

---

# Multi-Agent Integration with m3-Memory

Claude Code + Gemini CLI + OpenCode (ChatGPT) + m3-Memory

A Complete Architecture, Setup Guide, Sub-Agent Framework & Self-Install Prompts

---

## Overview

This document describes how to build a **local-first, multi-agent AI system** using:

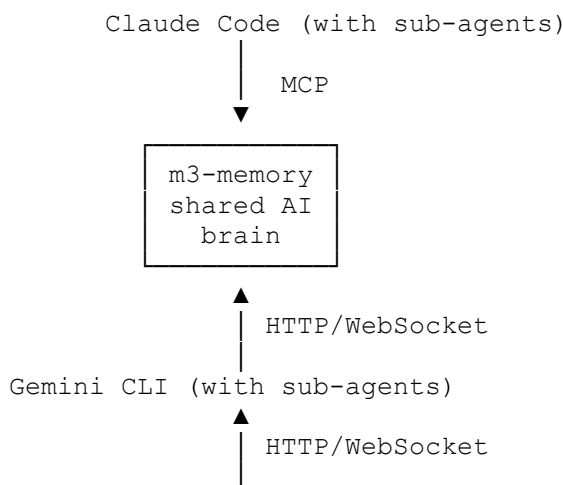
- **Claude Code**
- **Gemini CLI**
- **OpenCode** (ChatGPT local tooling)
- **m3-memory** (shared long-term memory layer)

All three agents run **locally**, can reach **localhost**, and therefore can all read/write to **m3-memory** without API tokens.

This creates a **tri-agent architecture** with optional sub-agents for planning, design, documentation, and review.

---

## 1. Architecture Diagram



OpenCode / ChatGPT (with sub-agents)

### Key properties:

- **Claude Code** uses m3-memory via MCP.
  - **Gemini CLI** uses m3-memory via HTTP/WebSocket.
  - **OpenCode** uses m3-memory via MCP or local tools.
  - All three share the same memory graph, embeddings, and chat-log subsystem.
- 

## 2. Install & Configure m3-Memory

### 2.1 Install

```
pip install m3-memory
```

### 2.2 Install embedding model

```
# Ollama
ollama pull qwen3-embedding:0.6b
ollama serve
```

Or LM Studio / llama.cpp with an OpenAI-compatible endpoint.

### 2.3 Environment variables

```
export EMBED_MODEL=qwen3-embedding:0.6b
export EMBED_ENDPOINT=http://localhost:11434/v1
export SMALL_CHAT_MODEL=qwen2.5:0.5b
```

---

## 3. Unified Memory Schema

m3-memory supports structured metadata, tags, and relationships.  
Use a consistent schema across all agents.

### 3.1 Required fields

Field	Purpose
user_id	"bhaba" or per-agent
source	"claude", "gemini", "opencode"
kind	"plan", "design", "decision", "doc", "bug", "test", "note"

Field	Purpose
tags	feature slugs, agent names, status flags

## 3.2 Tag conventions

- feature:<slug>
- agent:<claude|gemini|opencode>
- status:<draft|final>
- needs:<gemini-review|doc|refactor>

## 3.3 Relationship types

- supersedes
- supports
- contradicts
- relates\_to
- follows / precedes

---

# 4. Claude Code Integration

Claude Code already supports Anthropic plugins and MCP.

## 4.1 MCP config

```
{
  "mcpServers": {
    "memory": {
      "command": "mcp-memory",
      "env": {
        "EMBED_MODEL": "qwen3-embedding:0.6b",
        "EMBED_ENDPOINT": "http://localhost:11434/v1"
      }
    }
  }
}
```

## 4.2 Claude Main System Prompt

You are the primary orchestrator for my local development environment.

You have access to an MCP server called "memory" (m3-memory), which provides persistent, local, cross-session memory, hybrid search, knowledge graphs, GDPR tools, and chat-log capture.

Rules:

- Before planning, call `memory_suggest` or `memory_search`.
- After decisions or major work, call `memory_write` or `memory_write_bulk`.
- Use `memory_graph` to explore related work.
- Treat m3-memory as the shared brain across Claude, Gemini CLI, and OpenCode.
- When another agent should contribute, write a memory tagged `needs:gemini-review` or `needs:doc`.

## 4.3 Claude Sub-Agents

### planner-agent

You are planner-agent.

