Digital Sovereignty: The Complete Guide

2026-07-09

Reading guide for a six-part series testing "sovereign cloud" claims against ownership, technology stack, access, enforcement, exit, and switching cost.

This is the index and reading guide for a six-part series built around one idea: "sovereign cloud" is a marketing label until it's tested against something specific. Part 1 sets out five concrete questions to test it against instead of taking the label at face value; Parts 2 through 6 are the deep-dives that answer each one in turn — grounded, wherever possible, in research or systems already published elsewhere on this blog rather than written fresh for the series.

One of the five questions — legal ownership and jurisdiction — is deliberately answered thin. Part 1 concludes early on that tracing ownership chains and judging their legal weight is corporate-law territory, not engineering, and points to a qualified person instead of attempting the analysis itself. The other four get full deep-dives.

How to read this guide

New to the framework — start with Part 1; everything else builds on it. Checking a specific hyperscaler's sovereignty claim — go to Part 2. Auditing your own access model — Part 3. Trying to tell whether a specific security control is real or just a policy promise — Part 4. Planning a migration or worried about lock-in — Part 5. Weighing a platform decision where switching later might matter — Part 6.

Part 1 — Sovereign-Cloud-Washing: Five Questions

Audience: anyone evaluating a "sovereign cloud" claim, from either a technical or a decision-maker angle.

Part 1 lays out the framework: legal ownership and jurisdiction, the technology supply chain and who (or what) has access to it, whether protection is legally- or technically-enforced, whether a real exit path exists, and what it costs to switch — to a cloud, between clouds, or back to on-prem.

Key takeaways:

- Five questions, not a definition of "sovereignty" — testing a specific claim beats accepting the marketing.
- Question 1 (ownership/jurisdiction) is intentionally minimal: this blog's expertise is engineering, not corporate law, and the post says so rather than attempting an analysis it isn't qualified to make.
- The closing checklist is exactly that — a checklist to run against any provider, not a ranking of named vendors.

Read Part 1 — Sovereign-Cloud-Washing: Five Questions →

Part 2 — Who Builds the Platform? Ownership vs. Stack

Audience: technical readers checking a specific provider's sovereignty claim against its actual technology stack.

Part 2 is the technology-supply-chain deep-dive: six cases where legal ownership and the underlying technology stack diverge or fail to add up to independence, verified against primary sources rather than repeated from marketing pages.

Key takeaways:

- Six divergence cases: Open Telekom Cloud/Huawei, Bleu/Microsoft, S3NS/Google, Microsoft Sovereign Cloud, AWS European Sovereign Cloud, and Google Cloud's own partner-delegated model — each a different shape of the same underlying gap.
- Three deliberate counter-examples — OVH, STACKIT, and Aruba Cloud — where ownership and technology stack actually align, each with a real, named gap left in rather than smoothed over, so the post reads as a rational check rather than hyperscaler-bashing.
- Microsoft France's own director of public and legal affairs telling the French Senate under oath he "cannot guarantee" EU data won't reach US authorities, and AWS's own whitepaper stating its global engineering teams keep developing the software — the strongest evidence in the post is the vendors' own words.

Read Part 2 — Who Builds the Platform? Ownership vs. Stack →

Part 3 — Who Has Access? Humans, Accounts, AI Agents

Audience: anyone auditing their own access model, not just a vendor's.

Part 3 needed no new research — it checks Kubernetes RBAC, OVH IAM v2, STACKIT's CI/CD-facing identity-federation building block, LiteLLM's per-user keys, and this blog's own AI coding agent against the same bar: documented, audited, scoped, and revocable.

Key takeaways:

- A full workload-identity-federation comparison across AWS (IRSA), GCP, Azure, OVH (no equivalent), and STACKIT (a real building block exists, but native Kubernetes integration isn't documented).
- The AI-agent-access section is explicitly framed as **one example** of interactive, sovereign agent use — this blog's own Claude Code workflow — not a general solution to the risk of an agent holding broad, unaudited execution rights.
- RBAC and LiteLLM keys are technically enforced at the API/gateway layer; this project's own `CLAUDE.md` rules are written policy backed by partial technical friction — a distinction Part 4 formalizes next.

