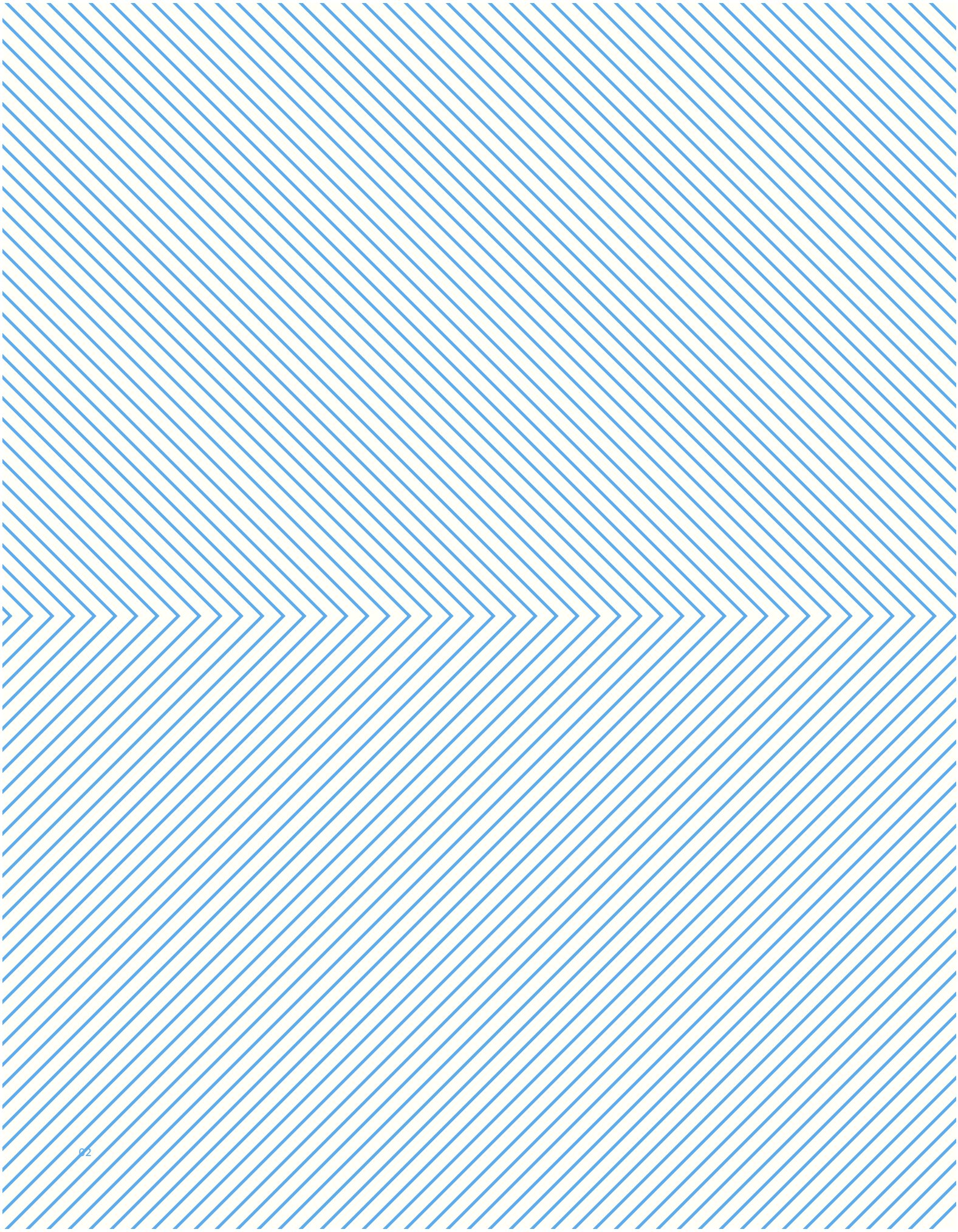
Palantir

Institutional Sovereignty in the Age of AI

15 steps that every government and company can take to compound their alpha in the age of AI



Foundations

I. Ensure zero data retention (ZDR).

II. Determine your AI decision tree.

III. Identify opportunities in your architecture.

Model Layer

IV. Guard against misaligned incentives.

V. Maximize model liquidity.

VI. Own the model flywheel.

Compute Layer

VII. Decide hardware based on assurance.

VIII. Own adaptable hardware for sensitive workflows.

IX. Verify compute you don't own.

Control Layer

X. Be model agnostic.

XI. Implement granular permissions.

XII. Audit & log.

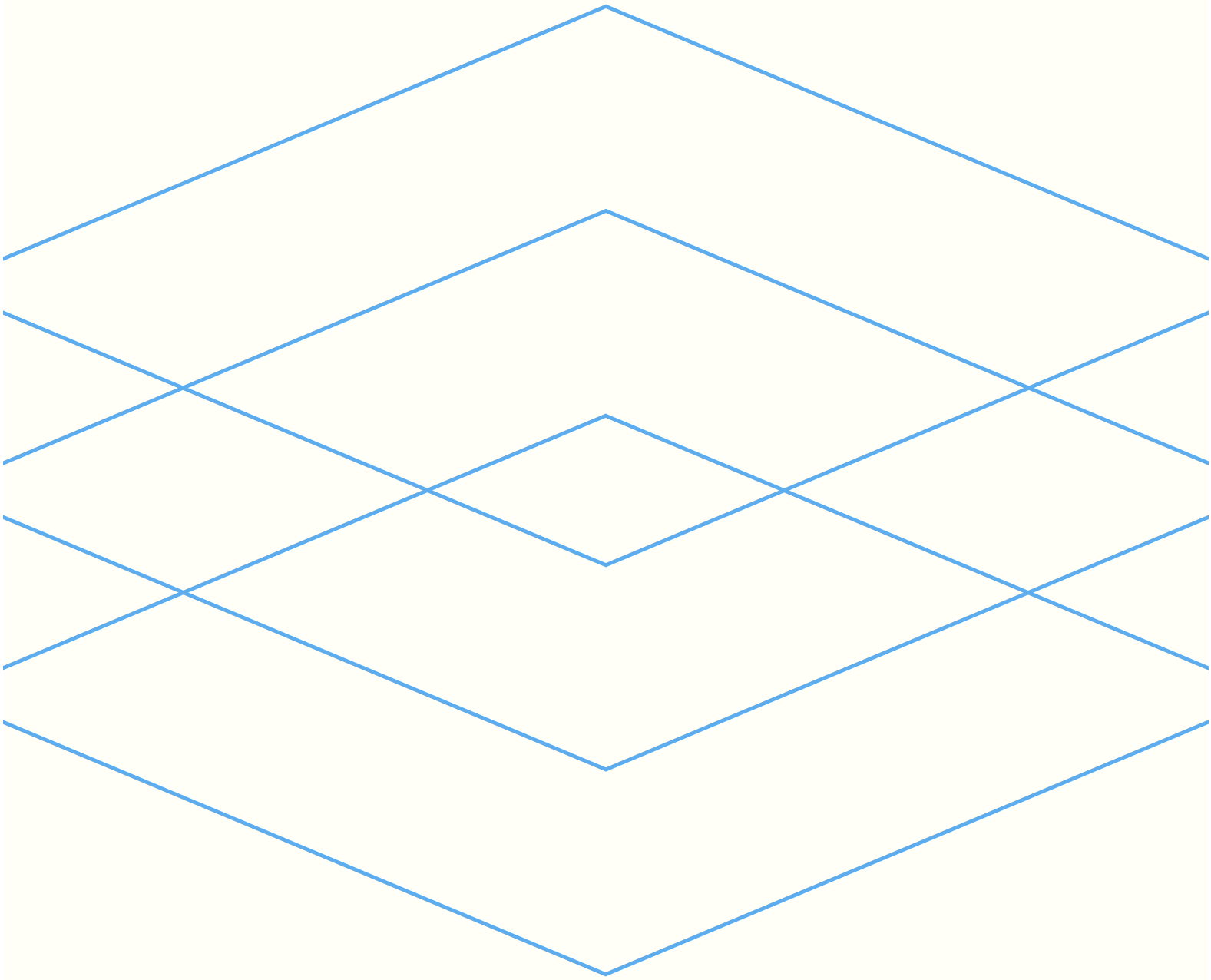
XIII. Practice adaptive cybersecurity.

XIV. Build by branching.

XV. Own the context flywheel.

Sovereignty is your alpha. It is your ownership of the value you create and your freedom to pursue new opportunity; a set of economic rights and the ability to expand them. In the age of AI, institutions are being led to believe their choices are narrower than they are — leaving critical decisions about how this technology is being deployed and applied to their data in the hands of others. In reality, you have complete agency over your AI model usage. This is a guide through those decisions.

Foundations



I.
Ensure zero data retention (ZDR).

The first step that any institution should take to achieve data sovereignty is to ensure that any LLM usage operates under a Zero Data Retention (ZDR) agreement. ZDR is a privacy concept whereby none of a customer's data in the form of prompts, outputs and telemetry is retained beyond the ephemeral processing required to generate the request. This ensures:

i) Your data is not stored to disk

ii) Your data is not used to train models

iii) Your data cannot be looked at by other humans

(ii) and (iii) are already nominally true of most enterprise contracts with model service providers. But the wording of contracts can be porous. ZDR, by ensuring that your data is never retained, presents structural barriers to illicit use. This is a vast improvement over simply trusting that your data will not be wielded against your interests, by either model providers or third parties. ZDR protects against misuse in both the present and the future by model providers, hedging against providers altering their enterprise data usage policies to allow more training on already-collected data. And it ensures that your data will not fall into the hands of other parties. Take litigation risk as one example: tellingly, model providers have noted in public non-ZDR contracts that they will delete retained data "unless we are legally required to retain them." Discovery during litigation can sweep up to millions of stored interactions with models, but that discovery cannot reach data fed to a model under a ZDR because that data was never stored.

