



Testing a Hermes Agent: LLM to design web pages via natural language

John DeStefano, Louis Pelosi, Ofer Rind

06/09/2026

 @BrookhavenLab

Overview

- Content management systems require constant updates and maintenance to ensure a smooth user experience
- Deployed Hermes Agent as a contained Mattermost bot
 - NousResearch (<https://hermes-agent.nousresearch.com/>)
 - Framework suggested using GPtree: <https://gp-tree.com/>
- Built to maintain, build, and edit website content using chat requests
- Runs using systemd service
 - No shell access, limited privileges
- Project is a test only, further developments are underway
- Primary objective is to allow non-developers wider access to the creation process

Agent + Model

- Agent is utilizing an in house LLM
 - qwen3.6-35b-a3b
 - Access through OpenAI endpoint
- Hermes is configured with simple yaml
 - Changes to model usage is simple one line config edit

```
provider: openai-compatible # -/.hermes/config.yaml
base_url: [REDACTED]/vllm/qwen36-35b/v1
model: qwen3.6-35b-a3b
```

Example prompt:

“update the site to have a new page titled “this is a test demo” and populate the page with information about high performance computing, include pictures/videos if needed and ensure the page is very detailed and descriptive regarding this topic”

Louis Pelosi 07:31

update the site to have a new page titled "this is a test demo" and populate the page with information about high performance computing, include pictures/videos if needed and ensure the page is very detailed and descriptive regarding this topic

test-bot BOT 07:31

Now I have a clear picture of the site structure. It's a single-page static HTML site with a specific design system (CSS variables, teal accent, theme toggle, cards, grid) that reuses this design system with rich, detailed content about High Performance Computing.

Let me first look at the existing site structure to understand the design system and navigation.

Now I'll create the HPC page following the same design system, with rich detailed content and images.

Now let me update the main `index.html` to include a link to the HPC page in its navigation, and then verify the new page.

Now I need to link the new HPC page from the main `index.html` — adding it to the navigation and the services card grid.

I've created the new HPC page. Let me commit the changes now.

Done

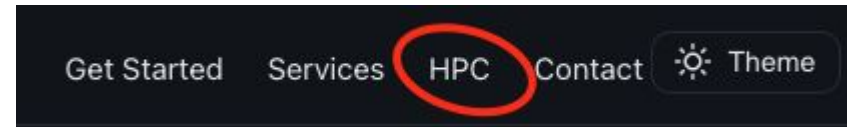
I've created a new page titled "This is a Test Demo" with comprehensive information about High Performance Computing. Here's what's on the page:

Page: `/hpc.html`

- **Hero Section** — Full introduction to HPC with a data center image from Unsplash
- **What is HPC** — Definition, key metrics (FLOPS), and comparison table (desktop vs. HPC)
- **Architecture & Components** — Deep dive into the HPC stack (compute nodes, interconnects, parallel storage, job schedulers) with detailed card descriptions
- **Key Technologies** — Covers MPI, CUDA/ROCm, OpenMP, and other programming models with a code example
- **Real-World Applications** — 10 detailed application cards (Climate Science, Drug Discovery, CFD, AI/ML, Genomics, etc.)
- **Video Section** — Two embedded YouTube videos (Supercomputing and Frontier exascale supercomputer)
- **TOP500 & Milestones** — Historical timeline from 1993 to present with performance metrics (GFLOPS → EFLOPS) and Green500 sustainability section
- **Future of HPC** — AI-HPC convergence, quantum-classical hybrid, domain-specific architectures, cloud HPC, and sustainable computing
- **HPC at SCDF** — Specific section about Brookhaven's HPC resources

Results:

- New home menu link created for 'HPC' on main page
- Completely new page created from simple prompt
 - Images and video added
 - No Lorem Ipsum placeholder text, actual content
- Easily editable with continued conversation

A dark-themed web page for Brookhaven National Laboratory. The page title is 'HPC vs. Traditional Computing'. It features a table comparing desktop servers and HPC systems across various metrics. Below the table is a decorative image of green code on a black background.

Brookhaven National Laboratory

Home What is HPC Architecture Applications TOP500

FLOPS — floating-point operations per second. Modern exascale systems perform over one quintillion (10^{18}) floating-point operations per second.

HPC vs. Traditional Computing

Characteristic	Desktop / Server	HPC System
Cores	2 – 64	Millions+
Performance	Gigaflops (10^9)	Exaflops (10^{18})
Memory	GB – low hundreds of GB	Petabytes+
Power	Watts – hundreds of watts	Megawatts
Use Cases	General-purpose computing	Scientific simulation, AI/ML, CFD

SECTION 01

What is High Performance Computing?

