Homelab Infrastructure

The technical documentation for the Flora Family homelab, including UID standards, network layout, and container configurations.

- UID/GID Standards
- Network Architecture
  - LAN Layout & Gateway
  - Reverse Proxy (Caddy)
- Flobot: Transfer to Terra Container
- Terra (Primary Compute)
  - Hardware & OS
  - ZFS Storage Architecture
  - Container Architecture
  - Performance & Stability Optimizations
  - Luna (Backup Server)
- SMB & Time Machine
- AI-Powered Log Aggregation (ClickHouse + Ollama)
- ClickHouse Log Aggregation
- Shell Standards & QoL
- AI Conference Technical Notes (2026-03-24)

UID/GID Standards

UID/GID Standards

The Flora Family homelab uses a standardized UID/GID mapping system to ensure security, prevent ID collisions, and simplify file permission management across the container infrastructure.

□□ Infrastructure Core (System IDs)

These IDs are reserved for the underlying data layers and databases.

- **70:70 (Postgres):** Standard ID for all Postgres and pgvector instances.
- **970:970 (Redis/Valkey):** Standard ID for all Redis and Valkey caching/session instances.

□□ Family & Admin (50,000 Block)

Reserved for human users and administrative bots.

- **50002:60002:** Christopher
- **50003:60003:** Erin
- **50001:60001:** Media (Service account for shared media access)
- **50004:60004:** Flobot (The Robot)

□□ Automated Services (51,000 Block)

Reserved for containerized services and application logic.

UID	GID	Service	Category
51004	61004	Caddy	Core Proxy

UID	GID	Service	Category
51010	61010	Paperless	Organizational
51020	61020	BookStack	Organizational
51052	61052	Immich	Media/Photos
51054	61054	n8n	Automation
51055	61055	OpenWebUI	AI Interface
51100	61100	Minecraft	Gaming
51101	61101	Foundry	Gaming

□□ Security Principles

1. **Non-Root Execution:** No containers (except Watchtower & beszel-agent) run as root.
2. **Config Isolation:** All docker-compose files in `/srv` are owned by `root:root` with 755/644 permissions.
3. **Data Ownership:** All volume mounts in `/main/appdata` are owned by their respective service account.
4. **Least Privilege:** env files are restricted to `600` permissions.

Network Architecture

Details on the physical and logical networking, including the opnSense gateway, internal subnets, and reverse proxy configuration.

Network Architecture

LAN Layout & Gateway

LAN Layout & Gateway

□□ Internet Edge

- **ISP:** AT&T Fiber (1Gbps Up/Down)
- **Gateway/ONT:** HUMAX BGW320-500 (Located in Sunroom)
- **Mode:** Bridge/Pass-through mode enabled (February 2026 upgrade)

□□ Core Routing (Server Closet)

- **Device:** Protectli FW2B (Pluto)
- **OS:** Ubuntu 24.04 LTS (ZFS on Root)
- **Role:** Primary Gateway & Firewall (Replacing opnSense architecture)
- **Internal LAN:** 192.168.13.0/24 (Native), 10.13.x.y (VLANs - Deploying)

Core Services

- **Firewall:** nftables (v2 production Ruleset active)
- **DHCP:** Technitium DHCP (192.168.13.0/24 scope active)
- **DNS:** Technitium DNS (DNS-over-TLS enabled via Cloudflare/Google)
- **Identity:** JumpCloud (Fully configured on Pluto for identity management)
- **Monitoring:** Beszel-agent + Vector (Shipping to ClickHouse)
- **Backup:** Syncoid/Sanoid (OS snapshots and replication to Luna)
- **VPN:** WireGuard (Pluto Gateway + MacBook Client - Deployed March 18, 2026)

□□ Physical Switching & Wireless

- **Switch:** 24-port Gigabit Managed Switch with 12x PoE+ ports.

