THE COMPLETE

OpenClaw

Deployment Guide

v2.0 — Updated Edition

Deploy your AI. Own your stack.

Reliable installs across macOS, Windows, Docker, Linux, and VPS.
Plus hardening, persistence, workspace architecture, and real-world troubleshooting.

By Zero Signal

Release Date: February 2026

What's New in v2.0

- **Node 22+ requirement** — updated from Node 18/20. The installer script now handles detection and installation automatically.
- **ClawDock helper system** — full section on the Docker management helpers that simplify day-to-day container operations.
- **One-liner installer script** — the new recommended install method for all platforms.
- **Workspace architecture deep-dive** — complete guide to SOUL.md, AGENTS.md, USER.md, TOOLS.md, IDENTITY.md, HEARTBEAT.md, and MEMORY.md.
- **Multi-model routing overview** — how to configure different AI models for different roles (orchestrator, coder, reviewer).
- **Ollama integration** — run local models for zero-cost operations like heartbeats.
- **Expanded troubleshooting** — 25+ real-world gotchas with verified solutions.
- **Security hardening checklist** — credential hygiene, file permissions, network exposure.
- **Cost optimization hints** — model routing, prompt caching, heartbeat offloading.
- **Multi-agent routing** — running multiple agents on one gateway for different channels or contacts.

Table of Contents

Note: Update the table of contents after opening in Word (right-click > Update Field > Update Entire Table).

1. Overview and Core Concepts

OpenClaw is a self-hosted AI gateway that runs on your hardware and routes requests to model providers (OpenRouter, Anthropic, OpenAI, Ollama, and others). It integrates with messaging channels like Telegram, WhatsApp, Discord, and Slack, and can run automations, tools, and skills autonomously.

Key Components

- **CLI** — the openclaw command used for setup, configuration, and management.
- **Gateway** — the long-running Node.js service process that handles conversations, tool execution, and integrations. This is the brain.
- **Web UI (Dashboard)** — an optional browser-based interface to interact with your agent, manage devices, and monitor activity.
- **Workspace** — the on-disk directory containing configuration, personality files, skills, memory, and project files. This is where your agent lives.

Default Ports

Port	Service	Notes
3000	Web UI	Browser access (non-Docker native installs)
18789	Gateway	Core service (WebSocket + HTTP). Docker and Compose deployments.
18790	Bridge	Internal communication between CLI and Gateway.
11434	Ollama	Local LLM server (if installed).

> **Port conflicts**
> If a port is already in use, OpenClaw will fail to start. Use lsof -i :<port> (macOS/Linux) or netstat -ano | findstr :<port> (Windows) to find the conflicting process.

How It Works

OpenClaw runs as a single Node.js process on your machine, listening on 127.0.0.1:18789 by default. When you send a message through Telegram, WhatsApp, or the Web UI, the Gateway receives it, loads your workspace context (personality files, memory, skills), routes it to the configured AI model, and executes any tool calls the model requests. The model's response flows back through the same channel.

Everything is file-based. No external databases. No Redis. No vector stores (unless you opt in). Just Markdown files and SQLite. Your agent reloads context from workspace files every time it processes a message.

2. Install Decision Tree

Choose the install path that matches your goals and hardware:

Path	Best For	Complexity
macOS native	Developers on Mac. Simplest workflow.	Low
Windows + WSL2	Recommended for Windows. Fewer edge cases.	Low–Medium
Windows native	PowerShell-only. Handle PATH and execution policy.	Medium
Docker (ClawDock)	Isolation and easy upgrades. Recommended for servers.	Medium
Kali Linux	Authorized security labs only.	Medium
VPS (headless)	24/7 operation. Ubuntu/Debian cloud server.	Medium–High

Decision Shortcuts

- Want the **easiest cross-platform reproducibility**? Docker Compose with ClawDock helpers.
- Want the **best Windows experience**? WSL2 + Ubuntu.
- Need **full host access** and native tooling? Native install.
- Running a **dedicated server** or home lab? Docker on VPS.
- Want to **save money** with local models? Any path + Ollama.

