OPENCLAW

Platform Deployment Guide

Mac vs PC vs VPS

Configuration, Use Cases & Troubleshooting

ScaleUP Media | February 2026

Executive Summary

OpenClaw can run on macOS, Windows, or a remote VPS. Each platform has distinct advantages, tradeoffs, and failure modes. This guide breaks down every platform across five dimensions: ideal use cases, installation and configuration, common issues and fixes, performance characteristics, and cost considerations. Whether you're running a single instance for personal automation or deploying production workloads for clients, this document tells you exactly what to expect.

Quick Comparison Matrix

Factor	Mac (macOS)	PC (Windows)	VPS (Linux)
Best For	Solo dev, testing, content creation workflows	Budget builds, gaming PC repurpose, local dev	Production, 24/7 uptime, client deployments, scaling
Uptime	Only when laptop is open/awake	Only when PC is on and not sleeping	24/7/365 without intervention
Setup Difficulty	Easy – native Unix, Homebrew	Medium – WSL2 or native Node quirks	Medium – SSH, Linux admin basics needed
Monthly Cost	$0 (your hardware + electricity)	$0 (your hardware + electricity)	$5–$50/mo depending on provider and specs
Performance	Good. M-series chips are fast	Good. Depends on hardware specs	Consistent. Dedicated resources, no throttling
Scaling	Limited to one machine	Limited to one machine	Spin up multiple instances easily
Security Risk	Low – macOS sandboxing helps	Medium – more attack surface	Varies – you own the hardening

---

Mac (macOS)

Ideal Use Cases

- **Personal automation workflows:** Running OpenClaw as your daily driver for content generation, social media scheduling, email sequences, and research tasks.
- **Development and testing:** Building and iterating on OpenClaw configurations before deploying to production on a VPS.

- **Content creators:** If you're creating TikTok/IG content around AI and need quick local access to demo or screen record OpenClaw in action.
- **Light client work:** Running a single client's automation during business hours when you're at your desk.

Configuration Setup

1. **Install prerequisites:** Install Homebrew if not already installed, then run: brew install node git. Verify with node -v (need v18+) and git --version.
2. **Clone the repo:** git clone [repo-url] openclaw && cd openclaw
3. **Install dependencies:** npm install (or yarn install). This is typically clean on Mac with no native build issues.
4. **Configure environment:** Copy .env.example to .env. Add your API keys (Anthropic, OpenAI, etc.). Set your model routing preferences (85% Haiku / 15% Sonnet for cost optimization).
5. **Run OpenClaw:** npm start or node index.js. Access the dashboard at localhost on your configured port.
6. **Prevent sleep:** Use caffeinate -s in a separate terminal window or install Amphetamine from the Mac App Store to keep your machine awake during long-running tasks.

Common Issues & Fixes

- **Sleep/lid close kills processes:** macOS aggressively sleeps when you close the lid. Use caffeinate, Amphetamine, or pmset via Terminal to prevent this. This is the #1 gotcha for Mac users.
- **Port conflicts:** If you're running other Node apps or dev servers, port collisions happen. Change the port in .env or use lsof -i :[port] to find and kill conflicting processes.
- **Node version mismatches:** If you have an older Node from a previous project, use nvm (Node Version Manager) to switch: nvm install 18 && nvm use 18.
- **Rosetta issues on M-series:** Some npm packages with native binaries may need Rosetta 2 on Apple Silicon. Run softwareupdate --install-rosetta if you see architecture errors during npm install.
- **File permission errors:** macOS may block access to certain directories. Avoid running with sudo. Instead, fix permissions: chown -R $(whoami) ~/.npm.
- **Network timeouts to APIs:** Mac's built-in firewall or Little Snitch / Lulu can block outbound connections to Anthropic/OpenAI endpoints. Whitelist the API domains.

Performance Notes

Apple Silicon Macs (M1/M2/M3/M4) are excellent for running OpenClaw locally. The unified memory architecture means Node.js runs very efficiently with minimal swapping. An M1 MacBook Air with 8GB RAM can comfortably handle a single OpenClaw instance. For

multi-instance setups, 16GB+ is recommended. Battery life is a real concern for laptop users – OpenClaw making continuous API calls will drain faster than typical browsing. Keep it plugged in for long sessions.