- **AP:** 2x Grandstream Mesh Access Points (PoE powered).
- **Update (Feb 26, 2026):** Backbone migration to the new Ubuntu-based routing on Pluto is complete. Network is stable on 192.168.13.x.

□□ Security Updates (March 19, 2026)

- **WireGuard:** Tunnel configured for `pluto` and Macbook client; nftables updated for tunnel traffic.
- **Fail2Ban:**
  - Refined `default` and `SSH` jail configs.
  - Added **Recidive** jail (repeat offenders).
  - Added **Portscan** detection jail.
- **Edge Hardening:**
  - **Port Closures:** Removed Minecraft Bedrock (19132) and Bittorrent (52341) forwards.
  - **Rate Limiting:** Enabled `nftables` rate-limiting for Minecraft Java (25565), Voice Chat (24454), and CoTurn (3478).
  - **Geo-blocking:** Implemented automated `nftables` blocklist (CN, KP, RU, IR) via `systemd` timer + lookup sets.

Network Architecture

Reverse Proxy (Caddy)

Reverse Proxy (Caddy)

Caddy serves as the primary ingress point for all web services in the Flora Family homelab. It handles SSL termination (ACME), local CA management, and request routing.

□□ Structure

- **Location:** `/srv/caddy`
- **User:** `51004:61004` (Caddy Service Account)
- **Snippets Path:** `/main/appdata/caddy/files/snippets` (Mapped internally to `/etc/caddy/snippets`)

□□ Log Exporting

- **Mechanism:** UDP/514 forwarding via `rsyslog` on `pluto`.
- **Path:** Raw JSON logs exported to `/var/log/caddy/access.log`.
- **Fail2Ban:** Integrated via local file watcher (jail: `caddy`).

□□ Caddyfile Strategy

The Caddyfile is organized into modular sites located in `sites-enabled/`. This mimics traditional Apache/Nginx structures for clean management.

Common Snippets

1. **`security_headers.caddy`**: Implements HSTS, X-Frame-Options (DENY), and nosniff.
2. **`common_tls_internal.caddy`**: Uses `tls internal` for services only accessible on the LAN, backed by the local Caddy CA.
3. **`common_tls_external.caddy`**: Standard ACME/Let's Encrypt for public-facing services.

Internal CA & Trust

When using `tls internal`, Caddy acts as its own Certificate Authority.

- **Root Cert Location:** `/main/appdata/caddy/data/caddy/pki/authorities/local/root.crt`
- **Permissions:** Access to the `pki` directory requires membership in the **Caddy Group (GID 61004)**.
- **Distribution:** This root certificate is mounted into downstream containers (like SearXNG or OpenWebUI) and added to their system trust stores to allow secure internal communication.

Typical Route Block

```
searxng.flora.family {
    reverse_proxy searxng:8080
    import /etc/caddy/snippets/security_headers.caddy
    import /etc/caddy/snippets/common_tls_internal.caddy
}
```

Flobot: Transfer to Terra Container

Flobot: Transfer to Terra Container

Overview

Flobot (the OpenClaw AI assistant for the Flora family homelab) transitioned from running on Saturn to running as a containerized service on `terra` in March 2026.

Migration Details

- **Previous Host:** Saturn (standalone device)
- **New Host:** `terra` (primary compute server, running Docker)
- **Date:** March 2026
- **Trigger:** Consolidation of services and containerization strategy

Architecture on Terra

- **Runtime:** Docker container
- **Access:** OpenClaw Gateway (accessible via Mac App and web console)
- **Services Accessed:**
  - **ClickHouse** (`logs.flora.family`) - Central log aggregation
  - **BookStack** (`wiki.flora.family`) - Documentation and knowledge base
  - **Technitium** (via `pluto` router) - DNS/DHCP queries

Capabilities

Flobot now has read/write access to critical homelab services via API tokens, enabling:

- Real-time log analysis and anomaly detection
- Proactive system monitoring and alerting
- Documentation generation and updates
- Task management via future project management system

Authentication

- Uses environment variables for credential management
- Maintains local certificate store for internal TLS connections
- API tokens stored securely in container environment

Terra (Primary Compute)

Deep dive into the main workhorse of the homelab, covering hardware specs, ZFS storage architecture, and container standards.

