CONTROL PLANE GOVERNED RUN VERIFIED EVIDENCE

SUPERINFRA / PRODUCT WHITEPAPER

Closed-loop infrastructure operations

A control plane for continuously understanding infrastructure, governing agentic work, and verifying real-world outcomes.

STATUS	VERSION	DATE
Conceptual architecture	1.0	3 August 2026

ABSTRACT

Superinfra is an AI-native control plane through which human-led, agent-operated teams understand and run infrastructure across their existing clouds, repositories, infrastructure-as-code systems, observability products, incident tools, knowledge sources, and collaboration software.

Its central abstraction is a governed Run. Any operational Trigger becomes a Run with an explicit Intent, a reviewable Plan, a policy decision, bounded execution, independent Verification, and durable Evidence. Each verified outcome is reconciled into a live Infrastructure Graph. The result is a closed loop between what an Organization knows, what it wants, what changed, and what is actually true.

Intent enters Verified reality returns

This whitepaper describes the intended product model. It is not a representation that every capability is implemented today.

1 The infrastructure operating gap

Cloud infrastructure became programmable without becoming coherent. An ordinary change may require a developer request, repository inspection, Terraform planning, cloud access, policy review, cost estimation, deployment, monitoring, documentation, and an audit record. Each system sees a fragment. The operator supplies the missing context manually.

AI accelerates the production of code and operational suggestions, but it does not remove the need for control. Google's 2025 DORA research found high AI adoption alongside lower trust and a continuing negative relationship between AI adoption and delivery stability. The relevant conclusion for infrastructure is that AI amplifies the surrounding system. Testing, internal context, version control, feedback loops, and a high-quality platform determine whether speed becomes value or instability.

The missing product is not another isolated generator. It is a control plane that maintains context continuously, constrains what agents may do, makes work reviewable, verifies the external outcome, and learns from an evidence-rich operating history.

THE PROBLEM	THE CONTROL-PLANE RESPONSE
Context is distributed across code, cloud state, telemetry, tickets, documentation, and people.	Continuous Knowledge Onboarding maintains one evidence-backed Infrastructure Graph.
Agents can produce change faster than organizations can govern or verify it.	Every Trigger becomes a policy-governed Run with bounded execution and independent Verification.
Command success is commonly treated as outcome success.	Success criteria are established before execution and checked against the real system.

2 The operating model

Intent enters

Verified reality returns

Trigger

A Trigger is anything that begins operational work: a human instruction, new repository, pull request, developer ticket, scheduled obligation, incident, alert, drift finding, cost anomaly, policy violation, or event from a Connected System.

Run and Intent

The Run is the governed unit of work. It establishes an Intent, the desired outcome inferred by Superinfra or confirmed by a person, and binds together every consequential decision and action until the work is verified or safely terminated.

Plan

A Plan is a reviewable proposal containing current context, desired transition, dependencies, expected cost, risk and blast radius, proposed infrastructure-as-code or system changes, tests, recovery approach, and explicit success criteria.

Policy decision

Approval Policies evaluate the Run. Environment, resource scope, risk class, actor, action type, cost, timing, and separation-of-duties requirements determine whether the Run is blocked, routed for approval, or permitted to execute autonomously.

Execution

Infrastructure Agents reason about bounded parts of the Run and invoke approved tools within scoped credentials. Deterministic systems execute material actions. Superinfra captures each consequential Tool Call, the authority under which it ran, the input scope, and the outcome.

Independent Verification

Execution success and outcome success are different. Verification evaluates the real system against success criteria established before execution. The verifier does not accept the executing agent's assertion as proof.

Evidence and reconciliation

Evidence preserves what the Run understood, proposed, authorized, changed, and verified. A Verified Run reconciles the new state into the Infrastructure Graph. Divergence remains visible as a Finding rather than being hidden behind a completed status.

3 The knowledge model

Continuous Knowledge Onboarding

Superinfra treats onboarding as a permanent operating function. Connectors synchronize changes continuously. Structured facts and unstructured knowledge are related into a usable model. Provenance and freshness remain attached to important claims. Contradictions and missing ownership become explicit work.

The Infrastructure Graph spans four kinds of state:

STATE	EXAMPLES	WHAT THE GRAPH PRESERVES
Declared	Terraform, OpenTofu, Helm, repository configuration	What code says should exist, including version and source
Observed	AWS, Azure, GCP, Kubernetes, runtime APIs	What providers report exists now, with freshness
Operational	Health, incidents, alerts, deployment history	How infrastructure is behaving and changing
Organizational	Owners, policies, tickets, documentation, approvals	Why it exists, who is accountable, and which rules apply

Findings become work, not wallpaper

When declared and observed state disagree, an owner is missing, or an operational signal needs attention, the graph preserves the mismatch as a Finding. A person can inspect it or start a governed Run from that exact context. Superinfra does not quietly edit the diagram until reality looks tidy.


CONTROL PLANE


GOVERNED RUN


VERIFIED EVIDENCE

4 Safe agency

Agents reason; deterministic systems execute and verify

An AI model is useful where infrastructure work is ambiguous: interpreting a developer request, connecting symptoms to recent changes, selecting relevant knowledge, proposing alternatives, or responding to a bounded exception. Deterministic systems are preferable where repeatability matters: policy evaluation, credential scope, schema validation, infrastructure-as-code planning, test execution, state comparison, evidence capture, and billing.

Superinfra combines these strengths rather than pretending that an AI system is deterministic. The agent loop is observable and evaluated. Tool access is allow-listed and scoped. Secrets are handled outside normal model context wherever possible. Material side effects require a policy decision. Verification checks the external result.

