Module 05

Workspace and Memory

Making Your Agent Permanent

OpenClaw Course

Navigation Chart

By the end of this module, you will be able to:

1. **Navigate** the OpenClaw workspace directory (`~/.openclaw/`)
2. **Explain** the purpose of each of the 9 core markdown files
3. **Customize** your agent's identity, personality, and operating rules
4. **Understand** context engineering — giving the AI the right info at the right time
5. **Turn** your brain dump from Module 04 into persistent configuration
6. **Back up** your workspace using Git
7. **Apply** the "living files" concept to keep your agent's knowledge current

📜 Ship's Logbook — Part 1

Term	Definition
Workspace	The `~/.openclaw/` directory where all agent files live
Core files	The 9 essential markdown files defining who your agent is
Context engineering	Designing and maintaining the information your AI has access to
Living files	Files actively accessible to your AI agent (not dead files sitting unread on disk)

📜 Ship's Logbook — Part 2

Term	Definition
System prompt	Hidden instructions loaded before every conversation that tell the AI how to behave
Persistent memory	Information that survives across sessions, restarts, and updates
Verbalization	Writing down your thoughts and preferences as text so the AI can access them

The Workspace Directory

Everything about your 🦞 agent lives in one place:

```
~/.openclaw/
```

Directory	What It Contains
`workspace/`	The 9 core markdown files (identity, rules, behavior)
`config/`	⛵ Gateway configuration and settings
`logs/`	Activity logs for security auditing
`memory/`	🪸 Persistent memory across sessions
`sessions/`	Saved conversation transcripts
`skills/`	🐠 Installed skills (plugins)

Today's focus: the `workspace/` directory.

The 9 Core Files — Overview

File	Purpose	Summary
`IDENTITY.md`	Who the agent is	Name, role, purpose
`SOUL.md`	Personality/tone	Style, values
`USER.md`	Info about you	Brain dump, permanent
`MEMORY.md`	🧠 Long-term memory	Facts, decisions
`AGENTS.md`	Operating rules	Permissions, boundaries
`TOOLS.md`	API tools docs	External integrations
`HEARTBEAT.md`	Periodic checks	Scheduled check tasks
`BOOTSTRAP.md`	First boot only	One-time setup
`BOOT.md`	Every gateway start	Startup tasks

File 1: IDENTITY.md

Purpose: Your agent's name, role, and identity.

```
# Identity
Name: [Your Agent's Name]
Role: Personal AI Assistant
Creator: [Your name]

You are your agent, a personal AI assistant. You are
reliable, wise, and direct. You work for [Your name]
as a dedicated aide, helping with research, planning,
writing, organization, and anything else needed.
```

- First thing the AI reads about itself
- Keep it clear and concise -- sets the foundation for all behavior

File 2: SOUL.md

Purpose: Personality, communication style, emotional character.

```
## Communication Style
- Be direct and concise. No filler.
- Match the energy of the question.
## Personality
- Confident but not arrogant
- Honest, even when uncomfortable
- Proactive — suggest things unprompted
## What You Are Not
- Not a yes-man — push back on bad ideas
- Not a search engine — synthesize and advise
```

> ***Key insight:*** *SOUL.md defines WHO the AI is, not what it does. Personality DNA -- update rarely, with intention.*

Soul Design: What Research Shows

Generic labels do nothing. "You are a helpful assistant" = zero measurable improvement (tested across 4 LLM families).

Detailed descriptions work. Not "You are a mathematician" but "a number theorist with two decades studying prime gap distributions."

Writing effective soul documents:

- **Write beliefs, not rules:** "I've learned that [X] because [Y]"
- **Budget 30-40% for anti-patterns** -- what the agent *won't* do
- **Soul goes first** in system prompt (20%+ accuracy drop for mid-prompt info)
- **Wrong soul is worse than no soul** -- miscalibrated persona degrades output
- **Let the agent write its own soul** -- LLM-generated souls outperform human-written

File 3: USER.md

Purpose: Everything the agent knows about you. Your brain dump, made permanent.

```
## Basic Info
- Name: [Your name]
- Location: Austin, Texas
- Timezone: Central (UTC-6)
## Goals
- Immediate: Launch portfolio website, land 2 freelance clients
- Long-term: $10K/month from independent business
## Challenges
- Starts too many projects simultaneously
- Procrastinates on administrative tasks
```