PC (Windows)

Ideal Use Cases

- **Budget-conscious developers:** If you already have a Windows desktop or gaming PC, there's no reason to buy a Mac. Your existing hardware works fine.
- **Dual-purpose machines:** Gaming PC by night, OpenClaw workhorse by day. Windows lets you do both without dedicated hardware.
- **Local development before VPS:** Build and test your configurations locally before deploying to a Linux VPS. WSL2 gives you an almost-identical Linux environment.
- **Teams on mixed hardware:** If your team or mentees use Windows, you need to know the Windows workflow inside and out.

Configuration Setup

You have two paths on Windows. WSL2 (recommended) gives you a real Linux environment inside Windows and avoids most Windows-specific issues. Native Windows works but has more edge cases.

Path A – WSL2 (Recommended):

1. **Enable WSL2:** Open PowerShell as Admin and run: wsl --install. This installs Ubuntu by default. Restart your machine.
2. **Set up the Linux environment:** Open Ubuntu from Start menu. Run: sudo apt update && sudo apt install -y nodejs npm git. Verify versions.
3. **Clone and install:** Same as Mac from here. git clone, npm install, configure .env, npm start. You're running real Linux now.

Path B – Native Windows:

4. **Install Node.js:** Download the LTS installer from nodejs.org. Use the default settings. This also installs npm.
5. **Install Git:** Download Git for Windows. During install, select "Use Git from the Windows Command Prompt" and "Checkout as-is, commit Unix-style line endings".

6. **Clone and install:** Use PowerShell or Git Bash. Same commands: git clone, npm install, configure .env, npm start.

Common Issues & Fixes

- **Line ending nightmares (CRLF vs LF):** Windows uses CRLF line endings. Linux uses LF. If you clone on Windows natively and later move to a VPS, config files can break. Fix: git config --global core.autocrlf input. For WSL2 users, always clone inside the WSL filesystem, not /mnt/c/.
- **Path length limits:** Windows has a 260-character path limit by default. Deep node_modules folders hit this. Fix: Enable long paths via Group Policy or registry: HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\LongPathsEnabled = 1.
- **Windows Defender slowing npm install:** Defender scans every file npm creates, making installs 3–10x slower. Add your project directory and node_modules to Defender's exclusion list.
- **WSL2 memory hogging:** WSL2 can consume all available RAM if unchecked. Create a .wslconfig file in your Windows user directory to cap memory: [wsl2] memory=4GB.
- **Firewall blocking API calls:** Windows Firewall may block Node.js outbound connections on first run. Click "Allow access" when prompted, or manually add a firewall rule for node.exe.
- **Sleep and power settings:** Like Mac, Windows will sleep and kill your processes. Go to Settings > Power > set "Sleep" to Never when plugged in. For desktops, disable hibernate.
- **npm global install permission errors:** On native Windows, global npm installs sometimes fail. Run PowerShell as Administrator, or better yet, use npx instead of global installs.
- **WSL2 clock drift:** If your WSL2 instance is suspended (laptop sleep), the clock can drift, causing API authentication failures with timestamp-sensitive endpoints. Fix: sudo hwclock -s inside WSL after waking.

Performance Notes

Windows performance depends entirely on your hardware. A modern desktop with 16GB+ RAM and an SSD runs OpenClaw with zero issues. The main bottleneck is never your local machine – it's API response times and rate limits. WSL2 adds minimal overhead (typically less than 5%) compared to native Linux. The one area where Windows falls short is filesystem performance inside WSL2 when accessing files on the Windows filesystem (/mnt/c/). Always keep your project files inside the WSL2 Linux filesystem for best performance.

