OpenClaw Setup Guide: Run a Local AI Agent

February 2, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Install Node.js 22+, run `npx openclaw@latest`, and follow the onboarding wizard. It walks you through connecting an LLM provider (Anthropic, OpenAI, or local via Ollama), linking a messaging platform (WhatsApp via QR code scan, Telegram via BotFather token), and enabling skills. The gateway itself needs only 2-4 GB RAM. Running local models through Ollama requires a GPU — an 8B model at Q4 needs ~5 GB VRAM. For external access, set up Cloudflare Tunnel instead of opening ports. Read our security guide before connecting any real accounts — use throwaway credentials for testing.

📚 **More on this topic:** OpenClaw Security Guide · Best Models for OpenClaw · OpenClaw vs Commercial Agents · Ollama Troubleshooting · Planning Tool

Before you start: Read our OpenClaw Security Guide. OpenClaw is powerful but carries real security risks — exposed instances, prompt injection vulnerabilities, and an immature plugin ecosystem. This setup guide shows you how to get it running. The security guide tells you how to not get burned. Read both.

OpenClaw is an open-source AI agent that runs on your hardware and connects to the messaging apps you already use. You message it on WhatsApp or Telegram, and it actually does things — triages your inbox, drafts emails, books flights, writes code, manages your calendar. Over 145,000 GitHub stars. It's the fastest-growing open-source project in GitHub history.

This guide gets you from zero to a working OpenClaw instance, step by step.

What You're Setting Up

OpenClaw has three layers:

Layer	What It Does	Your Options
Gateway	Routes messages between you and the AI	Runs on your machine (Node.js or Docker)
LLM Backend	The intelligence — processes your requests	Anthropic Claude, OpenAI GPT-4, or **Ollama** (local, free)

Layer	What It Does	Your Options
Skills	Gives the agent hands — browser, shell, files, calendar	50+ bundled, more via ClawHub marketplace

The gateway is lightweight. The LLM backend is where your hardware matters — especially if you want to run models locally through Ollama instead of paying for API access.

Hardware Requirements

Gateway Only (Cloud LLM via API)

If you're using Anthropic or OpenAI as the backend, the gateway itself is lightweight:

Spec	Minimum	Recommended
RAM	2 GB	4 GB
CPU	1 core	2 cores
Storage	10 GB	20 GB
GPU	Not needed	Not needed

A Mac Mini, Raspberry Pi 4, old laptop, or any cheap VPS works. The gateway just routes messages — the heavy lifting happens at Anthropic's or OpenAI's servers.

Cost: OpenClaw itself is free (MIT license). You pay for API tokens — typically $10-150/month depending on how much you use it. Claude Sonnet is the most cost-effective option for most people.

Gateway + Local LLM (via Ollama)

If you want everything running on your hardware with no API costs, you need enough resources for both the gateway and the model:

Spec	Minimum	Recommended
RAM	16 GB	32 GB
GPU VRAM	6 GB (8B model, Q4)	12-16 GB (for comfortable 8B or 14B)
CPU	4 cores	8 cores

Spec	Minimum	Recommended
Storage	30 GB	50 GB (models are large)

OpenClaw's documentation notes that models need at least 64,000 token context length to reliably complete multi-step tasks. That means you need models that support long context and enough VRAM to actually use it. See our VRAM requirements guide for specifics by model.

Cost: $0 ongoing. Just your electricity bill.

The honest tradeoff: Local models through Ollama are free and private, but noticeably less capable than Claude for complex agent tasks. Cloud APIs cost money but produce better results. Most people start with a cloud API and switch to local once they understand what works.

Installation

Prerequisites

- **Node.js 22 or higher** — check with `node --version`
- **macOS or Linux** — Windows users need WSL2
- **An LLM provider** — Anthropic API key, OpenAI API key, or Ollama installed locally

Method 1: npm (Recommended for Getting Started)

```
npx openclaw@latest
```

That's it. This downloads and launches the onboarding wizard, which walks you through everything: provider setup, messaging channels, skills configuration.

