▾ Vosj

Architecture & AKS Deployment — devstations, MCP server and portals

Technical design document · vosj.com · a project of Gus IT LLC · Apache-2.0

1. Overview

Vosj is a multi-cloud migration and replatforming engine. It moves applications and workloads between platforms (Microsoft Azure, Azure Local, AWS, Google Cloud, VMware, Hyper-V, on-premises) and replatforms them where appropriate — for example, containerizing a .NET application onto Azure Kubernetes Service (AKS). Every workload is driven through an audited, data-driven phase-gate pipeline (V·O·S·J), and every cutover is verified before it happens (fail-closed).

2. The phase-gate engine

A migration is a finite-state machine with four ordered stages — Vault, Orchestrate, Shift, Jump. The state machine is not hardcoded: it is compiled from a selectable framework template (e.g. the Microsoft Cloud Adoption Framework). State transitions are HMAC-signed and written to an append-only log. Gate sign-off is performed by a human; no automated agent signs a gate.

Stage	Function
Vault	Discovery & assessment: inventory, dependency graph, 7-R disposition, TCO, CI/CD maturity.
Orchestrate	Wave planning: target & landing zone, framework template, cutover sequence.
Shift	Incremental migration (Strangler-Fig); source + target run in parallel; connectors replicate.
Jump	Verified cutover (checksums / row-counts / smoke); fail-closed; record deploy; decommission.

3. The plugin seam

Behaviour is provided through a small set of versioned interfaces, so the engine is unchanged while implementations differ:

- `Connector` — source/target adapters: discover · replicate · verify · cutover · rollback. `verify()` is mandatory.
- `Executor` — how the work of each step is performed.
- `StateStore` — where migration state is persisted.
- `GateSigner` — gate sign-off; human-only, fail-closed.
- `AssessmentProvider` — the CI/CD & DevOps maturity scorecard.

4. Deployment on AKS — devstations, MCP server & portals

The execution fabric runs on Azure Kubernetes Service. It is the same fabric in both editions; only the AI driver differs.

- **Portals** — browser entry points: the web IDE access and the migration console.
- **Devstation pods** — code-server IDE environments where engineers work, one per engineer.

- **MCP server** — a Model Context Protocol server that acts as the order and tool channel for automated work.
- **Vosj engine + connectors + StateStore** — the migration runtime.
- **AI driver** — Community Edition: bring your own AI or drive manually; Enterprise: the Luca AI (managed personas + digital twins).
- **Azure control plane** — Microsoft Entra ID + ARM; landing zone of AKS / ACR / Azure SQL / Key Vault, with GitOps and OpenTelemetry.

Reference deployment: an AKS cluster hosts the portals, the devstation pods, the MCP server and the Vosj engine. Engineers reach the devstations through the portals; the MCP server channels automated steps to the engine and connectors; the Luca AI (Enterprise) or a bring-your-own agent (Community) connects to the MCP server to perform routine work. Workloads land in the Azure target zone and are reconciled by GitOps.

5. Open-core boundary

Community Edition (open source, Apache-2.0) includes the engine, the framework-template model, the CLI, starter connectors, the CI/CD & DevOps 365° assessment, and the MCP server + devstations. You run the execution fabric yourself and bring your own AI, or drive it manually.

Enterprise adds the Luca AI — managed AI personas and per-engineer digital twins that drive the MCP/devstation fabric autonomously (including off-hours) — plus governance (SSO, RBAC, full audit), advanced connectors and support. Humans sign every gate; no AI self-signs.
