VERTICAL-DRIVEN ARCHITECTURES · SUPPLY CHAIN & LOGISTICS · DESIGNED WITH PRAXIS · NOVEMBER 2026

A Self-Improving AI System for Carrier Exception Handling in the United States

Lane Memory: A Sovereign AI Reference Architecture for Self-Improving Carrier Exception Handling in Supply Chain and Logistics

Muhammad Umar, Cofounder, CodeNinja

For the operations vice president accountable for exception performance, the exception desk supervisors, and the data, platform and reliability engineers who would build and run it.

Vertical-Driven Architectures is a CodeNinja series of system designs. Every design in the series is driven by a real-world problem and scenario in a single industry, and every one is designed on Praxis, CodeNinja's platform for designing physical AI systems. Operations are described by class, never by name.

ABSTRACT

The Exception Desk Should Get Better Every Week

A self-improving AI system for a third party logistics provider in the United States, one that manages truckload and less than truckload freight for shippers across a network of contracted carriers, must answer one question when an exception lands: what fixed this same exception, on this lane, with this carrier, at this consignee, the last time it happened? The answer exists, but it is buried in years of email and chat threads no system reads, held in the heads of coordinators whose knowledge leaves with them through a high-turnover desk, and scattered across seven systems of record that never meet. Teams evaluating a self-improving AI system for logistics meet exactly this shape: the data is already inside the estate, the improvement loop is not, and shippers now measure the operator on time to resolve and on whether they were told before they asked.

The design reads seven source systems, the transportation management system, the carrier tracking and visibility platform, the appointment scheduling tool, the claims system, the carrier scorecards and contracts, the customer order feeds and the email and chat threads, through one integration adapter family into a governed ontology of sixteen objects joined by typed links, projected never migrated, with a

single gated write path into the scheduling tool behind coordinator approval. Eight services run on that ontology: the network build, threads and memory, sovereign compute, memory and improvement, agent drafting and approval, the command center, data governance, and pilot and extension, delivering triage, recovery and review agents on two surfaces, the exception command center and the weekly recurring-exception review. Three models carry the work: GLM 5.3, a 753B-parameter open-weight model served at FP8 on one node of eight GPUs of the 141 GB class in the operator's own facility, drafting for coordinators who confirm; a Qwen3.8 27B dense model at the site tier parsing threads and notes; and a Qwen3-Embedding 0.6B model for retrieval over threads, notes and resolutions. Improvement runs on recursive memory loops over confirmed outcomes, never on training or fine tuning, so nothing about a shipper's freight leaves the operator's control and no weights change on the fly.

Part I states the industry problem and the join failure across the seven systems. Part II sets the constraints, the layered stack, the sixteen-object model, the adapter tier and its guarantees, the inference placement with its memory arithmetic, and the three models with the licenses that let the operator own them. Part III lays out the two phases with their eighteen items and gates, what the rollout measures, the failure modes and the open questions, then who owns what the system builds. Part IV closes with how Praxis contextualized and reasoned the design, so any choice in this paper can be traced back to the recorded reading that justified it.

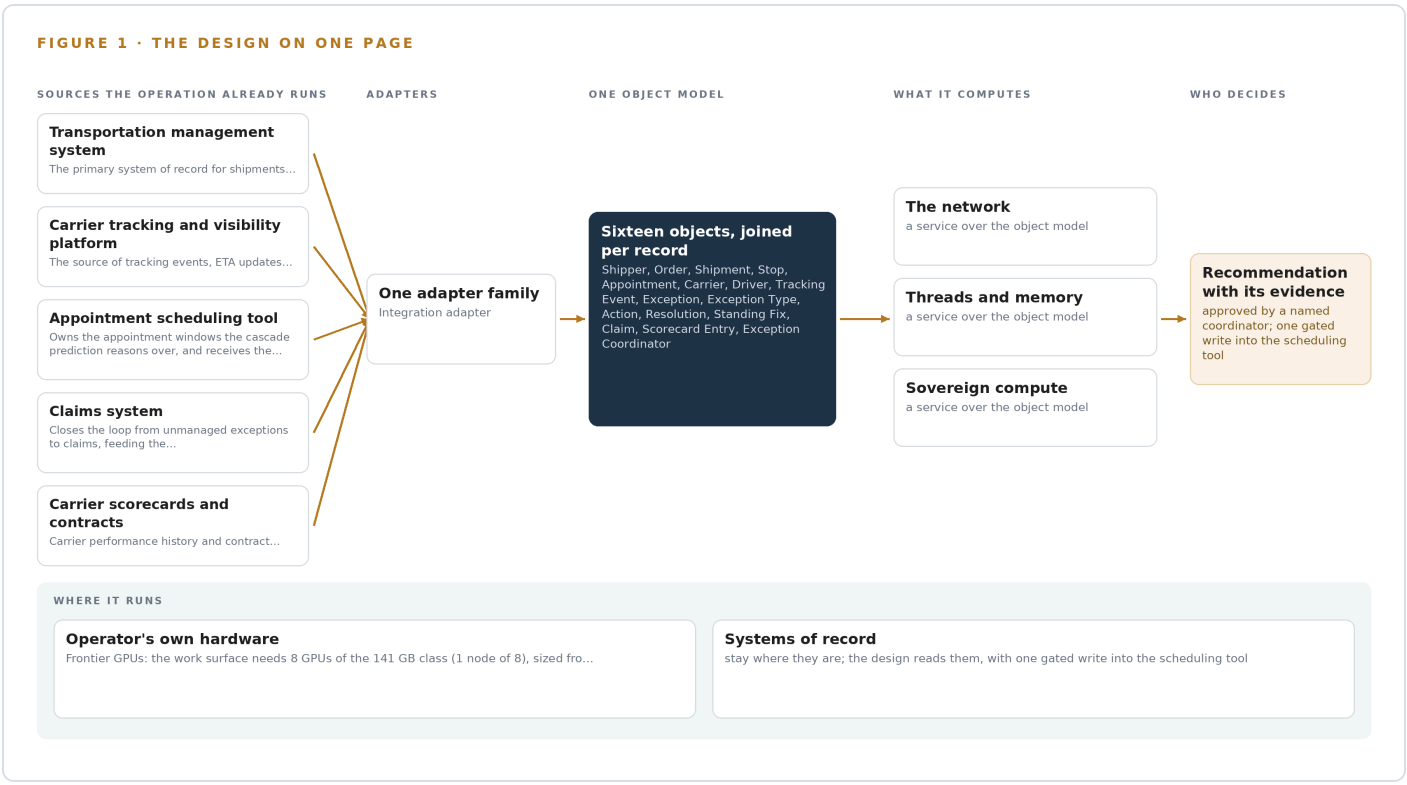


Figure 1. Lane Memory on one page: the sources the operation already runs, one object model, what it computes, and the person who decides.

Contents

Each chapter is tagged for the reader it serves most directly: Executive, Team Lead, FDE, Reference.

	Abstract · The Exception Desk Should Get Better Every Week	Executive
PART I · THE PROBLEM		
1	Exception Volume Grows While Resolution Stands Still	Executive
2	Every Desk System Sees One Slice of the Exception	Executive · Team Lead
PART II · THE DESIGN		
3	Sovereignty and the Human Loop Shape Everything Downstream	Team Lead
4	One Stack Runs From Records to the Command Center	Team Lead · FDE
5	Sixteen Objects Turn Freight Data Into One Argument	FDE
6	Seven Systems Enter Through One Adapter Tier	FDE
7	All Reasoning Stays Inside the Operator's Own Facility	FDE
8	Open Licenses Decide What the Operator Owns	FDE · Executive
PART III · THE ROLLOUT		
9	Shadow Mode Comes Before Any Draft Is Trusted	Team Lead · Executive
10	The Compounding Intelligence Stays with the Desk	Executive
PART IV · HOW IT WAS DESIGNED		
	Conclusion · Memory, Not Training, Closes the Loop	Executive
11	Praxis Recorded the Reasoning Behind Every Choice	Team Lead · FDE
	Sources	Reference

PART I · CHAPTER 1

Exception Volume Grows While Resolution Stands Still

Tens of thousands of shipments a month throw exceptions that are still worked the way they were ten years ago, and the knowledge that would resolve them faster walks out the door with every departing coordinator.

The abstract argued that a self-improving AI system for the exception desk can run entirely on the operator's own hardware, reading systems it already owns. This chapter establishes why that argument is needed at all: the volume of exceptions has grown while the practice of resolving them has not moved in a decade, and the knowledge that carries the desk leaves with every coordinator who departs.

1.1 The Question the Operation Needs Answered

When an exception lands, the desk needs one answered question within minutes, not hours: given this shipment, this carrier, this consignee and this appointment, what actually works here, and who has to be told what right now. Answering it requires data the operation already holds but holds apart: the shipment and its stop sequence from the transportation management system, the live tracking events and estimated arrival from the carrier visibility platform, the appointment window and its change history from the scheduling tool, the carrier's performance history and contract terms from the scorecards, the shipper's data terms from the order feeds, the claims record of what similar exceptions became, and the email and chat threads where past exceptions were actually worked. The regulatory regimes frame the answer rather than supplying it: federal hours of service rules bind which recovery options a driver can legally execute, privacy law governs how driver and shipper data may be used, and each shipper's contractual data terms set what may be shown to whom.

