OPEN INNOVATION

SOVEREIGN AI INFRASTRUCTURE

Building Strategic Infrastructure for Control and Governance in AI Operations

WHITEPAPER BY OPEN INNOVATION AI

Content

Executive Summary

The global AI landscape is splitting in two. For years, the default assumption was that intelligence would live in a handful of hyper-scale clouds, centralized platforms optimized for efficiency, scale, and convenience. That model worked when AI was mostly experimentation and product features.

But as AI becomes foundational to critical operations like healthcare systems, financial markets, industrial automation, public services, and defense-adjacent use cases, organizations start treating AI less like “software you rent” and more like strategic infrastructure you must control.

Sovereign AI is the end-to-end capability to build, run, and govern AI datasets, training pipelines, model artifacts, and inference services under an operator’s authority, with minimal hidden dependencies.

This capability is framed across four dimensions:

- Deployment sovereignty
- Model sovereignty
- Operational sovereignty
- Human capability sovereignty

The emphasis is the same across all dimensions: verifiability. Organizations must be able to prove who accessed what, where workloads ran, what policies were enforced, and how models were promoted across their lifecycle.

If the goal is sovereign AI, the architecture must be designed so that control is enforceable and evidence is automatic not something reconstructed during an incident or audit.

Sovereignty starts with physics. The foundation of any sovereign AI system lies in the physical layer, which supports all higher layers of control and governance.

This need emerges from three primary pressures:

- Economic: AI becomes a long-term productivity engine, and dependency introduces strategic risk
- Security: reducing exposure to external control planes and supply chain fragility
- Fit: models must reflect local requirements, constraints, and policies

Meeting these requirements forces a rethinking of the entire AI stack moving away from opaque, multi-tenant public clouds toward dedicated AI factories where performance is deterministic and governance is enforceable.

Introduction:

Sovereign AI Infrastructure Framework

The global AI landscape is splitting in two. For years, the default assumption was that intelligence would live in a handful of hyperscale clouds, centralized platforms optimized for efficiency, scale, and convenience. That model worked when AI was mostly experimentation and product features.

But as AI becomes foundational to critical operations like healthcare systems, financial markets, industrial automation, public services, and defense-adjacent use cases, organizations start treating AI less like “software you rent” and more like strategic infrastructure you must control. That is the practical origin of Sovereign AI.
At Open Innovation AI, we anchor Sovereign AI to one practical idea: sovereignty is only real when it can be operated and proven. It is not enough to say “data stays here” or “we have a private cluster.” Sovereign AI is the end-to-end capability to build, run, and govern AI datasets, training pipelines, model artifacts, and inference services under an operator’s authority, with minimal hidden dependencies.

We frame that capability across four dimensions. Deployment sovereignty governs where workloads run, who controls the operational surface, what the platform depends on, what can change without your consent, and what evidence your platform emits by default. Model sovereignty governs how models are built, versioned, and promoted across their lifecycle. Operational sovereignty governs who runs the platform day to day and under what processes and change controls. Human capability sovereignty ensures the organization can build and operate what it owns without hidden dependency on external expertise. Across all four, the emphasis is the same: verifiability. Being able to prove who accessed what, where workloads ran, what policies were enforced, and how model provenance was maintained. If the goal is sovereign AI, the architecture must be built so that control is enforceable and evidence is automatic, not something you reconstruct during an incident or an audit.

Each dimension deserves its own treatment. In this post, we focus on one concern that cuts across all four: the physical layer. It is the foundation that every higher-layer sovereignty claim rests on, and it is frequently underspecified in discussions that rush toward orchestration and governance.

That need usually emerges from three pressures

- Economic: AI becomes a long-term productivity engine. Dependency on someone else’s capacity and pricing becomes a strategic risk.
- Security: operators want to reduce exposure to external control planes, supply-chain fragility, and legal compulsion that could affect availability or confidentiality.
- Fit: models must reflect local languages, taxonomies, safety rules, and operational constraints, often requiring private datasets that cannot be handled casually.

The Physical Layer:

Engineering the Sovereign AI Factory

Sovereignty starts with physics. If the building, power chain, cooling system, or physical access controls are brittle, every higher-layer promise is conditional. AI training loads are not “bursty web traffic.” They are sustained, synchronized campaigns: thousands of accelerators moving in lockstep, sensitive to transient instability.

That’s why sovereign facilities are designed around determinism, and why you should treat physical engineering as part of your deployment sovereignty story because it decides whether you control your runtime conditions or merely hope for them.

- Power: modern AI racks routinely move into multi—tens of kilowatts, and rack-scale systems can demand 100 kW+. At that point, the problem is no longer “do we have enough power?” but “can we deliver clean power consistently under load transitions, failovers, and maintenance?” You should design and validate for transitions, not averages. High-resolution metering becomes operational, not optional, because it’s how you detect drift before it becomes downtime, and how you produce evidence that performance issues were not caused by power instability.

- Cooling: air cooling eventually hits ceilings in fan power, airflow constraints, and hot-spot risk. Direct-to-chip liquid cooling improves stability at high density. Immersion cooling can be even more efficient, but it changes servicing workflows and facility integration. In sovereign environments, cooling choices are as much about resilience and operational control as thermodynamics. You should choose a cooling approach that preserves deterministic operation under peak load while still allowing controlled maintenance without destabilizing adjacent racks.