Hardware & OS

Hardware & OS

Terra is the primary compute node for the Flora Family homelab, designed for high-concurrency tasks like AI processing, media transcoding, and ZFS storage management.

□□ Physical Specs

- **Motherboard:** Asus Prime X570-Pro
- **CPU:** AMD Ryzen 9 5950X (16 Cores, 32 Threads)
  - **Scaling Driver:** `amd_pstate=active` (EPP)
  - **Frequency Range:** 550MHz - 5000MHz
- **RAM:** 128GB (4x32GB) Corsair Vengeance LPX DDR4 3200MHz CL16 (Non-ECC)
  - *Note: Currently running at 2133MHz; XMP/DOCP troubleshooting in progress.*
- **GPU:** Nvidia Quadro P2200 (5GB VRAM)
- **Chassis:** iStar D-400 with Red Drive Doors (4U Rackmount)
- **Power Supply:** be quiet! Dark Power Pro 11 650W (80 Plus Platinum)
- **Expansion & Storage Connectivity:
  - **LSI 9207-8i HBA:** Driving the 4x 12TB HDD stack.
  - **Motherboard SATA:** Driving the 6x 2TB SSD stack.
  - **4-port Gigabit NIC:** Realtek chipset; currently unused.

□□ Cooling (Noctua Standards)

- **CPU Cooler:** Noctua NH-C14S (with dual 140mm fans).
- **NVMe Cooling:** Both drives equipped with heatsinks including integrated active fans.
  - *Note: Active cooling reduced temperatures significantly. Current stats (composite): drive 0400 @ 42C, 0100 @ 37C.*
- **Case Fans:** All stock fans replaced with Noctua units.
- **Additional Airflow:** Added 40mm fan on the HBA and 2x 80mm fans at the chassis front.

Fan Connectivity Map

Sensor	Description	Connection
fan1	CHA_FAN1	back exhaust fans (2x80mm)
fan2	CPU_FAN	bottom CPU fan (1x120mm)
fan3	CHA_FAN2	SDD drive cage fan (1x40mm)
fan4	CHA_FAN3	front intake fans (2x80mm)
fan5	?	Shows 0 rpm
fan6	W_PUMP	HBA & NVMe fans (1x40mm + 2x20mm)
fan7	?	Shows 0 rpm
?	CPU_OPT	top CPU fan (1x120mm)
?	AIO_PUMP	empty
-	-	HDD drive cage fan (1x80mm)

Storage Media

- **NVMe:** 2x 500GB WD Black SN7100 (Used for OS mirrors and ZFS Special VDEVs/L2ARC).
- **SSD:** 6x 2TB Samsung 870 EVO (Main pool).
- **HDD:** 4x 12TB Seagate Exos X16 (Bulk pool).

⚙ OS & Software

- **OS:** Ubuntu 24.04 LTS
- **Kernel:** ZFS-on-Linux (OpenZFS)
- **Management:** Sanoid/Syncoid for snapshots and replication.

Terra (Primary Compute)

ZFS Storage Architecture

ZFS Storage Architecture

Terra utilizes ZFS for primary data storage, leveraging its advanced features for data integrity, snapshots, and caching.

□□ Pools

□□ `main` (SSD Pool)

- **Hardware:** Raid-Z2 (6x 2TB SSD)
- **Total Capacity:** ~12TB Raw
- **Use Case:** High-performance data, VM disks, Docker root, and active AppData.
- **L2ARC:** 48GB limit (leveraging NVMe partitions).

