

Sovereign AI Architecture

Dal modello AI al Sovereign AI Execution Layer

Autore: Andrea Ciofani

Reviewed by AI

Abstract

La discussione sull'AI enterprise è ancora spesso concentrata sulla scelta del modello: quale LLM utilizzare, quale possiede le migliori capacità di reasoning, quale offre il costo per token più basso o quale raggiunge i risultati migliori nei benchmark.

Questa impostazione rischia di diventare rapidamente obsoleta.

In un'infrastruttura AI moderna il modello è soltanto uno dei componenti del sistema. La capacità realmente strategica consiste nel poter utilizzare **modelli differenti, specializzati e sostituibili**, orchestrandoli dinamicamente in funzione del problema da risolvere.

Se questa architettura viene realizzata interamente all'interno di infrastrutture controllate dall'organizzazione, emerge un paradigma ulteriore:

Sovereign AI Architecture.

Non significa semplicemente installare un LLM su un server locale.

Significa controllare l'intera catena di esecuzione dell'intelligenza artificiale:

**Compute → Models → Routing → Agents → Context → Memory → Tools → Data →
Execution → Observability → Governance.**

L'obiettivo non è quindi costruire un'infrastruttura attorno a un determinato modello.

È costruire un'infrastruttura nella quale i modelli diventano risorse computazionali intercambiabili.

Guida alla lettura

Il documento segue integralmente la struttura concettuale della fonte, con un percorso dalla dipendenza dal singolo modello alla gestione delle capacità AI come infrastruttura.

Fondamenti e livello di esecuzione

01 Dal modello all'infrastruttura	3
02 Perché Sovereign AI	4
03 Il Sovereign AI Execution Layer	5

Routing e orchestrazione

04 Il Model Router	6
05 Il router deve diventare intelligente	7
06 Agent Orchestrator	8

Contesto strumenti e compute

07 Sovereign RAG	9
08 Tool Gateway e MCP	10
09 Agent Sandbox	11
10 GPU Cluster	12

Sicurezza osservabilità e controllo

11 Security by architecture	13
12 Data classification e Sovereign Routing	14
13 Observability ed Evaluation	15
14 Control Plane e Data Plane	16

Architettura integrata e AI Factory

15 Architettura complessiva	17
16 Il vero significato di AI Sovereignty	18
17 Dal GPU Cluster alla Sovereign AI Factory	19
18 AI come infrastruttura	20
19 Conclusione	21

Fonte e convenzioni

Fonte unica: Testo incollato.txt, Sovereign AI Architecture. Testo, esempi, tecnologie e criteri provengono dal file allegato; non sono stati aggiunti fatti esterni. Le didascalie e i raggruppamenti visivi sono elaborazioni editoriali.

Le frecce riproducono relazioni concettuali, flussi o dipendenze della fonte; non specificano protocolli, tempi o una topologia implementativa. I diagrammi sono incorporati come immagini vettoriali con fallback ad alta risoluzione; gli SVG originali sono forniti separatamente per modificarli.

Dal modello all'infrastruttura

La prima generazione di applicazioni basate su LLM seguiva generalmente un'architettura estremamente semplice:

L'applicazione dipendeva direttamente dal modello e dal provider.

La generazione successiva ha introdotto RAG, vector database, function calling e agenti:

Il passo successivo consiste nel separare definitivamente l'applicazione dal modello.

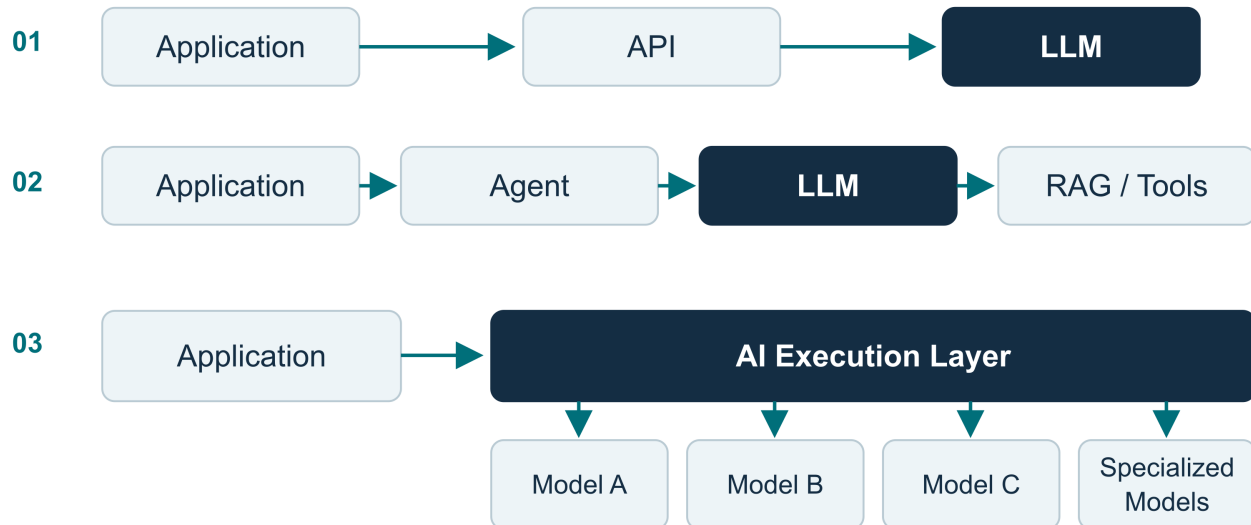


Figura 01 Tre generazioni di architettura applicativa

L'applicazione non deve conoscere quale modello verrà utilizzato.

Deve dichiarare **che cosa deve essere fatto**.

L'infrastruttura decide **come farlo**.

Questo cambia radicalmente il ruolo dell'LLM.

Il modello smette di essere la piattaforma e diventa una risorsa della piattaforma.

Perché Sovereign AI

La dipendenza esclusiva da API esterne introduce una serie di problemi strutturali per organizzazioni che trattano informazioni sensibili, strategiche o regolamentate.

Tra questi:

- trasferimento dei dati verso infrastrutture esterne;
- dipendenza tecnologica dal provider;
- variazioni imprevedibili dei prezzi;
- rate limit;
- indisponibilità del servizio;
- modifica o ritiro dei modelli;
- impossibilità di controllare completamente il ciclo di inferenza;
- difficoltà nella gestione di informazioni classificate o proprietarie;
- requisiti di data residency;
- difficoltà nel garantire una completa auditabilità.

La risposta più semplice sembrerebbe essere installare un modello open-weight in locale.

Ma questo risolve solo una parte del problema.

Un LLM on-premise isolato non costituisce una vera infrastruttura AI sovrana.

La sovranità richiede il controllo dell'intero **AI execution stack**.