VPS (Linux Server)

Ideal Use Cases

- **Production deployments:** Any automation that needs to run 24/7 without depending on your laptop being open. Client work, scheduled tasks, monitoring – VPS is the only real answer.
- **Client-facing systems:** Deploying OpenClaw instances for clients like the 1,200-location restaurant chain. You need guaranteed uptime and dedicated resources.
- **Multi-instance scaling:** Running multiple OpenClaw instances for different products or clients. Easy to spin up additional VPS instances as needed.
- **Team access:** Multiple team members can SSH in and manage the same deployment. No dependency on one person's laptop.
- **Security isolation:** API keys live on the server, not on your personal machine. If your laptop gets stolen, your keys aren't compromised.

Configuration Setup

1. **Choose a provider:** DigitalOcean, Hetzner, Vultr, or Linode are all solid. For OpenClaw, a $10–$20/mo VPS (2 vCPU, 4GB RAM, 80GB SSD) is more than enough for most use cases. Hetzner gives the best value in EU/US regions.
2. **Provision the server:** Select Ubuntu 22.04 or 24.04 LTS. Add your SSH key during setup (never use password auth). Choose a region close to your primary API provider's servers for lower latency.
3. **Secure the server:** SSH in as root, then: create a non-root user (adduser openclaw), grant sudo (usermod -aG sudo openclaw), disable root SSH login in /etc/ssh/sshd_config (PermitRootLogin no), change SSH port from 22, enable UFW firewall (ufw allow [ssh-port] && ufw allow 80 && ufw allow 443 && ufw enable), and install fail2ban.
4. **Install Node.js:** Use NodeSource: curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash - && sudo apt install -y nodejs. Verify with node -v.
5. **Clone, install, configure:** Same as local: git clone, npm install, .env configuration. Set your model routing (85/15 Haiku/Sonnet split for cost control).
6. **Process management with PM2:** npm install -g pm2 && pm2 start index.js --name openclaw && pm2 startup && pm2 save. This ensures OpenClaw auto-restarts on crashes and server reboots. Use pm2 logs openclaw to monitor.
7. **Reverse proxy (optional but recommended):** Install Nginx: sudo apt install nginx. Configure a server block pointing to localhost:[your-port]. Add SSL with Certbot: sudo certbot --nginx -d yourdomain.com. Now your OpenClaw dashboard is accessible via HTTPS.

Common Issues & Fixes

- **SSH connection drops during setup:** Long-running commands over SSH can timeout. Use tmux or screen before starting installs: tmux new -s openclaw. If disconnected, reattach with tmux attach -t openclaw.
- **Memory issues on cheap VPS:** 1GB RAM VPS will choke during npm install. Add swap space: sudo fallocate -l 2G /swapfile && sudo chmod 600 /swapfile && sudo mkswap /swapfile && sudo swapon /swapfile. Make it permanent in /etc/fstab.
- **Unattended upgrades breaking Node:** Ubuntu's auto-updates can occasionally restart services or update system libraries that break Node. Configure /etc/apt/apt.conf.d/50unattended-upgrades to exclude nodejs packages.
- **Disk space filling up:** Log files grow fast. Set up log rotation: pm2 install pm2-logrotate && pm2 set pm2-logrotate:max_size 10M && pm2 set pm2-logrotate:retain 7.
- **API keys exposed in .env:** Never commit .env to git. Add it to .gitignore. For extra security, use environment variables directly or a secrets manager. Restrict file permissions: chmod 600 .env.
- **Port already in use after crash:** If OpenClaw crashes without cleanup, the port stays occupied. Fix: lsof -i :[port] | grep LISTEN, then kill the PID. PM2 usually handles this automatically.
- **DNS/SSL certificate issues:** Certbot certificates expire every 90 days. Certbot sets up auto-renewal by default, but verify: sudo certbot renew --dry-run. Set a calendar reminder to check.
- **Server clock drift:** VPS clocks can drift, causing API auth failures. Install NTP: sudo apt install ntp. Verify with timedatectl – ensure NTP is synchronized.

Performance Notes

VPS performance is consistent and predictable, which is the whole point. Unlike your laptop, a VPS doesn't throttle when it gets warm, doesn't sleep when you close a lid, and doesn't share resources with your browser tabs and Spotify. For OpenClaw specifically, the bottleneck is always API latency, not local compute. A $10/mo VPS handles single-instance workloads easily. For multi-instance or heavy concurrent usage, step up to 4GB+ RAM. Monitor with pm2 monit and set up alerts for memory usage above 80%.