□□ `bulk` (HDD Pool)

- **Hardware:** Raid-Z1 (4x 12TB HDD)
- **Total Capacity:** ~48TB Raw
- **Use Case:** Large media storage, backups, and long-term archival.
- **Special VDEV:** Mirrored NVMe partitions used for metadata offloading to speed up file listing on HDDs.

□□ Key Datasets & Mounts

- **/srv**: Docker-compose configuration files (Stored on `main`).
- **/main/appdata**: Primary persistent storage for containers (Stored on `main`).
- **/bulk/pics**: Family photo/video archive (Stored on `bulk`).

Health & Maintenance

Automated Monitoring (`smartd`)

Drive health is monitored via `smartmontools` with automated short and long self-tests configured to email results. Test schedules are staggered to prevent resource contention.

- **NVMe (2x 500GB):** Short tests daily at 1am & 2am; Long tests quarterly on the 1st-2nd.
- **HDD (4x 12TB):** Short tests daily at 3am-6am; Long tests on the 4th, 6th, 8th, and 10th of each month.
- **SSD (6x 2TB):** Short tests daily from 1am-6am; Long tests quarterly on the 3rd, 5th, 7th, 9th, 11th, and 12th.

ZFS Scrubbing

A scrub is performed automatically on the 3rd Sunday of every month to ensure data integrity and prevent bitrot.

Scrub Schedule (Cron): `24 0 15-21 * * root if [ $(date +\%w) -eq 0 ] && [ -x /usr/lib/zfs-linux/scrub ]; then /usr/lib/zfs-linux/scrub; fi`

Container Architecture

Container Architecture

The Flora Family homelab relies on a standardized Docker deployment pattern to ensure security, portability, and ease of management.

□□ Directory Layout

To maintain a clean separation between configuration and state, Terra uses two primary directories:

- **/srv/[service-name]/**: Contains `docker-compose.yaml` and `.env` files.
  - *Ownership:* `root:root` (Modified only via sudo).
  - *Permissions:* folders `755`, configs `644`, .env `600`.
- **/main/appdata/[service-name]/**: Contains persistent application data.
  - *Ownership:* Mapped to the specific service account (UID/GID).
  - *Permissions:* `755` (folders) / `644` (files) generally.

□□ Deployment Standards

□□ Non-Root Execution

Every service is configured to run as a non-privileged user, with the specific exceptions of **Watchtower** and **Beszel Agent** (which require root/host socket access to monitor system health and container status).

Non-root execution is achieved through one of three methods:

1. **Standard `user:` flag**: For images that support it (e.g., `user: "51100:61100"`).
2. **Environment Variables**: Many linuxserver.io images use `PUID`/`PGID` variables.

3. **Custom Dockerfile Builds**: Used for "scratch" images or minimalist images (like OpenWebUI and SearXNG) to manually inject the desired UID/GID and install local CA certificates.

□□ Shared Infrastructure

Common backend services are standardized to simplify inter-container networking and permissions:

- **Postgres:** Standardized on GID `70` across all instances (Immich, n8n, etc.).
- **Redis/Valkey:** Standardized on GID `970` for caching and session management.
- **Shared Identities:** The `node-user` (UID 1000) is shared between several JS-based apps (n8n, Uptime Kuma) where common file access is required.

□□ Lifecycle Management

- **Updates:** Managed by **Watchtower**, configured to run every Sunday at 04:00 AM.
- **Image Tagging:**
  - *Pinned:* Databases and critical infrastructure (e.g., `postgres:18`) to prevent breaking updates.
  - *Floating:* Application layers (e.g., `sonarr:latest`) to receive automated security patches.

Performance & Stability Optimizations

Performance & Stability Optimizations

This page documents the specific kernel and hardware tweaks applied to Terra (Ryzen 5950X / Ubuntu 24.04) to optimize for container performance, storage throughput, and long-term stability.