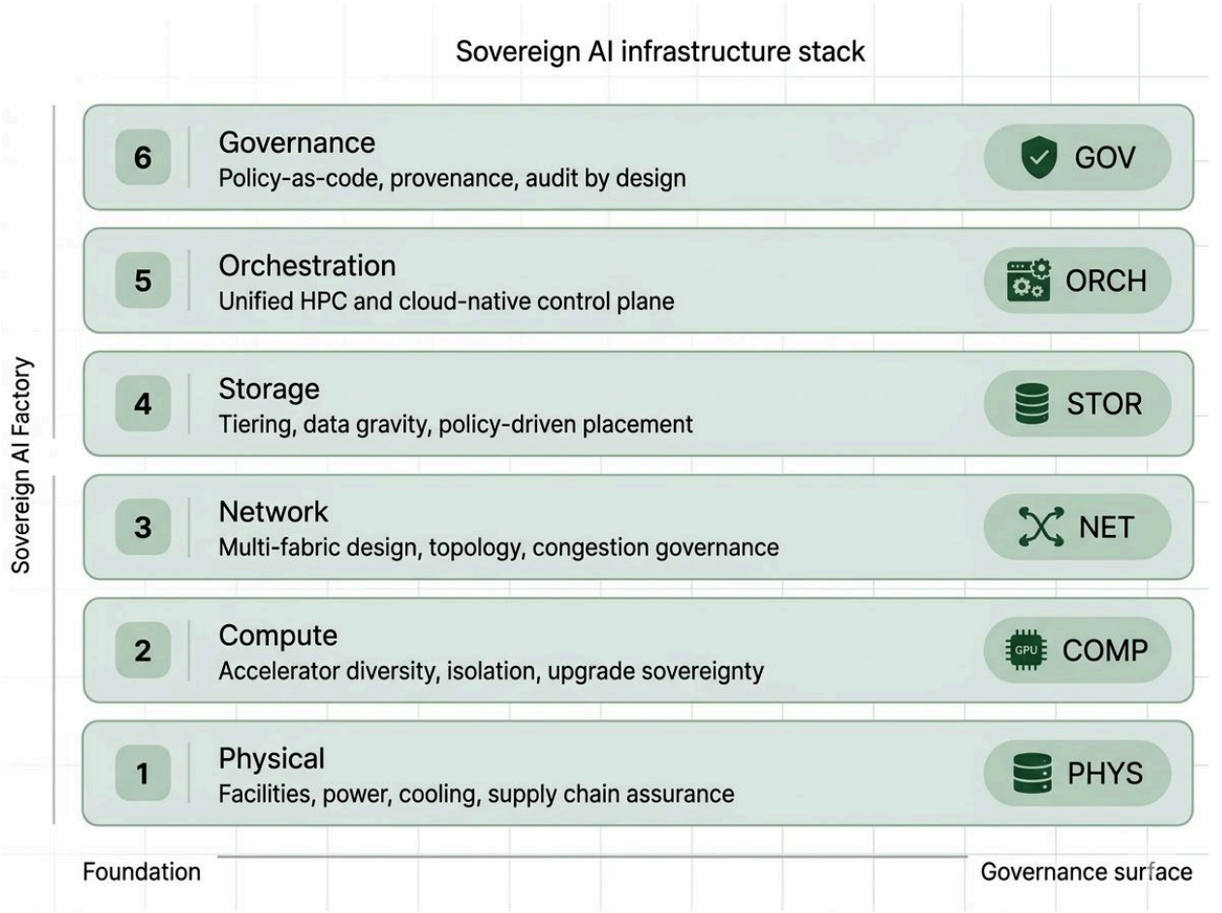
- Physical assurance: segmentation, controlled access workflows, and chain-of-custody procedures belong to the physical layer. Hardware Root of Trust (measured boot, firmware signing, attestation) turns “we think this server is clean” into “we can prove it’s running an approved baseline before joining a sensitive partition.” If deployment sovereignty is about enforceable control, then the physical layer is where you first enforce “who can touch what” and “what is allowed to join.”

Supply chain dependency is also physical-layer reality. Sovereignty is undermined if regulated partitions can be silently altered by untracked components, inconsistent firmware baselines, or emergency substitutions without provenance. You should run your AI factory with an approved component baseline mindset: what hardware is allowed, what firmware is allowed, what proof is required, and what the spares strategy is for regulated zones. This isn't bureaucracy; it's operational survivability when lead times stretch, exports tighten, or vendor availability shifts.

The Physical Layer:

Engineering the Sovereign AI Factory

With the physical foundation established, the rest of this post works through the remaining layers of a sovereign AI stack. Figure 1 illustrates all six layers and serves as a reference map for the sections that follow. Each layer is numbered so the reader can track exactly where in the architecture the discussion sits at any point.



Together, these layers form a sovereign AI factory: an environment where performance is deterministic, governance is enforceable, and every claim about control can be proven rather than assumed. The sections below work through each layer in turn.

The Compute Layer:

Silicon Diversity, Isolation, and Optionality

Compute sovereignty is about three things: predictable performance, isolation, and optionality. The goal is not only to run models fast, but to avoid turning your AI roadmap into a single-vendor dependency with a single operational failure mode.

Modern AI platforms increasingly move toward heterogeneous compute pools i.e. different accelerator families for different workload classes (large-scale training, high-memory inference, embeddings/RAG pipelines, batch analytics) all presented through a unified control plane. The point isn't just performance. It's avoiding a single strategic chokepoint while still delivering consistent results. You should treat accelerator diversity as a risk-control lever, not a procurement preference, because accelerator scarcity and platform shifts are now normal operating conditions.

