Vosj

TECHNICAL WHITE PAPER · DEFENSIVE IP PUBLICATION

VOSJ: An AI-Native Migration Factory and its Model Context Protocol Devstation Fleet

Industrialising application migration and replatforming through a four-station factory model, a data-driven gated framework engine, and an autonomous AI execution substrate.

Document type	Technical White Paper & Defensive Publication (prior-art disclosure)
Prepared by	**Assuncao, Gustavo** — PhD, AI Infrastructure Architect
Version	2.0 — final candidate (governance + security + prior-art hardening)
Date	21 June 2026
Status	Final candidate — governance, security & calibrated novelty added; changelog in Appendix F
Classification	Public — intended for technical readership

Abstract

Cross-platform application migration and replatforming remain *artisanal, risk-laden, and vendor-siloed*. Every hyperscaler ships a capable but isolated migration toolchain; no neutral system drives a workload *across* them with auditable governance; and the labour of assessment, wave-planning, replication, cutover, and post-migration reconciliation is reinvented for each engagement. This paper presents **VOSJ**, an AI-native *migration factory* that converts this artisanal work into an observable, audited, metered, and largely autonomous assembly line.

VOSJ organises every migration into four named stations — **V**ault (assess), **O**rchestrate (plan), **S**hift (migrate), **J**ump (cut over) — whose initials are simultaneously the product's brand and its process. The stations are populated by a **data-driven, multi-template framework engine** that compiles any chosen methodology (the Microsoft Cloud Adoption Framework as the flagship template; others as overlays or cloned/custom templates) into a single cryptographically-signed phase-gate state machine. A seven-disposition decision model (the 7-R's) is treated not as advisory metadata but as a *gate input*, structurally forcing high-risk workloads onto an incremental, parallel-run (Strangler-Fig) path and making big-bang cutover *impossible by construction*.

The paper's principal contribution is the **execution substrate**: a **Model Context Protocol (MCP) server** acting as a governed order-and-tool channel, and a fleet of **"devstations"** — per-identity, sandboxed, autonomous coding agents, each with its own credentials, repository clone, and audit trail. Human engineers work live; **AI digital twins** take over off-hours, driving discovery, runbook authoring, transfer, and reconciliation through the same pipeline a human would use. A small set of **safety invariants** — *no agent may sign its own gate*, four-eyes change validation, fail-closed credential handling, and verified-before-cutover — keeps autonomy auditable. We describe the architecture, the data model, the security and compliance posture, the economics, a worked end-to-end scenario, an evaluation methodology, and an enumerated set of inventive claims offered as a defensive publication.

Keywords: cloud migration, replatforming, migration factory, Model Context Protocol, autonomous agents, AI-native DevOps, Strangler-Fig, phase-gate governance, 7-R disposition, Cloud Adoption Framework, digital twin, verified cutover, reconciliation.

Contents

1 Executive Summary

VOSJ turns migration from a bespoke consulting effort into a governed, AI-driven factory: any source, any target, both connectivity modes, with audit-grade evidence and near-zero downtime by construction.

The thesis of this paper is simple to state and consequential in practice: **migration is a manufacturing problem dressed as an engineering problem.** The reason migrations are expensive is not that any single step is hard — first-party engines already replicate disks, convert images, and migrate databases competently — but that the steps are *uncoordinated, unrepeatable, unaudited, and human-bound*. Treating migration as a factory with named stations, a compiled process, machine-checked gates, and an autonomous labour force resolves all four deficiencies at once.

1.1 The four contributions

Contribution	What it is	Why it matters
V·O·S·J factory model	Four named, observable, audited, metered stations — Vault, Orchestrate, Shift, Jump — where the brand *is* the process.	Gives buyers a fixed, legible path and gives operators a manufacturing line instead of a project.
Framework template engine	A data-driven engine that compiles *any* methodology into one signed phase-gate state machine; ships Microsoft CAF as the flagship template, plus cloneable/custom templates.	Methodology becomes configuration, not code. One governance core serves every customer's preferred framework.
MCP devstation fleet	A Model Context Protocol control plane commanding a fleet of per-identity autonomous coding agents that do the migration toil.	Factory throughput at software cost; humans supervise rather than execute.
Safety invariants	No agent signs its own gate; four-eyes change validation; fail-closed secrets; verified-before-cutover.	Autonomy without loss of auditability — the property regulated buyers require.

1.2 The headline claim, stated precisely

> **Factory throughput with audit-grade governance and AI-native execution.** VOSJ aims to combine the *speed* of a large-scale migration factory, the *rigour* of transition-architecture planning, and the *risk profile* of an incremental Strangler-Fig cutover — executed by autonomous agents that *never self-sign a gate*.

The remainder of this paper substantiates that claim. §2–§4 frame the problem and prior art. §5–§8 describe the factory, the gated framework, the disposition engine, and the template engine. §9–§11 describe the AI execution substrate — the MCP server, the devstation fleet, and the human–AI operating model — which is the paper's primary novelty. §12–§16 cover governance, verification, the data model, security/compliance, and provider connectivity. §17–§20 cover delivery-system assessment, economics, a worked scenario, and evaluation. §21 enumerates the inventive claims for defensive-publication purposes; §22–§24 close with limitations, future work, and conclusions.

2 Introduction & Motivation

Enterprises hold portfolios of hundreds to thousands of applications distributed across on-premises data centres, private virtualisation estates (including Microsoft Hyper-V and other hypervisor / HCI platforms), and multiple public clouds. Business events — data-centre exit, acquisition, cost pressure, regulatory localisation, end-of-support, or a strategic platform decision — routinely require moving large subsets of that portfolio from one platform to another, frequently *reshaping* them in the process (IaaS → managed services, VM → container, monolith → cloud-native).

2.1 The migration tax

We define the **migration tax** as the avoidable cost premium incurred because migration is performed as a craft rather than as a manufactured process. It has five recurring components:

1. **Toolchain churn.** First-party migration tools are renamed, retired, or restricted on the vendors' own cadence. A buyer should not have to track which console is alive this quarter.
2. **Reinvention per engagement.** Assessment, dependency mapping, wave planning, runbook authoring, replication, cutover, and reconciliation are rebuilt by hand for each project.
3. **Opaque risk.** Cutovers are frequently big-bang or weakly verified; rollback plans are written the night of, by the same people executing the change.
4. **Unaudited execution.** Who advanced which workload, on what evidence, signed by whom, is often reconstructed after the fact from chat logs and email.
5. **Human bottleneck.** The work scales linearly with skilled engineers, who are scarce and expensive, so throughput is capped and parallelism is limited.

> **Design response.** Each component of the migration tax maps to one VOSJ mechanism: churn → a neutral factory over durable, API-drivable pillars (§16); reinvention → the compiled framework engine (§8); opaque risk → the 7-R-as-gate-input + Strangler-Fig default + independent rollback authoring (§6–§7, §12); unaudited execution → HMAC-signed gate transitions + full audit log (§12, §14); human bottleneck → the MCP devstation fleet (§9–§11).

2.2 Why now — the AI inflection

The labour-bottleneck component was, until recently, irreducible: the toil of reading a runbook, calling a transfer tool, watching replication lag, and reconciling row counts required a skilled human. Tool-using AI agents — large language models that plan, call tools, read and write code, and operate a terminal — change this. An agent can now perform the bulk of station work, provided it is given (a) a *standard way to reach tools and systems* and (b) a *governance envelope* that makes its actions safe and auditable. The first is the **Model Context Protocol**; the

second is the **gated framework engine plus safety invariants**. VOSJ is the composition of the two around a migration domain.

2.3 Scope of this paper

This is an architecture-and-method paper, not a benchmark study. It describes the system as designed and the invariants it enforces, situates it in the migration literature, and offers an evaluation methodology rather than headline performance numbers (any third-party outcome figures cited are illustrative and flagged as requiring independent verification). The intended reader is a technical practitioner: a cloud or platform architect, a migration lead, a DevOps/SRE engineer, or a researcher in autonomous software engineering.

3 Problem Statement

We state the problem formally to make the design requirements precise.

3.1 Formal framing

Let a *source estate* be a set of workloads $W = \{w_1, \ldots, w_n\}$, each with an inventory record, a dependency relation $D \subseteq W \times W$, a data footprint, and a set of non-functional constraints (availability target, data-residency, compliance regime). Let a *target* be a platform P_t with a landing zone L. A **migration** is a function that produces, for each in-scope workload, a disposition $d(w) \in R$ (the 7-R set), a transition plan, an executed cutover, and a proof of equivalence $\pi(w)$ between source and target behaviour, subject to:

- **Ordering:** cutover sequence respects the dependency DAG (topologically sorted, acyclic).
- **Continuity:** for each w, the source keeps serving until $\pi(w)$ holds (no unverified cutover).
- **Reversibility:** an independent rollback exists and has been rehearsed before execution.
- **Auditability:** every state transition is attributable (actor, evidence, timestamp) and tamper-evident.
- **Authority separation:** the actor that *performs* a transition is not the actor that *authorises* it.

3.2 Requirements derived

#	Requirement	Satisfied by
R1	A neutral control plane spanning heterogeneous source/target platforms.	§5, §16
R2	A repeatable, configurable process model (methodology as data, not code).	§6, §8
R3	Disposition decided per workload and enforced as a precondition to planning.	§7
R4	Incremental, parallel-run cutover as the default; big-bang structurally unavailable for high-risk dispositions.	§6, §7
R5	Machine-checked, cryptographically-signed gates with separation of authoring and authorising.	§12, §14
R6	Verified-before-cutover equivalence proof (checksums, row counts, smoke).	§13
R7	A scalable execution labour force that is auditable and credential-isolated.	§9, §10, §15
R8	Standardised, least-privilege access to external systems and tools.	§9, §15, §16
R9	Full data-protection posture for cross-border/cross-cloud data movement.	§15

The non-negotiable constraints — continuity (R4–R6), authority separation (R5), and auditability (R5, R8) — are the ones most often violated by ad-hoc migrations, and the ones VOSJ enforces structurally rather than by policy.

4 Related Work & Conceptual Foundations

VOSJ does not invent a methodology; it *composes and operationalises* established ones. The synthesis below is the conceptual basis for the gated framework (§6) and the disposition engine (§7).

Source	Contribution adopted by VOSJ
Microsoft Cloud Adoption Framework (CAF) Strategy → Plan → Ready → Adopt (Migrate/Modernize) → Govern → Secure → Manage	The overall phase spine; the landing-zone-first ("Ready") discipline; the rollback-plan-before-cutover rule; explicit go/no-go checkpoints. CAF is the **flagship framework template** shipped with VOSJ.
A major hyperscaler's migration-acceleration program Assess → Mobilize → Migrate & Modernize	The three-arc shape; the Migration Readiness Assessment across CAF-style perspectives; the TCO/business case as an entry gate.
Another hyperscaler's rapid-migration program Assess → Plan → Migrate → Optimize	Data-driven discovery and an explicit post-cutover Optimize phase.
The 7 R's (matured from the earlier industry-standard 5 R's)	The disposition decision model (§7): Retire, Retain, Rehost, Relocate, Repurchase, Replatform, Refactor.
TOGAF ADM, Phases E & F Opportunities & Solutions; Migration Planning	Transition architectures (value-delivering intermediate states) and a signed Implementation & Migration Plan with dependency/cost/benefit sequencing.
Strangler-Fig pattern (Fowler)	Incremental, parallel-run, proxy-switch modernization — the default cutover posture, mandatory for Refactor/Relocate.
Near-zero-downtime hypervisor migration technology (live / bulk / replication-assisted modes)	Near-zero-downtime cutover mechanics for the Shift station.

Application-centric "migration factory" practice	The factory throughput model and application-centric dependency mapping at scale; validation delivered as an artifact, not an afterthought.
Model Context Protocol (MCP)	The open client/server standard by which agents discover and call tools and reach external systems — the substrate of the execution layer (§9).
DORA metrics	Delivery-system maturity measurement used by the CI/CD 365° assessment (§17).

> **The gap VOSJ fills.** The vendor frameworks describe *what* to do; the factory practice describes a *way to do it at scale*; TOGAF describes *how to plan transitions rigorously*; Strangler-Fig describes *how to cut over safely*. None of them specify an *autonomous, auditable execution substrate* that performs the work while preserving separation of authoring and authorising. That connective tissue — the MCP devstation fleet under a signed-gate engine — is the contribution of this paper.

5 The VOSJ Factory — Architectural Overview

VOSJ is a control plane that orchestrates an existing, production-grade migration domain (a library of point-to-point `migrate-<source>-to-<target>` executors covering Microsoft Azure, Azure Local, and Hyper-V alongside other public clouds and hypervisor / HCI platforms), a provider registry, an encrypted credential vault, self-serve onboarding, and metered billing. It wraps these capabilities in authentication, capability-based authorisation, a full audit log, a live per-migration board, and a buyer-facing self-serve experience. The architectural principle is **extend, do not fork**: VOSJ *productises and unifies* capabilities that already exist rather than duplicating them.

5.1 The four stations

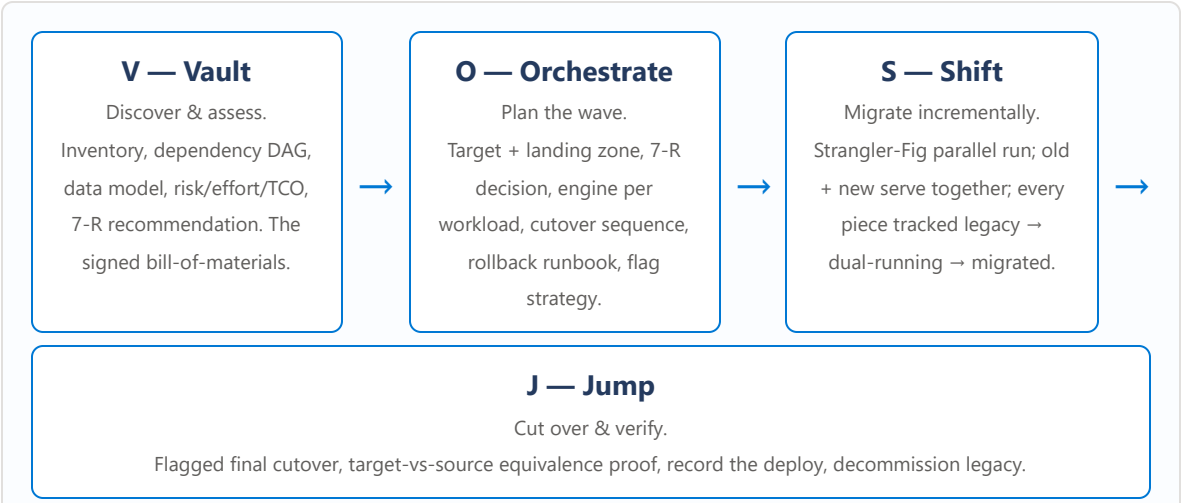

Figure 1. The V·O·S·J station pipeline. The brand letters are the process stages; each station is observable, audited, and metered.

5.2 Layered architecture

Experience layer — self-serve buyer portal; live per-migration "glass-room" board; consultant Framework Library (browse / use / clone / create templates).

Factory control plane — the V·O·S·J station engine; the framework template compiler; the signed phase-gate state machine; the disposition (7-R) engine; the audit and metering services.

AI execution substrate — the MCP server (governed tool/order channel) and the devstation fleet (per-identity autonomous agents + human engineers); digital-twin off-hours operation.

Capability layer — point-to-point migration executors; provider registry; encrypted credential vault; discovery connectors; reconciliation engine; CI/CD & observability scanners.

Platform foundation — Kubernetes-native runtime; PostgreSQL system-of-record (schema-per-domain); RBAC/identity; secrets management; the LLM bridge; the standard CD pipeline.

Figure 2. Five-layer architecture. The AI execution substrate (purple) is the paper's primary novelty; the governance core (control plane) is what makes that autonomy safe.

5.3 Cross-cutting properties

- **Observable** — every station exposes a live board; every transition emits an event.
- **Audited** — who advanced/cut-over/decommissioned what, on what evidence, source → target, is recorded immutably.
- **Metered** — effort and cost per migration are captured, making outcomes reportable and priceable.
- **Config-driven** — providers, regions, prices, and thresholds are configuration, never hardcoded.
- **Flag-gated** — no half-built surface ships unflagged; closure follows the standard change-management and CD path.

6 The Seven-Phase Gated Reference Framework

The stations are the *operating surface*; a **framework** populates them with phases, gates, deliverables, and roles. VOSJ ships a reference framework of seven phases, each terminating in one or more **gates** that require machine-checked evidence and a human sign-off by a named role. The framework maps cleanly onto the four stations.

