Agentic Threat Hunting Framework (ATHF)

Quick Start • **Installation** • **Documentation** • **Examples**

> Give your threat hunting program memory and agency.

The **Agentic Threat Hunting Framework (ATHF)** is the memory and automation layer for your threat hunting program. It gives your hunts structure, persistence, and context - making every past investigation accessible to both humans and AI.

ATHF works with any hunting methodology (PEAK, TaHiTI, or your own process). It's not a replacement; it's the layer that makes your existing process AI-ready.

What is ATHF?

ATHF provides structure and persistence for threat hunting programs. It's a markdown-based framework that:

- Documents hunts using the LOCK pattern (Learn → Observe → Check → Keep)
- Maintains a searchable repository of past investigations
- Enables AI assistants to reference your environment and previous work
- Works with any SIEM/EDR platform

The Problem

Most threat hunting programs lose valuable context once a hunt ends. Notes live in Slack or tickets, queries are written once and forgotten, and lessons learned exist only in analysts' heads.

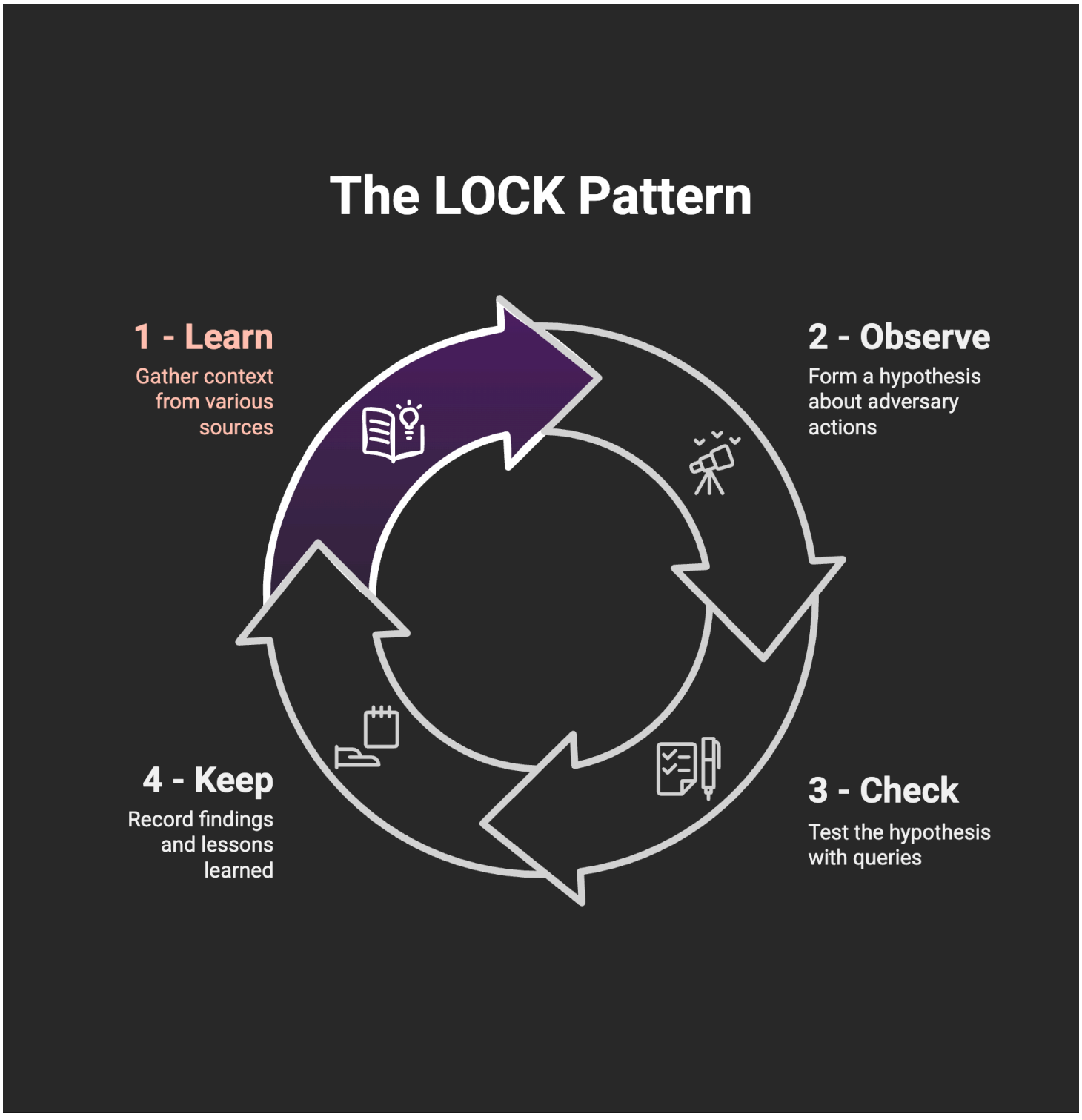
Even AI tools start from zero every time without access to your environment, your data, or your past hunts.

