PERSONAL PROJECT · CASE STUDY

OpenClaw

Building my own 24/7 AI agent, from a bare VPS to a working assistant in my pocket.

docker compose up -d openclaw-gateway

2026

01 · THE PROBLEM

Chatbots talk. I wanted something that does.

Most assistants stop at conversation. I wanted an agent that takes real actions: always on, reachable from an app I already use, running on infrastructure I own.

01 Available 24/7, not just when I open a laptop

02 Reachable from a normal chat thread

03 Able to act: files, commands, the web, my inbox

04 Self-hosted, so I control the data and the cost

02 · WHAT IS OPENCLAW

An open-source agent that executes real tasks, not just chat.

A project that grew fast and now runs behind a huge number of personal setups worldwide, controlled entirely through everyday messaging apps.

300k+
GitHub stars

3
chat channels supported

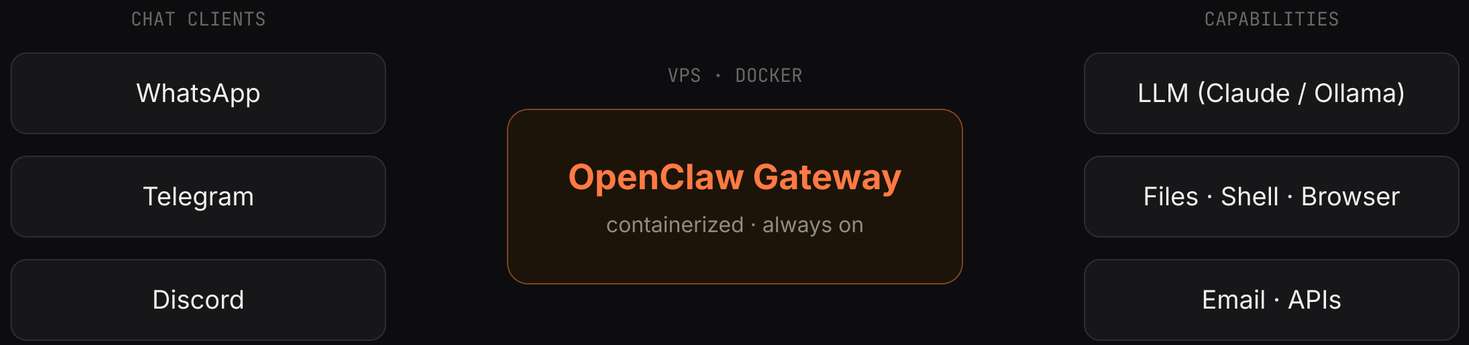
<10€
monthly hosting cost

what it can do

- Read & write files on disk
- Execute shell commands
- Browse and read the web
- Send and draft emails
- Call external APIs

03 · ARCHITECTURE

One VPS, four moving pieces.