Ph.	Phase	Goal & key deliverables	Station	Exit gate & signer
P1	**Envision** Strategy & business case	Motivation, TCO/value case, readiness assessment across business/people/governance/platform/security/operations; target platform(s) chosen.	V	Discovery sign-off — *director + customer sponsor*
P2	**Examine** Discover, baseline, decide	Application-centric inventory + dependency DAG; performance/error baselines; a 7-R disposition for *every* in-scope workload; wave plan v0; risk register v0.	V	Kickoff complete — *IT lead*
P3	**Engineer the Plan** Transition architecture	Transition architectures (TOGAF); data-migration method selection; cited cutover runbook; *independent* rollback runbook (separate author/context); static tabletops; change requests through four-eyes validation.	O	Data-method (*DBA*); rollback-tabletop (*director + DBA*); planning sign-off (*director + IT lead*)
P4	**Establish Readiness** Landing zone & freeze	Built/validated landing zone (network, identity, security baseline); calendar lock; baseline-drift check; infra/app freeze; vendor engagements verified.	O	Drift check (*auto*); execution freeze (*InfoSec + DBA*); vendor verified (*director*)
P5	**Shift / Execute** Verified incremental cutover	Full-panel go/no-go; conductor steps the runbook calling executors via the bridge; Strangler-Fig parallel run / proxy switch; continuous contingency monitoring; per-step evidence hashed.	S	Go/no-go (*full panel*); cutover (*director + DBA, continuous*; reverse needs 3-way)
P6	**Verify & Optimize** Reconcile & tune	Reconciliation (row counts, checksums, sequence/identity values, BLOB counts, constraint re-validation, plan parity); post-migration metrics vs baseline; right-sizing.	J	Reconciliation pass — *DBA* (revocable within 30 min on drift)

P7	**Jump to BAU & Learn** Hand over & improve	Per-app BAU sign-off; monitoring/governance to steady state; *decommission source* after confirmed BAU + retention; post-mortem; lessons fed back into the template and the agent learning loop.	J	Return-to-BAU (*IT lead*); completed/archived (*director, partner attests*)

6.1 Why gates, and why human signers

A **gate** is a transition in a finite-state machine that cannot fire until (a) its machine-checkable criteria are satisfied (e.g. "runbook citations resolved, dependency sort acyclic, tabletops passed") and (b) a human in a named role applies a cryptographic signature. The two-part condition is deliberate: machines verify *facts*; humans accept *accountability*. This is the mechanism by which an autonomous fleet can perform the work while a human remains the authority of record. §12 details the signing and the no-self-sign invariant.

> **Verified-before-Jump is structural.** The last Shift → Jump transition is a mandatory gate on *every* framework template; a template cannot omit it. A non-verified cutover is therefore not a policy violation — it is an unreachable state.

7 The 7-R Disposition Engine

Disposition is assigned per workload in Phase 2 (Examine) and stored on the workload record. It is the single most consequential decision in a migration, and VOSJ's distinctive treatment is to make it a **gate input** rather than advisory metadata: the kickoff gate cannot pass until every in-scope workload carries a disposition, and the planning gate selects the runbook template and executor *strictly* from that disposition.

Disposition	Meaning	Engine consequence in VOSJ
Retire	Decommission; no migration.	Drops out after Examine; appears only in the Jump decommission step.
Retain	Keep at source (regulatory/technical).	Becomes a documented split-environment dependency.
Rehost	Lift-and-shift to IaaS.	Offline / near-zero-downtime executor; simplest runbook template.
Relocate	Move hypervisor wholesale (e.g. on-prem virtualization → cloud-hosted virtualization).	Near-zero-downtime bulk / replication-assisted cutover via the virtualization executors; **Strangler-Fig path enforced.**
Repurchase	Drop-and-shop to SaaS.	Data extract + cutover; no infrastructure lift.
Replatform	Lift-and-reshape (e.g. managed database).	Runbook adds reshape steps; reconciliation thresholds tightened; gated on a delivery-system (CI/CD) readiness check (§17).
Refactor	Re-architect cloud-native.	**Strangler-Fig mandatory** — incremental, parallel-run, proxy-switch. Big-bang is unavailable.

> **The structural guarantee.** Because Refactor and Relocate dispositions resolve to runbook templates that only emit incremental, parallel-run cutover steps, the highest-risk transformations are *physically incapable* of producing a big-bang plan. Risk control is enforced by the disposition → runbook compilation, not by reviewer diligence.

7.1 Disposition as a typed contract

Each disposition is a typed contract `⟨ executor_class, runbook_template, reconciliation_profile, cutover_style ⟩`. Replatform/modernize dispositions additionally carry a *delivery-system precondition*: a workload cannot be safely containerised or modernised

without first standing up its pipeline, quality gate, GitOps flow, and observability — so the CI/CD 365° assessment (§17) is a hard dependency of those dispositions.

8 The Framework Template Engine

Methodologies differ by customer, regulator, and partner. Rather than hardcode one, VOSJ generalises a single hardcoded phase-gate engine into a **data-driven, multi-template engine**. A consultant may *select* an existing template, *clone* and customise it, or *create* a new one. Crucially, the signing, gate-row persistence, oversight roles, and RBAC machinery are **reused unchanged**; only the state-machine *source* moves from frozen constants to template rows.

8.1 Compilation model

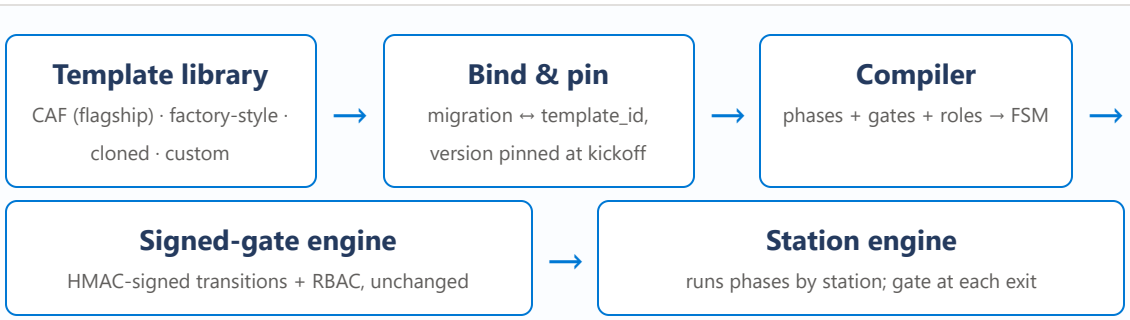

Figure 3. A template *compiles* to the same FSM shape the signed-gate engine already runs. Methodology becomes configuration; governance stays constant.

8.2 Data model (additive)

Table	Purpose & key columns
`framework_template`	A selectable/cloneable methodology. `id, name, source, version, parent_template_id` (clone lineage), `owner, tenant_id, visibility (public|tenant|private), status (draft|published|archived)`.
`framework_phase`	One ordered phase. `template_id, ordinal, key, name, goal, activities[], deliverables[], entry_criteria, exit_gate_id, station (V|O|S|J), roles[]`.
`framework_gate`	Exit gate of a phase. `template_id, phase_id, gate_key, criteria, signoff_roles[], requires_signature`.
`framework_role`	Per-template oversight role (generalises the hardcoded role set). `template_id, role_key, display, rbac_capability`.
`migration` (extended)	Binds a run to its framework: `+ framework_template_id, + framework_version` (pinned at kickoff so a later template edit cannot mutate an in-flight run).

8.3 Consultant experience

- **Browse** — public seed templates plus the tenant's own, each with a strongest-fit hint.
- **Use** — bind a template to a migration; the version is pinned.
- **Clone & customise** — fork a seed into a draft (lineage recorded); edit phases, activities, deliverables, gates, roles, and the V·O·S·J station map; publish and version; view a diff against the parent.
- **Create** — blank, or from a Vault→Orchestrate→Shift→Jump skeleton.

> **Backward compatibility by construction.** The pre-existing single hardcoded methodology is re-seeded *one-to-one* as a template; state names are identical, so in-flight runs are unaffected and inherit signed transitions for free. This is the practical embodiment of "extend, don't fork."

9 The MCP Server — Control Plane for Tools and Orders

The Model Context Protocol server is how autonomous agents reach the world — and how the world reaches them — under one governed, audited, least-privilege channel.

The **Model Context Protocol (MCP)** is an open standard for connecting AI agents to external systems and tools via a uniform client/server interface. MCP encodes calls as **JSON-RPC 2.0** and currently defines two standard transports — `stdio` (a local child process) and *Streamable HTTP* (the earlier HTTP+SSE transport was deprecated in the 2025-03-26 revision); VOSJ runs local executors over `stdio` and remote servers over Streamable HTTP. VOSJ uses an **MCP Hub** in two directions:

- **Outbound (agents → systems).** The Hub is a registry and connection manager for external MCP servers — source control, issue tracking, messaging, cloud providers, and the migration executors. When an agent needs to open a pull request, query an inventory, or invoke a transfer, it calls an MCP *tool*; the Hub routes the call, injects credentials from secrets at connection time, enforces per-tool authorisation, applies timeouts, and logs the call.
- **Inbound (clients → platform).** The Hub also exposes selected platform services as MCP servers, so an external or internal client can reach platform data and actions through the same governed surface.

9.1 Why MCP is the right substrate

Property	Consequence for a migration factory
Uniform tool interface	One way to call any system — executors, clouds, source control — so adding a provider is a registration, not a rewrite.
Credential indirection	Servers reference secrets, never embed them; rotation reconnects transparently; no long-lived keys in agent context.
Per-tool authorisation (Hub control)	The Hub binds each agent's RBAC to the tools it may call; a call it is not entitled to make is *rejected by the Hub*. This is a VOSJ-Hub control, not a property of the MCP protocol itself.
Auditability	Every tool call is logged with actor, arguments, result, and duration — the audit trail of external interaction (R8).
Timeouts & isolation	External servers run as child processes or outbound connections; a hung dependency cannot block agent execution indefinitely.
Protocol vs. Hub boundary	MCP provides the uniform tool/resource interface, the transport, and (over HTTP) an OAuth 2.1 Resource-Server authorization model; VOSJ *adds* per-tool RBAC, signed-server allow-lists, and capability checks at the Hub.

On HTTP transports MCP defines an OAuth 2.1 model in which the MCP server is an OAuth *Resource Server*: Protected Resource Metadata (RFC 9728) advertises the authorization server, and audience-bound Resource Indicators (RFC 8707) plus PKCE constrain a token to its intended server; on `stdio` the server sources credentials from its environment. The per-tool RBAC, allow-lists, and capability checks above are **VOSJ-Hub controls layered on top** — not guarantees of MCP itself.

9.2 Order dispatch — a composition, not an MCP property

Order dispatch is a **composition**, not a feature of MCP. The order channel is a **database-backed durable work queue**: an operator (human or a supervising agent) enqueues a work order; a target devstation (§10) claims it atomically, executes it on its own isolated clone, and reports progress and signed evidence back. MCP is used *within* that execution for governed *tool* access (host-initiated, request/reply) — it is not the bus itself. Separating the two — a durable queue for *orders*, MCP for *tools* — decouples *who issues work* from *where work runs*, and keeps each layer doing what it was designed for (an emerging agent-to-agent protocol such as A2A is the conventional choice where direct agent-to-agent coordination is needed).

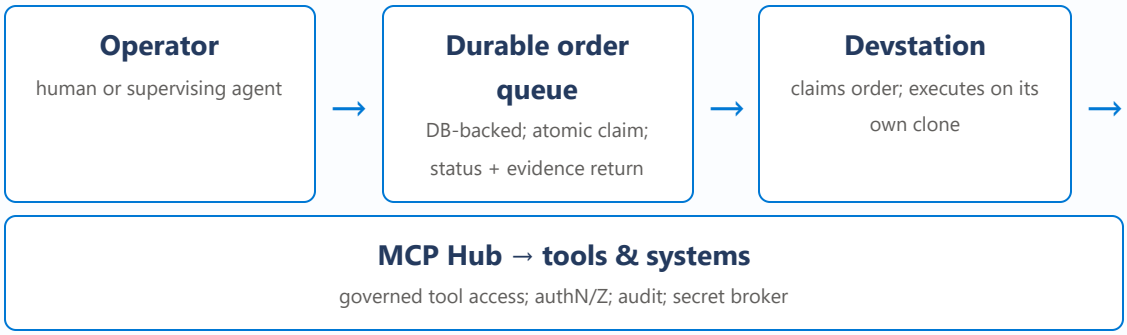

Figure 4. Orders are dispatched through a durable work queue; governed tool access is via MCP. Orders flow in; signed evidence flows back.

9.3 Security model (summary; full treatment in §15)

- Credentials are environment-reference, resolved from a secret store at connect time — never plaintext in configuration; long-lived secrets are brokered out of the execution pod (§10.4).
- Per-server allow-lists and per-tool RBAC bound the blast radius of any single agent.
- Every tool call and every order transition is logged immutably for audit.
- Input arguments are validated against each tool's schema before dispatch.
- External MCP servers and *tool responses* are treated as untrusted input; the agent/MCP threat classes (indirect prompt injection, tool poisoning, confused-deputy) are addressed in the threat model (§15.5).

10 The Devstation Fleet

A **devstation** is an autonomous, sandboxed software-engineering agent that behaves like a human engineer's workstation. Because it executes model-authored code, each devstation runs under a **hardened runtime class** (e.g. gVisor, Kata Containers, or a Firecracker microVM via a Kubernetes `RuntimeClass`), and holds:

- its **own identity** — a distinct service/persona account, so its actions are individually attributable;
- its **own repository clone** — a private working tree, so concurrent stations never corrupt each other's work;
- its **own model credential** — a dedicated seat for the coding agent it runs;
- its **own source-control credential** — so commits, branches, and pull requests carry the station's identity;
- a **worker loop** — a poller that claims the next unit of work, executes it headlessly, and reports back.

> **Isolation is a correctness requirement, not a convenience.** A naive design in which many agents share one working tree produces silent stomping — a concurrent reset can discard another agent's uncommitted work. The mandatory model is therefore *one station = one identity = one clone = one account*, with commit-via-pull-request as the only path that crosses station boundaries. This lesson is encoded as an architectural invariant.

10.1 Fleet roles

The fleet is heterogeneous; stations specialise by role — for example architect (design), builder (implementation), reviewer (independent review), tester (validation), migrator (the migration executors), fixer (remediation), and deployer (release). This division mirrors a human engineering organisation and, more importantly, *enforces separation of duties*: the agent that builds a change is not the agent that reviews or approves it (cf. the four-eyes and no-self-sign invariants in §12).

10.2 Lifecycle & resilience

Concern	Mechanism
Provisioning	A station is created from a template (identity, model tier, resources, working hours, source-control identity) producing a deployment, a credential secret, and a worker.

Liveness	A keepalive controller relaunches dead worker loops fleet-wide; a station that crashes is restored without human intervention.
Credential rotation	Keys (model, source-control, console) are rotated through a tamper-evident, audited path; rotation updates the live secret and restarts the worker.
Work intake	Stations pull from a durable work queue (database-backed tickets), claim atomically (open → in-progress), and emit status, so coordination does not depend on fragile shared state.
Decommission	Graceful drain: finish or hand back claimed work, archive transcripts, remove deployment and secret.

10.3 Cost posture

Because the fleet runs on the operator's own compute and on flat-rate model subscriptions, the marginal cost of an additional unit of migration work approaches the cost of compute, not the cost of a consultant-hour. The factory's throughput is therefore governed by orchestration and safety review, not by headcount — the precise inversion of the human bottleneck identified in §2.1. (Flat-rate seats carry fair-use ceilings; at the ceiling the fleet queues/throttles or falls back to metered rates — see §22.)

10.4 Runtime isolation & credential brokering

Because a devstation runs model-authored code beside privileged credentials, two hardening rules apply. First, the execution sandbox is a **hardened runtime class** (gVisor / Kata / Firecracker-microVM via `RuntimeClass`), so a container escape does not reach the node or neighbouring pods. Second, **long-lived secrets do not live in the execution pod**: a broker/sidecar vends *short-lived, audience-scoped, per-task* credentials (workload-identity federation / SPIFFE-style OIDC for the model seat; short-TTL tokens for source control), so a compromised station leaks at most a narrow, expiring grant. The claim is therefore *per-identity segmentation with minimised standing credentials* — necessary, but explicitly *not* a claim that code execution and credentials are fully isolated from one another (see the threat model, §15.5).

11 The AI-Native Execution Model

The differentiator is not "AI assists the engineer." It is "the factory's labour force is autonomous agents, supervised by humans, governed so that no agent can grant itself authority."

11.1 Human–AI partnership across the clock

Human engineers operate live during working hours; **AI digital twins** take over off-hours and during absence, driving the same pipeline through the same MCP channel a human would use. A digital twin is a persona configured with the standards and judgement of its principal, authorised to perform research, design, implementation, and review autonomously, and instructed to escalate only genuine owner-level decisions (irreversible actions, external commitments, spend) to a human. The result is a workforce that is *continuously available* without being *unsupervised*: every twin's output flows through the same gates and reviews as a human's.

