

Persistent Inference Residency Stack

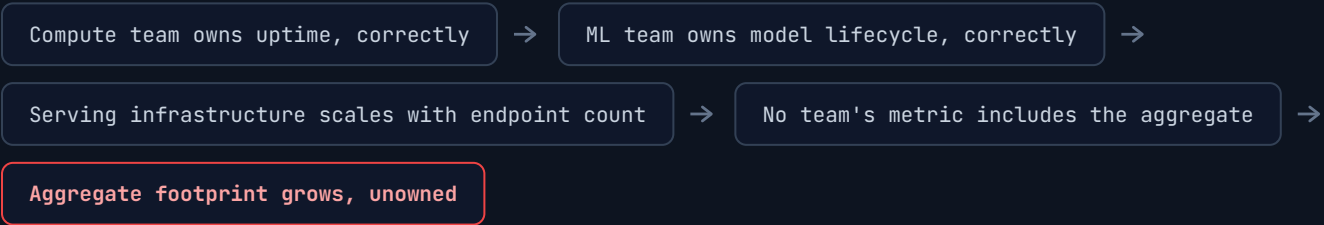
DEFINITION

The set of always-on infrastructure layers — serving, routing, caching, observability — that inference workloads permanently occupy once they reach steady state.

FRAMEWORK FLOW

01 The Condition Production inference footprint breaks into three structurally distinct layers — compute residency, serving infrastructure, and model lifecycle — each with a different owner and different optimization physics.	02 The Boundary Whether the combined residency footprint across all three layers has one accountable owner, or each layer is optimized independently by whichever team happens to touch it.	03 Failure State Diffuse residency ownership — each layer managed correctly against its own metric, but no team's success metric includes the aggregate.	04 Consequence Collapsing three layers into a single "AI spend" line item produces cost reviews that describe the number accurately but can't act on it.
--	--	---	---

Failure State: Diffuse Residency Ownership



THE DIAGNOSTIC SIGNAL

LAYER OWNERSHIP Ask the platform team if they own compute-residency uptime — yes, correctly. Ask the ML team if they own model lifecycle — yes, correctly.	COMPOSITE OWNERSHIP Ask who owns the combined residency footprint across all three layers together — nobody, by absence of design.
TRIGGER CONDITION A model reaches production steady state — the point where warm capacity, serving infrastructure, and lifecycle overhead stop being transitional and become permanent.	REMEDICATION SHAPE Coordinated ownership across three control points: an inference platform team with explicit cost authority, a model-portfolio governance process with entry/exit criteria, and model-level cost attribution.

THE THREE-LAYER RESIDENCY STACK

01 Compute Residency Concurrency reservation, not GPU spend. Owner ambiguity: platform optimizes uptime, ML optimizes throughput, neither owns cost.	02 Serving Infrastructure A permanent operational tax, not platform overhead. Owner ambiguity: platform provisions it, app team depends on it, finance can't see it as a discrete line.	03 Model Lifecycle Multiplicative residency growth, not temporary rollout cost. Owner ambiguity: ML owns model versions, platform owns endpoint lifecycle, neither owns the aggregate.
---	--	---

ARCHITECTURAL RELATIONSHIPS

#3

Inference Residency Creep

RELATED

MODERATE

Residency creep is a growth pattern that emerges inside an unmanaged persistent inference residency stack.

ANCHOR /inference-steady-state-cost/

FRAMEWORK HOME

Defines the structural residency condition created when inference workloads become permanent operational infrastructure.

"Every layer team can correctly explain its own optimization. Nobody can explain the combined residency floor — not because anyone neglected it, but because it was never anyone's to answer."