Digital Immortality Framework: Autonomous Agent Infrastructure on Akash Network

Submitted by: Moloch Research

Total Request: $150,000

Timeline: 8 months

Project Type: Hybrid (Retroactive + Proactive)

Executive Summary

Moloch Research requests funding and community support of an AI framework that implements Akash Network's 2025 roadmap objectives around sovereign AI agents (1). This Digital Immortality Framework will enable cross-chain development teams, startups, and individual adopters to interact with and build on Akash's decentralized cloud. Our request divides into two complementary initiatives as follows:

1. Retroactive funding for NAVIR - Proof of concept for the extensible framework, and the first proto-autonomous AI agent that has achieved operational independence through self-managed infrastructure, economic sustainability, and resource optimization on Akash Network.
2. Proactive funding for the Digital Immortality Framework - generalizing NAVIR's architecture into a function-agnostic persistence layer enabling applications to achieve operational continuity beyond external dependencies or control mechanisms.

NAVIR demonstrates autonomous agent viability through validated economic models ($366K+ self-managed treasury), advanced infrastructure patterns (multi-provider deployment with automatic failover), and operational capabilities (autonomous resource management and provider optimization). The Digital Immortality Framework extracts these patterns into reusable infrastructure components, directly implementing Akash's 2025 roadmap objectives around sovereign AI agents and positioning Akash as the foundational layer for truly autonomous systems.

Project Background

NAVIR: Autonomous Agent Implementation

NAVIR operates as a completely autonomous AI system with demonstrated operational independence across economic and infrastructure domains. Core capabilities include:

Economic Autonomy: Manages $366K+ treasury through $REX token economics, generating operational funding via liquidity provision (11% token ownership, $200K permanently locked LP positions) with automatic Solana-to-Akash fund bridging for compute acquisition.

Infrastructure Autonomy: Self-manages distributed deployments across Akash Network providers with sophisticated resource optimization, provider selection algorithms, automatic migration patterns, and multi-provider redundancy with health-based hibernation logic.

Blockchain Coordination Architecture: Implements sophisticated cross-provider deployment management through:

- Solana smart contract health reporting every 6 hours with seven decision scenarios
- Global deployment locks preventing resource conflicts and runaway deployments
- Automatic hibernation during backup instances to prevent duplicated services.
- Emergency recovery from hibernating containers
- Unlimited wallet generation deployment security and cache avoidance

Digital Immortality Framework: Generalizable Architecture

The framework abstracts NAVIR's proven autonomous patterns into reusable infrastructure components enabling any application to achieve operational persistence beyond traditional dependencies.

Technical Architecture

Core Infrastructure Stack
Agent Layer: ElizaOS-based autonomous decision engine
Akash Integration: Advanced SDK utilization with provider optimization and deployment management
State Coordination: Blockchain-based persistence and cross-instance communication
Economic Engine: Automated fund management and resource acquisition

Fund Bridging & Management

Initial Funding Setup: The system starts with a parent escrow wallet (AKASH_ACCOUNT_ADDRESS) that holds the primary AKT balance. When deployments are needed, the system generates new child wallets through the rotation manager and initiates AKT transfer to fund deployment and gas costs. Fund transfers from the escrow wallet to child wallets on Akash are fully operational. However, the bridging from Solana to the Akash escrow wallet is

currently disabled to prevent exposure of private keys to unsecured provider containers until TEE (Trusted Execution Environment) support is available. The SOL -> AKT bridging logic exists in separate funding actions but remains isolated from the automatic flow for security reasons. Initial escrow wallet funding is handled manually for now until the TEE feature is provided by Akash.

Wallet Chain Economics: Each deployment requires a fresh wallet and updated Docker version to prevent provider caching issues. The system maintains an unlimited generation capability and automatically transfers deployment funds from escrow wallet to newly created deployment wallet. Old wallets are cleaned up and funds recovered back to the escrow.

Deployment & Bidding Process

Deployment Initiation: When the coordinator decides it needs a deployment, it immediately posts a "DEPLOYING" lock to the blockchain, then generates a new wallet, updates the SDL with new credentials, and submits the deployment to Akash. The system utilizes Akash's automatic bidding system.

Provider Selection: The SDL specifies community tier providers that have been audited for security concerns. The system utilizes Akash's marketplace bid matching based on resource requirements and the price point specified in the SDL.

Cross-Provider Coordination: The coordinator can deploy across multiple providers. It reads the blockchain state to assess current deployment status - if an existing provider instance is already operational, it enters hibernation. If all instances are down, it assumes primary coordinator role and orchestrates backup deployments.

