Hermes AI Agent

Local VM Deployment & Secure Automation Platform

PROJECT SNAPSHOT

A secure, persistent AI-agent environment running inside an isolated Ubuntu Server VM, integrating OpenAI, Telegram, local system tools, web/browser automation, memory, scheduling, and tested recovery.

PLATFORM	AI PROVIDER	REMOTE INTERFACE	SECURITY
Ubuntu Server 22.04 LTS	OpenAI / Codex	Telegram Gateway	SSH keys + UFW + allowlist

What I built

I designed and deployed a self-hosted Hermes AI Agent environment on Oracle VirtualBox, deliberately moving beyond a basic chatbot installation. **The system was built to behave like a persistent, remotely accessible agent service with real tool execution, security boundaries, and recovery validation.**

- Remote interaction through Telegram, backed by OpenAI/Codex inference.
- Controlled terminal, file, browser, web-search, memory, skills, and cron capabilities inside the VM.
- Automatic gateway startup and recovery after logout, service restart, reboot, and snapshot rollback.
- Security hardening across Linux permissions, Telegram access control, SSH, firewalling, and privilege boundaries.

Architecture



The VM uses NAT for outbound Internet access and a host-only adapter for administrative access. A persistent private management address (192.168.56.10) separates VM administration from ordinary Internet traffic.

Engineering highlights

END-TO-END AGENT PATH

Validated Telegram → Hermes Gateway → OpenAI → local tool execution → Telegram response, including remote hostname execution.

PERSISTENT SERVICE

Installed Hermes Gateway as a systemd user service and proved operation after SSH logout, gateway restart, and full VM reboot.

AUTOMATION

Verified cross-session memory, scheduled cron delivery to Telegram, web search, skills, and browser automation.

RECOVERY

Created and restored a hardened VirtualBox snapshot, then proved filesystem rollback and successful post-restore Telegram/OpenAI operation.

NETWORK DESIGN

Used NAT for outbound services and host-only networking for isolated management, with a static management IP for repeatable administration.

EVIDENCE-DRIVEN BUILD

Every major phase was validated with observable outputs rather than assumed from configuration alone.

Troubleshooting that mattered

Diagnosing real deployment failures and proving the fixes.

Persistent host-only networking

The host-only NIC did not remain usable after reboot. I diagnosed the interface state, created a persistent Netplan configuration, corrected YAML indentation detected by `netplan generate`, assigned 192.168.56.10/24, and validated routing and SSH after reboot.

Minimal Ubuntu vs. Chromium

Local browser automation repeatedly failed because the minimal server image lacked Chromium runtime libraries. I traced each missing dependency (ATK, AT-SPI bridge, CUPS, XKB, XComposite, XDamage, XFixes, XRandR, GBM, Cairo, Pango, ALSA) until Hermes successfully launched Chromium and inspected the official documentation.

SSH configuration precedence

A first hardening file did not disable password authentication because cloud-init had already defined the setting. I diagnosed the effective configuration with `sshd -T`, moved the administrator policy to `00-hermes-hardening.conf`, and revalidated the final state.

Snapshot recovery mistake

My first restore selected the current state rather than the hardened snapshot. After identifying the error, I restored `02-Hermes-Hardened-Working`, confirmed the post-snapshot test marker disappeared, and verified Hermes and Telegram recovered successfully.

Security posture

- ✓ Dedicated non-root `hermes-agent` account
- ✓ Owner-only Hermes directory; `.env` and `config.yaml` set to 600
- ✓ Telegram allowlist configured; gateway-wide allow-all disabled
- ✓ Dedicated Ed25519 SSH key for this VM
- ✓ Root SSH login disabled; password SSH disabled
- ✓ UFW default-deny inbound policy
- ✓ SSH restricted to 192.168.56.0/24 host-only subnet
- ✓ Hermes internal listener bound to loopback
- ✓ Direct `/etc/shadow` access denied
- ✓ Non-interactive sudo elevation denied
- ✓ No obvious plaintext secrets found in gateway logs/process arguments

SECURITY RESULT

Hermes can execute useful local tools as the normal Linux account, but tested OS controls prevent silent privilege escalation. Remote Telegram control is restricted to an explicit allowlist.

Recovery & operational proof

- Gateway remained operational after SSH logout.

IS A WES COPY

- Gateway automatically recovered after full VM reboot.
- Manual gateway restart recovered cleanly and Telegram reconnected.
- Hardened snapshot restore removed a post-snapshot marker file as expected.
- After restore, `hermes-gateway.service` returned active (running) and Telegram returned `HERMES_SNAPSHOT_RECOVERY_OK`.

Technology stack

AI / Agent	Hermes Agent; OpenAI / Codex; GPT-6.1-Sol; Telegram Bot integration
Infrastructure	Oracle VirtualBox; Ubuntu Server 22.04 LTS; systemd; Netplan
Security	OpenSSH; Ed25519; UFW; Linux file permissions; Telegram allowlisting
Automation	Chromium browser automation; web search; cron; persistent memory; Hermes Skills
Validation	hermes doctor; systemd status; SSH effective config; firewall tests; snapshot restore testing

What this project demonstrates

Linux administration • AI-agent deployment • virtualization • networking • systemd • SSH hardening • firewalling • access control • browser automation • API integration • troubleshooting • persistence • recovery testing

TAKEAWAY

I did not treat the project as complete because the agent launched. I validated the full lifecycle: install → configure → integrate → secure → test → troubleshoot → persist → recover. The result is a working local AI-agent platform with documented operational and security boundaries.

Evidence available

A deeper engineering evidence report is maintained separately for technical review, including configuration decisions, validation outputs, troubleshooting sequences, security tests, and recovery evidence.

Portfolio-ready recruiter summary. Detailed implementation evidence available on request.