Your Health Records, On Your Terms.

Two ways in:
a 3-minute chat-only path, or a 30-minute fully-private self-host.

“ You own your records. The cloud is just a courier. ”

Edition · 2026-05-05
github.com/aks129/HealthClawGuardrails

STEP →

Pick your path

There are two ways to get your records into a conversation with Claude. Both keep your data under your control. Pick whichever matches the time you have and how comfortable you are with a terminal.

	PATH A · No terminal	PATH B · Self-host
Time	≈ 3 minutes	≈ 30 minutes
What you do	Click around in claude.ai. That's it.	Run a few commands on your laptop
Tools you need	A claude.ai account. A health-system login.	claude.ai + Docker + Python
Where your records live	In your active Claude conversation. Closed when you close the chat.	On your machine. Never leave.
What you can do	Ask anything: care gaps, lab trends, prep for a visit	Same — plus Telegram bots, audit trail, full guardrails
Best for	"I want to talk to Claude about my records."	"I want my own private store with agents."
Where it lives	Pages 3 – 5	Pages 6 – 11

Not sure? *Start with Path A. It takes 3 minutes and you'll know within 10 if it's enough for what you want. Path B is always there if you want to go deeper later — and many people end up running both.*

Privacy guarantee (both paths)

Records flow EHR → connector → wherever you chose to put them. The cloud only ever appears as the OAuth provider that proves you're you. PHI redaction is built in.

PATH A **No terminal needed**

STEP A · 1

Connect HealthEx (3 minutes)

HealthEx is a free service that connects to the US health-data exchange networks (Carequality, CommonWell, eHealth Exchange) and pulls records from any participating EHR — Epic, Cerner, MEDITECH, athenahealth, AllScripts, and most major US health systems.

1. Go to **healthex.io** and create a free account.
2. In HealthEx, connect each health system you've used. Each is a browser SMART-on-FHIR login — same as logging into MyChart.
3. Open **claude.ai** → **Settings** → **Integrations** → find **HealthEx** → click **Connect**.
4. Authorize once more in the popup. The HealthEx tools are now live in every Claude conversation you start.

Confirm it worked

In a fresh Claude conversation, paste:

PROMPT — paste into Claude

Check when my health records were last updated.

Claude calls the `update_and_check_recent_records` tool and reports your sync status. If you see a friendly summary, you're done with setup — no terminal, no install, no Docker. Skip to **Step A·2** for prompts to try next.

Privacy in Path A. *HealthEx reads your records read-only — it cannot write to your EHR. Records arrive in the active Claude conversation and are gone when you close it. Anthropic doesn't keep them. Nothing is uploaded anywhere else.*

STEP A · 2

Prompts you can paste right now

Each block below is a complete prompt — copy it into Claude as-is. Edit anything in [brackets] for your own situation.

Get the lay of the land

PROMPT — paste into Claude

Get my health summary. Show me active conditions, current medications, recent labs, immunizations, and any allergies — all on one page.

PROMPT — paste into Claude

Build a chronological timeline of my medical conditions from first documented to present. For each active condition, note how long I've had it and whether there's documented treatment.

Trends over time

PROMPT — paste into Claude

Pull my lab results for the last 5 years. Identify any values that have been trending in a concerning direction, even if still within the normal range. Flag anything that's been consistently at the edge of the reference range.

PROMPT — paste into Claude

Compare my most recent labs to my results from 2 years ago. What has improved? What has gotten worse?

Preventive care & gaps

PROMPT — paste into Claude

Based on my age, gender, conditions, and immunization history, identify any preventive care I may be overdue for. Reference USPSTF guidelines for screening recommendations.

Pre-appointment prep

PROMPT — paste into Claude

I have an appointment with a [cardiologist] on [date] for [reason]. Pull my relevant history and prepare a 1-page summary I can bring. Include relevant conditions, current medications, recent labs, and 3 questions I should ask based on gaps in my record.

Medication review

PROMPT — paste into Claude

Review my medication history for the last 5 years. Identify any meds that were started then stopped (and why if documented), any dosage changes over time, and any gaps in chronic medication coverage.

STEP A · 3

Want to see the guardrails? Try the public demo

If you're curious what the HealthClaw guardrails look like in action — PHI redaction, data-quality flags, audit trails — there's a public demo you can poke at without installing anything.

In your browser (zero install)

Open **healthclaw.io/r6-dashboard**. You'll see a live, interactive dashboard pre-seeded with a sample patient (Maria Rivera) whose record has intentional data-quality issues. Try the Curatr panel — it'll flag an ICD-9 code that should be ICD-10, and propose the exact fix.