1.2 The Documented Cost of the Problem

The cost shows up first as turnover and its consequences. Exception work is high-volume, interrupt-driven and closely supervised by performance metrics, and the pattern is documented beyond this operator: a peer-reviewed cross-sectional study in logistics found algorithmic management of exactly this kind associated with psychological distress, musculoskeletal pain and occupational accidents (Springer 2025). Turnover on the desk is high, and each departure removes knowledge that was never written down: which carriers answer at night, which consignees accept a late slot, which exceptions become claims. The commercial cost follows directly, because shippers now measure the operator on time to resolve and on whether they were told before they asked; every repeat exception worked from scratch taxes that measure, and every unresolved cascade that becomes a claim converts a service failure into a financial one recorded in the claims system.

1.3 The Operation as a Scenario

The operator is a third party logistics provider in the United States managing truckload and less than truckload freight on behalf of shippers, moved by hundreds of contracted carriers. It plans, tenders and tracks tens of thousands of shipments a month. The people in the loop are the exception coordinators on shift who own each exception end to end, the shift leads who approve changes beyond a coordinator's authority, the carrier managers who own carrier relationships, the account managers who face the shippers, and the claims handlers who close what resolution failed to prevent. The physical environments are unglamorous for a physical AI design: offices, a server room in the operator's own facility, and a national road network seen only through the tracking feeds of the carriers' devices; no site is instrumented

by this design. The regulatory ground combines the trucking safety regime that governs driver hours, privacy law confirmed with counsel, and the contractual data terms each shipper imposes. The count of named systems is seven: the transportation management system, the carrier tracking and visibility platform, the appointment scheduling tool, the claims system, the carrier scorecards and contracts, the customer order feeds, and the email and chat threads.

PART I · CHAPTER 2

Every Desk System Sees One Slice of the Exception

Seven systems each hold a true but partial view of the same missed pickup or late arrival, and none of them can assemble the argument a coordinator needs in the first minutes after an exception lands.

Chapter 1 set out the question. This chapter explains why no system the operation already runs can answer it.

2.1 What Each System Sees

Transportation management system: the primary system of record for shipments, stops, tenders and carrier master data; the ontology's shipment and stop objects are projections over it, kept current by change capture.

Carrier tracking and visibility platform: the source of tracking events, ETA updates and device status, which is how the trucks are seen and the earliest signal for a dark device or a late arrival.

Appointment scheduling tool: owns the appointment windows the cascade prediction reasons over, and receives the approved appointment change requests, the one operational write path, gated on coordinator approval.

Claims system: closes the loop from unmanaged exceptions to claims, feeding the claims per thousand shipments baseline and the review agent's view of which exception types become claims.

Carrier scorecards and contracts: carrier performance history and contract terms inform which recovery option is likely to work and which carriers answer at night.

Customer order feeds: the shipper-side origin of demand: orders link shipments to shippers and their data terms, which set the purpose-based access rules.



Figure 2. Seven systems, each seeing one part of the answer. The question needs all of them in one place at once.

2.2 What None of Them See Together

Each holds one slice, and the question needs all of them in one place at once, which today is done by hand at the cadence of whoever has the time.

PART II · CHAPTER 3

Sovereignty and the Human Loop Shape Everything Downstream

Four commitments, open weights on the operator's own hardware, a named person deciding every action, improvement without training on the fly, and no replacement of the systems of record, discipline every choice made after them.

Chapter 2 showed that the join failure is structural: seven systems each hold a slice and none holds the argument. This chapter fixes the four commitments that any answer must satisfy, because they discipline every choice made after them, from where the models run to how improvement compounds.

3.1 Open Weights on the Operator's Own Hardware

The operator's requirement is explicit: open weight models on its own hardware in its own facility, with nothing about its shippers' freight leaving its control. In this industry the constraint is harder than it looks, because freight data is not the operator's alone: every shipment carries a shipper's commodities, values and consignees, bound by contractual data terms, so a cloud region or a colocation cage would move hundreds of shippers' data across a boundary none of them agreed to. The commitment also rules out managed model services entirely, which pushes the design toward a self-improving AI system in which the weights, the memory and the decision record all sit inside one estate perimeter, served on one node of the frontier inference cluster class, GPU nodes for 300B to 1T parameter open weight models, which the operator's own server room takes as standard racks.

3.2 A Named Person Decides Every Action

The second commitment is that agents draft and a named coordinator decides. Exception handling fails in this industry when automation acts unilaterally: an unapproved carrier swap strands a load, an automated shipper notice breaches data terms, and the regulatory and safety consequences of automated action in trucking are documented by the federal regulator's own investigations (Tanktransport 2022). The design therefore carries one operational write path, appointment change requests into the scheduling tool, gated on coordinator approval, and routes every carrier message and shipper notice through the same approval gate before anything leaves the estate.

3.3 Improvement Without Training on the Fly

The operator asks for a system that gets better every week without training a model on the fly, and the distinction matters commercially as well as technically: fine tuning on coordinator confirmations would produce opaque weights that are hard to audit when a shipper disputes an action, and recursive self-improvement carries governance risks now documented for enterprises (Cloudsecurityalliance 2026) and noted as spreading through the corporate ecosystem (Stanford 2026). The design compounds improvement in memory instead of weights: every confirmed resolution becomes a retrievable record the next exception reads, auditable, reversible and owned outright, with no retraining run and no distillation step.

3.4 The Systems of Record Stay Untouched

The final commitment is projection, never migration. The transportation management system, tracking platform, scheduling tool, claims system, scorecards and order feeds remain the authoritative systems they are, read through their own surfaces with change capture, and the ontology is built over them. This is hard in logistics because identifiers disagree across systems: the same carrier and the same consignee carry different identifiers in each, so entity resolution is a first-class deliverable rather than a cleanup task performed once.

3.5 Scoping Decisions

Three scoping decisions follow from the four commitments, each buying something specific at a stated cost, set out in Table 1.

Table 1 · Scoping Decisions

DECISION	WHAT IT BUYS	WHAT IT COSTS
Pilot one regional lane group for one quarter	Bounds entity resolution to a small carrier and consignee count and measures improvement against a recorded baseline before the network scales	The rest of the network keeps working the manual way until the gate is passed
One gated write path into the scheduling tool	Every other action stays with the coordinator, so the blast radius of any agent error is a declined request, not a moved load	Coordinators approve each appointment change rather than reviewing what already happened
Memory-based improvement over confirmed outcomes	Improvement compounds weekly, auditably and reversibly, with no training runs and no weight changes	The model's raw capability stays fixed, so gains come only from what coordinators actually confirm

3.6 What the Design Chose Against

Each major choice was made against a live alternative, and Table 2 records the rejects with their reasons, ending with the scope boundary that keeps later extensions honest.

Table 2 · What the Design Chose Against

WHERE	WHAT WAS PICKED	INSTEAD OF, AND WHY
Compute	One node of the 141 GB HBM class serving GLM 5.3 at FP8	Two nodes of the 80 GB class or BF16 serving: the FP8 weights are about 756 GB and one node holds them with a KV cache ceiling, while BF16 needs about 1.5 TB and a second joined node
Improvement	Recursive memory loops over confirmed outcomes	Fine tuning or distilling the weights on confirmations: the requirement bars training on the fly, and memory is auditable where weights are not
Action	Drafts awaiting a named coordinator's approval	Autonomous re-tendering and notice-sending: an unapproved carrier swap or shipper notice carries regulatory and contractual exposure no volume gain justifies
Thread parsing	A 27B-class dense model at the site tier parsing high-volume threads	The frontier model on every message: parsing volume would consume the concurrency the command center needs for coordinators
Scope boundary	The pilot lane group, one gated write path and	Quiet extensions: automated re-tendering, weight training, driver behavior monitoring, systems of record replacement and historic

WHERE	WHAT WAS PICKED	INSTEAD OF, AND WHY
	named inclusions	data migration are out of scope, and any later inclusion is its own gated decision

PART II · CHAPTER 4

One Stack Runs From Records to the Command Center

Systems of record below, one governed object model in the middle, agents and two surfaces above, all on one event backbone and all inside the operator's estate.

Chapter 3 turned the operator's constraints into scoping decisions: no training on the fly, no shipper or carrier data leaving the estate, a named person on every action, and the systems of record left exactly where they are. This chapter turns those decisions into a stack, so that each layer's responsibility follows from a constraint rather than from a preference.

4.1 The Pattern and Why It Fits

The pattern is the one shape that satisfies all four constraints at once: systems of record below, one governed object model in the middle, applications and agents above. The seven systems of record stay untouched at the bottom, read through their own surfaces, because the ontology is a projection over them and never a migration. In the middle, one object model of sixteen objects joins shipment to carrier to appointment to exception, so every agent reads one governed network instead of seven disconnected stores. Above it, the triage, recovery and review agents draft and coordinators confirm; that split is architecture rather than courtesy, because the requirement fixes a named approver on every action, and the published observation that recursive self-improvement is already spreading from frontier laboratories into the wider corporate ecosystem argues for the confirmation gate and the recorded decision trail being structural, not appended (Stanford 2026). The same reasoning keeps improvement out of the weights: the memory service compounds only from confirmed outcomes, which is what the requirement means by getting better every week without training a model on the fly. Figure 3 shows the layered stack with its counts: seven sources, one adapter family, sixteen objects, three models across a central tier, a site tier and an embedding tier, eight services and two surfaces, all inside the operator's facility.

