, retention and failure alerting are validated; daily 2:00 AM is a suggested schedule, not a verified installed task

- Exercise the supplied restore check and a separate whole-system recovery drill; monthly is a suggested review cadence

7. ACCESS AND ENDPOINTS

7.1 Proxy Routes (deployment domain placeholder):

LightRAG: HTTPS deployment-domain /lightrag/ route; intended to require Authelia authorization

Grafana: HTTPS deployment-domain /grafana/ route; intended to require Authelia authorization

Jaeger: HTTPS deployment-domain /jaeger/ route; intended to require Authelia authorization

Prometheus: HTTPS deployment-domain /prometheus/ route; intended to require Authelia authorization

Authelia: HTTPS deployment-domain /authelia/ route; login portal, not itself protected by auth_request

7.2 Local-Only Services (not exposed externally):

SonarQube: http://localhost:9000 (static code analysis admin)

AlertManager: http://localhost:9093 (alert routing dashboard)

LightRAG Direct: http://localhost:9621 (bypasses Nginx, development only)

Loki API: http://localhost:3100 (log queries)

Jaeger OTEL: http://localhost:4317/4318 (trace ingest receivers)

8. RAG-ANYTHING USAGE

Multimodal document processing for PDFs, images, tables, and equations. Activated via Docker Compose rag profile.

Processing Commands:

Single file: docker compose --profile rag run --rm raganything /app/data/inputs/yourfile.pdf

Batch directory: docker compose --profile rag run --rm raganything --dir /app/data/inputs/

Query hybrid mode: docker compose --profile rag run --rm raganything --query "What are the main findings?" --mode hybrid

Discovered input formats include PDF, DOCX, XLSX, PPTX, text and common images. Parser compatibility, tables/equations and vision behavior require format-specific tests.

Query modes: naive, local, global, hybrid (recommended)

9. RESOURCE ALLOCATION

Core memory limits total 4,352 MiB (4.25 GiB). An 8 GiB host is a baseline planning assumption; optional jobs and host overhead require additional capacity:

Core Services:

LightRAG: 1024MB memory, 2.0 CPU

SonarQube: 1536MB memory, 2.0 CPU

Prometheus: 384MB memory, 1.0 CPU

Grafana: 384MB memory, 1.0 CPU

Loki: 384MB memory, 1.0 CPU

Jaeger: 256MB memory, 1.0 CPU

Nginx: 128MB memory, 0.5 CPU

Authelia: 128MB memory, 0.5 CPU

AlertManager: 128MB memory, 0.5 CPU

All nine core services define health checks; optional services do not all do so. A health check is not proof of a working authenticated RAG journey.

10. TECHNOLOGY STACK SUMMARY

Historical language-composition figures from the earlier edition were Shell scripts 77.2%, Python 20.0%, Dockerfiles 2.8%. They were not remeasured for this release; the repository also includes deployment configuration and expanded documentation.

Core Technologies:

LLM integration: configurable model settings; custom RAG-Anything callbacks are currently Gemini-specific

RAG Engine: LightRAG 1.4.6

Multimodal Processor: RAG-Anything 1.2.10

Reverse Proxy: Nginx 1.28-alpine

Authentication: Authelia 4.39.15 (2FA/TOTP/WebAuthn)

Container Orchestration: Docker Compose; validate compatibility with the installed version rather than treating Docker Desktop's version as Compose's

Metrics: Prometheus v2.53.5

Logging: Loki 3.6.0 + Docker Driver

Tracing: Jaeger 1.53 + OpenTelemetry

Dashboards: Grafana 10.2.3

Code Analysis: SonarQube 10.8-community

IaC Scanning: Checkov (latest)

CVE Scanning: Trivy 0.69.3

SSL Management: Certbot v3.3.0

Backup: Restic (encrypted, incremental)

11. MAINTENANCE AND OPERATIONS

11.1 Backup Strategy:

scripts/backup.sh provides a manually runnable Restic workflow (encrypted, deduplicated, incremental). Scheduling and application-consistent snapshot handling are operator responsibilities.

Scope: configuration, LightRAG data and Grafana files in separate snapshots. Identity SQLite and Grafana database files are excluded; uploads and RAG-Anything output/index storage are not covered.

