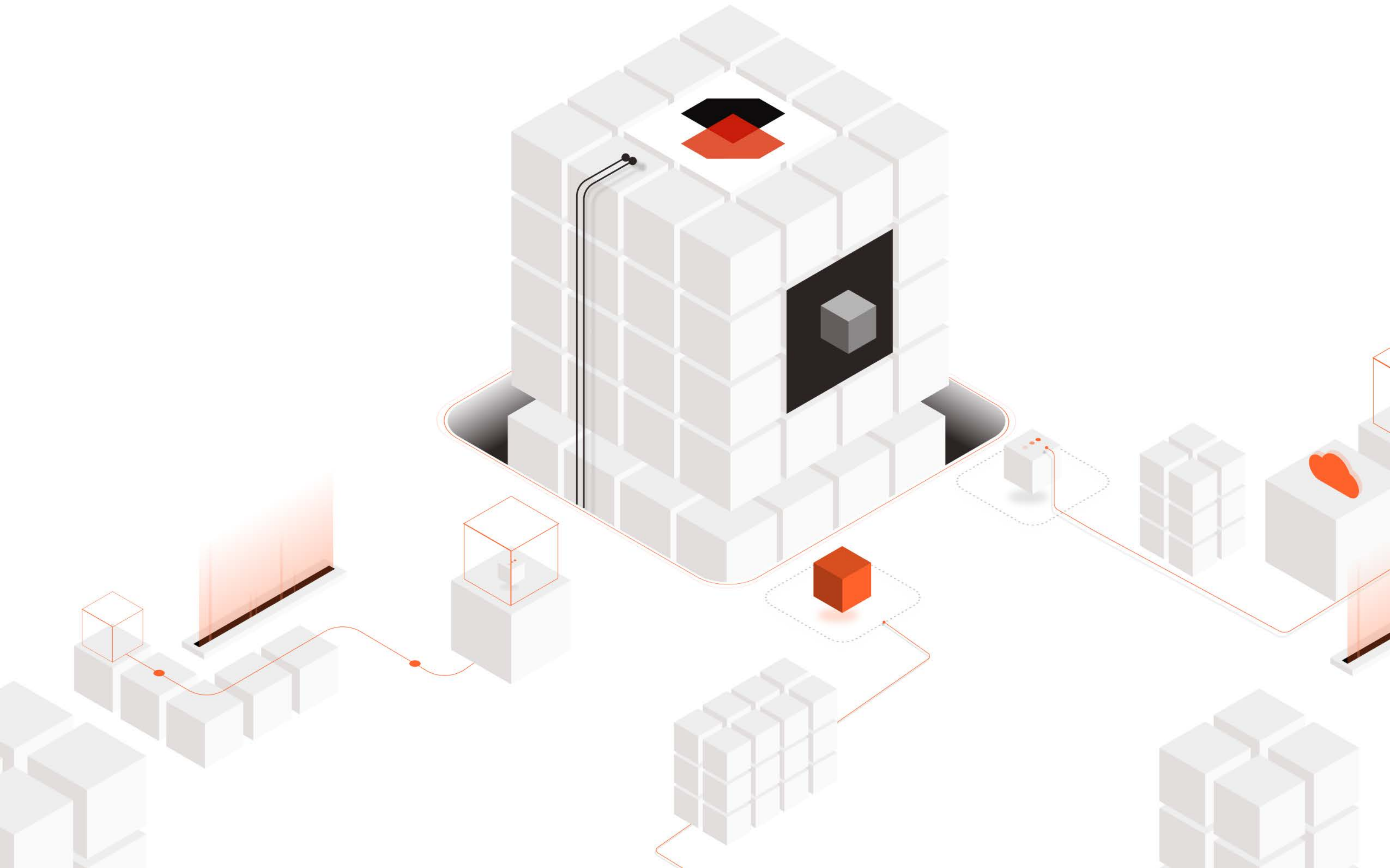
Predibase

Build vs. Buy: The Hidden Cost of DIY AI

Guide to choosing a GenAI stack for AI Developers & Leaders



The AI Engineer's Dilemma

In the rapidly evolving world of Generative AI, the decision to build a GenAI stack from scratch or buy a managed platform is pivotal. This ebook cuts through the hype to reveal the hidden complexities that lie beneath the surface of seemingly simple AI deployments. From data pipelines and infrastructure scaling to model optimization and cost management, these unseen challenges can overwhelm even the most capable engineering teams.