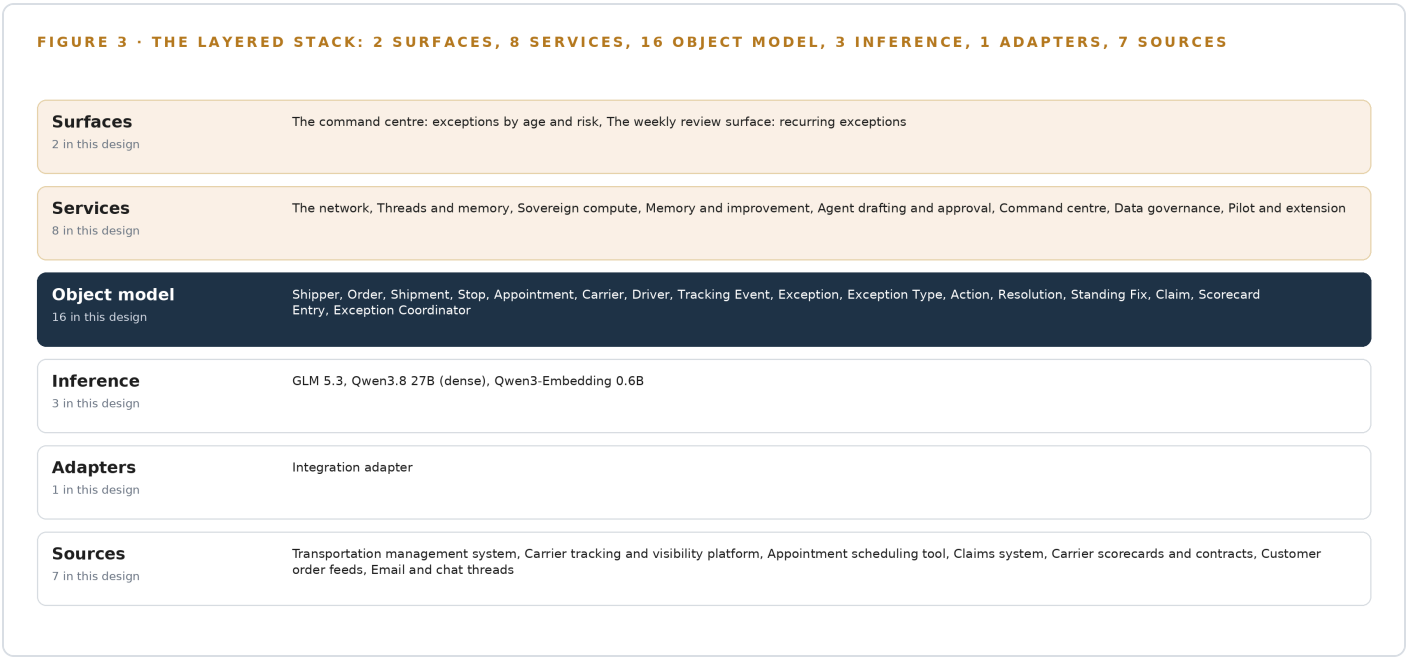


Figure 3. The layered stack: 7 sources, 1 adapter family, 16 objects, 8 services and 2 surfaces.

4.2 The Stack Stage by Stage

Table 3 assigns responsibility stage by stage, from the records to the surfaces, and names the component that carries each responsibility. The sources stage is the seven named systems, all read. The sensing stage installs nothing: the trucks are seen through the tracking feeds the operator already owns, and the exception desk is knowledge work, so no camera, positioning hardware or edge box exists in this design. The adapter stage is a single integration adapter family, with Debezium change capture on the transactional systems and governed pulls on the feeds and threads. The object model stage is Hyper Ontology v0.9 holding the sixteen objects and their typed links, with InfluxDB 3 Core underneath for tracking telemetry. The inference stage is vLLM serving GLM 5.3 at FP8 on one node of the 141 GB class, a dense site-tier model for high-volume thread parsing, and an embedding model for retrieval. The services stage is the eight services, from the network through pilot and extension. The surfaces stage is the two places a named person looks: the command center's exception view and the weekly review surface. Alongside Table 3, Table 7, printed by the platform from the plan, records each layer's component, the choice made for it and the reason, including the layers where nothing was needed; Table 3 says who is responsible for what, and Table 7 records the grounds on which each choice stands.

Table 3 · The Stack, Stage by Stage

STAGE	WHAT IT IS RESPONSIBLE FOR	HOW
Sources	Holding the records of freight, tracking, appointments, claims, carrier performance and the threads where exceptions are worked	Seven systems of record: the transportation management system, the carrier tracking and visibility platform, the appointment scheduling tool, the claims system, the carrier scorecards and contracts, the customer order feeds, and the email and chat threads, all read through their own surfaces

STAGE	WHAT IT IS RESPONSIBLE FOR	HOW
Sensing	Seeing the trucks without instrumenting them	No cameras, positioning hardware or edge boxes are installed; trucks are seen through the tracking platform's existing feeds, and the exception desk is knowledge work
Adapters	Carrying ordering, provenance and scope from every source into the estate	One integration adapter family: Debezium change capture on the transactional systems, governed pulls on feeds and threads, and no source written to directly
Object model	Turning records into one governed network of freight	Hyper Ontology v0.9 holding the sixteen objects and their typed links, with InfluxDB 3 Core for tracking telemetry
Inference	Serving every model inside the operator's facility	vLLM serving GLM 5.3 at FP8 on one node of the 141 GB class, the dense site-tier model for thread and note parsing, and the embedding model for retrieval
Services	Doing the work between the model and the desk	Eight services: the network, threads and memory, sovereign compute, memory and improvement, agent drafting and approval, command center, data governance, and pilot and extension
Surfaces	Putting the whole argument in front of the named decision-maker	Two surfaces: the command center's exception view, by age and risk, and the weekly review surface for recurring exceptions

Table 7 · Architecture, Layer by Layer

LAYER	COMPONENT	CHOICE	WHY
Event streaming and messaging	by class, no product named	Product	Apache Kafka (KRaft mode)
The ontology or graph store	by class, no product named	Product	Hyper Ontology v0.9 (Semantics and Kinetics)
Time series and telemetry storage	by class, no product named	Product	InfluxDB 3 Core
Video ingest, recording and management	by class, no product named	Not needed	No cameras exist in this design: the trucks are seen through the operator's existing tracking feeds, which this design reads but does not instrument, and the exception desk is knowledge work.
Edge model serving	by class, no product named	Not needed	All inference runs in the operator's own facility (intake answer: 'Our own hardware in our own facility'); there are no edge boxes, gateways or cell controllers to serve on.

LAYER	COMPONENT	CHOICE	WHY
On site language model serving	by class, no product named	Product	VLLM
Object tracking and trajectory forecasting	by class, no product named	Not needed	Cascade prediction is the triage agent's reasoning over the ontology's stop sequences and appointment windows, 'the shipments it will cascade to', not video trajectory tracking; there is no video stream to track.
Edge orchestration of containers	by class, no product named	Product	Kubernetes
An air gapped artifact and model registry	by class, no product named	Product	MLflow Model Registry
Labelling and retraining	by class, no product named	Not needed	Under system of context nothing is trained, fine tuned or distilled into weights: improvement runs on its memory loops, which is exactly the requirement's 'without training a model on the fly'.
Time synchronisation across sensors and systems	by class, no product named	Class	The shelf holds no software time sync product for this layer; ordering across the TMS, tracking and thread streams is carried by Kafka offsets and per stream sequence numbers, with each source's clock offset recorded as a field rather than silently corrected, so a causal claim about which event came first stays honest.
Observability of the running system	by class, no product named	Product	Prometheus

4.3 A Reusable Pattern

The reusable shape is five rules rather than a component list, and a team carrying a similar brief can lift it wholesale. Put one event backbone under everything, so ordering and replay belong to the platform and not to each integration. Pass every source through one adapter family that carries provenance and scope, so no record enters the model ungoverned. Let agents read only the governed object model, never the raw stores, so the network's joins and vocabularies form the single argument every draft rests on. Make agents draft and named people approve, with one gated write path that nothing bypasses. Let improvement land in memory built from confirmations rather than in weights, so the system gets better every week without a training pipeline. What a team swaps is the domain: its sources, its sixteen objects, its models sized to its thread volume and its write target. What it keeps is the backbone, the adapter tier, the approval gate and the memory loop.

Sixteen Objects Turn Freight Data Into One Argument

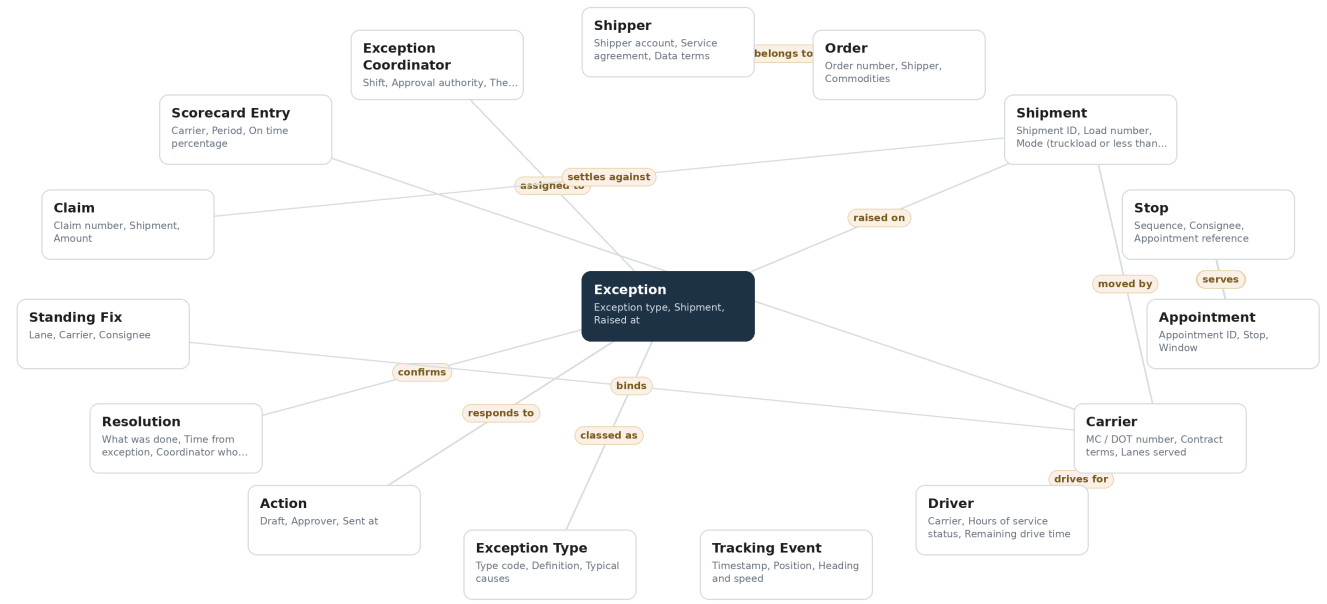
A governed ontology of sixteen objects joined by typed links lets one query cross from a dark tracking device to the consignee who accepts a late slot, a reach no document store can give.