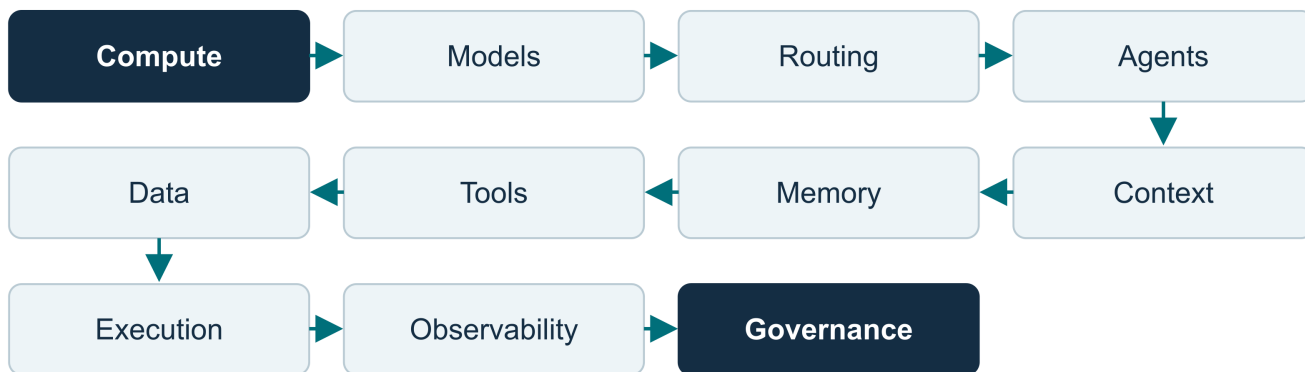


Figura 02 La sovranità riguarda l'intera catena di esecuzione

Il Sovereign AI Execution Layer

L'architettura proposta introduce un livello intermedio tra applicazioni e infrastruttura AI.

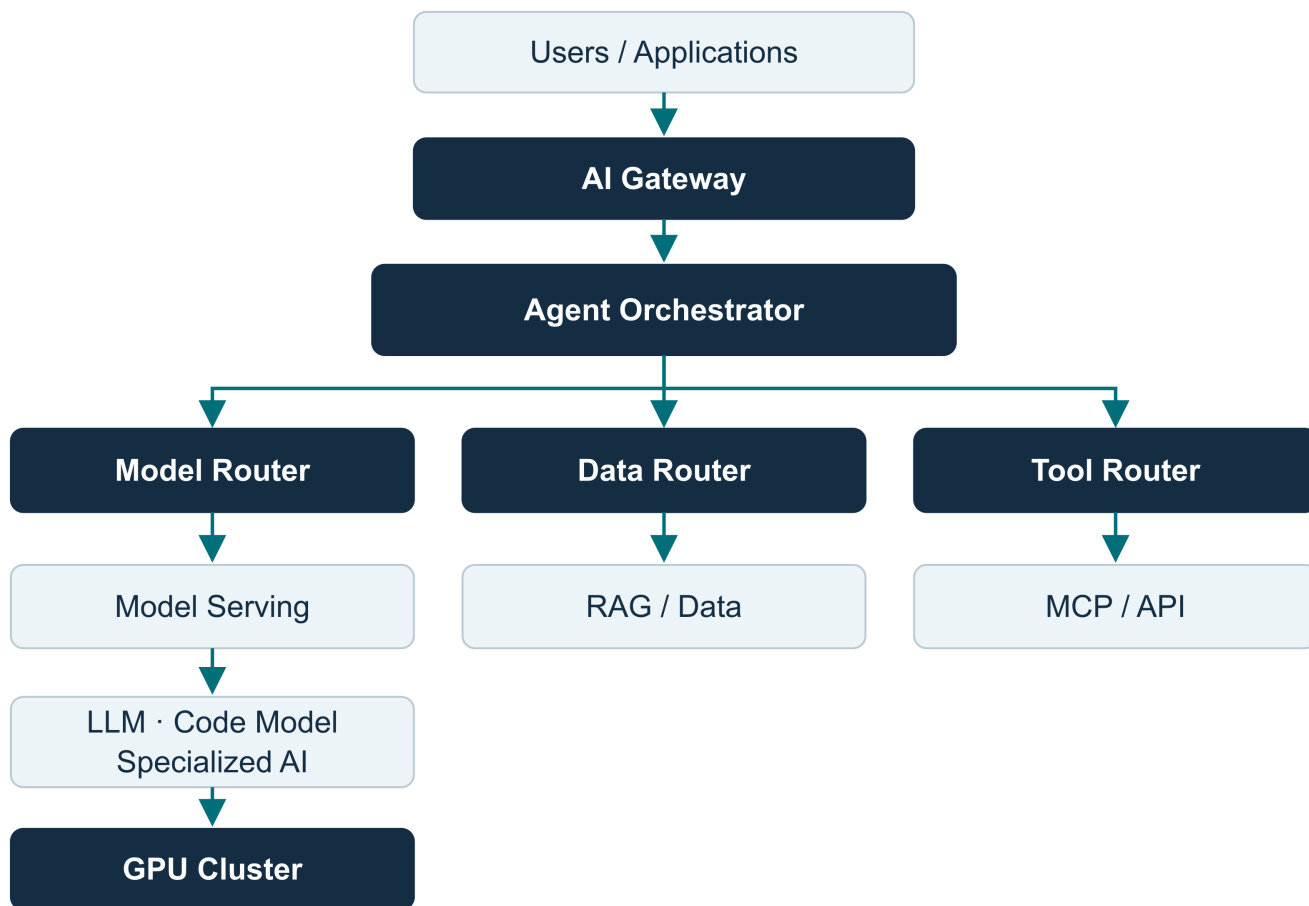


Figura 03 Execution Layer tra applicazioni modelli strumenti e dati

Tutti questi componenti possono essere eseguiti:

on-premise · in private cloud · in sovereign cloud · oppure attraverso una combinazione controllata dei tre.

Il requisito fondamentale è che la governance dell'infrastruttura rimanga sotto il controllo dell'organizzazione.

Il Model Router

Il cuore dell'architettura è il **Model Router**.

Non esiste infatti un modello migliore in assoluto.

Un modello può essere particolarmente efficace nel reasoning, un altro nel coding, un altro nell'elaborazione documentale, un altro ancora può essere estremamente veloce ed economico.

Un'infrastruttura potrebbe quindi ospitare contemporaneamente famiglie di modelli come:

Famiglie di modelli citate

Qwen · Llama · Mistral · DeepSeek · GLM · Kimi · Gemma · modelli proprietari fine-tuned

insieme a modelli specializzati per:

Modelli specializzati

embedding · reranking · OCR / vision · speech-to-text · text-to-speech · classification · computer vision · anomaly detection

Il router seleziona dinamicamente la risorsa appropriata.