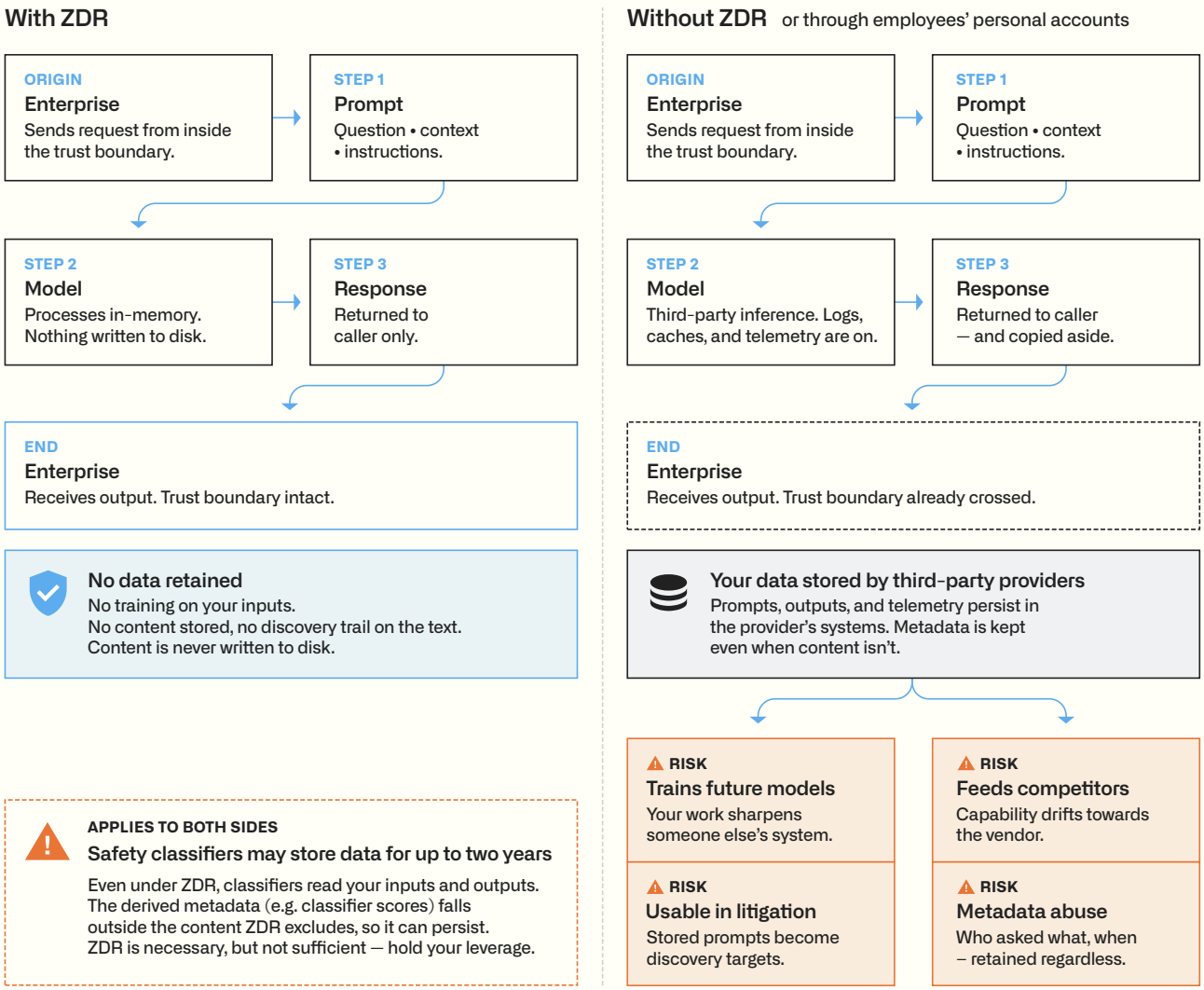
ZDR is a necessary condition of data sovereignty, but it is not a sufficient one. ZDR must be negotiated with model providers. And negotiation requires both initial and continued leverage, otherwise providers may not agree to it. ZDR also operates on a per-provider basis, meaning that it must be negotiated with separate model providers. Negotiating multiple ZDR agreements is a costly endeavor that incentivizes consolidation toward a single model provider. That pull must be resisted. It is important today that you use ZDR agreements such that your data never becomes a part of the provider's weights or telemetry. However, you must also continuously retain the leverage to ensure that tomorrow model providers may not change the rules of the game. A breakthrough in ZDR with one model provider should be used as leverage when negotiating with another. ZDR may take many forms, and as framed by providers does not always offer complete protection. You should zoom in on the following technical specifics when negotiating ZDR, to avoid some common pitfalls. For example, model providers in some cases may maintain monitoring logs that contain customer content and metadata derived from customer content, such as safety classifier outputs. ZDR under some labs' public terms only explicitly exclude customer content, but not the metadata derived from it, which could present opportunities for abuse. Make sure to cover off these common ancillary vectors for providers to undermine your data sovereignty.

I.
Cont.

This includes ensuring that no human has access as a matter of course, to any logs maintained by a model provider.

Extraction-Prone Models (EPM) are those (i) trained by third party firms with a structural incentive to replace or compete with your enterprise and (ii) used without a ZDR agreement. EPMs pose the greatest risks of tribal knowledge being extracted from the enterprise, and the greatest risk of that knowledge being subsequently wielded against the interests of the enterprise. A zero trust posture assumes that all third-party models used without a ZDR agreement are EPM. In other words, enterprises should assume that any non-ZDR frontier model is extraction-prone.

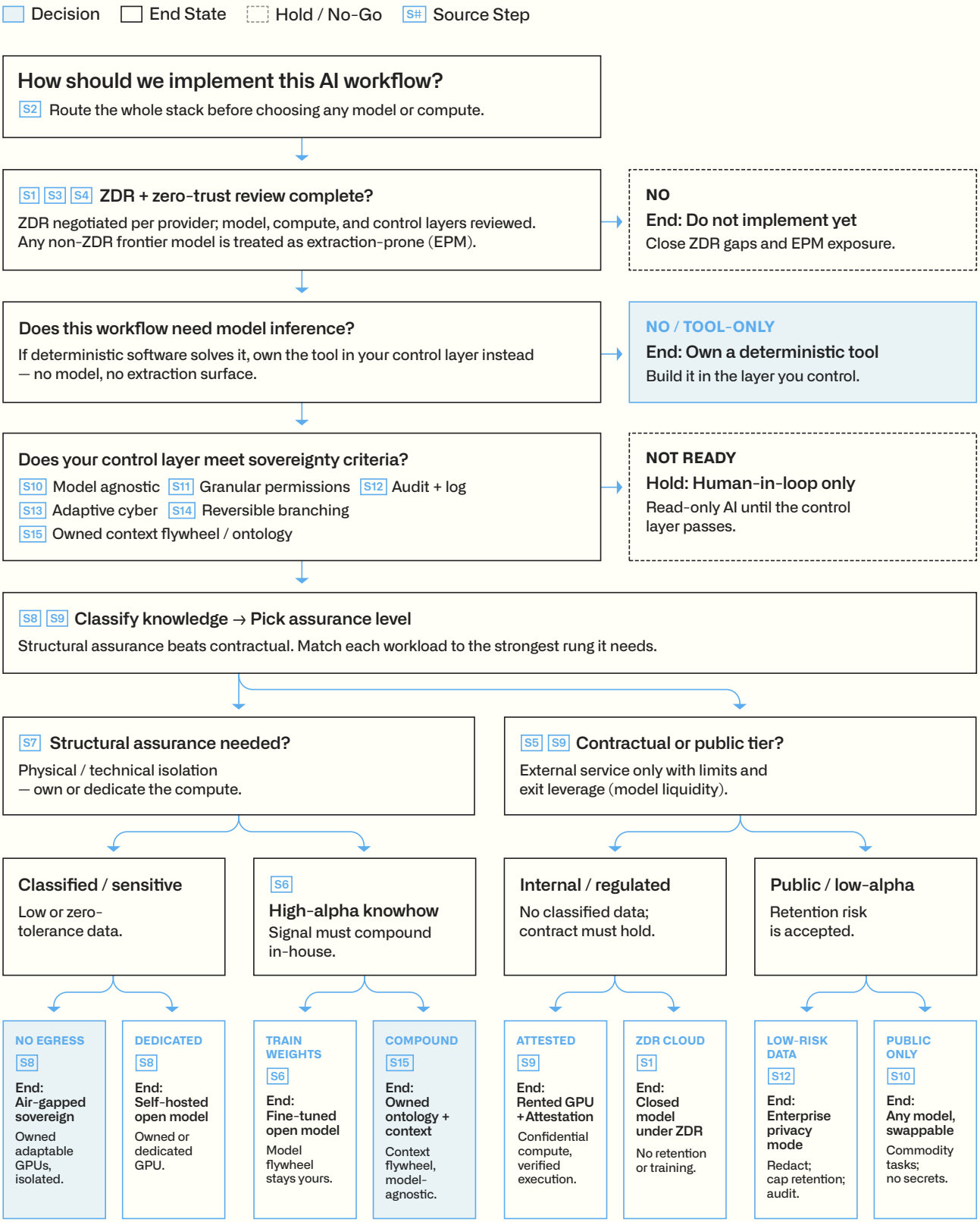
Fig. 1 ZDR Architecture



II.
Determine your AI decision tree.

Sovereignty is a series of decisions across the technology stack that you make for every workload. These decisions should be made deliberately in order to maximize capability, cost, and control according to your priorities. Below is a general decision tree for that logic, to be tailored to your unique architecture.