Retention: 7 daily, 4 weekly, 6 monthly snapshots. Auto-prune after each backup.

Suggested schedule: daily at 2:00 AM using an operator-managed scheduler after the workflow is validated; no installed schedule was checked.

The supplied verification checks integrity and restores only the newest snapshot, not all backup groups, then removes its fixed temporary restore directory. Review that path before use. A complete recovery drill must restore every required data/identity group on a separate host.

11.2 SSL Certificate Renewal:

Certbot v3.3.0 defines a renewal loop, and Nginx has an ACME challenge route. Operators must first issue certificates, align lineage/mount paths, and test renewal plus Nginx reload; current renewal success was not measured.

11.3 Log Retention:

Loki configuration requests 31-day retention (744 hours). Verify effective deletion, storage capacity and backup retention in the running deployment.

11.4 Security Updates:

Review image and dependency advisories; use Trivy and SonarQube as assessment inputs. Scans do not update images automatically. Pin reviewed versions/digests, test upgrades, and retain a rollback path.

12. CONCLUSION

The RAG Production Stack supplies an open-source starting point for authenticated RAG services and operational tooling. Production use, particularly with regulated information, requires resolving the documented integration gaps, implementing deployment-specific controls, and obtaining workload, recovery, security and legal evidence. No healthcare certification, SOC 2 attestation, GDPR determination or measured production availability is established by this reference repository.

The modular Compose baseline can be adapted and extended. Large-scale or highly available operation requires deliberate changes to state storage, identity, queues, authorization, networking and deployment architecture. The following added sections define the future roadmap and acceptance gates.

13. OPEN-SOURCE PROJECT AND APPLICATION BOUNDARIES

The repository is open source under the MIT license. Repository-authored code and documentation may be used, modified and redistributed subject to that license. Third-party applications, container images, plugins, model SDKs and datasets retain their own terms. The assembled runtime must not be described as entirely MIT-licensed without reviewing those components.

The repository supplies a self-hosted reference stack, not a managed hosting service. Community support does not create a response-time guarantee or contractual SLA. Organizations operating a deployment own their customer terms, privacy notices, service commitments, costs and operational controls.

Frontend, backend and client contracts

The browser UI and HTTP API come from the upstream LightRAG image. No independent custom frontend application or custom HTTP route implementation is tracked here. The custom Python component is a RAG-Anything ingestion/query CLI. Its Gemini-specific callbacks are separate from LightRAG's upstream provider bindings.

Before integrating a client, inspect the OpenAPI schema and authentication behavior of the exact deployed image. Validate UI assets, redirects, cookies, base paths, streaming, cancellation and error handling through Nginx. Machine clients need an explicitly supported authentication contract; a browser login does not define service-account authorization.

Future API work should define versioned document submission/status/deletion and query contracts, idempotent ingestion, bounded retries and deadlines, stable error codes, pagination, quotas, request identifiers and audit events. Enforce document and tenant authorization before retrieval, downloads, citations and cache hits.

Open-source contribution and support guide

14. CURRENT INTEGRATION GAPS AND CONFIGURATION OWNERSHIP

Resolve these baseline gaps before expanding the system or describing it as production-validated. Each item requires a reproducible test and evidence from the intended deployment.

- Networking: provide controlled model-provider egress; repair LightRAG-to-Jaeger and Prometheus-to-LightRAG reachability without publicly exposing backend services.
- Observability: use compatible exporters for Nginx/SonarQube, verify Grafana's provisioning search path, prove actual trace spans, and configure/test an Alertmanager delivery channel.
- Identity and ingress: validate Authelia's example schema and Nginx authorization endpoint/headers for the selected versions; exercise MFA, logout, revocation, denial behavior, subpath assets and health-check protocol/trust.

- CLI correctness: processing exceptions currently do not propagate a reliable batch failure status; directory discovery is non-recursive; the vision callback does not explicitly forward image-specific arguments.
- Reproducibility and recovery: lock compatible dependencies and image digests, include missing identity/upload/index state in backup scope, and restore all snapshot groups onto a separate host.

