Setting Up Mem0 Plugin with NemoClaw

This guide walks you through installing NemoClaw and configuring the `@mem0/openclaw-mem0` plugin for persistent long-term memory in your OpenClaw agent.

Prerequisites

Hardware

Resource	Recommended	Minimum
CPU	4+ vCPU	2 vCPU
RAM	16 GB	8 GB
Disk	40 GB free	20 GB free

Software

- **OS**: Ubuntu 22.04 LTS or later (native), or macOS/Windows via Docker
- **Docker**: Installed and running
- **Node.js**: v20 or later (the NemoClaw installer will install this for you if missing)
- **npm**: v10 or later (installed with Node.js)

Accounts

- **NVIDIA account** — sign up at build.nvidia.com, then generate an API key at build.nvidia.com/settings/api-keys (starts with `nvapi-`)
- **Mem0 account** — sign up at app.mem0.ai, then generate an API key from the dashboard (starts with `m0-`)

Part 1: Install NemoClaw

Choose **Option A** (Ubuntu server) or **Option B** (Docker container).

Option A: Ubuntu Server (Native)

Step A1: Install Docker

```
sudo apt-get update
sudo apt-get install -y docker.io
sudo systemctl enable --now docker
sudo usermod -aG docker $USER
```

Log out and back in for the group change to take effect, then verify:

```
docker ps
```

> **If you see `permission denied while trying to connect to the Docker daemon socket`:**
> Your current shell session doesn't have the `docker` group yet. Run `newgrp docker` to reload group membership without logging out, then retry `docker ps`.

Step A2: Fix cgroup v2 (Ubuntu 24.04 only)

Ubuntu 24.04 defaults to cgroup v2, which causes k3s (used by NemoClaw) to fail.

```
sudo python3 -c "
import json, os
p = '/etc/docker/daemon.json'
c = json.load(open(p)) if os.path.exists(p) else {}
c['default-cgroupns-mode'] = 'host'
json.dump(c, open(p, 'w'), indent=2)
"
sudo systemctl restart docker
```

> Skip this step if you are on Ubuntu 22.04.

> **If you skip this step on Ubuntu 24.04**, onboarding will fail at step 2/7 with:
>
> ```
> Error: K8s namespace not ready
> timed out waiting for namespace 'openshell' to exist
> ```
>
> Come back and run the commands above, then re-run `nemoclaw onboard`.

Step A3: Install NemoClaw

```
curl -fsSL https://nvidia.com/nemoclaw.sh | bash
```

This installs Node.js (via nvm), npm, NemoClaw, and the OpenShell CLI.

Reload your shell so that `nemoclaw` is on your PATH:

```
source ~/.bashrc
```

Verify:

```
nemoclaw --help
```

Step A4: Run Onboarding

```
nemoclaw onboard
```

The wizard walks through 7 interactive steps:

Step	What it does	What you enter
1/7	Preflight checks	(automatic)
2/7	Start OpenShell gateway	(automatic — deploys k3s in Docker, takes 1-2 min)
3/7	Create sandbox	Enter a name, e.g. `my-assistant`
4/7	Configure inference	Paste your NVIDIA API key (`nvapi-...`)
5/7	Set up inference provider	(automatic)
6/7	Set up OpenClaw inside sandbox	(automatic)
7/7	Policy presets	Type `Y` to apply suggested presets (pypi, npm)

> **If step 7 fails with `sandbox not found` :**
> The gateway was restarted during onboarding and lost the sandbox state. Re-run `nemoclaw onboard` . When prompted that the sandbox already exists, choose `y` to recreate it.

Step A5: Verify

```
nemoclaw my-assistant status
```

You should see the sandbox status as `Ready` . Connect to it:

```
nemoclaw my-assistant connect
```

You should see `sandbox@my-assistant:~$`. Type `exit` to disconnect.

Now skip to Part 2: Install the Mem0 Plugin.