ATHF changes that by giving your hunts structure, persistence, and context.

Read more: docs/why-athf.md

The LOCK Pattern

Every threat hunt follows the same basic loop: **Learn** → **Observe** → **Check** → **Keep**.



- **Learn:** Gather context from threat intel, alerts, or anomalies
- **Observe:** Form a hypothesis about adversary behavior
- **Check:** Test hypotheses with targeted queries
- **Keep:** Record findings and lessons learned

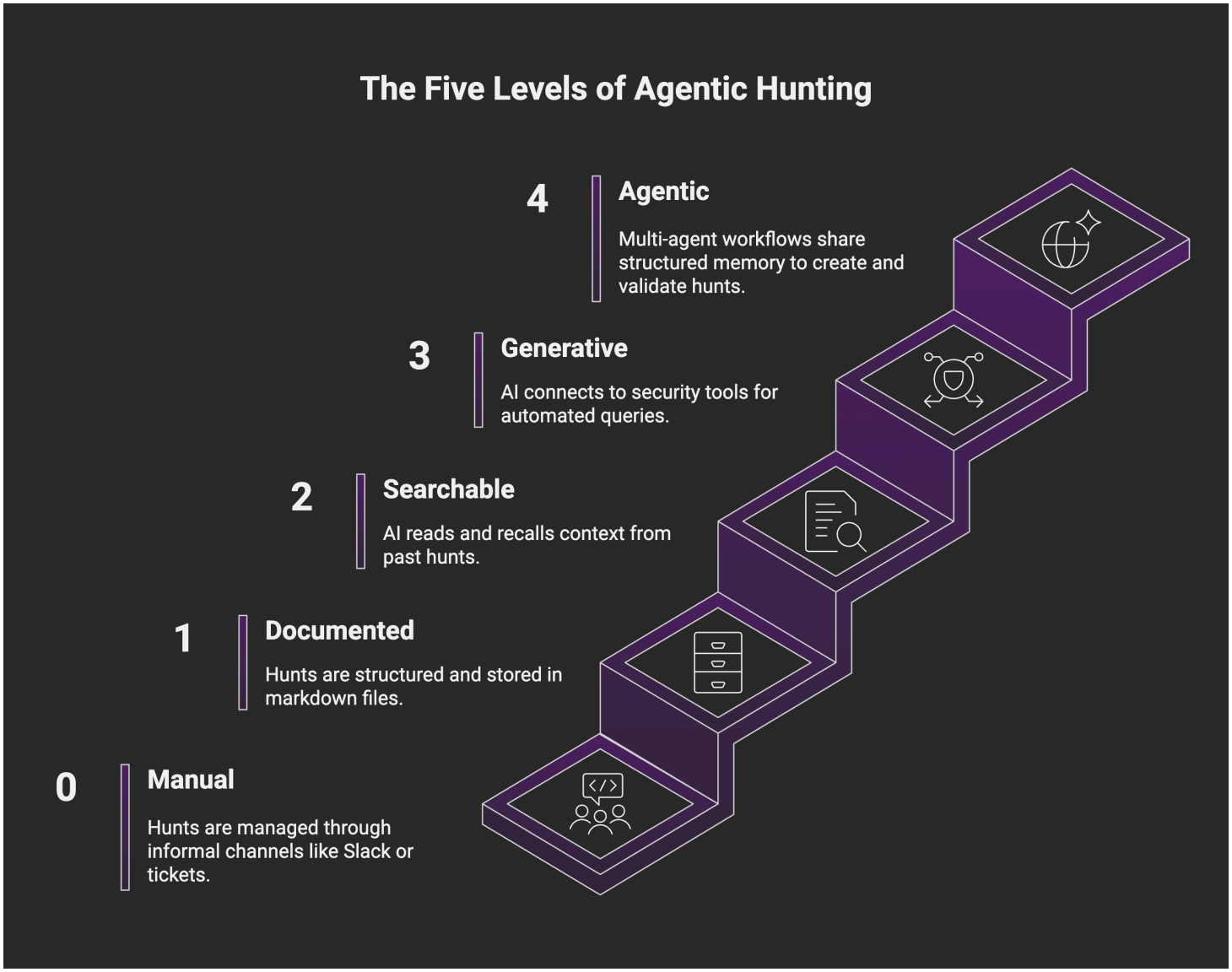
Why LOCK? It's small enough to use and strict enough for agents to interpret. By capturing every hunt in this format, ATHF makes it possible for AI assistants to recall prior work and suggest refined queries based on past results.

Read more: docs/lock-pattern.md

The Five Levels of Agentic Hunting

ATHF defines a simple maturity model. Each level builds on the previous one.