Drawing on real-world case studies and expert insights, we provide a strategic framework to help you:

- Objectively assess your organization's readiness for building a custom AI platform.
- Understand the true costs and trade-offs of the build vs. buy decision.
- Make informed choices that align with your unique business needs and long-term goals.

Discover how leading companies like Checkr and Convirza have navigated this critical decision, and learn how a managed platform can unlock the power of AI, drive innovation, and create a competitive edge without reinventing the wheel.

Are you ready to navigate the GenAI platform maze?

Table of Contents

Chapter 1: Understanding Your AI Needs 4

Chapter 2: The Building Blocks of GenAI Infra 12

Chapter 3: Pros/Cons of Building vs. Buying 33

Chapter 4: The TCO of Building AI Infra 36

Chapter 5: Decision Making Framework 39

Chapter 6: Real-World Build vs. Buy Case Study 47

Chapter 7: Best Practices & Next Steps 51

Chapter 1: Understanding Your AI Needs

Organizations face a pivotal choice when adopting generative AI: should they build their own infrastructure or rely on managed platform? The appeal of building in-house is understandable—greater control, custom capabilities, and the possibility of reducing costs by avoiding unnecessary features.

However, the reality of building robust infra is often more complex than it appears. Before choosing to "roll your own," you should take a hard look at your capabilities and goals.

Understanding Your AI Resources and Ambitions

The foundation for deciding whether to build or buy your AI infrastructure starts by evaluating your team's capabilities, your current AI needs, and your long-term goals.

Strategic Alignment: Focus on What Sets You Apart

Before investing engineering time into building GenAI infrastructure, ask a simple question: Is your competitive edge in infrastructure or in the models and products you build on top? If your value comes from proprietary data, custom trained models, or the applications you're developing, then rolling your own stack might be a distraction.

Building a custom platform makes sense if you're pioneering novel techniques in distributed training or ultra low latency model serving. If that’s not your game, you’re better off using a managed platform. It lets your team focus on:

- Designing and deploying better models
- Exploiting unique datasets
- Solving real business problems

Training a unique model does not require a unique platform.

Time Horizon: Today’s Needs vs. Tomorrow’s Growth

Custom-built solutions might feel like a win in the early days—they're tightly scoped, optimized, and feel "in control". But they rarely scale cleanly:

- Infrastructure debt compounds,
- New model architectures break early designs, and
- Engineering time shifts to maintenance, not innovation.

Managed platforms, by contrast, are designed to scale and typically support new model and latest techniques. Align your choice with your growth plans.

Assessing Capabilities: Do Your Teams Have What It Takes

When deciding whether to build or buy GenAI infrastructure, organizations must evaluate their technical maturity, cultural readiness, and production readiness.

Technical Maturity

Building a GenAI platform requires strong data infrastructure, specialized engineering, and ML experience. Honestly assess your capabilities:

- Can your data infrastructure handle large-scale GenAI workloads?
- Do you have the engineering resources for monitoring, troubleshooting, and 24/7 support?
- How experienced are you in deploying and scaling complex AI systems in production?

Realistic talent assessment is crucial. Lack of technical maturity makes building an in-house GenAI platform challenging, especially with multiple use cases.

Cultural Readiness

Cultural readiness encompasses several key factors:

- Adaptability to Change - the GenAI landscape evolves rapidly. It's important to ask yourself: How quickly does your team adopt new technologies and shift to agile approaches? Building your own stack takes a team that can pivot quickly.
- Risk Tolerance - what's your organization's appetite for risk? Building your own infra demands significant investment with no guaranteed success. Would a managed platform, with its predictable path and risk mitigation, be a better fit? And how important is predictable AI outcomes to your business.
- Feedback and Iteration - agility and rapid iteration are crucial, especially in fast-evolving AI. Organizations with strong feedback loops and iterative development are better equipped to build in-house.
- Collaboration - developing GenAI applications demands expertise from data scientists, ML engineers, domain experts and leadership. Are your technical and non-technical teams collaborating and tightly integrated?