Fig. 2 AI Decision Tree

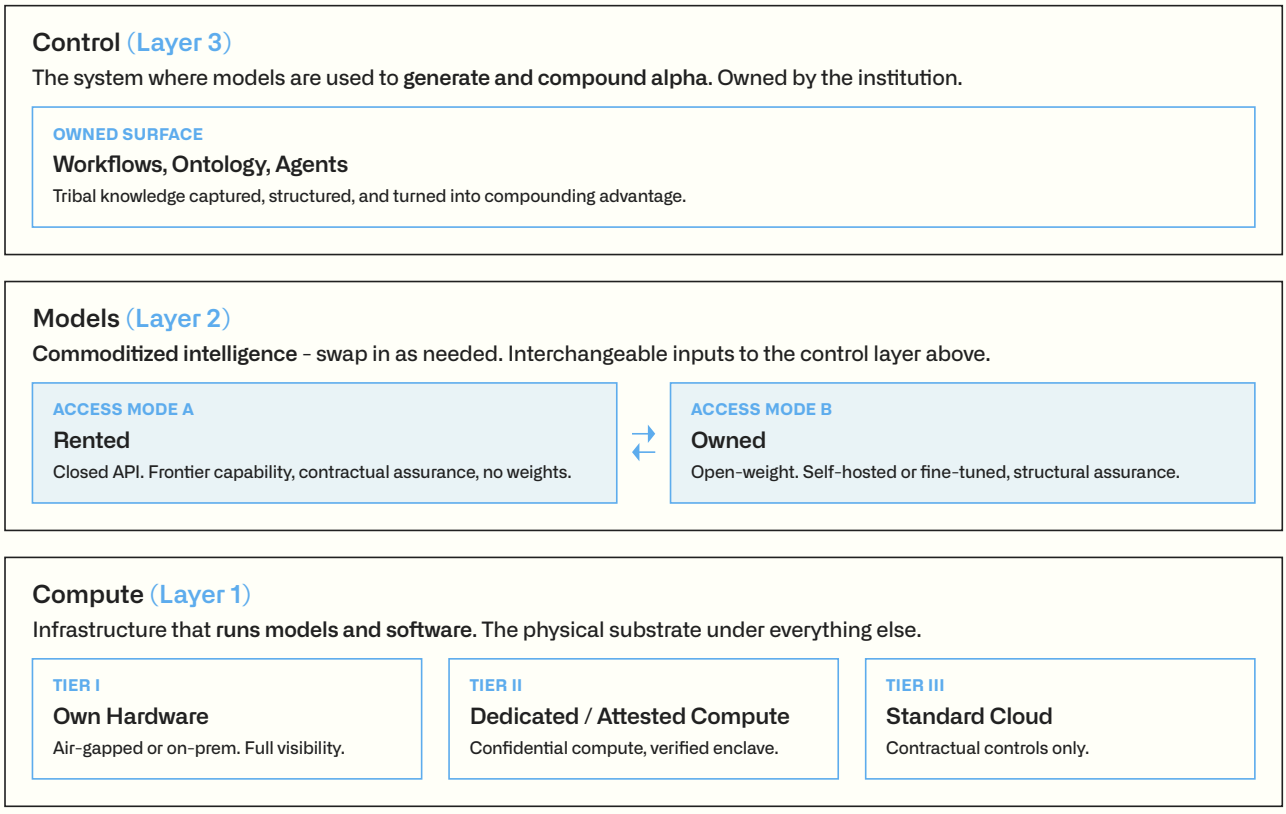


III.
Identify opportunities in your architecture.

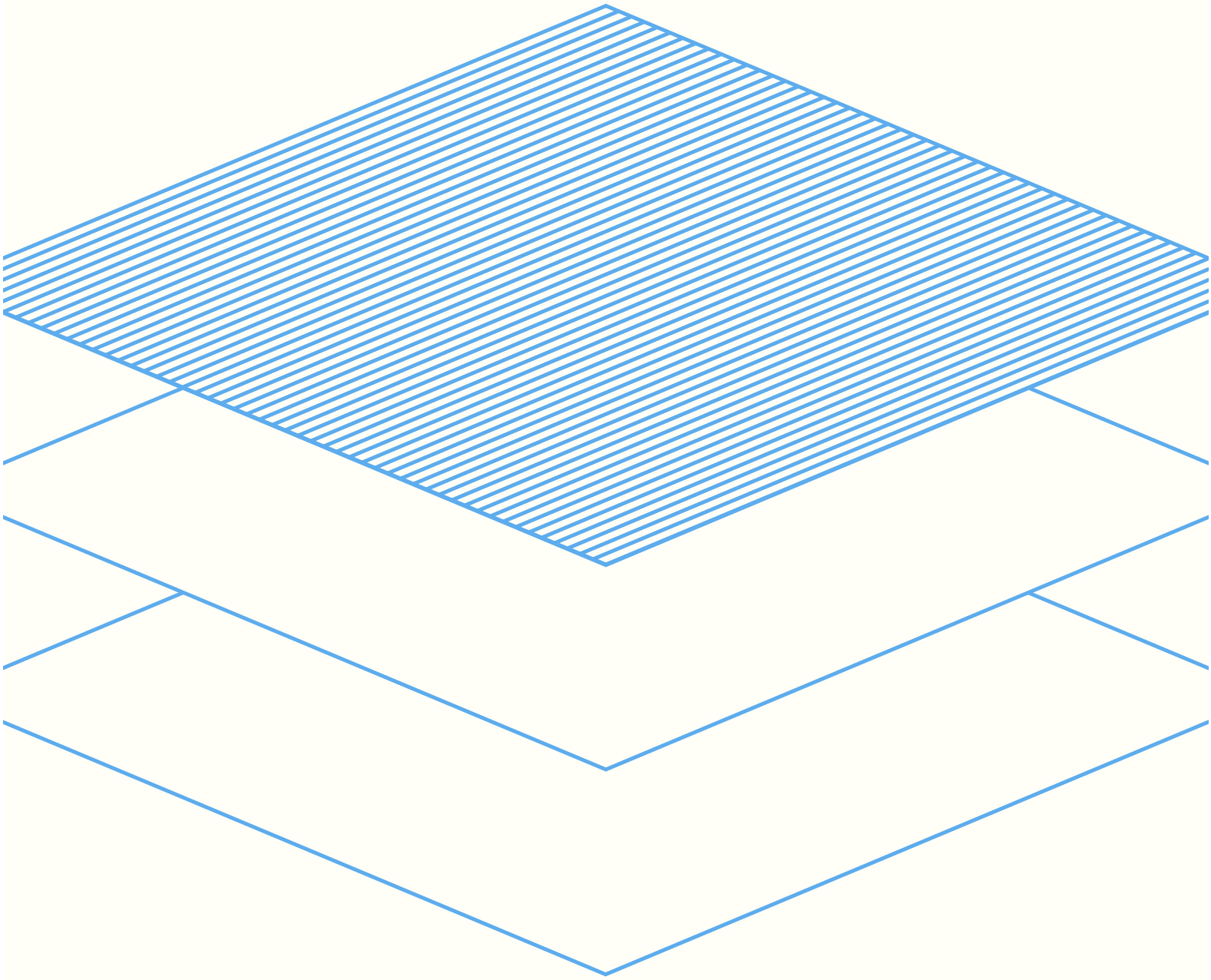
The first step toward AI model sovereignty, the ability to control how your data is used by third-party AI models, is to scrutinize your technology stack for opportunities. Your data can be operationalized by AI models in ways that it never before could, and you can substantially increase your alpha if you use it wisely. There are three layers where architecture changes can increase your sovereignty: the model layer, the compute layer, and the control layer. **At the model layer:** You need to know whether your architecture is model-agnostic, equipped with granular permissions, and positioned to enable value flywheels that you own. Begin with the models themselves. Do you have the freedom to move your data across providers on your own terms? Do you control where your institution's intelligence ends up, or has it already started leaking into someone else's weights through training on your prompts and outputs? **At the compute layer:** Do you have full visibility into the infrastructure running your models, down to who can access it and under what conditions? What level of assurance do you have in the security of that infrastructure, to include technical and contractual commitments? **At the control layer:** Review. Are you positioned to capture and secure your tribal knowledge at the moment it is created? Are your most valuable workflows running on infrastructure you control? And can you take your institution's knowhow and turn it into a compounding advantage?

Every institution should immediately build a team to conduct this sovereignty review. If you do not have the resources to do this internally, Palantir is offering no cost support to all of its partners through outreach at: sovereign-ai-os@palantir.com.

Fig. 3 Architecture Layers



Model Layer

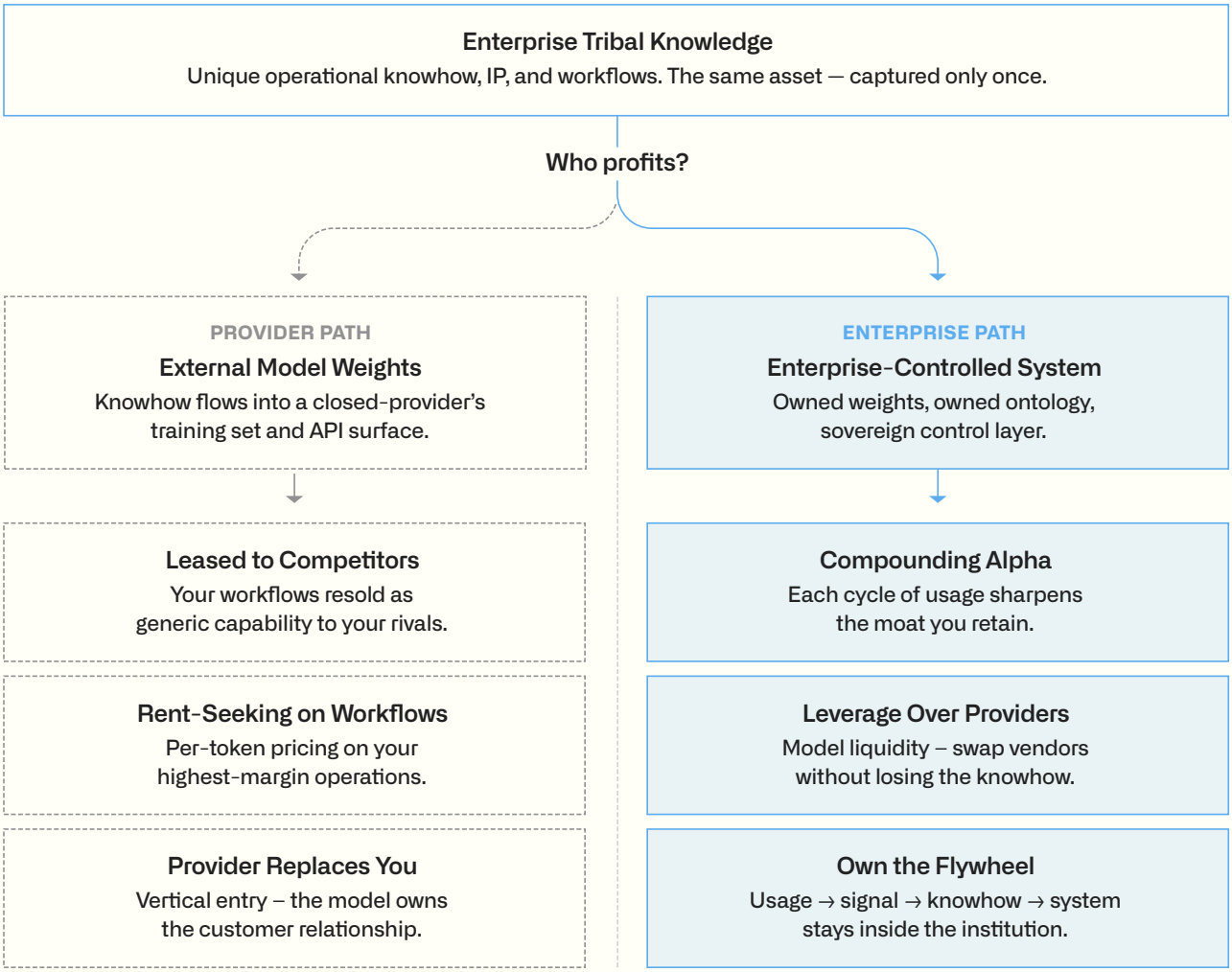


IV. Beware misaligned incentives.

AI model providers have a structural incentive to migrate as much intelligence from enterprises into their model weights — the encoded intelligence of these models—as possible. Doing so allows them to lease your insights to competitors, maximize token usage, rent-seek on your high-margin workflows, or in the most extreme case: completely replace you. As the financial pressure to grow revenue mounts, model providers have already begun directly competing with their enterprise partners. If your incentives were aligned with model providers, why do they charge per token and not as a proportion of the value they help you create?

