SHSHELL.COM

RESEARCH REPORT

BY SUDEEP DEVKOTA

JUNE 2026

https://shshell.com

SHSHELL.COM

GETTING STARTED WITH OPENCLAW

A practical setup report for installing, running, verifying, securing, updating, and troubleshooting OpenClaw across macOS, Linux, Windows, WSL2, Docker, Podman, Nix, and source builds.

RESEARCH REPORT

BY SUDEEP DEVKOTA

JUNE 2026

https://shshell.com

Contents

Getting Started with OpenClaw

OpenClaw is a self-hosted personal AI assistant and gateway. You run a Gateway process on your own machine, server, or container, connect it to model providers and communication channels, and then interact with the assistant through the CLI, companion apps, chat surfaces, or the browser Control UI.

This report is written for a first-time operator who wants a working setup, not only a list of install commands. It covers the practical install paths, where OpenClaw runs, how to verify the Gateway, how to send a first request, how to choose the right platform route, and what to lock down before exposing anything beyond localhost.

The research base for this guide is the official OpenClaw GitHub repository, the official install documentation, the getting started guide, the Gateway runbook, the Windows platform guide, the macOS platform guide, the Docker guide, the Podman guide, the Nix guide, the CLI agent reference, and the updating guide. Commands below reflect those sources as reviewed on June 6, 2026.

> **EXECUTIVE INSIGHT**
>
> *Treat OpenClaw as local infrastructure with real authority over files, shell commands, channels, and sessions. Install it like an agent runtime, not like a casual chat app.*

The Fastest Working Path

If you are on macOS, Linux, or WSL2 and want the shortest route, use the official installer:

```
curl -fsSL https://openclaw.ai/install.sh | bash
```

If you are on native Windows and want the terminal-first CLI/Gateway path, use PowerShell:

```
iwr -useb https://openclaw.ai/install.ps1 | iex
```

Then verify:

```
openclaw --version
openclaw doctor
```

```
openclaw gateway status
```

For most local users, the recommended first operational command is:

```
openclaw onboard --install-daemon
```

Onboarding creates the local configuration, helps connect model credentials and channels, and installs a managed Gateway service. On macOS that service is managed through launchd. On Linux and WSL2 it is normally a systemd user service. On native Windows it uses a Scheduled Task when available, with a Startup-folder fallback if task creation is denied.

What You Are Installing

OpenClaw has three pieces that matter during setup:

Piece	What it does	Where it usually runs
CLI	Runs setup, health checks, agent turns, channel commands, updates, and Gateway operations	Your terminal, host OS, or container helper
Gateway	Long-running control plane for sessions, channels, tools, nodes, HTTP APIs, and WebSocket control	Local machine, WSL2, VPS, Docker, Podman, or Nix-managed service
Companion apps/nodes	Optional platform-specific surfaces for status, chat, voice, screen/camera/canvas, and local capabilities	macOS app, Windows Hub, mobile apps, or node hosts

The Gateway defaults to loopback binding and authenticates clients by default. That is the right starting point. Do not publish the Gateway directly to the internet. If you need remote access, prefer a VPN/Tailscale path or an SSH tunnel, and keep Gateway authentication enabled.

Platform Decision Matrix

Choose the install path based on where the Gateway should live.

Situation	Best path	Why
Personal Mac, local assistant	macOS app or installer script	The macOS app manages permissions and can manage or attach to the local Gateway
Linux workstation	installer script or npm global install	Fastest setup, systemd user service support
Windows desktop user	Windows Hub	Native setup, tray status, chat, diagnostics, WSL Gateway provisioning
Windows terminal-first operator	PowerShell installer or WSL2	Native CLI is supported; WSL2 remains the most Linux-compatible runtime
Cloud VPS	Linux install with supervised Gateway	Simple always-on process, easier remote channel access
Disposable or isolated environment	Docker	Containerized Gateway with local or prebuilt images
Rootless Linux container host	Podman	Non-root container with host-managed state
Declarative workstation/server	Nix/Home Manager	Pinned package, reproducible config, rollback
Contributor or bleeding-edge user	Source checkout	Use pnpm build/watch flows and local repo development

Prerequisites

Use Node 24 unless you have a reason not to. Official OpenClaw sources describe Node 24 as the recommended runtime. Some pages mention Node 22.14+ or Node 22.19+ as supported lower bounds; using Node 24 avoids that ambiguity.

Common prerequisites:

- A terminal with shell access.
- Node 24 recommended.
- A model provider credential or subscription path, depending on your chosen provider.
- `pnpm` only if building from source.
- Docker or Podman only if using containerized Gateway paths.
- A local, non-synced state directory. On macOS, avoid iCloud or other cloud-synced folders for OpenClaw state.

