Hosting and deployment

Hetzner

Goal

Run a persistent OpenClaw Gateway on a Hetzner VPS using Docker, with durable state, baked-in binaries, and safe restart behavior.

If you want “OpenClaw 24/7 for ~$5”, this is the simplest reliable setup. Hetzner pricing changes; pick the smallest Debian/Ubuntu VPS and scale up if you hit OOMs.

What are we doing (simple terms)?

- Rent a small Linux server (Hetzner VPS)
- Install Docker (isolated app runtime)
- Start the OpenClaw Gateway in Docker
- Persist `~/.openclaw` + `~/.openclaw/workspace` on the host (survives restarts/rebuilds)
- Access the Control UI from your laptop via an SSH tunnel

The Gateway can be accessed via:

- SSH port forwarding from your laptop
- Direct port exposure if you manage firewalling and tokens yourself

This guide assumes Ubuntu or Debian on Hetzner.
If you are on another Linux VPS, map packages accordingly. For the generic Docker flow, see **Docker**.

Quick path (experienced operators)

1. Provision Hetzner VPS
2. Install Docker
3. Clone OpenClaw repository
4. Create persistent host directories
5. Configure `.env` and `docker-compose.yml`
6. Bake required binaries into the image
7. `docker compose up -d`
8. Verify persistence and Gateway access

What you need

- Hetzner VPS with root access
- SSH access from your laptop
- Basic comfort with SSH + copy/paste
- ~20 minutes
- Docker and Docker Compose
- Model auth credentials
- Optional provider credentials
  - WhatsApp QR
  - Telegram bot token

Gmail OAuth

›

1) Provision the VPS

Create an Ubuntu or Debian VPS in Hetzner.

Connect as root:

```
ssh root@YOUR_VPS_IP
```

This guide assumes the VPS is stateful. Do not treat it as disposable infrastructure.

2) Install Docker (on the VPS)

```
apt-get update
apt-get install -y git curl ca-certificates
curl -fsSL https://get.docker.com | sh
```

Verify:

```
docker --version
docker compose version
```

3) Clone the OpenClaw repository

```
git clone https://github.com/openclaw/openclaw.git
cd openclaw
```

This guide assumes you will build a custom image to guarantee binary persistence.

4) Create persistent host directories

Docker containers are ephemeral. All long-lived state must live on the host.

```
mkdir -p /root/.openclaw/workspace

# Set ownership to the container user (uid 1000):
chown -R 1000:1000 /root/.openclaw
```

5) Configure environment variables

Create `.env` in the repository root.

```
OPENCLAW_IMAGE=openclaw:latest
OPENCLAW_GATEWAY_TOKEN=change-me-now
OPENCLAW_GATEWAY_BIND=lan
OPENCLAW_GATEWAY_PORT=18789

OPENCLAW_CONFIG_DIR=/root/.openclaw
OPENCLAW_WORKSPACE_DIR=/root/.openclaw/workspace

GOG_KEYRING_PASSWORD=change-me-now
XDG_CONFIG_HOME=/home/node/.openclaw
```

Generate strong secrets:

```
openssl rand -hex 32
```

Do not commit this file.

6) Docker Compose configuration

Create or update `docker-compose.yml` .

```
services:
  openclaw-gateway:
    image: ${OPENCLAW_IMAGE}
    build: .
    restart: unless-stopped
    env_file:
      - .env
    environment:
      - HOME=/home/node
      - NODE_ENV=production
      - TERM=xterm-256color
      - OPENCLAW_GATEWAY_BIND=${OPENCLAW_GATEWAY_BIND}
      - OPENCLAW_GATEWAY_PORT=${OPENCLAW_GATEWAY_PORT}
      - OPENCLAW_GATEWAY_TOKEN=${OPENCLAW_GATEWAY_TOKEN}
      - GOG_KEYRING_PASSWORD=${GOG_KEYRING_PASSWORD}
      - XDG_CONFIG_HOME=${XDG_CONFIG_HOME}
      - PATH=/home/linuxbrew/.linuxbrew/bin:/usr/local/sbin:/usr/local/bin:/usr/sl
    volumes:
      - ${OPENCLAW_CONFIG_DIR}:/home/node/.openclaw
      - ${OPENCLAW_WORKSPACE_DIR}:/home/node/.openclaw/workspace
    ports:
      # Recommended: keep the Gateway loopback-only on the VPS; access via SSH tu
      # To expose it publicly, remove the `127.0.0.1:` prefix and firewall accord
      - "127.0.0.1:${OPENCLAW_GATEWAY_PORT}:18789"
    command:
      [
        "node",
        "dist/index.js",
        "gateway",
        "--bind",
        "${OPENCLAW_GATEWAY_BIND}",
        "--port",
        "${OPENCLAW_GATEWAY_PORT}",
        "--allow-unconfigured",
      ]
```

`--allow-unconfigured` is only for bootstrap convenience, it is not a replacement for a proper gateway configuration. Still set auth

(`gateway.auth.token` or password) and use safe bind settings for your deployment.

