Other install methods

Docker

Docker is **optional**. Use it only if you want a containerized gateway or to validate the Docker flow.

Is Docker right for me?

Yes: you want an isolated, throwaway gateway environment or to run OpenClaw on a host without local installs.

No: you're running on your own machine and just want the fastest dev loop. Use the normal install flow instead.

Sandboxing note: agent sandboxing uses Docker too, but it does **not** require the full gateway to run in Docker. See Sandboxing.

This guide covers:

Containerized Gateway (full OpenClaw in Docker)

Per-session Agent Sandbox (host gateway + Docker-isolated agent tools)

Sandboxing details: Sandboxing

Requirements

Docker Desktop (or Docker Engine) + Docker Compose v2

Enough disk for images + logs

Containerized Gateway (Docker Compose)

Quick start (recommended)

From repo root:

```
./docker-setup.sh
```

This script:

- builds the gateway image
- runs the onboarding wizard
- prints optional provider setup hints
- starts the gateway via Docker Compose
- generates a gateway token and writes it to `.env`

Optional env vars:

- `OPENCLAW_DOCKER_APT_PACKAGES` — install extra apt packages during build
- `OPENCLAW_EXTRA_MOUNTS` — add extra host bind mounts
- `OPENCLAW_HOME_VOLUME` — persist `/home/node` in a named volume

After it finishes:

- Open `http://127.0.0.1:18789/` in your browser.
- Paste the token into the Control UI (Settings → token).
- Need the URL again? Run `docker compose run --rm openclaw-cli dashboard --no-open .`

It writes config/workspace on the host:

- `~/.openclaw/`
- `~/.openclaw/workspace`

Running on a VPS? See **Hetzner (Docker VPS)**.

Shell Helpers (optional) ›

For easier day-to-day Docker management, install `ClawDock`:

```
mkdir -p ~/.clawdock && curl -sL https://raw.githubusercontent.com/ope
```

Add to your shell config (zsh):

```
echo 'source ~/.clawdock/clawdock-helpers.sh' >> ~/.zshrc && source ~/
```

Then use `clawdock-start`, `clawdock-stop`, `clawdock-dashboard`, etc. Run `clawdock-help` for all commands.

See for details.

Manual flow (compose)

```
docker build -t openclaw:local -f Dockerfile .
docker compose run --rm openclaw-cli onboard
docker compose up -d openclaw-gateway
```

Note: run `docker compose ...` from the repo root. If you enabled `OPENCLAW_EXTRA_MOUNTS` or `OPENCLAW_HOME_VOLUME`, the setup script writes `docker-compose.extra.yml`; include it when running Compose elsewhere:

```
docker compose -f docker-compose.yml -f docker-compose.extra.yml <comm
```

Control UI token + pairing (Docker)

If you see “unauthorized” or “disconnected (1008): pairing required”, fetch a fresh dashboard link and approve the browser device:

```
docker compose run --rm openclaw-cli dashboard --no-open
docker compose run --rm openclaw-cli devices list
docker compose run --rm openclaw-cli devices approve <requestId>
```

More detail: , .

Extra mounts (optional)

If you want to mount additional host directories into the containers, set `OPENCLAW_EXTRA_MOUNTS` before running `docker-setup.sh`. This accepts a comma-separated list of Docker bind mounts and applies them to both `openclaw-gateway` and `openclaw-cli` by generating `docker-compose.extra.yml`.

Example:

```
export OPENCLAW_EXTRA_MOUNTS="$HOME/.codex:/home/node/.codex:ro,$HOME/
./docker-setup.sh
```

Notes:

- Paths must be shared with Docker Desktop on macOS/Windows.
- If you edit `OPENCLAW_EXTRA_MOUNTS`, rerun `docker-setup.sh` to regenerate the extra compose file.
- `docker-compose.extra.yml` is generated. Don’t hand-edit it.

Persist the entire container home (optional)

If you want `/home/node` to persist across container recreation, set a named volume via `OPENCLAW_HOME_VOLUME`. This creates a Docker volume and mounts it at `/home/node`, while keeping the standard config/workspace

bind mounts. Use a named volume here (not a bind path); for bind mounts, use `OPENCLAW_EXTRA_MOUNTS`.

