A Deep Dive on Nemoclaw

March 28th, 2026

Background

Major AI service providers have structured their internal architecture to be able to provide persistent memory across chats, route API requests through authentication & privacy layers, enable agents to access tools, and orchestrate how prompts are routed to subagents to provide the best possible response.

This entire package is what differentiates a service like Claude, ChatGPT, and contrasts it against running models locally through vLLM or Ollama. These large-scale AI services are able to process images, generate excel documents, and provide high-level reasoning.

The competitive advantage between AI service providers is the ability to do this effectively, at scale, and at a low cost.

OpenClaw provided the open-source foundation for firms to be able to deploy their own version of this system. Nemoclaw completed the package, with security enhancements, to make the project enterprise ready.

Overview

Nemoclaw operates at a 4-level stack:

Plugins	• Integrates modular components to the system • Examples include embeddings, retrieval, LLM inference, memory, tools, and data processing • Implemented as typescript packages
Blueprints	• A recipe to create an entire agent environment • Configured when the user runs the *nemoclaw onboard* command • Nemoclaw contains example security policies for accessing common services like pypi, telegram, and npm • Users can configure custom policies that can be used by the blueprint engine to build nemoclaw sandboxes • Custom policies control filesystem access and network egress • Nothing in user-space can violate these policies, filtering is done at the syscall level

Sandboxes	• A sandbox is a Kubernetes pod running a rust binary called openshell-sandbox, which is called a “supervisor” • This binary is responsible for building the container according to blueprint specifications, including filesystem, virtual networks, and http servers
Inference	• The agent never talks with the model provider directly • Agent communicates over a virtual network interface • Openshell proxy intercepts it, injects credentials, and routes it

Configuration Files & Commands

Plugins	
Sandbox name & inference provider	*user@host: ~/.nemoclaw/config.yaml*
API keys	*user@host: ~/.nemoclaw/credentials.json*
Blueprints	
Blueprint configuration	*user@host: $(npm root -g)/nemoclaw/nemoclaw-blueprint/blueprint.yaml*
Blueprint engine	*user@host: $(npm root -g)/nemoclaw/src/blueprint/runner.ts*
Main policy configuration	*user@host: $(npm root -g)/nemoclaw/nemoclaw-blueprint/policies/openclaw-sandbox.yaml*
Example policies	*user@host: $(npm root -g)/nemoclaw/nemoclaw-blueprint/policies/presets/*
Sandboxes	
View sandboxes	*user@host: openshell sandbox list*
View active policy	*user@host: openshell policy get <sandbox> --full*
View logs	*user@host: openshell logs <sandbox> --tail --source sandbox*
Open terminal UI	*user@host: openshell term*
Inference	
Inference model status	*user@host: openshell inference status*
Registered credentials	*user@host: openshell provider list*
Switch model command	*user@host: openshell inference set --provider <p> --model <m>*

WSL Setup

1) Set up the environment
    a. Enable WSL2 in Windows Features and reboot
    b. Install Ubuntu 22.04 LTS or later
    c. Install Docker Desktop on host
    d. Install npm, nodejs, and add to $PATH
    e. Generate an NVIDIA API key and add it as an environment variable
2) Install NemoClaw
    a. Open Ubuntu WSL2 in Terminal
    b. curl -fsSL https://nvidia.com/nemoclaw.sh | bash
    c. Run command: nemoclaw onboard
    d. Run command: nemoclaw sandbox list
    e. Run command: openshell policy get <sandbox-name> --full
    f. Run command: nemoclaw <sandbox-name> connect
    g. Run command: exit → You will return to host
3) Inspect Setup
    a. Run command: openshell logs <sandbox-name> --tail --source sandbox
    b. Run command: openshell term
    c. Inspect files from the configuration table above

Gaps

Due to how new the project is, there are no guides that outline complete NemoClaw deployment in a corporation and through a platform provider such as Azure or AWS.

References

https://docs.nvidia.com/nemoclaw/latest/

https://docs.nvidia.com/openshell/latest/

https://nemoclawai.io/blog/

https://deepwiki.com/NVIDIA/OpenShell

https://deepwiki.com/NVIDIA/OpenShell/3-core-concepts

https://deepwiki.com/NVIDIA/OpenShell/4-cli-reference