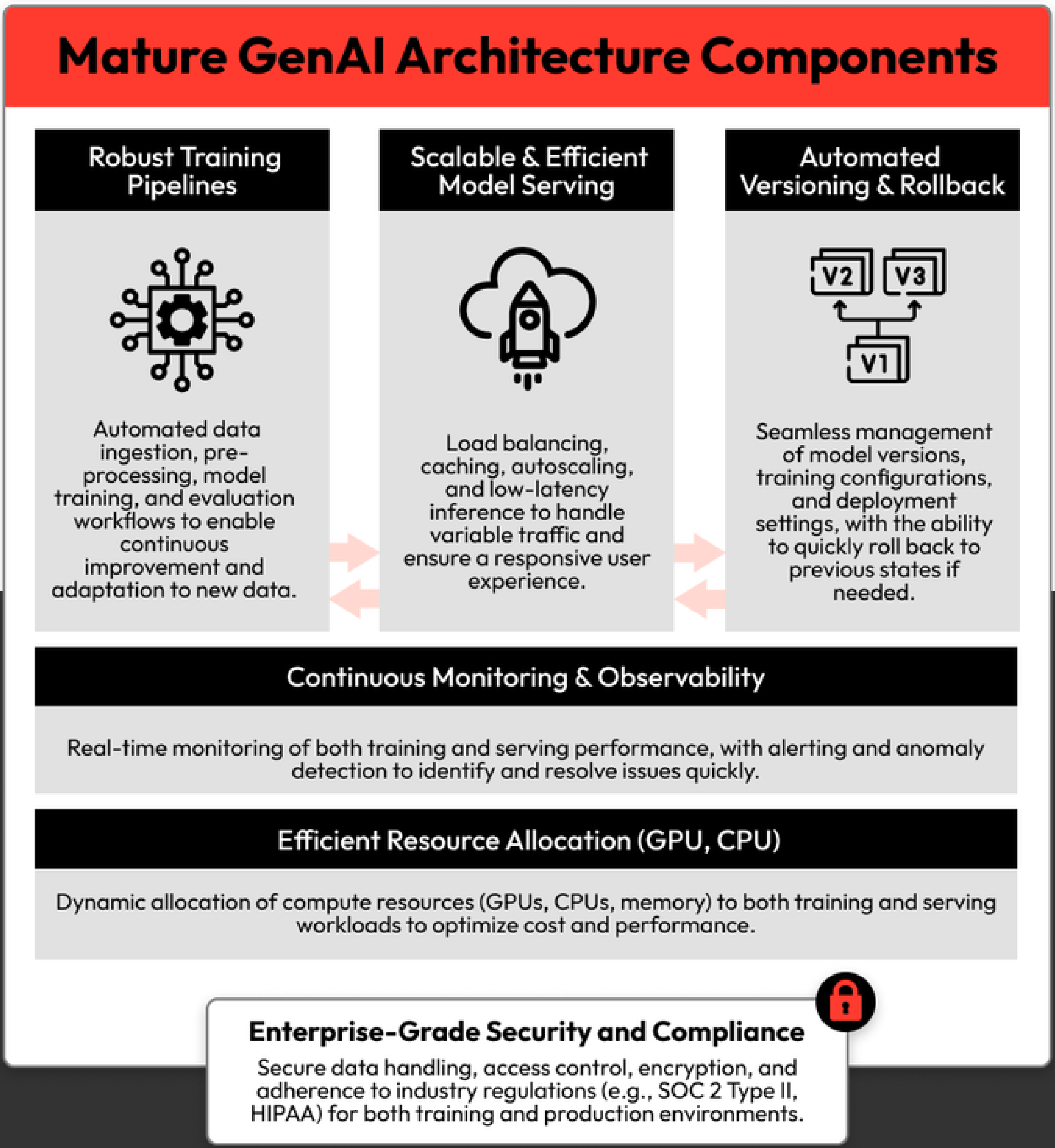
Production Readiness: Does Your AI Stack Match Your Ambitions

The number of production AI use cases your organization plans to support is a key indicator of readiness.