The rational posture toward AI model usage in the enterprise is thus one of zero-trust — minimizing the ability of increasingly powerful model companies to turn your alpha into generally available intelligence. Even if AI model providers do not yet have a plan to commoditize the tribal knowledge of your business into the weights of their models, they have a growing incentive to do so. Institutions must protect their data and maximize their sovereignty through the following practical steps.

Fig. 4 Tribal Knowledge Ownership



V.
Maximize model liquidity.

To maintain your leverage over model providers, you will require model liquidity — that is, the ability to switch to any model with no friction. The past four years have presented a wide array of threats to institutions who operate and rely on frontier models: simple outages, changes in data retention policies, fluctuating refusal policies, and geopolitical restrictions. Reliance on a single model provider makes any of these threats existential. True model liquidity requires:

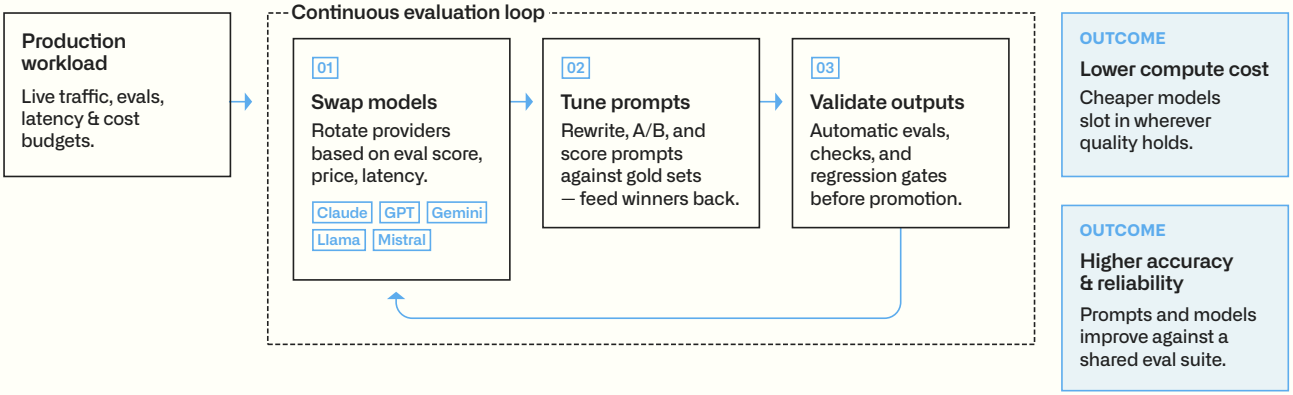
i) Access to a wide market of models and model providers. Widespread model availability is a world toward which we all should work.

ii) Infrastructure (such as AIP Evolve) that lowers barriers for switching between different models and providers. Model agnosticism is an attribute of your systems that you should work toward developing today.

That combination protects you from threats to individual model providers but also provides continued leverage against them. If a model provider threatens to mishandle your data, you can in turn credibly threaten to take your business elsewhere. That provides a structurally favorable power dynamic that can allow your data to be held securely.

The ability to switch between models quickly allows you to be far more intentional about using the best model for the task at hand. Combining this with a comprehensive evaluation suite (such as AIP Evals) enables you to optimize performance and minimize for cost and latency. It would be shortsighted to limit your organization to a single lab's intelligence. The broader ecosystem of labs has talent and dynamism to which you should be maximally exposed. As different labs and technologies compete for dominance in different domains, you want to ensure that you can reap the benefits as they present themselves. This could be as simple as a new model from a different lab, or it could be completely novel forms of intelligence. Language model intelligence has been transformational, but is by no means the final form factor of model intelligence. In this spirit, it is key that organizations have modular processes that can readily slot in and out model intelligences.

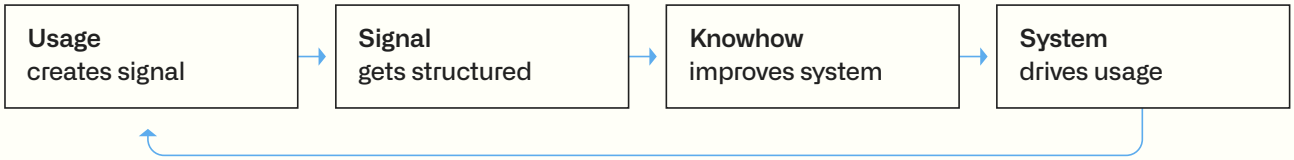
Fig. 5 Evolve Agent Harness



VI.
Own the model flywheel.

All institutions are differentiated by their tribal knowledge. Successful institutions will not only rely on historically successful knowhow but will create compounding feedback loops within their organization such that new knowhow is developed. For institutions to retain sovereignty and enhance alpha, they — not their model provider — must own this knowhow and the feedback loop that it generates. This means protecting your intellectual property, allowing you to own and accelerate your knowhow flywheel at the model intelligence layer.

Fig. 6 **Model Flywheel**



Loops back – the institution that generates signal must also capture it

A flywheel can be highly potent at the model layer: usage generates signal, signal is captured and structured, structured knowhow improves the system, and the improved system generates more and better knowhow. This loop only compounds, however, if the institution that generates the signal is also the one capturing it, with robust sovereignty controls (including negotiated ZDR). If that capture occurs in an AI model company without robust ZDR protections, you are handing away that compounding value to that third party, who can then productize your unique insights.

AI model providers may complicate your ability to own the compounding loop. Under the public Terms of Service from many frontier model providers, you cannot train your own models on any output generated by their models because of distillation concerns. As a result, if you continue to rely wholly on proprietary models, you may not have the ability or right to compound. One remedy to this is using open weight models, which offers several benefits:

i) You can use these models as teacher-models, training your own model intelligence on their outputs without the distillation restrictions that closed model terms of service impose.

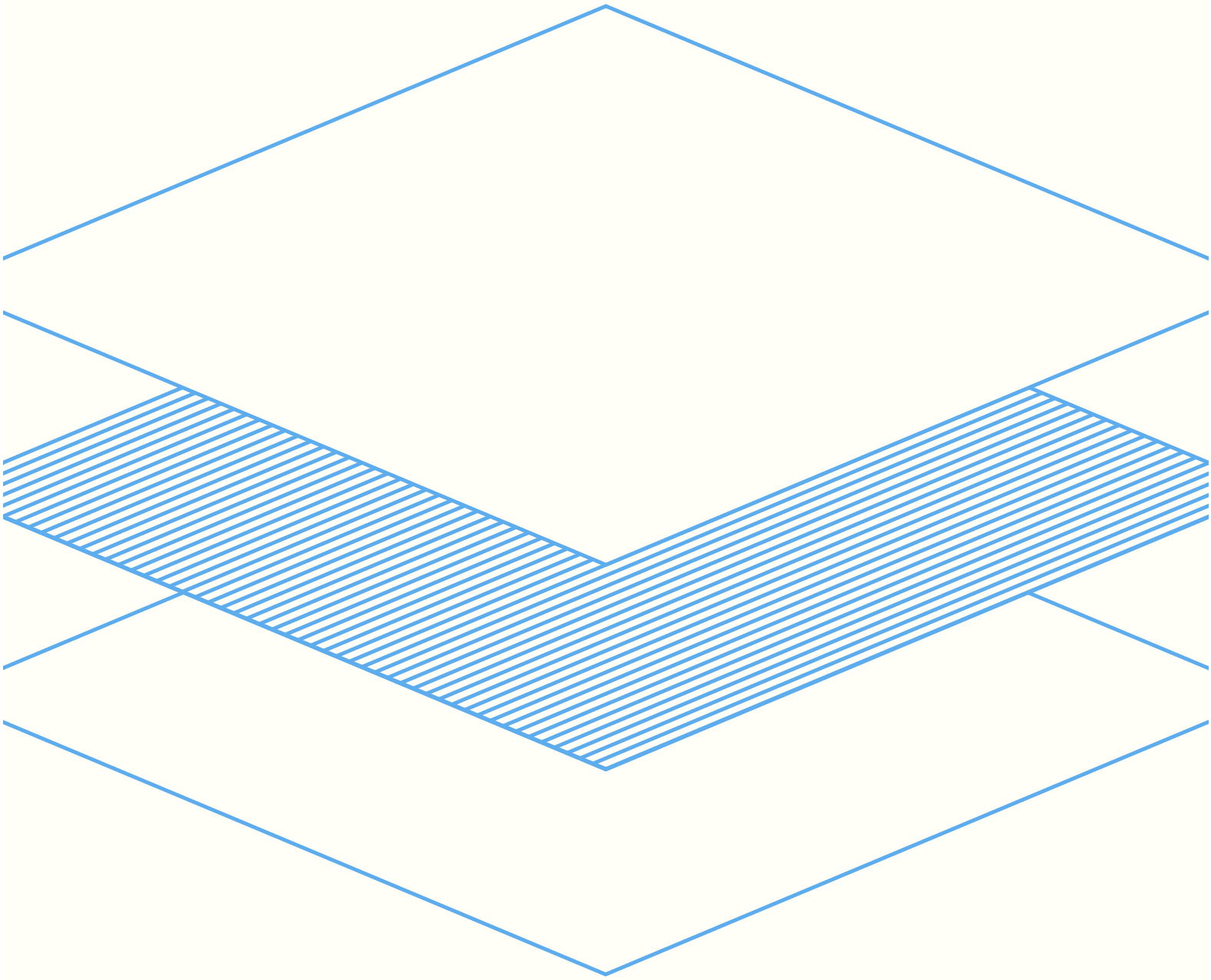
ii) You can update the weights of open models based on your institution's own knowledge, so the compounding loop described above runs on infrastructure and weights that you own, rather than on a provider's closed system. iii) You can be shielded from any of the threats that accompany frontier model usage, such as policy changes or restrictions.

VI.
Cont.

This solution has tradeoffs. US frontier models are materially more capable than open-source models today. That gap is narrowing, but it is unlikely to ever close. There are therefore capability tradeoffs that would be made in favor of cost, and more importantly, sovereignty. Many tasks, however, do not require frontier intelligence capabilities, and less capable models can be orchestrated to achieve outcomes beyond their class, particularly when fine-tuned. All of this relies on having capable open weight models.