Method 2: Docker (Recommended for Security)

Docker isolates OpenClaw from your host system. This is the better choice if you plan to run it long-term. See our security guide for why isolation matters.

```
git clone https://github.com/openclaw/openclaw.git
cd openclaw
./docker-setup.sh
```

The setup script:

1. Builds the OpenClaw Docker image
2. Runs the onboarding wizard inside a container
3. Generates a gateway token for the Control UI
4. Creates config directories (`~/.openclaw` for config, `~/openclaw/workspace` for agent files)
5. Starts the gateway via Docker Compose

Docker volumes:

Mount	Purpose
`~/.openclaw`	Configuration, memory, API keys
`~/openclaw/workspace`	Files the agent can directly access

For hardened Docker setups (non-root user, read-only filesystem, dropped capabilities), see the Docker hardening section in our security guide.

Method 3: One-Click Cloud Deploy

If you don't want to manage hardware at all:

- **DigitalOcean** — 1-click deploy with pre-installed dependencies
- **Hostinger** — VPS template with Docker
- **Vultr** — deployment guide available in their docs

These start at $5-10/month and handle the infrastructure. You still need an LLM API key.

The Onboarding Wizard

Whether you use npm or Docker, the onboarding wizard is the same. It walks through four steps:

Step 1: LLM Provider

Choose your backend:

Provider	Cost	Quality	Privacy
Anthropic (Claude)	~$3/M input tokens	Best for agent tasks	Queries go to Anthropic
OpenAI (GPT-4)	~$2.50/M input tokens	Strong	Queries go to OpenAI
Google (Gemini)	Varies	Good	Queries go to Google
OpenRouter	Varies	Access to many models	Cloud gateway
Ollama (local)	Free	Depends on model/hardware	Fully private

Enter your API key when prompted. For Ollama, see the dedicated section below.

Step 2: Messaging Channel

Pick how you'll talk to OpenClaw. You can configure multiple channels.

Step 3: Channel Allowlist

Choose who can message the bot. By default, OpenClaw uses a pairing system — unknown senders get a pairing code and their messages aren't processed until you approve them.

Step 4: Skills

Skills extend what OpenClaw can do. The wizard asks if you want to configure them now. **Skip this on first run** — get basic chat working first, then add skills once you're comfortable.

After the wizard completes, you get a tokenized URL for the Control UI. Open it in your browser to monitor conversations, manage settings, and view agent activity.

Connecting WhatsApp

WhatsApp is the most popular channel for OpenClaw. Setup takes about 2 minutes.

1. Select **WhatsApp** during the onboarding wizard (or add it later in the Control UI)
2. A QR code appears in your terminal or browser
3. On your phone: open **WhatsApp → Settings → Linked Devices → Link a Device**

4. Scan the QR code

That's it. OpenClaw now receives your WhatsApp messages.

Testing tip: Message yourself first. WhatsApp has a "Message yourself" feature — use it to test without spamming contacts. By default, OpenClaw marks incoming messages as read (blue ticks) once accepted.

Important: WhatsApp on Bun runtime is unreliable. Use Node.js for the gateway if you're connecting WhatsApp.

Security warning: Every WhatsApp message OpenClaw reads is potential prompt injection surface. Do not connect your primary WhatsApp account during initial testing. Use a separate number with a throwaway SIM or a VoIP number. See our security guide for details on prompt injection risks.

Connecting Telegram

Telegram is simpler to set up and better for testing because you create a dedicated bot account.

1. Open Telegram and message **@BotFather**
2. Send `/newbot` and follow the prompts to create a bot
3. BotFather gives you a **bot token** — copy it
4. Enter the token during OpenClaw's onboarding wizard (or add it in settings)
5. OpenClaw sends you a pairing code via Telegram — enter it to link your account

Your Telegram bot is now connected. Message the bot directly to interact with OpenClaw.

Telegram is better for testing because the bot has its own account — your personal messages stay separate. For WhatsApp, the bot runs on your number, which blurs the line.

Connecting Ollama (Local LLM Backend)

This is the setup that makes OpenClaw fully local — no API costs, no data leaving your machine.

Prerequisites

1. **Ollama installed and running** — see our getting started guide

2. **A model pulled** with sufficient context length (64K+ recommended)

```
# Pull a recommended model
ollama pull qwen3:8b

# Or for better agent performance if you have the VRAM
ollama pull qwen3:14b
```

Method 1: Auto-Discovery (Simplest)

Set one environment variable and OpenClaw finds your Ollama models automatically:

```
export OLLAMA_API_KEY="ollama-local"
```

When this is set and you haven't defined explicit Ollama configuration, OpenClaw discovers all models from your local Ollama instance at `http://127.0.0.1:11434`.