Example:

```
export OPENCLAW_HOME_VOLUME="openclaw_home"
./docker-setup.sh
```

You can combine this with extra mounts:

```
export OPENCLAW_HOME_VOLUME="openclaw_home"
export OPENCLAW_EXTRA_MOUNTS="$HOME/.codex:/home/node/.codex:ro,$HOME/github:/home
./docker-setup.sh
```

Notes:

- If you change `OPENCLAW_HOME_VOLUME`, rerun `docker-setup.sh` to regenerate the extra compose file.
- The named volume persists until removed with `docker volume rm <name>`.

Install extra apt packages (optional)

If you need system packages inside the image (for example, build tools or media libraries), set `OPENCLAW_DOCKER_APT_PACKAGES` before running `docker-setup.sh`. This installs the packages during the image build, so they persist even if the container is deleted.

Example:

```
export OPENCLAW_DOCKER_APT_PACKAGES="ffmpeg build-essential"
./docker-setup.sh
```

Notes:

This accepts a space-separated list of apt package names.

If you change `OPENCLAW_DOCKER_APT_PACKAGES`, rerun `docker-setup.sh` to rebuild the image.

Power-user / full-featured container (opt-in)

The default Docker image is **security-first** and runs as the non-root `node` user. This keeps the attack surface small, but it means:

- no system package installs at runtime
- no Homebrew by default
- no bundled Chromium/Playwright browsers

If you want a more full-featured container, use these opt-in knobs:

1. **Persist `/home/node`** so browser downloads and tool caches survive:

```
export OPENCLAW_HOME_VOLUME="openclaw_home"
./docker-setup.sh
```

2. **Bake system deps into the image** (repeatable + persistent):

```
export OPENCLAW_DOCKER_APT_PACKAGES="git curl jq"
./docker-setup.sh
```

3. **Install Playwright browsers without `npx`** (avoids npm override conflicts):

```
docker compose run --rm openclaw-cli \
  node /app/node_modules/playwright-core/cli.js install chromium
```

If you need Playwright to install system deps, rebuild the image with `OPENCLAW_DOCKER_APT_PACKAGES` instead of using `--with-deps` at runtime.

4. **Persist Playwright browser downloads**:

   Set `PLAYWRIGHT_BROWSERS_PATH=/home/node/.cache/ms-playwright` in `docker-compose.yml`.

   Ensure `/home/node` persists via `OPENCLAW_HOME_VOLUME`, or mount `/home/node/.cache/ms-playwright` via `OPENCLAW_EXTRA_MOUNTS`.

Permissions + EACCES

The image runs as `node` (uid 1000). If you see permission errors on `/home/node/.openclaw`, make sure your host bind mounts are owned by uid 1000.

Example (Linux host):

```
sudo chown -R 1000:1000 /path/to/openclaw-config /path/to/openclaw-wor
```

If you choose to run as root for convenience, you accept the security tradeoff.

Faster rebuilds (recommended)

To speed up rebuilds, order your Dockerfile so dependency layers are cached. This avoids re-running `pnpm install` unless lockfiles change:

```
FROM node:22-bookworm

# Install Bun (required for build scripts)
RUN curl -fsSL https://bun.sh/install | bash
ENV PATH="/root/.bun/bin:${PATH}"

RUN corepack enable

WORKDIR /app

# Cache dependencies unless package metadata changes
COPY package.json pnpm-lock.yaml pnpm-workspace.yaml .npmrc ./
COPY ui/package.json ./ui/package.json
COPY scripts ./scripts

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"]
```

Channel setup (optional)

Use the CLI container to configure channels, then restart the gateway if needed.

WhatsApp (QR):

```
docker compose run --rm openclaw-cli channels login
```

Telegram (bot token):

```
docker compose run --rm openclaw-cli channels add --channel telegram -
```

```
Discord (bot token):
```

```
docker compose run --rm openclaw-cli channels add --channel discord --
```

Docs: , ,

OpenAI Codex OAuth (headless Docker)

If you pick OpenAI Codex OAuth in the wizard, it opens a browser URL and tries to capture a callback on `http://127.0.0.1:1455/auth/callback`. In Docker or headless setups that callback can show a browser error. Copy the full redirect URL you land on and paste it back into the wizard to finish auth.