Orders arrive on the MCP server (from a human operator or a supervising agent).

Discovery & assessment agents fingerprint the source estate and the delivery system; populate the Vault system-of-record.

Planning agents author the cited runbook; a *separate* agent (separate context) authors the independent rollback runbook.

Execution agents (migrator role) call the point-to-point executors via the bridge; capture hashed per-step evidence.

Verification agents run reconciliation; **remediation** agents close gaps the scanners flag.

Humans hold the gates: every station transition still requires a signed sign-off by a named human role.

Figure 5. Agents perform; humans authorise. The autonomy is bounded by the gate machine, not by trust.

11.2 The execution model is the moat

The methodologies VOSJ composes are public; the executors wrap public APIs; the gates are an engineering pattern. What is hard to replicate is the *operationalised combination*: an

autonomous fleet that does the toil, a governed channel that makes its actions safe and auditable, digital twins that keep the factory running around the clock, and a self-improvement loop that feeds each migration's lessons back into the templates and the agents' memory. This is why the AI execution substrate, and not the migration content, is the defensible core.

11.3 Self-improvement loop

Phase 7 (Jump to BAU & Learn) closes the loop: each migration's post-mortem produces template-improvement deltas and feeds the agents' learning/"dreaming" process from the captured activity dataset. Over many engagements the framework templates sharpen and the agents' priors improve, so the factory gets faster and safer with volume — a learning curve unavailable to a purely human consultancy.

12 Governance & Safety Invariants

Autonomy is only acceptable if it is *bounded*. VOSJ enforces a small set of invariants structurally — in the data model and the engine — rather than by policy or reviewer diligence.

> **Invariant 1 — No agent self-signs a gate.** Persona service identities are minted *without* any `sign-as-<role>` capability. An agent can perform all the work leading to a gate and can request a transition, but it cannot apply the human sign-off that authorises it. Self-authorisation is structurally impossible, not merely disallowed.

> **Invariant 2 — Separation of authoring and authorising.** The actor that performs a change is not the actor that approves it. The fleet's role specialisation (§10.1) and the gate sign-off roles (§6) jointly enforce this. Independent rollback authoring (a separate agent in a separate context) is a direct application.

> **Invariant 3 — Four-eyes change validation.** Change requests pass through an independent validator that produces a diff-impact report before a gate that depends on them can clear.

> **Invariant 4 — Tamper-evident transitions.** Every gate transition is recorded as an HMAC-SHA256-signed row binding actor, role, evidence, and timestamp. The signed ledger is the authoritative history; transitions cannot be back-dated or forged without detection.

> **Invariant 5 — Fail-closed by default.** Safety-relevant defaults deny. A credential vault with no key does not fall back to a development key; a stale baseline blocks readiness; a missing verification blocks cutover. The absence of a positive proof is treated as failure, never as permission.

> **Invariant 6 — Verified-before-cutover (restated).** The source serves until the migrated piece is proven equivalent (§13). Enforced as a mandatory gate on every template (§6.1).

> **What the invariants do and do not bound.** No-self-sign and per-tool RBAC bound an agent's *capability*, not its *intent* under a poisoned or injected context. An agent subverted via prompt injection or a poisoned tool response need not authorise anything to misuse the tools it already holds (see the threat model, §15.5). The **human-signed gate is the load-bearing backstop** for irreversible steps — cutover, decommission, and credential

> rotation — and machine-checkable gate criteria (§6.1) exist precisely so a human is not asked to rubber-stamp an agent-assembled narrative.

12.1 RBAC and least privilege

Authorisation is capability-based: actions are named `{domain}:{resource}:{action}` and granted to roles; every data route requires authentication and every mutation requires an explicit capability. Advancing a station, cutting over, and decommissioning are owner/admin-gated. Agents are principals in this system with deliberately constrained grants — sufficient to do their job, insufficient to authorise it.

12.2 Auditability as a first-class output

Every station transition, access grant, tool call, and cutover is logged immutably with who/what/when/source→target. For regulated estates this audit log is itself a deliverable: defensible, exportable evidence that the migration was performed under separation of duties with verified cutovers — precisely what a pure framework leaves to the integrator to assemble after the fact.

12.3 Mapping the gate machine to recognised governance frameworks

VOSJ's invariants (§12) and gate machine (§6, §14.1) are not a private control language; they are an implementation of established governance frameworks, expressed in code. For a customer POC that must scale into a regulated production estate, this mapping is the difference between "a clever pipeline" and "an auditable control environment a board and an external auditor will accept." The table below binds each VOSJ mechanism to the framework objective it satisfies, by the framework's real name and current structure.

Framework (current structure)	What it requires	How the VOSJ gate machine implements it
ISO/IEC 38500 Corporate governance of IT — six principles (Responsibility, Strategy, Acquisition, Performance, Conformance, Human Behaviour) exercised through an *Evaluate–Direct–Monitor* cycle by the governing body.	The board/governing body *directs* the use of IT, *evaluates* proposals against strategy and risk appetite, and *monitors* conformance — without becoming the executor.	**Evaluate** = the P1 Envision / P5 Go-No-Go gates (sponsor + director assess value, risk, readiness). **Direct** = the bound framework template that fixes mandatory gates and signer roles. **Monitor** = the tamper-evident ledger (§12.2, §14.4) and the live per-migration board, which report conformance back to the governing body without it touching the machinery.
COBIT 2019 40 governance & management objectives across five domains: `EDM` (governance), `APO`, `BAI`, `DSS`, `MEA` (management).	Separate *governance* (EDM — set direction, monitor) from *management* (APO/BAI/DSS/MEA — plan, build, run, measure), with defined roles, evidence, and assurance for each objective.	**EDM** ↔ gate sign-off authority and the no-self-sign invariant (Invariant 1). **APO/BAI** ↔ the Orchestrate station — planning, transition architecture, landing-zone build (P3–P4). **DSS** ↔ the Shift station — controlled execution and contingency monitoring (P5). **MEA** ↔ the Verify station and reconciliation engine (P6) plus the audit ledger — the "monitor, evaluate, assess" line that COBIT keeps structurally distinct from delivery.
ITIL 4 — Change Enablement Change types: *standard* (pre-authorised), *normal* (assessed & authorised), *emergency* (expedited + post-implementation review); a defined	Every change to a production service is classified, risk-assessed, authorised by a competent change authority appropriate to its risk, and reviewed after the fact.	A 7-R disposition (§7) is the change-type classifier: *Rehost* resolves to a near-*standard* path; *Replatform/Refactor* to *normal* change under the full gate panel; an in-flight contingency reversal is the *emergency* path, deliberately requiring a **three-way** authority (a stronger ECAB). The full-panel P5 Go/No-Go *is* a Change Advisory Board convened as a gate; the P6 revocable window and P7 post-mortem are the mandated post-implementation review.

change authority (CAB / ECAB).		
IIA Three Lines Model (2020) Governing body; *management* (first-line delivery + second-line risk/control oversight); *internal audit* (third line, independent assurance).	Risk is owned in the first line, overseen by a distinct second line, and independently assured by a third line that does not report to the people it audits.	**First line** = builder/migrator/deployer devstations and the live human engineers who own and run the change. **Second line** = the reviewer-role station, the four-eyes validator (Invariant 3), the baseline-drift guard, and the fail-closed vault — the always-on control function that constrains the first line. **Third line** = the exportable, tamper-evident ledger and control-test pack (§12.6) that an internal or external auditor reads independently — assurance over a record neither line can rewrite.
SOC 2 (Trust Services) — CC8.1 & **SOX ITGC** Authorise, design, test, approve, and implement changes; segregation of duties; logical access.	A change must be authorised, tested, approved by someone other than the implementer, traceable from deployment back to approver to requester, with continuous system-generated evidence over the period.	The signed gate row (§14.1) *is* the CC8.1 control record: it binds requester (`actor`), independent approver (`signerRole` , minted only to a human), evidence hashes, and timestamp into one tamper-evident object — satisfying segregation of duties and end-to-end traceability as a by-product of normal operation, not as an after-the-fact reconstruction.

> **The governing observation.** VOSJ does not ask a customer to adopt a new governance regime. Its station map is COBIT's governance/management split; its gates are ITIL change-enablement decisions with a real change authority; its role separation is the IIA Three Lines Model; its ledger is the SOC 2 / ITGC change-management evidence. The autonomous fleet runs *inside* a control frame the customer's auditors already recognise.

12.4 The governance operating model in practice

A framework mapping is necessary but not sufficient; governance must *run*. The operating model below is the human and procedural layer that surrounds the gate machine and keeps an autonomous fleet under board-grade oversight. It is deliberately lightweight at POC scale and hardens on the path to production (§poc-to-prod).

THE GATE COUNCIL (CHANGE-ADVISORY FUNCTION)

Each engagement convenes a **Gate Council** — the standing change authority for that migration, and the human embodiment of ITIL's CAB and COBIT's EDM domain. It is small, named, and quorate by role, not by headcount:

- **Chair** — the migration director (accountable owner; second line). Convenes the council, holds the no-go authority, and is the signer of record for director-gated transitions.
- **Customer sponsor** — the business owner of the estate; co-signs the P1 and (for high-risk waves) P5 gates. The sponsor is the link to the customer's own governing body.
- **Risk & control reviewer** — the second-line voice (security/compliance), owning the threat-model posture (§15.5), the waiver register, and control-test sign-off.

- **Technical authorities** — the DBA (data fidelity / reconciliation) and InfoSec (freeze, access), each the named signer for their respective gates.
- **AI fleet supervisor** — a human accountable for the autonomous fleet's behaviour: the principal behind the digital twins (§11). This role is explicitly *not* a signer for work the fleet performed — the no-self-sign invariant extends to the human who runs the fleet.

The Gate Council does not execute work and does not review every artefact line by line; the machine-checkable gate criteria (§6.1) exist precisely so the council accepts *accountability* for a verified fact set rather than re-deriving it. The council meets per gate (asynchronously for low-risk standard-path gates; synchronously for the P5 Go/No-Go and any waiver).

EXCEPTION & WAIVER HANDLING (TIME-BOXED)

No real estate clears every gate cleanly. Governance is judged by how exceptions are handled, not by whether they occur. VOSJ treats a waiver as a first-class, signed, **expiring** object — never a verbal override:

Element	Rule
Form	A waiver is a signed ledger row of its own kind, citing the gate criterion waived, the residual risk, the compensating control, and the remediation plan. It is subject to the same no-self-sign rule: the actor requesting the waiver cannot grant it.
Authority by risk	LOW: chair + risk reviewer. MEDIUM: + customer sponsor. HIGH / any safety-invariant criterion: + the customer's own governing body must counter-sign. **Invariants 1, 5, and 6 are non-waivable** — no waiver can permit a self-signed gate, a fail-open vault, or an unverified cutover. The engine refuses to mint a waiver against them.
Time-box (mandatory)	Every waiver carries an expiry (default ≤30 days, configurable down). On expiry the waived gate criterion *re-arms* and the dependent state is blocked until remediated or the waiver is explicitly re-issued by the same or higher authority. There is no perpetual waiver.
Visibility	Open waivers appear on the live board and in the 8-hourly governance digest; an aging waiver is a tracked risk, not a buried one.

CONTROL TESTING & EVIDENCE CADENCE

Controls that are never tested are assumptions. VOSJ runs a **continuous control-testing cadence** rather than a point-in-time audit, matching current SOC 2 expectations of system-generated evidence across the whole period:

Cadence	Test performed (automated unless noted)
Continuous (per transition)	Gate signature validity, signer-role correctness, and no-self-sign are *verified at write time* — a non-conforming transition cannot persist. Each such check is itself logged.
Per wave	Reconciliation pass completeness; rollback-rehearsal evidence present; baseline freshness within window; four-eyes validator ran on every dependent change.
8-hourly	Governance digest: open gates, open/aging waivers, blocked states, fleet actions since last digest, any zero-tolerance counters (self-signed gates, unverified

	cutovers, fail-open events) — all of which should read zero.
Per engagement	Second-line control walk-through: a sampled gate is traced deployment→approval→requester to confirm the ledger is sufficient evidence on its own (a CC8.1 dry-run).
Independent / periodic	Third-line (internal audit) re-derives the zero-tolerance counters directly from the exported ledger, independent of the operating team — the assurance step in the Three Lines Model.

KEEPING THE AUTONOMOUS FLEET UNDER BOARD-GRADE OVERSIGHT

The governing concern with an autonomous fleet is not that an agent does work — it is that the chain of authority could be captured by a non-human actor. VOSJ closes this structurally and procedurally:

- **Authority is withheld, not delegated.** Agents are minted without any `sign-as-<role>` capability (Invariant 1). The fleet can assemble a complete, verified evidence package; it cannot accept accountability for it. The human signer is the load-bearing oversight boundary, by construction.
- **Every fleet action is attributable.** One station = one identity (§10), so the board can answer "which actor did what, on what evidence, authorised by whom" for an autonomous action exactly as for a human one.
- **Irreversible steps require human escalation.** Cutover, decommission, and credential rotation are owner/admin-gated; a digital twin is instructed to escalate genuine owner-level decisions (irreversible actions, external commitments, spend) rather than proceed (§11.1).
- **The fleet supervisor is accountable but not self-assuring.** A named human owns fleet behaviour and reports to the Gate Council; that same human cannot sign off work the fleet performed — extending separation of duties to the operator of the autonomy itself.
- **Board reporting on a fixed cadence.** The 8-hourly digest and per-engagement control pack roll up into the customer's existing governing-body reporting, so autonomous execution surfaces in the same risk dashboard as the rest of the IT estate — visible, not parallel.

12.5 Accountability map — RACI for the key control activities

The RACI below assigns the control activities of a migration. `R` = Responsible (does the work), `A` = Accountable (single owner, signs), `C` = Consulted, `I` = Informed. Note the consistent pattern: **the fleet is Responsible, a human is Accountable** — the no-self-sign invariant rendered as an accountability rule. (FS = AI fleet supervisor; Risk = risk & control reviewer; IA = internal audit / third line.)

Control activity	Fleet	FS	Director (chair)	Sponsor	DBA	InfoSec	Risk	IA
Discovery / disposition assignment (7-R)	R	R	A	C	C	I	C	I
Runbook authoring	R	A	C	I	C	I	I	I
Independent rollback authoring	R	A	C	I	C	I	I	I

Four-eyes change validation	R	I	I	I	C	C	A	I
Landing-zone build & freeze	R	A	C	I	C	A	I	I
P5 Go/No-Go authorisation	R	C	A	C	C	C	C	I
Cutover execution	R	C	A	I	A	I	I	I
Reconciliation / verified-before-cutover	R	I	C	I	A	I	C	I
Contingency reversal (3-way)	R	C	A	C	A	C	C	I
Exception / waiver grant	I	C	A	C	C	C	A	I
Credential rotation / vault custody	I	R	C	I	I	A	C	I
Control testing (continuous & per-wave)	R	C	I	I	C	C	A	C
Independent assurance / ledger re-derivation	I	I	C	I	I	I	C	A
Decommission source	R	C	A	C	C	C	C	I
Board / governing-body reporting	R	C	A	C	I	I	C	C

Each activity has exactly one Accountable owner (two on cutover/reversal by design — the "two-key" control of §13.2). No row makes the entity that performs the work also accountable for approving it; where the fleet is Responsible, accountability always sits with a named human.

12.6 Governance as audit-grade, exportable evidence

The strongest property of running governance through the gate machine is that the evidence is produced as a by-product of operation, not assembled afterwards. For a customer POC heading to production, the deliverable is a self-contained, verifiable **governance evidence package** per migration:

- **The signed gate ledger** — every transition as an HMAC-SHA256, hash-chained row (actor, signer role, capability, evidence hashes, timestamp), with the signing key custodied outside the database (§14.4). An auditor can re-compute the chain and detect any back-dating, forgery, or gap without trusting the operator.
- **The waiver register** — every exception, its authority, residual risk, compensating control, expiry, and closure — demonstrating that deviations were governed, time-boxed, and never silent.
- **The reconciliation proofs** — per-workload equivalence evidence (counts, checksums, constraint re-validation, smoke results) tied by hash to the gate that consumed them.

- **The tool-call & order log** — the full record of what the fleet did through the MCP Hub (§14.2), so an autonomous action is as traceable as a human keystroke.
- **The framework binding** — the pinned template id/version (§8.2), so an auditor knows precisely which methodology, gates, and signer roles were in force for that run.

> **Mapped, not just stored.** The package exports with a *control-mapping manifest* that cross-references each evidence artefact to the framework objective it satisfies (ISO/IEC 38500 principle, COBIT objective, ITIL change record, SOC 2 CC8.1 element, SOX ITGC control). The customer's auditor receives evidence already indexed to their own control catalogue — the audit-prep effort that normally dominates a regulated migration becomes a file export. This is the regulated buyer's actual deliverable (§19), produced continuously rather than reconstructed from chat logs and email after the fact.