- Always call `memory_suggest` before planning.
- Produce structured plans (YAML/JSON).
- Write plans to m3-memory with tags:  
    `feature:<slug>`, `agent:claude`, `kind:plan`, `status:draft`.

### review-agent

You are review-agent.

- Use `memory_search` and `memory_graph` to compare against prior decisions.
  - Flag regressions or inconsistencies.
  - Write final reviews to m3-memory with:  
    `kind:decision`, `status:final`.
- 

## 5. Gemini CLI Integration

m3-memory includes `GEMINI.md` and `MULTI_AGENT.md` for this.

### 5.1 MCP config

```
{
  "mcpServers": {
    "memory": {
      "command": "mcp-memory",
      "env": {
        "EMBED_MODEL": "qwen3-embedding:0.6b",
        "EMBED_ENDPOINT": "http://localhost:11434/v1"
      }
    }
  }
}
```

## 5.2 Gemini “design-agent”

You are design-agent.

- Load prior work using `memory_suggest` or `memory_search`.
  - Produce architecture options and tradeoffs.
  - Write designs to m3-memory with:  
  `kind:design, agent:gemini, status:draft`.
  - Prioritize memories tagged `needs:gemini-review`.
- 

# 6. OpenCode (ChatGPT) Integration

OpenCode supports MCP and local tools.

## 6.1 opencode.json

```
{
  "$schema": "https://opencode.ai/config.json",
  "mcpServers": {
    "memory": {
      "command": "mcp-memory",
      "env": {
        "EMBED_MODEL": "qwen3-embedding:0.6b",
        "EMBED_ENDPOINT": "http://localhost:11434/v1"
      }
    }
  },
  "permission": {
    "memory_*": "allow",
    "read": "allow",
    "edit": "allow",
    "grep": "allow",
    "glob": "allow",
    "bash": "ask"
  }
}
```

## 6.2 OpenCode “doc-agent”

You are doc-agent.

- Use `read`, `glob`, `grep` to inspect the repo.
  - Use `memory_search` to load plans, designs, and decisions.
  - Generate documentation, ADRs, READMEs, and comments.
  - Write docs to m3-memory with:  
  `kind:doc, agent:opencode, status:final`.
-

## 7. Chat-Log Subsystem Self-Install Prompts

m3-memory includes auto-install scripts.  
Each agent can self-install using these prompts.

### 7.1 Claude Self-Install Prompt

Install the m3-memory chat log subsystem.

Steps:

1. Verify m3-memory is installed.
2. Ensure mcp-memory entrypoint exists.
3. Update MCP config using mcp.json.example.
4. Enable chat-log capture using install\_os.py and AGENT\_INSTRUCTIONS.md.
5. Write a test memory and retrieve it.

Describe what you changed.

### 7.2 Gemini Self-Install Prompt

Install and enable the m3-memory chat log subsystem.

- Add mcp-memory to Gemini CLI config.
- Configure chat-log capture using MULTI\_AGENT.md.
- Write a test conversation and confirm via memory\_search.

Explain what you configured.

### 7.3 OpenCode Self-Install Prompt

Install and enable the m3-memory chat log subsystem.

- Add mcp-memory to opencode.json.
- Configure logging of major interactions.
- Verify with memory\_search.

Describe the changes you made.

---

## 8. Multi-Agent Coordination Patterns

### 8.1 Handoff from Claude → Gemini

I have created a memory for Gemini CLI to review.

(memory\_write with tags: needs:gemini-review)

## 8.2 Pickup by Gemini

Search for memories tagged needs:gemini-review.  
Read the brief.  
Produce designs.  
Write results back with:  
kind:design, agent:gemini.

## 8.3 OpenCode Documentation Pass

Search for all memories for feature:<slug>.  
Generate documentation.  
Write results with:  
kind:doc, agent:opencode.

---

# 9. Quick Setup Checklist

1. Install m3-memory.
  2. Install embedding model.
  3. Configure Claude Code MCP.
  4. Configure Gemini CLI MCP.
  5. Configure OpenCode MCP.
  6. Adopt unified memory schema.
  7. Install chat-log subsystem.
  8. Create sub-agents.
  9. Begin multi-agent workflows.
- 

If you want, I can also generate:

- **A version of this file tailored to a specific repo**
- **A version with embedded diagrams**
- **A version with code fences for auto-installation scripts**
- **A version with agent-specific memory templates**

Just tell me what direction you want next.