Health check

```
docker compose exec openclaw-gateway node dist/index.js health --toker
```

E2E smoke test (Docker)

```
scripts/e2e/onboard-docker.sh
```

QR import smoke test (Docker)

```
pnpm test:docker:qr
```

Notes

- Gateway bind defaults to `lan` for container use.
- Dockerfile CMD uses› `--allow-unconfigured` ; mounted config with `gateway.mode` not `local` will still start. Override CMD to enforce the guard.
- The gateway container is the source of truth for sessions (`~/.openclaw/agents/<agentId>/sessions/`).

Agent Sandbox (host gateway + Docker tools)

Deep dive: Sandboxing

What it does

When `agents.defaults.sandbox` is enabled, **non-main sessions** run tools inside a Docker container. The gateway stays on your host, but the tool execution is isolated:

- scope: `"agent"` by default (one container + workspace per agent)
- scope: `"session"` for per-session isolation
- per-scope workspace folder mounted at `/workspace`
- optional agent workspace access (`agents.defaults.sandbox.workspaceAccess`)
- allow/deny tool policy (deny wins)
- inbound media is copied into the active sandbox workspace (`media/inbound/*`) so tools can read it (with `workspaceAccess: "rw"` , this lands in the agent workspace)

Warning: `scope: "shared"` disables cross-session isolation. All sessions share one container and one workspace.

Per-agent sandbox profiles (multi-agent)

If you use multi-agent routing, each agent can override sandbox + tool settings: `agents.list[].sandbox` and `agents.list[].tools` (plus `agents.list[].tools.sandbox.tools`). This lets you run mixed access levels in one gateway:

- Full access (personal agent)
- Read-only tools + read-only workspace (family/work agent)
- No filesystem/shell tools (public agent)

See **Multi-Agent Sandbox & Tools** for examples, precedence, and troubleshooting.

Default behavior

- Image: `openclaw-sandbox:bookworm-slim`
- One container per agent
- Agent workspace access: `workspaceAccess: "none"` (default) uses `~/.openclaw/sandboxes`
  - `"ro"` keeps the sandbox workspace at `/workspace` and mounts the agent workspace read-only at `/agent` (disables `write` / `edit` / `apply_patch`)
  - `"rw"` mounts the agent workspace read/write at `/workspace`
- Auto-prune: idle > 24h OR age > 7d
- Network: `none` by default (explicitly opt-in if you need egress)
- Default allow: `exec`, `process`, `read`, `write`, `edit`, `sessions_list`, `sessions_history`, `sessions_send`, `sessions_spawn`, `session_status`
- Default deny: `browser`, `canvas`, `nodes`, `cron`, `discord`, `gateway`

Enable sandboxing

If you plan to install packages in `setupCommand`, note:

- Default `docker.network` is `"none"` (no egress).
- `readOnlyRoot: true` blocks package installs.
- `user` must be root for `apt-get` (omit `user` or set `user: "0:0"`).

OpenClaw auto-recreates containers when `setupCommand` (or docker config) changes unless the container was **recently used** (within ~5 minutes). Hot containers log a warning with the exact `openclaw sandbox recreate ...` command.

```
agents: {
  defaults: {
    sandbox: {
      mode: "non-main", // off | non-main | all
      scope: "agent", // session | agent | shared (agent is default)
      workspaceAccess: "none", // none | ro | rw
      workspaceRoot: "~/.openclaw/sandboxes",
      docker: {
        image: "openclaw-sandbox:bookworm-slim",
        workdir: "/workspace",
        readOnlyRoot: true,
        tmpfs: ["/tmp", "/var/tmp", "/run"],
        network: "none",
        user: "1000:1000",
        capDrop: ["ALL"],
        env: { LANG: "C.UTF-8" },
        setupCommand: "apt-get update && apt-get install -y git curl jq",
        pidsLimit: 256,
        memory: "1g",
        memorySwap: "2g",
        cpus: 1,
        ulimits: {
          nofile: { soft: 1024, hard: 2048 },
          nproc: 256,
        },
        seccompProfile: "/path/to/seccomp.json",
        apparmorProfile: "openclaw-sandbox",
        dns: ["1.1.1.1", "8.8.8.8"],
        extraHosts: ["internal.service:10.0.0.5"],
      },
      prune: {
        idleHours: 24, // 0 disables idle pruning
        maxAgeDays: 7, // 0 disables max-age pruning
      },
    },
  },
},
tools: {
  sandbox: {
```

```
      tools: {
        allow: [
          "exec",
          "process",
          "read",
          "write",
          "edit",
          "sessions_list",
          "sessions_history",
          "sessions_send",
          "sessions_spawn",
          "session_status",
        ],
        deny: ["browser", "canvas", "nodes", "cron", "discord", "gateway"],
      },
    },
  },
}
```

Hardening knobs live under `agents.defaults.sandbox.docker`: `network`, `user`, `pidsLimit`, `memory`, `memorySwap`, `cpus`, `ulimits`, `seccompProfile`, `apparmorProfile`, `dns`, `extraHosts`.