Isolation is the second pillar, and it's where deployment sovereignty becomes measurable. Bare metal matters for massive training to maximize RDMA paths and GPU-direct performance. SR-IOV and similar mechanisms provide near-native NIC access with strong separation for multi-tenant environments. GPU partitioning, where supported, improves utilization and reduces interference for inference workloads. Confidential computing extends this further for the most sensitive workloads: nodes must prove they are in a verified state before they are eligible to run regulated jobs. If you can't enforce these admission rules, your "sovereign" posture is simply policy text without platform enforcement.

Operationally, you should also plan for upgrade sovereignty: how drivers are rolled out, how firmware baselines are maintained, how regressions are detected, and how you roll back safely. A sovereign platform is not only "owned," it is operable under change without losing performance determinism or audit continuity.

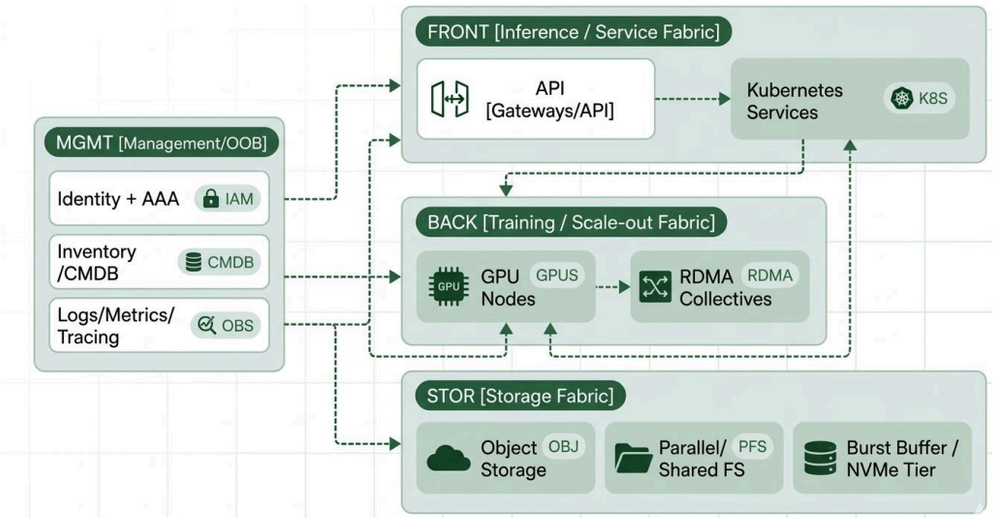
The Network Layer:

Multi-Fabric by Design

In large-scale AI, the network is not plumbing, it's part of the machine. Distributed training is dominated by collective communication. If latency jitters or congestion becomes unstable, GPUs wait, utilization drops, and expensive infrastructure idles.

Sovereign architectures typically build multiple network domains, because the traffic types and risk profiles are different. Training/scale-out fabric (backend) is built for low jitter, high throughput, RDMA-optimized behavior. Storage fabric is throughput-heavy and must tolerate checkpoint storms and burst behavior without collapsing. Inference/service fabric (frontend) emphasizes segmentation, observability, and stable north—south access patterns. Management/OOB is the control surface of the factory and must be isolated, tightly governed, and treated as a production security boundary.

The figure below illustrates different network fabrics connecting to a server. Storage fabric can either be distributed or dedicated to data centers. Inference fabric acts as frontend and is used for user interactions. Training fabric acts as a backend and is used for training or learning purposes.



Fabric choice InfiniBand vs RoCEv2:
A critical decision based on performance guarantees and operational control, not preference. InfiniBand offers deterministic behavior and mature HPC tooling, while RoCEv2 provides Ethernet flexibility but requires careful congestion management. In both cases, performance must be measurable and provable, not assumed.

Public Cloud vs Sovereign AI Network:
Public cloud networks are powerful but limit direct control and can lack determinism due to multi-tenant environments. In contrast, a sovereign AI factory enables full control, making the network a predictable, auditable system.

Training Network Architecture:

ROD vs RUD, and Why Topology Shapes Operations

The network design for AI/ML workloads requires a balance between performance, cost-efficiency, reliability, and scalability. An organization needs a durable network that can cope with the challenging nature of AI/ML calculations while staying flexible to future expansion and technological improvements. This can be achieved by applying optimized topologies, weighing cost trade-offs, and using strong failover strategies.

AI/ML workloads have demanding networking requirements that necessitate high-speed connections between server and leaf switches which is typically between 200 Gbps and 400 Gbps. The network design therefore must include specialized switches, high-bandwidth cabling, and topology adjustments to ensure optimal performance and throughput.

Servers such as the Nvidia A100 or H100 are equipped with 8 GPUs, and each GPU may be linked to one or two NICs. While multiple NICs per GPU is feasible, cost considerations around optics, NICs, and server port scalability often favor the implementation of a single NIC per GPU. This approach reduces expenses and eliminates port-level redundancy, removing the requirement for multi-homing on the server side. That choice increases the importance of deterministic fabric behavior because the network becomes the dominant lever for keeping utilization high.

The GPU-to-leaf connectivity options include multiple GPUs connected to a single leaf or a one-to-one GPU-to-leaf configuration. For example, for a server housing 8 GPUs, it is essential to have 8 leaf connections, forming what is commonly referred to as a rail-optimized topology. This topology choice is preferred because of its efficiency in handling high-throughput AI/ML workloads; it offers enhanced data transfer capabilities and optimized performance.