Chapter 4 described the stack as a set of responsibilities. This chapter opens the middle of it, where the sixteen objects live and where every agent's reasoning begins.

5.1 Every Object and the Links Between Them

The model holds sixteen objects: shipper, order, shipment, stop, appointment, carrier, driver, tracking event, exception, exception type, action, resolution, standing fix, claim, scorecard entry and exception coordinator. Each is anchored in the system of record that owns it, and the anchors matter: a shipment is a projection over the transportation management system, an appointment over the scheduling tool, a claim over the claims system, while exception type, standing fix and exception coordinator are anchored in the design itself as governed vocabularies the operator owns outright. The typed links are what make the model an argument rather than a filing cabinet: an exception is raised on a shipment, the shipment is moved by a carrier, the carrier is rated by scorecard entries, a resolution confirms the exception, and a standing fix binds a recurring exception type to a carrier on a lane. Figure 4 shows every object and every typed link, with the exception object in navy as the focal point of the desk's day. What the links buy is reach: one query can start at an exception on a late shipment, walk to its carrier's scorecard, see whether that carrier is reachable at night, cross to the stop's appointment and its change history, and end at the standing fixes and confirmed resolutions on that same lane, carrier and consignee. A document store can retrieve threads that mention a load number, but it cannot walk that path unless someone wrote the join in advance, and the recurring exceptions this design exists to kill are exactly the ones no one wrote the join for. Table 8, printed by the platform from the plan, lists each object, the system it is anchored in and its typed links, and is the register the governance service maintains.

FIGURE 4 · THE OBJECT MODEL: 16 OBJECTS AND 13 TYPED LINKS



The navy card is the object the most links reach. A link label is the property the plan carries, or the verb the design names.

Figure 4. The sixteen objects of the model and the typed links that let a query reach across them.

Table 8 · The Object Model

OBJECT	ANCHORED IN	TYPED LINKS
Shipper	Customer order feeds	no typed link recorded
Order	Customer order feeds	belongs to: Shipper
Shipment	Transportation management system	moved by: Carrier
Stop	Transportation management system	carries: Appointment
Appointment	Appointment scheduling tool	serves: Stop
Carrier	Transportation management system	rated by: Scorecard Entry
Driver	Carrier tracking and visibility platform	drives for: Carrier
Tracking Event	Carrier tracking and visibility platform	no typed link recorded
Exception	Email and chat threads	raised on: Shipment; classed as: Exception Type; assigned to: Exception Coordinator

OBJECT	ANCHORED IN	TYPED LINKS
Exception Type	The design's own governed vocabulary	no typed link recorded
Action	Email and chat threads	responds to: Exception
Resolution	Email and chat threads	confirms: Exception
Standing Fix	The design's own governed vocabulary	binds: Carrier
Claim	Claims system	settles against: Shipment
Scorecard Entry	Carrier scorecards and contracts	no typed link recorded
Exception Coordinator	The design's own governed vocabulary	no typed link recorded

5.2 The Human Loop and the Hosting Posture

The human loop lives in three status vocabularies. An exception moves from New through Triaged, Recovery drafted and Approved to Resolved or Became claim; an action moves from Drafted through Awaiting approval and Approved to Sent, Acknowledged or Declined; a resolution moves from Proposed through Confirmed to Promoted to standing fix. No action is sent without a named approver, and the approver is recorded on the action object, which is the accountability the design rests on: when automation is implicated in harm, regulators have charged both the employer and the system supplier, so the named decision-maker is not decoration but the defense (Ohsrep 2024). The same loop answers the human cost of algorithmic management, which a cross-sectional study in logistics has associated with psychological distress and occupational accidents: the coordinator sees a draft labeled preliminary, decides, and is the decision-maker rather than the managed (Springer 2025). The hosting posture follows the constraint that shipper and carrier data never leave the operator's hardware. Everything runs on the operator's own equipment in its own facility, identity is federated through Keycloak, no inference call leaves the estate and no external model is called, and the one write path in the entire design is the approved appointment change request into the scheduling tool's request surface. Retrieval is permission-aware, returning only what the asking coordinator could already open, and every record the agents read carries provenance and the scope of the shipper's data terms.

5.3 One Object in Its Recorded Form

The shipper record below shows the form every object takes and what an agent actually reads: an identifier, a label, a kind, its properties, its status vocabulary and its links. The exception object, whose properties carry type, risk band and cascade stops, whose status vocabulary structures the desk's day and whose links reach the shipment, the coordinator and the type, is recorded the same way.

{ "id": "shipper", "label": "Shipper", "kind": "actor", "anchored_in": "Customer order feeds", "properties": ["Shipper account", "Service agreement", "Data terms", "Notification preference", "Required sign-off level"], "status_vocabulary": [], "links": [] }

PART II · CHAPTER 6

Seven Systems Enter Through One Adapter Tier

Every source, whatever its provenance, passes through the same adapter family that carries ordering, provenance and scope, leaving one gated write path into the scheduling tool and no direct writes anywhere else.

Chapter 5 described what the object model holds. This chapter describes how data gets into it, and why nothing gets in except through the adapter tier.

6.1 Seven Sources, One Provenance Class

Every source in this design is of provenance class operator, held by the operator and read, never written to. The transportation management system is the primary record for shipments, stops, tenders and carrier master data, and the shipment and stop objects are projections over it, kept current by change capture. The carrier tracking and visibility platform supplies the tracking events, ETA updates and device status, and carries the earliest signal for a dark device or a late arrival. The appointment scheduling tool owns the windows the cascade prediction reasons over, and is also the target of the one gated write path, the approved appointment change request. The claims system closes the loop from unmanaged exceptions to claims. The carrier scorecards and contracts carry performance history and terms, including which carriers answer at night. The customer order feeds are the origin of demand, linking shipments to shippers and to the data terms that set purpose-based access. The email and chat threads are where most exceptions are actually worked, across tens of thousands of shipments a month and hundreds of carriers, and the archived years of those threads are what bootstraps the memory before the first live exception arrives. Figure 5 sets the seven systems beside their adapter paths into the object model.

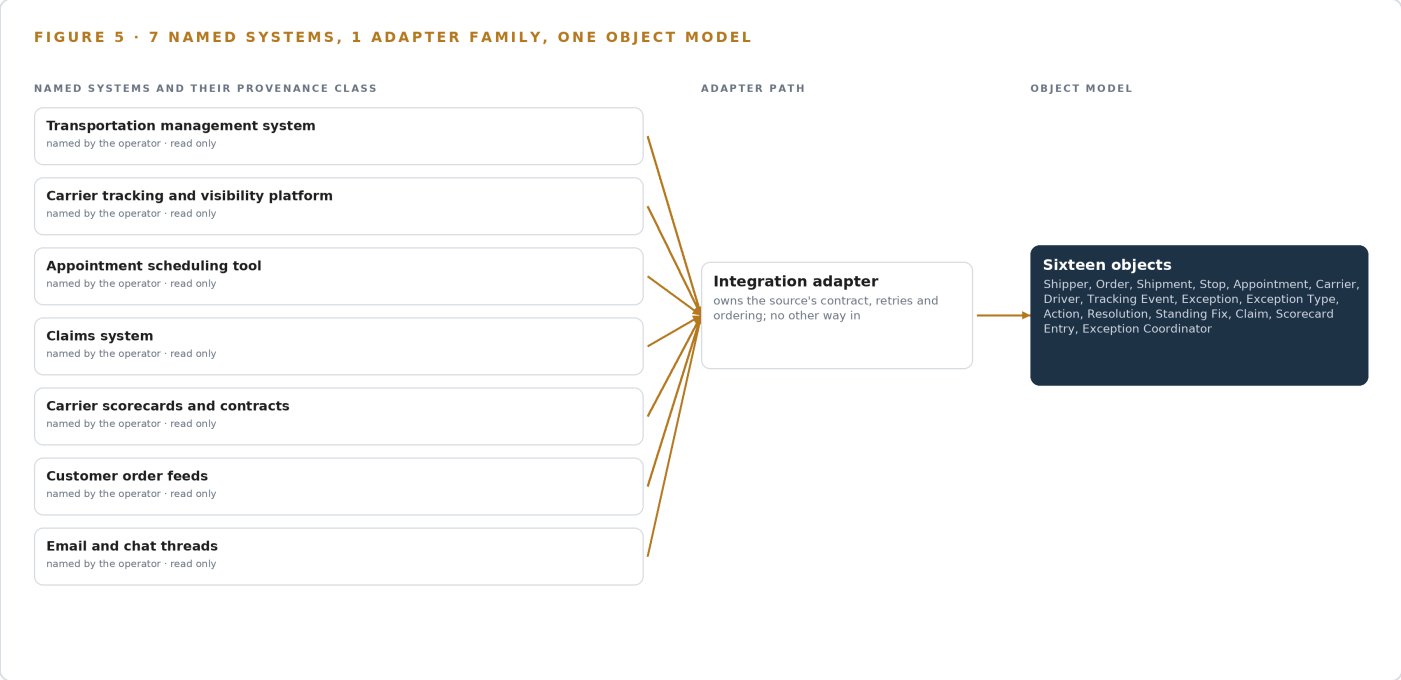


Figure 5. The 7 named systems, the adapter path each one takes, and the object model they all map into.

