LightRAG Health Monitoring Runbook

LightRAG Health Monitoring Runbook

> **Domain note (2026-05-17):** References to `lightrag.basicconsulting.no` and `ollama.basicconsulting.no` are legacy hostnames. Current live endpoints: `lightrag.alai.no` and `ollama.alai.no`.

Status: ACTIVE
Created: 2026-04-21
Owner: FlowForge (AgentForge)
Related: MC #8545, INFRA-CF-001

Purpose

Continuous health monitoring for LightRAG stack (Azure VM + Cloudflare) following the 2026-04-20 outage fix (CF Browser Integrity Check configuration).

This runbook covers:

- Health check script usage
- Interpreting results
- Automated monitoring setup
- Troubleshooting common issues
- Rollback procedures

Architecture Overview

LightRAG runs on Azure VM (20.240.61.67:9621) and is exposed via Cloudflare tunnel at `https://lightrag.basicconsulting.no`. The system depends on:

1. **Azure VM** — Docker containers (lightrag + neo4j)
2. **Cloudflare tunnel** — Routes traffic through Mac Studio relay
3. **Cloudflare Access** — Authentication via service tokens
4. **Cloudflare BIC rule** — Allows automation clients (Python UA)
5. **Ollama upstream** — `https://ollama.basicconsulting.no` for LLM inference

See: Azure LightRAG Migration Runbook

Health Check Script

Location

`~/system/tools/lightrag-health.sh`

Manual Execution

```
bash ~/system/tools/lightrag-health.sh
```

Output

- **Terminal:** Colored status summary (green/yellow/red per layer)
- **JSON:** `~/system/evidence/lightrag-health-YYYYMMDD-HHMMSS.json` (machine-readable)
- **Markdown:** `~/system/evidence/lightrag-health-YYYYMMDD-HHMMSS.md` (human-readable)

Exit Codes

- `0` — All checks passed (healthy)
- `1` — Warnings detected (degraded but operational)
- `2` — Errors detected (critical issues)

Check Layers

Layer 1: Azure VM Health

Check	What it tests	Healthy criteria
`direct_access`	Direct HTTP to VM IP:port	HTTP 200, status=healthy
`docker_containers`	Container status via SSH	lightrag + neo4j running, healthy

Note: SSH access currently unavailable (publickey auth). Manual verification required via Azure Portal or after SSH key setup.

Layer 2: Cloudflare Network

Check	What it tests	Healthy criteria
`cf_tunnel`	HTTPS via CF tunnel	HTTP 200, latency < 2s
`cf_bic_rule`	BIC rule configuration	Rule enabled, covers both endpoints
`python_ua`	Python client access	HTTP 200 with Python UA

Critical: `python_ua` check verifies the CF-BIC-001 rule is active. If this fails with HTTP 403, automation clients (pi-orchestrator, lightrag-outbox-ingest.js) will break.

Layer 3: Application Health

Check	What it tests	Healthy criteria
`health_endpoint`	`/health` endpoint	status=healthy, pipeline_busy=false
`query_endpoint`	`/query` with naive mode	HTTP 200, valid response, < 30s

Note: First query after idle may take longer (cold start). If timeout, retry once.

Layer 4: Ollama Upstream

Check	What it tests	Healthy criteria
`api_tags`	Ollama model availability	qwen2.5-coder:32b + bge-m3 present

Critical: LightRAG requires these specific models. If missing, queries will fail.

Interpreting Results

Green (Exit 0) — Healthy

All critical checks passed. System operational.

Action: None required.

Yellow (Exit 1) — Warnings

Non-critical issues detected. System degraded but operational.

Common warnings:

- SSH access unavailable (known limitation)
- CF API token unavailable (can't verify BIC rule, but Python UA test compensates)
- Slow response times (> 2s but < 30s)

Action: Review warning details. Monitor next check. Escalate if warnings persist 3+ checks.

Red (Exit 2) — Errors

Critical issues detected. System may be non-operational or partially failed.

