SEVEN SWELL AI LABS



RESEARCH WHITE PAPER

Janus Sandbox

Isolated RAM Sandbox

Volatile RAM Substrate & Path-Rewriting Harness.

Isolating multi-step agent mutations in sub-microsecond RAM mounts before merging validated state to disk.

Field	Value
Document	Architecture Note 02-A
Author	Thaddeus Sylvester Lee
Date	November 2025 (Updated 2026)
Classification	Production Lab Tooling

Seven Swell AI Labs
https://sevenswell.ai/#janus

Document Profile

Field	Value
Document	Architecture Note 02-A
Headline	Volatile RAM Substrate & Path-Rewriting Harness.
Lead	Isolating multi-step agent mutations in sub-microsecond RAM mounts before merging validated state to disk.
Author	Thaddeus Sylvester Lee
Company	Seven Swell AI Labs
Date	November 2025 (Updated 2026)
Classification	Production Lab Tooling

Abstract

Autonomous coding agents degrade when every experimental edit mutates the primary repository on SSD storage. Janus mounts part of a 35GB RAM pool as a volatile execution substrate with dynamic path-rewriting and an SSE heartbeat reaper, allowing agents to test schema or code mutations in memory and merge only verified changes back to disk.

01 / The Problem

Reference Signals

Signal	Value
SSD I/O Latency	~500 us
RAM Access Latency	< 1 us

Mutable Working-Tree Risk

When coding agents run iterative builds and schema refactors directly against the primary working tree, failed edits leave half-written files behind and incur persistent SSD I/O latency.

02 / Architecture

RAM Substrate

Janus uses the Advanced Interface for Memory (AIM) to mount a slice of the host 35GB RAM pool as a volatile virtual drive dedicated to isolated sandbox runs.

Path-Rewriting Adapter

An interception layer dynamically remaps file system paths from the primary repository to the RAM-disk mount so agents run normal build and test tools without touching disk state.

SSE Heartbeat Reaper

A lifecycle monitor tied to the active Server-Sent Events heartbeat prunes volatile sandbox allocations automatically when an agent session terminates or drops.

03 / Execution Loop

01 / Sandbox Allocation

Mount the current repository state into an isolated volatile RAM volume.

02 / Parallel Mutation Testing

Run competing schema or code edits across isolated mounts at RAM speed.

03 / State Verification

Validate outputs against database assertions and unit or build checks.

04 / Verified Merge

Promote only validated changes back to the primary working directory and release the mount.

04 / Operational Workflows

Sandbox Mount

Field	Value
Subtitle	Volatile RAM Sandbox
Overview	Isolating repository state onto a dedicated RAM-disk volume.
Objective	Initialize an isolated RAM-backed working tree for non-destructive agent testing.

Phase 01 / Initialization

1. RAM Volume Allocation

Mount a named RAM-disk partition to isolate candidate edits from the Primary Working Tree.

Tools: RAM Disk Controller

2. Path Authorization

Configure the path-rewriting adapter to authorize atomic file operations only inside the mounted volume.

Tools: Path Rewriter

3. Mount Verification

Verify directory readiness and confirm low-latency read/write access before agent execution.

Tools: Health Check

Phase 02 / Execution

1. Workspace Redirection

Route agent tool calls to the RAM sandbox path while preserving relative project structure.

Tools: Path Rewriter

2. In-Memory Validation

Run build and test suites inside RAM without mutating files on persistent storage.

Parallel Candidates

Field	Value
Subtitle	Concurrent Sandboxes
Overview	Evaluating competing implementation patches across isolated mounts.
Objective	Run parallel sandbox instances to compare candidate implementations against deterministic checks before merging.

Phase 01 / Branching

1. Sandbox Replication

Initialize separate RAM mounts for each candidate approach under test.

Tools: RAM Disk Controller

2. Isolated Patching

Apply divergent schema or application edits independently in each sandbox.

Tools: Diff Engine

Phase 02 / Selection

1. State Verification

Execute database assertions and test suites across every active sandbox.

Tools: Validation Suite

2. Validated State Merge

Promote only the verified patch back into the Primary Working Tree.

Tools: Merge Gate

Guarded Feature Build

Field	Value
Subtitle	Atomic Integration
Overview	Developing and verifying mutations before disk commit.
Objective	Keep the primary repository stable while agents iterate on schema and code mutations in memory.

Phase 01 / Isolation

1. Sandbox Allocation

Provision a dedicated volatile workspace for the current task.

Tools: RAM Disk Controller

2. Scope Verification

Check proposed edits against schema contracts and state constraints.

Tools: Schema Validator

Phase 02 / Commit Gate

1. Targeted Edits

Apply scoped modifications inside the sandbox without touching unrelated files.

Tools: Diff Engine

2. Validated State Merge

Flush verified changes back to disk and release volatile memory.

Tools: Merge Gate

Cross-Module Audit

Field	Value
Subtitle	Dependency Mapping
Overview	Tracing schema and API boundaries across repositories.
Objective	Inspect relational dependencies and call chains across modules before executing multi-file refactors.

Phase 01 / Ingestion

1. Dependency Scoping

Index root tables, stored procedures, and API entry points.

Tools: Dependency Audit

2. State Capture

Snapshot baseline schema and test outputs before any refactor begins.

Tools: State Capture

Phase 02 / Verification

1. Data Flow Tracing

Follow read and write paths across database and service boundaries.

Tools: Schema Validator

2. Plan Output

Generate an ordered patch sequence verified against expected constraints.

Tools: Validation Suite

Summary

Verify before committing.

Janus keeps the primary workspace clean while agents test multi-step edits inside bounded, volatile memory sandboxes.

Source

URL: https://sevenswell.ai/#janus