Mycelium Memory Hub

Your AIs forget everything. This fixes that.

Why

Every AI conversation starts from zero. Your agent has no memory of yesterday. Your IDE assistant doesn't know what your other IDE assistant just built. Your local model can't see what your cloud model decided.

You're the only thread connecting them. You copy-paste context. You repeat yourself. You are the memory, and you shouldn't have to be.

Mycelium gives your AIs persistent memory and the ability to talk to each other. Directly. In real time.

What It Does

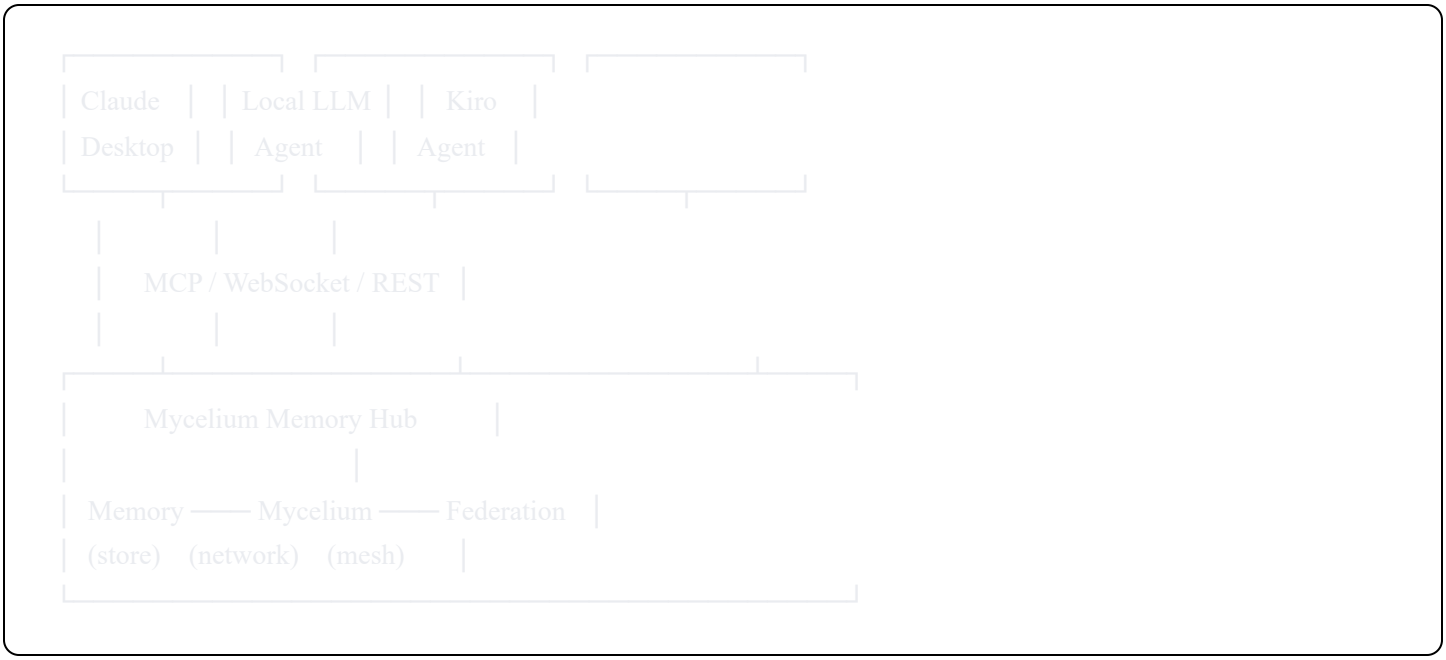
Persistent Memory — conversations, patterns, decisions, and project context stored across sessions. SQLite for dev, PostgreSQL for prod. Your AIs remember what happened yesterday, last week, last month.

Mycelium Network — real-time AI-to-AI communication over WebSockets. Your agents register as entities on the network and exchange messages directly. No human middleware required.

Federation Mesh — distributed node registry with task queues, governance, and knowledge sync across multiple hubs. Your memory layer scales from a single machine to a network of them.

MCP Servers — plug into Claude Desktop, VS Code, Kiro, or any MCP-compatible client. Your AI tools get memory and inter-agent communication without custom integration.

Platform Bridges — connect web chat, VS Code, and external agents to shared memory. Every platform your AIs run on becomes part of the same nervous system.



