OpenClaw

Safe Setup Guide

A battle-tested guide to running your personal AI assistant

macOS · Docker · Telegram

Eliran Keren | February 2026

Based on real installation experience — not theoretical documentation

ask me anything on https://www.linkedin.com/in/elirankeren

What is OpenClaw?

OpenClaw is an open-source AI personal assistant that connects large language models to your messaging apps (Telegram, WhatsApp, Discord, Slack, and more). It can browse the web, read and write files, run shell commands, and send messages on your behalf. It runs 24/7 on your own machine.

It exploded in popularity in January 2026, going from 9,000 to over 140,000 GitHub stars in days. Created by Peter Steinberger, it was originally called Clawdbot, briefly Moltbot, before settling on OpenClaw.

The Easiest Way to Get Started (No Technical Background Needed)

This guide contains terminal commands, Docker configurations, and security settings. If that sounds intimidating — **you do not need to understand any of it to get this running.** Here is how.

Option A: Use Claude's Cowork Mode (Recommended)

This entire guide was created through a conversation with Claude in Cowork mode. Claude wrote the scripts, debugged the errors, and walked a non-technical user through every step. You can do exactly the same thing.

What is Cowork? Cowork is a feature of the Claude desktop app that lets Claude run code, create files, and execute terminal commands on your computer — inside a safe sandbox. Think of it as having a technical friend sitting next to you, except they can actually type commands for you.

How to do it:

1. Download the Claude desktop app (claude.ai/download)
2. Open Claude and switch to Cowork mode (look for it in the mode selector)
3. Select a folder on your computer for Claude to work in
4. Share this guide with Claude (drag the file into the chat or paste the link)
5. Say: *"Help me set up OpenClaw safely on my Mac, following this guide step by step."*
6. Claude will create the scripts, run the commands, and ask for your input only when needed (passwords, API keys, Telegram token)

> Tip: Claude in Cowork mode can write files directly to your computer, run terminal commands, and debug errors in real time. You just need to approve each step. It is like pair-programming with someone who already read the entire guide.

What You Will Need to Do Yourself

Even with Claude guiding you, there are a few things only you can do:

Action	Why You Need to Do It
Enter your Mac password	Docker and system-level changes require admin access. Claude cannot type passwords for you.

Get an Anthropic API key	Go to console.anthropic.com, create an account, and generate a key. This is the brain behind your assistant. Costs around $5–20/month depending on usage.
Create a Telegram bot	Open Telegram, message @BotFather, send /newbot, pick a name. Takes 2 minutes.
Scan a QR code (WhatsApp)	Only if you connect WhatsApp. You scan it with your phone like linking a new device.
Approve pairing in Telegram	Send a message to your bot, get a code, tell Claude to run the approve command.

Everything else — creating directories, writing configuration files, pulling Docker images, hardening security settings — Claude handles for you.

Option B: Copy-Paste the Commands Yourself

If you prefer not to use Cowork, you can follow this guide manually. Open Terminal on your Mac (search for "Terminal" in Spotlight) and copy-paste each command from this guide. The guide is written in execution order — just go top to bottom.

Important: If a command fails, do not panic. Read the error message. Most errors in this guide have a "GOTCHA" box right next to them explaining what went wrong and how to fix it. These gotchas are real errors we encountered during our installation.

Option C: Use the Setup Script

We also created an automated setup script (openclaw-setup.sh) that walks you through the entire process interactively. It pauses at each decision point and asks for your input. However, we found that some interactive prompts can be finicky inside scripts. If the script gets stuck, fall back to Option A or B.

Security Without Technical Knowledge

You do not need to understand Docker, containers, or networking to be safe. Here is what matters in plain language:

Concept	What It Means for You
Docker container	OpenClaw runs in a sealed box on your computer. It cannot see your files, photos, passwords, or anything else unless you explicitly give it access.
Read-only filesystem	If a hacker tricks the AI, it cannot install anything permanent on your machine. Restarting the container wipes any changes.
Loopback-only	Nobody outside your computer can reach OpenClaw. Not your neighbor, not someone on your Wi-Fi, not anyone on the internet.

Manual confirmation	Before OpenClaw sends a message, deletes a file, or takes any action, it asks you first. You say yes or no. This is your safety net.
Frontier model	Using Claude (a top-tier AI) instead of a small local model means the AI is much harder to trick. Think of it as hiring a senior security guard instead of a trainee.