Current State	Future Needs	Maturity Level
Do you have a single, simple use case (e.g., a basic chatbot) or multiple, diverse GenAI applications (e.g., complex chatbots, content generation, personalized rec.)?	Will you primarily need to scale similar use cases (e.g., expanding the functionality of an existing chatbot) or develop a diverse portfolio of GenAI applications (e.g., adding image generation, code completion, etc).	How prepared are you to support multiple use cases simultaneously, with varying resource demands, performance requirements, and maintenance schedules

While a single small project might be manageable with a DIY approach or open-source tools, scaling to multiple mission-critical use cases significantly increases complexity.

As you scale AI at your organization, you will need a GenAI platform that can support a mature set of capabilities.


Mature GenAI Architecture Components
Robust Training Pipelines
Automated data ingestion, pre-processing, model training, and evaluation workflows to enable continuous improvement and adaptation to new data.
Scalable & Efficient Model Serving
Load balancing, caching, autoscaling, and low-latency inference to handle variable traffic and ensure a responsive user experience.
Automated Versioning & Rollback
V2
V3
V1
Seamless management of model versions, training configurations, and deployment settings, with the ability to quickly roll back to previous states if needed.
Continuous Monitoring & Observability
Real-time monitoring of both training and serving performance, with alerting and anomaly detection to identify and resolve issues quickly.
Efficient Resource Allocation (GPU, CPU)
Dynamic allocation of compute resources (GPUs, CPUs, memory) to both training and serving workloads to optimize cost and performance.
Enterprise-Grade Security and Compliance
Secure data handling, access control, encryption, and adherence to industry regulations (e.g., SOC 2 Type II, HIPAA) for both training and production environments.

Carefully assess your organization's ability to build and maintain such a platform in-house vs. leveraging a managed solution. Does your team have the expertise, resources, and long-term AI vision when making this decision.

Decision Factors

When evaluating whether to build or buy GenAI infrastructure, organizations must consider several critical factors:

Scalability Requirements

Anticipate current and future capacity needs for both training and serving. While training scalability handles complex models and larger datasets, serving scalability manages growing user demands.

Examples:

- **Convirza**: Expanding from 20 to 60 AI indicators required scaling both training infrastructure and serving capacity.
- **Checkr:** Currently processing millions of backgrounds checks and continuing to grow rapidly every month, Checkr needed a low latency system that can handle ever-growing list of classification tasks.

To prevent costly redesigns and ensure responsiveness, designing a system that seamlessly scales, both accommodating demand and managing resources efficiently, is essential, and helps in the long run.

Security & Compliance

Aligning with internal policies and regulations is paramount, especially for sensitive data. Building in-house provides control to tailor security and ensure compliance (e.g., HIPAA, SOC 2 Type II). Pre-built platforms offer robust security features and certifications, potentially reducing regulatory risks.

Customization vs. Standardization

While custom platforms offer control, avoid over-customizing for current needs, which limits future agility. Pre-built platforms provide a standardized, adaptable foundation for faster deployment and pivoting. Choose a solution that scales with you.

Cost Structures

Evaluate direct (software/hardware) and indirect (talent, training, downtime) costs. Building in-house has higher upfront investments. Buying solutions offer lower initial costs but recurring fees. Consider the total cost of ownership (TCO) in your analysis.

Chapter 2: The Building Blocks of GenAI Infra

The DIY Illusion

Spinning up a GenAI stack can look deceptively easy. Your CTO points to Unsloth for fine-tuning, vLLM for blazing-fast inference, and a buffet of other open-source packages. *Clone a repo, tweak a config, and we're in production by Friday… right?*

Not quite. Under that quick-start sheen lurks an iceberg of hidden effort: provisioning and right-sizing GPUs, building robust data pipelines, scheduling continual retraining, squeezing out latency with kernel-level hacks, and—most punishing of all—standing up 24 × 7 monitoring and incident response. Miss even one of those pieces and you'll spend more time firefighting than shipping features.