Important

OpenClaw now requires Node 22+. The installer script handles detection and installation automatically. If installing manually, verify your Node version first: node --version

3. macOS Installation

Prerequisites: macOS Ventura (13) or newer. The installer script handles Node.js installation.

Recommended: Installer Script

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

This handles Node detection, CLI installation, and launches the onboarding wizard in one step.

Manual: Homebrew + npm

```
brew install node
node --version   # Must be 22+
npm install -g openclaw
openclaw onboard --install-daemon
```

Start and Verify

```
openclaw gateway start
openclaw doctor         # Check for config issues
openclaw status         # Gateway status
openclaw dashboard      # Open browser UI
```

npm permission errors

If you get global npm permission errors, set a user-local prefix instead of using sudo: mkdir -p ~/.npm-global && npm config set prefix ~/.npm-global && add ~/.npm-global/bin to your PATH.

4. Windows Installation

Path A: WSL2 (Recommended)

WSL2 reduces PATH and permissions edge cases and gives you a Linux environment.

```
# Run PowerShell as Administrator
wsl --install
```

Inside Ubuntu (WSL2):

```
sudo apt update && sudo apt upgrade -y
curl -fsSL https://openclaw.ai/install.sh | bash
```

The installer script handles Node 22 installation inside WSL2 automatically.

Path B: Native Windows (PowerShell)

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

Or manually:

1. Install Node.js 22 LTS from nodejs.org. Ensure **Add to PATH** is checked.
2. Set execution policy if PowerShell blocks npm scripts:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

3. Install and onboard:

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

Windows Firewall (LAN Access Only)

Only needed if accessing the Web UI from other devices on your network:

```
New-NetFirewallRule -DisplayName "OpenClaw Gateway" -Direction Inbound -
Protocol TCP -LocalPort 18789 -Action Allow
```

WSL2 browser access

If the browser can't reach localhost from WSL2, use the WSL IP address or ensure port forwarding is active. Run: ip addr show eth0 | grep inet

5. Docker Installation + ClawDock

Docker is recommended for isolation and repeatable deployments. The ClawDock helper system makes day-to-day Docker management painless.

Prerequisites

- Docker Desktop (Windows/Mac) or Docker Engine (Linux)
- Docker Compose v2
- Git (to clone the repository)

Quick Start

```
git clone https://github.com/openclaw/openclaw.git
cd openclaw
./docker-setup.sh
```

The setup script creates two important directories on your host:

- **~/.openclaw/** — configuration, memory, API keys, agent settings
- **~/.openclaw/workspace/** — working directory accessible to the agent

Then build and start:

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

Install ClawDock Helpers

ClawDock is a set of shell functions that wrap common Docker Compose commands. Install once:

```
mkdir -p ~/.clawdock && curl -sL
https://raw.githubusercontent.com/openclaw/openclaw/main/scripts/shell-
helpers/clawdock-helpers.sh -o ~/.clawdock/clawdock-helpers.sh
```

Add to your shell profile (.bashrc, .zshrc, or Git Bash profile):

```
source ~/.clawdock/clawdock-helpers.sh
```

ClawDock Command Reference

Command	Purpose
clawdock-start	Start the gateway container
clawdock-stop	Stop the gateway
clawdock-restart	Restart the gateway
clawdock-status	Check container status
clawdock-logs	View live logs (follows)

clawdock-shell	Shell into the container
clawdock-dashboard	Open Web UI in browser
clawdock-devices	List device pairings
clawdock-approve <id>	Approve a device pairing request
clawdock-fix-token	Reconfigure gateway token
clawdock-rebuild	Rebuild the Docker image
clawdock-health	Run a health check
clawdock-token	Show the gateway auth token
clawdock-clean	Remove containers and volumes (nuclear option)

First-Time Setup (Docker)

4. **clawdock-start** — Start the gateway
5. **clawdock-fix-token** — Configure the gateway token
6. **clawdock-dashboard** — Open the Web UI
7. **clawdock-devices** — Check if pairing is needed
8. **clawdock-approve <id>** — Approve the browser device

Windows Docker Bind Mounts

Host paths must be bound to container paths. A bare host path without a target is treated as a volume name and will fail on Windows.

```
# CORRECT (Windows bind mount)
volumes:
  - type: bind
    source: D:/OpenClaw_Shared/openclaw_share
    target: /shared
```