Most teams will live at Levels 1–2. Everything beyond that is optional maturity.



Level	Capability	What You Get
0	Ad-hoc	Hunts exist in Slack, tickets, or analyst notes
1	Documented	Persistent hunt records using LOCK
2	Searchable	AI reads and recalls your hunts

Level	Capability	What You Get
3	Generative	AI executes queries via MCP tools
4	Agentic	Autonomous agents monitor and act

Level 1: Operational within a day
Level 2: Operational within a week
Level 3: 2-4 weeks (optional)
Level 4: 1-3 months (optional)

Read more: docs/maturity-model.md

🚀 Quick Start

Option 1: Python CLI (Recommended)

```
# Install from source (PyPI package coming soon)
git clone https://github.com/Nebulock-Inc/agentic-threat-hunting-framework
cd agentic-threat-hunting-framework
pip install -e .

# Initialize your hunt program
athf init

# Create your first hunt
athf hunt new --technique T1003.001 --title "LSASS Credential Dumping"
```

Option 2: Pure Markdown (No Installation)

```
# Clone the repository
git clone https://github.com/Nebulock-Inc/agentic-threat-hunting-framework
cd agentic-threat-hunting-framework

# Copy a template and start documenting
cp templates/HUNT_LOCK.md hunts/H-0001.md

# Customize AGENTS.md with your environment
# Add your SIEM, EDR, and data sources
```

Choose your AI assistant: Claude Code, GitHub Copilot, or Cursor - any tool that can read your repository files.

Full guide: docs/getting-started.md

🔧 CLI Commands

ATHF includes a full-featured CLI for managing your hunts. Here's a quick reference:

Initialize Workspace

```
athf init                          # Interactive setup
athf init --non-interactive        # Use defaults
athf init --siem sentinel --edr defender
```

Create Hunts

```
athf hunt new                      # Interactive mode
athf hunt new \
  --technique T1003.001 \
  --title "LSASS Dumping Detection" \
  --platform windows
```

List & Search

```
athf hunt list                     # Show all hunts
athf hunt list --status completed  # Filter by status
athf hunt list --output json       # JSON output
athf hunt search "kerberoasting"   # Full-text search
```

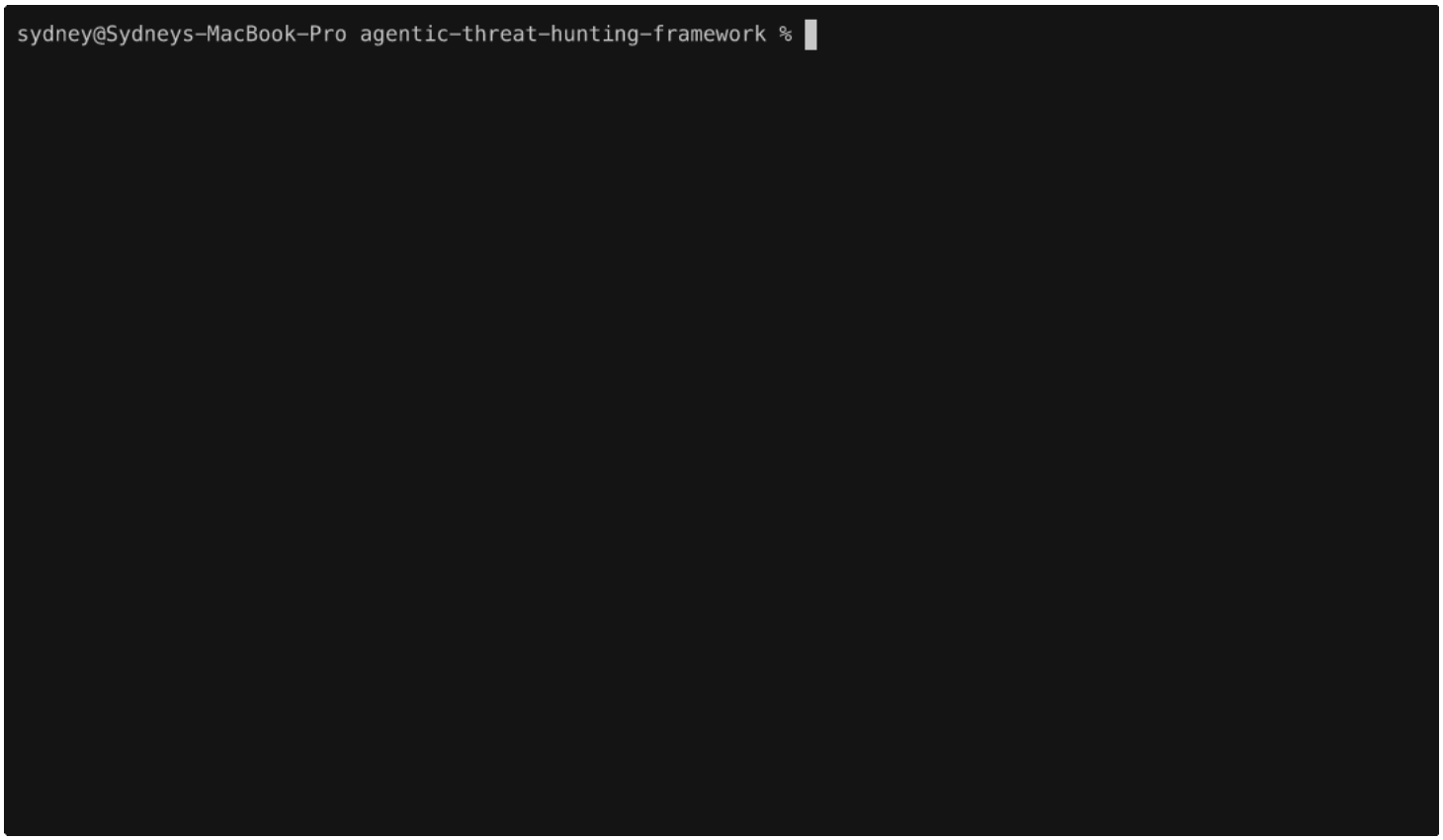
Validate & Stats

```
athf hunt validate                 # Validate all hunts
athf hunt validate H-0001          # Validate specific hunt
athf hunt stats                    # Show statistics
athf hunt coverage                 # MITRE ATT&CK coverage
```