The most common "roll-your-own" traps

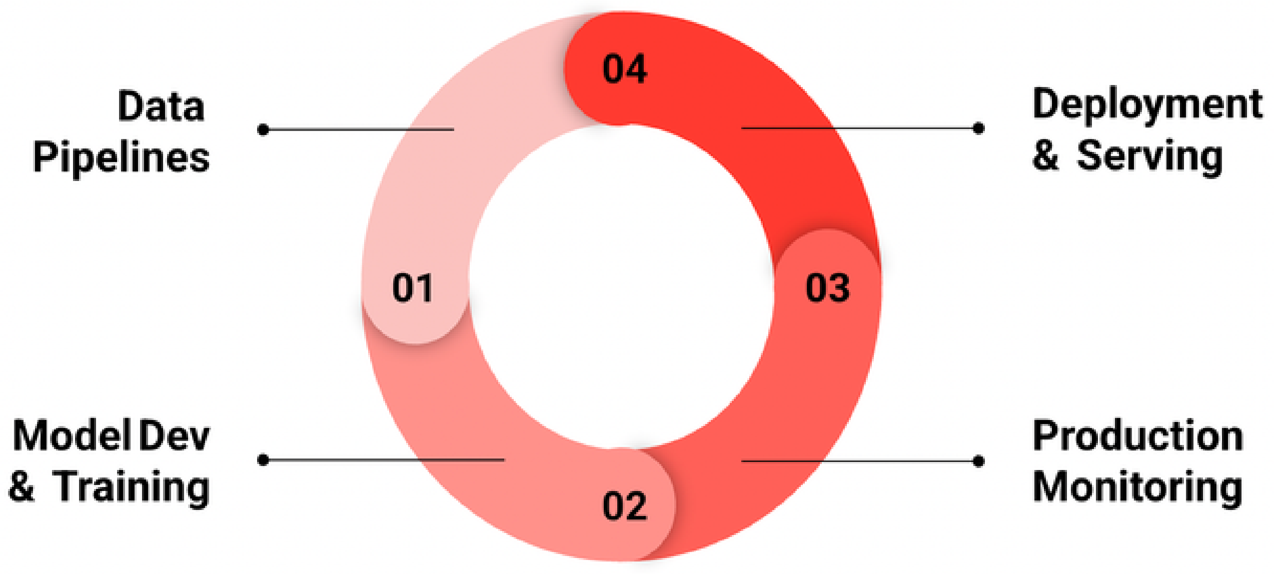
- **Operational drag** – Production performance hinges on ruthless model optimization (quantization, fine-tuning), distributed serving, and always-on monitoring/incident response—managing individual tools for each workflow is a massive burden.
- **Invisible total cost of ownership (TCO)** – OSS licensing may be free, but infra, talent, and maintenance bills snowball fast.
- **DIY lock-in** – Heavy customization around niche OSS corners makes future migration painfully slow.
- **Talent bottlenecks** – Kernel-level optimization and distributed serving expertise are scarce (and pricey to keep).

Before we plunge further, let's break down the *core building blocks* of a modern GenAI platform so you can weigh exactly what you'll need to own versus outsource

Core Components of a GenAI Platform

A robust Generative AI platform is like a finely tuned engine, with interconnected components working in harmon. Core components include:

1. **Data Pipelines.** Your models are only as good as their fuel. A tight pipeline ingests messy, multi-format data, cleans it on the fly, and streams high-quality batches to training jobs.
2. **Model Dev & Training.** Pick the right architecture, run disciplined experiments, and fine-tune fast. Track your versions and use clear metrics to zero-in on the best performer.
3. **Deployment & Serving.** Package once, ship anywhere. Containerize models with dependencies baked in, drop them onto autoscaling clusters, and keep latency predictable even when traffic spikes.
4. **Monitoring & Maintenance.** Production models never sleep. Track accuracy, detect drift, and trigger retraining pipelines the moment performance dips.