La decisione può considerare:

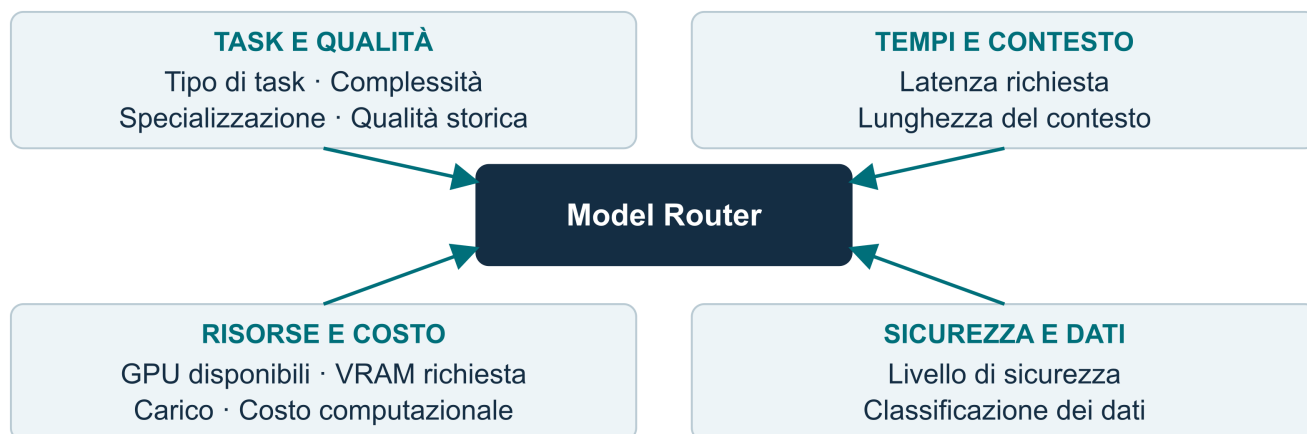


Figura 04 Criteri di selezione del modello raggruppati per lettura

Una richiesta semplice potrebbe essere indirizzata verso un modello da 7–14B parametri.

Un problema complesso potrebbe essere assegnato a un modello molto più grande.

Un task di programmazione potrebbe essere indirizzato verso un modello specializzato nel coding.

Il risultato è un utilizzo molto più efficiente dell'infrastruttura GPU.

Il router deve diventare intelligente

Una prima implementazione può utilizzare regole statiche:

Condizione esemplificativa	Destinazione
task = coding	coding-model
classification = confidential	secure-model-pool
context > 100k	long-context-model
task = simple	small-fast-model

Ma questa è soltanto la fase iniziale.

Il router può evolvere in un sistema capace di apprendere dalle esecuzioni precedenti.

Per ogni richiesta possono essere registrati:

Dati registrabili per richiesta

model selected · latency · tokens · GPU time · VRAM utilization · task success · evaluation score · user feedback · energy consumption

Il routing diventa quindi un problema di ottimizzazione.

L'obiettivo può essere espresso concettualmente come:

massimizzare qualità e affidabilità minimizzando latenza, consumo GPU e costo computazionale.

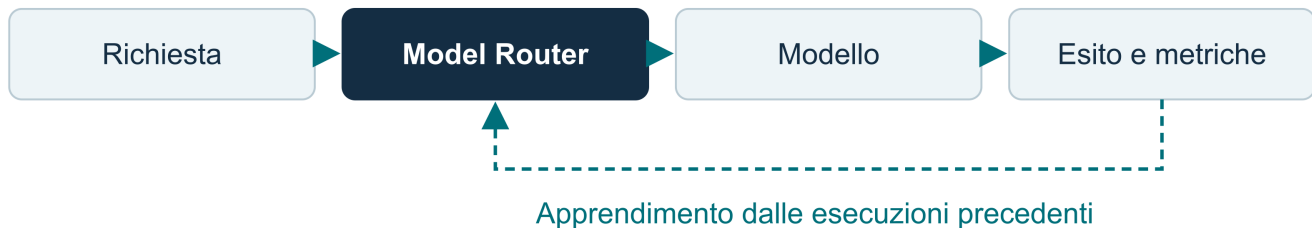


Figura 05 Evoluzione concettuale dal routing statico al routing adattivo

Agent Orchestrator

Il Model Router risolve solo una parte del problema.

Una richiesta complessa deve spesso essere suddivisa in più attività.

Per esempio:

Analizza i documenti del progetto, identifica i rischi contrattuali, confrontali con la normativa interna e prepara un report per la direzione.

Il sistema potrebbe decomporre automaticamente il problema:

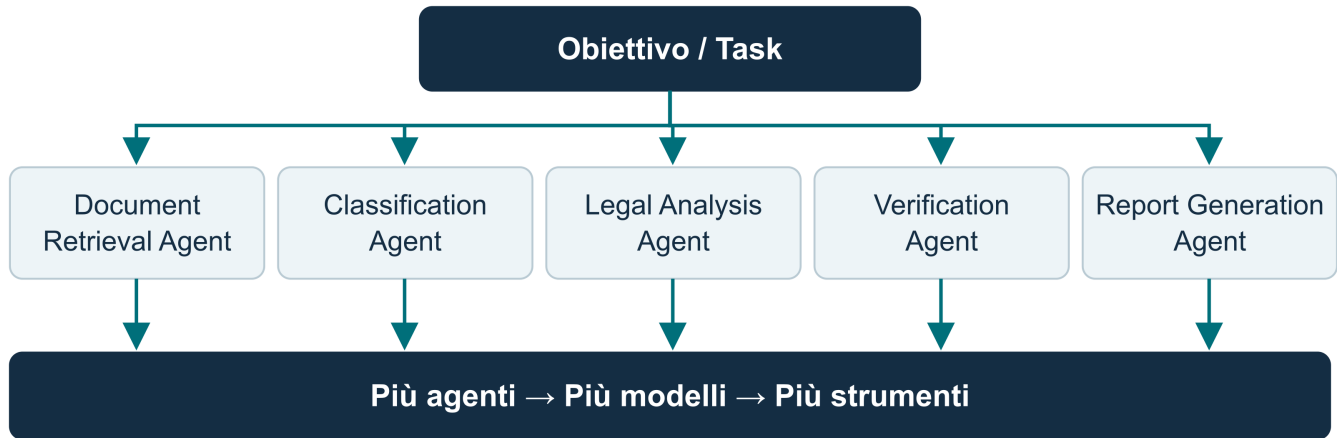


Figura 06 Decomposizione esemplificativa di un obiettivo in agenti

Ogni agente può utilizzare un modello differente.

Si passa quindi da:

one request → one model

a:

one objective → multiple agents → multiple models → multiple tools.

Sovereign RAG

I modelli non dovrebbero contenere tutta la conoscenza aziendale.

La conoscenza privata deve rimanere separata dal modello.