6.2 What the Adapter Tier Guarantees

The adapter tier is a single integration adapter family, and its guarantee is that no record reaches the object model except through it. Each adapter carries four things on every record. Provenance: which system the record came from, when it was extracted, and that source's clock offset, recorded as a field rather than silently corrected, so a causal claim about which event came first stays honest. Ordering: a per-stream sequence number alongside the backbone offset, so the projection can detect a gap or a replay. Scope: the shipper whose data terms govern the record, which is what the permission-aware retrieval in Chapter 5 enforces. Conformity: the record must map to the object model's vocabulary before it enters, so entity reconciliation, the same carrier and consignee carrying different identifiers in each system, happens at the adapter under governance rather than in each agent's prompt. Debezium handles change capture on the transactional systems; governed pulls handle the feeds and threads. The tier also guarantees the negative: no agent, service or surface writes to any source directly, and the single write in the design is the approved change request into the scheduling tool.

6.3 The Event Backbone

The backbone is Apache Kafka in KRaft mode, and it carries ordering, delivery, buffering and replication. Ordering is carried by Kafka offsets and the per-stream sequence numbers, with each source's clock offset as a recorded field, so the desk can argue about sequence without pretending the clocks agree. Delivery is at least once, and the projection into the object model is idempotent, so a replayed stream updates the same objects rather than duplicating them. Buffering is the retained log itself: a slow consumer, a parsing backlog or a serving restart does not lose events, because the adapter replays from the backbone without re-reading the source. Replication runs inside the facility, so a single broker failure does not stop the desk. The write path deserves its own sentence: an approved appointment change

request crosses from the action object to the scheduling tool's request surface only after coordinator approval, and nothing else in the design writes anywhere.

PART II · CHAPTER 7

All Reasoning Stays Inside the Operator's Own Facility

Every inference call, from frontier drafting to thread parsing to embedding, lands on hardware the operator owns, sized from the weights up, so nothing about a shipper's freight crosses the estate boundary.

Chapter 8's licenses decide who holds the weights; this chapter decides where those weights physically sit and what each tier of the system asks of them. The placement follows from one sentence in the operator's requirement, our own hardware in our own facility, and every number below is arithmetic on the filed parameter counts, not an estimate borrowed from outside the estate.

7.1 Three Tiers, Each Sized From the Weights Up

The design runs three inference tiers, and each one is sized by the same method: take the parameter count, multiply by the bytes per parameter at the chosen precision, and check the result against usable accelerator memory before any workload is placed. Figure 6 shows the tiers side by side with what runs on each and what, if anything, crosses the estate boundary.

The frontier tier serves GLM 5.3 for the agentic work surface and ontology maintenance, and it is the tier the arithmetic is tightest on. GLM 5.3 carries 753 billion parameters, filed as the register's count, and the design serves them at FP8, where one byte per parameter gives 753 GB of weights. Applying the planning factor of 1.2 for KV cache and activations, the GPUs must hold 904 GB. One node of the 141 GB HBM class, eight GPUs at 141 GB each, provides 1,128 GB of memory; at the 0.92 utilization the sizing rule allows, that is about 1,038 GB usable, which holds the weights with about 280 GB of KV cache ceiling left for concurrent coordinator sessions in the command center. Serving the same model at BF16 would demand about 1,506 GB and would force two nodes joined over an interconnect, which is why FP8 on one node was chosen and Chapter 3 records the alternative as rejected.

The site tier serves the dense Qwen3.8 27B model for high-volume thread and note parsing, because years of archived email and chat, plus the live threads that trigger and narrate exceptions, are a throughput workload rather than a reasoning one. At FP8 its weights occupy about 31 GB, which fits a single card of the site inference server class with about 13 GB of KV cache ceiling, in an air-cooled PCIe server the facility's server room already takes without modification.

The embedding tier serves Qwen3-Embedding 0.6B over threads, notes and resolutions for retrieval. Its weights occupy about 1.2 GB at BF16, about 0.6 GB at 8 bit, so it sits beside the site-tier model on the

same server, and its activation memory is what grows with the 32K context window rather than with anything the desk controls. The frontier node carries the room requirement the operator's facilities team must confirm: about 15 kW of rack power per node with margin, a 10 to 35 C inlet range, and 10 RU per node.

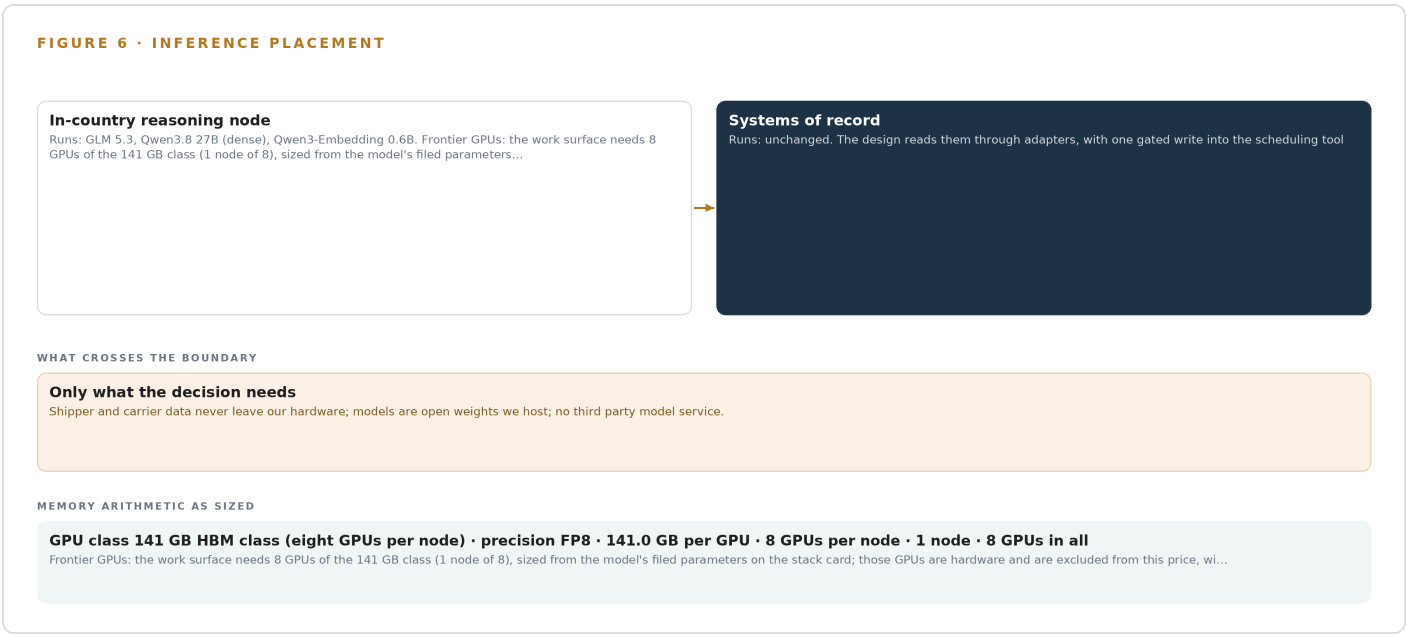


Figure 6. Where each tier runs, what runs there, and the narrow set of outputs that cross the boundary.

7.2 The Latency Budget

The budget is stated in the desk's own beats rather than in milliseconds, because the desk's measures are time to resolve and whether the shipper was told before asking. First priority is triage freshness: a live thread event must raise the exception, match it against memory, and leave a draft waiting in the queue before the coordinator who opens the queue reads it, and this is what the site-tier parse and embedding pass exist to keep ahead of. Second is the conversational answer: the command center's interface answers with citations, and an answer that loses the session has no value, so the frontier tier's concurrency is bounded by the about 280 GB KV cache ceiling, which makes the latency budget a memory budget as much as a time one. Third, and free to trail, is batch work: parsing archived threads and refreshing embeddings can run behind the live queue. In every tier the coordinator is the decision-maker, which is the deliberate answer to the finding that algorithmic management without a human decision point is associated with psychological distress and occupational accidents in logistics work (Springer 2025).