Permission errors

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

Persistence

Workspace files persist across container restarts via volume mounts. Everything else inside the container resets. Key persistent locations:

- **/home/node/.openclaw/workspace/** — your agent's personality, memory, skills, projects

- **/home/node/.openclaw/openclaw.json** — main configuration file
- **/home/node/.openclaw/credentials/** — API keys

Home volume persistence

To persist the entire /home/node directory across container recreation, set OPENCLAW_HOME_VOLUME before running docker-setup.sh. This creates a named Docker volume mounted at /home/node.

6. Kali Linux / Security Lab Setup

> **Authorized use only**
> This section is for authorized security testing only — your own systems or systems you have explicit written permission to test.

```
sudo apt update && sudo apt upgrade -y
curl -fsSL https://openclaw.ai/install.sh | bash
```

Or manually:

```
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt install -y nodejs
sudo npm install -g openclaw
openclaw onboard --install-daemon
openclaw gateway start
```

Operational Guidance

- Keep work in a **dedicated lab workspace**. Do not mix personal tokens on shared machines.
- **Log all outputs.** Keep evidence and scope notes for every engagement.
- **Rate-limit and scope all scans.** Be explicit about targets and permissions.
- Use a **separate API key** with spending limits for lab work.

7. VPS Deployment (Headless)

For 24/7 operation, deploy to an Ubuntu/Debian VPS. Docker is recommended for clean upgrades and isolation.

Native Install

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

Or manually:

```
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt install -y nodejs
sudo npm install -g openclaw
mkdir -p /opt/openclaw-workspace
cd /opt/openclaw-workspace
openclaw onboard --install-daemon
```

Reverse Proxy + TLS

- Terminate TLS at **nginx** and proxy to the localhost-only OpenClaw port.
- Use strong security headers and restrict access via VPN or IP allowlist.
- Never expose the gateway port (18789) directly to the public internet.

Recommended VPS providers

Hetzner, DigitalOcean, and Railway all have documented OpenClaw deployment guides. Check docs.openclaw.ai/install for platform-specific instructions.

8. Workspace Architecture

Your agent wakes up with amnesia every session. The workspace files are its brain. Understanding this architecture is the difference between a generic chatbot and a personalized AI operator.

Directory Structure

```
~/.openclaw/workspace/
├── AGENTS.md          # Operating manual (REQUIRED)
├── SOUL.md            # Core personality
├── IDENTITY.md        # External identity (name, emoji, role)
├── USER.md            # About the human (you)
├── TOOLS.md           # Local tool notes and environment cheat sheet
├── MEMORY.md          # Long-term curated memory
├── HEARTBEAT.md       # Periodic check-in tasks
├── memory/            # Daily logs
│   ├── 2026-02-24.md
│   └── 2026-02-25.md
├── skills/            # Custom skills
└── templates/         # Project templates
```

File Roles

File	Purpose	Loaded When
AGENTS.md	Operating manual. Boot sequence, rules, delegation protocol.	Every session (required)
SOUL.md	Core personality. Who the agent IS at its foundation.	Every session
IDENTITY.md	External identity: name, emoji, role, communication class.	Every session
USER.md	About you. Goals, preferences, context the agent needs.	Every session
TOOLS.md	Environment cheat sheet. Ports, paths, gotchas, model routing.	Every session
MEMORY.md	Curated long-term insights. Lessons, preferences, project learnings.	Main sessions only
HEARTBEAT.md	What to check during periodic heartbeat polls.	Heartbeat events
memory/*.md	Daily logs. Raw notes of what happened each session.	Today + yesterday

Boot Sequence

Every session, your agent loads context in this order:

9. Read **SOUL.md** — who it is
10. Read **IDENTITY.md** — name and external persona
11. Read **USER.md** — who it's helping
12. Read **AGENTS.md** — operating rules and protocols
13. Read **TOOLS.md** — environment context
14. Read **memory/today.md + yesterday.md** — recent context
15. If main session: also read **MEMORY.md** (long-term memory)
16. Check **HEARTBEAT.md** for periodic tasks

Key principle

If it's not written down, it didn't happen. The agent's mental notes die when the session ends. Files survive. Always write decisions, lessons, and blockers to the daily log or relevant workspace file.