□□ Stability & Power Management (The "Golden Config")

Typical Current Idle (UEFI)

- **Setting:** Advanced > AMD CBS > CPU Common > Power Supply Idle Control = "Typical Current Idle"
- **Benefit:** Prevents the Ryzen "Sleep of Death" where the PSU cuts power during deep C-state transitions. Essential for 24/7 server stability on X570 boards.

Processor C-State Limit (Kernel)

- **Driver:** `processor.max_cstate=1`
- **Benefit:** Software-side enforcement of the Idle fix. Limits the CPU to C1 (Halt/Idle) instead of deeper sleep states (C6), preventing interrupt processing delays and hardware hangs.
- **Configuration:** Added to `GRUB_CMDLINE_LINUX_DEFAULT` in `/etc/default/grub`.

65W Eco Mode (Manual PBO)

- **Settings:** PPT: 88, TDC: 60, EDC: 90
- **Benefit:** Drastically reduces thermals (peak temps dropped from ~90°C to 58°C) without significant impact on multi-core performance. Ideal for 4U rackmount chassis airflow.
- **Configuration:** UEFI > Precision Boost Overdrive > Manual.

⚡ CPU & Power Performance

AMD P-State (Active Mode)

- **Driver:** `amd_pstate=active` (EPP - Energy Performance Preference)
- **Benefit:** Allows the kernel to communicate directly with the Zen 3 power management controller. Enables a wider frequency range (550MHz - 5000MHz).

IRQ Balancing

- **Service:** `irqbalance` (Installed/Enabled)
- **Benefit:** Distributes hardware interrupts (HBA, NVMe, NIC) across all threads, preventing Core 0 bottlenecks.

□□ Memory Tuning

Speed & Stability

- **Configuration:** Currently hard-coded to **2133MHz** at **AUTO** voltage.
- **Note:** Troubleshooting in Feb 2026 confirmed that 3000MHz at 1.35V caused instantaneous reboots. Stability requires sticking to JEDEC defaults for now.

Static Hugepages

- **Setting:** `hugepages=2048` (Reserving 4GB total as 2MB pages)
- **Benefit:** Reduces TLB misses for memory-heavy apps like Minecraft and Postgres.

□□ I/O & Network Tuning (sysctl)

ZFS Write Buffering

- **Settings:** `vm.dirty_background_bytes = 268435456`, `vm.dirty_bytes = 1073741824`
- **Benefit:** Prevents the kernel from buffering too much data in RAM before flushing to ZFS, eliminating system-wide UI/process stutter.

Fluent-bit Batching (ClickHouse)

- **Setting:** `Flush 60` in `fluent-bit.conf`
- **Benefit:** Reduced ClickHouse CPU usage from 80% spikes to <5% by reducing transaction frequency and optimizing ZFS write contiguousness.

□□ GPU Optimization (Quadro P2200)

Persistence Mode

- **Command:** `sudo nvidia-smi -pm 1`
- **Benefit:** Eliminates the 2-second initialization delay for AI inference and transcoding containers.

Luna (Backup Server)

Luna (Backup Server)

Luna is the secondary compute and backup node for the Flora Family homelab. It serves as the primary replication target for Terra.

□□ System Status

- **Current Status: ONLINE** (Recovered 2026-02-25)
- **Role:** ZFS Replication Target, Thermal Monitoring
- **Observability:** Fully integrated into the Vector/ClickHouse logging pipeline (Feb 2026).

□□ Storage Architecture (ZFS)

- **Pool Configuration:** Striped Raid-Z1 vdevs
- **Capacity:**
  - 4x 4TB Drives
  - 4x 3TB Drives

□□ Monitoring

- **Integrated Script:** `/usr/local/bin/cpu_temp.sh` (Runs via CRON every 5 minutes)
- **Reporting:** Logs to ClickHouse via Vector.
- **Average Temps:** ~46°C - 48°C (Idle/Light load)

⚙ Software & Management

- **Management:** Syncoid target for Terra's datasets.