OpenClaw stores normal local state under `~/.openclaw` unless you override paths. Keep that directory private. It can include config, sessions, provider auth metadata, channel credentials, allowlists, workspace files, and secrets references.

Install on macOS

You have two practical options on macOS: the companion app path or the CLI installer path.

Option A: macOS App

Use the macOS app if you want native menu-bar status, permissions management, local node capabilities, and a smoother first-run experience.

Typical flow:

1. Install and launch OpenClaw.app from the official release path.
2. Complete the permissions checklist.
3. Choose Local mode if the Gateway should run on this Mac.
4. Install the global CLI if you want terminal access.
5. Verify from the terminal:

```
openclaw gateway status
openclaw doctor
```

The macOS app owns platform permissions such as Notifications, Accessibility, Screen Recording, Microphone, Speech Recognition, Camera, and Automation. It can run or attach to a

local Gateway. In remote mode, it connects to a Gateway elsewhere and can expose local Mac capabilities as a node.

Option B: CLI Installer

Use the installer when you want the terminal route:

```
curl -fsSL https://openclaw.ai/install.sh | bash
```

To install without launching onboarding:

```
curl -fsSL https://openclaw.ai/install.sh | bash -s -- --no-onboard
```

Then run:

```
openclaw onboard --install-daemon
openclaw gateway status
openclaw doctor
```

If the LaunchAgent is not installed or needs repair:

```
openclaw gateway install
openclaw gateway restart
openclaw gateway status
```

Install on Linux

For a local Linux workstation, use the installer:

```
curl -fsSL https://openclaw.ai/install.sh | bash
```

Then:

```
openclaw onboard --install-daemon
openclaw gateway status
openclaw doctor
```

Linux installs normally use a systemd user service. On a headless or always-on machine, user services can stop when the user logs out unless lingering is enabled:

```
sudo loginctl enable-linger "$USER"
openclaw gateway install
systemctl --user status openclaw-gateway.service --no-pager
```

For multi-user or production-like servers, consider a system service instead of a user service. Keep one lifecycle owner for the Gateway. Do not let a system unit and a user unit both manage the same profile and port.

Install on Windows

Windows has three useful paths: Windows Hub, native PowerShell CLI, and WSL2.

Option A: Windows Hub

Use Windows Hub if you want the native Windows desktop experience. The official Windows guide describes it as the recommended path for desktop users. It provides tray status, setup, chat, diagnostics, connection management, Windows node mode, and local MCP mode.

Typical flow:

1. Download the signed x64 or ARM64 companion installer from the official OpenClaw releases.
2. Launch OpenClaw Companion from the Start menu or tray.
3. Choose local setup if you want the app to provision an app-owned WSL Gateway.
4. Wait for the tray icon to show a healthy connection.
5. Open Command Center and confirm Gateway, pairing, node, and channel status.

This path does not mutate your existing Ubuntu WSL distro when it provisions the app-owned Gateway distro.

Option B: Native Windows CLI

Use PowerShell:

```
iwr -useb https://openclaw.ai/install.ps1 | iex
```

Verify:

```
openclaw --version
openclaw doctor
openclaw gateway status --json
```

Install the Gateway service:

```
openclaw gateway install
openclaw gateway status --json
```

If you only want a foreground CLI/Gateway run:

```
openclaw onboard --non-interactive --skip-health
openclaw gateway run
```

Option C: WSL2

Use WSL2 when you want the most Linux-compatible Gateway runtime on Windows.

Install or choose a distro:

```
wsl --install
wsl --list --online
wsl --install -d Ubuntu-24.04
```

Enable systemd inside WSL:

```
sudo tee /etc/wsl.conf >/dev/null <<'EOF'
[boot]
systemd=true
EOF
```

Restart WSL from PowerShell:

```
wsl --shutdown
```

Then install inside the WSL distro:

```
curl -fsSL https://openclaw.ai/install.sh | bash
openclaw gateway status
```

For headless WSL setups, enable lingering inside WSL:

```
sudo loginctl enable-linger "$(whoami)"
openclaw gateway install
```

Install with npm, pnpm, or Bun

If you already manage Node yourself, install the CLI package directly.

npm

```
npm install -g openclaw@latest
openclaw onboard --install-daemon
```

If `sharp` fails because of a globally installed libvips, the official install docs suggest:

```
SHARP_IGNORE_GLOBAL_LIBVIPS=1 npm install -g openclaw@latest
```

pnpm

```
pnpm add -g openclaw@latest
pnpm approve-builds -g
openclaw onboard --install-daemon
```

Run `pnpm approve-builds -g` because pnpm requires explicit approval for packages with build scripts.