Option B: Docker Container (macOS / Windows / any host)

Step B1: Install Docker Desktop

- **macOS**: Download from docker.com/products/docker-desktop and install
- **Windows**: Download Docker Desktop and enable WSL 2 backend

Verify Docker is running:

```
docker ps
```

Step B2: Create an Ubuntu Container

```
docker run -d \
  --name nemoclaw-dev \
  --privileged \
  --network host \
  -v /var/run/docker.sock:/var/run/docker.sock \
  ubuntu:24.04 sleep infinity
```

> **`--network host` is required.** Without it, the container cannot reach the OpenShell gateway at `127.0.0.1:8080` and onboarding will fail at step 2/7 with `Gateway failed to start`.

Step B3: Install Dependencies

```
docker exec nemoclaw-dev bash -c "apt-get update && apt-get install -y cu
```

Step B4: Install NemoClaw

```
docker exec -it nemoclaw-dev bash -c "curl -fsSL https://nvidia.com/nemoc
```

Step B5: Open a Container Shell

All remaining commands run inside this shell:

```
docker exec -it nemoclaw-dev bash
```

Load nvm so that `nemoclaw` is available:

```
export NVM_DIR="$HOME/.nvm" && source "$NVM_DIR/nvm.sh"
```

Verify:

```
nemoclaw --help
```

> **You must run `export NVM_DIR=...` every time you open a new shell in the container.** To avoid this, the NemoClaw installer already added it to `~/.bashrc`, but `docker exec` doesn't always source it. You can also run `source ~/.bashrc`.

Step B6: Run Onboarding

```
nemoclaw onboard
```

Follow the same 7-step wizard:

1. Preflight checks (automatic)
2. Start gateway (automatic, takes 1-2 minutes)
3. Sandbox name — enter e.g. `my-assistant`
4. NVIDIA API key — paste your `nvapi-...` key
5. Inference provider (automatic)
6. OpenClaw setup (automatic)
7. Policy presets — type `Y` to apply pypi and npm

> **If step 7 fails with `sandbox not found`:**
> Re-run `nemoclaw onboard`. When prompted that the sandbox already exists, choose `y` to recreate it.

Step B7: Verify

```
nemoclaw my-assistant status
nemoclaw my-assistant connect
```

You should see `sandbox@my-assistant:~$` . Type `exit` to disconnect.

> **For all remaining commands:** Commands labeled "outside the sandbox" should be run in the container shell (`docker exec -it nemoclaw-dev bash`), not on your host machine. Commands labeled "inside the sandbox" should be run after `nemoclaw my-assistant connect` .

Part 2: Install the Mem0 Plugin

Method 1: Install via OpenClaw CLI

Connect to the sandbox:

```
nemoclaw my-assistant connect
```

Inside the sandbox, run:

```
openclaw plugins install @mem0/openclaw-mem0
```

If this succeeds, type `exit` and skip to Part 3: Update the Network Policy.

> **If you see `npm error 403 Forbidden` :**
>
> ```
> npm error 403 403 Forbidden - GET
> https://registry.npmjs.org/@mem0%2fopenclaw-mem0
> ```
>
> This is caused by the OpenShell gateway's TLS proxy blocking scoped npm packages (the `%2f` in the URL). Type `exit` to leave the sandbox and use Method 2 below instead.

Method 2: Manual Installation

Run all commands in this section **outside the sandbox** (on the Ubuntu server, or in the Docker container shell).

Step 1: Download the plugin

```
cd /tmp
npm pack @mem0/openclaw-mem0
```

This creates `mem0-openclaw-mem0-X.Y.Z.tgz`.

Step 2: Install dependencies

```
mkdir -p /tmp/openclaw-mem0-full
cd /tmp/openclaw-mem0-full
tar xzf /tmp/mem0-openclaw-mem0-*.tgz --strip-components=1
npm install --omit=dev
```

Step 3: Bundle into a tarball

```
cd /tmp
tar czf openclaw-mem0-full.tgz -C openclaw-mem0-full .
```

Step 4: Upload into the sandbox

```
openshell sandbox upload my-assistant \
  /tmp/openclaw-mem0-full.tgz \
  /sandbox/openclaw-mem0-full.tgz
```

Expected output: `Upload complete`.