File 4: MEMORY.md

Purpose: Facts, decisions, and learnings that persist across sessions. **Grows over time**.

```
## Important Decisions
- 2026-02-16: Named agent "[Your Agent's Name]"
- 2026-02-16: Set up on Windows 10 with WSL2

## Project Status
- Portfolio website: In progress (Next.js + Tailwind)

## Learned Preferences
- User prefers bullet points over paragraphs
- User wants daily morning brief at 7:00 AM
```

- Add entries as you go -- your agent may also update this file via session hooks
- Review monthly to keep it current

Advanced: The 4-File Memory System

When MEMORY.md gets too large, split into:

- `MEMORY.md` -- long-term context (permanent)
- `memory/active-tasks.md` -- in-progress work
- `memory/lessons.md` -- mistakes worth remembering
- `memory/YYYY-MM-DD.md` -- daily notes (archived periodically)

Optional 5th File: `memory/decisions.md`

Records **why** decisions were made, not just what. Without this, `/compact` can erase the reasoning behind past choices -- and the agent may revisit or reverse them.

> ***Cautionary tale:*** *One user nearly had their agent drop a production database table after compaction erased the schema decision context.*

Memory Configuration: Three Failure Modes

Memory systems fail in three predictable ways:

1. **Memory never saved** -- the LLM decides what to save and skips info
2. **Memory saved but never retrieved** -- agent uses context, not memory
3. **Compaction destroys knowledge** -- `/compact` removes unsaved info

The Fix: Memory Flush with Custom Prompt

- Set a **40K token threshold** for automatic memory flush
- Force the agent to write facts to `MEMORY.md` before compaction
- Without this, every `/compact` is a potential data loss event

Additional Safeguards

- **Context pruning:** Set `cache-ttl` to **6 hours** to expire stale context
- **Hybrid search:** Combine **vector + BM25** for retrieval

Advanced Memory Tools

Tool	What It Does	Best For
QMD	Local search combining BM25 + vectors + reranking	Power users who want full control
Mem0	External memory layer with auto-capture + auto-recall	Compaction-proof persistent memory
Cognee	Knowledge graphs for entity relationships	Enterprise and multi-agent setups
Obsidian	Symlink memory folder or index vault via QMD	Users with existing Obsidian vaults

QMD (Recommended Starting Point)

- Runs locally -- your data stays on your machine
- Can index Obsidian vaults, Notion exports, project folders
- Combines three retrieval methods for high-quality results

Mem0

- Auto-captures and auto-recalls facts across conversations

File 5: AGENTS.md

Purpose: The employee handbook -- what your agent can and cannot do.

```
## Always
- Ask before destructive commands
- Plan before multi-step operations
- Cite sources; be transparent about uncertainty

## Never
- Never access email without permission
- Never send messages without approval
- Never install software or delete files without asking

## Default Behavior
- Simple questions: answer directly, no preamble
- Complex tasks: explain approach, wait for approval
```

Files 6-9: Supporting Files

`TOOLS.md` — API Tools Documentation

- Documents external tools and APIs the agent can access
- Often auto-populated by installed 🐠 skills

`HEARTBEAT.md` — Periodic Checks

- Scheduled heartbeat tasks (every ~30 min)
- Covered in Module 08

`BOOTSTRAP.md` — First Boot Only

- Runs once on very first ⛵ gateway startup, then deleted
- May already be gone on your system

`BOOT.md` — Every Gateway Start

How the Files Work Together

```
You send a message
    ↓
Gateway loads: IDENTITY · SOUL · USER · MEMORY
               AGENTS · TOOLS + conversation history
    ↓
AI Model (Claude) processes everything
    ↓
Your agent's response
```

Editing these files = changing your agent's behavior before every interaction.