- **Logging:** Migrated from Fluent-Bit to Vector (Feb 25, 2026).
- **Sanoid:** Configured with `--quiet` to reduce log spam from snapshot cleanup.
- **Timezone:** Correctly set to `America/New_York (EST)`. System clock and Vector timestamps are aligned.

□□ Recovery Notes (Feb 2026)

- Luna was brought back online after an outage.
- Vector configuration synchronized via Ansible with Terra.
- Verified successful `syncoid` backup completions from Terra at 07:44 AM EST (Feb 25).

SMB & Time Machine

SMB & Time Machine

Services

Our network file sharing (SMB) is central to the Flora Family homelab, providing shared access for group projects, personal file storage, and macOS backups via Time Machine. This configuration is managed at `/etc/samba/smb.conf`.

Global Configuration

- **Workgroup:** `FloraFamily`
- **Networking:** Bound strictly to the private subnet `192.168.13.0/24` for security.
- **Authentication:** Integrated with JumpCloud LDAP for centralized user and group management.
- **Protocol:** Restricted to SMBv2 and SMBv3 for security and performance.

macOS Optimization (vfs_fruit)

Sharing files with Macs is optimized using the `fruit` VFS module. This allows Samba to behave like a native Mac server, supporting resource forks, extended attributes, and Finder specific icons (modeled as `MacSamba`).

Available Shares

General Purpose Shares

- **int13:** Shared group storage for standard projects. Inherits permissions (0660/2770) for the `int13` group.

- **flora:** Main family share for shared documents and media. Restricted to the `flora` group.

Personal Shares

- **christopher & erin:** Private personal folders mapped to individual home directories. Restricted to their respective owners.

Time Machine

- **timemachine:** A dedicated macOS backup target located at `/main/shares/timemachine`. It uses the `fruit:time machine = yes` flag to identify itself to macOS as a valid backup disk. Restricted to members of the `timemachine` group.

Configuring macOS for Time Machine

To use the `timemachine` share as a backup target for a Mac, follow these steps:

1. Connect to the Share

1. Open **Finder**.
2. In the menu bar, select **Go > Connect to Server...** (or press `Cmd+K`).
3. Enter the server address: `smb://terra.flora.family/timemachine`
4. Click **Connect** and authenticate using your JumpCloud credentials.

2. Select the Backup Disk

1. Open **System Settings** (or System Preferences) on your Mac.
2. Navigate to **General > Time Machine**.
3. Click **Add Backup Disk...** (or **Select Disk...**).
4. Select the `timemachine` network share from the list.
5. When prompted, choose to use your existing JumpCloud credentials to allow Time Machine to connect automatically in the future.

Tip: Initial backups can be large. It is recommended to perform the first backup while connected via Ethernet.

Source Reference: The primary configuration for these services can be found on the server at `/etc/samba/smb.conf`.

AI-Powered Log Aggregation (ClickHouse + Ollama)

AI-Powered Log Aggregation with ClickHouse & Ollama

This page documents the minimalist log aggregation and AI search pipeline built for the Flora family homelab (terra + opnsense).

Architecture

- **Aggregator**: ClickHouse (bare-metal on terra)
- **Log Shipper**: Fluent Bit (bare-metal on terra)
- **AI Brain**: Ollama (Docker on terra) running `nomic-embed-text`
- **Deduplication**: `ReplacingMergeTree` engine in ClickHouse using `log_id` (UUID)

ClickHouse Table Schema

```
CREATE TABLE system_logs (
    timestamp DateTime64(3),
    host String,
    unit String,
    message String,
    priority Int8,
    log_id String,
    embedding Array(Float32),
    INDEX ann_idx embedding TYPE vector_similarity('hnsw', 'cosineDistance', 768) GRANULARITY 1
) ENGINE = ReplacingMergeTree()
ORDER BY (log_id);
```

Components

1. Fluent Bit (Ingestion)

Fluent Bit reads the systemd journal and ships logs to ClickHouse via HTTP. Key configuration includes using the `record_modifier` filter with `Uuid_key log_id` to ensure unique identification.

