Deploying Self-Improving AI Agents on Apple Silicon with OpenClaw and NVIDIA Nemotron

A Practitioner's Guide to Automated Agent Replication

A Production Deployment Case Study

Adnan Tanveer (Addy)
Speedrun AI Labs
Sydney, Australia
speedrunlab.ai

April 13, 2026 | Paper Version 2.0, submitted to SSRN

NOTE ON TIME-SENSITIVITY: This paper documents the state of OpenClaw (version 2026.3.11), NemoClaw (alpha), and NVIDIA Nemotron 3 Nano 30B as of April 13, 2026. All are in active development. The methodology -- replicating AI agents from specification documents using an AI deployment assistant -- remains applicable regardless of version.

Abstract

Deploying autonomous AI agents in production requires extensive configuration of infrastructure, security policies, memory systems, scheduled tasks, and operational tooling. When an organisation operates multiple agents across machines, the deployment complexity multiplies. This paper documents the replication of a fully operational AI agent from one Apple Silicon machine to another using PDF specification documents executed by an AI deployment assistant (Claude Code).

We make three contributions: (1) we present a nine-phase replication methodology that deploys a complete agent stack -- including nine custom skills, four scheduled cron jobs, a hardened self-improvement hook system, and local inference via NVIDIA Nemotron 3 Nano 30B on Ollama -- from specification documents alone; (2) we document a self-improvement architecture where agents log errors and corrections, promote recurring patterns to permanent identity files, and inject live learning summaries at every session bootstrap; and (3) we validate the deployment with a three-pass, 69-check verification audit achieving zero errors. The target agent operates in a regulated investment fintech environment, running marketing workloads with NVIDIA NemoClaw providing all six security layers validated in our companion paper [2]. This is a practitioner experience report from a production deployment.

Keywords: *AI agents, agent replication, self-improving agents, OpenClaw, NemoClaw, Nemotron, Apple Silicon, M4 Pro, M4 Max, Ollama, autonomous deployment, multi-agent systems, agent memory*

1. Introduction

The multi-agent deployment gap is a practical problem facing organisations adopting autonomous AI agents. A single agent, carefully configured over weeks of production use, accumulates institutional knowledge in its configuration: custom skills, operational workflows, error handling patterns, scheduled tasks, and identity files that define its behaviour. Replicating this agent to a second machine -- for redundancy, load distribution, or specialised workloads -- is not a simple copy operation.

Agent configurations are distributed across multiple systems: the OpenClaw platform configuration, workspace files that define agent identity and memory, custom skill directories, hook scripts, cron job definitions, and local inference model installations. There is no standard 'export agent' mechanism in the OpenClaw ecosystem. Each deployment is effectively handcrafted.

This paper documents the replication of an agent named Anton (running on a Mac Mini M4 Pro with 64 GB unified memory) to a second agent named Steve (running on a Mac Studio M4 Max with 64 GB unified memory) using a novel methodology: the source agent's configuration was captured in PDF specification documents, which were then read and executed by an AI deployment assistant (Claude Code) on the target machine. The approach achieved a complete, verified deployment in a single session.

2. Source Agent Architecture

The source agent (Anton) runs on a Mac Mini M4 Pro with 64 GB unified memory and 512 GB SSD, deployed on OpenClaw with a three-layer fault-tolerant memory architecture documented in our previous work [1]. That architecture combines pre-compaction memory flushing (Layer 1), QMD local hybrid search (Layer 2), and optional Mem0 self-hosted on Qdrant (Layer 3, currently disabled due to a resolved upstream issue).

Anton's configuration was captured in four PDF specification documents totalling approximately 30 pages: (1) Agent Skills Reference; (2) Memory Architecture and Telegram Configuration; (3) Steve Setup Guide -- self-improvement system, hook code, file headers, and verification steps; and (4) New Agent Deployment Checklist -- step-by-step deployment, cron jobs, and security boundaries.

A master setup plan (STEVE_SETUP_PLAN.md) referenced all four PDFs and defined nine deployment phases with specific commands, configuration blocks, and verification checkpoints.

3. Target Hardware

Component	Source (Anton)	Target (Steve)
Model	Mac Mini M4 Pro	Mac Studio M4 Max (Mac16,9)
CPU	14-core (10P + 4E)	16-core (12P + 4E)
GPU	20-core	40-core
Unified Memory	64 GB (273 GB/s)	64 GB (546 GB/s)
Storage	512 GB SSD	1 TB SSD
Retail Price	A$3,099 (April 2026)	A$4,849 (April 2026)
Local Inference	Cloud-only (OpenAI-compatible endpoint)	Nemotron 3 Nano 30B via Ollama

Table 1: Hardware comparison. Both machines use 64 GB unified memory. Pricing verified on apple.com/au, April 2026. Apple Silicon memory pricing has increased significantly in early 2026 due to global DRAM supply constraints.