Common errors:

- Query endpoint timeout (> 30s)
- HTTP 403 from Python UA (BIC rule disabled)
- Ollama models missing
- Direct VM access failed

Action:

1. Review error details in JSON evidence
2. Follow troubleshooting section below
3. If unresolved after 30 min, consider rollback (see Azure LightRAG Migration Runbook)

Automated Monitoring Setup

LaunchAgent Installation (DRAFT — Pending Alem Approval)

Draft file: `~/system/evidence/lightrag-monitor-launchagent-draft.plist`

Schedule: Daily at 9:00 AM (frequent for 4-week monitoring period)

Installation steps (when approved):

```
# 1. Copy draft to LaunchAgents
cp ~/system/evidence/lightrag-monitor-launchagent-draft.plist \
   ~/Library/LaunchAgents/com.john.lightrag-monitor.plist

# 2. Load the agent
launchctl load ~/Library/LaunchAgents/com.john.lightrag-monitor.plist

# 3. Start immediately (optional)
launchctl start com.john.lightrag-monitor
```

Manual trigger:

```
launchctl kickstart -k gui/$(id -u)/com.john.lightrag-monitor
```

Logs:

- stdout: `~/system/logs/lightrag-monitor/stdout.log`
- stderr: `~/system/logs/lightrag-monitor/stderr.log`

Slack Alerts (To Be Implemented)

When LaunchAgent detects exit code 2 (errors), send alert to `#alerts` channel:

```
node ~/system/tools/slack.js send alerts "□□LightRAG health check FAILED at $(date). Check
~/system/evidence/lightrag-health-*.json"
```

This requires wrapping the health check script in a post-execution hook (see plist comments).

Health History Database

Location: `~/system/databases/lightrag-health.db`

Schema: `~/system/tools/lightrag-health-db-init.sql`

Tables

- `health_checks` — Overall check results
- `health_check_details` — Individual layer/check results

Views

- `health_checks_summary` — Last 30 checks
- `health_checks_trend` — Daily aggregates

Query Examples

Last 10 checks:

```
sqlite3 ~/system/databases/lightrag-health.db \
  "SELECT timestamp, overall_status, errors, warnings FROM health_checks ORDER BY created_at
DESC LIMIT 10;"
```

Trend over last 7 days:

```
sqlite3 ~/system/databases/lightrag-health.db \
  "SELECT * FROM health_checks_trend WHERE check_date >= date('now', '-7 days');"
```

All errors in last 24 hours:

```
sqlite3 ~/system/databases/lightrag-health.db \
  "SELECT hc.timestamp, hcd.layer, hcd.check_name, hcd.message FROM health_checks hc
   JOIN health_check_details hcd ON hc.id = hcd.health_check_id
   WHERE hcd.status = 'error' AND hc.created_at >= datetime('now', '-24 hours');"
```

Note: Database logging will be implemented in next iteration of the health script.

Troubleshooting

Issue: Query endpoint timeout (HTTP 000, 35s)

Possible causes:

1. First query after idle (cold start)
2. Ollama FORGE overloaded
3. Network path issue (Mac Studio → CF → Azure → CF → Mac Studio)

Diagnosis:

```
# Test Ollama upstream directly
curl -s https://ollama.basicconsulting.no/api/tags \
  -H "CF-Access-Client-Id: $(grep CF_ACCESS_CLIENT_ID ~/Library/LaunchAgents/com.john.pi-
orchestrator.plist | sed 's/.*<string>\(.*\)<\/string>/\1/')" \
  -H "CF-Access-Client-Secret: $(grep CF_ACCESS_CLIENT_SECRET
~/Library/LaunchAgents/com.john.pi-orchestrator.plist | sed
's/.*<string>\(.*\)<\/string>/\1/')" | jq '.models | length'

# Check if FORGE is responding
curl http://10.0.0.2:11434/api/ps

# Test query directly with extended timeout
curl -s --max-time 60 \
  -H "CF-Access-Client-Id: ..." \
  -H "CF-Access-Client-Secret: ..." \
  -H "Content-Type: application/json" \
  -X POST \
  -d '{"query":"test","mode":"naive","only_need_context":false}' \
  https://lightrag.basicconsulting.no/query
```

Fix:

- If cold start: Retry once, should succeed
- If FORGE overloaded: Identify competing workload, throttle/stop
- If persistent: Check Azure LightRAG Migration Runbook for tunnel troubleshooting

---

Issue: Python UA blocked (HTTP 403)

Root cause: CF Browser Integrity Check rule disabled or misconfigured.