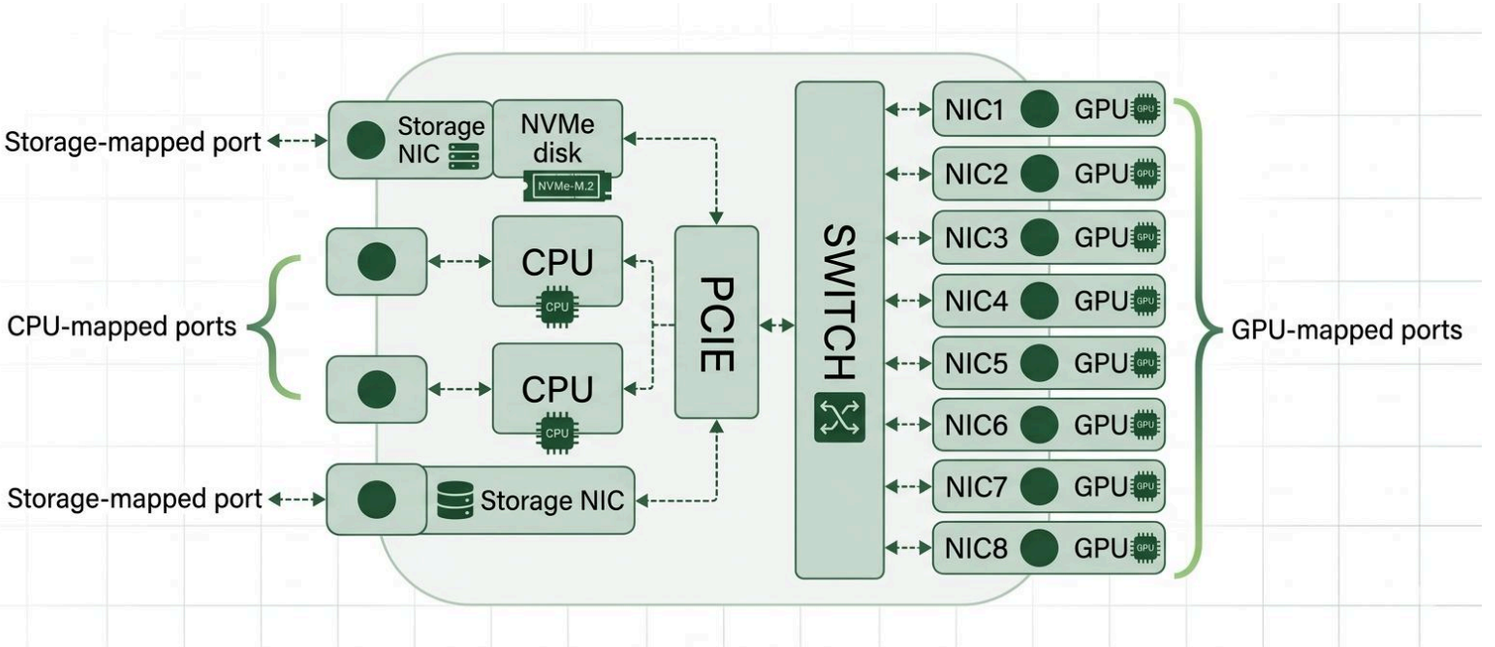
Training Network Architecture:

ROD vs RUD, and Why Topology Shapes Operations

There are two design options for GPU-to-leaf connectivity:

- One-to-one GPU-to-leaf configuration, referred to in this book as rail-optimized design (ROD). (There is a similar-sounding concept, rail-only design, where there is no communication between the different rails. We discuss this design in subsequent sections on the network fabric.)
- Multiple GPUs connected to a single leaf, referred to in this book as rail-unified design (RUD).

The figure below shows the different network ports found on GPU-based servers. GPUs use specific ports on a server to communicate to other GPUs across servers. These servers are mainly used for training in training networks. Certain ports within a server are used for CPU communication across servers or for northbound connections. These ports are mainly used for inference and are part of frontend networks. In addition, some ports are used for Non-Volatile Memory Express (NVMe) communication for fast data transfer.

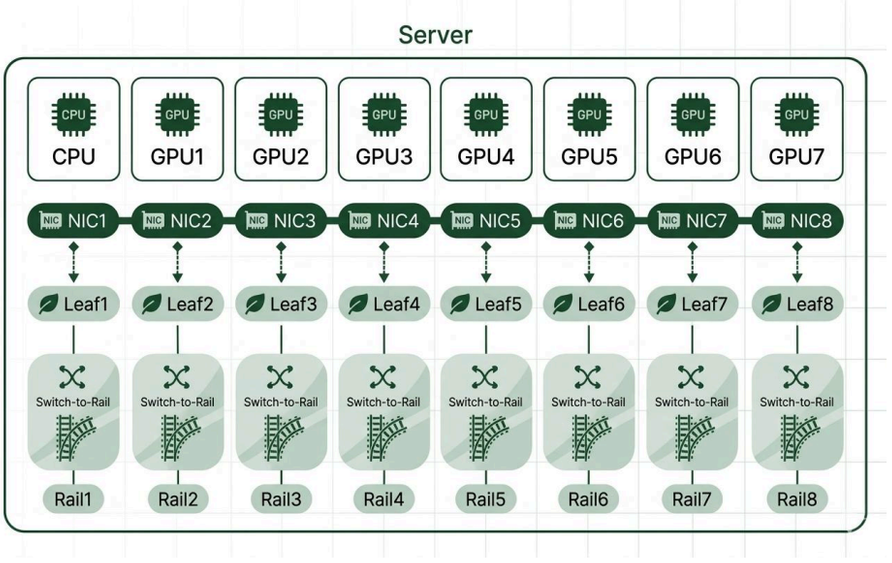


Two patterns show up repeatedly.