In Claude Desktop (one config-file edit)

If you want Claude to talk to the demo over MCP — same way it would talk to your own self-hosted instance — paste this into your Claude Desktop config:

```
{
  "mcpServers": {
    "healthclaw-demo": {
      "type": "streamable-http",
      "url":  "https://healthclaw.up.railway.app/mcp",
      "headers": { "X-Tenant-ID": "desktop-demo" }
    }
  }
}
```

Restart Claude Desktop. Then ask:

PROMPT — paste into Claude

Use the healthclaw-demo tools. Get a step-up token, then search for all Patients in the demo store. Read the Condition for the patient, then run curatr_evaluate on it — it has an ICD-9 code that should be flagged.

That's the whole non-technical path

Records connected. Prompts ready. Demo to play with. Zero terminal commands. If that's enough for what you need — you're done. If you want a fully-private store on your own machine plus Telegram bots and an audit trail you control, the rest of this guide is Path B.

Hint: *Many people start in Path A, get comfortable, and graduate to Path B for the records they really care about. There's no wrong order.*

PATH B **Self-host the full stack**

STEP B · 0

What you're about to build

By the end of Path B your machine — laptop, Mac mini, Linux box — will hold a complete, private, agent-mediated view of your records. Nothing leaves the box unless you say so.

The stack, top to bottom:

You	`Telegram · Claude Desktop · Claude Code · Web`
▼	
OpenClaw	`Personas · Skills · Multi-channel gateway  :4319`
▼	
HealthClaw	`PHI redaction · Audit · Step-up auth · MCP  :5000`
▼	
FHIR Server	`HAPI or Medplum — your records live here  :8080`
▼	
Your records	`Never leave this machine.`

Prerequisites

macOS / Linux	`uname -sm`
Python 3.11+	`python3 --version`
Node 22+	`node --version`
Docker (optional)	`docker --version`
git	`git --version`
~30 minutes	—

STEP B · 1A

Install OpenClaw

OpenClaw is your local AI gateway. It runs the agent personas, exposes them on whichever channels you want — Telegram, WhatsApp, iMessage, Slack, web — and gives them access to your installed skills.

Already installed?

```
which openclaw
openclaw --version
ls ~/.openclaw 2>/dev/null
```

If those return cleanly, skip ahead.

Fresh install

The canonical install lives at **openclaw.ai**. Use whatever the homepage says — install method changes between releases. Historical commands the repo has used:

```
# A — npm (most common)
npx -y @openclaw/cli init

# B — Homebrew tap
brew install openclaw/tap/openclaw

openclaw auth login          # OAuth in browser
openclaw gateway start       # http://localhost:4319
openclaw status              # confirm running
```

Tip. *For an always-on Mac mini setup with LaunchAgent + caffeinate, see docs/mac-mini-setup.md in the HealthClaw repo.*

STEP B · 1B

Stand up an open-source FHIR server

Your records need somewhere to live. Two solid choices — pick one.

Option A — HAPI FHIR · recommended

Reference Java implementation. Easiest to run, no auth on localhost, perfect for first-timers.

```
# Already up?
curl -sf http://localhost:8080/fhir/metadata >/dev/null && echo OK

# Otherwise — Docker is fastest
docker run -d --name hapi-fhir \
  -p 8080:8080 \
  hapiproject/hapi:latest

# After ~30s
curl -s http://localhost:8080/fhir/metadata | head -c 200
```

Option B — Medplum · more features, OAuth-ready

Full Medplum platform: FHIR R4 + auth + dashboard + audit. Heavier setup but gives you a UI to browse resources.

```
git clone https://github.com/medplum/medplum
cd medplum
docker compose -f compose-dev.yaml up -d
open http://localhost:3000   # admin UI
```

Public sandboxes exist *(hapi.fhir.org, r4.smarthealthit.org) but* ***do not put real records there*** *— they're shared, wiped weekly, and visible to anyone.*

STEP B · 2

Connect your real records (privacy-first)

Authenticate with your providers and pull records into the FHIR server you just stood up. Records flow EHR → connector → your machine. The cloud is only the OAuth provider.

Pick a connector

Service	Best for	Auth
HealthEx	US Epic / Cerner / CommonWell users — easiest	OAuth
fhir-skills (Mandel)	DIY SMART-on-FHIR savvy users	SMART
Flexpa	Apps that want a hosted FHIR endpoint	OAuth
Direct SMART	Power users who want zero middleware	OAuth+PKCE
TEFCA IAS / Fasten	Cross-network records via QHIN	Stitch

Recommended path — HealthEx

Lowest friction, cleanest output, first-class HealthClaw integration via the `/export` slash command.

```
# 1. Sign up at healthex.io
# 2. claude.ai → Settings → Integrations → HealthEx → Connect
# 3. Connect your health systems (Epic, Cerner, etc.)
# 4. Stash your token in macOS Keychain so agents find it
security add-generic-password -s healthex -a me -w '<your-token>'
```