Il layer RAG può integrare:

Sorgenti integrabili

Document Management · Database · ERP · CRM · Knowledge Base · Email · SharePoint · File Server
· Data Lake · Telemetry · Technical Documentation

attraverso una pipeline controllata:

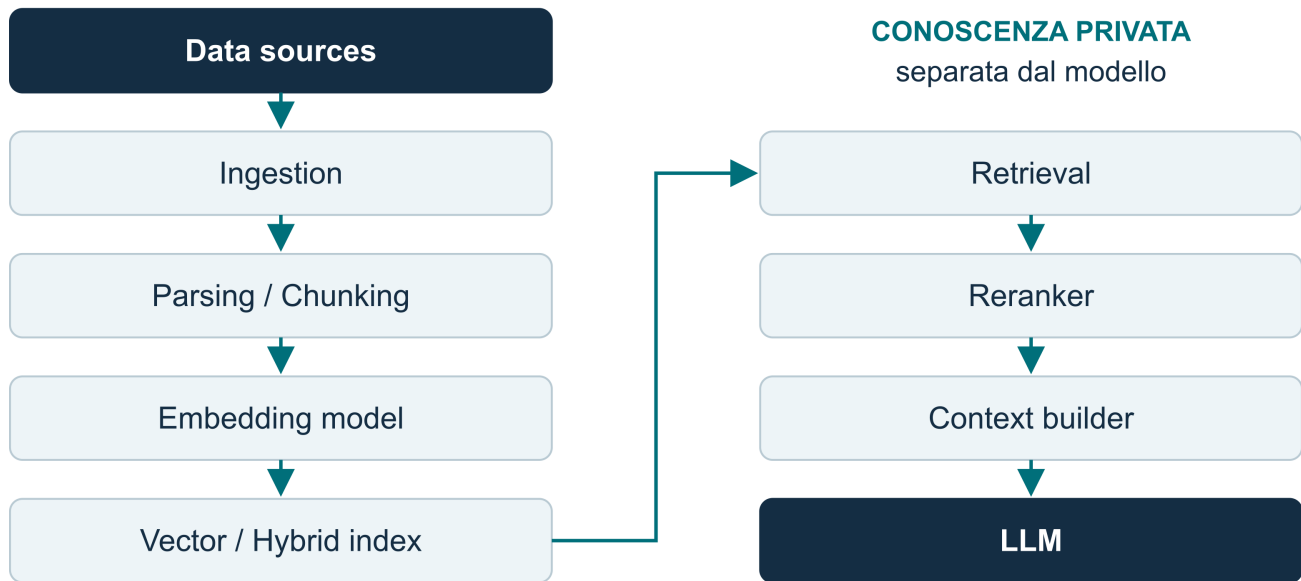


Figura 07 Pipeline Sovereign RAG nell'ordine della fonte

Anche embedding model e reranker possono essere eseguiti localmente.

Nessun documento deve necessariamente lasciare l'infrastruttura.

Tool Gateway e MCP

Un agente realmente utile non deve soltanto generare testo.

Deve poter agire sui sistemi.

Il Sovereign AI Layer può quindi integrare un **Tool Gateway**.

Attraverso API interne o MCP possono essere esposti strumenti come:

Strumenti esposti

CRM · ERP · database · ticketing · Git repositories · monitoring · email · document management · Kubernetes · CI/CD · business applications · industrial systems

Ma l'agente non dovrebbe accedere direttamente a questi sistemi.

Il gateway deve applicare:

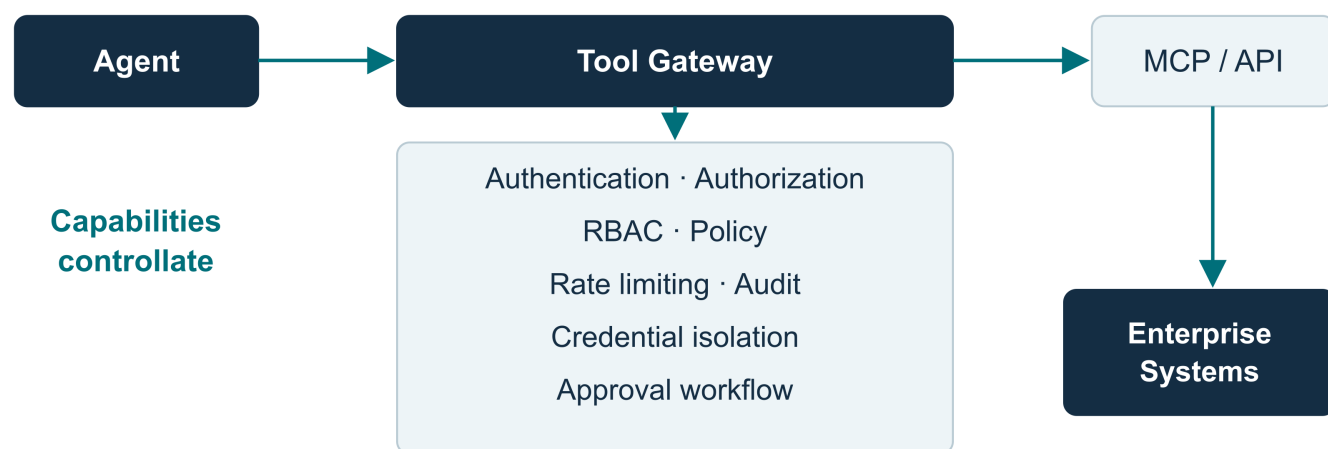


Figura 08 Mediazione del Tool Gateway e controlli di accesso

L'AI non riceve quindi accesso indiscriminato all'infrastruttura.

Riceve capabilities controllate.

Agent Sandbox

Questo introduce un altro elemento fondamentale.

Gli agenti capaci di eseguire codice o utilizzare strumenti non dovrebbero operare direttamente sui server dell'organizzazione.

Ogni workload agentico dovrebbe essere isolato.

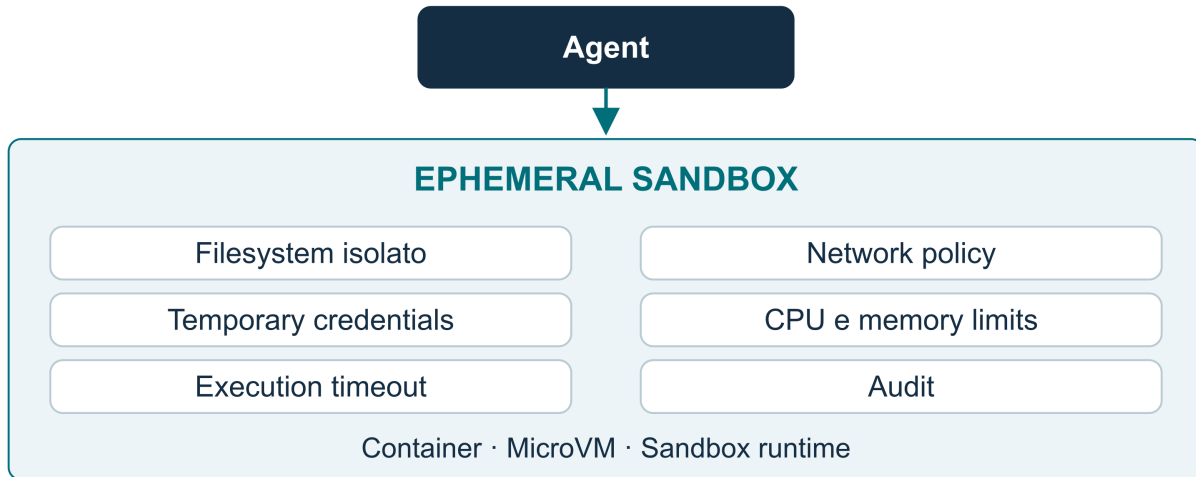


Figura 09 Isolamento del workload agentico nella sandbox effimera

Le tecnologie utilizzabili possono comprendere container, microVM o sandbox runtime.