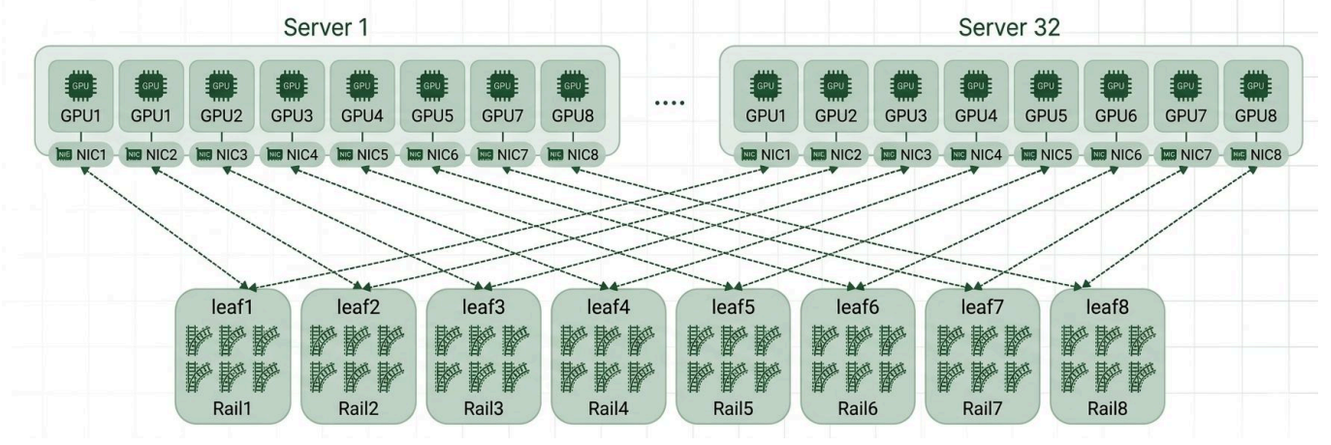
Rail-Optimized Design (ROD)

Each GPU maps to a different leaf switch i.e. one GPU per “rail” to keep intra-rail latency minimal. Eight leaf connections for an 8-GPU server forms a stripe/row pattern. You should choose ROD when you are optimizing for high-scale training efficiency and want clearer fault isolation boundaries, because a leaf failure impacts a rail rather than the entire host. The trade-off is that cabling intensity increases fast and operational discipline becomes non-negotiable; your port mapping, labeling, and validation processes must be treated as part of the platform, not as “build-time details.”

Figure below displays the mapping of GPUs to a rail.