Step 5: Extract inside the sandbox

Connect to the sandbox:

```
nemoclaw my-assistant connect
```

Inside the sandbox:

```
mkdir -p ~/.openclaw/extensions/openclaw-mem0
tar xzf /sandbox/openclaw-mem0-full.tgz/openclaw-mem0-full.tgz \
  -C ~/.openclaw/extensions/openclaw-mem0
```

Verify the files:

```
ls ~/.openclaw/extensions/openclaw-mem0/
```

Expected output:

```
README.md  dist  node_modules  openclaw.plugin.json  package-lock.json
package.json
```

Stay connected to the sandbox for Part 3.

Part 3: Update the Network Policy

The Mem0 plugin needs to reach `api.mem0.ai`, which is blocked by default. You must add it to the sandbox network policy.

Run all commands in this section **outside the sandbox**. If you are connected to the sandbox, type `exit` first.

Step 1: Locate the baseline policy

```
find / -path "*/nemoclaw-blueprint/policies/openclaw-sandbox.yaml" 2>/dev
```

Typical location:

```
~/.nvm/versions/node/v22.22.1/lib/node_modules/nemoclaw/nemoclaw-
blueprint/policies/openclaw-sandbox.yaml
```

Step 2: Create a custom policy file

```
cp <path-from-step-1>/openclaw-sandbox.yaml /tmp/custom-policy.yaml
```

Open `/tmp/custom-policy.yaml` in an editor and add the following block under `network_policies:`, before the `telegram:` section:

```
    mem0_api:
      name: mem0_api
      endpoints:
        - host: api.mem0.ai
          port: 443
          access: full
      binaries:
```

```
      - { path: /usr/local/bin/node }
      - { path: /usr/local/bin/openclaw }
```

> **The `binaries` list is required.** Without it, the proxy denies all requests even though the endpoint is allowed. You will see this error in the gateway logs:
>
> `binary '/usr/local/bin/node' not allowed in policy 'mem0_api'`

Step 3: Apply the policy

```
openshell policy set my-assistant --policy /tmp/custom-policy.yaml --wait
```

Expected output:

✓ `Policy version N submitted`
✓ `Policy version N loaded`

> **If you see `sandbox not found`:**
> The gateway may have restarted. Run `openshell sandbox list` to verify the sandbox exists. If not, re-run `nemoclaw onboard`.

Part 4: Configure the Plugin

Connect to the sandbox:

```
nemoclaw my-assistant connect
```

Run the following commands **inside the sandbox** (`sandbox@my-assistant:~$`):

Step 1: Set the memory slot to Mem0

```
openclaw config set plugins.slots.memory openclaw-mem0
```

> **If you skip this step**, the plugin will show as `disabled` in `openclaw plugins list` with the message:
>
> `plugin disabled (memory slot set to "memory-core") but config is present`

> Run the command above to fix it.

Step 2: Enable the plugin

```
openclaw config set plugins.entries.openclaw-mem0.enabled true
```

Step 3: Set your Mem0 API key

```
openclaw config set plugins.entries.openclaw-mem0.config.apiKey "m0-YOUR_
```

Replace `m0-YOUR_MEM0_API_KEY` with your actual key from [app.mem0.ai](app.mem0.ai).

Step 4: Set a user ID

```
openclaw config set plugins.entries.openclaw-mem0.config.userId "your-use
```

Replace `your-user-id` with any unique identifier (e.g. `alice`, `user_123`). All memories are scoped to this ID.

Step 5: Restart the OpenClaw gateway

```
nemoclaw-start
```

> **If you see `systemctl not available` or `systemd user services are unavailable`:**
> This is expected inside the sandbox container. `nemoclaw-start` is the correct command — it runs the gateway in foreground mode without systemd.