In un'infrastruttura Kubernetes, la sandbox può diventare la vera **unità di esecuzione dell'agente**.

Questo produce un parallelismo importante:

Kubernetes orchestra workload software.

Il Sovereign AI Layer orchestra **workload cognitivi**.

GPU Cluster

Alla base dell'architettura troviamo il compute.

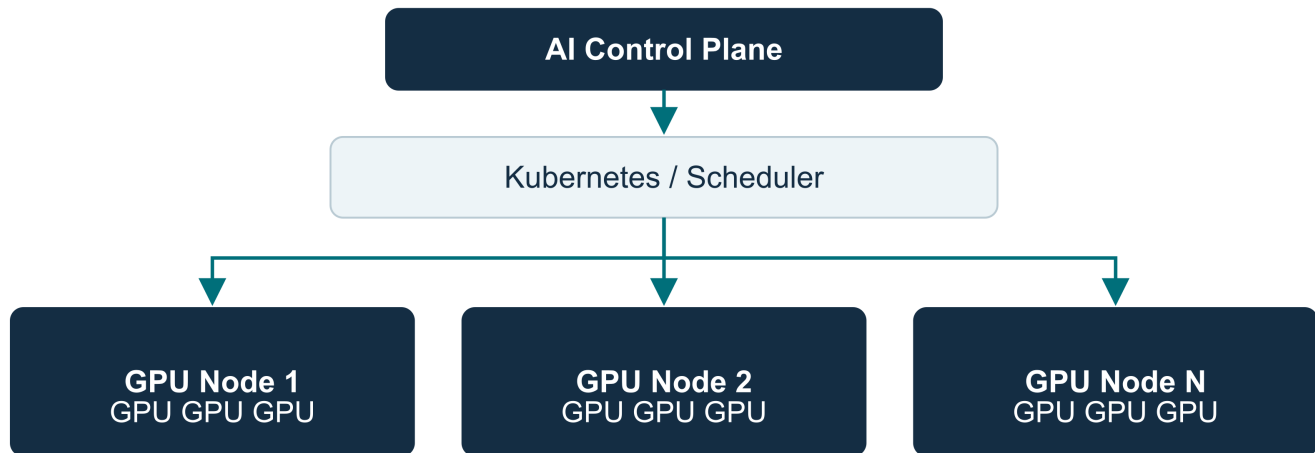


Figura 10 Orchestrazione del cluster con nodi GPU distribuiti

Il serving layer può utilizzare motori specializzati come:

Motori di serving citati

vLLM · SGLang · TensorRT-LLM · Triton Inference Server

Il sistema deve poter gestire:

tensor parallelism · pipeline parallelism · quantizzazione · batching · KV cache · prefix caching · model loading/unloading · GPU scheduling · autoscaling · inference distribution.

In questo modo l'infrastruttura GPU non è legata a un singolo modello.

Diventa un AI Compute Fabric.

Security by architecture

La sovranità non coincide automaticamente con la sicurezza.

Installare un modello nel proprio data center elimina alcuni rischi ma ne introduce altri.

Serve quindi un vero security layer:

Componenti del security layer

Zero Trust · RBAC / ABAC · Secrets Management · Network Segmentation · Encryption · Model Registry · Artifact Signing · Audit Logging · Prompt Injection Protection · Data Loss Prevention · Policy Engine · Supply Chain Security

Particolare attenzione deve essere dedicata alla supply chain dei modelli.

Un modello scaricato da un repository esterno è a tutti gli effetti un **artefatto software eseguibile all'interno dell'infrastruttura**.

Deve quindi essere sottoposto a processi di:

MODEL SUPPLY CHAIN



Il modello entra in infrastruttura come artefatto software

Figura 11 Sequenza di controllo della supply chain dei modelli

Data classification e Sovereign Routing

Un'evoluzione particolarmente importante consiste nell'integrare la classificazione dei dati nel routing.

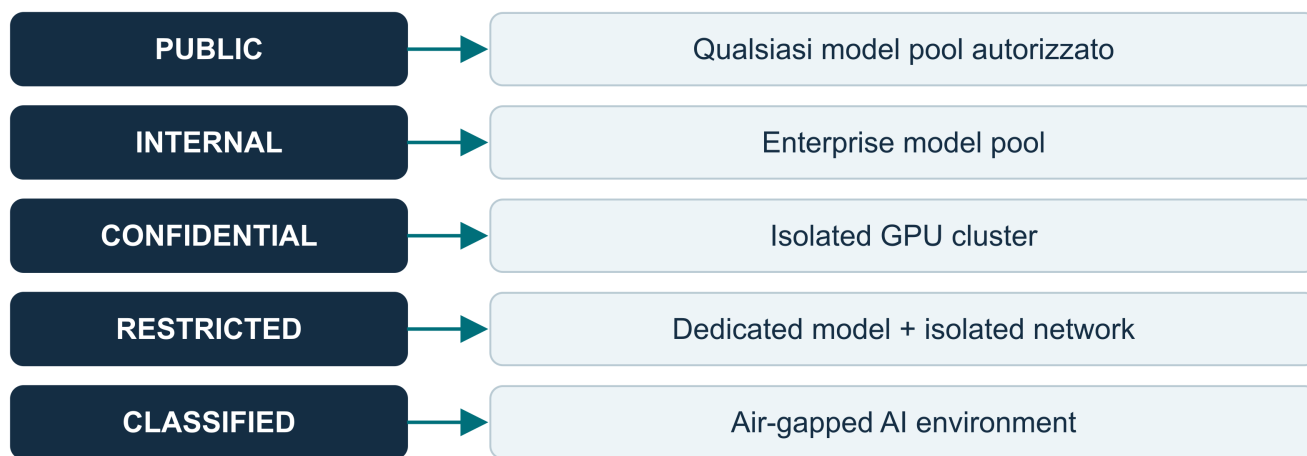
Ogni richiesta può essere classificata, ad esempio, come:

Classificazioni di esempio PUBLIC · INTERNAL · CONFIDENTIAL · RESTRICTED · CLASSIFIED

Il livello di classificazione determina automaticamente:

quali modelli possono essere utilizzati · quali GPU pool · quali dataset · quali strumenti · quali sandbox · quali reti.

Per esempio:



La classificazione determina le risorse consentite

Modelli · GPU pool · Dataset · Strumenti · Sandbox · Reti

Figura 12 Esempi di classificazione e risorse autorizzate

Questo porta il routing oltre il semplice concetto di performance.

Il router diventa anche un Policy Enforcement Point.