9. Multi-Model Routing

One of OpenClaw's most powerful features is the ability to route different tasks to different AI models. You're not locked to one LLM — configure Claude for complex reasoning, Grok for fast orchestration, a local model for cheap operations.

How Model Routing Works

Your openclaw.json configuration defines a primary model and fallbacks. The primary handles most conversations. Fallbacks activate when the primary is unavailable or rate-limited.

Beyond the default, you can configure aliases that let you switch models mid-conversation with slash commands like /model sonnet or /model grok.

Routing Strategy: Orchestrator, Coder, Reviewer

Advanced users configure three-tier model routing for autonomous project work:

Role	Responsibility	Model Characteristics
Orchestrator	Planning, delegation, coordination. Never writes code.	Fast, cheap, good at reasoning and task decomposition.
Coder	All coding, implementation, building, debugging.	Strong at code generation. Handles complex architectures.
Reviewer	Code review, spec compliance, quality assurance.	Thorough analysis. Good at finding bugs and gaps.

The orchestrator receives your task, breaks it into phases, and spawns subagents (coder and reviewer) for each phase. Subagents are isolated sessions that execute one task and report back.

> **Advanced topic**
> Full multi-model delegation protocol configuration, subagent routing rules, and project execution templates are covered in the Advanced Operations Guide (available separately).

Ollama: Local Models for Zero-Cost Operations

Ollama is a local LLM runtime that runs models directly on your hardware. OpenClaw integrates with Ollama's API, enabling you to route cheap operations (like heartbeats) to a free local model while keeping expensive tasks on cloud providers.

Quick Setup

17. Install Ollama from ollama.ai (macOS/Linux) or the Windows installer.
18. Pull a lightweight model:

```
ollama pull llama3.2:3b
```

19. Register Ollama as a provider in openclaw.json under models.providers.ollama.
20. Set the baseUrl. For Docker deployments, use http://host.docker.internal:11434 to reach the host machine.

Cost savings

Heartbeats run frequently (every 1–2 hours). Routing them to a local model eliminates hundreds of API calls per month at zero cost. Even a 3B parameter model is sufficient for heartbeat health checks.

10. Advanced Usage

Sub-Agent Delegation

OpenClaw can spawn specialized subagents for isolated work. The main agent receives a task, breaks it down, and delegates to workers:

- **Main agent** receives task and plans execution
- **Spawns coder subagent** (can be a different, specialized model) for implementation
- **Spawns reviewer subagent** for quality checks
- **Merges results** and reports back

Context fragmentation

Subagents wake up with ZERO context. They don't see the main agent's conversation history, workspace files, or what other subagents are doing. The orchestrator must pass full specifications in every spawn task. Summarizing or abbreviating specs is the #1 cause of subagent failures.

Cron and Scheduled Tasks

OpenClaw supports scheduling tasks via cron configuration. Combined with heartbeats, your agent can proactively check for events, send reminders, run maintenance, and more — without you sending a message.

Multi-Channel Setup

Connect multiple messaging platforms simultaneously:

- **Telegram** — create a bot via @BotFather, provide the token during setup
- **WhatsApp** — pair via QR code (use a dedicated number, not your personal one)
- **Discord** — create a bot application, provide the token and guild ID
- **Slack** — create an app with appropriate scopes

Dedicated number for WhatsApp

When you connect WhatsApp, EVERY message becomes agent input — including your friends, family, and bank verification codes. Always use a dedicated number for the assistant.

Multi-Agent Routing

A single Gateway can host multiple agents, each with its own workspace, personality, memory, and session store. Route different channels, contacts, or groups to different agents:

- **Personal agent** for DMs with casual tone and calendar access

- **Work agent** for team channels with formal style and documentation access
- **Alerts agent** for monitoring with restricted tools and allowlisted contacts

Agents are configured via agents.list in openclaw.json with bindings that map channels and peers to specific agent IDs.