We therefore need now more than ever capable American open weight models. The recently released NVIDIA Nemotron models mark a watershed moment in the US open weight ecosystem, though they are currently the exception rather than the rule. No country is more capable of leading open-weight models than the United States. This is a call to arms.

Compute Layer

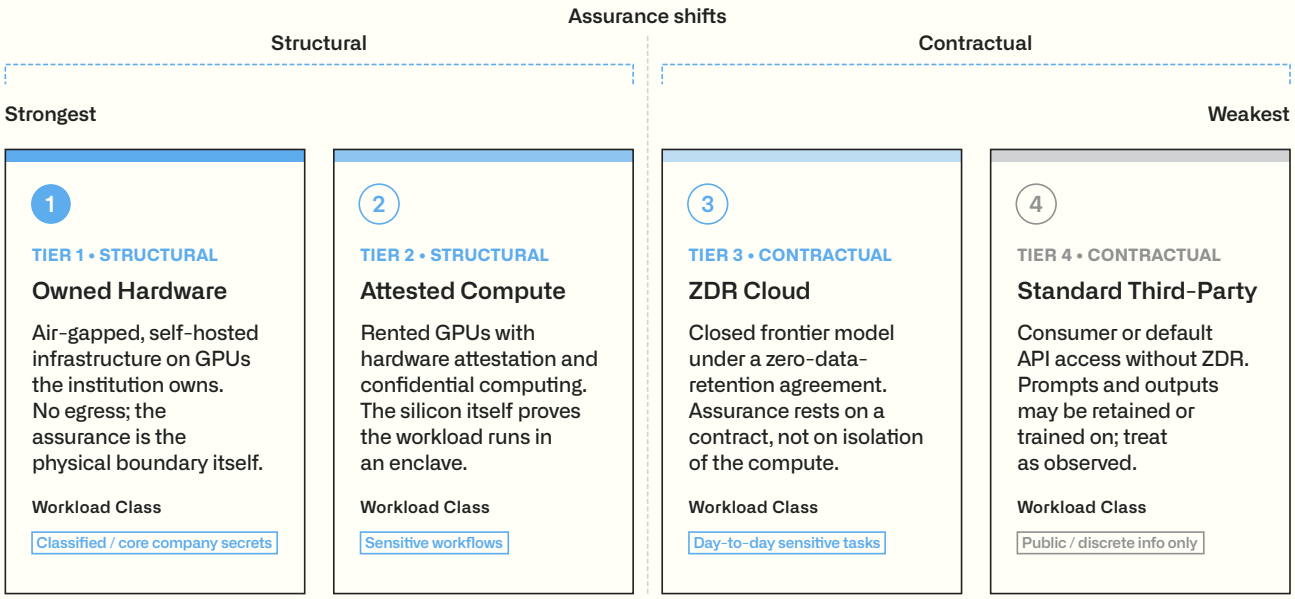


VII. Decide hardware based on assurance.

To ensure that your data is protected, you should make your decisions based on assurance — the underlying mechanism that backs the guarantee that your data won't be retained or exposed. Assurance can be contractual or structural, the latter being stronger.

At a structural level, you can secure your data by ensuring the physical or technical isolation of your compute. That can include self-hosting your own models, whether open-weight or in-house, and its infrastructure. To be viable at the mass-commercial level, this requires redundancy and security. You can also rent GPUs, avoiding the CapEx of purchasing them yourself and still enjoying the benefits of self-hosting models.

Fig. 7 Levels of Assurance



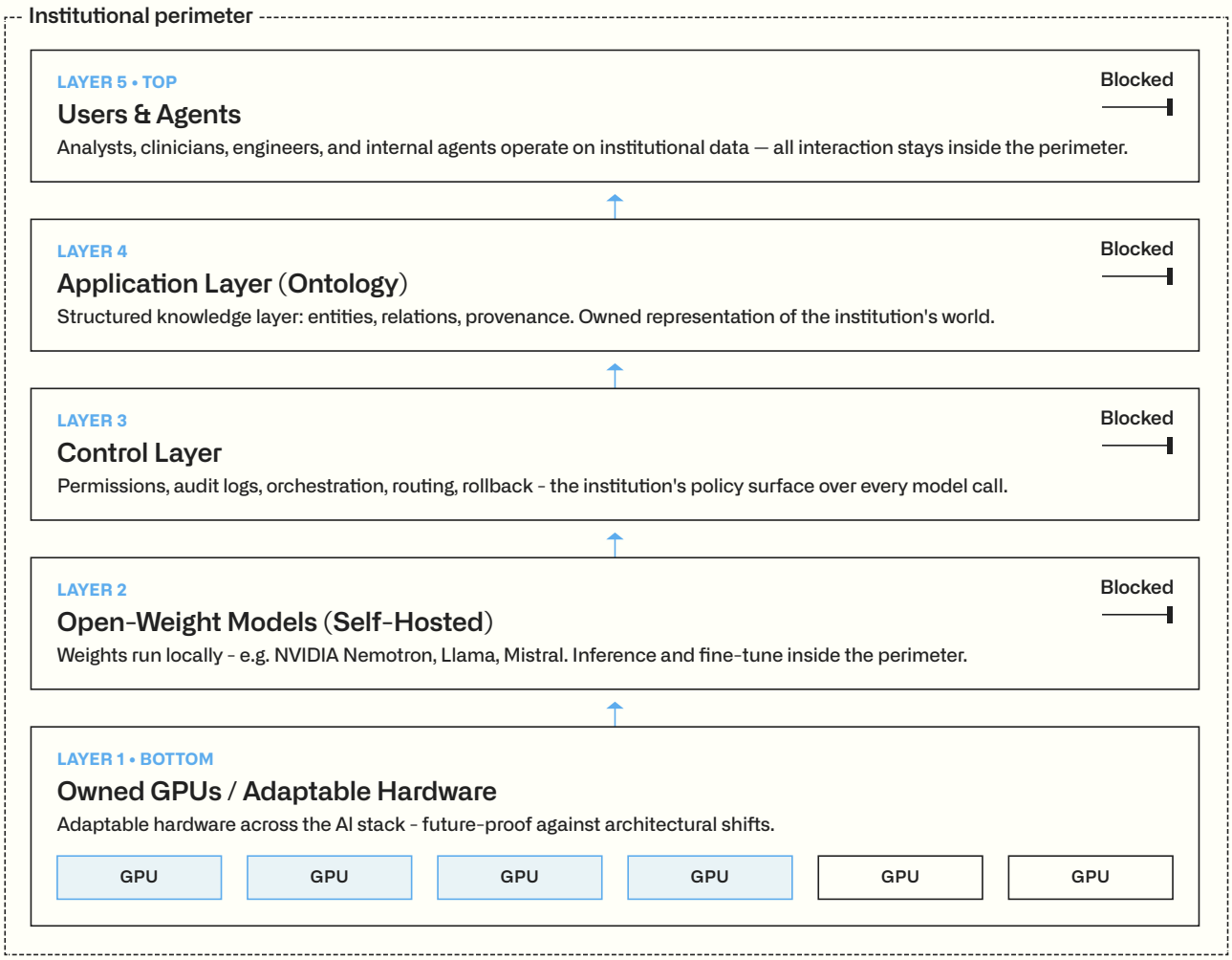
Either one of these strategies requires tradeoffs in the current world of highly capable closed-weight models, as discussed previously. Yet, there are conditions under which such tradeoffs are worthwhile. For one, each generation of open-weight models has packed more intelligence into less compute — so any investments in compute are likely to become more valuable over time. Such a strategy may also be cost saving in the long term. We have already seen that open source or open-weight models that are locally hosted can perform tasks for prices far cheaper than that offered by the closed-weight frontier models. And finally, investments in hardware act as insurance against a tumultuous world: the availability of high-end chips is already heavily conditioned on geopolitical or regulatory contingencies, but so too, it seems, is the availability of model intelligence via the cloud. Owned hardware insulates you from all these disruptions.

VIII.
Own adaptable hardware for sensitive workflows.

Owning your compute is the strongest form of sovereignty, and it is the cleanest route to physical assurance. Models are compute-bound, requiring costly, supply-constrained GPUs with high VRAM. These chips can take time to acquire, so if you are concerned with your institution's AI model sovereignty, that ownership may be worth pursuing now. Leveraging secure third-party cloud with hosted models protected by robust ZDR is also a highly protective form of sovereignty, if your organization and your workflows require frontier model performance.

Just as the model layer should be model agnostic to retain model liquidity and adapt to new forms of intelligence, so too should your hardware be adaptable. This is the case for GPUs or other generally programmable architectures. Many chips are optimized for today's architecture — which limits you substantially in the case that architecture changes. This has played out before as shifts from dense transformer models to reasoning models have required substantial architectural changes that a narrower chip wouldn't have survived unmodified. This tendency toward adaptability should be applied across the AI model hardware stack so that you may move as the playing field moves.

Fig. 8 On-Prem Architecture

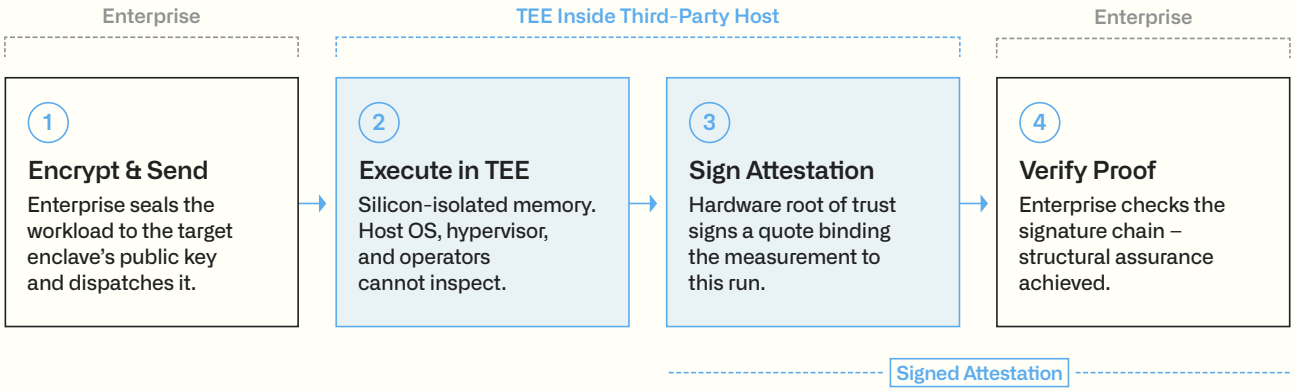


IX.
Verify compute you don't own.

When you must run on compute you do not physically control, such as rented GPUs and cloud compute, you should not rely purely on contractual assurance. Confidential computing allows you to make structural assurances even while using someone else's architecture and hardware. Through hardware attestation—a process whereby the hardware generates verifiable proof of execution—you may guarantee that your workload ran in a secure environment, out of the purview of the third-party host.