Bun

```
bun add -g openclaw@latest
openclaw onboard --install-daemon
```

Bun is supported for the global CLI install path. Node remains the recommended Gateway runtime.

Install from Source

Use source only if you are contributing, debugging internals, or intentionally running a development checkout.

```
git clone https://github.com/openclaw/openclaw.git
cd openclaw
pnpm install
pnpm ui:build
```

```
pnpm build
pnpm link --global
openclaw onboard --install-daemon
```

You can also skip the global link and run commands from inside the repo:

```
pnpm openclaw setup
pnpm openclaw gateway --port 18789 --verbose
```

For active Gateway development, the official setup docs describe:

```
pnpm install
pnpm openclaw setup
pnpm gateway:watch
```

Keep personal config and workspace outside the repo:

```
~/.openclaw/openclaw.json
~/.openclaw/workspace/
```

That keeps updates and git operations from damaging local prompts, skills, sessions, and credentials.

Install with Docker

Use Docker if you want a containerized Gateway or a throwaway test environment. For a normal personal workstation, the official docs recommend the normal install flow instead.

Prerequisites:

- Docker Desktop or Docker Engine.
- Docker Compose v2.
- At least 2 GB RAM for image build.
- Enough disk for images and logs.

Clone the repo:

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

Build and run the local image:

```
./scripts/docker/setup.sh
```

To use the prebuilt image:

```
export OPENCLAW_IMAGE="ghcr.io/openclaw/openclaw:latest"
./scripts/docker/setup.sh
```

Open the Control UI:

```
http://127.0.0.1:18789/
```

The setup script writes a token to `.env` by default. Use that token in the browser settings unless you changed the container config to password auth.

Useful post-start command:

```
docker compose run --rm openclaw-cli dashboard --no-open
```

For channels:

```
# WhatsApp QR login
docker compose run --rm openclaw-cli channels login

# Telegram bot token
docker compose run --rm openclaw-cli channels add --channel telegram --token "$TELEG
RAM_BOT_TOKEN"

# Discord bot token
docker compose run --rm openclaw-cli channels add --channel discord --token "$DISCOR
D_BOT_TOKEN"
```

The Docker path is not just `docker run openclaw`. The official flow relies on repo scripts, Compose configuration, onboarding, token generation, and mounted state.

Install with Podman

Use Podman when you want a rootless container managed by your non-root user.

Prerequisites:

- Podman in rootless mode.
- OpenClaw CLI installed on the host.

- Optional systemd user service if using Quadlet.

Clone the repo:

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

One-time setup:

```
./scripts/podman/setup.sh
```

Start the Gateway container:

```
./scripts/run-openclaw-podman.sh launch
```

Run onboarding inside the container:

```
./scripts/run-openclaw-podman.sh launch setup
```

Then open:

```
http://127.0.0.1:18789/
```

Use the token from:

```
~/.openclaw/.env
```

To manage the container through the host CLI:

```
export OPENCLAW_CONTAINER=openclaw
openclaw gateway status --deep
openclaw doctor
openclaw channels login
```

For Quadlet-managed autostart on Linux:

```
./scripts/podman/setup.sh --quadlet
systemctl --user start openclaw.service
systemctl --user status openclaw.service
```

On headless hosts:

```
sudo loginctl enable-linger "$(whoami)"
```

Install with Nix

Use Nix if you want declarative setup, pinned dependencies, Home Manager integration, and rollback.

The official OpenClaw Nix page points to nix-openclaw as the source of truth.

High-level flow:

1. Install Nix if you do not have it.
2. Create a local flake using the nix-openclaw agent-first template.
3. Configure model and channel secrets.
4. Run:

```
home-manager switch
```

Nix-managed installs use deterministic behavior. When `OPENCLAW_NIX_MODE=1` is set, OpenClaw treats runtime config as immutable and disables self-mutation flows such as mutating updates, plugin installs, `doctor --fix`, and `openclaw config set` writes against the managed config. Agents and operators should edit the Nix source instead.

On macOS, enable Nix mode for the GUI app:

```
defaults write ai.openclaw.mac openclaw.nixMode -bool true
```

Use Nix if reproducibility matters more than installer convenience. Use the standard installer if you want the least operational ceremony.

Where and How to Run OpenClaw

After installation, you can run OpenClaw in three common modes.

Managed Daemon Mode

This is the recommended day-to-day setup.

```
openclaw onboard --install-daemon
openclaw gateway status
```

The Gateway starts at login or boot depending on platform and service configuration. Use this for a personal assistant, channels, scheduled workflows, and always-on operation.