Observability ed Evaluation

Un sistema multi-model e multi-agent non può essere gestito come una black box.

Ogni esecuzione deve produrre una trace.

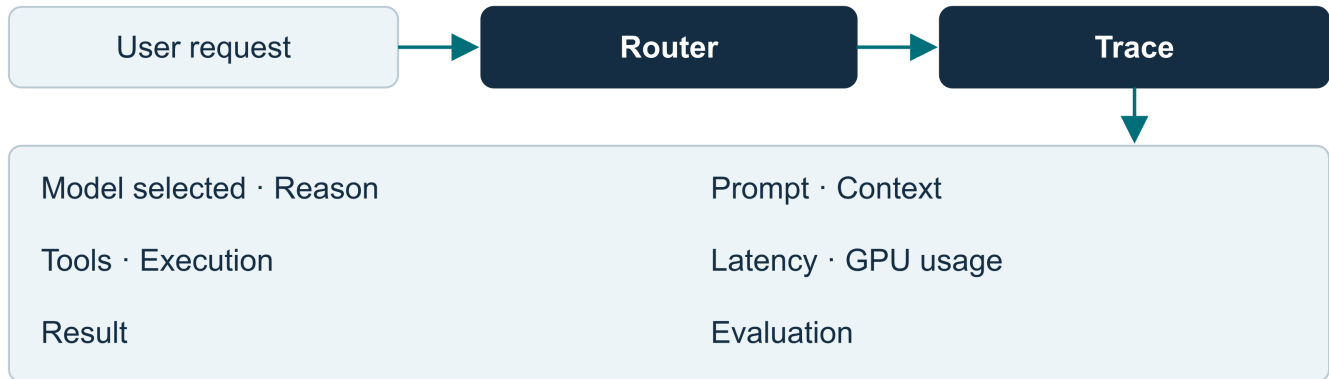


Figura 13 Informazioni associate alla trace di esecuzione

OpenTelemetry e sistemi analoghi possono costituire la base dell'osservabilità.

Ma le metriche tradizionali non sono sufficienti.

Oltre a CPU, RAM, GPU e latency occorre osservare:

Metriche AI da osservare

answer quality · groundedness · hallucination rate · retrieval quality · tool success · agent success · cost per task · GPU seconds per task · energy per task · routing effectiveness

Nasce quindi una nuova disciplina:

AI Infrastructure Observability.

Control Plane e Data Plane

Una possibile architettura enterprise completa può essere organizzata in due grandi livelli.

La separazione permette di controllare centralmente l'intelligenza distribuita nell'organizzazione.

