LightClaw: A 7,000-Line Python Alternative to OpenClaw

February 23, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** LightClaw is a lightweight Python AI agent framework that replicates the core OpenClaw experience: persistent memory, scheduled messages, skills from ClawHub, and agent delegation. It's ~7,000 lines of Python vs OpenClaw's 40,000+ lines of TypeScript. Telegram-only. Six LLM providers (OpenAI, Anthropic, Google, xAI, DeepSeek, Z-AI). Runs on a $5/month VPS. MIT licensed, one developer, 12 stars, created February 16, 2026. Interesting proof-of-concept, not battle-tested.

📚 **Related:** How OpenClaw Works · Best OpenClaw Alternatives · OpenClaw Security Guide · OpenClaw Token Optimization

OpenClaw hit 200,000+ GitHub stars because it turns an LLM into a persistent AI assistant that texts you, checks your email, and manages your calendar. It also ships with 40,000+ lines of TypeScript, a complex plugin system, and a security track record that made Cisco call it “a security nightmare.”

The growing response: strip it down. Keep the useful parts, ditch the rest.

LightClaw (GitHub) is the latest entry in this space. Created February 16, 2026, by Othmane Blial. MIT license. ~7,000 lines of Python. The tagline: “The Featherweight Core of OpenClaw.”

Here’s what it does, what it doesn’t, and whether the “lightweight alternative” approach actually works.

What LightClaw Is

A Telegram-based AI assistant with persistent memory, scheduled messaging, skill support, and agent delegation. The pipeline is intentionally simple:

```
Telegram Message -> Memory Recall -> LLM -> HTML Format -> Reply
```

It skips the message bus, plugin registry, and tool orchestration layer. Direct pipeline from input to LLM to response.

Size Comparison

Metric	LightClaw	OpenClaw
Language	Python	TypeScript
Codebase	~7,000 lines	40,000+ lines
Python files	22	N/A
Total files (excl assets)	~36	Hundreds
Dependencies	6 packages	45+ packages
RAM usage	<50 MB (claimed)	Significantly more
GPU required	No (API-based)	No (API-based)
Docker required	No	Optional

The largest files give you a sense of where the complexity lives:

File	Size	Role
core/bot/commands.py	~1,500 lines	All slash commands
core/bot/file_ops.py	~970 lines	Workspace file operations
lightclaw CLI	~930 lines	Entrypoint, onboarding
core/bot/delegation.py	~750 lines	Agent delegation logic
skills.py	~620 lines	ClawHub + local skill manager

Features

LLM Providers (6)

Provider	Models
OpenAI	GPT-5.2, GPT-5.2-mini
Anthropic	Opus 4.5, Sonnet 4.5
Google	Gemini 3 Flash, Gemini 2.5 Flash

Provider	Models
xAI	Grok-4
DeepSeek	deepseek-chat, deepseek-reasoner
Z-AI	GLM-5, GLM-4.7

No local model support. All inference goes through cloud APIs. If you want Ollama integration, you'd need to add it yourself.

Memory

SQLite-backed persistent memory with TF-IDF vector embeddings. When you send a message, the bot recalls related past conversations via cosine similarity and includes them as context.

Commands: `/memory` (stats), `/recall <query>` (search), `/wipe_memory` (clear).

This isn't a full RAG pipeline. It's lightweight keyword-matching with TF-IDF vectors. Works for personal assistant use where you want it to remember your preferences and past conversations. Won't scale to document-level retrieval.

Skills

Integrates with ClawHub's skill registry. You can browse, install, and activate skills per-chat. Also supports local custom skills.

Worth noting: ClawHub had a malware incident where 26% of skills contained vulnerabilities. LightClaw uses the same skill source. Treat all third-party skills as untrusted.

Agent Delegation

LightClaw can hand off tasks to local coding agents: Codex, Claude Code, OpenCode. It spawns the agent, passes your task, and reports workspace deltas (created/updated/deleted files) back through Telegram.

Optional safety mode: `LOCAL_AGENT_SAFETY_MODE=strict` with deny patterns to prevent dangerous commands.

Scheduled Messages

Heartbeat scheduler (periodic check-ins) and cron scheduler (specific times). Same concept as OpenClaw's heartbeats and crons, implemented in a few hundred lines of Python backed by a JSON store.

Context Management

Auto-summarization when conversations exceed 20 messages or 75% of the context window. Emergency compression (drop oldest 50%) if context-too-long errors hit. Token estimation uses a ~2.5 chars/token heuristic.

Personality System

Markdown-based personality files in `.lightclaw/personality/`:

- `IDENTITY.md` – who the bot is
- `SOUL.md` – behavioral guidelines
- `USER.md` – info about the user
- `HEARTBEAT.md` – heartbeat prompt

Voice

Groq Whisper transcription for voice messages (optional, requires Groq API key).

Setup

One-Command

```
git clone https://github.com/OthmaneBlial/lightclaw.git
cd lightclaw
bash setup.sh
```

The setup wizard walks through:

1. Choose LLM provider
2. Enter API key
3. Create Telegram bot via @BotFather
4. Optional voice transcription setup
5. Auto-start

Manual

```
git clone https://github.com/OthmaneBlial/lightclaw.git
cd lightclaw
pip install -r requirements.txt
./lightclaw onboard    # Creates .env, .lightclaw/ directory, personality files
# Edit .env with API keys
./lightclaw run
```

There's also a `./lightclaw chat` mode for local terminal interaction without Telegram.