Foreground Debug Mode

Use this when troubleshooting startup, config, credentials, ports, or channel behavior.

```
openclaw gateway stop
openclaw gateway --port 18789 --verbose
```

Or:

```
openclaw gateway run
```

If the Gateway refuses to start because local mode is not configured, run onboarding/setup instead of forcing it:

```
openclaw setup
```

For ad-hoc development only, the Gateway CLI supports unconfigured runs:

```
openclaw gateway --allow-unconfigured
```

Remote Access Mode

Prefer SSH tunnel, Tailscale, or another private network.

```
ssh -N -L 18789:127.0.0.1:18789 user@host
```

Then connect local clients to:

```
ws://127.0.0.1:18789
```

An SSH tunnel does not bypass Gateway authentication. Clients still need the configured token, password, or identity-bearing auth path.

First Verification Checklist

Run these after setup:

```
openclaw --version
openclaw gateway status
openclaw doctor
openclaw health
```

For deeper service discovery:

```
openclaw gateway status --deep --json
```

For logs:

```
openclaw logs --follow
```

For channel readiness:

```
openclaw channels status --probe
```

For Control UI:

```
openclaw dashboard --no-open
```

Default local URL:

```
http://127.0.0.1:18789/
```

Healthy expectations:

- The CLI is available on PATH.
- `openclaw doctor` does not report critical config or DM policy issues.
- Gateway status shows a running runtime.
- Connectivity probe passes.
- Channel probes match the channels you configured.
- Control UI requires the expected local auth token or password.

Send a First Agent Request

Once onboarding has created a configured agent, run a local smoke test:

```
openclaw agent --agent main --message "Create a five-item checklist for verifying th
is OpenClaw setup." --local
```

For a Gateway-backed run:

```
openclaw agent --agent main --session-key smoke-test --message "Summarize the curren
t OpenClaw status."
```

If you want structured output for scripts:

```
openclaw agent --agent main --session-key smoke-test --message "Return setup status
as JSON." --json
```

If your default agent is not named `main`, replace `main` with the configured agent ID.

Connect a Messaging Channel

OpenClaw can connect to many chat surfaces, including Telegram, WhatsApp, Slack, Discord, Signal, iMessage, Microsoft Teams, Google Chat, Matrix, LINE, WeChat, and others. Start with one channel and verify pairing before adding more.

Telegram

Create a bot token with BotFather, then:

```
openclaw channels add --channel telegram --token "$TELEGRAM_BOT_TOKEN"
openclaw channels status --probe
```

WhatsApp

Use QR login:

```
openclaw channels login
openclaw channels status --probe
```

Delivery Test

For a configured target:

```
openclaw message send --target +1234567890 --message "Hello from OpenClaw"
```

Use channel-specific docs for exact target shapes and account settings. Do not open inbound DMs broadly until pairing and allowlists are understood.

Security Baseline

OpenClaw is powerful because it can operate against files, shell commands, browsers, channels, sessions, and local nodes. That also makes its security boundary important.

Use this baseline:

- Keep the Gateway bound to loopback unless you have a specific remote access plan.
- Keep Gateway auth enabled, even for local clients.
- Use SSH tunnels, VPN, or Tailscale for remote access instead of public exposure.
- Treat inbound messages as untrusted input.
- Keep DM pairing and allowlists enabled for chat channels.
- Run `openclaw doctor` after setup and after updates.
- Prefer least-privilege tool profiles for shared or multi-user contexts.
- Use sandboxing for non-main sessions when exposing agent access beyond yourself.
- Do not run the Gateway as root.
- Do not store secrets in public repos, shared logs, prompts, or channel messages.
- Back up `~/.openclaw` carefully because it can contain sensitive state.

If you run OpenClaw on a VPS, publish only what must be reachable. A safe default is Gateway on loopback, access through SSH tunnel or Tailscale, and channel webhooks only when the official channel setup requires them.

Updating OpenClaw

Use the built-in updater first:

```
openclaw update
```

Preview before applying:

```
openclaw update --dry-run
```

Switch channels:

```
openclaw update --channel beta
openclaw update --channel dev
```

After updating:

```
openclaw doctor
openclaw gateway restart
openclaw health
```

If the updater fails on an npm install, re-run the installer:

```
curl -fsSL https://openclaw.ai/install.sh | bash -s -- --install-method npm
```

For manual npm recovery:

```
openclaw gateway stop
npm install -g openclaw@latest
openclaw gateway install --force
openclaw gateway restart
openclaw doctor
```

For source checkouts:

```
cd openclaw
git pull
pnpm install
pnpm build
openclaw gateway restart
```

For Docker or Podman installs, rebuild or pull the image, then restart the container or service. Do not expect `openclaw update` inside a container to replace the image cleanly.