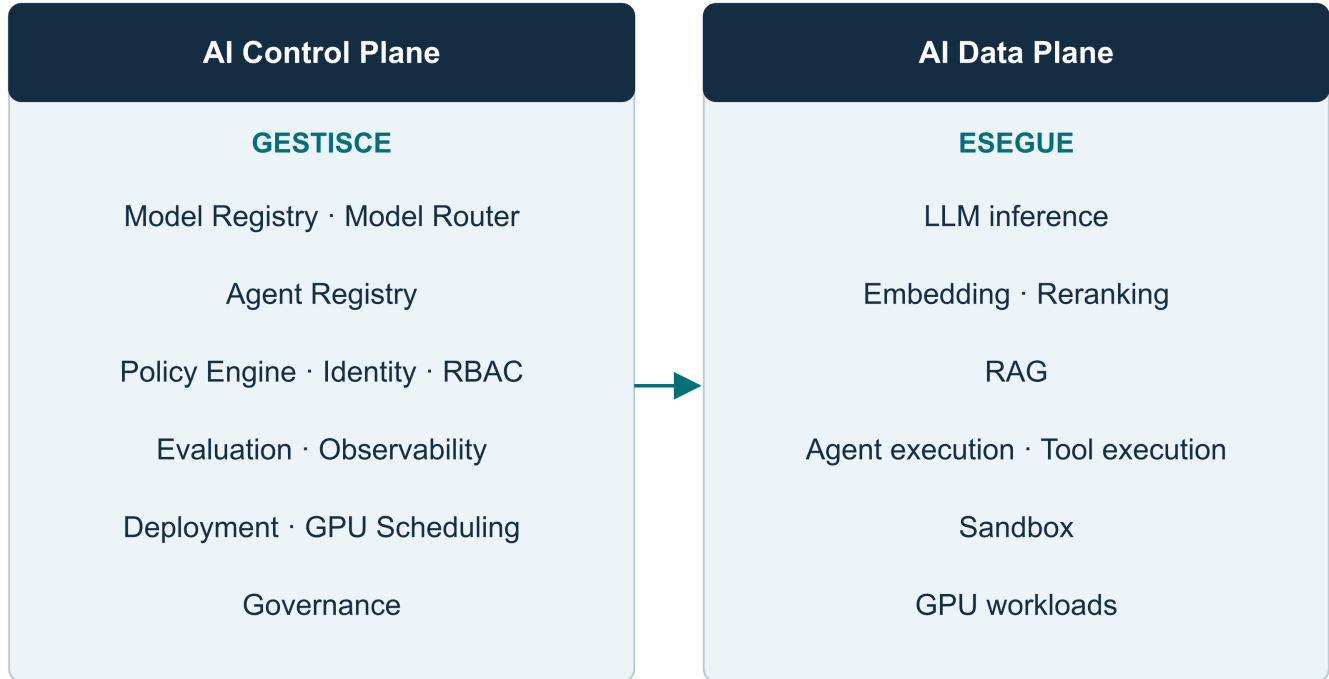


Figura 14 Separazione tra governo centralizzato ed esecuzione

Architettura complessiva

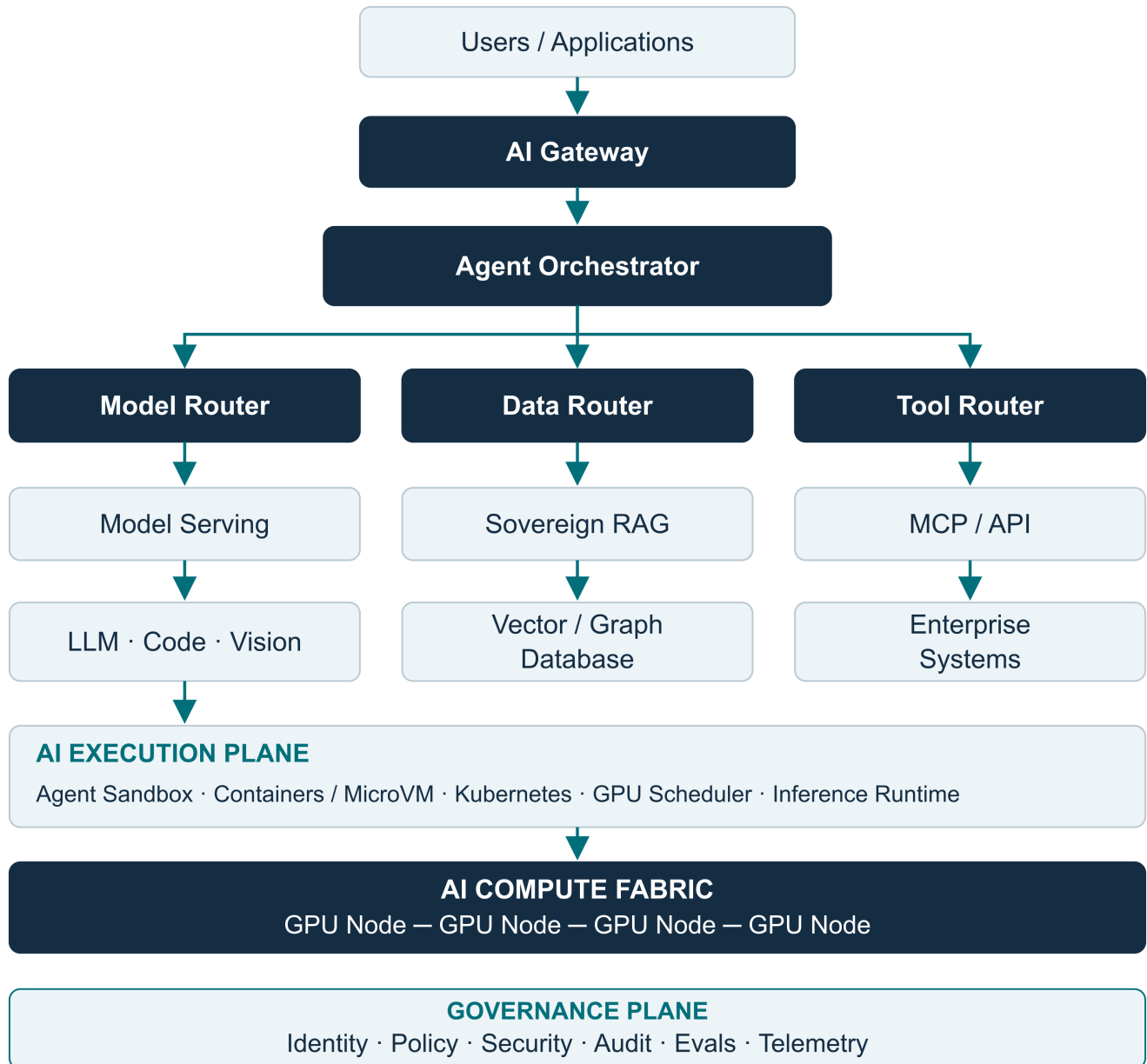


Figura 15 Architettura complessiva e livelli di esecuzione compute e governance

Il vero significato di AI Sovereignty

La sovranità non significa necessariamente rifiutare il cloud.

Significa **non dipendere strutturalmente da esso**.

Una vera architettura sovrana deve poter decidere dove eseguire un workload.

La policy potrebbe stabilire:

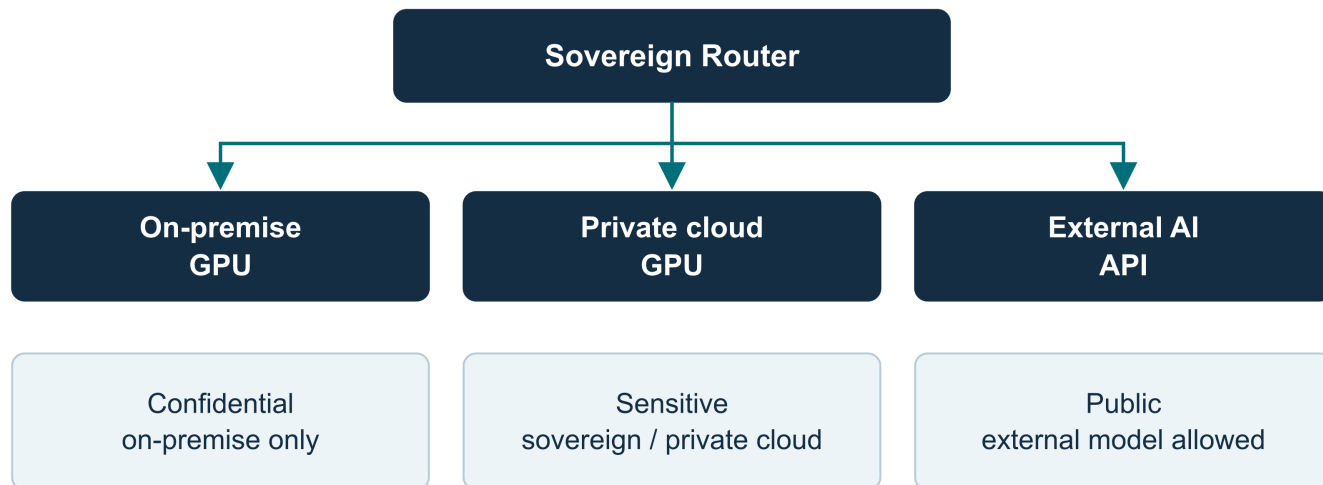


Figura 16 Destinazioni del Sovereign Router e policy esemplificative

Nell'esempio, "Sensitive" è il termine usato dalla fonte per il routing geografico; non viene introdotta una corrispondenza aggiuntiva con le classi della sezione 12.

L'uso di servizi esterni diventa quindi una scelta di routing, non una dipendenza architetturale.

E in un ambiente realmente isolato il ramo "External AI" può semplicemente non esistere.

Dal GPU Cluster alla Sovereign AI Factory

Questa distinzione è strategica.

Comprare GPU e installare alcuni LLM significa costruire un **GPU cluster**.

Costruire:

GPU cluster + inference + model registry + routing + agent orchestration + RAG + MCP + sandbox + observability + governance

significa costruire qualcosa di molto diverso:

una Sovereign AI Factory.

Il suo output non sono token.

Il suo output sono capacità cognitive disponibili come infrastruttura.

Le applicazioni aziendali possono richiedere:

Capability richieste dalle applicazioni

reason() · search() · classify() · extract() · translate() · code() · analyse() · plan() · execute() · verify()

senza conoscere il modello sottostante.