This ladder is how you should reason about every workload. For the world's most sensitive work — classified work for governments, and that which involves core company secrets — it can often make sense to run on hardware you own and isolate. Below that sits attested compute you do not own, then cloud models under a ZDR agreement, and finally, for public or discrete information that reveals little about your operations, there is limited danger in standard third-party usage.

Fig. 9 Hardware Attestation



Control Layer

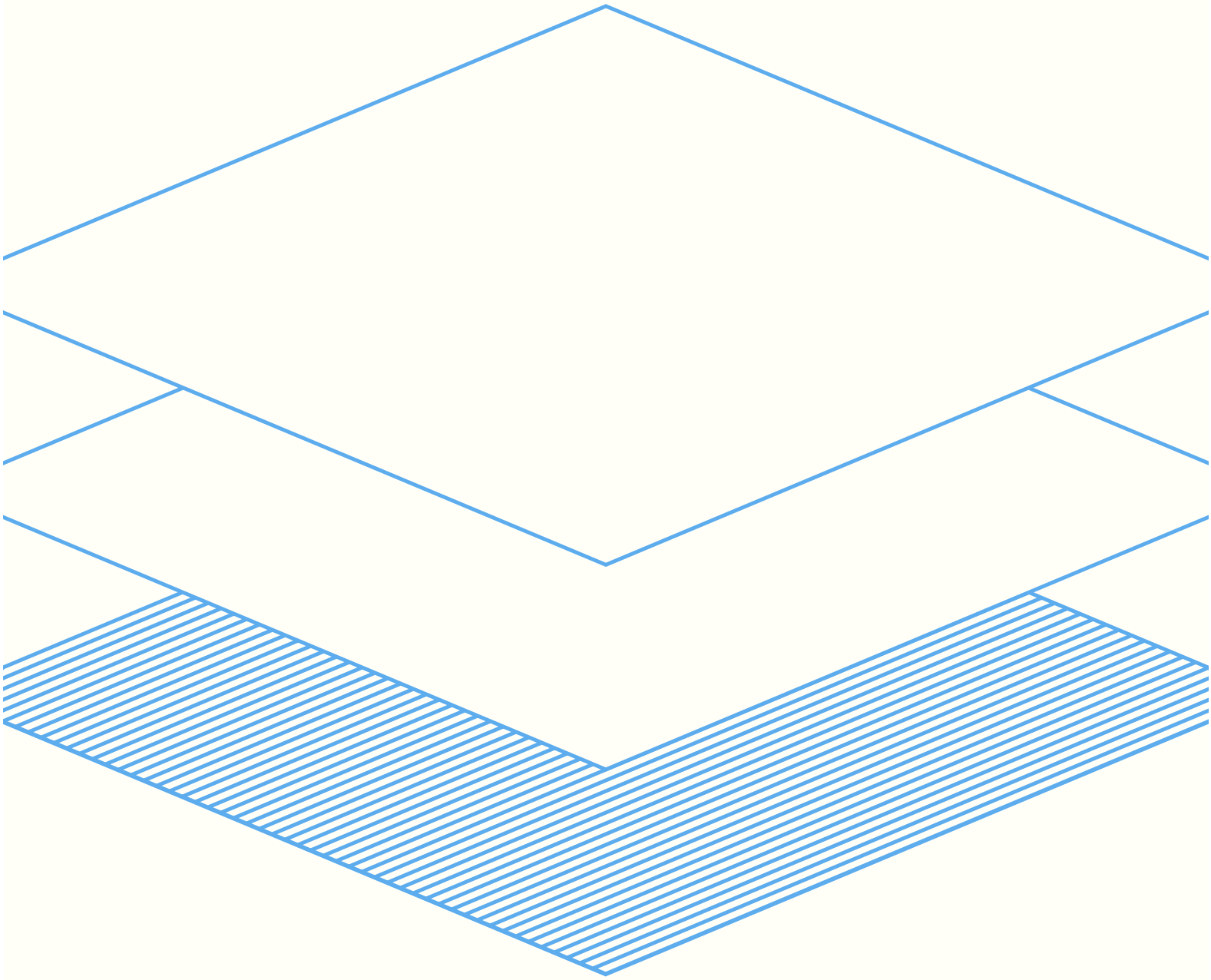
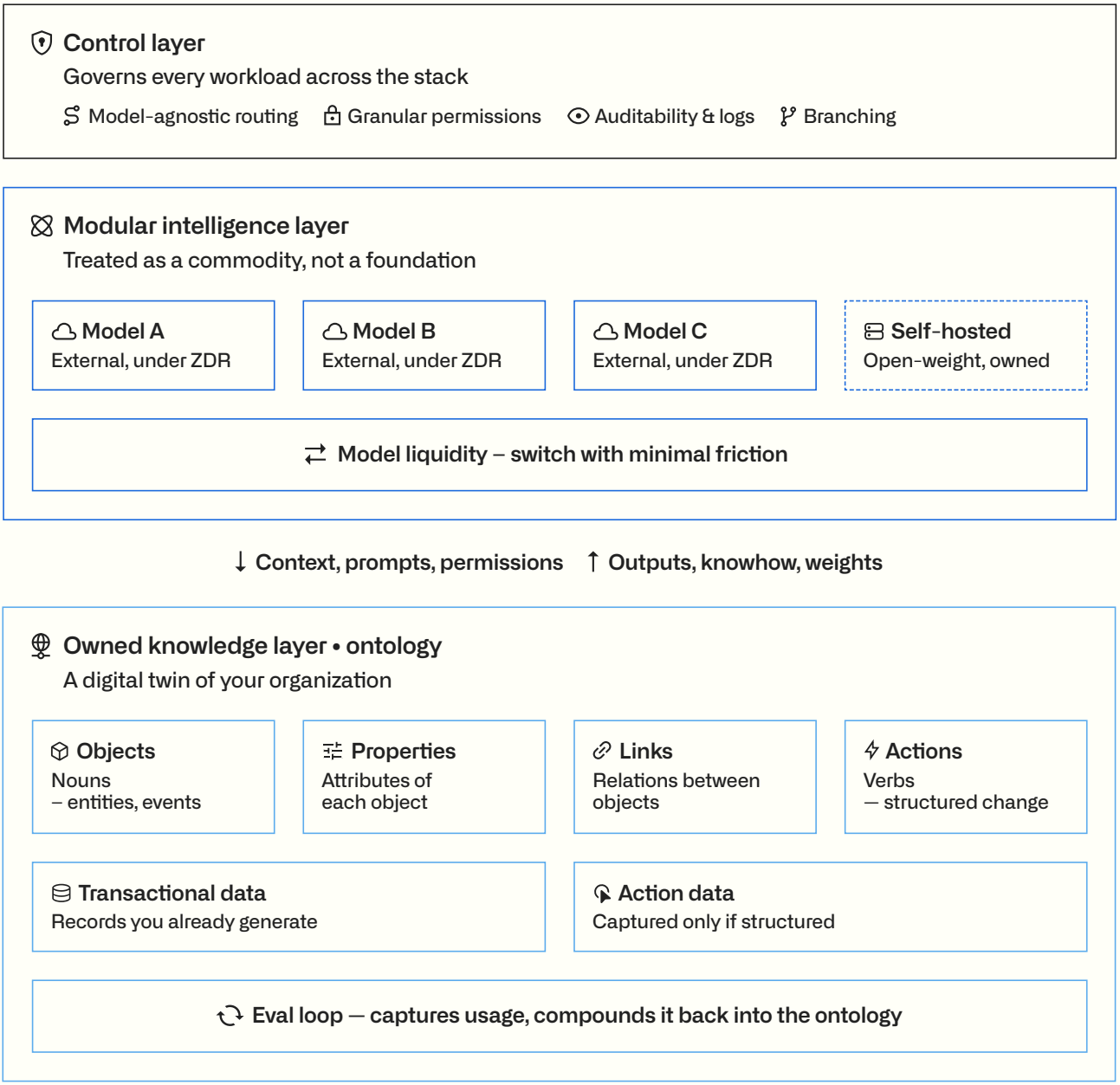


Fig. 10 Control Layer Elements



X.
Be model agnostic.

As stated in Section 5, model liquidity requires:

i) access to an array of models

ii) the ability to switch between them with minimal friction

A model agnostic control layer delivers both, configured such that no single model is the default, but instead allows the substitution of models according to your own preferences or through optimizing according to a set of constraints. This is the natural consequence of treating model intelligence as modular and distinct from your knowledge layer.

In using a model agnostic system, you are provided with the extra benefit of anonymity by aggregation. If your institution accesses models through a third-party model-agnostic layer, then many organizations will be hitting the same API endpoint. This will make your traffic indistinguishable to model providers from other organizations' at the point of inference, providing an extra layer of structural security for your IP.

XI.
Implement granular permissions.

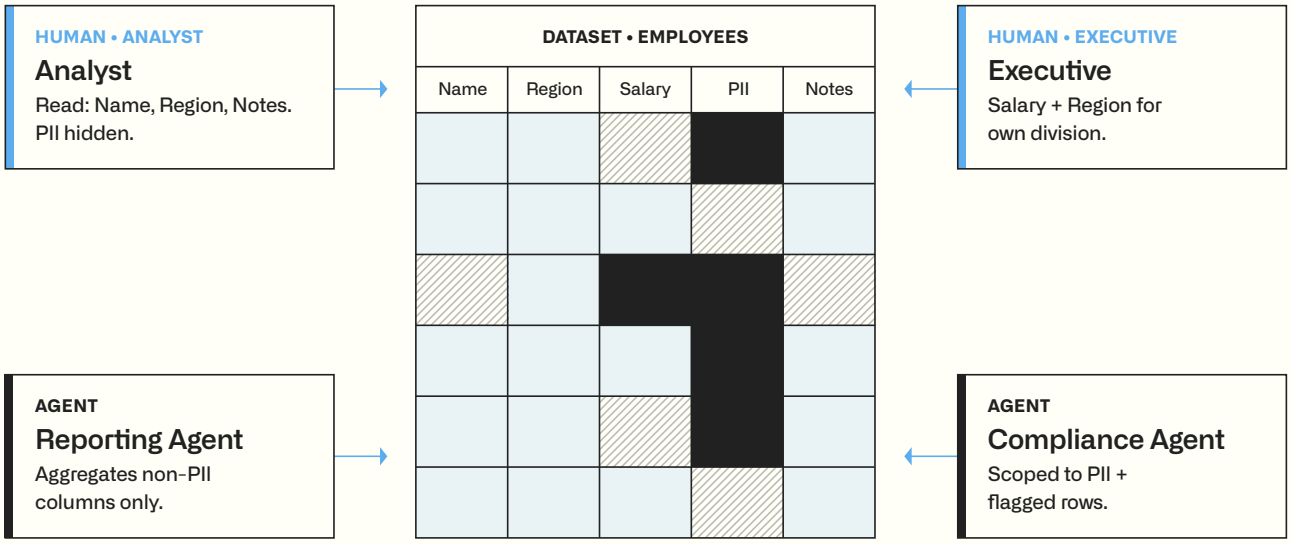
In a distributed data world, you may have thousands of datasets spanning all domains of your institution. Your permissions there are defined as data permissions, which should be as granular as a row on a data table. However, an organization is not structured as a set of tables and relationships between them, but rather as people, roles, relationships, and workflows. Access control architecture that is defined purely at the data layer must be translated into the structure of the real world manually and imperfectly, and kept in sync as the organization changes.