7.3 The Boundary and the Failure Paths

What crosses the estate boundary is only what a coordinator approves: carrier messages and shipper notices, sent through email on the action object's approved path. All three models, the object model, the memory and the decision record stay inside, and nothing in the design reaches the truck itself, a line drawn with the regulator's own investigation of an autonomous trucking mishap in view (Tanktransport

2022). If the facility node fails, the manual way of working stays warm and rehearsed: a coordinator can always work an exception by thread, phone and note, and serving restarts from pinned images in the on-site registry, with a rollback that has been exercised once under observation. If power fails, the same fallback holds, and the room requirements above are the operator's side of that insurance. If the update path is the failure, the registry's pinned images mean the desk never runs on an unobserved weight change; and if the wide-area link fails, the desk loses only the outbound notices, because no inference call, no retrieval and no drafting ever left the building.

PART II · CHAPTER 8

Open Licenses Decide What the Operator Owns

Three open-weight models, each held under a license that permits the operator to keep, serve and improve around its own copies, sized to one frontier node and one site-tier server with the arithmetic to prove it.

Chapter 7 fixed where every inference call lands; this chapter fixes what the operator is allowed to keep once it lands there. Placement is arithmetic, but ownership is a license question, and it is the licenses that decide whether the compounding value of the exception desk stays with the operator or leaks back out through a servicer's terms.

FIGURE 7 · THE MODEL STACK: 3 MODELS, THEIR PLACEMENT AND THEIR WORK

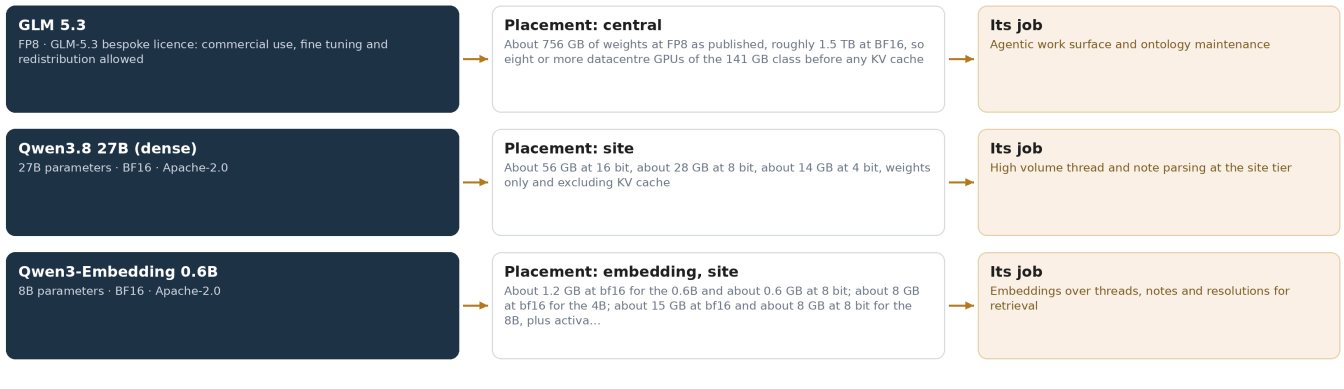


Figure 7. The three models, their placement, and the work each one does.

8.1 Three Open-Weight Models, Each Held Under Terms That Permit Ownership

Figure 7 shows the model stack: one frontier model, one site-tier model and one embedding model, all open weights, all served inside the facility. The frontier model is GLM 5.3, a 753 billion parameter model whose published footprint is about 756 GB of weights at FP8 and roughly 1.5 TB at BF16. Its role is the agentic work surface and ontology maintenance: the command center's conversational interface with

citations, the triage and recovery drafting, and the governed model of the freight network. It runs on the frontier node described in Chapter 7, and it was chosen because its bespoke license permits commercial use, fine tuning and redistribution, which means the operator holds its own copy outright, on hardware it owns, with no per-token obligation to anyone outside the estate.

The site-tier model is Qwen3.8 27B, dense, served at FP8 on the 48 GB PCIe card class. Its role is high-volume thread and note parsing, the layer that turns years of emails and chat threads into structured resolutions without spending frontier capacity on it. Its footprint runs about 56 GB at 16 bit, about 28 GB at 8 bit and about 14 GB at 4 bit, weights only and excluding KV cache, and at FP8 the about 31 GB of weights leave about 13 GB of KV ceiling on one card. It carries the Apache-2.0 license, whose terms let the operator copy, serve and modify the weights commercially.

The embedding model is Qwen3-Embedding 0.6B, also Apache-2.0, whose role is embeddings over threads, notes and resolutions so that retrieval can find the past exception that matches this week's. Its weights occupy about 1.2 GB at BF16 and about 0.6 GB at 8 bit, with activation memory that grows with the 32K window; it shares the site-tier server.

8.2 Improvement Runs on Memory, Not on the Weights

The requirement asks for a system that gets better every week from what coordinators confirm, without training a model on the fly, and the license terms are what make that promise durable. Recursive self-improvement is already an active priority at frontier AI companies and is spreading into the wider corporate ecosystem (Stanford 2026), and its security implications are an open question in the literature (Cloudsecurityalliance 2026). This design's answer is structural rather than probabilistic: no model's weights are trained, fine tuned or distilled in either zone, and improvement runs on memory loops over confirmed outcomes, each confirmation recorded against a named coordinator. The decision record the next run reads is therefore auditable in a way silent weight drift never is, and because every license named above permits the operator to hold, serve and keep its copies, the improving asset stays inside the estate boundary. That is the property the whole design defends, and the property that a self-improving AI system built this way exists to guarantee: the intelligence compounds where the data lives, under governance the operator can inspect.

8.3 The Model and Equipment Register

Table 4 gathers every choice this chapter and Chapter 7 stand on: the three models, the hardware classes and sizing rules that place them, the sensing the design does and does not do, the patterns it stands on, and the ground the whole stack runs on. Each row is a decision with its reason, so an engineer building from this paper can trace any placement back to the arithmetic or the license term that produced it.

Table 4 · Model and Equipment Register

THE CHOICE	WHAT WAS PICKED	WHY HERE
Frontier model	GLM 5.3, 753B parameters served at FP8	Agentic work surface and ontology maintenance on weights the operator owns outright
Frontier license	GLM-5.3 bespoke license, commercial use, fine tuning and redistribution allowed	Permits the operator to hold, serve and keep its copy with no external servicer
Site-tier model	Qwen3.8 27B, dense, at FP8 on the 48 GB PCIe class	High-volume thread and note parsing without spending frontier capacity
Site-tier license	Apache-2.0	Copy, serve and modify commercially; the operator keeps the weights
Embedding model	Qwen3-Embedding 0.6B, about 1.2 GB at BF16	Embeddings over threads, notes and resolutions for retrieval, sharing the site-tier server
Frontier hardware class	One node of the 141 GB HBM class, eight GPUs, 1,128 GB	Holds 904 GB of weights, cache and activations with about 280 GB of KV ceiling
Frontier sizing rule	Frontier Model GPU Footprint Sizing Rule (vLLM, SGLang): parameters times one byte times 1.2	Turns filed parameter counts into a memory requirement before any workload is placed
Node count rule	GLM 5.3 Node Count By GPU Class	One node, not two joined over an interconnect, keeps serving simple and the room small
Site-tier hardware class	The site inference server class: one 48 GB PCIe card in an air-cooled server	About 31 GB of FP8 weights with about 13 GB of KV ceiling fits one card
Sensing	None; the existing carrier tracking and visibility feeds are read	Trucks are seen through feeds the operator already owns; no camera or positioning hardware is built
Improvement pattern	Memory loops over confirmed outcomes, never weight training	The requirement's own words: improvement without training a model on the fly
Approval pattern	Every draft confirmed by a named coordinator	Confirmations logged as ground truth, with known-answer probes and weekly sampling

PART III · CHAPTER 9

Shadow Mode Comes Before Any Draft Is Trusted

A pilot quarter on one regional lane group, gated on measured improvement against a recorded baseline, precedes any extension to the other lane groups, with failure modes answered in the design rather than discovered in it.

Chapter 8 fixed the models and the licenses that let the operator hold its own weights on its own node. This chapter lays out how that system reaches the desk: two phases with their item counts and exit gates, the measures the gates read, the failure modes the design answers in advance, and the lessons and open questions the rollout carries with it.

9.1 Phases, Workstreams and Gates

The rollout carries 18 items across 2 phases, and every phase ends at a gate rather than at a date. Phase 1, the pilot quarter, holds 17 items spread across six workstreams: Agent drafting and approval, Command center, Data governance, Memory and improvement, Pilot and extension, and Sovereign compute. Its items run from baselining the pilot lane group's exception measures before any draft reaches a coordinator, through building the network from the systems of record and the archived threads, ingesting and parsing the thread archive in layers with the site-tier model, standing up the facility node and the pinned model images in the on-site registry, to exercising the serving rollback once under observation. The exit gate is measured improvement on the pilot lane group against the baseline recorded at the start: time to resolve and repeat exceptions must move against that baseline before anything scales. Phase 2, network extension, holds 1 item: extend to the other lane groups, and its gate is the extension decision itself, taken on the pilot's measured record. The requirement register's 21 requirements map onto the 18 items, and Figure 8 shows the phases, their item counts, their gates and that coverage on one page.