Troubleshooting

`openclaw` command not found

Check Node, global npm prefix, and PATH:

```
node -v
npm prefix -g
echo "$PATH"
```

If the global npm bin directory is missing:

```
export PATH="$(npm prefix -g)/bin:$PATH"
```

Add that line to `~/.zshrc` or `~/.bashrc`, then open a new terminal.

Gateway will not start

Check:

```
openclaw gateway status --deep
openclaw doctor
openclaw logs --follow
```

Common causes:

- `gateway.mode=local` is missing.
- Another process is using port `18789`.
- Gateway auth is misconfigured.
- Service metadata points to an old Node or CLI path.
- The user service stopped after logout on Linux because lingering is disabled.

Repair common service drift:

```
openclaw doctor --fix
openclaw gateway install --force
openclaw gateway restart
```

Port conflict

Check what is listening:

```
lsof -i :18789
```

Run on a different port:

```
openclaw gateway --port 19001
```

If you change the configured Gateway port, reinstall or repair the service so launchd, systemd, or Scheduled Tasks start the correct port.

Windows local setup fails

Check the Windows Hub setup log:

```
notepad "$env:LOCALAPPDATA\OpenClawTray\Logs\Setup\easy-setup-latest.txt"
```

Common causes include disabled WSL, blocked virtualization, stale app-owned WSL state, or a network failure while installing the Gateway package.

Container setup fails

For Docker:

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

For Podman:

```
podman logs -f openclaw
podman ps
```

Common causes:

- Not running setup from the repo root.
- Insufficient RAM during image build.
- Host config/workspace directory ownership mismatch.
- Wrong container target for host CLI commands.
- SELinux bind-mount labeling issues on Linux.

Channel does not respond

Run:

```
openclaw channels status --probe
openclaw doctor
openclaw logs --follow
```

Check:

- Token validity.
- Bot/channel permissions.
- Pairing status.
- Allowlist and DM policy.
- Gateway reachability.

- Whether the channel expects a webhook or long-running connection.

Recommended First-Day Runbook

1. Install using the platform path that matches where the Gateway should live.
2. Run onboarding with daemon installation.
3. Verify CLI, doctor, Gateway status, and health.
4. Open the Control UI locally.
5. Send one local agent smoke test.
6. Configure one messaging channel.
7. Run channel probe.
8. Confirm pairing/allowlist behavior.
9. Review `~/.openclaw` backup and secret handling.
10. Run `openclaw update --dry-run` so you understand the update path before you need it.

Minimal Setup Recipes

macOS or Linux Local

```
curl -fsSL https://openclaw.ai/install.sh | bash
openclaw onboard --install-daemon
openclaw doctor
openclaw gateway status
openclaw agent --agent main --message "Verify this install." --local
```

Windows Native CLI

```
iwr -useb https://openclaw.ai/install.ps1 | iex
openclaw gateway install
openclaw doctor
openclaw gateway status --json
```

WSL2

```
curl -fsSL https://openclaw.ai/install.sh | bash
openclaw onboard --install-daemon
```

```
sudo loginctl enable-linger "$(whoami)"
openclaw gateway status
```

Docker

```
git clone https://github.com/openclaw/openclaw.git
cd openclaw
export OPENCLAW_IMAGE="ghcr.io/openclaw/openclaw:latest"
./scripts/docker/setup.sh
docker compose run --rm openclaw-cli dashboard --no-open
```

Podman

```
git clone https://github.com/openclaw/openclaw.git
cd openclaw
./scripts/podman/setup.sh
./scripts/run-openclaw-podman.sh launch setup
export OPENCLAW_CONTAINER=openclaw
openclaw gateway status --deep
```

Source Checkout

```
git clone https://github.com/openclaw/openclaw.git
cd openclaw
pnpm install
pnpm ui:build
pnpm build
pnpm link --global
openclaw onboard --install-daemon
```

Final Recommendation

For most users, start with the official installer on macOS/Linux/WSL2 or Windows Hub on Windows. Use Docker or Podman only when you specifically want an isolated Gateway. Use Nix only when declarative reproducibility is part of your operating model. Use source only for development or debugging.

The fastest successful setup is not the one with the fewest commands. It is the one where the Gateway is supervised, authenticated, verified, easy to update, and scoped to the trust boundary you actually intend.

ShShell.com
https://shshell.com