What Does It Cost to Leave — or Arrive?

2026-07-09

Switching cost in practice: skills gaps, familiarity, and a working example of designing for reversibility instead of assuming permanence. Part 6 of 6.

Part 1 deliberately kept Question 5 at the general level too: on-prem is largely sovereign, but "engineering time re-architecting for a new platform's primitives, process changes… and a skills gap no one accounted for" apply whichever direction you're moving. This post is that deep-dive — the last of the five, using examples already built and published rather than new claims about any provider.

Engineering time and process change

The managed-vs-self-hosted log archiving comparison already named this cost plainly, in the middle of what looks like a pure pricing table: the self-hosted ClickHouse path on OVH, at roughly €54,000 over seven years, is the cheapest option by a wide margin — but the post states outright that it "requires Kubernetes operational experience but eliminates vendor lock-in entirely." That's not a footnote. The cheapest number on the page isn't a free choice; it's a different bill, paid in required skill rather than a vendor invoice, and it doesn't show up in the cost comparison at all.

Familiarity is the same cost, viewed from both directions

The other half of this shows up directly in a completely different decision: I've used OVH's API since 2011, across dedicated servers, OpenStack Public Cloud, and Octavia load balancers, and that familiarity is what makes reaching for OVH a five-minute decision for a new project rather than a week of evaluation. Read that the other way for what it says about leaving, not just arriving: the exact familiarity that turns building something new into a five-minute decision is what an organization is giving up the moment it moves to a platform it doesn't know yet — or gets asked to leave one it does. It's one cost, not two, depending only on which direction you're looking from.

Designing for reversibility instead of assuming permanence

Most of the time, that cost sits latent until something forces the question. The LiteLLM gateway in front of a self-hosted vLLM endpoint is a working example of building the alternative in ahead of time instead: when the GPU node is scaled to zero and a request arrives cold, LiteLLM automatically fails over to Claude or OpenAI — client-transparent, no re-pointing, no manual intervention. That's a genuine reduction in switching cost, engineered rather than assumed.

It is not free, and that's the part worth keeping. A fallback request is billed at the commercial provider's per-token rate instead of the self-hosted rate, and a burst of cold-start fallbacks produces a real, measurable spike — monitorable through LiteLLM's `/spend/logs` endpoint and a dedicated Prometheus metric, precisely because it's expected to happen and needs watching. The point isn't

that reversibility is free. It's that a switching cost discovered for the first time during an actual failure tends to run higher than the same cost paid in small, known increments beforehand.

Why this question doesn't fit in a comparison table

Questions 1 through 4 in this series — ownership, technology stack, enforcement, exit terms — all describe a provider or platform as it stands today, which is exactly why they fit into the comparison tables Parts 1, 2, and 4 built. This question doesn't, because it isn't about a static state at all — it's about what happens during the transition between two states, and a table has no column for "how long the team takes to become competent on the destination platform." That's also why this post, alone in the series, couldn't be answered by researching a named vendor: the cost lives in the team doing the switching, not in the platform being switched to or from.

None of this is a recommendation for any specific switching strategy. A team with deep operational experience on one platform and none on another will reasonably weigh that gap differently than a team building greenfield with no existing investment anywhere — and a workload that can tolerate a LiteLLM-style engineered fallback has different requirements than one that can't. This post names where the cost hides. It doesn't tell a reader what to do about it.

This is the final part of the series. ← Back to the Guide

Related

- Sovereign-Cloud-Washing: Five Questions — Question 5, where this framework gap was first named
- LLM Inference on OVH MKS: LiteLLM Gateway — the cold-start fallback mechanism behind this post's reversibility section
- AWS vs. GCP vs. Azure vs. OVHcloud: Managed Log Archiving — the self-hosted ClickHouse cost/skills tradeoff