STEP B · 3

Install HealthClaw Guardrails

HealthClaw sits between any agent and your FHIR server, enforcing PHI redaction, immutable audit trails, step-up auth on writes, and tenant isolation on every request.

```
git clone https://github.com/aks129/HealthClawGuardrails
cd HealthClawGuardrails
uv sync || pip install -e .

cp .env.example .env
# In .env:
#   STEP_UP_SECRET=$(python3 -c 'import secrets; print(secrets.token_hex(32))')
#   FHIR_UPSTREAM_URL=http://localhost:8080/fhir   # HAPI from Step 1b

docker-compose up -d --build       # full stack (Flask + Redis + MCP)
curl http://localhost:5000/r6/fhir/health
```

Wire the OpenClaw personas

One script seeds Sally (PCP), Mary (pharmacy), Dom (fitness), and Kristy (scheduler) — each persona's `AGENTS.md` already lists every HealthClaw slash command and tool.

```
./scripts/seed_openclaw_workspaces.sh
./scripts/update_agent_prompts.sh
./scripts/bot_commands_install.sh   # installs ~/.healthclaw/commands.py
```

Connect Claude Desktop / Code via MCP

```
{
  "mcpServers": {
    "healthclaw-local": {
      "type": "http",
      "url":  "http://localhost:3001/mcp",
      "headers": { "X-Tenant-ID": "my-health" }
    }
  }
}
```

STEP B · 4

Pull data through HealthClaw

Three ways. Pick whichever fits your moment.

A · One slash command from any bot

DM Sally / Mary / Dom / Kristy on Telegram:

```
/export                                              # HealthEx → redact → file
/import ~/.healthclaw/exports/healthex-<date>.json # bundle → FHIR via guardrails
```

B · Run the scripts directly

```
HEALTHEX_AUTH_TOKEN="$(security find-generic-password -s healthex -w)" \
  python scripts/export_healthex_mcp.py \
  --tenant-id my-health \
  --output exports/healthex-$(date +%Y-%m-%d).json

python scripts/import_healthex.py \
  --bundle-file exports/healthex-$(date +%Y-%m-%d).json \
  --tenant-id my-health \
  --step-up-secret "$STEP_UP_SECRET"
```

C · Conversationally via Claude

If HealthEx is connected to claude.ai, just ask:

```
Pull my complete health history across all categories going back
15 years, fully paginated. Build a de-identified FHIR R4 transaction
bundle with US Core resources and write it to
healthclaw-bundle-<date>.json.
```

What gets redacted. *HumanName → initials, Address → state+country, Identifier → SHA-256 hash, birthDate → year, telecom → ***, narrative div → empty. Clinical codes (ICD-10, SNOMED, LOINC, RxNorm, CVX) are* ***preserved*** *— they're the signal you actually want to analyze.*

STEP B · 5

Verify everything works

A 60-second checklist. Every line should green-light.

Liveness

```
openclaw status                                          # → running
curl -sf http://localhost:8080/fhir/metadata | head -c 80
curl -sf http://localhost:5000/r6/fhir/health
curl -sf http://localhost:3001/mcp/health || true     # MCP orchestrator
```

Records present

```
curl -s "http://localhost:8080/fhir/Patient?_summary=count" \
  | python3 -c "import sys,json; print('Patients:', json.load(sys.stdin).get('total'))"
```

Agent smoke test

In Claude with the `healthclaw-local` MCP server connected:

```
Use the healthclaw-local tools. Get a step-up token, search for all
Patients in tenant my-health, then read one Condition and one
Observation. Confirm responses include _mcp_summary and that PHI
looks redacted.
```

Telegram smoke test

DM any persona:

```
/health
/summary
/conditions
```

Each should return a structured response paraphrased by the LLM. Names appear as initials. Identifiers prefixed with `redacted:sha256:`. AuditEvents recorded for every read.

STEP →

Where to go from here

HealthClaw ships with eight specialised skills that go deeper than this guide. Loaded into Claude Desktop / Code, they trigger on the matching prompts.

`getting-started`	This guide. End-to-end onboarding.
`fhir-r6-guardrails`	The 14 MCP tools and their guarantees
`phi-redaction`	Exactly what gets redacted and why
`fhir-upstream-proxy`	Wire HAPI, Medplum, Epic upstream
`personal-health-records`	Conversational HealthEx data pull workflow
`healthex-export-redacted`	MCP-SDK + in-process PHI redaction (the /export path)
`healthex-export`	Tenant-to-tenant copy inside HealthClaw
`curatr`	Find data quality issues; propose patient-approved fixes
`fasten-connect`	TEFCA IAS · Stitch widget · QHIN cross-network

Browse all skills at **healthclaw.io/skills** · or in the repo at `github.com/aks129/HealthClawGuardrails/tree/main/skills`.

HealthClaw · OpenClaw

Open source · MIT License · A healthclaw.io project