Requirements

- Python 3.10+
- A Telegram bot token (free from @BotFather)
- An API key for at least one LLM provider
- No GPU, no Docker, no build steps

Architecture

Clean flat module structure with a mixin composition pattern:

```
lightclaw/
  lightclaw           # CLI entrypoint
  config.py           # .env loading
  memory.py           # SQLite + TF-IDF
  providers.py        # Unified LLM client
  skills.py           # ClawHub + local skills
  core/
    app.py            # Runtime startup
    personality.py    # Prompt builder
    voice.py          # Whisper transcription
    bot/
      base.py         # Shared state
      commands.py     # Slash commands
      handlers.py     # Message handling
      file_ops.py     # File operations
      delegation.py   # Agent delegation
      context.py      # Summarization
      messaging.py    # Telegram send/chunk
```

The bot class composes base, commands, handlers, file_ops, delegation, context, and messaging as mixins. No abstraction layers, no dependency injection, no message bus. If you want to understand what the bot does on any given message, you can trace from `handlers.py` to the response in a few hundred lines.

What It Can't Do (That OpenClaw Can)

Feature	OpenClaw	LightClaw
Multi-channel (WhatsApp, Discord, Slack, SMS)	Yes	Telegram only
Tool orchestration / function calling	Yes	No
Browser automation / computer use	Yes	No
MCP (Model Context Protocol)	Yes	No
Web search	Built-in	No
Vision/multimodal	Yes	Photo handler exists but no vision model
Webhook-based messaging	Yes	Polling only
Sandboxed execution	Container-based	Direct on host
Multi-user at scale	Yes	Personal use only
Community / ecosystem	200K+ stars, active	12 stars, 1 contributor

---

Security

LightClaw's README opens with a security disclaimer: "not a 'safe by default' security product."

The argument for LightClaw's security model is size. You can read the entire codebase in an afternoon. With OpenClaw, auditing the full codebase is a multi-week project.

Mitigations available:

- `TELEGRAM_ALLOWED_USERS` – allowlist specific Telegram user IDs
- `LOCAL_AGENT_SAFETY_MODE=strict` – deny patterns for delegation

- Recommendation to run in a VM or container with least privilege

The argument against: one developer, no security audit, no bug bounty, no dedicated security team, seven days of history.

Honest Assessment

The architecture is clean and readable. Six provider support with a unified interface is well-implemented. The memory system (SQLite + TF-IDF) is a reasonable approach for personal assistant recall. Setup takes about five minutes. The mixin pattern keeps the code navigable despite the 7,000-line size.

On the other hand, Telegram-only limits the audience significantly. No local model support means you're paying for every API call. The ClawHub skill integration inherits the same security concerns that plague OpenClaw's skill ecosystem. No web search or vision means it's a chat-only assistant.

The bigger question is whether the "lightweight alternative" category has staying power, or if these projects appear, get a few dozen stars, and go quiet. LightClaw is seven days old. The commit history shows intense activity from Feb 16-20, then silence. That's not unusual for a new project, but it's too early to know if this becomes a maintained tool or another GitHub archive.

The Lightweight Landscape

LightClaw isn't alone. Other entries in this space:

Project	Language	Size	Distinguishing Feature
LightClaw	Python	~7,000 lines	6 providers, ClawHub skills
NanoClaw	Python	~500 lines core	Container-based security
Nanobot	Python	~4,000 lines	Research-ready, multi-agent
PicoClaw	Go	<10 MB RAM	Boots in <1 second
ZeroClaw	Rust	3.4 MB binary	<5 MB RAM

Who Should Use This

LightClaw makes sense if you want a Telegram-based AI assistant you can actually read and understand. You don't need multi-channel support, you're comfortable with cloud API calls, and you want something running in five minutes without Docker.

Skip it if you need WhatsApp, Discord, or Slack. Same if you want local model inference or production-grade reliability.

Consider forking it if the architecture appeals to you but the feature set doesn't. LightClaw's README explicitly encourages forking. The module structure makes it straightforward to add a new messaging transport or swap in a local inference backend.

Bottom Line

LightClaw proves that the core OpenClaw experience (persistent AI assistant with memory, scheduling, and skills) fits in ~7,000 lines of Python. The code is readable, the setup takes five minutes, and you can audit the whole thing in an afternoon.

It's also one week old, has one developer, and supports one messaging platform. The "lightweight OpenClaw alternative" category is getting crowded, and it's too early to know which of these projects will still be maintained in three months.

If you want to run OpenClaw today, the real thing has the ecosystem. If you want something you can read, understand, and customize without spending a week in the codebase, LightClaw is worth a look. Just set your expectations accordingly.

GitHub: OthmaneBlial/lightclaw

📚 **OpenClaw:** How OpenClaw Works · Setup Guide · Token Optimization · Security Guide

📚 **Alternatives:** Best OpenClaw Alternatives · mycoSwarm vs Exo vs Petals

---

Source: https://insiderllm.com/guides/lightclaw-lightweight-openclaw-alternative/