Decision Framework: Which Platform When?

If You're…	Use…
Just starting out, learning OpenClaw	Mac or PC – whichever you own. Zero cost to start.

Building automations for your own business	Start local (Mac/PC), move to VPS once stable.
Running client deployments	VPS only. Non-negotiable for uptime SLAs.
Creating content / screen recording demos	Mac or PC locally. You need visual access to the UI.
Running 3+ instances simultaneously	VPS. Better resource isolation and management.
On a tight budget	Your existing Mac/PC. VPS can wait.
Scaling to production for a large client	VPS with PM2, Nginx, SSL, monitoring, and backups.

Cost Optimization by Platform

Local (Mac/PC)

Your only costs are electricity and API usage. The 85/15 Haiku/Sonnet routing strategy applies regardless of platform and is where the real savings come from – dropping from $1,500–$3,000/mo to $100–$200/mo. Local machines add roughly $5–15/mo in electricity for heavy usage. The machine is a sunk cost you've already paid.

VPS

VPS costs are predictable and scale linearly. For a single OpenClaw instance, a $10/mo Hetzner CX21 (2 vCPU, 4GB) is plenty. If you're deploying for clients at $3–5K/mo retainers, the $10–50 VPS cost is negligible. For the 1,200-location restaurant deal, you'd want dedicated infrastructure – a $50–100/mo VPS with 8GB+ RAM, automated backups, and monitoring. Still a rounding error on a $120K/mo contract.

The Real Cost Lever

Platform costs are noise. API costs are signal. Focus your optimization energy on model routing, token reduction, caching strategies, and prompt engineering. The platform you run OpenClaw on is a $0–50/mo decision. Your API routing strategy is a $100–$3,000/mo decision.

Security Hardening Checklist

Security Measure	Mac	PC	VPS

API keys in .env (not hardcoded)	✓ Required	✓ Required	✓ Required
.env in .gitignore	✓ Required	✓ Required	✓ Required
File permissions (chmod 600 .env)	Recommended	N/A (use WSL)	✓ Required
Non-root user for running OpenClaw	Default	Default	✓ Required
SSH key auth (disable passwords)	N/A	N/A	✓ Required
Firewall (UFW / Windows Firewall)	Recommended	Enabled by default	✓ Required
fail2ban for brute force protection	N/A	N/A	✓ Required
SSL/HTTPS for dashboard access	N/A (localhost)	N/A (localhost)	✓ Required
Regular OS updates	Recommended	Recommended	✓ Required
Automated backups	Time Machine	Optional	✓ Required

Migration Path: Local to VPS

Most people should start local and migrate to VPS when their workflow demands it. Here's the clean migration path:

- **Keep your config portable:** Use .env files consistently. Your .env on Mac should be nearly identical to your .env on VPS. Same keys, same model routing, same settings. Only the server-specific values change (ports, domains, SSL paths).
- **Use Git for deployment:** Push your local config to a private repo. On the VPS, git pull to deploy. Never SCP files around manually – you'll lose track of versions.
- **Test locally, deploy to VPS:** Make changes on your Mac/PC, test them, push to git, pull on VPS. Treat your local machine as staging and the VPS as production.
- **Maintain local as backup:** Even after moving to VPS, keep your local setup working. If your VPS has issues, you can run locally while you fix things. Redundancy matters.

Final Recommendations

For THE SPRINT members just getting started: use whatever machine you have right now. Mac, PC, doesn't matter. Get OpenClaw running locally, learn the configuration, optimize your model routing, and start seeing results. The platform is the least important variable in your success.

Once you're generating revenue or running client work, migrate to a VPS. The $10–50/mo cost pays for itself instantly in reliability and professionalism. Your clients don't care what computer you own – they care that their automation runs 24/7 without you babysitting it.

The single most impactful thing you can do on any platform is get your model routing right. The 85/15 Haiku/Sonnet split with intelligent task classification is where the 97% cost reduction comes from. That works identically on Mac, PC, and VPS. Focus there first, platform second.