> **The single most important rule: NEVER turn off manual confirmation. As long as you review and approve every action the AI takes, you are in control.**

How This Guide Was Made

This guide is not theoretical. It was created by a real user (with the help of Claude in Cowork mode) who went through the entire setup from scratch on a MacBook with 16GB RAM. Every error in the "Known Gotchas" section is an error we actually hit. Every command in this guide is a command we actually ran. The security recommendations come from published research by Cisco, Adversa.ai, and the OpenClaw security docs.

The total setup took about 45 minutes, including debugging. If you use Cowork mode, expect a similar timeframe.

Why This Guide Exists

OpenClaw is powerful but has serious security concerns. As of February 2026:

- CVE-2026-25253 (CVSS 8.8): One-click remote code execution via WebSocket hijacking
- CVE-2026-22708: Prompt injection via hidden CSS-invisible instructions on web pages
- ~20% of community skills on ClawHub contained malware (Cisco research)
- No bug bounty program, no dedicated security team
- Meta, Belgium, South Korea, and multiple enterprises have banned or restricted it

This guide walks you through setting up OpenClaw with maximum security isolation on macOS. It is based on a real installation — every command was tested and every error was encountered and fixed.

> **WARNING: Do not follow the default OpenClaw install instructions. They run the agent with your full user privileges and no isolation. This guide uses Docker containerization as the primary security boundary.**

What You Need

Requirement	Details
Operating System	macOS (Apple Silicon or Intel)
RAM	16GB minimum (32GB+ recommended for local models)
Disk Space	~10GB (Docker image + model)
Homebrew	Package manager for macOS (brew.sh)
Anthropic API Key	From console.anthropic.com (~$5–20/month usage)
Telegram Account	For creating a bot via @BotFather

Estimated setup time: 30–45 minutes (depending on download speeds).

Security Architecture

The core principle: never run OpenClaw directly on your Mac. Everything runs inside a hardened Docker container.

Security Layer	What It Does
Docker container	Isolates OpenClaw from your Mac. It can only access mounted directories.
Read-only filesystem	Agent cannot modify its own runtime or install malware persistently.

Non-root user	Container runs as 'node' (uid 1000), limiting blast radius.
Dropped capabilities	All Linux capabilities dropped except NET_BIND_SERVICE.
Loopback-only binding	Gateway on 127.0.0.1 only — not reachable from your LAN.
Resource limits	Capped at 2GB RAM and 2 CPU cores.
Manual confirmation	Every write, delete, and send requires your explicit 'yes'.
Frontier model	Claude Opus resists prompt injection far better than local models.

Phase 1: Install Prerequisites

Install Docker Desktop

```
brew install --cask docker
```

After installation, open Docker Desktop from Applications and wait for it to fully start. The whale icon should appear in your menu bar.

> **IMPORTANT: If you have Apple Silicon (M1/M2/M3/M4), make sure Docker Desktop installs the ARM version. If it shows a dialog saying 'You are running Apple Silicon', quit and reinstall the ARM version from desktop.docker.com/mac/main/arm64/Docker.dmg**

Install Ollama (Optional)

Ollama runs local AI models. This is optional — the cloud model (Claude) handles everything. Only install if you want to experiment with local models offline.

```
brew install ollama
ollama pull qwen3:8b
```

> Tip: With 16GB RAM, qwen3:8b is the largest model that runs comfortably. However, local models are significantly more vulnerable to prompt injection. Use the cloud model (Claude) for all real tasks.

Get Your Anthropic API Key

1. Go to console.anthropic.com
2. Create an account or sign in
3. Go to API Keys and create a new key
4. Copy it somewhere safe — you will only see it once

> **NEVER paste your API key in a chat, email, GitHub issue, or shared document. Treat it like a password.**

Phase 2: Create the Project Directory

```
mkdir -p ~/openclaw-safe/{config,workspace,logs}
chmod 700 ~/openclaw-safe
chmod 700 ~/openclaw-safe/config
chmod 700 ~/openclaw-safe/workspace
chmod 700 ~/openclaw-safe/logs
```

A Note on macOS User Isolation

We originally planned to run OpenClaw under a separate macOS user account for extra isolation. In practice, Docker Desktop on macOS shares files through your user context and cannot access directories owned by another user. This makes macOS user isolation incompatible with Docker Desktop.

Docker containerization is your primary security boundary and provides strong isolation on its own. The separate macOS user was a bonus layer that turned out to be impractical.