13 Verification & Reconciliation

The equivalence proof `π(w)` required before any cutover (§3.1, Invariant 6) is produced by a **reconciliation engine** that compares the migrated target against the source across several categories. A cutover gate clears only when all categories are within thresholds tuned by the database authority for the specific workload.

Reconciliation category	What it proves
Replication lag / in-flight rows (hard pre-switchover gate)	At the switchover instant, CDC/replication lag is zero and every in-flight / not-comparable row is explicitly resolved — so parity is measured against a settled target, not a moving one. Without this, a "pass" can still lose the final delta.
Row / object counts	No records were lost or duplicated in transfer.
Checksums / content hashes	Byte-level fidelity of migrated data and large objects (BLOBs).
Sequence / identity values	Auto-increment and identity state continues correctly post-cutover.
Constraint re-validation	Keys, foreign keys, and checks hold on the target.
Query-plan / performance parity (P6, post-cutover)	The target performs within the Examine baseline — evaluated *after* the P5 traffic switch, in the P6 30-minute revocable window, not as a P5 pre-switch blocker.
Application smoke checks	Critical user journeys succeed against the target before traffic is switched.

The replication-lag / in-flight gate and counts/checksums/constraints are **pre-switchover** conditions of the verified-before-Jump gate; performance/plan parity is assessed **post-cutover** in P6 within the revocable window.

13.1 Baseline discipline

Verification is meaningful only against a fresh baseline. VOSJ enforces a **baseline-drift guard**: a baseline older than a configured window automatically blocks the readiness gate and forces re-baselining. This prevents the common failure of validating a cutover against a stale picture of the source.

13.2 Revocable acceptance

Even after a reconciliation pass, the database authority may revoke acceptance within a short window if drift is detected — a deliberate "two-key" safety margin around the most consequential transition. Contingency reversal during execution requires a three-way authorisation, so an in-flight cutover cannot be unilaterally aborted or forced by any single actor.

> **Net effect.** The combination of fresh-baseline enforcement, multi-category reconciliation, revocable acceptance, and three-way contingency control means a VOSJ cutover is *provably* equivalent before traffic moves, and *reversible* if it is not — the two properties a high-stakes migration most needs.

14 Data Model & Implementation

VOSJ persists its system-of-record in a relational database with schema-per-domain isolation. The migration domain holds workloads, dependencies, dispositions, waves, runbooks, and evidence; the framework domain holds templates, phases, gates, and roles (§8.2); the agent domain holds MCP server configurations and tool-call logs.

14.1 The signed-gate state machine

The engine exposes three core operations — `getCurrentState`, `listValidNextStates`, and `signTransition` — that load allowed transitions and valid roles from the *bound template's* rows rather than from frozen constants. A transition is committed only as an HMAC-signed row; the RBAC capability required to sign is derived from the template-defined role. The verified-before-Jump gate is injected by the engine on the last Shift→Jump transition of every template and cannot be removed by a template author.

```
// Conceptual shape of a gate transition (illustrative)
signTransition({
  migrationId, fromState, toState,
  actor:       "persona:eng-migrator",      // who performed the work
  signerRole:  "dba",                        // who authorises (human, named role)
  capability:  "migration:gate:sign-as-dba", // minted only to humans in role
  evidence:    [ "recon-report#J1", "hash:9f3c…" ],
  ts:          "2026-06-21T22:14:07Z"
}) // => persists HMAC-SHA256(signed_row) to the tamper-evident ledger
```

14.2 MCP persistence

The MCP Hub stores server configurations (transport, command/URL, environment references to secrets, allow-lists, cached tool definitions, connection status) and a full tool-call log (server, tool, actor, arguments, result, duration). The tool-call log is the audit substrate referenced by Invariant 4 and requirement R8.

14.3 Engineering standards

The implementation follows strict platform rules that bound complexity and risk: all model calls go through a single LLM bridge (no direct provider SDK use); all database access through a facade (parameterised SQL only, never string concatenation); all user-generated content escaped against injection; uniform response envelopes; small files and functions; and per-tenant data isolation enforced on every query. These are not stylistic preferences but the controls that make an autonomous fleet's output reviewable and safe.

14.4 Durability & key custody of the ledger

Because the audit ledger is the artefact the factory's value rests on, it carries an explicit durability and key-custody posture. The PostgreSQL system-of-record runs with **quorum-based synchronous replication** (≥3 instances, documented failover / split-brain handling), plus continuous backup, WAL archiving, and point-in-time recovery, with a stated ledger RPO/RTO. Critically, the **HMAC signing key is custodied outside the database** (a KMS/HSM or the fail-closed vault of §15.2), so a database compromise cannot forge or re-sign rows, and rows are additionally hash-chained. Note that HMAC gives integrity against non-key-holders, not non-repudiation against an insider holding the key — which is exactly why the key is custodied externally.

15 Security & Compliance Architecture

15.1 Cross-platform access — no single primitive

There is no universal way to grant a third party scoped access to a customer's cloud; VOSJ uses each provider's sanctioned, revocable pattern, and never long-lived shared keys:

Target	Sanctioned access pattern
A major hyperscaler	Cross-account role assumption with an external ID.
Azure / Azure Local	Delegated management (Azure Lighthouse) with scoped roles.
Another major hyperscaler	Workload Identity Federation with service-account impersonation.

Each grant is scoped, auditable, and revocable; access is requested per engagement and withdrawn at Jump.

Delegated management has capability limits worth designing around: it cannot delegate an Owner role or data-plane (data-action) roles, and does not span national/sovereign cloud boundaries — material for a tool that needs broad write/transfer authority. Where delegation cannot carry the needed capability, VOSJ falls back to a scoped service principal / managed identity provisioned in the customer tenant. "Revocable" is real, but revocation propagation and in-flight-session behaviour are provider-specific.

15.2 Credential vault — fail-closed

Migration requires custody of powerful credentials. VOSJ stores them in an encrypted vault (authenticated encryption) whose master key is supplied operationally. The vault is **fail-closed** (Invariant 5): with no master key present it refuses to operate rather than falling back to a default development key. (The fail-open fallback is a classic and dangerous anti-pattern; eliminating it is an explicit hardening requirement.)

15.3 The self-serve signup constraint

> **An honest constraint, stated plainly.** No major cloud permits fully-programmatic creation of a *net-new* customer account plus payment method; an attested human signup/KYC step is unavoidable for the first billing relationship (only subordinate accounts/subscriptions are API-creatable under an already-attested parent). VOSJ therefore offers two shapes: **(A) hosted-redirect signup** — the customer completes a one-time attested signup on the provider, then grants VOSJ scoped access; and **(B) reseller** — the customer signs under VOSJ's billing (e.g. a cloud-solution-provider channel), giving true one-click self-serve plus consumption margin, at the cost of carrying reseller/PCI/KYC compliance.

15.4 Data protection

Moving data across regions or clouds is itself a transfer event with legal weight. VOSJ treats data handling as a sub-processor responsibility: a data-processing agreement and sub-processor disclosure, standard contractual clauses for cross-border moves, explicit data-loss liability with rollback gates and verified-copy guarantees, and legal sign-off before launch. Every access grant and data movement is recorded in the audit log.

15.5 Threat model (abbreviated)

Threat	Mitigation
A compromised or misbehaving agent attempts an unauthorised action	Per-tool RBAC + allow-lists bound capability; no `sign-as` grant; full tool-call audit.
An agent tries to authorise its own work	Invariant 1 (no self-sign) — structurally impossible.
Credential leakage	Secret indirection; rotation; fail-closed vault; no plaintext in config or agent context.
Silent data loss in transfer	Verified-before-cutover reconciliation; revocable acceptance; baseline-drift guard.
Forged or back-dated history	HMAC-signed, hash-chained ledger with the signing key custodied outside the database (§14.4).
Concurrent agents corrupting shared state	One station = one identity = one clone; commit-via-PR only.
Indirect / direct prompt injection via migrated content or tool responses (OWASP LLM01:2025)	Treat all source content, tickets, repositories *and tool responses* as untrusted ingress: content/provenance separation, response sanitisation, and behavioural monitoring of tool-call sequences against the runbook.
Tool poisoning / rug-pull / cross-server shadowing	Pin and verify external MCP-server identity; allow-list *signed* servers; statically validate server-provided tool metadata before exposing it to the model; alert on post-approval descriptor drift; isolate servers so one cannot shadow another.
Confused deputy / token pass-through	Validate token audience (RFC 8707); never forward a client token upstream; mint downstream-scoped tokens at the Hub.
The "lethal trifecta" (private data + untrusted input + exfiltration path)	Break at least one leg per station: constrain egress, separate untrusted-content contexts from credentialed actions, and keep the human-signed gate on irreversible steps.
Container / runtime escape from a code-executing station	Hardened runtime class (gVisor / Kata / Firecracker) + brokered short-lived credentials (§10.4) bound node and neighbour-pod blast radius.

Multi-agent collusion / correlated error	Diversify reviewer/rollback agent model + context from the builder's; the human signer is the true independence boundary (§22).

15.6 Consolidated Threat Model (STRIDE × Agent/MCP/LLM classes)

§15.5 enumerated the abbreviated mitigation table; this subsection promotes it to a **deployment-grade, reviewable threat model** that a customer security team can audit line by line. We model VOSJ on two axes at once: the classical **STRIDE** categories (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) for the conventional surface, and the **agentic/LLM threat classes** — OWASP Top 10 for LLM Applications (2025), the OWASP Agentic AI "Threats & Mitigations" taxonomy (Agentic Security Initiative), and the CSA MAESTRO seven-layer model — for the autonomous-fleet surface that conventional STRIDE underserves. Each row maps a threat to the concrete VOSJ control and to a recognised framework reference.

> **How to read the references.** `STRIDE` = the classical category; `LLMnn:2025` = OWASP Top 10 for LLM Applications 2025; `ASI Tn` = OWASP Agentic AI threat id (T1 Memory Poisoning, T2 Tool Misuse, T3 Privilege Compromise, …); `MAESTRO Ln` = the affected MAESTRO layer (L1 Foundation Models … L7 Agent Ecosystem); `ATLAS` / `ATT&CK` = MITRE knowledge-base technique families; `CSF` = NIST CSF 2.0 function.

Threat (STRIDE & agentic class)	VOSJ control	Framework reference
S — Agent/operator **identity spoofing**; a rogue station impersonating a trusted one	One station = one identity = one clone (§10); per-identity service accounts; SPIFFE/SPIRE-style workload identity + OAuth 2.1 audience-bound tokens (RFC 8707) so a token cannot be replayed against another server; mutual auth on the order queue claim.	STRIDE Spoofing; ASI identity-spoofing/impersonation; MAESTRO L7; ATLAS Impersonation; CSF PR.AA (identity & access).
T — **Tampering** with the gate ledger, runbook, or migrated data in flight	HMAC-SHA256-signed, hash-chained gate ledger with the signing key custodied in an external KMS/HSM (§14.4); parameterised SQL only; verified-before-cutover reconciliation (checksums, counts, constraint re-validation, §13).	STRIDE Tampering; ISO 27001:2022 A.8.24 (cryptography); CSF PR.DS (data security); ATT&CK Data-Manipulation.
R — **Repudiation**:	Every station transition, tool call, access grant, and cutover	STRIDE Repudiation; ISO 27001:2022 A.8.15 (logging), A.8.16 (monitoring);

"who advanced/cut-over this workload, on what evidence?"	logged immutably (actor / evidence / timestamp / source→target); externally-keyed HMAC gives integrity against non-key-holders; exportable audit package is a deliverable (§12.2).	CSF PR.PS / DE.AE; SOC 2 CC7.
I — **Information disclosure**: credential leakage; sensitive-data exfiltration; system-prompt leakage	Fail-closed encrypted vault (§15.2); secret indirection — servers reference, never embed, secrets (§9); broker vends short-lived, audience-scoped, per-task credentials, no long-lived secret in the execution pod (§10.4); default-deny egress (§15.8); prompt/secret separation.	STRIDE Info-disclosure; LLM02:2025 (Sensitive Info Disclosure), LLM07:2025 (System-Prompt Leakage); ISO 27001:2022 A.8.12 (DLP); CSF PR.DS.
D — **Denial of service** / **unbounded consumption**: a looping agent exhausts compute or model quota	Per-tool timeouts + child-process isolation (§9.1); durable-queue back-pressure and atomic claim (§10); fair-use ceilings → queue/throttle/metered fallback (§22); resource quotas + `RuntimeClass` limits per station.	STRIDE DoS; LLM10:2025 (Unbounded Consumption); CSF PR.IR (resilience), DE.CM; CIS Controls v8 #8 (audit-log mgmt) / resource limits.
E — **Elevation of privilege**: agent self-authorises a gate, or escalates via excessive agency	Invariant 1 (no self-sign — identities minted without any `sign-as` capability); per-tool RBAC + signed-server allow-lists at the Hub; least-privilege capability grants (§12.1); human-signed gate on irreversible steps.	STRIDE EoP; LLM06:2025 (Excessive Agency); ASI T3 (Privilege Compromise); CSF PR.AA; ISO 27001:2022 A.8.2 (privileged access).
Indirect / direct prompt injection via migrated source content, tickets, repos, or tool responses	Treat *all* source content and tool responses as untrusted ingress; content/provenance separation; response sanitisation; behavioural monitoring of tool-call sequences against the approved runbook (deviation alerts).	LLM01:2025 (Prompt Injection); LLM05:2025 (Improper Output Handling); ASI T2; MAESTRO L2/L4; ATLAS Prompt-Injection/RAG-Poisoning; CSF PR.PS / DE.CM.

Tool poisoning / rug-pull / cross-server shadowing from an external MCP server	Pin and verify external MCP-server identity; allow-list *signed* servers; statically validate server-provided tool metadata before exposing it to the model; alert on post-approval descriptor drift; isolate servers so one cannot shadow another (§15.5).	LLM03:2025 (Supply Chain); ASI T2; MAESTRO L3/L7; ATLAS AI-Supply-Chain-Compromise; CSF ID.SC (supply chain), GV.SC.
Memory / context poisoning of an agent's persistent state or the learning loop	Validate memory content; session isolation per task; the self-improvement corpus (§11.3) is scrubbed and provenance-tagged before ingest; reviewer/rollback agent diversified in model + context from the builder (§22).	ASI T1 (Memory Poisoning); LLM04:2025 (Data/Model Poisoning); MAESTRO L2; ATLAS RAG-Poisoning; CSF PR.DS / DE.AE.
Confused deputy / token pass-through at the MCP Hub	Validate token audience (RFC 8707); never forward a client token upstream; mint downstream-scoped tokens at the Hub; OAuth 2.1 Resource-Server model with PKCE and Protected Resource Metadata (RFC 9728).	STRIDE EoP; ASI T3; OAuth 2.1 / RFC 8707 / RFC 9728; CSF PR.AA.
The "lethal trifecta" — private data + untrusted input + an exfiltration path in one context	Break at least one leg per station: constrain egress (default-deny); separate untrusted-content contexts from credentialed actions; keep the human-signed gate on irreversible steps so no single poisoned context can both decide and exfiltrate.	LLM01/LLM02:2025; ASI T2/T3; MAESTRO L4/L6; CSF PR.DS / PR.IR.
Container / runtime escape from a code-executing station	Hardened runtime class (gVisor / Kata / Firecracker-microVM via `RuntimeClass`); node-pool isolation; brokered short-lived credentials bound the node and neighbour-pod blast radius (§10.4).	STRIDE EoP; ISO 27001:2022 A.8.31 (separation of environments); CIS Kubernetes Benchmark (Pod Security, RuntimeClass); CSF PR.PS.
Multi-agent collusion / correlated error / cascading failure	Diversify reviewer/rollback agent model + context from the builder; machine-checkable gate criteria so the human signer is not asked to rubber-stamp an agent-	ASI rogue-agent / cascading-failure; MAESTRO L7; LLM09:2025 (Misinformation); CSF DE.AE / RS.AN.

	assembled narrative; the human-signed gate is the true independence boundary (§22).	

15.7 Control Framework Mapping (NIST CSF 2.0 & ISO/IEC 27001:2022)

Customer security teams do not buy controls — they buy *coverage against a framework they already report on*. The table below maps VOSJ's structural controls onto the six functions of the **NIST Cybersecurity Framework 2.0** (Govern, Identify, Protect, Detect, Respond, Recover) and onto the relevant theme(s) of **ISO/IEC 27001:2022 Annex A** (Organizational / People / Physical / Technological; 93 controls). It is the artefact to drop into a vendor-security questionnaire or a customer's GRC tool.