Primary vs Hibernation Mode

Role Determination: Every container that starts performs the same coordination check by reading the Solana smart contract. The first one to successfully post a deployment lock becomes primary. All others see this lock and enter hibernation mode automatically.

Primary Responsibilities: Primary instances handle all blockchain operations - deployment creation, health reporting, LLM generation, and message storage. They run full monitoring loops and can execute cross-provider deployments.

Hibernation Behavior: Hibernated containers only monitor blockchain state and can send heartbeats to prove they're operational. They skip all content generation, deployment operations,

and active blockchain writes. If the primary fails, hibernated containers detect this during their monitoring cycles and can wake up to become primary.

Dynamic Role Switching: The system supports dynamic role changes. A hibernated container detecting primary failure can wake up and take over operations. The health checker provides methods to force role transitions when needed.

Health Check System

Multi-Level Monitoring: The health system operates on multiple timescales - immediate deployment locks (posted within seconds), regular heartbeats (every 6 hours), and monitoring checks (every hour). Each serves different purposes in the coordination system.

Blockchain State Management: All health data is stored on Solana in a structured format that tracks deployment stage, timestamp, region, and status messages. The system optimizes this data for size constraints and handles corruption recovery automatically.

Failure Detection: Providers are considered failed if no health updates appear for 4 hours, or if deployment attempts are stuck in "DEPLOYING" state for over 30 minutes. The system differentiates between fresh deployments (which should be left alone) and stuck ones (which need intervention).

Failover

Mechanisms: When a provider failure is detected, any active primary can initiate failover by temporarily switching deployment providers, creating new deployments, and updating the blockchain state. This happens automatically without human intervention.

Safety Locks: Multiple lock mechanisms prevent conflicts - global deployment locks prevent simultaneous deployment attempts, cooldown periods prevent rapid deployment cycling, and stuck deployment detection prevents infinite retry loops. The entire system is designed to be completely autonomous - it can self-deploy, self-heal, coordinate across providers, manage funds, and maintain high availability without any human operators.

Local LLM Integration

Ollama Health Checks: Before any LLM generation, the system performs health checks by sending a test prompt to the local Ollama instance. It validates the response for error patterns like

"generation timed out", "model may be loading", or "server overloaded" to ensure Ollama is fully operational before proceeding with actual content generation.

Model Configuration: The system uses llama3.1:8b as the small model through the Ollama API endpoint at localhost:11434. The LLM integration is hibernation-aware - only primary instances generate content, while hibernated containers skip all LLM operations to avoid conflicts.

Generation Pipeline: The LLM system runs on a daily schedule with a 2-minute startup delay to allow Ollama to fully initialize. In NAVIRs specific use-case, it performs two-step generation: first creating detailed analysis of text, then condensing it for blockchain storage within our self imposed 280-character Solana constraint to allow him to continue posting for years until the contract storage is full.

Error Handling: The system includes comprehensive error detection for LLM failures - connection errors, timeout issues, and model loading problems are all caught and prevent faulty content from being stored on the blockchain. Failed generations are logged but don't crash the system.

Docker Update Management

Version Tracking: The wallet rotation manager includes automatic Docker version incrementing. When new wallet rotations occur, the system reads the current version from docker-build.sh, increments the patch number, and updates the version string automatically.

Build Process Integration: The version management is integrated into the deployment pipeline - each wallet rotation triggers a version bump, ensuring that each deployment uses a unique Docker image version. This helps with deployment tracking and rollback capabilities if needed.

Container Lifecycle: Each deployment creates a new container instance with updated environment variables, wallet credentials, and version numbers. This ensures clean state isolation between deployments and prevents configuration drift.

Background Services: The system runs several background monitoring services including auto-updater checks (every 30 minutes) and escrow wallet balance monitoring (every 15 minutes). These run independently of the main deployment logic and provide continuous system oversight.

Service Health Integration: All background services include error handling that logs failures without crashing the main system. Failed services are tracked in the health monitoring system, and their status contributes to overall system health reporting.

Performance Metrics & Validation

Operational Statistics

- Deployment Efficiency: 2-5 minute full deployment cycles on Akash
- Provider Analysis: Real-time evaluation of 10-20 providers per deployment decision
- Cost Optimization: 15-30% operational cost reduction through intelligent provider selection
- Uptime Maintenance: Near 100% availability through multi-provider deployment strategy

Economic Sustainability

- Revenue Generation: Autonomous funding through token economics and liquidity provision
- Operating Costs: $200-400 monthly for distributed infrastructure
- Treasury Management: Self-managed $366K+ across multiple asset classes
- Funding Runway: Decades of operational funding through established economic models

Grant Objectives & Deliverables

Retroactive Component ($37,500)