Method 2: Explicit Configuration

For more control, edit `~/.openclaw/openclaw.json`:

```
{
  "models": {
    "providers": {
      "ollama": {
        "baseUrl": "http://127.0.0.1:11434/v1",
        "apiKey": "ollama-local",
        "api": "openai-completions",
        "models": [
          { "id": "qwen3:8b" }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "ollama/qwen3:8b"
      }
    }
```

```
  }
}
```

Method 3: Ollama Launch Command

Ollama's February 2026 update added a direct integration:

```
ollama launch openclaw
```

This auto-configures the connection between OpenClaw and your local models, handles model discovery, and starts the gateway.

Recommended Ollama Settings

Set these environment variables for the Ollama server before starting it:

```
export OLLAMA_CONTEXT_LENGTH=16384
export OLLAMA_FLASH_ATTENTION=1
export OLLAMA_NEW_ENGINE=1
```

- `OLLAMA_CONTEXT_LENGTH` — agents need long context for multi-step tasks. 16K is a reasonable starting point; increase to 32K or 64K if you have the VRAM
- `OLLAMA_FLASH_ATTENTION` — reduces VRAM usage, speeds up inference
- `OLLAMA_NEW_ENGINE` — uses Ollama's newer inference engine

Which Model for OpenClaw?

Model	VRAM (Q4)	Agent Quality	Speed
Qwen 3 8B	~5 GB	Decent — handles basic tasks	Fast
Qwen 3 14B	~9 GB	Good — reliable for most workflows	Moderate
Llama 3.1 8B	~5 GB	Decent — large fine-tune ecosystem	Fast
Qwen 2.5 Coder 14B	~9 GB	Strong for coding-focused tasks	Moderate
Qwen 3 32B	~20 GB	Very good — closest to cloud quality	Slower

The honest assessment: Local 7-8B models work for simple tasks (quick questions, basic summaries, short commands) but struggle with complex multi-step agent workflows. If you're running tasks like "triage my inbox, draft responses, and schedule follow-ups," you'll get significantly better results from Claude or GPT-4 via API. A local 14B-32B model bridges some of that gap if you have the VRAM.

For model details, see our Llama 3 guide and best models for chat guide.

Skills Setup

Skills are what make OpenClaw an agent instead of a chatbot. They give it the ability to take actions — browse the web, run shell commands, manage files, access calendars.

Listing Available Skills

```
openclaw skills list
```

Browsing and Installing from ClawHub

```
npx clawdhub
```

This opens an interactive browser for the community skill marketplace.

Which Skills to Enable

Start minimal. Every enabled skill is additional attack surface.

Skill	What It Does	Risk Level	Enable First?
Web search	Searches the internet	Medium — outbound HTTP	Yes, useful
File read/write	Access files in workspace	Medium — agent can read/ modify files	Yes, within workspace only
Shell exec	Run terminal commands	**High** — full command execution	Only if you need it

Skill	What It Does	Risk Level	Enable First?
Browser automation	Control a web browser	**High** — navigates arbitrary URLs	Only if you need it
Calendar	Read/write calendar events	Medium — requires OAuth	After testing
Email	Read/send email	**High** — prompt injection via inbound email	After reading the security guide

Start with web search and file access. Add shell and browser only when you have a specific use case that requires them. Disable everything you're not actively using.

ClawHub warning: The community marketplace has had security issues — a researcher demonstrated that malicious skills can be uploaded and downloaded by thousands of users with no moderation. An analysis of 31,000 skills found 26% contained at least one vulnerability. Stick to bundled skills and well-known community skills. See our security guide for details.

Cloudflare Tunnel (Secure External Access)

If you want to access OpenClaw from outside your home network — or if messaging platforms need webhook callbacks to reach your instance — you need a way to expose it to the internet.

Do not open ports on your router. An AI agent with shell access sitting behind an open port is inviting disaster.

Use Cloudflare Tunnel instead. It creates an encrypted connection from your machine to Cloudflare's network — no inbound ports required.