When these four pillars seamlessly lock together, you get a true end-to-end GenAI engine—one that ingests data, trains models, ships them to production, and keeps everything humming without constant hero work. The payoff? Engineers spend time inventing new user experiences, not wrestling with infra.

But, achieving this is anything but trivial. To unpack just how daunting that integration can be, we'll lean on an **iceberg analogy** in the next section.

The "Iceberg" Ahead

The iceberg chart reveals the deceptively small "visible" slice (quick demos and clean APIs) sitting atop a massive, unseen mass of production hurdles.

Surface-level Quick Wins

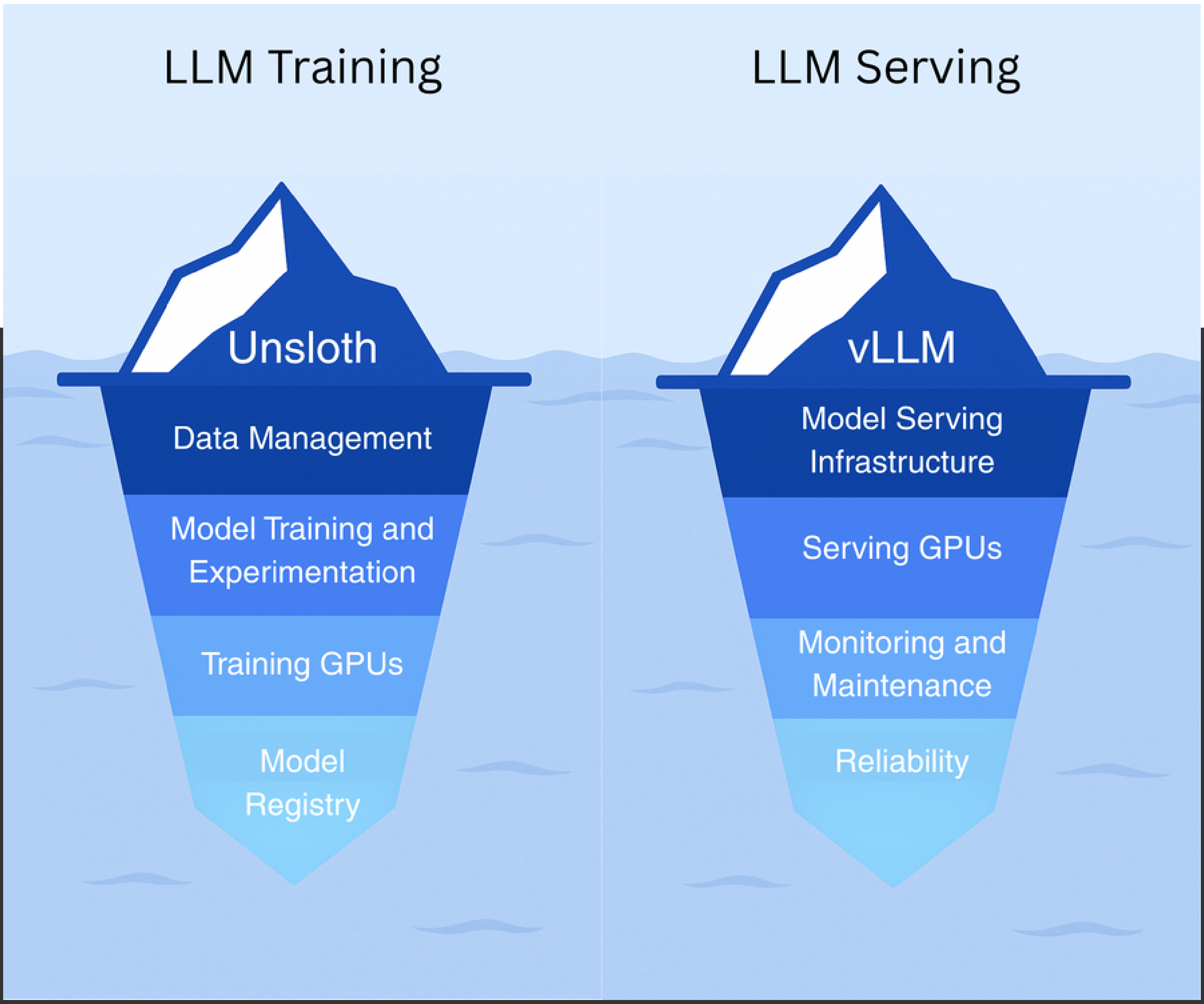
With open-source tools like Unsloth for fine-tuning and vLLM for inference, you can spin up a proof-of-concept in a weekend, but that demo is only the tip of the iceberg. Once real latency targets, traffic spikes, and cost ceilings enter the picture, the quick win reveals its limits: the hard part is transforming that POC into a production-grade, scalable, and cost-efficient system.