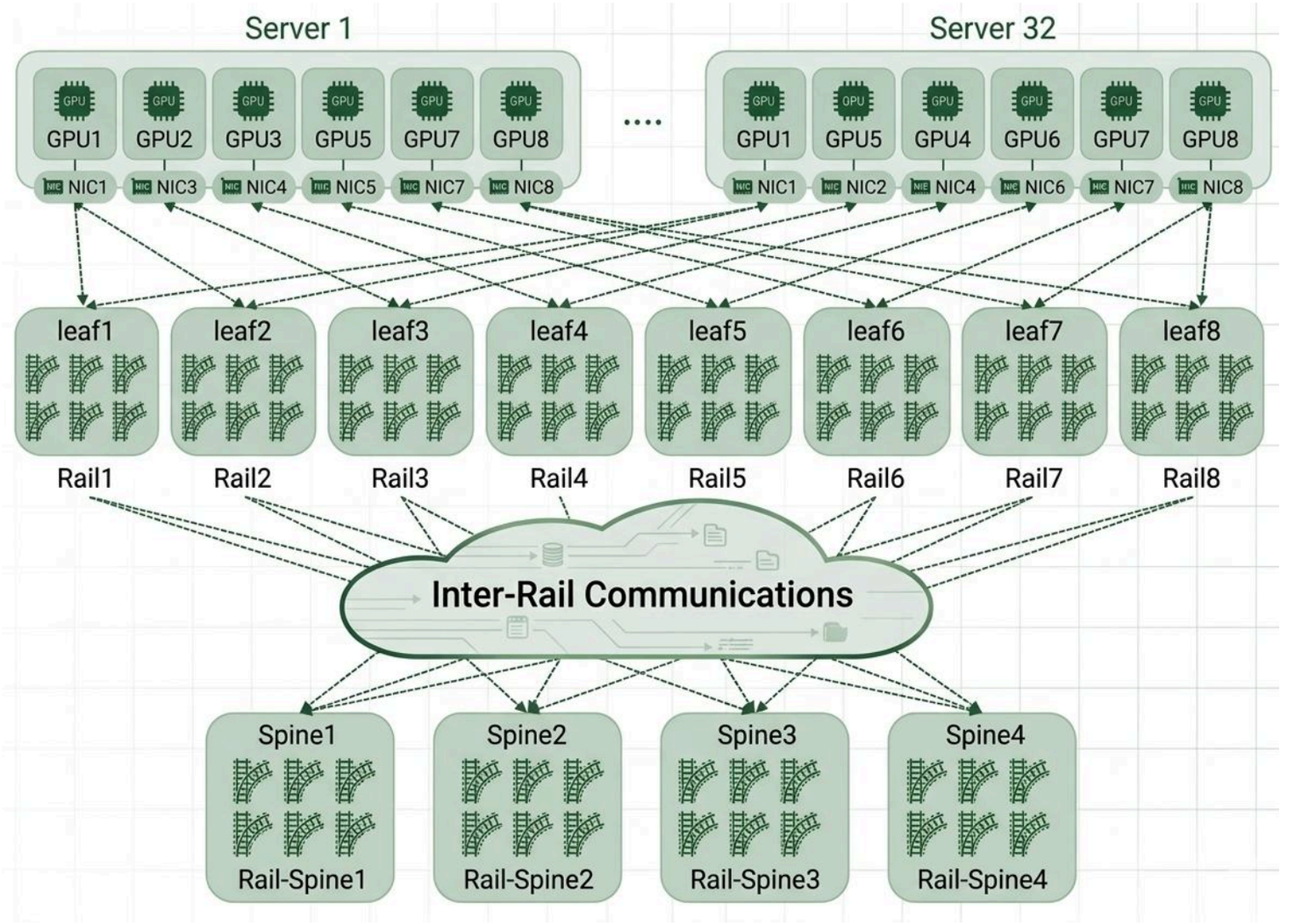


To keep the latency within a rail to a minimum, each GPU is connected to a leaf switch; this is referred to as rail-optimized topology. Eight leaf nodes connecting to 8 different GPUs is referred to as a row, or a stripe. Each leaf switch can establish connections with several servers. For instance, a 64 × 400 Gbps switch with a 1:1 oversubscription ratio can have 32 × 400 Gbps links toward the servers and an additional 32 × 400 Gbps links toward the spine, as illustrated in Figure below.



Rail-Optimized Design (ROD)

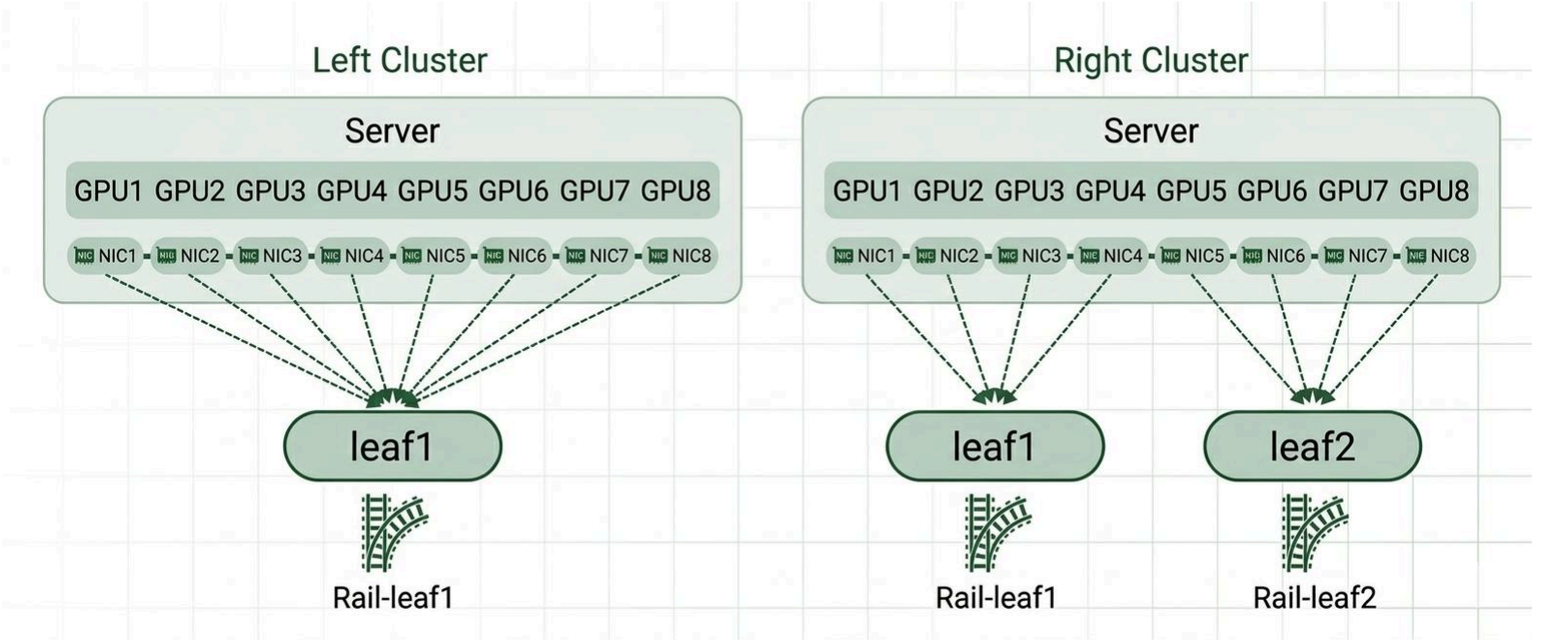
Note that this topology enables switching only within the same rail—in other words, on the same GPU number on all the servers through the leaf; this is called intra-rail communication. There is minimal latency within a rail. For communication between different GPUs across rails, the traffic needs to flow via the spine layer, as shown below. This is called inter-rail communication, and its latency is doubled compared to the latency in intra-rail communication.



Rail-Unified Design (RUD)

Multiple GPUs connect to the same leaf (all 8 to one leaf, or 4+4 to two leaves). You should choose RUD when your deployment scale, build velocity, and cabling simplicity matter most, and when the fault domain trade-off is acceptable. The operational implication is straightforward: a leaf failure can isolate an entire server, and performance/fault domains become “fatter,” which can be perfectly reasonable in smaller pods but becomes a sharper risk as training jobs and costs scale.