Three Trust Modes

OBSERVE	APPROVE	AUTONOMOUS
Understand without changing	Prepare, then ask	Act inside policy
Superinfra discovers, explains, and recommends. Connectors used for action remain disabled.	Superinfra constructs the complete Plan and routes it to authorized people before execution.	Superinfra executes without case-by-case approval, but only inside explicit scope and risk boundaries.

Trust Mode is not global. A customer can make routine development provisioning Autonomous, production resizing Approve, and identity-policy changes Observe. Both solo founders and regulated enterprises use the same control model.

Data and tenancy

The product necessarily processes authorized customer data to provide context and operate infrastructure. Its obligation is minimization and control, not an inaccurate claim that data is never read. The intended model is least-privilege Connectors, tenant isolation, encryption, customer content excluded from shared-model training, human access disabled by default, configurable retention, secret redaction, regional processing, private inference options, and complete attribution of agent access to a Run and policy decision.

Security properties must be demonstrated through design review, automated isolation tests, threat modeling, independent assessment, incident readiness, and an explicit claims register. This document does not claim that Superinfra is currently certified or audited.

5 A complete example

Consider a new GitHub monorepo with a Node.js backend and React frontend. The Organization already has one production service in AWS, a Terraform repository convention, a staging Approval Policy, and Datadog connected for health signals.

USER INTENT

> Create a production-ready staging environment for this repository, following existing conventions and right-sized for expected usage.

01 Repository Trigger

The new repository updates the graph. Superinfra identifies application components, build requirements, and relevant organizational ownership.

02 Context assembly

The Run relates the repository to existing AWS topology, Terraform modules, naming standards, deployment workflow, policy, and recent operating history.

03 Desired state

Superinfra adapts the Organization's convention. If a required element has no precedent, it selects an Infrastructure Blueprint and marks the new decision.

04 Visual Plan

The customer sees proposed topology, estimated cost, infrastructure-as-code diff, risks, tests, recovery approach, and success criteria.

05 Policy

The staging scope permits automatic execution under a cost ceiling, but a proposed public endpoint causes an authorized security review.

06 Execution

Constrained tools create a repository change, run validation and tests, apply the approved configuration, and deploy the application.

07 Verification

Independent checks confirm expected resources, reachability, health signals, access policy, and approved cost range.

08 Reconciliation

The Run becomes a Verified Run. The graph shows the new environment and dependencies; the Operations Feed preserves approvals, Tool Calls, results, and Evidence.

If Verification fails, the Run does not become successful merely because deployment completed. It follows the approved recovery approach, preserves partial state, and either restores the prior state or creates a clearly owned exception. The graph remains an account of reality.

6 Greenfield and brownfield

DIMENSION	GREENFIELD ORGANIZATION	BROWNFIELD ORGANIZATION
Starting knowledge	Repositories, requirements, a few explicit choices	Cloud state, infrastructure as code, multiple tools, documents, history, and tacit conventions
Primary value	Opinionated Blueprints, fast delivery, maintainable infrastructure as code, early governance	Reconciliation, ownership, drift, standardized Plans, safe change across fragmentation
Human questions	Only material cost, compliance, risk, and product choices	Contradictions, unsupported exceptions, unclear ownership, and convention precedence
First proof	A verified environment created from application context	A trustworthy graph plus a verified change that respects existing practice

The internal model and Run lifecycle remain the same. The difference is where the initial context comes from and how much reconciliation is required before action.

7 Product boundaries

- Superinfra does not replace AWS, Azure, GCP, GitHub, Terraform, Datadog, PagerDuty, Jira, or every connected product. It orchestrates their capabilities through one control plane.
- It does not rebuild a raw metrics, logs, and traces platform in P1. It relates observability evidence to infrastructure context and Runs.
- It does not silently convert an editable graph into unreviewed production state. Graph actions create governed Runs.
- It does not require one model provider. Managed inference, bring-your-own-key, and private model endpoints sit behind the same agent runtime.
- It does not promise that all Connectors have equal depth. Discovery, graph, planning, execution, and Verification coverage must be stated separately.
- It does not use customer content to train shared models.

8 Expansion of the loop

The initial control plane supports provisioning, change, developer requests, drift, configuration risk, and the contextual work required to operate them. Later products reuse the same primitives rather than forming disconnected modules.

P2 - Recurring operational intelligence

Continuous cost optimization and rightsizing; AI-assisted incident diagnosis and remediation; all governed through Trust Modes and Verified Runs.

P3 - High-risk operations

Advanced security automation, runtime response, and cross-cloud migration using the accumulated graph, Desired State Model, policies, and Evidence.

9 The compounding system

The durable product is not any one model call. It is the interaction between an Organization-specific context graph, a versioned desired state, policy-bound tool access, outcome-labeled Run traces, reusable Blueprints, and a record of trusted execution.

That system improves locally as it learns the customer's conventions and history. It improves generally through platform-authored evaluations, reusable integrations, and privacy-safe aggregate performance signals, not by pooling customer content into an unrestricted training corpus. As trust grows, more Runs move from Observe to Approve to Autonomous. The control plane becomes both more useful and more deeply embedded in daily infrastructure work.

MISSION

Make infrastructure of any scale operable by the smallest capable team.

Sources

1. HashiCorp, 2024 State of Cloud Strategy. Vendor-sponsored survey of nearly 1,200 respondents; used as directional evidence for skills, waste, automation, and platform-standardization gaps.

2. Google Cloud DORA, 2025 State of AI-assisted Software Development; see also the official announcement. Used for evidence on AI adoption, trust, platform context, and stability.

3. Stack Overflow Developer Survey 2025. Used in the supporting research note for developer trust and security signals.

4. Rootly, How to leave PagerDuty without missing a page; Rootly vs. PagerDuty. First-party vendor positioning, observed 3 August 2026.