API Key Management

- Store keys in **environment variables** or .env files. Never hardcode.
- Add **.env** to .gitignore. Commit .env.example instead.
- **Rotate keys** regularly and immediately after suspected exposure.
- Use **least-privilege scopes** where providers support it.
- Set **daily spending limits** on your API provider dashboard.

11. Security Hardening

OpenClaw is powerful because it has real access to your filesystem, terminal, APIs, and messaging channels. That capability makes security non-optional.

Hardening Checklist

Priority	Action	Why
P0	Remove credentials from git remote URLs	PATs in .git/config are readable by any agent or process
P0	Use git credential manager (encrypted storage)	Replaces plaintext token storage
P0	Lock down openclaw.json file permissions (600)	Prevents unauthorized reading of gateway/Telegram tokens
P1	Rotate exposed tokens immediately	Any token that was in a readable file should be considered compromised
P1	Verify backup status and document restore procedure	Workspace files are your agent's brain — losing them means starting over
P1	Add .env and credentials/ to .gitignore	Prevents accidental commits of secrets
P2	Enable gateway log persistence	Audit trail of all agent activity survives container restarts
P2	Schedule periodic health scans	Catch drift and new exposures before they become problems
P2	Set daily API spending limits	Prevents runaway costs from agent loops or prompt injection

Network Exposure

By default, the gateway binds to loopback (127.0.0.1) — only accessible from the local machine. This is the safest configuration.

- **Never expose port 18789 to the public internet** without a reverse proxy and authentication.
- If accessing from your LAN, bind to 0.0.0.0 but ensure your router firewall blocks external access.
- For remote access, use **Tailscale** (built into OpenClaw) or an SSH tunnel.

Agent Boundaries

Configure workspace rules so agents know their limits:

- Agents should **never read openclaw.json** directly (it contains tokens)
- Agents should **never output credentials** in chat, files, or logs
- Agents should **never modify files outside the workspace** directory
- Agents should use **trash** over rm for deletions
- Agents should **git commit before risky operations** as a safety checkpoint

Advanced hardening

Full security hardening implementation, including automated audit scripts, token rotation procedures, and agent sandboxing configuration, is covered in the Advanced Operations Guide.

12. Troubleshooting and Diagnostics

Fast Diagnostics

```
node --version            # Must be 22+
openclaw --version        # CLI version
openclaw doctor           # Auto-diagnose config issues
openclaw status           # Gateway status
openclaw gateway logs     # Gateway logs (native)
clawdock-logs             # Gateway logs (Docker)
```

Common Failure Patterns

Problem	Likely Cause	Fix
Gateway won't start	Wrong Node version, port conflict, or invalid config	Check node --version (need 22+). Check port: lsof -i :18789
API calls failing	Missing API key, wrong provider, network issues	Verify key in credentials dir. Test: curl https://openrouter.ai/api/v1/models
Memory not persisting	Workspace not mounted or not writable	Docker: check volume mounts. Native: check directory permissions
Container restart loop	Invalid config key in openclaw.json	Check clawdock-logs. Common: agents.defaults.named key crashes gateway
Permission denied on CLI	Binary lacks execute permission or not in PATH	chmod +x $(which openclaw) or reinstall. In Docker: use clawdock commands instead
WSL2 can't reach localhost	Port forwarding not active	Use WSL IP: ip addr show eth0 | grep inet
Dashboard shows 404 on routes	SPA routing not configured on host	Add rewrites config (vercel.json or nginx try_files)
Heartbeat using wrong model	heartbeat.model override bug (#9556)	Known issue. Monitor logs. Primary model is used regardless of config
Subagent uses wrong model	Model pin ignored in spawn (#10963)	Retry with /model command before spawn. Check clawdock-logs
Gateway origin error	Non-loopback bind without allowedOrigins	Use ClawDock setup (handles automatically). Don't manually edit compose
Docker can't reach host services	Container localhost ≠ host localhost	Use host.docker.internal instead of localhost or 127.0.0.1
Ollama port bind error	Ollama already running (auto-starts on boot)	Don't run ollama serve. It's already running. Verify: ollama list

Docker-Specific Diagnostics

```
docker compose ps                    # Container status
```

```
docker compose logs -f openclaw-gateway   # Live logs
clawdock-health                       # Run health check
clawdock-shell                        # Shell into container
```

Config Validation

The most common cause of gateway crash loops is an invalid key in openclaw.json. Known problematic keys:

- **agents.defaults.named** — NOT supported. Causes immediate crash. Use session spawning instead.
- **heartbeat.model** — accepted but may be ignored (bug #9556).
- **cache.enabled / cache.ttl** — NOT real config keys. Documented in some third-party guides but not supported by OpenClaw.