You can connect all 8 GPUs of a server to one leaf switch, you can connect 4 GPUs to two leaf switches, and so on, as illustrated below. The advantage of this topology, called rail-unified design (RUD), is simplified cabling, easier scaling, and well-established architecture. The challenge is that if the single leaf goes down, the server is completely isolated.



At higher scale, the fabric tends to evolve beyond a single three-stage Clos into five-stage/seven-stage Clos, block/brick designs, and sometimes alternative topologies (Dragonfly, Torus) to manage hop count, diameter, and cost trade-offs. Emerging approaches like scheduled fabrics (cell-based switching + VOQ concepts) aim to make congestion behavior more predictable under synchronized AI traffic. In a sovereign framing, this is not “topology fashion”; it’s the platform’s attempt to turn collective-heavy traffic into predictable, governable behavior.

Rack Design:

ToR vs MoR vs EoR is Also an Optics Decision

Rack design is often presented as “where do I put the switch?” In AI factories, it’s equally “what do I do with the cables?”

Top-of-Rack (ToR) favors short intra-rack runs and simpler servicing, and it can be particularly effective where you can exploit DAC/AEC and keep build complexity low. Middle-of-Row (MoR) centralizes switching and can reduce heat in compute racks, but it increases cross-rack cabling and pathway complexity, which increases the need for structured cabling discipline. End-of-Row (EoR) shifts even more weight onto optics and pathway planning; it may simplify certain maintenance patterns but typically increases link budget exposure and operational risk if documentation and validation aren’t rigorous.

In other words: topology choices and rack placement decisions silently determine your optics mix, your link budgets, your pathway utilization, and the probability of human error. In sovereign deployments, those “silent” decisions are exactly where reliability and auditability are won or lost.

Optics and Cable Management: The Hidden Layer That Decides Reliability, Cost, and Build Speed

In AI fabrics, optics and cabling are not “procurement details.” They are part of the architecture because they drive time-to-deploy, failure rates, power, operational clarity, and supply-chain assurance.

Picking the right interconnect type (DAC/AEC vs optical)

You should default to the simplest interconnect that meets your reach and density requirements, because simplicity is a reliability feature. DAC is often cheapest and simplest for very short reaches, but it can become physically unwieldy at high density and has strict distance limits. AEC extends copper usability with better signal integrity while keeping copper-like operations. Optical transceivers and fiber become required as soon as distance, pathway routing, or density makes copper impractical; this is common in MoR/EoR designs and larger pods where structured cabling is unavoidable.

Form factors and breakouts become topology tools

As fabrics move from 400G to 800G, you don’t just “upgrade speed”, you redesign how ports are consumed. Multi-planar designs, breakout strategies, and rail-aligned mapping decisions become architectural constraints that you must standardize and enforce. If you multiply cables per host via breakouts and planes, unmanaged cabling becomes a reliability problem and a deployment bottleneck, which directly undermines the “factory” goal of repeatable scale.

Rack Design:

ToR vs MoR vs EoR is Also an Optics Decision

Link budgets and cleanliness are day-0 requirements, not day-2 fixes

At scale, many “network issues” are really optical hygiene issues: connector contamination, improper mating and polarity mistakes, bend radius violations, pathway crush, excessive insertion loss from patching, and confusing patch-panel layering that hides root cause. In a sovereign AI factory, you should operationalize optical discipline as a standard workflow: inspect and clean before mating, document and enforce loss budgets, standardize patching layers so troubleshooting is deterministic, and trend DOM/DDM telemetry because slow power drift is often the earliest warning signal you will get.

Optics power is part of the facility budget

At high densities, transceiver power becomes meaningful. When you are counting kilowatts per rack and fighting for cooling headroom, optics power is no longer invisible. Wherever feasible, you should evaluate approaches that reduce optics power draw and heat output, because the downstream effect is not only lower consumption but better stability and more usable headroom for compute.

Cable management is governance

Sovereignty is about verifiability, and cabling discipline is part of the evidence chain. You should be able to trace, audit, and prove what is connected where, what changed, when it changed, and under whose authority. Serial-number and vendor provenance matter in regulated partitions. Standard labeling matters because it reduces human error under pressure. Golden port-maps and as-built validation tied into your CMDB matter because they turn troubleshooting from guesswork into repeatable diagnosis. Controlled change windows matter because unscheduled “quick fixes” are where evidence chains break.

If you can’t prove which cable/optic is connected where and when it changed, you don’t really have a sovereign, auditable factory. You have a fragile one.

The Storage Layer:

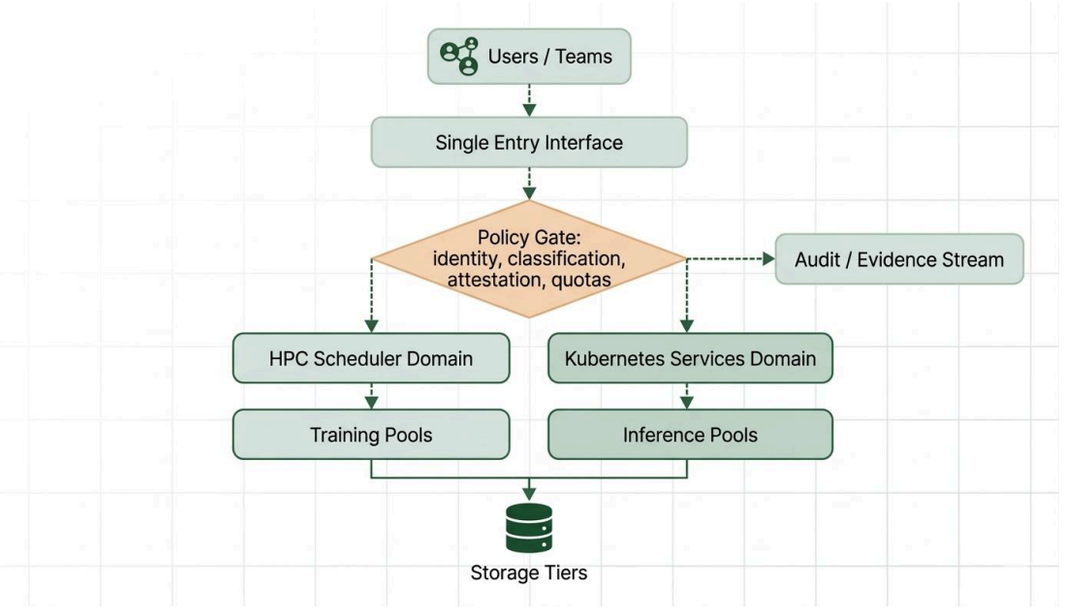
Data Gravity, Tiering, and Policy-Driven Placement

AI is ultimately a data system. Storage determines whether GPUs are fed consistently and whether provenance can be proven.

Sovereign storage architectures typically embrace tiering: object storage for datasets, model artifacts, logs, and retention; high-performance shared or parallel file systems for training pipelines and coordinated checkpointing; NVMe tiers, node-local or shared, as burst buffers to absorb checkpoint storms and hot working sets. The sovereign requirement is that performance and governance must coexist: dataset versioning and lineage, artifact signing and traceability, and policy-driven placement that controls what can cache where, who can mount what, and under what attested conditions.

The Orchestration Layer: The Operating System of Sovereignty

Without a strong orchestration layer, you don't have a sovereign platform, you have expensive hardware plus manual processes.



Sovereign platforms must reconcile two operating models: HPC-style training with gang scheduling, topology awareness, and predictable allocation; and cloud-native inference/services with declarative deployments, self-healing, and autoscaling. A unified control plane becomes a policy-aware broker: users submit work through one interface; the platform routes it to the right substrate based on workload type, data classification, required capabilities, and governance constraints. It also enables dynamic capacity management: reallocating nodes between partitions, enforcing quotas, integrating checkpointing with preemption, and maintaining consistent identity and audit across both worlds.

The Governance Layer:

Policy-as-Code and Evidence by Design

Governance is what turns “private AI” into “sovereign AI.”

Policy-as-code enforces residency constraints, allowed runtimes, encryption requirements, network segmentation, and attestation prerequisites at admission time. Context-aware access goes beyond simple RBAC by incorporating device posture, network zone, workload classification, approval state, and time windows. Auditability and provenance are mandatory: dataset access, model promotion, policy changes, administrative actions, and artifact movement should produce structured events that can be correlated end-to-end.

In a deployment sovereignty framing, governance is not a compliance wrapper you apply later. It is a set of platform guarantees: what is enforced, what is logged, what is provable, and what remains outside your control.

Deployment Models:

Fortress, Sovereign Cloud, Hybrid Sovereign

Sovereign AI is not one-size-fits-all. Organizations choose deployment models based on sensitivity, operational maturity, and economics. The trade-offs between a fully air-gapped on-premises factory, a sovereign cloud with managed operations, and a hybrid model that classifies workloads across boundaries are significant enough to deserve a dedicated treatment.

We hear this question consistently from the market: "Why should we not just use XYZ sovereign public cloud instead of building on-premises?" It is exactly the right question to ask, and the honest answer depends on what you are actually trying to prove, to whom, and under what conditions. The comparison across control-plane exposure, performance determinism, evidence continuity, dependency risk, and operational burden tells a far more nuanced story than the marketing around any single deployment model.

We will cover that in full in the next post in this series, where we break down each deployment model across the dimensions that matter most for regulated and strategic AI workloads.

Conclusion

Sovereign AI infrastructure is the blueprint for the next generation of strategic digital capability. It's the AI Factory mindset: engineered power and cooling envelopes, deterministic fabrics, tiered storage designed around checkpoint economics, orchestration that unifies HPC and cloud-native execution, and governance that enforces policy while producing evidence.

At Open Innovation AI, we work with organizations navigating exactly this journey. If you are thinking through your sovereign AI strategy or want to pressure-test your current architecture, we would love to have that conversation. Reach out to the Open Innovation AI team and let us know where you are in the journey.

References

NVIDIA Hopper architecture whitepaper (GTC22): https://www.advancedclustering.com/wp-content/uploads/2022/03/gtc22-whitepaper-hopper.pdf

NVIDIA Ampere architecture whitepaper: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

Torus interconnect overview: Torus interconnect

ABOUT OPEN INNOVATION AI

Leading the Future of AI Infrastructure

At Open Innovation AI, we design solutions that bridge the gap between research and real-world AI deployment, making AI faster, more secure, and enterprise-ready.

Author

Oluwatosin Aramide
Senior Network Engineer
Linkedin.com/in/iamaramide/

Senior Network Engineer with 16+ years designing large-scale data center networks, AI infrastructure and hyperscaler GPU fabrics. Expertise in InfiniBand, RoCEv2, Kubernetes networking, and automation. Published researcher. Design-first, security built-in.

Visit openninnovation.ai to learn more