Step 6: Verify the plugin is loaded

Look for this line in the startup output:

```
openclaw-mem0: registered (mode: platform, user: your-user-id, graph: false, autoRecall: true, autoCapture: true)
```

You can also run:

```
openclaw plugins list
```

The `Memory (Mem0)` plugin should show status **loaded**.

> **If the plugin shows `disabled`:** Go back to Step 1 and set the memory slot.
>
> **If you see `recall failed: Connection error` or `capture failed: Connection error`:**
> The network policy is not applied or missing the `mem0_api` entry. Go back to Part 3.

Part 5: Test the Integration

All test commands run **inside the sandbox**.

If you exited, reconnect:

```
nemoclaw my-assistant connect
```

If the gateway isn't running, start it:

```
nemoclaw-start &
```

Test 1: Auto-capture (storing memories)

```
openclaw agent --agent main --local \
  -m "My name is Alice and I work on distributed systems" \
  --session-id test1
```

Look for this in the output:

```
openclaw-mem0: auto-captured 1 memories
```

This confirms the plugin stored facts from the conversation to Mem0.

> **If you see `capture failed: Connection error`:**
> The network policy is not allowing `api.mem0.ai`. Go to Part 3 and apply the policy.
>
> **If you see `Telemetry event capture failed: TypeError: fetch failed`:**
> This is harmless. The mem0ai SDK's telemetry endpoint (`us.i.posthog.com`) is blocked by

> the sandbox, but it does not affect memory functionality. You can safely ignore these errors.

Test 2: Auto-recall (retrieving memories across sessions)

Start a **new session** (different `--session-id`):

```
openclaw agent --agent main --local \
  -m "What do you know about me?" \
  --session-id test2
```

Look for:

```
openclaw-mem0: injecting 1 memories into context (1 long-term, 0 session)
```

The agent should respond with the information from the previous session ("Alice", "distributed systems").

Test 3: Interactive TUI

```
openclaw tui
```

Send messages and the plugin will automatically capture and recall memories in the background.

Plugin Configuration Reference

All options are set inside the sandbox via:

```
openclaw config set plugins.entries.openclaw-mem0.config.<key> <value>
```

Key	Type	Default	Description
`mode`	`"platform"` | `"open-source"`	`"platform"`	Backend mode
`apiKey`	string	—	Mem0 API key (platform mode, starts with `m0-`)
`userId`	string	`"default"`	Unique identifier for the user
`autoRecall`	boolean	`true`	Inject memories before each agent turn

Key	Type	Default	Description
`autoCapture`	boolean	`true`	Store facts after each agent turn
`topK`	number	`5`	Max memories per recall
`searchThreshold`	number	`0.3`	Min similarity score (0–1)
`orgId`	string	—	Mem0 organization ID
`projectId`	string	—	Mem0 project ID
`enableGraph`	boolean	`false`	Enable entity graph for relationships
`customInstructions`	string	—	Rules for what Mem0 should store/exclude

Agent Memory Tools

Once the plugin is active, the agent can use these tools during conversations:

Tool	Description
`memory_search`	Search memories by natural language
`memory_list`	List all stored memories for a user
`memory_store`	Explicitly save a fact
`memory_get`	Retrieve a memory by ID
`memory_forget`	Delete by ID or by query

CLI Commands

Run these inside the sandbox:

```
# Search all memories (long-term + session)
openclaw mem0 search "what languages does the user know"

# Search only long-term memories
openclaw mem0 search "user preferences" --scope long-term

# Search only session/short-term memories
```

```
openclaw mem0 search "current task" --scope session

# Memory stats
openclaw mem0 stats
```

Known Limitations

Open-source mode not supported in NemoClaw sandboxes

The Mem0 open-source mode requires calling `/v1/embeddings` on an external LLM provider (OpenAI, etc.). NemoClaw's sandbox proxy intercepts all OpenAI-compatible API requests but only allows `/v1/chat/completions` through. The `/v1/embeddings` endpoint is blocked at the inference interception layer (not the network policy layer). This is a limitation in OpenShell 0.0.10. Use **platform mode** instead.