One configuration boundary

Use private environment or secret-store inputs for credentials; central typed settings for environment/provider behavior; validated data/templates for maintained content; and dedicated modules for immutable protocol syntax. Model IDs, provider URLs, timeouts, token limits, prices, quotas and storage paths must not be duplicated across business logic.

Current configuration debt remains: model fallbacks, parser/storage settings, token limits, provider-specific imports and deployment-specific scan targets are still embedded in source or static configuration. The documentation release added an environment example and recorded this debt; it did not complete a configuration refactor.

Set explicit LLM_BINDING, LLM_MODEL, LLM_BINDING_API_KEY and any required LLM_BINDING_HOST for the selected upstream release. Likewise verify EMBEDDING_BINDING, EMBEDDING_MODEL, EMBEDDING_DIM, EMBEDDING_BINDING_API_KEY and any required host setting. The custom adapter separately reads GEMINI_API_KEY. PROJECT_PATH and PROJECT_TYPE select the healthcare scanner input. Empty example values are placeholders, not a working credential set.

Detailed configuration and verification backlog

15. FUTURE SCALABILITY: 1M+ SIMULTANEOUS USERS

The future target is at least one million simultaneous user sessions with a separately specified active-query workload. It is not a benchmark achieved by the current single-host Compose baseline. Connected users, requests per second and concurrent model generations must be measured separately.

Example workload and capacity calculation

Planning assumptions: 1,000,000 concurrent sessions; 10% actively querying; one query per active user per minute; 12-second mean query duration; 50% of answers served by an authorized valid application cache; and 2,000 input plus 500 output model tokens per uncached query.

Those assumptions imply about 1,667 queries per second and 20,000 in-flight queries. Approximately 833 queries per second reach the model provider, requiring about 1.67 million input tokens and 0.417 million output tokens per second. This simplified model assumes one provider call per uncached query; extraction, rewriting, reranking and retries add demand.

Sensitivity: if every session asks once per minute, demand becomes about 16,667 queries per second and 200,000 in-flight queries. One million actual simultaneous 12-second generations implies approximately 83,333 completions per second in steady state. Provider quotas, inference capacity and cost approval must be established for the selected scenario.

Distributed architecture required

The proposed design uses CDN-delivered UI assets; regional traffic filtering and ingress; stateless authenticated API replicas; highly available identity/session state; tenant-scoped caches; replicated or partitioned vector/graph storage; encrypted object storage; durable ingestion queues; isolated parser/embedding workers; and a model gateway with quotas, deadlines, circuit breakers and budget controls.

Separate interactive queries from bulk ingestion. Bound every queue, use idempotent jobs and retry limits with jitter, and define overload responses. Autoscale using queue age, active streams, request/token demand and tail latency rather than CPU alone. Define shard ownership, index consistency, migration and failover before distributing state.

Cache keys must include tenant, authorization-policy version, corpus/index revision, model/prompt version and freshness rules. Permission changes and deletions must invalidate affected entries. Cross-region operation additionally requires residency decisions, replication-lag limits, conflict policy and a tested recovery-region provider quota.

Validation ladder and cost controls

Progress through a measured baseline, then 1,000, 10,000, 100,000 and 1,000,000+ simultaneous sessions. At every stage publish the query arrival rate, corpus/tenant distribution, token sizes, cache hit rate, topology, versions, latency percentiles, errors, rejection rate and spending assumptions.

Proposed distributed test campaign: 30-minute warmup, 60-minute steady load, a 2× arrival-rate burst and a 24-hour soak; at the final stage include reconnect storms and zone loss under load. Progress only if authorization remains correct, quality/latency/error targets hold, queues recover and the approved budget is respected.

Total operating cost includes model tokens, compute, storage, network egress, observability, backups, security services and load generation. Record actual price sources and currencies, expected/uncached-peak/degraded-provider scenarios, retry amplification, headroom and applicable taxes. Set tenant and global spend ceilings with a tested stop mechanism. Open-source licensing does not make hosted operation free; obtain explicit budget approval before paid provisioning or tests.