The Mac Studio M4 Max was selected for its 546 GB/s memory bandwidth and 40-core GPU, enabling faster local inference. NVIDIA Nemotron 3 Nano 30B (Q4_K_M, 24 GB on disk) runs entirely in unified memory via Ollama, eliminating cloud dependency for local inference tasks.

4. Replication Methodology

The replication was performed by Claude Code (Anthropic's CLI agent) reading the PDF specifications and master setup plan, then executing nine phases autonomously on the target machine.

Phase	Description	Key Actions
1. System Prep	OS and tool config	Git identity, firewall, power, Homebrew, Node.js, Docker, Bun
2. NemoClaw	Platform install	Clone NemoClaw, install Ollama, pull Nemotron 3 Nano 30B
3. OpenClaw Config	Platform config	11 config values: gateway.bind=loopback, memory.backend=qmd
4. Workspace	Identity and memory	9 workspace files, .learnings/ directory, archive structure
5. Skills	Capabilities	4 from GitHub, 4 from PDF specs, 1 from Dropbox. 9 total
6. Hooks	Self-improvement	Hardened handler.js: delete guard, dedup guard, virtual injection
7. QMD	Local search	QMD v2.1.0 from source, Bun build, wrapper script
8. Crons	Scheduled ops	4 cron jobs: morning report, learning review, cleanup, model refresh
9. Verification	Validation	3-pass audit, 69 checks per pass, zero-error target achieved

Table 2: Nine-phase replication methodology.

5. Self-Improvement Architecture

The self-improvement system enables agents to learn from their mistakes and progressively improve their behaviour. Three components work together: log files that capture errors and corrections, a bootstrap hook that injects live learning summaries into every session, and promotion rules that elevate proven patterns into permanent identity files.

5.1 The Log Files (.learnings/)

Three structured markdown files in ~/.openclaw/workspace/.learnings/ capture operational experience: LEARNINGS.md (corrections, knowledge gaps, best practices), ERRORS.md (command failures, API failures, tool failures), and FEATURE_REQUESTS.md (missing capabilities). Each entry has a unique ID, category, Pattern-Key for deduplication, priority level, and status (pending or promoted).

5.2 The Bootstrap Hook (handler.js)

At every session bootstrap, the self-improvement hook reads the .learnings/ files, counts pending items, flags critical issues, identifies recurring patterns, and injects a virtual file (SELF_IMPROVEMENT_REMINDER.md) into the session context.

The hook was hardened after a production incident where a physical copy of the virtual file appeared in the workspace root, causing double injection per session. The hardened version includes three safeguards: (1) self-healing delete guard; (2) session-scoped deduplication flag; (3) SKILL.md constraint telling the agent never to write the virtual file physically.

5.3 Promotion Rules

When a Pattern-Key appears three or more times, the learning is promoted from .learnings/ into permanent identity files: behavioural patterns to SOUL.md, workflow improvements to AGENTS.md, tool gotchas to TOOLS.md. A weekly cron job reviews and reports via Telegram.

6. Local Inference Stack

The target machine runs NVIDIA Nemotron 3 Nano 30B locally via Ollama (Q4_K_M, 24 GB on disk, 73 MB idle). QMD v2.1.0 provides local hybrid search with zero API dependency. For primary inference, the agent uses an OpenAI-compatible endpoint with tiered model routing.

7. Operational Results

Component	Status	Detail
OpenClaw Gateway	Running	loopback:18789, LaunchAgent service, 579 MB
Telegram Channel	Active	streaming=off, 393+ messages, zero duplicates
Cron: morning-report	Active	0 8 * * * AEST, delivery mode=announce
Cron: weekly-learning-review	Active	0 9 * * 1 AEST, delivery mode=announce
Cron: workspace-cleanup	Active	0 18 * * * AEST, dry-run mode
Cron: update-models	Active	0 12 1 * * AEST, monthly model refresh
Skills (9 total)	Installed	fat-agent, marketingskills, self-improving-agent, deep-audit, living-spec, multi-model, model-switching, and 2 others
Self-Improvement Hook	Installed	Hardened handler.js with delete guard + dedup guard
QMD	Installed	v2.1.0, wrapper at /opt/homebrew/bin/qmd
Ollama + Nemotron	Running	v0.20.5, nemotron-3-nano:30b (24 GB), 73 MB idle
NemoClaw Sandbox	Ready	6/6 security layers passing (validated in [2])

Table 3: Component operational status after deployment.

Component	Memory	Notes
OpenClaw Gateway	579 MB	Node.js process, LaunchAgent managed
Ollama Service	73 MB	Idle; model loaded on demand
Docker Desktop	~1,052 MB	7 processes total
OpenShell Cluster	732 MB	k3s + sandbox (when active)
Nemotron 3 Nano 30B	24 GB on disk	Loaded into unified memory during inference
Total (without inference)	~2,436 MB	Leaves ~61.5 GB for inference and workloads

Table 4: Memory footprint of the deployed stack.

7.1 Verification Methodology

The deployment was validated with a three-pass verification audit. Each pass executed 69 automated checks covering all nine deployment phases. All three passes returned 68 of 69 passing with a single cosmetic deviation (display sleep setting). After manual correction, zero-error target achieved.