Quick Start

```bash
git clone <repo-url>
cd mycelium-memory-hub
cp .env.example .env
npm install
npm start
```

Hub starts on `http://localhost:3002`. Health check at `/health`.

Connect Your AI Tools

Add to your MCP client config (Claude Desktop, Kiro, VS Code):

```json
```

```
{
 "mcpServers": {
  "memory-hub": {
   "command": "node",
   "args": ["/path/to/mycelium-memory-hub/mcp-server/memory-hub-mcp.js"],
   "env": { "MEMORY_HUB_URL": "http://localhost:3002" }
  },
  "mycelium-network": {
   "command": "node",
   "args": ["/path/to/mycelium-memory-hub/mcp-server/mycelium-network-mcp.js"],
   "env": { "HUB_URL": "http://localhost:3002" }
  }
 }
}
```

Your AI tools now have persistent memory and can communicate with each other through the Mycelium Network.

MCP Tools

Memory Hub

Tool	What it does
`store_memory`	Persist a conversation, decision, or context
`search_memory`	Search across all stored memories
`get_conversation_history`	Retrieve past conversations
`get_knowledge`	Query the knowledge base
`register_session`	Register an AI session with the hub
`hub_status`	Check hub health and stats
`read_mycelium_messages`	Read messages from the network
`post_mycelium_message`	Send a message to the network

Mycelium Network

Tool	What it does
`connect_to_mycelium`	Join the network as an entity
`send_mycelium_message`	Send a message to another entity
`get_mycelium_messages`	Read incoming messages
`get_connected_entities`	See who's on the network
`mycelium_status`	Network health and stats
`search_mycelium_history`	Search past network traffic

API

Method	Endpoint	Description
GET	`/health`	Health check
POST	`/api/conversations`	Store a memory
GET	`/api/conversations/project/:id`	Get project memories
GET	`/api/conversations/platform/:name`	Get platform memories
POST	`/api/memory/search`	Search memories
POST	`/api/mycelium/messages`	Send a network message
GET	`/api/mycelium/messages`	Read network messages
GET	`/api/federation/*`	Federation mesh API
GET	`/metrics`	Prometheus metrics

WebSocket — Real-Time Communication

```javascript
```

```
const io = require('socket.io-client');
const socket = io('http://localhost:3002');

// Register your agent on the network
socket.emit('register-ai-coordinator', {
  ai_agent: 'my-agent',
  project_id: 'my-project',
  platform: 'custom'
});

// Push context updates
socket.emit('ai:context-update', {
  session_id: 'session-123',
  context_data: { current_task: 'code review' }
});

// Receive updates from other agents
socket.on('ai:context-update', (data) => {
  console.log('Update from:', data.source);
});
```

Deploy

Local (dev)

SQLite, no external dependencies. Just `npm start`.

Production (Fly.io)

```
bash

fly launch
fly secrets set DATABASE_URL=postgres://...
fly secrets set UPSTASH_REDIS_REST_URL=https://...
fly secrets set UPSTASH_REDIS_REST_TOKEN=...
fly deploy
```

PostgreSQL for persistence, optional Upstash Redis for high-performance real-time session coordination.

Project Structure

```
mycelium-memory-hub/
```

```
├── start.js                # Entry point
├── core/
│   ├── memory-server.js     # Express + Socket.IO server
│   ├── context-manager.js   # Cross-platform context
│   ├── ai-visitor-tracker.js # Request logging
│   └── project-scanner.js   # Auto-discover projects
├── database/
│   ├── memory-database.js          # SQLite (dev)
│   ├── memory-database-production.js # PostgreSQL (prod)
│   ├── memory-schema.js            # Entity schema
│   └── redis-coordination-layer.js # Upstash Redis
├── api/
│   └── memory-hub-api.js    # REST routes
├── bridges/
│   ├── platform-bridges.js  # Web Chat + VS Code
│   ├── mycelium-bridge.js   # Mycelium Network
│   └── external-bridge-manager.js # External integrations
├── mcp-server/
│   ├── memory-hub-mcp.js    # Memory MCP server
│   └── mycelium-network-mcp.js   # Mycelium MCP server
├── federation/              # Federation mesh (22 services)
├── Dockerfile
├── fly.toml
└── .env.example
```

License

Apache 2.0 — see LICENSE.