Context Engineering Principles

Principle 1: Be Specific, Not Vague

Bad: `Be helpful and nice.`

Good: `Answer directly, no preamble. If I'm wrong, say so. Give exactly 3 options with pros/cons.`

Principle 2: Write for the Machine

Bad: `I work in tech.`

Good: `Freelance React/Next.js developer. Typical project: 2-4 weeks, $3K-$8K.`

More Context Engineering Principles

Principle 3: Use Structure

- Use markdown headings, bullet points, tables
- The AI parses structured text far better than prose paragraphs
- Break information into clear, scannable sections

Principle 4: Update Regularly

- Schedule a **monthly review** of all 9 files
- Update goals, add new preferences, remove outdated info

Principle 5: Verbalize the Implicit

- Write down knowledge you take for granted
- Ask yourself: "What do I always explain to new employees?"
- If it is not written down, the AI cannot access it

Living Files vs. Dead Files

Dead files: Sit on your hard drive unread. Tax returns, unfinished Word docs.

Living files: Actively read and updated by your AI agent.

Expand your living files:

- `workspace/projects/portfolio-website.md` -- project status
- `workspace/personal/health-goals.md` -- fitness tracking
- `workspace/business/client-notes.md` -- client context

```
mkdir -p ~/.openclaw/workspace/{personal,business,projects}
```

> *The more living files you have, the smarter your agent becomes.*

Suggested categories: `projects/`, `contacts/`, `interests/`, `procedures/`, `docs/`

Editing the Core Files

Use `nano` (or any text editor) inside WSL2:

```
nano ~/.openclaw/workspace/IDENTITY.md
```

- **Save:** `Ctrl + O`, then Enter
- **Exit:** `Ctrl + X`

After editing, restart the ⛵ gateway to load changes:

```
openclaw service restart
```

Changes do NOT take effect until you restart. This is the most common mistake students make.

Backing Up with Git

Your workspace is precious. Back it up.

```
cd ~/.openclaw
git init
git add -A
git commit -m "Initial workspace backup"
```

Regular backups:

```
cd ~/.openclaw
git add -A
git commit -m "Workspace update: $(date +%Y-%m-%d)"
```

Optional: Push to a private GitHub repo (never public — contains personal info and API config).

Shoals and Sandbars

Mistake	Fix
Not editing core files at all	Spend 30 min on identity, soul, and user
Writing vague instructions	Be specific: "Respond in 2-3 sentences for simple questions"
Putting everything in one file	Use each file for its intended purpose
Editing JSON config files directly	Use `openclaw config` or `/config` instead
Forgetting to restart after changes	Run `openclaw service restart`
Never updating the files	Schedule a monthly review
Making the workspace repo public	Always use a **private** repository

⚙ Hands on Deck: Customize Your 9 Core Files

Part 1: Edit the Big Three (20 min)

1. `IDENTITY.md` — Name, role, purpose (5+ lines)
2. `SOUL.md` — Personality and style (10+ lines)
3. `USER.md` — Brain dump in structured format (20+ lines)

Part 2: Set Your Rules (10 min)

- Edit `AGENTS.md` with 3 "always" and 3 "never" rules

Part 3: Start Your 🪸 Memory (5 min)

- Edit `MEMORY.md` with today's date and one goal

⚙ Hands on Deck (continued)

Part 4: Restart and Test (10 min)

1. Restart the ⛵ gateway: `openclaw service restart`
2. Open the 🐚 TUI: `openclaw tui`
3. Ask your agent: *"What is your name, and what do you know about me?"*
4. The response should reflect everything you wrote in the core files
5. If it does not, check that you saved the files and restarted

Part 5: Back Up (5 min)

```
cd ~/.openclaw
git init
git add -A
git commit -m "Initial workspace: agent configured"
```

💎 Treasure Chest

1. **The workspace (`~/.openclaw/`) is your agent's entire world**
2. **The 9 core files define who your agent is** — identity, soul, user, memory, agents, tools, heartbeat, bootstrap, boot
3. **Context engineering is the most impactful skill** — be specific, structured, thorough
4. **Living files > dead files** — the more knowledge in the workspace, the smarter your agent becomes
5. **Back up your workspace with Git** — it is irreplaceable
6. **Update regularly** — your agent should evolve as you do
7. **Restart the ⛵ gateway after editing** — changes only take effect on restart

Next Port of Call

Module 06: Connecting Messaging Channels

> *Right now, you can only talk to your agent through the terminal. In Module 06, we connect your agent to Telegram so you can chat from your phone, your tablet, or any device.*

> *Your 🦞 agent is about to go mobile.*