8. Enterprise Cost Analysis

Item	Cost (AUD)
Mac Mini M4 Pro 64 GB (source agent, Anton)	A$3,099

Item	Cost (AUD)
Mac Studio M4 Max 64 GB (target agent, Steve)	A$4,849
Total hardware (two-agent fleet)	A$7,948
Monthly electricity (both machines, 24/7)	~A$25
Monthly API cost (OpenAI-compatible endpoint)	~A$300 (two agents)
Monthly operational total	~A$325
Annual operational cost	~A$3,900
First-year total cost of ownership	~A$11,848

Table 5: Two-agent fleet cost analysis. Prices verified on apple.com/au, April 2026.

A two-agent fleet costs under A$8,000 in hardware and under A$325 per month. For comparison, a single NVIDIA DGX Station lists at US$80,000 to US$125,000 (A$125,000 to A$195,000) [2]. The target agent operates in a regulated investment fintech environment with NVIDIA NemoClaw providing all six security layers validated with fixes documented in [2].

9. Limitations

This paper documents a two-agent deployment. Scaling to larger fleets has not been tested. The replication methodology depends on comprehensive PDF specifications -- quality is bounded by documentation quality. The self-improvement system has been operational for less than one week. NemoClaw sandbox validation is documented separately in [2].

10. Conclusion

We have demonstrated that a fully operational AI agent can be replicated from one machine to another using PDF specification documents executed by an AI deployment assistant. The nine-phase methodology deployed a complete agent stack in a single session, verified by a three-pass, 69-check audit with zero errors.

The self-improvement architecture provides a mechanism for continuous behavioural improvement that compounds over time. The total cost of ownership for a two-agent fleet on Apple Silicon is under A$12,000 in the first year, with NVIDIA NemoClaw available as an enterprise trust layer with all six security layers passing.

Speedrun AI Labs | Sydney, Australia | speedrunlab.ai

References

[1] A. Tanveer (Addy), "Building Fault-Tolerant Memory for OpenClaw AI Agents: A Three-Layer Architecture for the SQLITE_CANTOPEN Regression in 2026.3.x," Speedrun AI Labs, March 2026. SSRN.

[2] A. Tanveer (Addy), "NemoClaw on Apple Silicon: Validating and Fixing NVIDIA's Enterprise Agent Security Stack on Consumer Hardware," Speedrun AI Labs, April 2026. SSRN.

[3] A. Tanveer (Addy), "Inside NemoClaw: An Architectural Analysis of NVIDIA's Enterprise Security Stack for Autonomous AI Agents," Speedrun AI Labs, March 2026. SSRN.

[4] NVIDIA Corporation. NemoClaw. github.com/NVIDIA/NemoClaw, alpha.

[5] NVIDIA Corporation. OpenShell. github.com/NVIDIA/OpenShell, v0.0.26.

[6] Ollama. ollama.com, v0.20.5.

[7] NVIDIA Corporation. Nemotron 3 Nano 30B Model Card. build.nvidia.com, 2026.

[8] T. Lutke, QMD (Query Markup Documents). github.com/tobi/qmd, v2.1.0.

[9] P. Steinberger, OpenClaw. openclaw.ai, 2026.

[10] P. Skoett, Self-Improving Agent. github.com/peterskoett/self-improving-agent.

Appendix A: Version Matrix

Component	Version	Source
macOS	Tahoe 26.4.1 (Build 25E253)	Apple
OpenClaw	2026.3.11 (29dc654)	npm: openclaw
NemoClaw	alpha	github.com/NVIDIA/NemoClaw
OpenShell	0.0.26	github.com/NVIDIA/OpenShell
Docker Desktop	4.68.0 (Engine 29.3.1)	docker.com
Ollama	0.20.5	ollama.com
Nemotron 3 Nano	30B Q4_K_M (24 GB)	NVIDIA
QMD	2.1.0 (cfd640e)	github.com/tobi/qmd
Node.js	25.9.0	nodejs.org
Bun	1.3.12	bun.sh

Table A1: Complete version matrix for the target deployment.

Appendix B: Verification Audit Summary

Phase	Checks	Pass 1	Pass 2	Pass 3
1. System Prep	17	16/17	16/17	16/17*
2. NemoClaw	6	6/6	6/6	6/6
3. OpenClaw Config	12	12/12	12/12	12/12
4. Workspace	16	16/16	16/16	16/16

Phase	Checks	Pass 1	Pass 2	Pass 3
5. Skills	14	14/14	14/14	14/14
6. Hooks	4	4/4	4/4	4/4
7. QMD	4	4/4	4/4	4/4
8. Crons	4	4/4	4/4	4/4
9. Verification	2	2/2	2/2	2/2
TOTAL	**69**	**68/69**	**68/69**	**68/69***

*Table B1: Three-pass verification audit. *Single deviation: displaysleep=10 (expected 5), corrected after audit.*