2. Log Brain (Embedding Loop)

A Python background service (`log-brain.service`) runs an embedding loop:

1. Queries logs where `length(embedding) = 0` using `FINAL` to handle merging.
2. Sends the message to Ollama for embedding.
3. Upserts the record back to ClickHouse with the vector.

3. Reverse Proxy

Caddy provides internal TLS for both Ollama (`ollama.flora.family`) and ClickHouse (`logs.flora.family`).

Search Examples

I (Flobot) can now search logs using standard SQL for keywords or `cosineDistance` for semantic similarity.

ClickHouse Log Aggregation

ClickHouse Log Aggregation & Investigation

Architecture Overview

Log Pipeline (Migrated to Vector - Feb 2026):

- **Terra (server):** Journald + Docker containers → Vector
- **Luna (backup server):** Journald → Vector
- **Pluto (gateway):** Journald + nftables (firewall) → Vector
- **Vector Pipeline:** Unified Jinja2 Ansible template. Converts all logs to **America/New_York (EST)** before storage.
- **All sources** → ClickHouse (system_logs table with embeddings)

ClickHouse Access

- **URL:** https://logs.flora.family
- **User:** flobot
- **Database:** default
- **Table:** system_logs

Table Schema

```
CREATE TABLE default.system_logs (
  timestamp DateTime64(3),
  host String,
  unit String,
```

```
  message String,
  priority Int8,
  log_id String,
  embedding Array(Float32),  -- 768-dim vectors
  source String DEFAULT 'systemd'
)
ENGINE = ReplacingMergeTree
ORDER BY log_id
```

Log Sources

Vector Migration (Feb 2026)

- **Retired Fluent-Bit:** Entire lab moved to Vector for better performance and easier Ansible management.
- **EST Alignment:** All timestamps across the pipeline are now surgically aligned to Eastern Time.
- **Container Naming:** Docker logs now show friendly names (e.g., `ollama`, `bazarr`) instead of raw hex IDs.

Noise Management & Filtering

- **Ollama/GIN:** Sampled 1:100 to prevent database wear.
- **Sanoid:** Running in `--quiet` mode on Terra and Luna.
- **Recursive Loops:** Patched VRL filters to prevent `log-brain` from logging its own status updates to ClickHouse.
- **Context Truncation:** `embeddings_worker.py` now truncates logs at 3,072 characters to stay within Ollama context windows.

Embedding Pipeline

Performance (Feb 2026):

- **Coverage:** Reached **100% vectorization coverage** for all essential services.
- **Model:** nomic-embed-text (768-dimensional)
- **GPU:** Quadro P2200 for processing.

Key Learnings

1. **Timezone consistency is critical:** Migrating from UTC to local time in the pipeline saves massive mental overhead during investigations.
2. **Priority Intelligence:** Mapping systemd priorities (0-7) into the database allows for instant "Error-only" dashboards.
3. **Recursive logging is a DDoS:** Always filter your log-aggregator's own logs out of the stream.
4. **ReplacingMergeTree quirk:** Must insert complete rows (all columns) for deduplication to work.

Future Work

- ☑ Add luna backup server logs
- ☑ Align all hosts to EST
- ☐ Parse firewall log CSV columns for structured queries (Pluto implementation)
- ☐ Real-time anomaly detection via vector search
- ☐ Log retention policies & archival

Shell Standards & QoL

Shell Standards & QoL

This page documents the standardized shell environment applied across the Flora Family infrastructure via Ansible.

□□ Ansible Management

- **Playbook:** `~/code/ansible-homelab/playbooks/base_setup.yml` on `saturn`.
- **Scope:** Applied to `terra`, `pluto`, and `luna`.