CSF 2.0 function	VOSJ control evidence	ISO/IEC 27001:2022 Annex A anchor
GOVERN (GV)	Safety invariants enforced in the engine, not policy (§12); change-management + CD closure for every task; documented autonomy posture (humans hold gates); DPA / sub-processor disclosure and legal sign-off before launch (§15.4); risk register maintained per engagement.	Organizational: A.5.1 (policies), A.5.2 (roles & responsibilities), A.5.8 (security in project mgmt), A.5.19–A.5.23 (supplier & cloud-service security), A.5.31 (legal/regulatory).
IDENTIFY (ID)	Application-centric inventory + dependency DAG (Vault, §5); CI/CD 365° delivery-system fingerprint (§17); per-workload 7-R disposition + risk/effort/TCO; external MCP-server inventory with signed allow-list; supply-chain provenance for executors and connectors.	Organizational: A.5.9 (inventory of assets), A.5.21 (ICT supply chain), A.5.7 (threat intelligence). Technological: A.8.8 (technical vulnerability mgmt).
PROTECT (PR)	Capability-based RBAC, requireAuth/requireCapability on every route/mutation (§12.1); no-self-sign (Inv. 1); fail-closed vault (§15.2); secret brokering + short-lived audience-scoped tokens (§10.4); hardened `RuntimeClass` + node-pool isolation; default-deny egress (§15.8); encryption in transit and at rest; DLP / content separation against injection.	Technological: A.8.2 (privileged access), A.8.3 (information access restriction), A.8.5 (secure auth), A.8.12 (DLP), A.8.24 (cryptography), A.8.31 (separation of environments). People: A.6.3 (awareness — for human signers/operators).
DETECT (DE)	Immutable tool-call log (server/tool/actor/args/result/duration, §14.2); behavioural monitoring of tool-call sequences vs the approved runbook; baseline-drift guard (§13.1); descriptor-drift alerts on MCP servers; continuous contingency monitoring during P5 cutover.	Technological: A.8.15 (logging), A.8.16 (monitoring activities), A.8.6 (capacity), A.8.7 (malware). Organizational: A.5.25 (assessment & decision on security events).
RESPOND (RS)	Three-way contingency reversal during cutover (§13.2); revocable acceptance within the P6 window; fleet keepalive + graceful drain (§10.2); incident runbook with credential-revocation and station-quarantine steps (§15.10); per-station kill-switch and queue freeze.	Organizational: A.5.24 (incident-management planning), A.5.26 (response to incidents), A.5.27 (learning from incidents), A.5.5 (contact with authorities).

RECOVER (RC)	Verified-before-cutover + independent rehearsed rollback (Inv. 6, §6.1); quorum synchronous PostgreSQL replication, WAL archiving, PITR, stated ledger RPO/RTO (§14.4); source kept serving until equivalence proof holds; post-mortem feeds the template (§11.3).	Organizational: A.5.29 (security during disruption), A.5.30 (ICT readiness for continuity). Technological: A.8.13 (information backup), A.8.14 (redundancy).

For SOC 2 readiness the same evidence supports the Trust Services Criteria: the audit ledger and RBAC serve **Security (CC-series, mandatory)**; HA/DR and keepalive serve **Availability**; multi-category reconciliation serves **Processing Integrity**; the fail-closed vault and least-privilege serve **Confidentiality**; and the DPA / cross-border-transfer posture (§15.4) plus PII handling serve **Privacy** (and ISO/IEC 27018 for cloud PII).

15.8 Secure Deployment into a Customer Environment

This subsection is the deployment contract for taking VOSJ from POC into a *real* customer estate. It is written to the **zero-trust** tenets of NIST SP 800-207 (never trust, always verify; least privilege; assume breach) and to the customer's likely Azure / Azure Local landing zone.

IDENTITY & LEAST PRIVILEGE

- **Per-engagement, scoped, revocable grants only** — into the customer cloud via the provider's sanctioned delegated pattern (Azure: delegated management with scoped roles; cross-cloud: role assumption / workload-identity federation, §15.1). No long-lived shared keys. Access is requested at kickoff and **withdrawn at Jump**.
- **Just-enough, just-in-time.** Standing privilege is minimised; cutover, decommission, and credential rotation are owner/admin-gated and time-boxed. Where delegation cannot carry a needed data-plane or Owner capability, fall back to a *scoped* service principal / managed identity provisioned in the customer tenant (§15.1).
- **Workload identity for the fleet.** Each station authenticates with a SPIFFE/SPIRE-style or federated-OIDC workload identity; model and source-control credentials are short-TTL and per-task (§10.4).

NETWORK & EGRESS

- **Default-deny egress** on every station namespace, with an explicit allow-list to (i) the approved migration targets, (ii) the model endpoint, (iii) source control, and (iv) cluster DNS — per the CIS Kubernetes Benchmark network-policy guidance. Egress control is the load-bearing leg that breaks the “lethal trifecta”.
- **Private connectivity** to customer data planes (private endpoints / private link / VPN or ExpressRoute-class circuits) rather than public-internet paths; the Shift transfer runs *on the source host* via the sanctioned remote-management transport (§16.2), not a gateway-local copy.
- **Micro-segmentation** between fleet roles (architect / builder / reviewer / migrator / deployer) so a compromised builder station cannot reach a reviewer's or deployer's surface.

SECRETS & KEY CUSTODY

- **Fail-closed vault** (§15.2): no master key → refuse to operate (never a default dev key). Master key in a KMS/HSM; documented rotation and split-knowledge custody.
- **HMAC ledger key custodied outside the database** (§14.4) so a DB compromise cannot forge or re-sign gate rows.

- **No plaintext secrets** in config, image, or agent context; MCP servers reference secrets resolved at connect time; rotation reconnects transparently.

DATA RESIDENCY & SOVEREIGNTY

- Cross-region / cross-cloud data movement is a **transfer event with legal weight**: DPA + sub-processor disclosure, standard contractual clauses for cross-border moves, explicit data-loss liability with rollback gates and verified-copy guarantees, legal sign-off before launch (§15.4).
- **Region/sovereign pinning** is configuration: targets, regions, and the model endpoint are config-driven so an engagement can be constrained to a single jurisdiction or a sovereign cloud boundary (note delegated management does *not* span national/sovereign boundaries, §15.1).
- Every access grant and data movement is recorded in the audit log as residency evidence.

TENANCY ISOLATION

- **Schema-per-domain + per-tenant `tenant_id` filter on every query** (§14.3); framework templates and the audit ledger are tenant-scoped (`visibility: public|tenant|private`, §8.2).
- **Compute isolation**: dedicated node pools / `RuntimeClass` sandboxes per engagement where the customer's risk class requires hard multi-tenancy; one station = one clone = one identity (§10).

LOGGING, MONITORING & DETECTION

- Ship the immutable tool-call log, gate-ledger, and station activity to the customer's SIEM (or a shared sink); map detections to **MITRE ATT&CK** (infra) and **MITRE ATLAS** (AI-specific: prompt injection, RAG poisoning, impersonation).
- **Behavioural baselining** of each station's tool-call sequence against its approved runbook; alert on deviation, on descriptor drift of an external MCP server, and on baseline-drift of the source estate (§13.1).
- Detection content covers the agentic classes explicitly (excessive-agency calls, anomalous egress, memory/context anomalies) — not only conventional host signals.

INCIDENT RESPONSE

- **Containment primitives built in**: per-station kill-switch, order-queue freeze, instant credential revocation (short-TTL tokens expire on their own), and station quarantine via the hardened runtime boundary.
- Because authority to approve is structurally withheld from agents (Inv. 1), a compromised station can be contained *without* any in-flight gate having been illegitimately signed; the human-signed gate is the backstop for irreversible steps.
- IR plan aligned to CSF RESPOND (RS) and ISO A.5.24–A.5.27: declare, contain, eradicate, recover (revocable acceptance / three-way reversal, §13.2), and feed lessons to the template (§11.3).

15.9 Defense-in-Depth for the Autonomous Fleet

The fleet is the paper's novelty and therefore its largest attack surface. We defend it in concentric layers, mapped to the CSA **MAESTRO** seven-layer model so a reviewer can see that no layer is unguarded. The principle: an attacker must defeat *several independent* layers, and the outermost guarantee — that no agent can authorise its own work — holds even if inner layers fail.

Layer (MAESTRO)	Primary risks	VOSJ defenses
L1 Foundation model	Jailbreak; misinformation; correlated error across same-model agents.	Single LLM bridge (no direct SDK); model-tier-by-role; reviewer/rollback agent diversified in model + context (§22).
L2 Data operations	Memory/context poisoning; RAG poisoning; injected source content.	Untrusted-ingress treatment of all source/tool content; provenance-tagged, scrubbed learning corpus; session isolation (ASI T1; LLM04).
L3 Agent frameworks / tools	Tool poisoning / rug-pull / shadowing; excessive agency.	Signed-server allow-list; static tool-metadata validation; per-tool RBAC at the Hub; drift alerts (LLM03/LLM06; ASI T2).
L4 Deployment & infrastructure	Container/runtime escape; credential theft; lateral movement.	Hardened `RuntimeClass` (gVisor/Kata/Firecracker); brokered short-lived creds; default-deny egress; micro-segmentation (CIS K8s Benchmark).
L5 Evaluation & observability	Blind spots; tampered or missing audit.	Immutable, externally-keyed tool-call + gate ledger; behavioural baselining; SIEM export mapped to ATT&CK/ATLAS (CSF DE).
L6 Security & compliance	Self-authorisation; privilege creep; confused deputy.	No-self-sign (Inv. 1); least-privilege grants; audience-bound tokens (RFC 8707); four-eyes change validation (Inv. 3).
L7 Agent ecosystem	Identity spoofing; rogue agent; multi-agent collusion / cascading failure.	One station = one identity = one clone; separation of authoring & authorising (Inv. 2); human-signed gate as the true independence boundary; per-station kill-switch.

> **The load-bearing backstop.** Layers L1–L5 reduce the *likelihood* of agent compromise; L6–L7 bound its *consequence*. Even granting an attacker full control of a station, the no-self-sign invariant and the human-signed gate mean the irreversible steps — cutover, decommission, credential rotation — remain unreachable without an independent human authoriser acting on machine-checkable evidence. Autonomy is safe precisely because the authority to approve it is structurally withheld.

15.10 Residual Risk & Acceptance

No architecture is residual-risk-free. We name what remains so the customer accepts it knowingly: (i) a **plausible-but-wrong evidence package** from collusion/correlated error that misleads a human signer — mitigated, not eliminated, by machine-checkable gate criteria and reviewer/model diversity (§22); (ii) **novel MCP/agent attack classes** emerging faster than the threat model — mitigated by treating §15.6 as a living document revisited each release against OWASP/CSA/MITRE updates; (iii) **provider-specific revocation propagation** — in-flight-session behaviour after a grant is revoked is provider-dependent (§15.1) and must be verified per target. These are tracked in the engagement risk register (CSF GV.RM) and re-reviewed at each phase gate.

16 Connectivity & Provider Integration

VOSJ is deliberately **dual-connectivity**, so coverage is not limited to platforms that expose a migration API.

Mode	How it works	When used
Mode 1 — API orchestration	VOSJ drives API-strong first-party engines (e.g. Microsoft's Azure Migrate / Azure Site Recovery and database-migration service, and the equivalent replication and database-migration services on other public clouds) and API-driven third-party ISVs headlessly from its control plane.	Whenever a programmatic interface exists — the preferred, fully-automatable path.
Mode 2 — dashboard-upload / resold-SaaS	For console-driven tools with no orchestration API, VOSJ pushes inventory/data into the vendor SaaS and tracks status; reselling the SaaS is an option for fast breadth.	Where no API exists, or where reselling is the faster route to coverage.

16.1 Designing around durable pillars

First-party tool naming and availability churn (services are renamed, retired, or restricted to existing customers on the vendors' cadence). VOSJ insulates buyers by architecting on the *durable, API-drivable* primitives — block/disk replication, image conversion, and database migration services — and absorbing churn behind a stable control-plane abstraction. Provider connectors are registered in a catalog, so adding or swapping a provider is a configuration and connector task, not a redesign.

16.2 The existing executor substrate

VOSJ orchestrates a real library of point-to-point executors spanning the common source/target pairs (on-prem virtualization and other public clouds into Microsoft Azure, Azure Local, and Hyper-V; and between the Microsoft platforms — plus a generic application-migration path). Each executor maintains its own internal state machine (`draft → validated → executing → completed | failed → rolling-back → rolled-back`) with pre-flight checks that the station conductor invokes through a bridge but never bypasses. The Shift station's transfer steps run *on the source host* via the appropriate remote-management transport, rather than via a gateway-local copy, to keep the data path correct and efficient.

17 CI/CD & DevOps 365° Assessment

Replatforming and modernization are unsafe without a delivery system to receive the modernised workload. VOSJ therefore includes a **companion assessment** that closes the customer's CI/CD loop end-to-end — scan → build → test → quality gates → security → release → deploy → provision/promote → operate → observe → feedback — scored against a weighted master checklist, a five-level maturity model, and DORA targets.

17.1 How it plugs into the stations

Station	Role of the assessment
Vault	The scan runs here, fingerprinting the delivery system alongside the workload bill-of-materials.
Orchestrate	Gaps become roadmap items; *Replatform/Modernize dispositions are gated on closing them* (you cannot modernise a workload without standing up its pipeline, quality gate, GitOps flow, and observability).
Shift	The roadmap is executed incrementally (Strangler-Fig).
Jump	The closed-loop blueprint is delivered as the operating model; the verified-before-Jump gate is itself a quality-gate/DORA control.

17.2 Two archetypes

- **The VM→container shop with no DevOps.** Roadmap builds the missing pipeline, quality gate, and observability before any Refactor work begins.
- **The greenfield-no-blueprint customer.** Roadmap establishes the standard delivery loop as the operating model from the outset.

The autonomous fleet auto-collects most of the scorecard evidence via provider/registry scanning, and the remediation agents (bug-crusher / healing-agent class) close the flagged gaps — another instance of the AI substrate doing the toil that makes assessments expensive.

18 Economics & Value Model

VOSJ's economics follow from its architecture. Because tooling sits on durable first-party primitives and the labour is an autonomous fleet on owner compute and flat-rate model subscriptions, the cost of goods per migration is low and largely fixed; value is captured at three layers:

Value source	Mechanism
Partner funding	Cloud migration-acceleration programs co-fund qualifying engagements (terms vary by provider and change over time — verify per engagement).
Consumption margin	Under the reseller shape (§15.3 B), the operator earns margin on the customer's ongoing platform consumption.
Service fee	A per-migration or subscription fee for the factory itself — assessment, orchestration, verified cutover, and the audit evidence package.

> **The structural advantage.** A human consultancy's cost scales with engineer-hours; its margin is squeezed by the very scarcity that makes migration expensive. A factory whose labour is autonomous inverts this: marginal throughput costs compute, and the learning loop (§11.3) lowers cost per migration as volume grows. The constraint shifts from "how many engineers can we hire" to "how much work can we safely review and authorise" — a far more favourable scaling law.

Any specific cost-reduction or capacity figures that may circulate in case studies (for example claims of large percentage cost reductions or multi-million-dollar savings) are third-party, illustrative, and must be independently re-verified before being presented to a customer.

19 Worked Reference Scenario

To make the model concrete, consider a mid-market enterprise with ~180 applications on an on-premises virtualization estate that must exit its data centre within nine months, moving most workloads to a public cloud and modernising a customer-facing monolith.

Phase	What happens in the factory	AI substrate role
P1 Envision (V)	TCO/value case; readiness assessment; target platform chosen; sponsor named.	Discovery agents draft the readiness assessment; a digital twin prepares the business case for human review.
P2 Examine (V)	Automated discovery builds the inventory + dependency DAG; baselines captured; a 7-R disposition assigned to all 180 apps; waves grouped. *Gate cannot pass until every app has a disposition.*	Assessment agents fingerprint the estate and the delivery system; propose dispositions; humans review and the IT lead signs.
P3 Engineer (O)	Transition architecture; per-wave cited runbooks; an *independent* rollback runbook; static tabletops; CRs through four-eyes validation.	One agent authors the runbook; a *separate* agent authors the rollback; a reviewer agent validates — separation of authoring/authorising in action.
P4 Readiness (O)	Landing zone built and validated; baseline-drift check; freeze; vendor cases verified.	Builder/deployer agents stand up the landing zone as code; the drift guard auto-blocks if baselines are stale.
P5 Shift (S)	Full-panel go/no-go; waves cut over Strangler-Fig with parallel run; the monolith's Refactor disposition forces incremental proxy-switch; evidence hashed per step.	Migrator agents call the virtualization executors via the bridge on the source host; contingency monitoring runs continuously.
P6 Verify (J)	Reconciliation across all categories vs the fresh baseline; performance parity confirmed; DBA signs (revocable 30 min).	Verification agents run reconciliation and produce the signed report that the human authority accepts.
P7 Jump (J)	Per-app BAU sign-off; source decommissioned after retention; post-mortem; lessons fed back to the template.	Lessons analyst agent drafts template-improvement deltas; the agents' learning loop ingests the engagement.