Full scalability calculations and architecture

16. AVAILABILITY OBJECTIVES AND DISASTER RECOVERY

Proposed deployment objectives use a rolling 30-day measurement window: 99% for a validated pilot, 99.9% for a multi-zone service, and optionally 99.95% for a higher-reliability design. These are engineering targets, not measured history or a repository-supplied contractual SLA.

For time-based availability, the corresponding 30-day error budgets are 7 hours 12 minutes, 43 minutes 12 seconds, and 21 minutes 36 seconds. A 99% objective permits more than seven hours of unavailability per month; evaluate whether that is acceptable for the users. A request-based budget instead equals eligible requests multiplied by one minus the objective.

Measure authenticated UI access, login, completed queries, ingestion freshness and deletion effectiveness. Define eligible requests and deadlines before launch. Provider failures affecting legitimate users count as failures; do not hide overload by silently excluding rejected legitimate traffic. Track answer quality and latency separately from availability.

Initial latency proposals for a bounded short-answer class are p95 first token within 3 seconds and p95 completion within 20 seconds. Corpus, geography, token bounds and provider choice must be fixed in the test manifest. These proposals need workload-specific validation.

Use short- and long-window burn alerts with an accountable responder. If the error budget is exhausted, prioritize reliability and reviewed security fixes over feature rollout. A single host is a shared failure domain for application, identity, local state and monitoring; higher objectives require redundancy and exercised failover.

Proposed pilot recovery objectives are at most 24 hours of data loss and recovery within 8 hours; a distributed service should investigate at most 15 minutes of committed-source-data loss and recovery within 1 hour. Derived-index rebuild time is corpus-specific. Agree stricter objectives where appropriate and record actual recovery-point and elapsed-time measurements from a separate-host drill.

Reliability definitions and recovery gates

17. LEGAL, PRIVACY AND RESPONSIBLE OPERATION

This is an open-source technical readiness guide. Legal duties depend on jurisdiction, deployment role, data, contracts and intended use. The operator must obtain qualified review of its actual service before publishing binding policies or processing regulated data; the repository does not provide a completed customer contract or compliance certification.

Data governance and privacy lifecycle

Inventory identities, uploads, extracted content, chunks, embeddings, graph entities, questions, prompts, answers, telemetry, assessment reports and backups. For each, record purpose, owner, sensitivity, legal basis where required, recipients, region, retention and deletion procedure. Derived indexes can contain sensitive information.

Enforce access before retrieval; minimize sensitive content in logs; verify provider retention/training/subprocessor terms; and implement identity-verified export/deletion requests. Deletion must cover original files, vectors, graphs, caches and replicas. Explain backup expiry and replay later deletions or permission revocations after restoration.

For GDPR-related deployments, assess controller/processor roles, lawful basis, transparency, rights, international transfers and whether a DPIA or DPO is required. Do not adopt one universal retention duration or incident-notification deadline without reviewing the actual obligations.

Healthcare, assurance and AI governance

HIPAA applicability depends on covered-entity/business-associate relationships and protected health information. Assess safeguards and whether vendor arrangements require business associate agreements. FHIR validation, Inferno tests or scanner reports do not independently establish HIPAA compliance or ONC certification. SOC 2 requires organizational controls and independent attestation; a tool installation is not an attestation.

Clinical or consequential uses require separate intended-use, human-review, safety and regulatory evaluation. Assess applicable AI rules according to the role performed; open-source status does not resolve every obligation. Document intended use, limitations, evaluation data, model versions, output labeling, foreseeable misuse and escalation paths.

Operators must have rights to process uploaded content and define lawful hosted-service usage, complaint and removal procedures. Hosted-service terms should not be confused with additional restrictions on this repository's MIT license. Review third-party licenses for the exact resolved artifacts before distributing a runtime.

Before operator launch: establish privacy notices and service terms where applicable, vendor/data-processing agreements, retention schedules, incident procedures, accessibility review, license inventory, and any explicitly promised support or availability commitments.

Legal/privacy readiness and authoritative references

18. SECURITY DESIGN AND CONTRIBUTION PRACTICE

Protect browser-to-ingress, ingress-to-identity/application, application-to-tenant-data, parser-to-upload, provider, operator and CI boundaries. Required controls include document authorization, safe output rendering, prompt-injection defenses, isolated parsing, file/body limits, outbound-destination controls, quotas, secret rotation and supply-chain review.