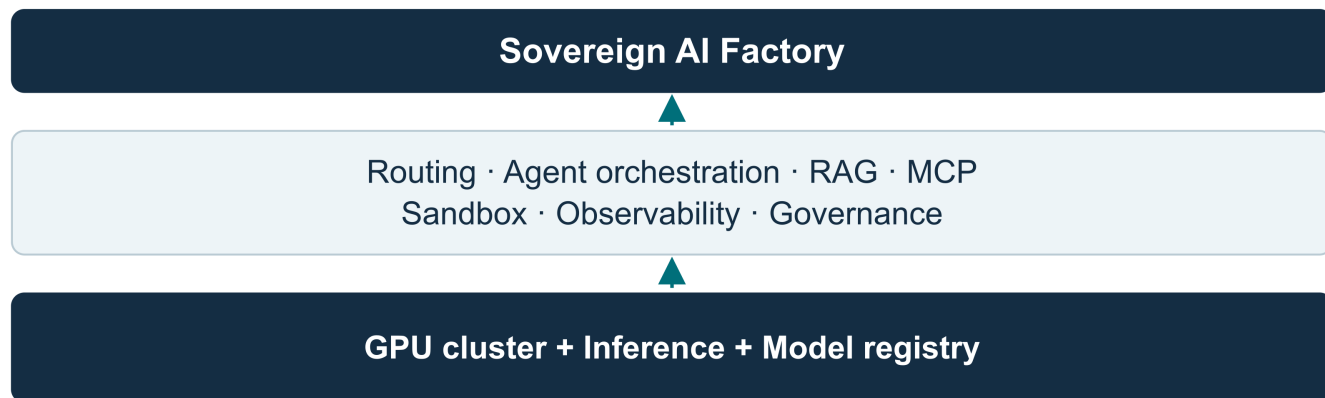


Figura 17 Componenti che trasformano il GPU cluster in AI Factory

AI come infrastruttura

Questo porta alla conseguenza architettuale più importante.

In futuro potremmo non progettare applicazioni "basate su Llama", "basate su Qwen" o su qualsiasi altro modello.

Esattamente come un'applicazione moderna non dovrebbe preoccuparsi del disco fisico sul quale viene memorizzato ogni singolo dato, l'applicazione AI non dovrebbe necessariamente conoscere il modello che eseguirà una determinata operazione.

Dovrebbe richiedere una **capability**.

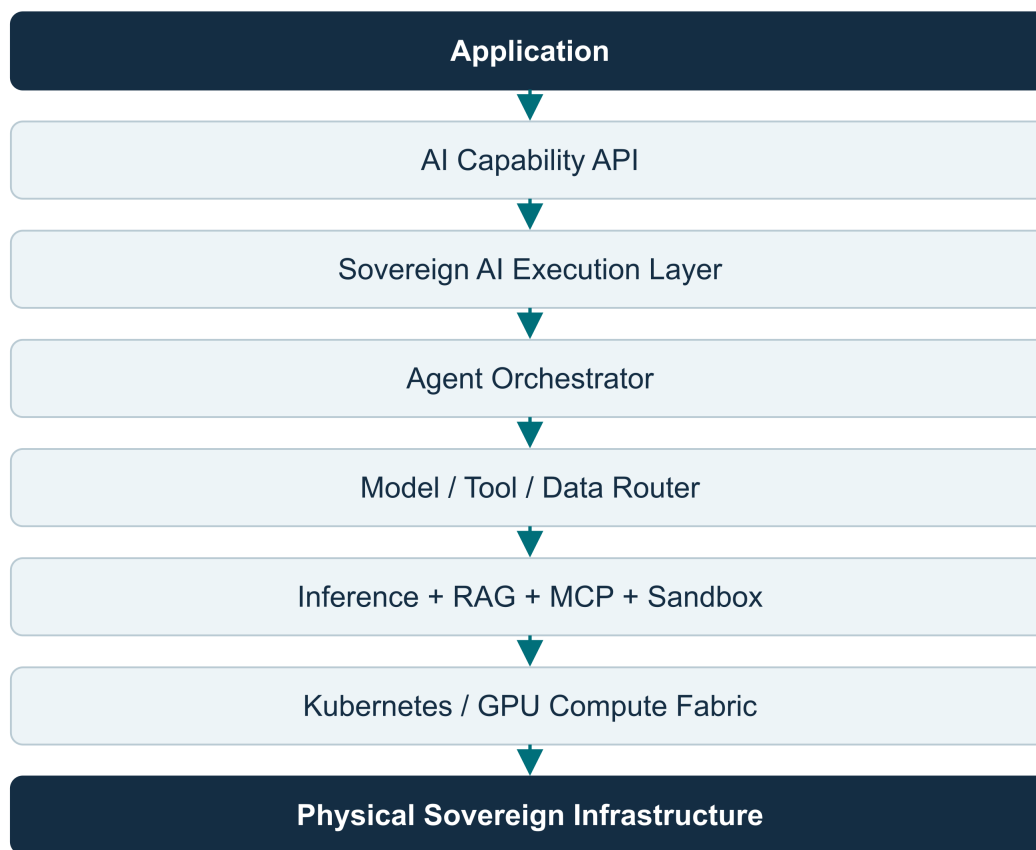


Figura 18 Stack dalla capability applicativa all'infrastruttura fisica

Questa astrazione rende i modelli sostituibili.

Ed è probabilmente uno degli elementi più importanti per proteggere un investimento infrastrutturale pluriennale in un mercato nel quale i modelli cambiano ogni pochi mesi.

Conclusione

La corsa al modello più potente rischia di nascondere un cambiamento architetturale molto più importante.

Gli LLM stanno progressivamente diventando componenti di un'infrastruttura più ampia.

Il vantaggio competitivo non consiste necessariamente nel possedere un singolo modello superiore.

Consiste nella capacità di **orchestrare modelli differenti, dati proprietari, strumenti e risorse computazionali all'interno di un ambiente controllato, osservabile e sicuro.**

Il passaggio fondamentale è quindi:

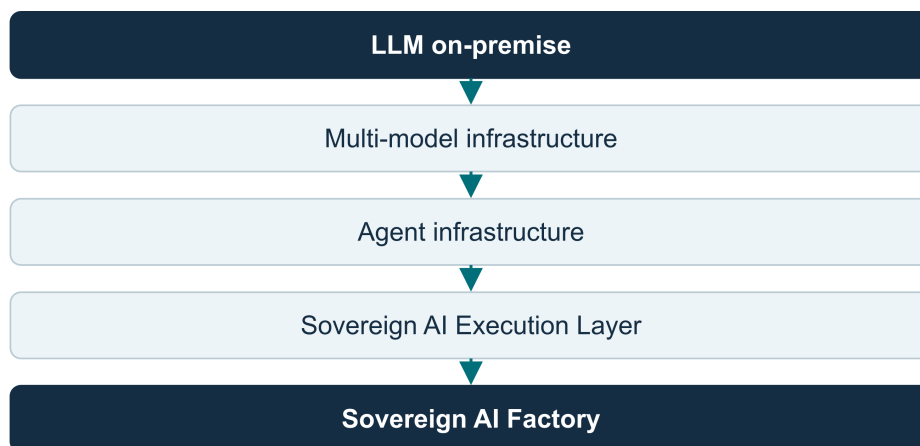


Figura 19 Progressione dal singolo LLM alla Sovereign AI Factory

In questa architettura il modello può cambiare.

L'hardware può evolvere.

Gli agenti possono essere sostituiti.

Nuovi tool possono essere integrati.

Nuove GPU possono entrare nel cluster.

Ma il livello strategico rimane stabile:

l'organizzazione possiede il control plane dell'intelligenza artificiale.

Ed è questa, più della semplice localizzazione fisica dei modelli, la definizione realmente interessante di **AI sovereignty**.