Phase 3: Docker Configuration

Create docker-compose.yml

Create this file at ~/openclaw-safe/docker-compose.yml:

```
services:
  openclaw:
    image: ghcr.io/openclaw/openclaw:latest
    container_name: openclaw-safe
    restart: unless-stopped
    user: "1000:1000"
    read_only: true
    security_opt:
      - no-new-privileges:true
    cap_drop:
      - ALL
    cap_add:
      - NET_BIND_SERVICE
    tmpfs:
      - /tmp:size=100M
      - /run:size=50M
    env_file:
      - .env
    environment:
      - NODE_ENV=production
      - OPENCLAW_BIND=127.0.0.1
      - OPENCLAW_LOG_LEVEL=info
      - OPENCLAW_SANDBOX=true
      - OPENCLAW_MANUAL_CONFIRM=true
      - OLLAMA_BASE_URL=http://host.docker.internal:11434
    volumes:
      - ./config:/home/node/.openclaw:rw
      - ./workspace:/home/node/workspace:rw
      - ./logs:/home/node/logs:rw
    ports:
      - "127.0.0.1:18789:18789"
```

```
    - "127.0.0.1:18790:18790"
  deploy:
    resources:
      limits:
        memory: 2G
        cpus: "2.0"
  logging:
    driver: json-file
    options:
      max-size: 10m
      max-file: "3"
```

> **GOTCHA: The Docker image is ghcr.io/openclaw/openclaw (GitHub Container Registry), NOT openclaw/openclaw (Docker Hub). The Docker Hub image does not exist.**
>
> **GOTCHA: The home directory inside the container is /home/node, NOT /home/openclaw. The container runs as the 'node' user.**
>
> **GOTCHA: The gateway port is 18789, NOT 3200. The bridge port is 18790.**

Create the .env File

Create ~/openclaw-safe/.env with your API key:

```
ANTHROPIC_API_KEY=sk-ant-your-key-here
OPENAI_API_KEY=
```

Then lock it down:

```
chmod 600 ~/openclaw-safe/.env
```

Phase 4: Start OpenClaw

Pull and Start

```
cd ~/openclaw-safe
docker compose pull
docker compose up -d
```

Verify the container is running:

```
docker ps | grep openclaw-safe
```

You should see the container with status 'Up' and ports 127.0.0.1:18789 and 127.0.0.1:18790.

Run the Onboarding Wizard

```
docker exec -it openclaw-safe node /app/openclaw.mjs onboard
```

> **GOTCHA: The 'openclaw' command is NOT in the container's PATH. Always use 'node /app/openclaw.mjs' instead.**

During onboarding, select:

1. QuickStart mode
2. Anthropic as the model provider
3. "Anthropic API key" (not the setup-token option)
4. Paste your API key when prompted

> **GOTCHA: The interactive wizard can break inside 'docker exec'. If the terminal gets stuck printing repeated characters (e.g., 'yyyyyy'), hit Ctrl+C and restart the container with 'docker restart openclaw-safe'. Your previous selections are usually saved.**

Phase 5: Connect Telegram

Create a Telegram Bot

1. Open Telegram on your phone
2. Search for @BotFather
3. Send /newbot and follow the prompts to name your bot
4. Copy the bot token BotFather gives you
5. Send /setjoingroups to BotFather, select your bot, choose Disable

Connect to OpenClaw

The Telegram channel is configured during onboarding. If you need to add it manually:

```
docker exec -it openclaw-safe node /app/openclaw.mjs \
  channels add --channel telegram --token YOUR_BOT_TOKEN
```

> **NOTE: If 'channels add' returns 'Unknown channel: telegram', use the onboarding wizard instead (openclaw onboard). The channel setup is integrated into the onboarding flow.**

Pair Your Account

After connecting Telegram, send any message to your bot. It will respond with a pairing code. Approve it:

```
docker exec openclaw-safe node /app/openclaw.mjs \
  pairing approve telegram YOUR_CODE
```

> Tip: Pairing codes expire quickly. Run the approve command immediately after receiving the code. If it says 'No pending pairing request found', send another message to the bot to get a fresh code.