Test cross-tenant denial, malicious retrieved instructions, script-bearing output, malformed/oversized documents, internal-address fetch attempts, long-input abuse, retry storms and direct-port/header bypasses. A read-only Docker socket mount still grants powerful API access and must be treated as privileged.

Use generated or appropriately licensed test documents. Report vulnerabilities privately through GitHub's reporting option when enabled; otherwise request a private channel without disclosing exploit details in public issues. Community maintenance does not imply a guaranteed response time.

Contributions should start with a concrete acceptance criterion or failing reproduction, keep scope bounded, externalize configuration, retain attribution, and include relevant documentation, security checks and real-flow evidence. Obtain independent review for nontrivial runtime changes. Never bypass a scanner finding merely to make CI green.

Security design and vulnerability reporting

19. IMPLEMENTATION ROADMAP AND RELEASE ACCEPTANCE

Phase 0 — Reproducible baseline: resolve integration gaps, centralize configuration, pin dependencies, correct CLI outcomes/vision forwarding, and repair recovery coverage. Gate: clean-checkout setup plus authenticated/unauthorized query, ingestion, deletion and whole-system restore tests.

Phase 1 — Measured pilot: instrument user journeys, define tenant/document policy, add bounded jobs/provider budgets and evaluate 1,000 sessions. Gate: declared workload results, a full availability measurement window, recovery evidence and acceptable retrieval quality.

Phase 2 — Distributed service: stateless API/UI delivery, shared identity, durable queues, supported shared storage, idempotent index publication and multi-zone rollout. Gate: 10,000-session validation with worker/node failure and quota saturation.

Phase 3 — Regional scale: tested partition ownership, capacity/lifecycle automation and 100,000-session workloads with ingestion and tenant skew. Gate: stable latency, deletion behavior, cost and recovery, plus the measured 99.9% objective over its full window.

Phase 4 — 1M+ sessions: distributed generators, sufficient provider/inference quotas, reconnect and zone-loss tests, and sanitized capacity/quality/cost publication. Gate: the declared workload meets objectives without data leakage, unbounded retries or exceeded spending limits.

Before each deployment, detect drift, make and test changes locally, deploy the reviewed artifact, and confirm matching hashes/revisions. Preserve rollback capability and test migrations. For RAG quality, use versioned known-answer, unanswerable, conflicting, stale, adversarial and permission-boundary examples; evaluate grounding, citations, abstention, freshness, leakage, latency and token consumption.

No delivery dates or funded staffing commitments are promised by this roadmap. Each phase needs an accountable maintainer/operator role, acceptance evidence and an approved operating budget.

Roadmap, testing and operations handbook

20. PUBLICATION EVIDENCE AND FURTHER READING

Public repository commit 5d465c0 updated 34 files, including all five pre-existing READMEs, the public handbook, environment example, publication safeguards and security CI. Remote main matched local HEAD after a normal push. GitHub's Gitleaks, Semgrep and Bandit jobs all passed.

Local verification resolved 136 documentation links and passed YAML/JSON parsing, Python AST checks, Bash syntax, environment-name coverage, private-file exclusions and isolated Compose resolution using test values. Secret scans found no matches. Semgrep's Nginx health-header inheritance finding was reproduced and corrected by using default_type for the response MIME type.

Not performed: live deployment/browser/provider validation, container builds, project type checking, independent review, a million-user load test, sustained uptime measurement, or whole-system recovery timing. The runtime integration backlog remains explicit. This evidence supports the source/documentation publication, not a production-readiness certification.

Public documentation index

Published commit and successful security workflow

Verification record and known limitations

Google SRE guidance on service-level objectives

OWASP guidance on GenAI security

European Commission guidance on data-protection principles

HHS guidance on HIPAA and cloud computing

European Commission AI Act overview

RAG Production Stack

Open-Source RAG Reference Infrastructure with Authentication and Observability Configuration

Built with Docker | LightRAG | Authelia | Prometheus | Grafana | Jaeger | Checkov | Trivy

Developed by Zahidul Islam

zahidul-islam.vercel.app | github.com/Zahidulislam2222