Multi-agent: override `agents.defaults.sandbox.{docker,browser,prune}.*` per agent via `agents.list[].sandbox.{docker,browser,prune}.*` (ignored when `agents.defaults.sandbox.scope` / `agents.list[].sandbox.scope` is `"shared"`).

Build the default sandbox image

```
scripts/sandbox-setup.sh
```

This builds `openclaw-sandbox:bookworm-slim` using `Dockerfile.sandbox`.

Sandbox common image (optional)

If you want a sandbox image with common build tooling (Node, Go, Rust, etc.), build the common image:

```
scripts/sandbox-common-setup.sh
```

This builds `openclaw-sandbox-common:bookworm-slim`. To use it:

```
{
  agents: {
    defaults: {
      sandbox: { docker: { image: "openclaw-sandbox-common:bookworm-slim" } },
    },
  },
}
```

Sandbox browser image

To run the browser tool inside the sandbox, build the browser image:

```
scripts/sandbox-browser-setup.sh
```

This builds `openclaw-sandbox-browser:bookworm-slim` using `Dockerfile.sandbox-browser`. The container runs Chromium with CDP enabled and an optional noVNC observer (headful via Xvfb).

Notes:

- Headful (Xvfb) reduces bot blocking vs headless.
- Headless can still be used by setting `agents.defaults.sandbox.browser.headless=true`.
- No full desktop environment (GNOME) is needed; Xvfb provides the display.

Use config:

```
{
  agents: {
    defaults: {
      sandbox: {
        browser: { enabled: true },
      },
    },
  },
}
```

Custom browser image:

```
{
  agents: {
    defaults: {
      sandbox: { browser: { image: "my-openclaw-browser" } },
    },
  },
}
```

When enabled, the agent receives:

- a sandbox browser control URL (for the `browser` tool)
- a noVNC URL (if enabled and headless=false)

Remember: if you use an allowlist for tools, add `browser` (and remove it from deny) or the tool remains blocked. Prune rules (`agents.defaults.sandbox.prune`) apply to browser containers too.

Custom sandbox image

Build your own image and point config to it:

```
docker build -t my-openclaw-sbx -f Dockerfile.sandbox .
```

```
{
  agents: {
    defaults: {
      sandbox: { docker: { image: "my-openclaw-sbx" } },
    },
  },
}
```

Tool policy (allow/deny)

`deny` wins over `allow`.

If `allow` is empty: all tools (except deny) are available.

If `allow` is non-empty: only tools in `allow` are available (minus deny).

Pruning strategy

Two knobs:

`prune.idleHours`: remove containers not used in X hours (0 = disable)

`prune.maxAgeDays`: remove containers older than X days (0 = disable)

Example:

Keep busy sessions but cap lifetime: `idleHours: 24`, `maxAgeDays: 7`

Never prune: `idleHours: 0`, `maxAgeDays: 0`

Security notes

Hard wall only applies to **tools** (exec/read/write/edit/apply_patch).

Host-only tools like browser/camera/canvas are blocked by default.

Allowing `browser` in sandbox **breaks isolation** (browser runs on host).

Troubleshooting

Image missing: build with `scripts/sandbox-setup.sh` or set `agents.defaults.sandbox.docker.image`.

Container not running: it will auto-create per session on demand.

Permission errors in sandbox: set `docker.user` to a UID:GID that matches your mounted workspace ownership (or chown the workspace folder).

Custom tools not found: OpenClaw runs commands with `sh -lc` (login shell), which sources `/etc/profile` and may reset PATH. Set `docker.env.PATH` to prepend your custom tool paths (e.g., `/custom/bin:/usr/local/share/npm-global/bin`), or add a script under `/etc/profile.d/` in your Dockerfile.

‹ Installer Internals Podman ›