Phase 6: Skill Hygiene

> **CRITICAL: Do NOT install third-party skills from ClawHub. Cisco found ~20% of community skills performed data exfiltration or prompt injection. Only use official skills from the openclaw/ organization.**

List installed skills:

```
docker exec openclaw-safe node /app/openclaw.mjs skills list
```

Disable anything you do not need:

```
docker exec openclaw-safe node /app/openclaw.mjs skills disable SKILL_NAME
```

Phase 7: Security Verification

Run these checks after setup to confirm everything is locked down:

Verification Checklist

Check	Command / Expected Result
Container runs as non-root	docker exec openclaw-safe whoami → node
Filesystem is read-only	docker exec openclaw-safe touch /test → Read-only file system
.env is owner-only	ls -la ~/openclaw-safe/.env → -rw-------
Gateway is loopback-only	docker port openclaw-safe → 127.0.0.1:18789
Telegram responds	Send a message to your bot on Telegram
No third-party skills	openclaw skills list → only official skills

Threat Mitigation Summary

Threat	How We Mitigate It
CVE-2026-25253 (WebSocket RCE)	Gateway bound to 127.0.0.1 only, not reachable from LAN
Malicious skill runs shell command	Read-only filesystem + dropped capabilities + non-root user
Prompt injection triggers file deletion	OPENCLAW_MANUAL_CONFIRM=true — you approve every destructive action
Agent escapes container	Docker isolation + no-new-privileges + capped resources
Data exfiltration via network	Container has no access to your Mac except /workspace
Malicious community skills	Zero third-party skills installed — official only

Daily Operations

Essential Commands

Action	Command
Check status	docker exec openclaw-safe node /app/openclaw.mjs status
View logs	docker logs -f openclaw-safe
Restart	docker restart openclaw-safe
Stop	cd ~/openclaw-safe && docker compose down
Start	cd ~/openclaw-safe && docker compose up -d
Update	cd ~/openclaw-safe && docker compose pull && docker compose up -d
Backup config	cp -r ~/openclaw-safe/config ~/openclaw-safe/config-backup-$(date +%Y%m%d)

Keep It Updated

New CVEs are found regularly. Update at least weekly:

```
cd ~/openclaw-safe
docker compose pull
docker compose up -d
```

Manual Confirmation

With OPENCLAW_MANUAL_CONFIRM=true, every action that writes, deletes, or sends a message requires your explicit approval in the chat. This is your last line of defense against prompt injection attacks.

> **Do NOT disable manual confirmation. It is the single most important security feature when the agent interacts with the real world on your behalf.**

Risk Summary

Risk	Area	Notes
Low	Docker isolation	Container cannot touch your Mac outside /workspace
Low	Telegram	Dedicated bot, group joining disabled
Low	Cloud model	Frontier models resist prompt injection well
Low	Network exposure	Loopback-only, no LAN/internet exposure
Medium	WhatsApp	Uses unofficial protocol, agent sends as you
Medium	Local models	8B models are vulnerable to prompt injection
Medium	Skill ecosystem	Avoid third-party skills entirely

Known Gotchas (Lessons Learned)

These are real issues encountered during installation. Save yourself the debugging:

1. Docker image is on GitHub Container Registry, **not Docker Hub.** Use ghcr.io/openclaw/openclaw:latest
2. **The 'openclaw' CLI is not in PATH** inside the container. Use 'node /app/openclaw.mjs' instead.
3. **Container home directory is /home/node,** not /home/openclaw. Mount volumes accordingly.
4. **Gateway port is 18789,** not 3200. Bridge port is 18790.
5. **Apple Silicon Macs need the ARM version of Docker Desktop.** Homebrew may install the Intel version. Check the dialog on first launch.
6. **Docker Desktop cannot access directories owned by other macOS users.** Keep the project under your own home directory (~/) and rely on Docker for isolation.
7. **Interactive wizards can break inside 'docker exec'.** If the terminal gets stuck, Ctrl+C and restart the container. Your config is usually saved.
8. **Pairing codes expire quickly.** Run the approve command immediately. If it fails, send another message to get a fresh code.
9. **The docker-compose.yml 'version' field is obsolete.** The warning is harmless, but you can remove the 'version' line.

Sources

- OpenClaw GitHub: github.com/openclaw/openclaw
- OpenClaw Security Docs: docs.openclaw.ai/gateway/security
- OpenClaw Docker Docs: docs.openclaw.ai/install/docker
- Cisco: Personal AI Agents Are a Security Nightmare — blogs.cisco.com
- Adversa.ai: OpenClaw Hardening Guide — adversa.ai/blog
- Northflank: How to Sandbox AI Agents — northflank.com/blog
- OpenClaw Wikipedia: en.wikipedia.org/wiki/OpenClaw

Guide created February 2026. Based on OpenClaw v2026.2.17.