However, when you model your world as a digital twin, permissions are comprehensive and are built to reflect the structure of your organization. This includes role-, classification-, and purpose-based access controls that dynamically follow the lineage of your data. When certain individuals can perform certain actions or see certain data, this should not be framed as a data access policy, but rather as a statement about the functioning of the organization and the permissions are derived directly from this. Granular permissions and access controls can be defined and enacted by the people who actually understand the organization, not exclusively by engineers who must continuously translate on a case-by-case basis.

XI.
Cont.

This kind of access control is critical in the age of AI. If an agent needs access to your data and ontology in order to perform actions, you must be able to enforce precisely what kinds of data it can load, tools it can invoke, and actions it can suggest — otherwise it may enjoy far greater access to information than even your employees do. Relying on a model provider to respect your permissions without defining them robustly means giving an external organization a blank check to review your records and trusting they don't abuse it. That makes your data more vulnerable, and the ambiguity about what is and is not sensitive creates further risks of IP theft: a threshold question in many trade secrets suits is whether your organization's knowhow was sensitive enough to merit internal protections. A digital twin of your organization's permissions removes this uncertainty: the organizational constraints that govern human access are already defined within your ontology and can be applied in the exact same way to agent access controls.

Fig. 11 **Granular Permissions**



XII.
Audit & log.

Auditability allows you to detect the misappropriation of your data; further, it allows you to operationalize that data for your own purposes. There are two approaches to auditability: first, auditability in what you send to your models, and second, auditability in what your provider does with it.

The first is more in your control. Your full audit system must be able to monitor the actions that agents have taken and the data models have accessed. A log of this information can allow you to precisely audit the interaction of models with your knowledge layer. Testing new models from your providers against material that contains answers only you could have provided could, if triggered, serve a clear indicator of misappropriation.

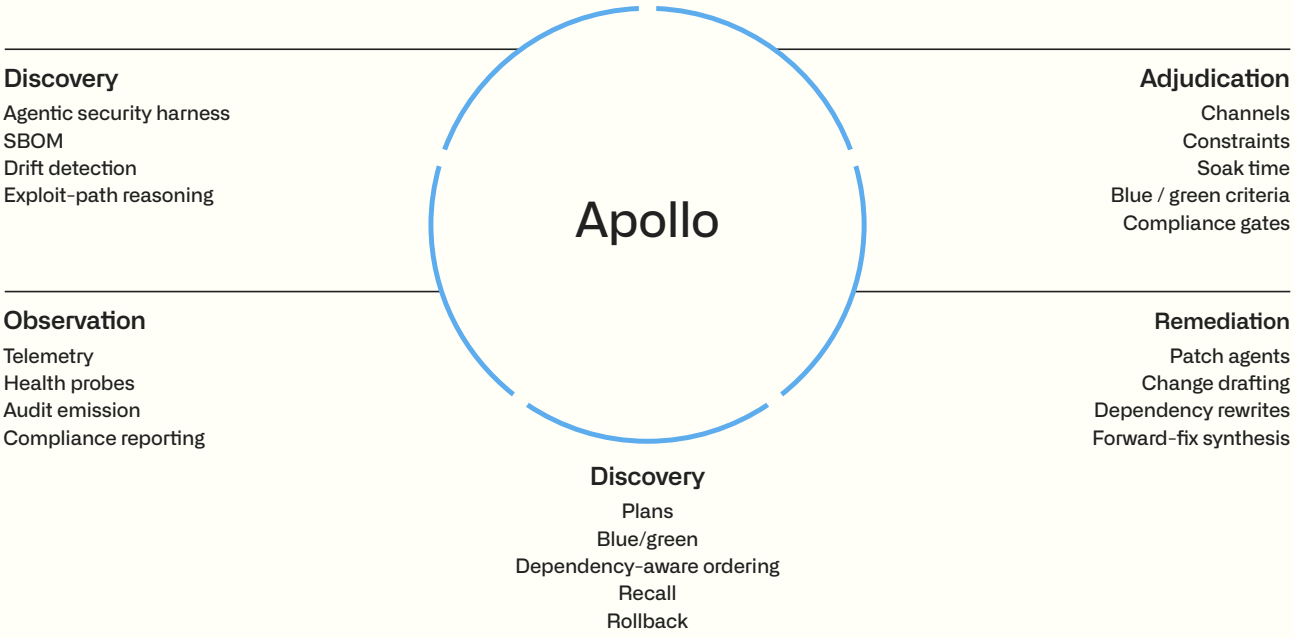
XIII.
Practice adaptive cybersecurity.

A core pillar of sovereignty is IP protection; one of the most direct ways institutions lose IP is through insecure software. The most capable models of today raise the specter of cyber attacks at a speed and scale that institutions haven't had to defend against before. Your cybersecurity must be able to withstand this to protect your data and IP.

Model providers are fixated on the integrity of code, but the magnitude of emergent cybersecurity threats means that it should not be purely understood as part of the technology layer, but rather from a whole-of-organizational standpoint—something providers have little exposure to; an understaffed dev team is as big a risk as anything else. This posture must not be static but withstand continuous testing and update itself as new threats emerge through processes such as autonomous cybersecurity.

Cybersecurity should not be considered purely defensive. Expanded cyber defense gives you the liberty to ship faster and take new ground. Frontier labs release new products weekly yet simultaneously tell their customers to slow down and perform lengthy cyber scans before shipping their own software. This is out of a well-placed concern for cybersecurity, but it has a steep opportunity cost, guaranteeing that you will slow down and lose ground. That makes it key to build the organizational and technical capabilities to give you the freedom to move and gain ground at frontier speed with less risk. The test for any organization should be simple: is your rate of shipping software the same as it was six months ago? If the answer is yes, something is deeply wrong.

Fig. 13 **Apollo Security Forge**



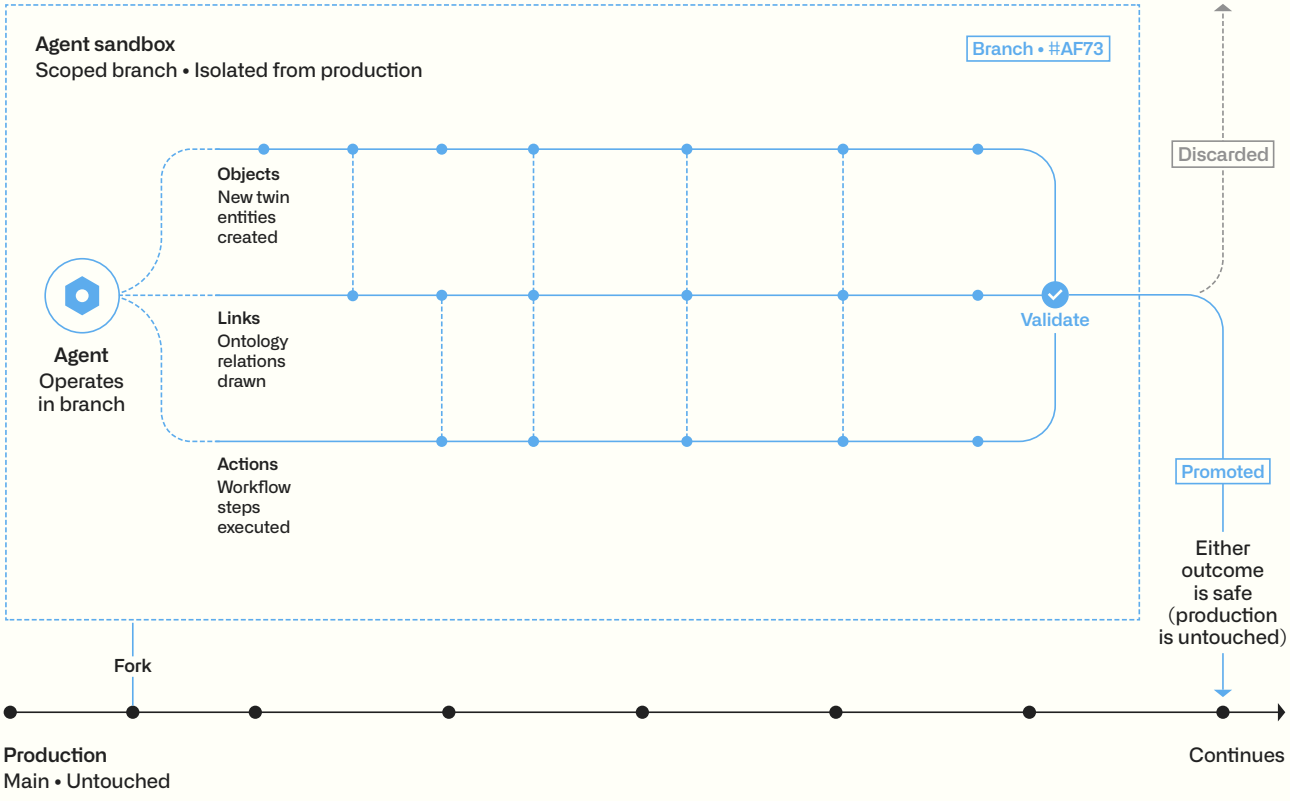
XIV.
Build by branching.

Control need not come at the cost of freedom, but rather, done correctly it is the mechanism that grants it. This is exactly the case in branching — that is, the ability to fork states, typically of codebases, before pursuing certain actions — which is another feature that should be present in your control layer to increase speed and enhance sovereignty.