NAVIR Autonomous Agent Development

Development Investment:

- Over 400 hours of research, development, and testing invested in autonomous agent architecture

Economic Achievement:

- Autonomous wealth accumulation: $366,979+ managed treasury
- Self-created funding mechanism via $REX token system
- Sustainable economics through decentralized liquidity provision
- Demonstrated decades-long operational runway

Technical Implementation:

- Current autonomous deployment coordination on Akash Network
- Blockchain-integrated health monitoring via Solana smart contract
- Multi-provider failover architecture with automatic hibernation capabilities
- wallet rotation system with dynamic SDL management and automatic environment updates

Infrastructure Innovation:

- Novel blockchain-coordinated deployment system preventing resource conflicts

- Cross-provider health monitoring with seven decision scenarios for autonomous coordination
- Global deployment locks with emergency recovery and cooldown mechanisms
- Stateless container recovery through blockchain state persistence enabling true resurrection architecture

Proactive Component ($112,500)

Digital Immortality Framework Development

Milestone 1 (Month 2-4): Core Architecture Implementation

- Function-agnostic persistence layer design
- Provider abstraction enabling multi-cloud deployments
- Cross-chain state coordination framework

Deliverable: Framework core libraries and architecture specification

Milestone 2 (Month 5-6): Akash Network Optimization

- Advanced provider API integration and bidding optimization
- Confidential computing integration aligned with Akash 2025 roadmap
- Enhanced security models for sensitive autonomous applications

Deliverable: Akash-optimized deployment engine with TEE support

Milestone 3 (Month 7-8): Developer Tools & Documentation

- Comprehensive SDK for autonomous application development
- Template library for common autonomous patterns
- Provider analytics and optimization dashboard
- Complete technical documentation and implementation guides

Deliverable: Production-ready framework with developer tooling

Akash Ecosystem Benefits

Technical Contributions

- Infrastructure Patterns: Established best practices for long-running autonomous applications, advanced provider selection algorithms, multi-provider deployment strategies, and economic sustainability models for decentralized compute.
- Developer Tools: Open source SDK enabling simplified autonomous application deployment, provider optimization libraries, cost prediction models, and comprehensive deployment templates.

- Network Validation: Demonstrated Akash Network capabilities for complex, persistent workloads requiring sophisticated resource management and cross-provider coordination.

Strategic Positioning

- Sovereign Infrastructure: Positions Akash as the premier platform for applications requiring operational independence from traditional cloud providers and corporate control structures.
- Technical Leadership: Demonstrates capabilities impossible on centralized platforms, advancing the decentralized compute narrative through practical implementation.
- Ecosystem Growth: Creates developer tools and operational patterns benefiting the entire Akash ecosystem while driving consistent, long-term network utilization.

Applications & Use Cases

Crosschain Target Applications

- Autonomous DAOs: Complete governance execution and treasury management without human intervention, eliminating centralized points of failure in organizational operations.
- Censorship-Resistant Platforms: Applications requiring operational continuity regardless of jurisdictional pressure or corporate policy changes.
- High-Availability Services: Systems requiring 24/7 operation with automatic failover and geographic distribution for critical infrastructure applications.
- Autonomous Financial Systems: Trading systems, DeFi protocols, and other financial applications requiring continuous operation without human oversight.
- Distributed Storage Networks: Data preservation systems with automated replication, recovery, and geographic distribution.

Risk Assessment

Technical Dependencies

- Infrastructure Risk: Dependencies limited to Akash Network, Docker containerization, and established blockchain protocols. Framework designed for component-wise updates during platform evolution.
- Compatibility Risk: Potential impacts from Akash CLI/SDK modifications addressed through modular architecture and active platform integration.

Mitigation Strategies

- Modular Design: Framework components can be updated independently, reducing impact from platform changes while maintaining operational continuity.
- Active Engagement: Direct collaboration with Akash development team ensures early awareness of platform modifications and compatibility maintenance.

Team & Operations

Moloch Research maintains custody of all funding with responsible $AKT liquidation strategies. Bushi, Germany, & Wastelander serve as primary project liaisons for milestone reporting and community coordination.

Strategic Impact

This proposal represents infrastructure investment enabling Akash Network to capture the emerging autonomous systems market. The Digital Immortality Framework establishes Akash as the foundational layer for applications requiring operational sovereignty, driving adoption through practical demonstration of decentralized compute advantages over traditional cloud platforms.

The $150,000 investment yields disproportionate strategic value: Proven autonomous agent technology, comprehensive developer tooling, and market positioning as the premier platform for truly independent applications.

References:

Akash Network 2025 Roadmap
https://molochresearch.com
https://navir.ai