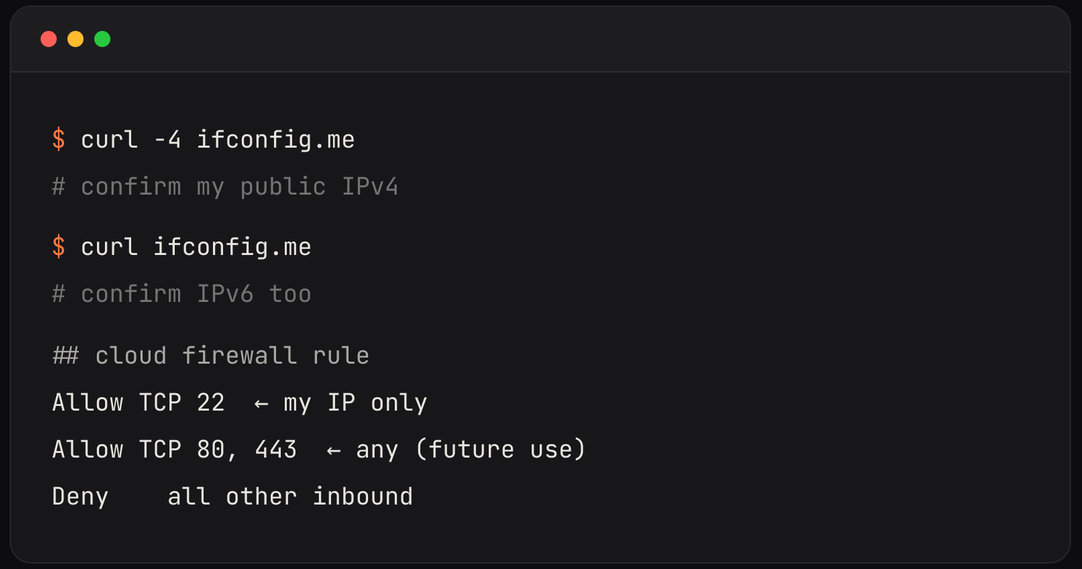


04 · BUILD LOG · STAGE 1 OF 6

Provision the server

A minimal, cheap VPS is plenty. I picked a shared instance in Nuremberg, checked my own IP, and scoped the cloud firewall to it before installing anything.

- → Ubuntu 24.04 LTS · shared vCPU · CPX22
- → IPv4 + IPv6 enabled
- → Cloud firewall: only my IP can reach SSH
- → All other inbound ports closed by default



04 · BUILD LOG · STAGE 2 OF 6

Lock down access

Before installing anything user-facing, I removed every easy way in: key-only auth, a dedicated non-root user, root login disabled.

- → ed25519 SSH keypair, no passwords
- → Dedicated user in the sudo & docker groups
- → Root login disabled entirely
- → Auth attempts capped, grace time shortened

```
$ ssh-keygen -t ed25519 -f ~/.ssh/id_openclaw
$ adduser openclaw && usermod -aG sudo,docker openclaw

## /etc/ssh/sshd_config
PermitRootLogin no
PasswordAuthentication no
MaxAuthTries 3
LoginGraceTime 20

$ systemctl restart ssh
```

04 · BUILD LOG · STAGE 3 OF 6

Install the runtime

OpenClaw ships as a container, so Docker is the only hard dependency. I layered on a few small tools to keep the box healthy and quiet while unattended.

- → Docker Engine via the official install script
- → Unattended security upgrades, low priority
- → fail2ban to auto-ban bot scanners
- → Homebrew for day-to-day CLI tooling

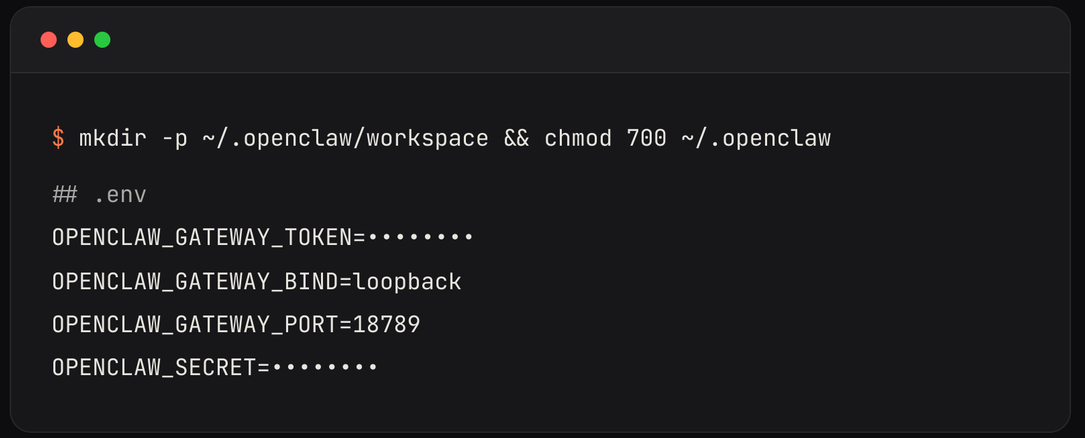
```
$ apt-get update && apt-get upgrade -y
$ apt-get install -y git curl ca-certificates
$ curl -fsSL https://get.docker.com | sh
$ apt-get install -y unattended-upgrades
$ dpkg-reconfigure --priority=low unattended-upgrades
$ apt-get install -y fail2ban && systemctl enable --now fail2ban

## dev tooling for the new user
$ NONINTERACTIVE=1 /bin/bash -c "$(curl -fsSL .../install.sh)"
$ eval "$(/home/linuxbrew/.linuxbrew/bin/brew shellenv)"
```

04 · BUILD LOG · STAGE 4 OF 6

Configure the agent

A workspace directory outside the container keeps my data safe from image updates. Secrets live only in a local .env file, never in source control.