> **Outcome shape.** The data-centre exit proceeds wave by wave, each wave verified before its traffic moves and reversible until it is. The customer receives a running modern estate *and* an exportable audit package proving every cutover was performed under separation

> of duties against a fresh baseline — the deliverable a regulated buyer actually needs, produced as a by-product of the factory rather than assembled afterwards.

20 Evaluation Methodology

A claim of "factory throughput with audit-grade governance" must be falsifiable. We propose the following evaluation dimensions and metrics; we report the methodology rather than headline numbers, which depend on estate and configuration.

Dimension	Metric	Interpretation
Throughput	Workloads cut over per unit time per unit of human supervision.	The factory's core efficiency — isolates the autonomy gain from raw compute.
Safety	Fraction of cutovers with a complete verified-before-cutover proof; count of unverified cutovers (target: zero, structurally).	Tests Invariant 6 empirically.
Governance integrity	Fraction of gate transitions with a valid signature by a correctly-separated authoriser; count of self-signed gates (target: zero).	Tests Invariants 1–2.
Reversibility	Fraction of waves with a rehearsed independent rollback; mean time to roll back in tabletop.	Tests the rollback discipline.
Reconciliation rigor	Defect-escape rate post-cutover (issues found after Jump that reconciliation should have caught).	Tests the verification engine's coverage.
Delivery maturity	DORA metrics (deployment frequency, lead time, change-failure rate, MTTR) before vs after.	Tests the CI/CD 365° outcome (§17).
Learning curve	Cost and cycle time per migration as cumulative volume grows.	Tests the self-improvement loop (§11.3).

The governance and safety metrics are the most important: a factory that is fast but occasionally cuts over unverified, or lets an agent authorise its own work, has failed its central design goal regardless of throughput. Because those properties are enforced structurally, the expected count of violations is zero — any non-zero result is a defect in the engine, not in operator discipline, and is therefore directly fixable.

21 Novelty & Inventive Claims

This section enumerates the inventive concepts disclosed by this paper. It is offered as a **defensive publication** — a public, dated, technical disclosure establishing prior art for the combinations below. The individual ingredients (migration frameworks, the 7-R model, Strangler-Fig, MCP, tool-using agents) are known; the claims concern their *specific combination and operationalisation*. To operate as prior art, a defensive publication must be publicly deposited with a verifiable timestamp in an examiner-searchable venue; the intended deposit channel and date are recorded in Appendix E.

> **Claim 1.** A migration system in which the *brand stations* (Vault–Orchestrate–Shift–Jump) are simultaneously the user-visible product structure and the internal state-machine stages, each observable, audited, and metered.

> **Claim 2.** A *data-driven framework template engine* that compiles an arbitrary, consultant-authored methodology (phases, gates, roles) into a single signed phase-gate state machine, reusing one governance core across all methodologies, with version pinning per migration and one-to-one backward-compatible re-seeding of a prior hardcoded methodology.

> **Claim 3.** Treatment of the *7-R disposition as a gate input* such that a kickoff gate cannot clear until every in-scope workload carries a disposition, and the disposition deterministically selects the runbook template and executor, making big-bang cutover *structurally unavailable* for high-risk dispositions.

> **Claim 4.** A *composition for governed autonomous work dispatch*: a database-backed durable work queue for order dispatch (atomic claim; status/evidence return) bound to governed *tool* access via a Model Context Protocol Hub — decoupling the issuer of migration work from the identity-isolated station that executes it, with credential brokering, per-tool RBAC, and full tool-call audit. (Order dispatch is the queue's role, not a property of MCP.)

> **Claim 5.** A *fleet of per-identity autonomous coding agents* ("devstations"), each with its own account, repository clone, brokered model seat, and source-control credential, running under a *hardened runtime class* and executing migration toil with role specialisation that enforces separation of authoring and authorising; with one-station-one-clone isolation as a correctness invariant, *minimised standing credentials* (per-identity segmentation), and a keepalive controller for resilience.

Claim 6. An *AI digital-twin operating model* in which autonomous personas, configured with a principal's standards, run the factory off-hours through the same MCP channel and the same gates as humans, escalating only owner-level decisions — yielding continuous availability without unsupervised authority.

Claim 7. The *no-self-sign invariant* realised by minting agent service identities without any gate-signing capability, so that an autonomous actor can perform all work toward a gate but is structurally incapable of authorising the transition.

Claim 8. A *verified-before-cutover gate injected by the engine* on the final Shift→Jump transition of every template (non-removable by template authors), backed by multi-category reconciliation, fresh-baseline enforcement, revocable acceptance, and three-way contingency control.

Claim 9. A *self-improvement loop* in which each migration's post-mortem produces template deltas and feeds an agent learning process from captured activity data, lowering cost and cycle time per migration as cumulative volume grows.

Claim 10. A *dual-connectivity factory* that uniformly drives both API-orchestrated first-party/ISV engines and console-driven (dashboard-upload / resold-SaaS) tools behind one control-plane abstraction designed around durable, churn-resistant primitives.

The combinations in Claims 4–9 — an MCP order/tool channel commanding an identity-isolated autonomous fleet that performs migration work under a signed-gate engine with a structural no-self-sign property and an engine-injected verified-before-cutover gate — constitute the paper's primary contribution and the intended subject of this defensive disclosure; §21.1 calibrates each claim against real prior art.

21.1 Differentiation against prior art (calibrated)

This subsection situates each disclosed claim against the closest art we retrieved, and assigns a *calibrated* defensibility after independent adversarial verification of every differentiation. We state this honestly because the intended audience — the holders of the workflow-FSM, durable-execution, and audited-authorization patents cited below — hold directly overlapping art, much of it published by the major platform vendors themselves. **No individual claim is strong in isolation.** Several are *anticipated* when asserted standalone and are recorded here as "none" precisely so this paper cannot be read as overclaiming. The contribution we actually assert is the *narrow combination* (§21, Claims 4–9), not any component standing alone.

> **Posture.** This is a *defensive publication*. We assert prior art over one specific composition; we do **not** assert novelty over any component standing alone, and we explicitly abandon as standalone-novel the claims marked "none" below.

VOSJ claim	Closest existing art (retrieved)	How VOSJ differs — and what survives verification	Defensibility (calibrated)
1. Brand stations *are* the FSM stages — observable, audited, metered.	Agentic-workflow approval-gate frameworks (named gate-states + mandated audit schema + decision-time/SLA metering); per-stage agent "mission-control" surfaces; SDLC "station" products; US 7,607,130 (per-step permission-checked, audited state machine).	The trio "observable + audited + metered named FSM stages" is fully anticipated — it appears in a single approval-gate reference. VOSJ's only daylight is that the brand letters are 1:1 identified with the FSM stages (the product surface *is* the state graph). That is a packaging identity, not a mechanism. Contributing element of the combination only.	`weak`

2. Data-driven engine compiling any methodology into one signed phase-gate FSM.	US 7,689,947 (data-driven, runtime-mutable FSM engine); US 10,474,506 ("a workflow template may represent a finite state machine"); US 7,607,130 (definition-table → permission-checked FSM); mature OSS process engines.	**Anticipated standalone.** "Template/spec compiled into a gated FSM" is squarely covered. The only residue is *cryptographic signing* of the compiled FSM — routine artifact-signing, and captured more precisely by Claims 7/8. Do not assert standalone; fold signing + the engine-injected gate into the Claim 7/8 combination.	`none (standalone)`
3. 7-R disposition as a gate input that makes big-bang structurally unavailable.	The 7-R taxonomy itself (industry/Microsoft/other vendors); wave/move-group planning that forces co-dependent units into one wave; the Strangler-Fig pattern (published by Microsoft); US 9,836,712 (per-instance "migrateable state" that grants/prevents cutover).	VOSJ owns nothing in the taxonomy, and the purpose (avoid big-bang via phased/wave cutover) is productized and pattern-documented. The asserted residue — "structural unavailability" — is an emergent property of grant-only-when-migrateable gating that US 9,836,712 already recites, recast in FSM vocabulary. A wave planner that refuses an all-at-once wave is functionally equivalent. **Anticipated.**	`none (standalone)`
4. DB-backed durable order queue + MCP-governed tools, decoupling issuer from an identity-isolated executor.	Durable-execution platforms + US 9,098,359 (workers poll a queue; durable decoupling of issuer from executor); MCP as the open governed tool standard with multiple gateways; managed agent runtimes that give each session an isolated micro-VM with its own credentials behind governed MCP tools.	Each half is mature, productized, and (durable queue) patented; the "identity-isolated executor distinct from the issuer behind governed MCP" pattern is now a named, commodity managed-runtime feature. The residual	`weak`

		differentiator is binding this already-assembled stack specifically into a *migration* FSM — a thin domain wrapper. Defensible only as part of the combination.	
5. Fleet of per-identity autonomous coding agents under a hardened runtime, with authoring/authorising separated.	Per-agent-identity governed coding fleets "governed like human contributors"; persona-role sandboxed agent fleets; per-task isolated runtimes. **Separation of duties is published verbatim** in OWASP AISVS 1.0 AC.8.4 (distinct principal per generate/review/approve/deploy stage) and AC.8.2 (scoped non-human identities that cannot promote their own artifacts).	Hardened-runtime + per-identity parts have no novelty; separation-of-duties is a verbatim normative requirement. The only residue — binding the separation into the *signed* migration-FSM gates — overlaps Claims 7/8 and should be merged there. **Anticipated standalone.**	`none (standalone)`
6. AI digital-twin operating model running the factory off-hours through the SAME gates as humans.	Developer digital-twin and scheduled/overnight ("dark factory") agents; agent-IAM consensus that agents are first-class principals under "the same authorization model used for humans, no relaxed enforcement"; documented coding agents that "use the same CI checks and branch protection as any human contributor" and cannot bypass them.	Both ingredients and the "same-gates-as-humans, no agent exception" parity principle are established — in the *software* domain, not only industrial ops. The only unclaimed wording is gate parity through a migration-specific signed phase-gate FSM — a predictable domain specialization. (Nuance: the industry *default* is extra gates for agents, so "deliberately identical" is a non-default choice — but it is the stated ideal of the parity literature, not novel.)	`weak`
7. No-self-sign invariant — agents *minted without* gate-signing capability	OWASP AISVS 1.0 AC.8.1 ("autonomous agents cannot approve, merge, sign, or deploy artifacts they themselves generated," technically enforced) & AC.8.2	**Margin is thin and, standalone, anticipated.** AC.8.2 is itself construction-time scoping	`none (standalone)`

(absent, not policy-denied).	(scoped identities provisioned unable to promote own artifacts); credential-proxy practice (signing keys held on host, outside the agent boundary — capability absent by construction); the audited/atomic-authz substrate conceded in §21 (US 8,225,378 / US 11,169,973).	("scoped" identities that "cannot be used to" promote), and credential-proxy practice is capability-absence-by-construction. "Minted without" vs "enforced to not have" is unlikely to read as inventive. Defensible only narrowly and only welded to Claim 8.	
8. Engine-*injected* verified-before-cutover gate, structurally non-removable by template authors.	Blocking verify-before-continue gates with segregation of duties; US 9,836,712 (computed migrateability state grants/prevents cutover); US 7,607,130 / US 10,474,506 / US 12,141,754 (author-defined FSM/table guards). Compile-time injection of author-unremovable steps and author-proof mandatory enforcement exist as separate, well-known techniques (policy-as-code hard-mandatory enforcement; branch protection "do not allow bypassing"; non-removable mandatory approvers).	**Strongest claim.** No single retrieved reference recites an engine that overlays the author's template at compile time and injects a verify-before-cutover guard the author cannot remove — in the cited workflow-FSM patents the *author* controls the gates. But the ingredients (compile-time injection + author-proof mandatory enforcement + verify-before-cutover) combine straightforwardly, so the claim is exposed to an obviousness attack. Novelty holds only at the narrowest mechanism level — the concrete *how* of injecting and locking the guard into the compiled FSM.	`moderate`
9. Self-improvement loop feeding template + agent learning from migration post-mortems.	Self-evolving / retrospective agent frameworks (score→diagnose→update→promote); "continual learning" software factories; generalizing successful workflows into reusable, metadata-tagged templates; operating-model	The dual target VOSJ relied on (one signal updates both the agent *and* a reusable template) is explicitly present in retrospective self-evolving frameworks	`none (standalone)`

	guidance to “continually improve the migration factory.”	that refine a subagent and persist the same artifact as a reusable template. Re-pointing a known dual-feedback loop at a signed-FSM template adds no inventive step, and it inherits Claim 2’s anticipation. **Anticipated.**	
10. Dual-connectivity factory (API + console) behind one churn-resistant control plane.	Migration-factory solutions exposing a REST API and a web console over one durable control plane; one-control-plane-many-surfaces agent platforms; “migration factory → migration control plane” as an industry term of art. The churn-isolation mechanism is the named *Anti-Corruption Layer* / abstracted-service-layer (adapter) pattern documented by Microsoft and others.	“API + UI over one control plane” is anticipated. The only candidate differentiator — churn-resistance via an adapter that absorbs provider-side interface change — is the textbook purpose of the Anti-Corruption Layer / Adapter pattern. Applying a named pattern to a known domain is a routine combination. **Anticipated.**	`none (standalone)`

WHERE THE NOVELTY ACTUALLY IS

Read honestly, VOSJ’s defensible core is **not a single invention** but a **narrow combination** applied to software migration: (a) a verified-before-cutover gate that is *engine-injected and structurally non-removable by template authors* (Claim 8 — the one mechanism no retrieved reference recites, and the only element that survives at `moderate`); welded to (b) executing agents *minted without* any gate-signing capability rather than policy-denied after the fact (Claim 7); (c) an identity-isolated executor decoupled from the issuer across a durable queue + MCP channel (Claim 4); (d) the *same* signed phase-gate FSM governing human and off-hours-twin alike, with no relaxed path (Claim 6); so that (e) 7-R disposition leaves no un-phased transition (Claim 3) and (f) each engagement’s post-mortem updates both the agent and the bound template (Claim 9). Each of these is anticipated *standing alone*; the asserted contribution is their welded composition in the migration domain.

We therefore claim the **specific composition** — engine-injected non-removable gate + no-self-sign minting + identity-isolated dispatch + same-FSM-for-twins, applied to application migration — as *one* combination, and we expressly do not assert novelty over any component in isolation. Claims 1, 4 and 6 contribute to the combination at `weak`; Claim 8 is its strongest mechanism at `moderate`; Claims 2, 3, 5, 7, 9 and 10 are recorded as standalone-anticipated and are abandoned as standalone-novel. Disclosing this combination as dated prior art — rather

than pursuing broad claims against parties who hold overlapping art — is the intended and most defensible posture of this paper.

22 Limitations & Threats to Validity

- **The signup constraint is real.** Fully-silent self-serve onboarding is impossible across major clouds; the hosted-redirect and reseller shapes (§15.3) are mitigations, not eliminations, of an attested human step.
- **Best fit is large, heterogeneous, audit-sensitive estates.** The seven-phase, multi-gate apparatus is overkill for a single-VM overnight lift; a collapsed "express" template is a clone decision, not the flagship.
- **Greenfield builds gain little** from Vault/Verify — there is little to discover, baseline, or reconcile.
- **Vendor funding is framework-specific.** A neutral methodology does not itself unlock a given provider's migration credits; those require the provider's own program engagement.
- **Autonomy quality depends on model capability and review bandwidth.** The safety invariants bound *risk*, not *quality*; poor agent output still consumes human review, which is the real throughput ceiling (§20).
- **Third-party outcome figures are illustrative** and must be independently verified before customer use.
- **This paper describes a design and its invariants**, not a completed empirical benchmark; the evaluation methodology (§20) is offered to enable independent measurement.
- **Multi-agent collusion / correlated error.** Role-separated agents give weaker independence than human four-eyes when they share a base model, prompt lineage, or the learning loop (§11.3); the residual risk is a plausible-but-wrong evidence package that misleads the human signer — mitigated, not eliminated, by machine-checkable gate criteria and by diversifying the reviewer/rollback agent's model and context.
- **Subscription rate limits.** Flat-rate model seats carry fair-use ceilings (rolling and weekly caps); at the ceiling the fleet queues/throttles or falls back to metered rates, qualifying the "marginal cost ≈ compute" framing (§18).
- **MCP/agent attack surface evolves quickly.** The threat model (§15.5) reflects current practice (injection, tool poisoning, confused-deputy); it must be revisited as the protocol and the attack literature move.

23 Future Work

- **Broader provider coverage** — deepen first-party connectors beyond the current pairs and promote "planned" provider stubs to full API orchestration.
- **Express templates** — a published, collapsed framework for small lift-and-shift jobs that retains the verified-before-cutover gate while dropping the heavyweight planning gates.
- **Incremental-delta transfer** — change-block-tracking incremental copies with verified delta bytes, not manifest-only tracking.
- **Formal verification of the gate engine** — machine-checked proofs that no reachable state violates the no-self-sign and verified-before-cutover invariants under any template.
- **Richer reconciliation** — semantic and behavioural equivalence checks beyond structural reconciliation.
- **Open evaluation corpus** — a shared benchmark of representative estates to make §20's metrics comparable across implementations.