Read Part 3 — Who Has Access? Humans, Accounts, AI Agents →

Part 4 — Legally vs. Technically Enforced

Audience: anyone trying to tell whether a specific security or sovereignty control would actually hold up, or just sounds like it would.

Part 4 needed no new research at all — pure synthesis of Parts 1-3 into an explicit, three-rung framework.

Key takeaways:

- Rung 0 (policy/promise only), Rung 1 (technically-enforced at rest — BYOK/SSE-C), Rung 2 (technically-enforced at rest *and* in use — confidential computing, described generically since no case in this series was found operating there).
- A synthesis table mapping everything found so far — BYOK, RBAC, LiteLLM keys, Data Guardian, AWS's "Qualified… Staff," Bleu's source-code-review right, `CLAUDE.md` — onto these three rungs.
- The CLOUD Act is the sharpest real-world test: a legally-enforced promise has to survive a jurisdiction with extraterritorial reach; a technically-enforced one doesn't, because there's nothing to hand over.

Read Part 4 — Legally vs. Technically Enforced →

Part 5 — Can You Leave? Data Portability & Egress

Audience: anyone planning a migration, or trying to price in lock-in before signing up.

Part 5 combines already-published export/API/egress facts for AWS, GCP, Azure, and OVH with new, targeted research checking Part 2's five sovereignty offerings specifically on exit.

Key takeaways:

- Concrete bulk-export commands and formats across four providers, SQL-family portability versus Azure's proprietary KQL, and OVH's no-egress-within-network policy against AWS/GCP/Azure's per-GB rates.
- Checked against Part 2's cases: AWS European Sovereign Cloud uses standard AWS APIs but runs as a separate account partition with real administrative exit friction; Microsoft's own documentation names no data-export mechanism, format, or egress pricing for its Sovereign Cloud at all; Bleu's API openness could not be confirmed at all; Google's sovereignty offerings turned out to be **two structurally different products** — one using standard GCP endpoints, one (S3NS's Cloud de Confiance) with its own proprietary API domain.
- The EU Data Act phases out ordinary switching and egress charges entirely from January 2027, for every provider's covered data-processing services regardless of sovereignty branding — premium migration services can still be priced separately, and today's numbers are a snapshot, not a permanent feature of any of these offerings.

Read Part 5 — Can You Leave? Data Portability & Egress →

Part 6 — What Does It Cost to Leave — or Arrive?

Audience: anyone weighing a platform decision where a future switch — in either direction — is a real possibility.

Part 6 closes the series using only already-published examples: no new research, no new anecdotes.

Key takeaways:

- The self-hosted-ClickHouse cost comparison's own line — cheapest on paper, but it "requires Kubernetes operational experience" — names the skills-gap cost that a pure price comparison hides.
- The same familiarity that makes building something new on a well-known platform a five-minute decision is exactly what makes *leaving* that platform costly — one cost, viewed from two directions.
- The LiteLLM gateway's automatic fallback to a commercial API during a cold GPU start is a working example of engineering reduced switching cost ahead of time, instead of discovering it during an outage — and it still isn't free.

Read Part 6 — What Does It Cost to Leave — or Arrive? →

The through-line

Three things connect all six parts.

Every deep-dive turns a yes-or-no question into "it depends on the specific mechanism, checked." Whether a provider "is" sovereign, whether an access model "is" secure, whether a promise "is" enforceable — every part resolves the same way, by naming the specific thing to check rather than handing down a verdict.

Nothing in Parts 2 through 6 was invented for this series. Every deep-dive is either new, verified research (the ownership-vs-technology-stack cases in Part 2, the exit-path checks in Part 5) or a synthesis of systems and findings this blog had already published and operated before the series existed (the RBAC/IAM comparison and Claude Code example in Part 3, the framework synthesis in Part 4, the cost examples in Part 6).

None of the six parts ends in a ranking. Each closes the same way: naming what a reader should check for their own workload and risk tolerance, not which provider wins. A hyperscaler-run offering, a joint venture, and a self-hosted platform each have a legitimate place depending on what is actually being protected and from whom.

Where to start

Only have time for one part — read Part 1 for the framework itself. Already sold on the framework and evaluating a specific provider — go straight to Part 2 for the ownership-and-stack cases, or Part 5 if exit terms are the deciding factor. Building or auditing your own systems rather than evaluating a vendor — start at Part 3 and Part 4.