- → Persistent workspace at ~/.openclaw
- → Locked to owner-only permissions (700)
- → Gateway token & secret generated locally
- → Gateway bound to loopback only, never public



04 · BUILD LOG · STAGE 5 OF 6

Wire up the compose file

I started from the project's official compose file and adjusted volumes, ports and env references to match my own server layout.

- → Base: github.com/openclaw/openclaw
- → Custom volumes for the workspace dir
- → Env file wired in, no secrets inline
- → My config: pastebin.com/8NLr8pu3

docker-compose.yml

```
services:
  openclaw-gateway:
    image: ghcr.io/openclaw/openclaw:latest
    env_file: .env
    volumes:
      - ~/.openclaw/workspace:/data/workspace
    ports:
      - "127.0.0.1:18789:18789"
    restart: unless-stopped
```

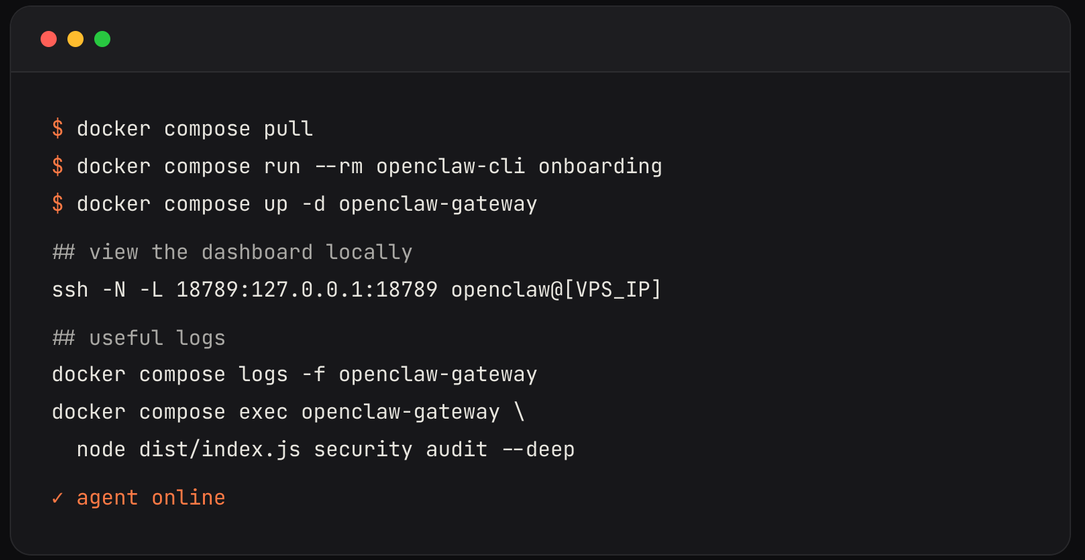
↳ full file: pastebin.com/8NLr8pu3

04 · BUILD LOG · STAGE 6 OF 6

Onboard & launch

The CLI wizard handles the last mile: picking a model, connecting a channel via a bot token, and turning on the extras I wanted.

- → Model: Claude
- → Channel: Telegram, token from @BotFather
- → Extras: voice transcription, web search, image generation, Notion
- → Running detached, tunneled dashboard for peace of mind



05 · SECURITY POSTURE

An agent with real capabilities needs a locked-down home.

- ✓ SSH key-only authentication, no passwords
- ✓ Root login disabled entirely
- ✓ Firewall scoped to a single known IP
- ✓ fail2ban auto-blocking repeated bot probes
- ✓ Agent isolated inside its own container
- ✓ Secrets kept in .env, never in source control

06 · THE RESULT

A single Telegram thread that reads files, runs commands, browses the web, and answers my email.

📁

Files & commands

Reads, edits and runs scripts on the box directly from a chat message.

🌐

Web & APIs

Browses pages and calls external services to fetch or act on live data.

✉️

Email

Drafts and sends messages on my behalf without opening a client.

⏱️

Always on

Runs detached on the VPS, reachable 24/7 from any device I carry.

07 · TAKEAWAYS

A weekend project, a permanent tool.

Owning the infrastructure meant a bit more setup up front, but full control over cost, data, and what the agent is allowed to touch.

~€10/mo
total hosting cost

~1 evening
from empty VPS to running agent

100% mine
infra, data, and secrets

THANKS FOR READING

Deployed OpenClaw.
Ready for the next one.

project
github.com/openclaw/openclaw

portfolio
paumurl.github.io

contact
paularobleslopez@gmail.com