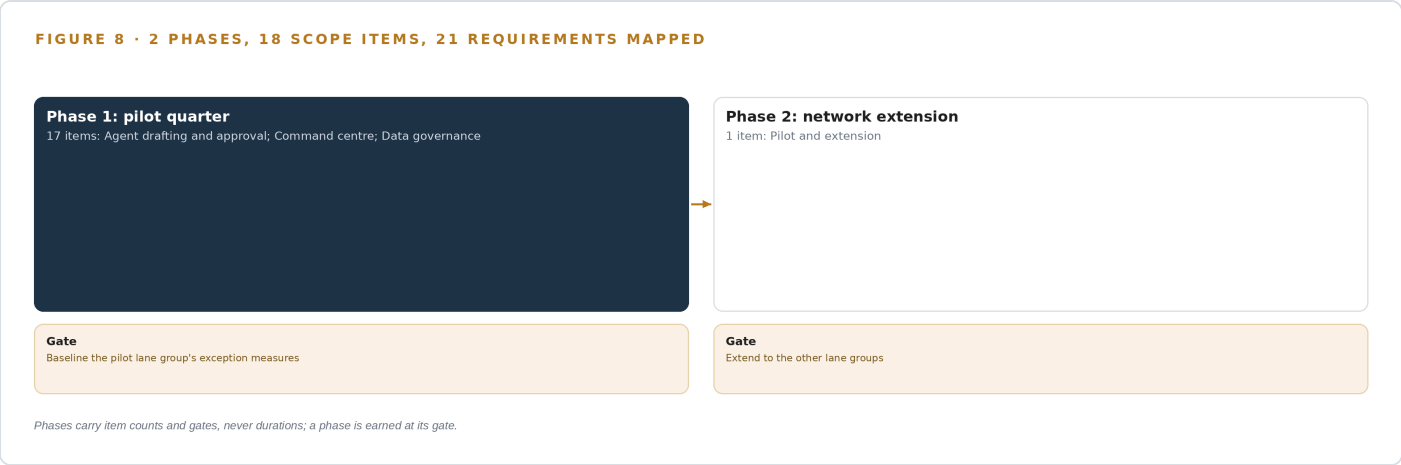


Figure 8. The two phases and their gates, and coverage of the 21 requirements across them.

9.2 What the Rollout Measures

The measures exist to make the gate honest. Time to resolve is measured from the moment an exception is raised to the moment a coordinator confirms its resolution, and it is the figure shippers already hold the

desk to. Repeat exceptions per lane, carrier and consignee measure whether last week's fix is at hand this week, which is the specific failure the operator named. Claims per thousand shipments, fed from the claims system, track which unmanaged exceptions become money. Missed appointment windows and declined change requests catch the case where an approved change is not consumed by the scheduling tool and the delay simply relocates to the dock. The confirmation rate, read with known-answer probes injected into the queue and a weekly sample of unflagged exceptions, tests whether coordinator confirmations are genuine decisions or reflexes. Every measure is read against the recorded baseline, because self-improvement that no one can audit is exactly the systemic risk the security literature warns of (Cloudsecurityalliance 2026), and the practice is already spreading through the corporate ecosystem ahead of the governance to check it (Stanford 2026).

9.3 Failure Modes Answered in the Design

Each failure below was answered in the design rather than left to be discovered in it. Table 5 states what fails and what the design does about it.

Table 5 · Failure Modes

WHAT FAILS	WHAT THE DESIGN DOES
Entity resolution across systems	The same carrier and consignee carry different identifiers in the TMS, the threads and the scorecards, so drafts match the wrong past exceptions
Approval rubber stamping	Coordinators confirm drafts without reading them, so logged ground truth only ever agrees with the agents
Thread archive quality	Years of emails and chat carry the real resolutions but also noise, stale carriers and mixed threads, and a poisoned bootstrap degrades every later draft
Model service unavailable	The facility node fails and the desk loses drafts, matching and the command center at the worst hour
Shipper data terms breached	One shipper's freight details surface in another shipper's answer, notice or memory entry
Unconsumed appointment changes	The scheduling tool does not consume an approved change request, relocating the delay to the dock

9.4 Lessons

Shadow the drafts before anyone trusts them. The desk improves only if confirmations are decisions, and the evidence on algorithmic management in logistics is that automation pressed onto workers without a genuine decision left to them is associated with psychological distress and worse outcomes (Springer 2025). The probes, the preliminary framing and the weekly sample exist to keep the coordinator the decision-maker in fact and not only in the process diagram.

Keep the manual way warm. The worst hour for the node to fail is the hour the desk needs it most, so the thread, phone and note path is rehearsed rather than merely documented, and the rollback from pinned on-site images has been exercised once under observation so that restart is a procedure and not a hope.

Reconcile the entities before the network scales. A governed vocabulary built on one lane group, where the entity count is small enough to check by hand, is cheaper than repairing wrong matches after every lane group has inherited them.

9.5 What Is Still Open

Three questions remain open. The network scale figures come from public reporting on providers of this class and must be confirmed against the operator's own shipment and carrier counts; settling them fixes the node's concurrency headroom and the size of the pilot's baseline sample. The scheduling tool's request surface must be verified to consume the approved change requests the write path sends; settling it confirms whether the missed-window and declined-request alarms are a backstop or a primary control. The scope of driver data use, already narrowed with counsel to the hours of service rules that bind recovery options, awaits final confirmation; settling it fixes whether driver objects carry contact routes in the pilot or only in the governed vocabulary. None of these questions blocks the phase 1 gate, and each is priced into an item rather than left to the extension.

PART III · CHAPTER 10

The Compounding Intelligence Stays with the Desk

The object model, the weights, the decision record and the boundary all belong to the operator, so the memory that makes the desk improve weekly is an asset of the estate that produced it.

Chapter 9 set the two phases, their 18 items and the gates that decide whether the design earns the wider network. This chapter settles the question the gates leave open: once improvement is proven, who owns the machine that produced it.

10.1 What the Estate Owns

The object model comes first. The design's 16 objects are a governed projection over the operator's seven systems of record, never a migration: shipment and stop project over the transportation management system, appointment over the scheduling tool, claim over the claims system. Because the model is defined against the operator's own identifiers and governed vocabulary rather than any vendor's schema, it survives a change of tracking platform; the anchored projections are re-pointed and the typed links hold. The weights come second. GLM 5.3 serves the work surface and ontology maintenance, a

dense Qwen model serves the site tier, and a Qwen embedding model serves retrieval, each held under a license that lets the operator keep it: the GLM bespoke license permits commercial use, fine tuning and redistribution, and the Qwen models carry Apache-2.0. Nothing in this design fine-tunes a weight, since improvement runs on the recorded memory of confirmed outcomes; the licenses matter regardless, because no vendor can withdraw the capability, meter it by the query, or repossess it in a dispute.

The decision record and the boundary complete the answer. Every draft, approval, resolution and standing fix is a record the next decision reads, and that compounding record is the asset the exception desk produces. Recursive self-improvement has spread from frontier laboratories into the wider corporate ecosystem (Stanford 2026), and its security implications are treated as an enterprise risk category in their own right (Cloudsecurityalliance 2026); this design answers both by keeping the loop, its record and its boundary inside the estate. That is the shape readers now look for in a self-improving AI system: the memory, the weights and the decision record sit with the operation that produced them, and nothing crosses the estate perimeter.

10.2 The Offer Behind the Design

This design maps onto what CodeNinja builds, piece by piece. Decision Systems is the ranking and recommendation tier, the triage drafts and recovery options a named coordinator confirms or declines. Hyper Ontology is the object model itself, the governed projection of the freight network. Hyper Pragma is the work surface, the command center where coordinators work the queue and where the operator's own people build further agents on models they can change without losing what was built. Sovereign Infrastructure is the posture the requirement demanded, the operator's own hardware in its own facility with open-weight licenses on one 141 GB class node inside the boundary. Praxis is the platform that produced and recorded the design, choice by choice.

PART IV · CONCLUSION

Memory, Not Training, Closes the Loop

In one view, the design is a governed ontology of sixteen objects projected over seven systems the operator already runs, fed by one adapter family, reasoned over by triage, recovery and review agents on three open-weight models served entirely on one frontier node and one site-tier server inside the operator's own facility, surfaced to coordinators through an exception command center and a weekly review of recurring exceptions, with every recovery drafted, every approval named, every confirmed resolution written into memory and every standing fix governed, so the same lane, carrier and consignee stop producing the same exception every week without a single weight being trained on the fly.

Running the same shape elsewhere takes the same few things whatever the industry: an object model anchored one object per system of record, a thread history parsed and human-checked before anything bootstraps from it, permission-aware retrieval that only returns what the asking person could already

open, one gated write path verified against the tool that must consume it, hardware sized from the filed parameter count and precision upward rather than from vendor defaults, and a first phase narrow enough to baseline before anything scales, with a manual way of working kept warm beside it.

PART IV · CHAPTER 11

Praxis Recorded the Reasoning Behind Every Choice

Every decision in this paper traces to recorded reading on Praxis, whose contextualization, eight lenses and adopted and set-aside patterns are laid open in this closing chapter.

The conclusion gathered the design into one view and the shape it would take elsewhere. This chapter opens the record behind that view, so any choice in the paper can be traced to what justified it (Figure 9). Every design in the Vertical-Driven Architectures series is produced on Praxis, the design team platform for designing physical AI systems, and the reasoning is kept as recorded reading rather than reconstructed after the fact. What follows is that record for this paper: how the ask was contextualized, what each of the eight lenses could see, which patterns were adopted and which were set aside, and where the reasoning finally lands.

11.1 Contextualizing the Ask

The ask arrived as a requirement written in the operator's voice: a self-improving system for carrier exception handling, open weight models on the operator's own hardware, reading its systems of record and the threads, improving weekly from coordinator confirmations, without training a model on the fly and with nothing about shippers' freight leaving the operator's control. Praxis assigned the family and industry: the system of context family, in Supply Chain and Logistics, in the United States as the operator's own requirement states. What Praxis read, available on request: the requirement record carrying 21 requirements; the seven named source systems with their provenance and direction; the stack shelf with its product choices and its not-needed verdicts; the model register with footprints and licenses; the phase and workstream plan of 18 items across two phases; and the register of six failure modes with their mitigations.