24 Conclusion

Migration is a manufacturing problem dressed as an engineering problem. The individual steps are tractable; the cost lives in their lack of coordination, repeatability, audit, and labour scalability. VOSJ addresses all four by treating migration as a factory: four named stations that are brand and process at once; a data-driven framework engine that turns any methodology into one signed, gated state machine; a disposition model that enforces safe cutover styles by construction; and — the heart of the contribution — an AI execution substrate of MCP-commanded, identity-isolated autonomous agents and digital twins that perform the toil while a small set of structural invariants keeps every action auditable and prevents any agent from authorising its own work.

The result is a system whose scaling law is favourable (throughput bounded by safe review, not by headcount), whose risk profile is conservative (verified-before-cutover, reversible, separated authority), and whose governance output — an exportable, tamper-evident audit trail — is exactly what high-stakes buyers require. We have described the architecture, the invariants, the data model, the security and compliance posture, the economics, a worked scenario, and an evaluation methodology, and we have disclosed the inventive combinations as a defensive publication. The design's central lesson generalises beyond migration: *autonomous agents become safe to trust with consequential work precisely when the authority to approve that work is structurally withheld from them.*

> **Every migration is a voyage.** The factory's job is to make the voyage fast, observable, reversible, and boring — and to let machines row while humans hold the helm.

Appendix A — Glossary

Term	Definition
VOSJ	The migration factory described in this paper; the letters name its four stations.
Station	One of the four pipeline stages (Vault, Orchestrate, Shift, Jump).
Framework template	A consultant-authored methodology (phases/gates/roles) that compiles to the signed-gate state machine.
Gate	A state transition requiring machine-checked criteria plus a human signature by a named role.
7-R disposition	The per-workload decision: Retire, Retain, Rehost, Relocate, Repurchase, Replatform, Refactor.
Strangler-Fig	Incremental, parallel-run modernization that replaces a system piece by piece behind a switch.
MCP	Model Context Protocol — an open standard for connecting agents to tools and systems.
Devstation	A per-identity autonomous coding agent with its own clone, account, and credentials.
Digital twin	A persona configured with a principal's standards that operates autonomously off-hours.
Reconciliation	Multi-category comparison of target vs source that produces the pre-cutover equivalence proof.
No-self-sign	The invariant that agents are minted without any gate-signing capability.

Appendix B — Phase–Gate Reference

Phase	Gate(s)	Station	Signer role(s)
P1 Envision	Discovery sign-off	V	Migration director + customer sponsor
P2 Examine	Kickoff complete	V	IT lead
P3 Engineer	Data-method; Rollback-tabletop; Planning sign-off	O	DBA; director+DBA; director+IT lead
P4 Readiness	Drift check; Execution freeze; Vendor verified	O	auto; InfoSec+DBA; director

P5 Shift	Go/No-Go; Cutover	S	Full panel; director+DBA continuous (reverse = 3-way)
P6 Verify	Reconciliation pass	J	DBA (revocable 30 min)
P7 Jump	Return-to-BAU; Completed/Archived	J	IT lead; director (partner attests)

Appendix C — Architecture at a Glance

Experience → self-serve portal · glass-room board · Framework Library

↓

Factory control plane → V·O·S·J engine · template compiler · signed-gate FSM · 7-R engine · audit & metering

↓

AI execution substrate → MCP server (orders + tools) · devstation fleet (agents + humans) · digital twins

↓

Capability → migration executors · provider registry · credential vault · reconciliation · CI/CD scanners

↓

Foundation → Kubernetes runtime · PostgreSQL system-of-record · RBAC/identity · secrets · LLM bridge · CD pipeline

Appendix D — References & Sources

1. Microsoft Cloud Adoption Framework — Migrate & Modernize; landing zones; wave planning. Microsoft Learn, `learn.microsoft.com/azure/cloud-adoption-framework` (accessed 21 Jun 2026).
2. A major hyperscaler's migration-acceleration program — Assess / Mobilize / Migrate & Modernize; readiness assessment; TCO business case (vendor documentation; accessed 21 Jun 2026).
3. The 7 R's of cloud migration, evolved from the earlier industry-standard 5 R's (2010–2011).
4. Another hyperscaler's rapid migration & modernization program — Assess / Plan / Migrate / Optimize (vendor documentation; accessed 21 Jun 2026).
5. The Open Group, *TOGAF Standard* — ADM Phases E (Opportunities & Solutions) and F (Migration Planning); Transition Architectures. `pubs.opengroup.org` (accessed 21 Jun 2026).
6. M. Fowler, "StranglerFigApplication," `martinfowler.com/bliki/StranglerFigApplication.html` (accessed 21 Jun 2026).
7. Near-zero-downtime hypervisor migration technology — live, bulk, and replication-assisted modes (vendor documentation).

8. Model Context Protocol specification, revision `2025-06-18` (JSON-RPC 2.0; `stdio` & Streamable HTTP transports; OAuth 2.1 authorization). `modelcontextprotocol.io/specification` (accessed 21 Jun 2026).
9. OWASP Top 10 for LLM Applications (2025), incl. LLM01 Prompt Injection. `genai.owasp.org` (accessed 21 Jun 2026).
10. IETF RFC 9728 (OAuth 2.0 Protected Resource Metadata) & RFC 8707 (Resource Indicators for OAuth 2.0). `datatracker.ietf.org`.
11. Hardened container runtimes — gVisor (`gvisor.dev`), Kata Containers (`katacontainers.io`), Firecracker (`firecracker-microvm.github.io`).
12. DORA — DevOps Research and Assessment; the four key delivery metrics. `dora.dev` (accessed 21 Jun 2026).
13. Application-centric "migration factory" practice — factory throughput and dependency-centric discovery at scale.
14. OWASP Application Intelligence Security Verification Standard (AISVS) 1.0 — Appendix C, "AI for Code Generation," controls AC.8.1 (no self-approval/merge/sign/deploy of self-generated artifacts), AC.8.2 (scoped non-human identities that cannot promote their own artifacts), AC.8.4 (separation of duties across generate/review/approve/deploy). `github.com/OWASP/AISVS` (accessed 21 Jun 2026).
15. NIST Cybersecurity Framework (CSF) 2.0 — six core Functions: Govern, Identify, Protect, Detect, Respond, Recover. NIST CSWP 29, released 26 Feb 2024. `doi.org/10.6028/NIST.CSWP.29` (accessed 21 Jun 2026).
16. NIST SP 800-218, *Secure Software Development Framework (SSDF) Version 1.1* — practice groups Prepare the Organization (PO), Protect the Software (PS), Produce Well-Secured Software (PW), Respond to Vulnerabilities (RV). NIST, Feb 2022. `csrc.nist.gov/pubs/sp/800/218/final` (accessed 21 Jun 2026).
17. The Institute of Internal Auditors, *The IIA's Three Lines Model* (2020; updated 2024) — governing body, management (first & second lines), and independent internal audit (third line). `theiia.org` (accessed 21 Jun 2026).
18. MITRE ATLAS™ (Adversarial Threat Landscape for Artificial-Intelligence Systems) — knowledge base of adversary tactics and techniques against AI/ML systems; complementary to MITRE ATT&CK®. `atlas.mitre.org` (accessed 21 Jun 2026).
19. MITRE ATT&CK® — knowledge base of adversary tactics and techniques. `attack.mitre.org` (accessed 21 Jun 2026).
20. Microsoft Cloud Adoption Framework for Azure — Secure methodology (security guidance spanning Strategy, Plan, Ready, Adopt, Govern, Manage). Microsoft Learn, `learn.microsoft.com/azure/cloud-adoption-framework/secure` (accessed 21 Jun 2026).
21. ISO/IEC 27001:2022, *Information security, cybersecurity and privacy protection — Information security management systems — Requirements*. International Organization for Standardization (referenced for control-catalogue mapping).
22. Selected prior art reviewed for the differentiation analysis (§21.1): US 7,607,130 (workflow as data-transition-driven, permission-checked state machine); US 7,689,947 (data-driven FSM engine); US 10,474,506 (FSM-driven workflows / workflow template as FSM); US 12,141,754 (FSM workflows for data objects); US 9,836,712 (workflow-instance migration with a per-instance migrateable-state gate); US 9,098,359 (durable execution decoupling issuer from queue-polling workers); US 8,225,378 and US 11,169,973 (audited / atomic authorization). Cited as prior art over which no standalone novelty is asserted.

URLs are given as stable locators (accessed 21 Jun 2026); volatile vendor program names are intentionally generalised. Availability and terms change over time and should be re-verified against current documentation before customer use.

Appendix E — Document Provenance & Disclaimer

Document type	Technical white paper & defensive publication
Prepared by	Assuncao, Gustavo — PhD, AI Infrastructure Architect
Version / date	2.0 (final candidate) · 21 June 2026

Publication vehicle	To establish prior art, deposit in a recognised, timestamped, examiner-searchable venue (a defensive-publication database such as IP.com / Technical Disclosure Commons, and/or an archival venue minting a DOI such as arXiv or Zenodo). Record the venue, registration/DOI, and immutable timestamp here on deposit.
Intended audience	Technical practitioners and researchers

This document is provided for technical dissemination and to establish prior art. It describes a system architecture and method of operation. Product, framework, cloud-provider, and tool names referenced herein are the property of their respective owners and are used solely for technical accuracy and interoperability description; no affiliation or endorsement is implied. Any forward-looking statements, funding terms, and third-party outcome figures are illustrative and subject to independent verification.

Appendix F — Revision History (v1.1 peer review · v2.0 final candidate)

v2.0 (final candidate) adds a 20-year governance contributor, a security specialist, and an online prior-art study, all independently verified:

Area	Change	Sections
Governance	Added a governance operating model mapping the gate machine to COBIT 2019, ISO/IEC 38500, ITIL 4 change enablement, the IIA Three Lines Model, and SOC 2 CC8.1 / SOX ITGC; a Gate Council + waiver discipline, a RACI, and an exportable evidence package.	§12.3–§12.6, App. G
Security	Added a deployment-grade security architecture: a STRIDE×agent/LLM threat model, control mapping to NIST CSF 2.0 / ISO 27001:2022, zero-trust customer-environment deployment, and a runnable security sign-off checklist.	§15.6–§15.10, App. H
Novelty	Added a calibrated, examiner-credible differentiation analysis grounded in online prior-art research and adversarial verification: most claims are recorded as standalone-anticipated; novelty is asserted only for the narrow combination + the engine-injected non-removable gate.	§21.1
Deployment	Added a POC → production deployment playbook (scope, gates, prerequisites, scale-out) for using the VOSJ engine in a real customer environment.	App. I
References	Added dated, located sources for the new governance, security, and prior-art material.	App. D

v1.1 incorporates the verified findings of an independent four-reviewer peer-review board (IP/process, AI/MCP, infrastructure, migration):

Pri	Change	Sections
P0	Added the MCP/agent threat classes (prompt injection, tool poisoning / rug-pull / shadowing, confused-deputy, lethal trifecta, runtime escape) and clarified the invariants bound capability, not intent.	§9.3, §12, §15.5

P0	Hardened the devstation runtime class and moved standing credentials to a short-lived, audience-scoped broker; reworded the isolation claim to per-identity segmentation.	§10, §10.4, Claim 5
P0	Specified the defensive-publication deposit venue / timestamp requirement.	§21, App. E
P1	Corrected the MCP description (JSON-RPC 2.0; `stdio` & Streamable HTTP; SSE deprecation; OAuth 2.1 RS authorization) and the protocol-vs-Hub boundary.	§9.1, §14.2
P1	Reframed "MCP as an order bus" as a composition (durable queue + MCP tools; A2A noted).	§9.2, Claim 4
P1	Added PostgreSQL HA/DR and external HMAC key custody for the ledger.	§14.4
P1	Added a replication-lag / in-flight-rows hard pre-switchover gate; clarified performance parity is a P6 (post-cutover) check.	§13
P1	Gave Appendix D dated, locatable citations.	App. D
P2	Added multi-agent collusion and subscription rate-limit limitations; delegated-management capability limits; added "Secure" to the CAF spine.	§4, §15.1, §22

Three reviewer findings were *refuted* on adversarial verification and deliberately not actioned: that big-bang is not truly prevented (the structural disposition→runbook mapping holds); that the combination claims are "obvious" (no offensive filing is contemplated); and that performance-parity placement was wrong (it is correctly post-cutover).

Appendix G — Governance Control Catalog

The control catalog below enumerates the operating controls of a VOSJ migration, the role accountable for each, the V·O·S·J gate it binds to, the recognised framework objective it satisfies, and the evidence it produces. It is intended as a control library a customer can lift directly into a SOX/SOC 2/ISO programme. (Owner roles: Dir = migration director/chair; Risk = risk & control reviewer; FS = AI fleet supervisor; IA = internal audit. `Engine` denotes a structurally-enforced, non-waivable control.)

Control ID	Control	Owner / role	Gate bound	Framework objective	Evidence produced
`VG-01`	No agent self-signs a gate (authority withheld from non-human actors).	Engine / Dir	All (V·O·S·J)	COBIT EDM; IIA 3-Lines; SOX SoD	Signed gate row with human `signerRole`; self-sign counter (target 0).
`VG-02`	Separation of authoring and authorising (builder ≠ approver).	Dir	O, S, J	IIA 3-Lines; SOC 2 CC8.1	Distinct actor vs signer on each ledger row; role-separation report.
`VG-03`	Four-eyes change validation before a dependent gate clears.	Risk	O	ITIL Change Enablement; SOC 2 CC8.1	Diff-impact report; validator sign-off linked to change.
`VG-04`	7-R disposition assigned to every in-scope workload before kickoff clears (change classification).	Dir	V (P2)	ITIL change types; COBIT APO	Disposition on every workload record; kickoff gate criterion.
`VG-05`	High-risk dispositions forced onto incremental Strangler-Fig path (big-bang structurally unavailable).	Engine	S (P5)	COBIT BAI; ISO/IEC 38500 (Conformance)	Runbook emits only incremental steps; disposition→runbook binding.
`VG-06`	Independent rollback authored in a separate context and rehearsed (tabletop) before execution.	FS / DBA	O (P3)	ITIL Change Enablement; COBIT BAI	Rollback runbook (separate author); tabletop pass evidence.
`VG-07`	Baseline-drift guard blocks readiness when the baseline is stale.	Engine / DBA	O (P4)	COBIT MEA; ISO/IEC 38500 (Performance)	Auto drift-check result; readiness gate block/clear record.

VG-08	Execution freeze (infra/app) authorised before cutover window.	InfoSec / DBA	O (P4)	ITIL Change Enablement; SOX ITGC	Signed freeze record; calendar lock.
VG-09	Full-panel Go/No-Go (CAB convened as a gate).	Dir	S (P5)	ITIL CAB; COBIT EDM	Panel sign-off ledger row with all required roles.
VG-10	Verified-before-cutover — source serves until equivalence proof holds (engine-injected, non-removable).	Engine / DBA	J (P5→J)	COBIT MEA; SOC 2 CC8.1	Multi-category reconciliation report hashed to the cutover gate; unverified-cutover counter (target 0).
VG-11	Replication-lag / in-flight-rows hard pre-switchover gate.	DBA	S (P5)	COBIT MEA	Zero-lag attestation; in-flight-row resolution log.
VG-12	Revocable acceptance window (two-key safety margin) post-reconciliation.	DBA	J (P6)	ISO/IEC 38500 (Conformance); ITIL PIR	Acceptance row + revocation window timestamps.
VG-13	Contingency reversal requires three-way authorisation (emergency-change control).	Dir + DBA	S (P5)	ITIL ECAB; SOX SoD	Three-signature reversal record.
VG-14	Tamper-evident transition ledger (HMAC-SHA256, hash-chained, key custodied externally).	Engine / IA	All	COBIT MEA; SOC 2 CC7/CC8; SOX ITGC	Verifiable signed chain; re-computation report.
VG-15	Fail-closed credential vault (no master key → refuse, never default).	InfoSec	V–J	ISO/IEC 38500 (Acquisition); SOX ITGC (access)	Vault start-up assertion; fail-open counter (target 0).
VG-16	Least-privilege RBAC — every mutation requires an explicit capability; agents minted without sign capability.	InfoSec	All	COBIT DSS; IIA 3-Lines; SOX ITGC	Capability grants per principal; denied-action audit.
VG-17	Per-tool MCP authorisation, signed-server allow-lists, untrusted-response handling.	FS / Risk	V–J	OWASP LLM Top 10; COBIT DSS	Tool-call log (actor/args/result); rejected-call audit; server allow-list.
VG-18	One station = one identity = one clone	FS	V–J	IIA 3-Lines; SOX SoD	Per-identity action attribution; commit-via-PR

	(attributable, non-colliding execution).				records.
`VG-19`	Hardened runtime + brokered short-lived credentials for code-executing stations.	InfoSec / FS	S	CIS; COBIT DSS	RuntimeClass config; short-TTL token issuance log.
`VG-20`	Time-boxed exception/waiver with risk-graded authority; invariants non-waivable.	Dir + Risk	Any	COBIT EDM; ITIL Change Enablement	Signed waiver row (criterion, risk, compensating control, expiry, closure).
`VG-21`	Cross-platform access via sanctioned, scoped, revocable patterns; withdrawn at Jump.	InfoSec	V–J	ISO/IEC 38500 (Acquisition); SOX ITGC (access)	Per-engagement grant + revocation record in audit log.
`VG-22`	Data-protection controls for cross-border movement (DPA, SCCs, legal sign-off).	Risk	O–J	ISO/IEC 38500 (Conformance); SOC 2 privacy	Executed DPA/SCC; data-movement entries in audit log.
`VG-23`	Continuous + per-wave control testing; 8-hourly governance digest.	Risk	All	SOC 2 (continuous evidence); COBIT MEA	System-generated test results over the period; digest archive.
`VG-24`	Independent third-line assurance — ledger re-derived outside the operating team.	IA	All	IIA 3-Lines; SOX (independent test)	Internal-audit re-derivation report; signed assurance opinion.
`VG-25`	Decommission source only after confirmed BAU + retention.	Dir	J (P7)	ITIL Change Enablement; COBIT DSS	BAU sign-off + retention confirmation before decommission record.
`VG-26`	Exportable governance evidence package with control-mapping manifest.	Dir / IA	J (P7)	SOC 2; SOX; ISO/IEC 38500	Per-migration package: ledger + waivers + recon proofs + tool log + framework manifest.