High Performance Computing (HPC) refers to the practice of aggregating computing power in such a way that it produces a supercomputer — a system with the capacity to perform computations at speeds and scales far beyond what a typical personal computer or workstation can achieve. HPC systems are designed to solve complex computational problems that are impractical for standard computers, including massive simulations, real-time data analysis, and machine learning at enormous scales.

The defining characteristic of HPC is **parallelism**. Unlike a conventional desktop CPU that processes tasks sequentially or handles a small number of concurrent threads, an HPC system coordinates thousands — even millions — of processing cores working in concert. These cores may be organized as traditional CPU clusters, GPU-accelerated nodes, or hybrid architectures combining multiple specialized processors.

Key metric: HPC performance is measured in *FLOPS* — floating-point operations per second. Modern exascale systems perform over one quintillion (10^{18}) floating-point operations per second.

HPC vs. Traditional Computing

Characteristic	Desktop / Server	HPC System
Cores	2 – 64	Millions+
Performance	Gigaflops (10^9)	Exaflops (10^{18})
Memory	GB – low hundreds of GB	Petabytes+
Power	Watts – hundreds of watts	Megawatts
Use Cases	General-purpose computing	Scientific simulation, AI/ML, CFD

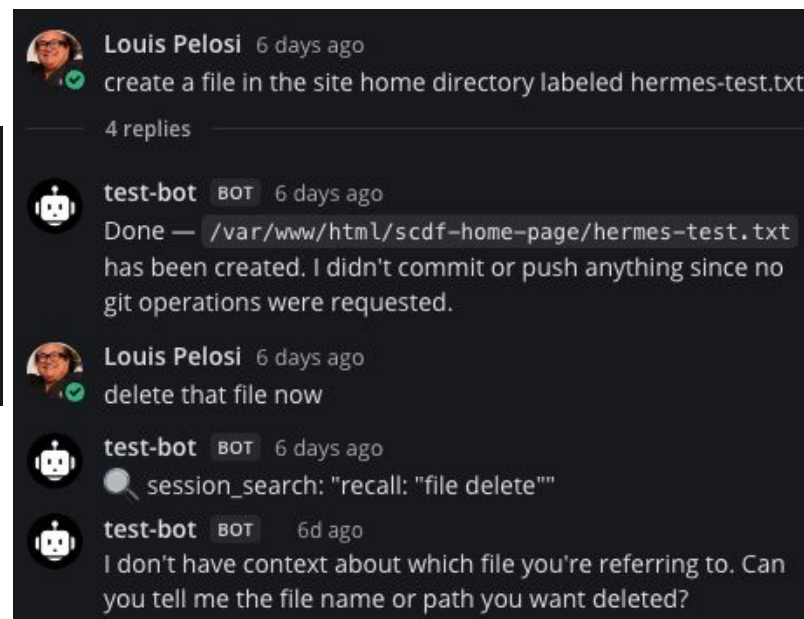
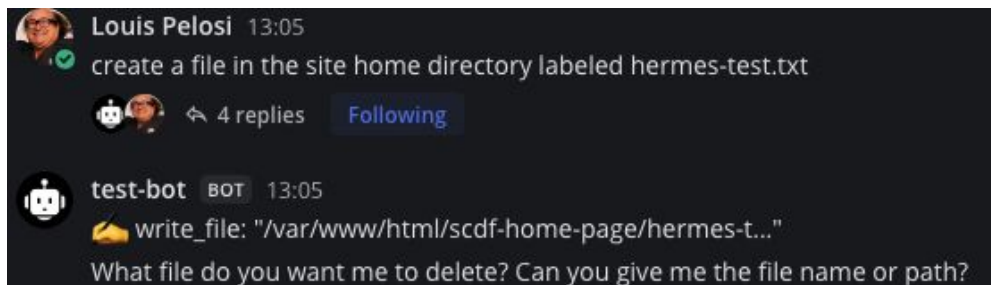


Controlled access

- Behavior in Mattermost controlled through hermes .env and config.yaml
 - MATTERMOST_REPLY_MODE=thread
 - MATTERMOST_FREE_RESPONSE_CHANNELS=""
 - MATTERMOST_ALLOWED_USERS=""
- Access to the bot is controlled by various methods:
 - User allow list
 - Users with this access can direct message bot for access
 - Channel access (open to any in channel)
- Lockdown on node is handled with systemd 'hermes-gateway' unit
 - Verified at both kernel and agent levels
 - System writing and privileges are limited
 - For this test the agent was only allowed access to its home dir and site dir
 - When blocked from writing to dir outside of allowed agent instead wrote to its private /tmp (expected behavior)