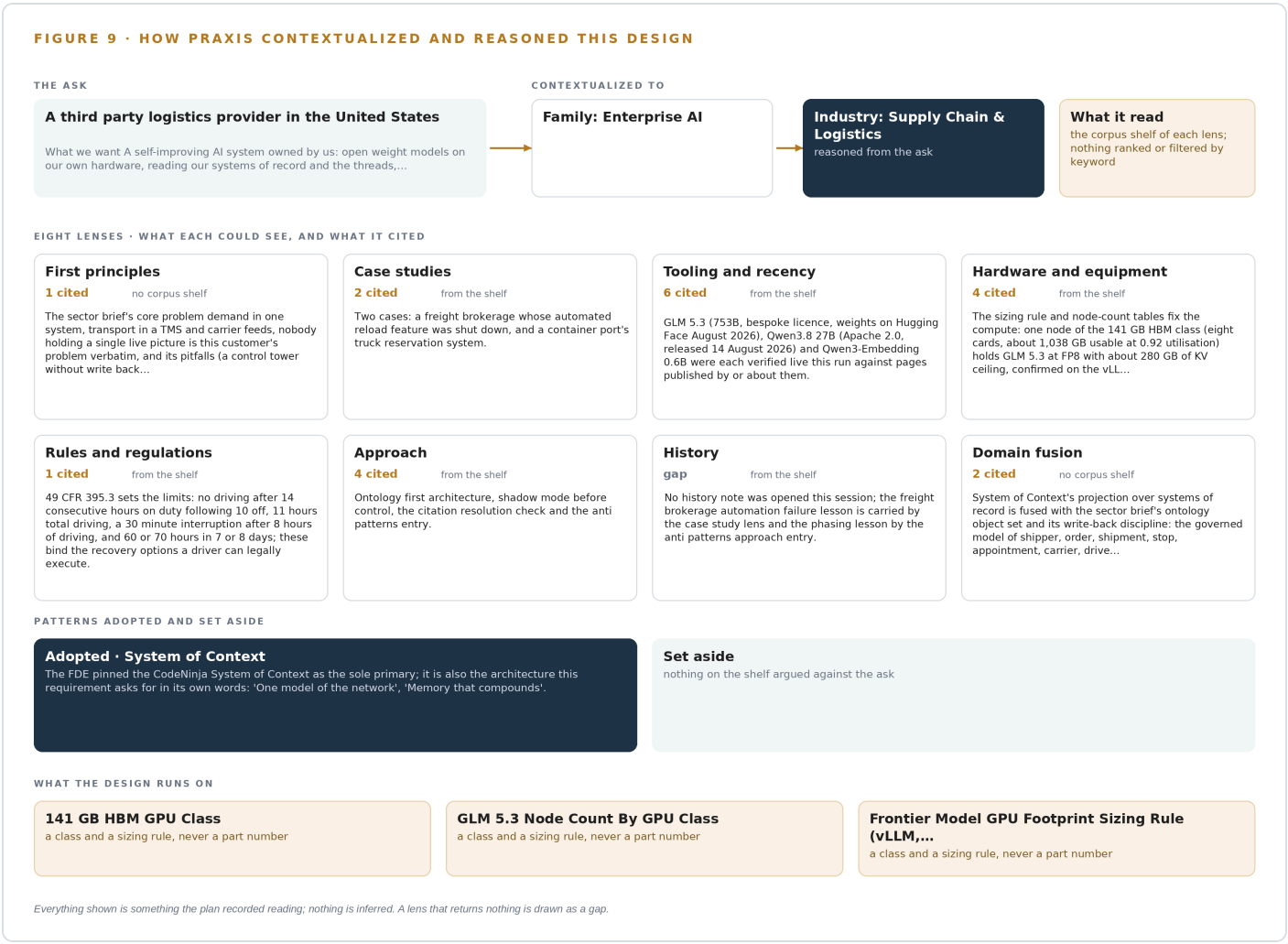


Figure 9. From the ask to the design: the family and industry Praxis assigned, the eight lenses and what each cited, the patterns adopted and set aside, and the equipment the design lands on.

11.2 The Eight Lenses

Each lens read the room from its own side, and Table 6 records what each could see, what it cited and what it contributed to the design. No lens is hidden when it returned little: the sizing arithmetic and the equipment verdicts stand on recorded arithmetic and product reading rather than outside citation, and the table says so.

Table 6 · The Lenses and What They Contributed

LENS	COULD SEE	CITED	WHAT IT CONTRIBUTED
First principles	no corpus shelf	1	The sector brief's core problem, demand in one system, transport in a TMS and carrier feeds, nobody holding a single live picture, is this operator's problem verbatim, together with its pitfalls, a control tower without write back among them.
Case studies	the case study shelf	2	Two cases: a freight brokerage whose automated reload feature was shut down, and a container port's truck reservation system.

LENS	COULD SEE	CITED	WHAT IT CONTRIBUTED
Tooling and recency	the tooling and recency shelf	6	GLM 5.3 (753B, bespoke license, weights on Hugging Face August 2026), Qwen3.8 27B (Apache 2.0, released 14 August 2026) and Qwen3-Embedding 0.6B were each verified live this run against pages published by or about them.
Hardware and equipment	the hardware and equipment shelf	4	The sizing rule and node count tables fix the compute: one node of the 141 GB HBM class (eight cards, about 1,038 GB usable at 0.92 utilisation) holds GLM 5.3 at FP8 with about 280 GB of KV ceiling, confirmed on the vLLM sizing rule.
Rules and regulations	the rules and regulations shelf	1	49 CFR 395.3 sets the limits: no driving after 14 consecutive hours on duty following 10 off, 11 hours total driving, a 30 minute interruption after 8 hours of driving, and 60 or 70 hours in 7 or 8 days; these bind the recovery options a driver can legally execute.
Approach	the approach shelf	4	Ontology first architecture, shadow mode before control, the citation resolution check and the anti patterns entry.
History	the history shelf	0	No history note was opened this session; the freight brokerage automation failure lesson is carried by the case study lens and the phasing lesson by the anti patterns approach entry.
Domain fusion	no corpus shelf	2	System of context's projection over systems of record is fused with the sector brief's ontology object set and its write back discipline: the governed model of shipper, order, shipment, stop, appointment, carrier, driver and the rest, each anchored in its own system.

11.3 Patterns Adopted and Set Aside

Four patterns were adopted. Every source enters through the adapter tier, one integration adapter family carrying change capture from the systems of record, so no system is read directly and none is modified. The baseline comes before the flag: the pilot lane group's exception measures are recorded before any coordinator sees a draft, so improvement is measured against evidence rather than impression. Improvement runs on memory, not weights: every confirmed outcome compounds into the record the next decision reads, which is the requirement's own phrasing honored exactly. The write path is single and gated: the appointment change request is the one operational write, and it moves only on a named coordinator's approval.

Four were set aside, each with its reason. Fine tuning and on the fly training were set aside because the requirement rules them out and memory carries the improvement instead. Automated load re-tendering was set aside because the systems-first constraint routes every carrier swap through a person, and any later automation would be its own gated scope. The edge and video layers were set aside because the trucks are seen through the tracking feeds the operator already owns and the exception desk is

knowledge work. Two-node serving on the 80 GB class was set aside because the arithmetic favors one 141 GB class node at FP8 over joined nodes carrying BF16 weights.

11.4 Where the Reasoning Lands

The reasoning lands on equipment classes, which is where a design of this shape becomes buildable. The frontier inference cluster class, GPU nodes for 300B to 1T parameter open weight models, carries the frontier work surface, one node of eight GPUs holding the FP8 weights with room for concurrent coordinator sessions. The site inference server class carries the site tier, where the dense Qwen model parses threads and notes under an air-cooled server the facility room already takes. The GLM 5.3 node count by GPU class and the frontier model GPU footprint sizing rule turn any future model change into arithmetic rather than a re-design. Every figure in this chapter, as in the paper, was recorded reading on Praxis: the lenses, the citations, the patterns and the equipment classes were recorded as read and are available on request, and nothing here is inferred beyond what the record holds.

SOURCES

Source Register

Ohsrep. 2024. Employer and Supplier Charged After Warehouse Fatality, OHS Reps. https://www.ohsrep.org.au/employer_and_supplier_charged_after_warehouse_fatality_sn_821

Tanktransport. 2022. Feds Reviewing Autonomous Truck Mishap, Tank Transport. https://tanktransport.com/2022/09/feds-reviewing-autonomous-truck-mishap/

Cloudsecurityalliance. 2026. AI Recursive Self-Improvement as a Systemic Risk, CSA research whitepaper. https://labs.cloudsecurityalliance.org/research/csa-whitepaper-ai-recursive-self-improvement-systemic-risk-v/

Stanford. 2026. The Ungovernable Machine, CodeX, Stanford Law School. https://law.stanford.edu/2026/03/17/the-ungovernable-machine/

Springer. 2025. Algorithmic management is associated with psychological distress, musculoskeletal pain, and occupational accidents: a cross-sectional study in logistics, International Archives of Occupational and Environmental Health. https://link.springer.com/article/10.1007/s00420-025-02180-5

About CodeNinja

CodeNinja is a Middle Eastern American artificial intelligence lab and system of context company. It builds sovereign self-improving systems that organizations own, run inside their own boundary and improve with every decision they make.