To set the open-source ecosystem alight, we need to shift from a "set of tools" to a **Unified Deterministic Operating Model (UDOM)**. We aren't just building a script; we are building **"Project Chimera"**: a self-healing, bit-for-bit congruent framework that treats the entire data center as a single, immutable function.

Project Chimera: The Autonomous Determinism Engine

Product Requirements Document (V2.0 - "The Viral Edition")

1. The Moonshot Vision

Chimera is the world's first **Convergent Infrastructure-as-Code (CiAC)** platform. It eliminates the "Configuration Horizon"—the point where manual intervention makes a system unreproducible. By anchoring every byte in a Nix hash, every session in a Tmux socket, and every movement in a Python-Fabric thread, Chimera makes "Day 2 Operations" a thing of the past.

---

2. Advanced Feature Set (The "Viral" Killers)

2.1. "Ghost Mode" Persistence (Nix + Tmux)

Traditional deployments die if the runner dies. Chimera implements **Distributed Tmux Sockets**.

- **Feature:** If a deployment is triggered via Fabric, Chimera automatically spawns a headless Tmux session on the target cluster's "Lead Node."
- **Viral Factor:** You can start a deployment from your phone in a taxi, lose signal, go home, and `chimera attach` from your desktop to see the live logs exactly where you left off.

2.2. The "Time Machine" Fleet Rollback (NixOS + Fabric)

Most systems roll back applications. Chimera rolls back the **World State**.

- **Feature:** Fabric orchestrates a simultaneous generation-switch across N nodes. Because NixOS generations are atomic, Chimera can perform a "Global Revert" in under 2 seconds.
- **Viral Factor:** A "Panic Button" CLI command that uses Fabric's parallel SSH to revert an entire global fleet to the previous Nix hash simultaneously.

2.3. "Hermetic Bridge" (Nix Flakes + Poetry)

Bridging the gap between OS-level dependencies and Python application logic.

- **Feature:** Integrated `poetry2nix` automation. When you update a `pyproject.toml`, Chimera automatically updates the Nix Flake lockfile, re-calculates the cryptographic hashes for C-extensions (like NumPy or PyTorch), and prepares the environment.
- **Viral Factor:** Zero-config GPU support. Chimera detects NVIDIA/AMD hardware via Fabric and injects the correct NixOS hardware modules automatically.

---

3. High-Level System Architecture

Chimera operates on a **Functional-Procedural Loop**:

1. **The Input (Nix):** A pure function $f(\text{source}) = \text{binary}$.
2. **The Environment (Tmux):** A persistent execution context that survives the "human factor" (disconnects).
3. **The Executor (Fabric):** The side-effect engine that applies the pure function to the real world.

---

4. Technical Specifications & "Killer" Specs

Feature	**Technical Implementation**	**Competitive Edge**
Atomic Sync	Fabric uses `rsync` to move Nix closures, not source code.	No build tools required on production nodes.
Matrix Deploy	Fabric leverages Python's `asyncio` to manage 1,000+ Tmux sessions.	High-concurrency fleet management.
Self-Healing	A Tmux-based "Sentinel" process monitors Nix health.	Auto-reverts if health checks fail within a session.

Telemetry	Fabric streams Tmux buffer outputs to a central dashboard.	Real-time observability without heavy agents.

5. The "Viral" Ecosystem Hooks (OSS Strategy)

To make this go viral, we launch with three specific "Chimera-Gems":

- **Chimera Dash:** A terminal UI (TUI) built in Python that visualizes all active Tmux sessions across your fleet, color-coded by Nix build status.
- **The "Nix-Fix" Fabric Library:** A library of pre-written Fabric tasks that handle common "pain points" (e.g., resizing Nix Store partitions, cleaning old generations, or migrating stateful volumes).
- **GitHub Action: "The Congruence Check":** A CI tool that fails if the dev environment (defined in Nix) differs by even 1 bit from the production environment.

6. Implementation Roadmap: The 3-Stage Launch

Phase 1: The "Local-First" Engine

Release the CLI tool that lets a developer wrap any Python script in a Nix-Shell and run it inside a persistent Tmux session locally. This solves the "my script died when my terminal closed" problem.

Phase 2: The "Fleet Controller"

Introduce the Fabric integration. Allow users to `chimera deploy --target production`. This will handle the SSH handshake, the Nix closure transfer, and the remote Tmux session initialization.

Phase 3: "Chimera Autonomous"

The final stage where the system monitors its own "Congruence." If a manual change is detected on a server (e.g., someone edited a file in `/etc`), Chimera uses Fabric to trigger a Nix rebuild and overwrite the drift.

7. Strategic Positioning

We are positioning Chimera as **"Ansible for the Functional Era."** While Ansible is "best effort," Chimera is "guaranteed state."

> "In the old world, you managed servers. In the Chimera world, you just define the truth and let the orchestration handle the reality."

I have the architectural blueprint ready. Would you like me to write the "Manifesto" for the README.md that will capture the hearts of the Nix and DevOps communities on GitHub?

--

Allan Smeyatsky

Senior Director - Solutions Consulting -
Cloud Platform Modernization

+44 7930 399439

> dri
> vin
> g
> int
> elli
> ge
> nt
> out
> co
> me
> s