Validate before restart

After editing openclaw.json, run: openclaw doctor (native) or clawdock-cli doctor (Docker). This catches config errors before they crash the gateway.

13. Cost Optimization

Running OpenClaw 24/7 with cloud AI models can get expensive fast. Here are the key levers for reducing costs while maintaining capability.

Model Routing by Task

Don't use your most expensive model for everything. Route tasks to the cheapest model that can handle them:

- **Expensive models** (Opus, GPT-4) — complex reasoning, long-form analysis, critical decisions
- **Mid-tier models** (Sonnet, Grok) — coding, orchestration, most daily tasks
- **Cheap models** (Haiku, small Ollama models) — heartbeats, simple lookups, routine checks

Heartbeat Offloading

Heartbeats run every 1–2 hours. If they use your primary model, that's hundreds of API calls per month on routine checks. Route heartbeats to a local Ollama model for zero cost.

Prompt Caching

When using Anthropic models (directly or via OpenRouter), prompt caching gives a 90% discount on repeated content. Your workspace files (SOUL.md, AGENTS.md, USER.md, etc.) are sent with every request — keeping them stable means they get cached automatically.

- **Don't edit personality files mid-session** — every edit invalidates the cache
- **Batch subagent spawns within 5 minutes** — second spawn reuses the cached system prompt
- **Keep HEARTBEAT.md small** — it's read on every heartbeat cycle

Context Management

The biggest hidden cost is context bloat. Long conversations accumulate tokens, and every message re-sends the full context.

- **Compaction mode: safeguard** — recommended. Summarizes older turns when context hits the threshold.
- **Reset sessions periodically** — daily or weekly resets keep session files from growing too large.
- **Separate stable from dynamic content** — reference docs in cached files, daily notes in uncached files.

Full cost optimization

Detailed token analysis, model-by-model cost comparisons, and advanced caching strategies are available in the Cost Optimization supplement.

Appendix A: Command Cheat Sheet

Command	Purpose
openclaw onboard	Run the setup wizard
openclaw gateway start	Start the gateway service
openclaw gateway restart	Restart the gateway
openclaw gateway logs	Show gateway logs
openclaw doctor	Check for config issues
openclaw status	Show gateway status
openclaw dashboard	Open browser UI
openclaw config get	Show current configuration
openclaw config set KEY VALUE	Update a config value
openclaw channels login	Connect a messaging channel
openclaw devices list	List paired devices
openclaw devices approve <id>	Approve a device pairing
openclaw --version	Show CLI version

ClawDock Commands (Docker)

Command	Purpose
clawdock-start	Start the gateway container
clawdock-stop	Stop the gateway
clawdock-restart	Restart the gateway
clawdock-logs	View live logs
clawdock-shell	Shell into the container
clawdock-dashboard	Open the Web UI
clawdock-health	Run a health check
clawdock-fix-token	Reconfigure gateway token
clawdock-rebuild	Rebuild the Docker image
clawdock-clean	Remove containers and volumes

Appendix B: Recommended Versions and Ports

Component	Minimum	Recommended
Node.js	22.x	22.x LTS (latest)
npm	10.x	Latest stable
Docker	24.x	Latest stable
Docker Compose	v2	Latest stable
Ubuntu	22.04 LTS	24.04 LTS
macOS	Ventura (13)	Sonoma (14) or newer
Ollama	0.1.x	Latest stable

Default Ports

Port	Service	Bind
3000	Web UI (native installs)	localhost
18789	Gateway (Docker/Compose)	loopback (127.0.0.1)
18790	Bridge (internal)	local
11434	Ollama (local LLM)	localhost

Built by Zero Signal. Deployed by you.

v2.0 — Updated Edition — February 2026