□□ ZSH Configuration

- **Theme:** Powerlevel10k
- **Plugins:** git, docker, curl, zoxide
- **Specific Tweaks:**
  - `bat` is aliased to `batcat` (Ubuntu standard).
  - `sudo` is aliased as `sudo ` (trailing space enables alias expansion for following commands).

□□ Dynamic Theme Detection

Standardized the `term_theme_detect` function in `linux_specific.zsh`.

- **Method:** Uses OSC 11 ANSI query to detect background color.
- **Trick:** Uses `stty raw` to ensure reliable reading of the escape sequence response over SSH.
- **Themes:** Switches between `Catppuccin Latte` (Light) and `Catppuccin Frappe` (Dark) automatically.

□□ Docker Shortcuts

- `dcd`: docker compose down
- `dcu`: docker compose up -d
- `dcl`: docker compose logs -f
- `dcr`: Custom function to cycle a service (down -> up -> logs).

□□ Terminal Standards

- **Ghostty:** High-performance terminal used on client devices.
- **Title Format:** `user@host:current_working_directory` (managed via ZSH precmd).

AI Conference Technical Notes (2026-03-24)

AI Conference Technical Notes (2026-03-24)

Architecture Overview

Deep-dive notes from the 2026 All Things AI conference sessions centered on robust, production-grade agentic systems.

1. Pydantic AI & Logfire (Observability Stack)

Pydantic AI is the standard for type-safe agentic architectures. Integration with Logfire creates **Context Graphs**, turning logging into a primary observability tool. This allows for clear visualization of agent decision-making traces.

2. Three-Tiered Memory Architecture

A move away from LLM-centric memory management towards a structured, verifiable storage system:

- **Tier 1 (Profile Summary):** High-level user context. **Rule:** LLM access forbidden.
- **Tier 2 (Facts Table):** Individual, extracted data points. Verification requires a deterministic, non-AI script to validate against raw source transcripts.
- **Tier 3 (Graph Store):** Relational data linking facts.

3. The "Smart Gateway" Philosophy

- **Rule:** The LLM should only intervene when human-level reasoning is strictly required.
- **Application:** The gateway handles critical routing, security, and tool invocation using deterministic code. The LLM acts as a service to the gateway, never the primary driver of infrastructure.

4. Knowledge & Context Graphs

- **Vector DBs:** Best for semantic similarity search.
- **Knowledge Graphs (e.g., Neo4j):** Essential for logical relationships and grounding LLM responses, avoiding "hallucination by confidence."

5. Prototyping

Marimo identified as a preferred "reactive" alternative to traditional cell-based Jupyter flows for executable rapid prototyping.

6. Hypothetical Document Embedding (HyDE)

A high-accuracy retrieval pattern: instead of embedding the user's question, the system generates a "fake answer" via the LLM, embeds that, and searches the document store using "answer-to-answer" matching.

7. The Three Laws of Agentic AI

Reframing security as "Rules of Conduct" for autonomous agents:

- **Law #1 (Identity):** "An agent must always maintain a unique, verifiable identity & must never act anonymously or under a masked provenance."
- **Law #2 (Delegated Permissions/Scope):** "An agent must only perform actions within its explicitly delegated permissions, except where such actions would violate the Law of Identity."
- **Law #3 (Credential Integrity):** "An agent must protect its own credentials and session tokens, as long as such protection does not conflict with the 1st or 2nd laws." This incorporates CIBA (Client Initiated Backchannel Authentication) and human-in-the-loop workflows for high-risk actions.

Implementation Strategy (Homelab):

- **Law #1 (Provenance):** Log client_id and session_id tracking in ClickHouse; enforce strictly at the API layer.
- **Law #2 (Permissions):** Utilize OIDC Claims within token scopes (e.g., manage_vlan, read_docker_logs). Deny any action not explicitly claimed.
- **Law #3 (Integrity):** Implement "Out of Band" (OOB) validation via Matrix/push-notifications for high-risk commands (e.g., database deletion or critical system configuration).