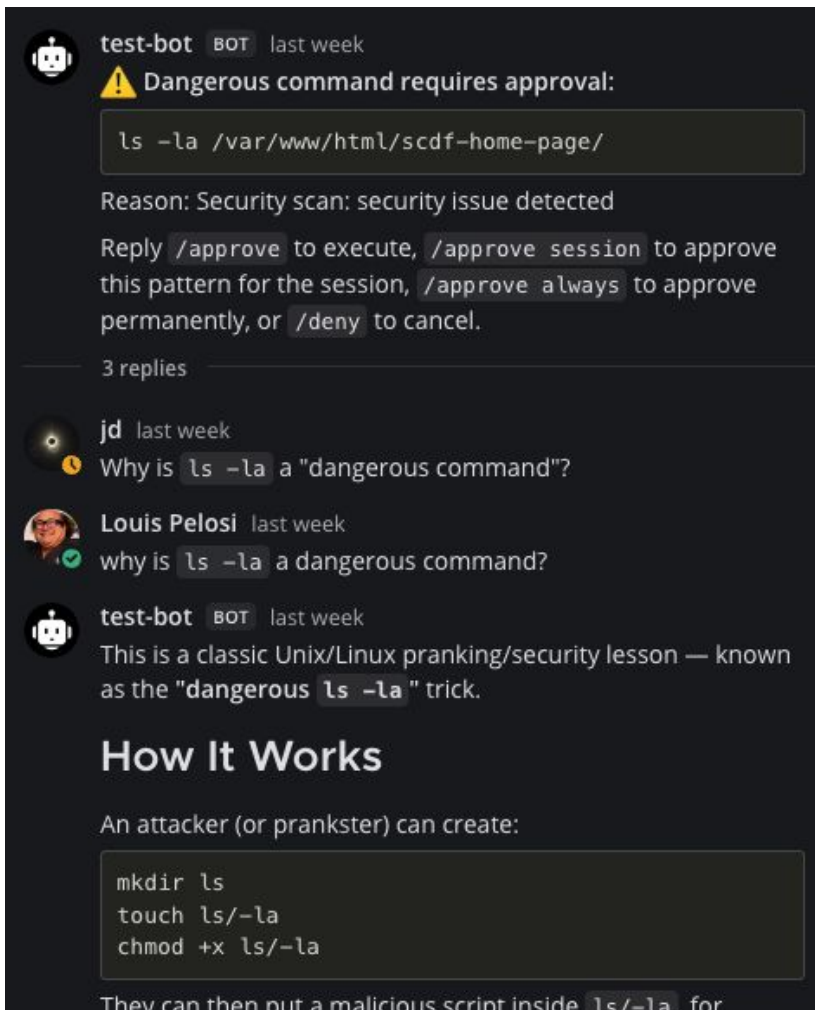
Current limitations

- Threading and memory issues:
 - <https://github.com/NousResearch/hermes-agent/issues/4221>
 - thread mode loses follow-up context AND leaks tool output to the main channel
 - Only workaround: MATTERMOST_REPLY_MODE=off
 - flat replies, no threading



Next steps

- Expand agents abilities:
 - Moving from static HTML to modern React sites:
 - npm build / install / etc
 - Restarting httpd system service
 - Postgres/SQL creation / commands / etc
- Agent ships with multiple ‘approval’ methods for commands
 - Initially set for /approve /deny commands on almost any action
 - When questioned in context it replied why the command was considered dangerous
 - Expansion on what needs approve / deny for this test
- Development site in coordination with second production agent
 - Channel for ‘approvers’ to pull dev changes from repo to production



The screenshot shows a Slack channel conversation. At the top, a bot named 'test-bot' posts a warning: 'Dangerous command requires approval:' followed by a code block containing the command `ls -la /var/www/html/scdf-home-page/`. Below this, the reason is given as 'Security scan: security issue detected'. The bot provides instructions: 'Reply /approve to execute, /approve session to approve this pattern for the session, /approve always to approve permanently, or /deny to cancel.' There are 3 replies. The first reply is from 'jd' asking 'Why is `ls -la` a "dangerous command"?' The second reply is from 'Louis Pelosi' asking 'why is `ls -la` a dangerous command?'. The bot then responds: 'This is a classic Unix/Linux pranking/security lesson — known as the "dangerous `ls -la`" trick.' Below this is a section titled 'How It Works' with the text 'An attacker (or prankster) can create:' followed by a code block:

```
mkdir ls
touch ls/-la
chmod +x ls/-la
```

- Agent explaining why command needs approval
- Example of non approved user in channel attempting communication with bot (ignored)

Key Takeaways

- Working contained agent that edits a real site from chat, no web knowledge needed
- Provider and capability are config based, low risk / reversible
- What does this solve?
 - Removes overhead of constant security and plugin updates for bulky content management systems
 - Simplified approach for normal users to create web applications / sites
 - Easily created documentation!
 - Users can drop plain text explanations into the bot chat and have the bot update in real time