Diagnosis:

```
# Test with Python UA
curl -s -w "\nHTTP: %{http_code}\n" \
  -A "Python/3.11 urllib/1.26" \
  -H "CF-Access-Client-Id: ..." \
  -H "CF-Access-Client-Secret: ..." \
  https://lightrag.basicconsulting.no/health
```

Fix:

1. Verify CF Configuration Rule (Ruleset `4fc2c122d04d4791a5d17409b097c510`, Rule `c5990f19f655441180ae886f4512de40`)
2. Ensure rule is enabled and expression includes `lightrag.basicconsulting.no`
3. See: `~/system/rules/cf-proxied-api-bic-whitelist.md`

Critical: This is a repeat of the 2026-04-20 outage. If rule is disabled, all automation breaks.

Issue: Ollama models missing

Symptoms: `api_tags` check fails or warns about missing models.

Required models:

- `qwen2.5-coder:32b-instruct-q8_0` (LLM inference)
- `bge-m3:latest` (embeddings)

Fix:

```
# SSH to FORGE (10.0.0.2)
ssh admin@10.0.0.2

# Pull missing models
ollama pull qwen2.5-coder:32b-instruct-q8_0
ollama pull bge-m3:latest

# Verify
ollama list | grep -E "(qwen2.5-coder:32b-instruct-q8_0|bge-m3:latest)"
```

Issue: Direct VM access failed

Symptoms: `direct_access` check returns HTTP error or timeout.

Diagnosis:

```
# Test direct HTTP
curl -s --connect-timeout 5 http://20.240.61.67:9621/health

# Check NSG rules (Mac Studio IP may have changed)
az network nsg rule show \
  -g rg-alai-lightrag \
```

```
  --nsg-name vm-alai-lightragNSG \
  -n allow-lightrag-macstudio \
  --query "sourceAddressPrefix"

# Compare to current ISP IP
curl -s https://ifconfig.co
```

Fix: If Mac Studio ISP IP rotated, update NSG rule:

```
NEW_IP=$(curl -s https://ifconfig.co)
az network nsg rule update \
  -g rg-alai-lightrag \
  --nsg-name vm-alai-lightragNSG \
  -n allow-lightrag-macstudio \
  --source-address-prefixes "${NEW_IP}/32"
```

Note: Azure resources (rg-alai-lightrag) are not currently visible via `az` CLI. This may indicate different subscription or access issue. Direct HTTP access confirms VM is operational.

Rollback Procedure

If LightRAG stack becomes unstable (exit code 2 persisting > 30 min, or CEO directive):

Follow: Azure LightRAG Migration Runbook → Section "Rollback Procedure"

Summary:

1. Revert consumer URLs from `https://lightrag.basicconsulting.no` to `http://localhost:9621`
2. Restart local Docker LightRAG
3. Verify local service
4. Optionally deprovision Azure VM

Expected rollback time: 5-15 minutes
Data loss risk: ZERO (local volumes preserved)

Maintenance

Weekly Tasks (First 4 Weeks)

- [ ] Review health check trend via database query
- [ ] Check for persistent warnings/errors
- [ ] Verify evidence files are being generated
- [ ] Compare latency trends (p50, p95)

After 4 Weeks

If system stable (no exit code 2 in 4 weeks):

- Reduce monitoring frequency from daily to weekly
- Update LaunchAgent `StartCalendarInterval` to run on Mondays only
- Archive old evidence files (keep last 30)

Evidence Files

All health checks generate timestamped evidence:

Location: `~/system/evidence/lightrag-health-YYYYMMDD-HHMMSS.*`

Retention: Keep last 30 days, archive older to Azure Blob Storage.

Example archive command (to be automated):

```
find ~/system/evidence -name "lightrag-health-*.json" -mtime +30 \
  | xargs tar -czf ~/system/evidence/archive-$(date +%Y%m).tar.gz

# Upload to Azure Blob
az storage blob upload \
  --account-name plockfrontstaging \
  --container-name evidence \
  --name lightrag-health-archive-$(date +%Y%m).tar.gz \
  --file ~/system/evidence/archive-$(date +%Y%m).tar.gz
```

Related Documentation

- Azure LightRAG Migration Runbook — Full migration details + rollback
- CF-BIC Whitelist Rule — INFRA-CF-001

- MC Task #8545 — Health monitoring project

Changelog

2026-04-21 — Initial version (baseline setup + first run)

Document Owner: FlowForge
Last Updated: 2026-04-21
Approved By: Pending Alem approval for LaunchAgent installation

Revision #9
Created 2026-04-21 11:16:05 UTC by John
Updated 2026-06-21 20:03:15 UTC by John