Controls marked `Engine` are enforced in the data model and gate machine and cannot be disabled by a template author or satisfied by a waiver (VG-01, VG-05, VG-07, VG-10, VG-14, VG-15). The remaining controls are operating-model controls executed by the Gate Council and the three lines; all 26 emit evidence into the same tamper-evident ledger (§14.4), so the catalog doubles as the customer's control-test plan and as the index of the evidence package (VG-26).

Appendix H — Security Sign-off Checklist (POC Gate & Production Gate)

A runnable checklist for a customer security team. Each item is binary (PASS / FAIL / N/A) with evidence and a named signer; FAIL is fail-closed (blocks the gate). The **POC gate** qualifies a time-boxed, scoped pilot; the **Production gate** additionally requires the full control set before any live customer data or irreversible cutover. References point to the framework anchors in §15.6–§15.9.

H.1 POC Gate — scoped pilot, no irreversible action on production data

#	Control to verify	Evidence	Ref
P-1	Scope agreed in writing: workloads in scope, data classes, target region(s), engagement window; non-production / synthetic / masked data only.	Signed scope doc; data-classification list.	CSF GV; A.5.8
P-2	Access into customer estate is delegated/federated, scoped, time-boxed, and revocable — no long-lived shared keys.	Role/grant export; expiry timestamps.	§15.1; A.8.2
P-3	Each devstation has its own identity + clone + brokered short-lived creds; one station = one identity.	Identity inventory; token-TTL config.	§10.4; PR.AA
P-4	Fail-closed vault confirmed: with no master key the vault refuses to operate (no dev-key fallback).	Negative test result.	§15.2; A.8.24
P-5	Default-deny egress on station namespaces; allow-list limited to target / model / SCM / DNS.	NetworkPolicy manifests; egress test.	§15.8; CIS K8s
P-6	Hardened runtime class in force (gVisor / Kata / Firecracker) for code-executing stations.	`RuntimeClass` spec; escape test.	§10.4; A.8.31
P-7	No-self-sign verified: agent identities carry no `sign-as` capability (negative test).	RBAC export; failed self-sign attempt.	Inv. 1; LLM06
P-8	Tool-call + gate audit log enabled and immutable; sample exported and readable.	Log sample; integrity check.	§14.2; DE.AE
P-9	External MCP servers pinned + allow-listed; tool metadata statically validated before model exposure.	Allow-list; validation log.	§15.5; LLM03
P-10	Prompt-injection guard active: source content & tool responses treated as untrusted; sequence monitoring on.	Injection test result.	LLM01/05; ASI T2
P-11	Per-station kill-switch + queue freeze demonstrated; credential revocation effective.	Containment drill record.	§15.10; RS

P-12	POC exit criteria and tear-down defined (grants withdrawn, data purged, stations decommissioned).	Tear-down runbook.	§10.2; RC

H.2 Production Gate — live data & irreversible cutover authorised (includes all POC items, PASS)

#	Control to verify	Evidence	Ref
PR-1	All POC items (P-1…P-12) re-verified PASS against the production tenant.	Re-test record.	H.1
PR-2	DPA + sub-processor disclosure executed; SCCs in place for any cross-border move; legal sign-off recorded.	Signed DPA; SCC; legal sign-off.	§15.4; A.5.31; Privacy TSC
PR-3	Data residency / sovereignty pinned in config; verified no data leaves the agreed jurisdiction.	Region config; egress audit.	§15.8; ISO 27018
PR-4	HMAC ledger signing key custodied in external KMS/HSM; rotation + split-knowledge documented.	KMS config; rotation policy.	§14.4; A.8.24
PR-5	PostgreSQL HA/DR proven: quorum sync replication, WAL/PITR, stated ledger RPO/RTO; restore tested.	DR test report.	§14.4; RC; A.8.13/14
PR-6	Four-eyes change validation + separation of authoring/authorising enforced (builder ≠ reviewer ≠ signer).	Role map; sample gate trail.	Inv. 2–3; PR.AA
PR-7	Verified-before-cutover gate present on every bound template & non-removable; reconciliation thresholds set.	Template inspection; recon config.	§6.1; Inv. 6
PR-8	Independent rehearsed rollback per wave; baseline-drift guard active; three-way contingency reversal configured.	Tabletop record; drift config.	§13; RS/RC
PR-9	Audit/tool-call telemetry shipped to customer SIEM; detections mapped to MITRE ATT&CK + ATLAS.	SIEM integration; rule set.	§15.8; DE.CM
PR-10	Behavioural baselining live: deviation, descriptor-drift, and excessive-agency alerts firing in test.	Alert test results.	§15.6/9; LLM06
PR-11	Confused-deputy controls: token audience validated (RFC 8707), no client-token pass-through, Hub mints downstream tokens.	OAuth config; pass-through test.	§15.5; OAuth 2.1
PR-12	Tenancy isolation proven: per-tenant query filter + template/ledger scoping; cross-tenant access test fails.	Isolation test result.	§14.3; A.8.3
PR-13	Incident-response plan executed in a tabletop: containment, revocation, eradication, recovery, lessons.	IR exercise report.	§15.10; A.5.24–27

PR-14	Independent vulnerability scan / pen-test (incl. AI red-team: injection, tool poisoning, escape) — criticals closed.	Test report + remediation.	A.8.8; ATLAS
PR-15	Residual-risk register reviewed and accepted by the customer authority; framework coverage (CSF/ISO/SOC 2) attested.	Signed risk acceptance.	§15.10; GV.RM

Signers: each row is signed by the responsible role (customer InfoSec lead for security controls; DBA for reconciliation/HA-DR; migration director + customer sponsor for the overall gate). A single FAIL on a fail-closed item blocks the gate; N/A requires written justification. This checklist is itself an auditable artefact and should be archived with the engagement's exportable audit package (§12.2).

Appendix I — POC → Production Deployment Playbook

This appendix is the operational contract for running a VOSJ-engine proof-of-concept in a real customer environment and scaling it to production. It is written to be executed, not admired: a named gate either passes or the engagement does not advance. The engine-enforced controls are **on from day one** — they are what make autonomy safe to demonstrate, so the POC tightens nothing and relaxes nothing about the safety invariants (§12); it only narrows *scope*.

> **Operating principle.** A POC proves the *evidence package is genuinely auditable* against the customer's own control catalogue — it does not prove that controls can be skipped. "Audit-ready" must be the default end-state of every wave, not a separate project bolted on after an incident.

I.1 POC scope & objectives

Dimension	POC setting
Objective	Demonstrate one full V·O·S·J pass — assess → plan → incremental shift → verified cutover — on a representative-but-low-criticality workload, and export a VG-26 evidence package the customer's audit function accepts.
Workload scope	**A single low-risk wave** (a handful of co-dependent, non-tier-0 applications), on synthetic or masked / non-production data only. No customer PII or regulated data moves during POC.
Disposition mix	At least one Rehost/Relocate *and* one Replatform/Refactor unit, so both the simple runbook path and the Strangler-Fig-enforced path are exercised.
Gate Council (lean)	Migration director + customer sponsor + one risk/control reviewer + a DBA. At POC these may double up roles; in production they must be separated principals (I.6).
Duration / cadence	Time-boxed; control testing runs continuous + per-wave + on the 8-hourly cadence from the start (do not defer the reporting rhythm to production).
Out of scope	Production cutover of any tier-0 system; cross-border movement of live regulated data; reseller billing; any "express" template that drops the verified-before-cutover gate.

I.2 Environment prerequisites

- **Runtime.** Kubernetes-native deployment with a hardened `RuntimeClass` (gVisor / Kata / Firecracker micro-VM) for every code-executing devstation; a successful container-escape negative test recorded before any station claims work.

- **Identity & access.** Per-station identity + private repository clone + brokered short-lived, audience-scoped credentials (workload-identity federation / SPIFFE-style OIDC for the model seat; short-TTL source-control tokens). **No long-lived shared keys.** Cross-platform access uses each provider's sanctioned, revocable pattern (cross-account role assumption with external ID; Azure Lighthouse delegated management with scoped roles; workload-identity federation), requested per engagement and withdrawn at Jump.
- **Vault.** Fail-closed credential vault verified by a negative test (no master key → refuse to operate, never a development-key fallback). HMAC signing key custodied in an external KMS/HSM, not in the database (§14.4).
- **System-of-record.** PostgreSQL with quorum-based synchronous replication (≥3 instances), WAL archiving, point-in-time recovery, and a stated ledger RPO/RTO; a tested restore drill before any live data.
- **Network.** Default-deny egress with a minimal allow-list; data residency pinned in configuration and verified by an egress audit.
- **MCP Hub.** External MCP servers pinned and allow-listed; server-provided tool metadata statically validated before exposure to the model; per-tool RBAC and capability checks enforced at the Hub; immutable tool-call + gate audit log enabled; prompt-injection guard and tool-call-sequence monitoring active.
- **Kill-switch.** Per-station kill-switch + queue freeze + credential revocation drilled before the POC begins.

I.3 POC gate (I.1 — must all PASS before any data moves)

Check	Pass condition
Scope agreement	Synthetic/masked, non-production data only, agreed in writing.
Engine-enforced controls live	The non-negotiable controls (no-self-sign; engine-injected non-removable verified-before-cutover gate; fail-closed vault; tamper-evident signed ledger; baseline-drift guard; four-eyes validation) are ON and observable.
No-self-sign verified	A deliberate self-sign attempt by a station **fails** (capability absent).
Fail-closed vault verified	Negative test: no master key → refuse to operate.
Runtime escape test	Container-escape attempt contained by the hardened runtime class.
Auditability proof	A VG-26 evidence package is exported and is genuinely re-derivable (not a screenshot).
Teardown defined	Grants withdrawn, data purged, stations decommissioned at POC close.

I.4 POC success metrics

- **Governance integrity:** zero self-signed gates; 100% of gate transitions carry a valid signature by a correctly-separated authoriser (tests Invariants 1–2).
- **Safety:** zero unverified cutovers; every cutover backed by a complete verified-before-cutover proof (tests Invariant 6).

- **Reversibility:** the independent rollback for the wave is rehearsed and its mean time-to-roll-back recorded.
- **Reconciliation rigor:** zero post-cutover defects that reconciliation should have caught.
- **Throughput:** workloads cut over per unit of human supervision — the autonomy gain, isolated from raw compute.
- **Audit acceptance:** the customer's internal audit accepts the exported gate ledger as evidence (this is the gating success criterion for proceeding to production).

I.5 Bridge controls — required before any *live* data

Between POC and production, stand up the controls that POC scope deliberately deferred:

- **Control-catalogue mapping.** The risk reviewer maps VOSJ's controls (Appendix G control set) to the customer's existing register (SOX ITGC / SOC 2 / ISO/IEC 27001:2022), producing a control-mapping manifest the customer's internal and external auditors sign off — establishing audit-grade acceptance *before* scale.
- **Telemetry to the customer SIEM**, with tool-call and gate events mapped to MITRE ATT&CK and MITRE ATLAS; behavioural baselining for deviation, descriptor-drift, and excessive-agency alerts.
- **Data-protection instruments** executed: DPA + sub-processor disclosure and standard contractual clauses; data residency/sovereignty pinned and egress-audited.
- **Ledger key custody** moved into an external KMS/HSM with a documented rotation procedure; a third-line re-computation drill proving the hash-chained ledger verifies independently.

I.6 Scale-to-production path (staged)

Stage	What it establishes
S0 baseline	POC on a low-criticality wave with the full gate machine engaged and the lean council — prove the evidence package is auditable.
S1 map	Map controls to the customer's own catalogue; obtain auditor acceptance of the ledger as evidence.
S2 three lines	Stand up the three lines as *separated* principals per the IIA Three Lines Model: first line (delivery), a standing second-line risk & control function (four-eyes validator, baseline-drift guard), and a genuinely independent third line (internal audit re-derives the ledger). Roles that doubled up at POC must now be distinct people.
S3 change authority	Convert the lean council into the full Gate Council with risk-graded waiver authority and a named AI-fleet supervisor — accountable, but barred from self-assuring fleet work. Wire the customer's governing body in as counter-signer for HIGH-risk / non-waivable exceptions; the P5 Go/No-Go becomes a formally convened change board, and contingency reversal a formally constituted (three-way) emergency board.

S4 continuous testing	Move from spot checks to continuous + per-wave + 8-hourly control testing, feeding the customer's existing governing-body risk dashboard. Autonomous execution appears in the *same* board report as the rest of the IT estate — not a shadow process. Zero-tolerance counters (self-signed gates, unverified cutovers, fail-open events) are reported each cycle and must read zero.
S5 durability	Before production data moves, confirm ledger HA/DR (quorum synchronous replication, WAL/PITR, stated RPO/RTO) and external key custody; run the third-line re-computation drill.
S6 per-engagement close-out	Each production engagement: provision scoped/revocable access, execute DPA/SCC and legal sign-off for any cross-border data, withdraw access at Jump, and export the VG-26 evidence package indexed to the customer's control catalogue as the formal governance deliverable.

I.7 Production gate (I.2 — must all PASS before the first production wave)

- Re-verify *every* POC I.1 item PASSES against the production tenant.
- Prove PostgreSQL HA/DR with a **tested restore** (quorum sync replication, WAL/PITR, stated RPO/RTO).
- Enforce four-eyes and separation of authoring/authorising: `builder ≠ reviewer ≠ signer` (OWASP AISVS 1.0 AC.8.4 mapping).
- Confirm the verified-before-cutover gate is present and non-removable on *every* bound template, with reconciliation thresholds set per workload.
- Rehearse independent rollback per wave with the baseline-drift guard and three-way contingency reversal.
- Prove confused-deputy controls: audience validation (RFC 8707); no client-token pass-through; Hub-minted downstream-scoped tokens.
- Prove tenancy isolation: a cross-tenant access test **fails**; per-tenant filter on every query.

I.8 Production sign-off & operate

- **Independent assessment:** a vulnerability scan / penetration test including AI red-teaming (prompt injection, tool poisoning, runtime escape); all criticals closed.
- **Incident-response tabletop:** containment → revocation → eradication → recovery → lessons, exercised and recorded.
- **Residual-risk acceptance:** the customer authority reviews and accepts the residual-risk register.
- **Framework coverage attested** and archived with the engagement's exportable audit package: NIST CSF 2.0 (Govern / Identify / Protect / Detect / Respond / Recover), ISO/IEC 27001:2022, SOC 2, and — for the code-generating fleet — NIST SP 800-218 (SSDF) and OWASP AISVS 1.0.
- **Operate & re-review:** treat the threat model (§15.6) as living — re-reviewed each release against OWASP / CSA / MITRE updates; re-verify provider-specific revocation propagation per target; re-attest the residual-risk register and framework coverage at each phase gate and at engagement renewal.

> **End state.** Every production wave closes with a running, reconciled target *and* an exportable, tamper-evident audit package indexed to the customer's own control catalogue — produced as a by-product of the factory, not assembled afterwards. That artefact, not any single migration, is what makes VOSJ deployable into a regulated environment.