Setup

1. **Create a free Cloudflare account** and add a domain (or use a free `.cfargotunnel.com` subdomain)
2. **Install cloudflared:**

```
# macOS
brew install cloudflared

# Linux (Debian/Ubuntu)
```

```
curl -L https://github.com/cloudflare/cloudflared/releases/latest/download/cloudflared-linux-amd
sudo dpkg -i cloudflared.deb
```

1. **Authenticate:**

```
cloudflared tunnel login
```

1. **Create a tunnel:**

```
cloudflared tunnel create openclaw
```

1. **Configure the tunnel** — create `~/.cloudflared/config.yml`:

```
tunnel: <your-tunnel-id>
credentials-file: /home/<user>/.cloudflared/<tunnel-id>.json

ingress:
  - hostname: openclaw.yourdomain.com
    service: http://localhost:3000
  - service: http_status:404
```

1. **Route DNS:**

```
cloudflared tunnel route dns openclaw openclaw.yourdomain.com
```

1. **Start the tunnel:**

```
cloudflared tunnel run openclaw
```

Your OpenClaw instance is now accessible at `https://openclaw.yourdomain.com` through Cloudflare's network, with no ports open on your router.

Alternative: Cloudflare Workers (Moltworker)

For full isolation, you can run OpenClaw inside Cloudflare's infrastructure using Moltworker. This moves the gateway off your hardware entirely — no local RCE risk. Requires a Workers Paid plan ($5/month) plus an API key for your LLM provider. See the Moltworker repo for setup instructions.

First Run Walkthrough

Once installation and configuration are complete:

1. **Start OpenClaw** (if not already running):

```
# npm method
npx openclaw@latest

# Docker method
cd openclaw && docker compose up -d
```

2. **Open the Control UI** — use the tokenized URL from the setup wizard. This is your dashboard for monitoring conversations and managing settings.
3. **Send a test message** on your configured channel:
   - WhatsApp: message yourself or send "hello" to the bot
   - Telegram: message your bot directly
4. **Try a simple task** to verify everything works:
   - "What time is it?"
   - "Summarize this article: [paste a URL]" (requires web search skill)
   - "List the files in my workspace" (requires file read skill)
5. **Check the Control UI** to see the conversation, token usage, and any errors.

If Things Don't Work

Bot doesn't respond on WhatsApp:

- Check that the gateway is running (`docker ps` or check your terminal)

- Re-scan the QR code — WhatsApp linked devices can disconnect
- Make sure you're using Node.js, not Bun, for WhatsApp

Bot doesn't respond on Telegram:

- Verify your bot token is correct
- Make sure you completed the pairing step
- Check that the gateway has internet access (for Telegram API)

Ollama model not found:

- Run `ollama list` to verify the model is pulled
- Check that `OLLAMA_API_KEY` is set to `"ollama-local"`
- Verify Ollama is running on port 11434: `curl http://localhost:11434`

Slow responses with local models:

- Check if the model fits in VRAM — partial offload to CPU is 10-30x slower
- Run `ollama ps` to see GPU utilization
- Try a smaller model or lower quantization
- See our Ollama troubleshooting guide for more

What to Do Next

Once you have basic chat working:

1. **Test with throwaway accounts** before connecting anything real
2. **Read the security guide** — understand the risks of each skill before enabling it
3. **Try simple automations** — "remind me to check email at 9am," "summarize this PDF," "what's on my calendar today"
4. **Gradually add skills** as you need them, one at a time
5. **Set up Cloudflare Tunnel** if you need remote access
6. **Consider Docker hardening** if you plan to run it long-term

What NOT to Do

- Don't connect your primary email, bank accounts, or work systems during testing
- Don't install skills from ClawHub without reviewing the source code

- Don't expose your instance to the internet without Cloudflare Tunnel
- Don't run OpenClaw on the same machine where you store sensitive files
- Don't assume "local" means "secure" — see our security guide

---

Bottom Line

OpenClaw is the most capable open-source AI agent available today. The setup is straightforward — `npx openclaw@latest` gets you through a wizard that handles everything. Connecting WhatsApp takes 2 minutes. Connecting Ollama for local inference is a single environment variable.

The hard part isn't installation. It's making informed decisions about what you connect, what skills you enable, and how much you trust an agent with shell access to your machine. The capabilities are real. The risks are also real.

Start small: install, connect Telegram (safer than WhatsApp for testing), try basic chat with a cloud API. If that works, connect Ollama for local inference. Add skills one at a time. Read the security guide before connecting anything you care about losing.

For hardware recommendations by budget, see our GPU buying guide and VRAM requirements guide.

Get notified when we publish new guides.

Subscribe — free, no spam