›

7) Bake required binaries into the image (critical)

Installing binaries inside a running container is a trap. Anything installed at runtime will be lost on restart.

All external binaries required by skills must be installed at image build time.

The examples below show three common binaries only:

- `gog` for Gmail access
- `goplaces` for Google Places
- `wacli` for WhatsApp

These are examples, not a complete list. You may install as many binaries as needed using the same pattern.

If you add new skills later that depend on additional binaries, you must:

1. Update the Dockerfile
2. Rebuild the image
3. Restart the containers

Example Dockerfile

```
FROM node:22-bookworm

RUN apt-get update && apt-get install -y socat && rm -rf /var/lib/apt/lists/*
                                                                              ›

# Example binary 1: Gmail CLI
RUN curl -L https://github.com/steipete/gog/releases/latest/download/gog_Linux_x8
  | tar -xz -C /usr/local/bin && chmod +x /usr/local/bin/gog

# Example binary 2: Google Places CLI
RUN curl -L https://github.com/steipete/goplaces/releases/latest/download/goplace
  | tar -xz -C /usr/local/bin && chmod +x /usr/local/bin/goplaces

# Example binary 3: WhatsApp CLI
RUN curl -L https://github.com/steipete/wacli/releases/latest/download/wacli_Linu
  | tar -xz -C /usr/local/bin && chmod +x /usr/local/bin/wacli

# Add more binaries below using the same pattern

WORKDIR /app
COPY package.json pnpm-lock.yaml pnpm-workspace.yaml .npmrc ./
COPY ui/package.json ./ui/package.json
COPY scripts ./scripts

RUN corepack enable
RUN pnpm install --frozen-lockfile

COPY . .
RUN pnpm build
RUN pnpm ui:install
RUN pnpm ui:build

ENV NODE_ENV=production

CMD ["node","dist/index.js"]
```

8) Build and launch

```
docker compose build
docker compose up -d openclaw-gateway
```

Verify binaries:

```
docker compose exec openclaw-gateway which gog
docker compose exec openclaw-gateway which goplaces
docker compose exec openclaw-gateway which wacli
```

Expected output:

```
/usr/local/bin/gog
/usr/local/bin/goplaces
/usr/local/bin/wacli
```

---

9) Verify Gateway

```
docker compose logs -f openclaw-gateway
```

Success:

```
[gateway] listening on ws://0.0.0.0:18789
```

From your laptop:

```
ssh -N -L 18789:127.0.0.1:18789 root@YOUR_VPS_IP
```

Open:

`http://127.0.0.1:18789/`

Paste your gateway token.

What persists where (source of truth)

OpenClaw runs in Docker, but Docker is not the source of truth. All long-lived state must survive restarts, rebuilds, and reboots.

Component	Location	Persistence mechanism	Notes
Gateway config	`/home/node/.openclaw/`	Host volume mount	Includes `openclaw.json` , tokens
Model auth profiles	`/home/node/.openclaw/`	Host volume mount	OAuth tokens, API keys
Skill configs	`/home/node/.openclaw/skills/`	Host volume mount	Skill-level state
Agent workspace	`/home/node/.openclaw/workspace/`	Host volume mount	Code and agent artifacts
WhatsApp session	`/home/node/.openclaw/`	Host volume mount	Preserves QR login
Gmail keyring	`/home/node/.openclaw/`	Host volume + password	Requires `GOG_KEYRING_PASSWORD`
External binaries	`/usr/local/bin/`	Docker image	Must be baked at build time
Node runtime	Container filesystem	Docker image	Rebuilt every image build
OS packages	Container filesystem	Docker image	Do not install at runtime
Docker container	Ephemeral	Restartable	Safe to destroy

Infrastructure as Code (Terraform)

For teams preferring infrastructure-as-code workflows, a community-maintained Terraform setup provides:

- Modular Terraform configuration with remote state management
- Automated provisioning via cloud-init
- Deployment scripts (bootstrap, deploy, backup/restore)
- Security hardening (firewall, UFW, SSH-only access)
- SSH tunnel configuration for gateway access

Repositories:

- Infrastructure: **openclaw-terraform-hetzner**
- Docker config: **openclaw-docker-config**

This approach complements the Docker setup above with reproducible deployments, version-controlled infrastructure, and automated disaster recovery.

> **Note:** Community-maintained. For issues or contributions, see the repository links above.

‹ Fly.io GCP ›