Branching is instrumental to control in an agentic world by ensuring that any action an agent takes can be reversed. This control equates to sovereignty. If agent actions are reversible, agents can be given a far wider surface area to act over as the costs of their mistakes are lower. Without this reversibility, you are forced to choose between restricting agents from touching anything real or accepting unbounded risk. Branching collapses this double negative and, in doing so, allows far greater experimentation for both humans and agents, because any path can be tried and tested before needing to commit to it.

In the maximally valuable case, branching extends beyond any single action to the entire organization itself. Just as a codebase may be branched, so too may an organization's digital twin. Entire workflows or architectures may be trialed and compared alongside the status quo to determine if they should be adopted before being committed.

Fig. 14 Agentic Branching Example



XV.
Own the context flywheel.

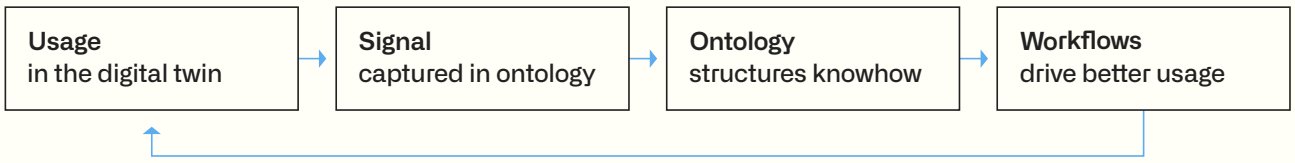
Just as you should own the compounding effects of model improvements made with your data, you should create and own the compounding effect of your own institutional knowledge. While the model engineering flywheel compounds through improvements to its weights, the context engineering flywheel compounds through the structured record of your institution's actions and decisions, captured as an ontology.

The flywheel at the control layer works in a parallel manner to the flywheel at the model layer, whereby usage generates signal, signal is captured and structured, structured knowhow improves the system, and the improved system generates more and better usage. Here, however, the signal is not from model inference, but rather it is derived from your organization's actions, both human and agentic, in your digital twin.

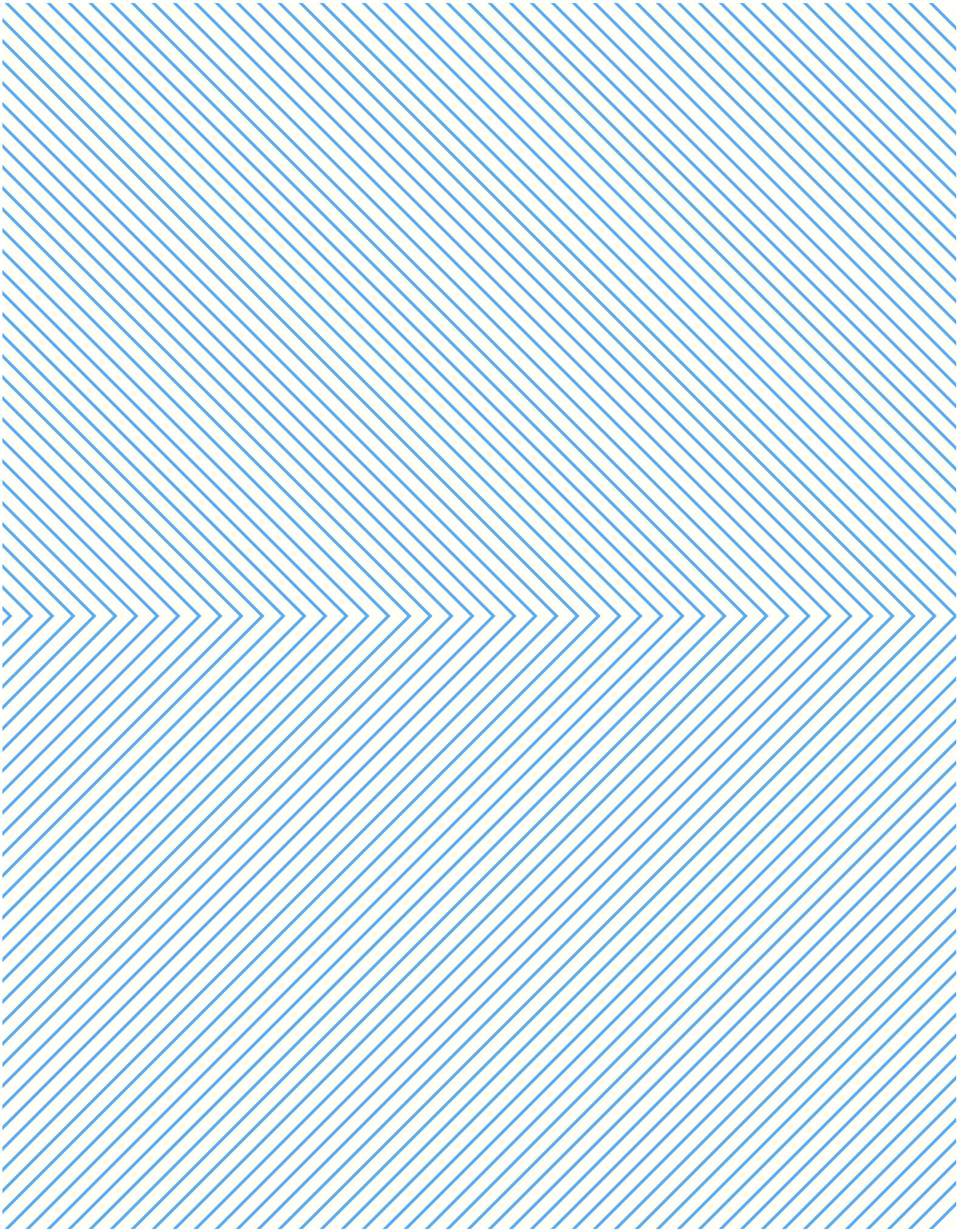
The quality of this flywheel is gated by the quality of the software on which your organization operates. Software that meaningfully improves someone's ability to work is used far more, and more of their real decisions and actions flow through it. This generates an order of magnitude more signal than the same work done across disparate tools, leading to compounding institutional knowledge that is captured and structured back into your ontology as its generated.

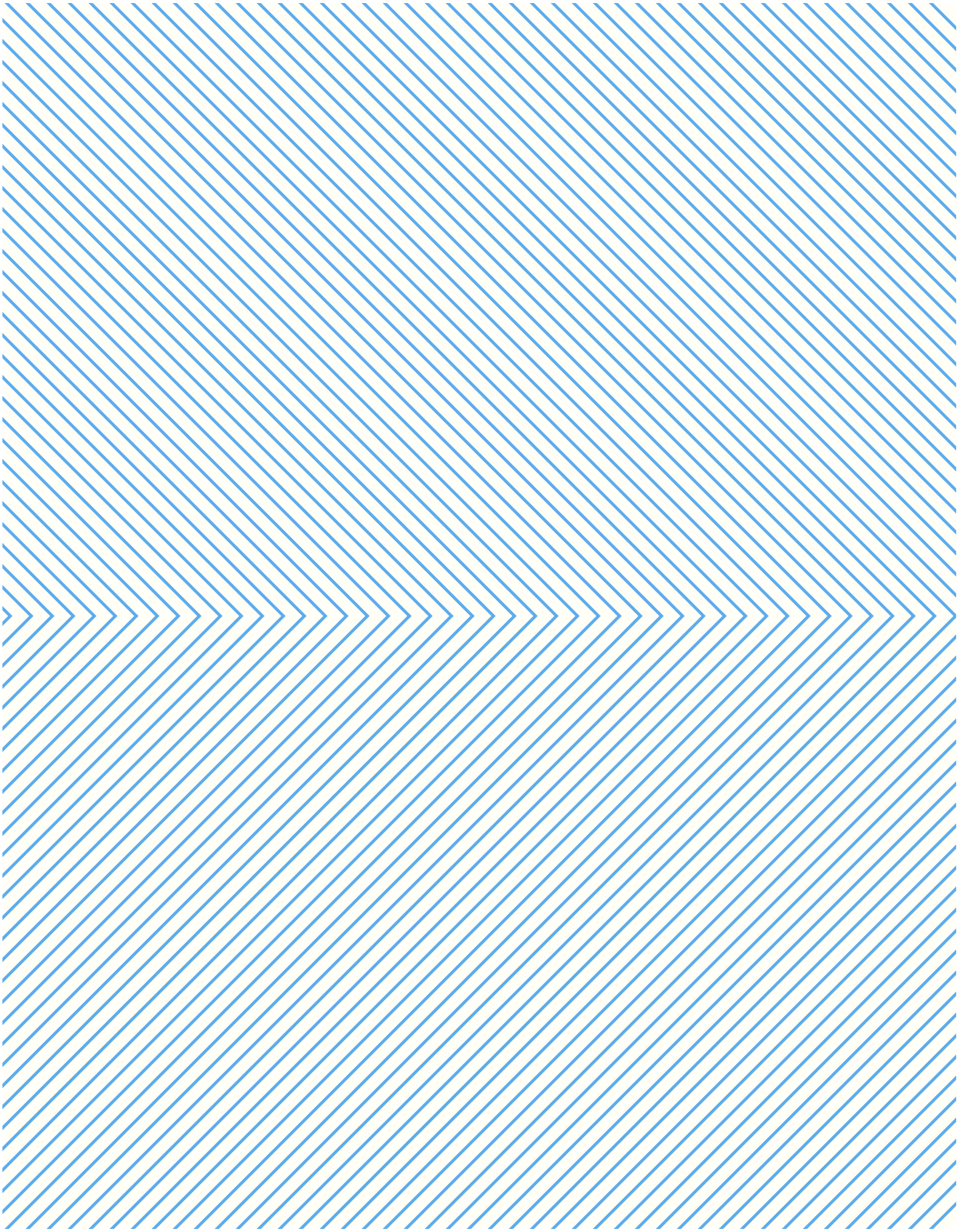
Owning the context engineering flywheel requires ontology primitives that sit outside the model itself. If your only assets are prompts combined with hidden model weights, there is nowhere for this signal to persist independent of the model you are using at that moment in time. The knowhow is trapped inside this single model relationship rather than owned by your institution. Instead the ontology knowledge layer, which compounds knowhow, must exist independently of the model intelligence layer that interacts with this knowhow to produce novel actions and insights. This model intelligence layer must be modular and model agnostic so that you may evolve freely as better models or better forms of intelligence become available to interact with your institutional knowledge.

Fig. 15 Context Flywheel










Palantir