Full documentation: CLI Reference

> **Note:** The CLI is fully functional and ready to use. Install from source using `pip install -e .` in the repository directory. PyPI package publication coming soon.

📺 See It In Action



Watch ATHF in action: initialize a workspace, create hunts, and explore your threat hunting catalog in under 60 seconds.

View example hunts →

Installation

Prerequisites

- Python 3.8+ (for CLI option)
- Git
- Your favorite AI code assistant

CLI Installation

From Source:

```
git clone https://github.com/Nebulock-Inc/agentic-threat-hunting-framework
cd agentic-threat-hunting-framework
pip install -e .
```

PyPI Installation (Coming Soon):

```
pip install athf-framework
```

Markdown-Only Setup (No CLI)

```
git clone https://github.com/Nebulock-Inc/agentic-threat-hunting-framework
cd agentic-threat-hunting-framework
```

Start documenting hunts in the `hunts/` directory using the LOCK pattern.

Documentation

Core Concepts

- Why ATHF Exists - The problem and solution
- The LOCK Pattern - Structure for all hunts
- Maturity Model - The five levels explained
- Getting Started - Step-by-step onboarding

Level-Specific Guides

- Level 1: Documented Hunts
- Level 2: Searchable Memory
- Level 3: Generative Capabilities
- Level 4: Agentic Workflows

Integration & Customization

- MCP Catalog - Available tool integrations

- Quickstart Guides - Setup for specific tools
- Using ATHF - Adoption and customization

🎖️ Featured Hunts

H-0001: macOS Information Stealer Detection

Detected Atomic Stealer collecting Safari cookies via AppleScript.
Result: 1 true positive, host isolated before exfiltration.

Key Insight: Behavior-based detection outperformed signature-based approaches. Process signature validation identified unsigned malware attempting data collection.

View full hunt → | See more examples →

Why This Matters

You might wonder how this interacts with frameworks like PEAK. PEAK gives you a solid method for how to hunt. ATHF builds on that foundation by giving you structure, memory, and continuity. PEAK guides the work. ATHF ensures you capture the work, organize it, and reuse it across future hunts.

Agentic threat hunting is not about replacing analysts. It's about building systems that can:

- Remember what has been done before
- Learn from past successes and mistakes
- Support human judgment with contextual recall

When your framework has memory, you stop losing knowledge to turnover or forgotten notes. When your AI assistant can reference that memory, it becomes a force multiplier.

💬 Community & Support

- **GitHub Discussions:** Ask questions, share hunts
- **Issues:** Report bugs or request features
- **Adoption Guide:** See USING_ATHF.md for how to use ATHF in your organization
- **LinkedIn:** Nebulock Inc. - Follow for updates

📖 Using ATHF

ATHF is a framework to internalize, not a platform to extend. Fork it, customize it, make it yours.

Repository: https://github.com/Nebulock-Inc/agentic-threat-hunting-framework

See USING_ATHF.md for adoption guidance. Your hunts stay yours—sharing back is optional but appreciated (Discussions).

The goal is to help every threat hunting team move from ad-hoc memory to structured, agentic capability.

Start small. Document one hunt. Add structure. Build memory.

Memory is the multiplier. Agency is the force.
Once your program can remember, everything else becomes possible.

Happy hunting!