Below the Waterline Production Challenges

True productionization means 24/7 uptime, sub-second latency at peak load, cost guardrails, audit-grade security, and continuous model refreshes as data and architectures evolve. That demands three pillars: clean data pipelines, an automated training loop, and a rock-solid serving layer. Miss one, and your "quick win" becomes technical debt.

The iceberg rule applies: what you ship in week one is only 10 percent of what a production-grade GenAI system requires. The other 90 percent—ongoing data ingestion, model retraining, scaling, and ops—determines whether you delight users or drown in pager alerts.

Let’s dive below the waterline of our “iceberg” to examine the two core pillars of a production-grade GenAI stack: **model training** and **serving.**



LLM Training

Shipping a production-ready LLM is no weekend side project. In this section we'll dive beneath the training iceberg and expose the real lift: unifying messy enterprise data, orchestrating fault-tolerant distributed GPUs, continuously tuning models for sharper accuracy, and locking it all down with iron-clad version control and governance.

Data Management

As a cornerstone of GenAI systems, the application's quality hinges on its training data. While a prototype might rely on a HuggingFace dataset, you probably want to use your own enterprise data for the production version, presenting new complexities and challenges.

Core Capability	Description
Data Ingestion	There is a high chance your data will be fragmented across different platforms: some data might live in Snowflake, other in AWS S3, more in CRM, BigQuery, and various APIs. This scattered data poses integration challenges. Connectors from those different sources will need to be implemented.
Data Cleaning and Validation	You will also most likely encounter raw data complexities with materials in mixed formats (Word, PDF, HTML), corrupted files needing repair or inconsistent formats. The data will require cleaning through a process that includes repairing corrupted files, standardizing file formats, and validating automated labels through a dual system of manual review cycles to catch subtle errors. These steps ensured the raw data was accurate, consistent, and ready for further processing.
Data Transformation	After cleaning, you will need to perform transformations to prepare the data for training. This includes translating multilingual content into a unified language format, removing PII from chat logs, and using engineering scripts to convert processed data into prompt-completion training pairs required by LLMs. These transformations ensure the data is structured and aligned with the specific requirements of Generative Models.

Model Training and Experimentation

While your data team works on the data pipeline, your ML scientists can focus on model selection. They can use Unsloth or Hugging Face's AutoTrainer to rigorously evaluate and compare open-source foundational models, benchmarking performance metrics to identify the optimal architecture for their use case.

Core Capability	Description
Recovery Protocols	Fine-tuning large models is a time-intensive task, often taking days or weeks. This lengthy process is prone to errors from hardware failures, network instability, and memory spikes. To mitigate data loss, automatic checkpointing and recovery systems are crucial. Checkpoints save model states periodically, enabling resumption from the last saved state after interruptions like hardware failures or network issues. This seamless recovery saves significant time and resources by avoiding the need to restart training from scratch.
Batch Size Tuning	Larger batch sizes leverage the GPU's parallel processing capabilities, reducing training time by processing more data in parallel, as long as they fit within the GPU's memory constraints. However, increasing batch size raises the risk of Out of Memory (OOM) errors, especially with longer input sequences or larger models. Balancing batch size to avoid OOM while maximizing GPU utilization is critical. Implementing a dynamic batch size tuning mechanism, which adjusts batch sizes based on available GPU memory, ensures optimal utilization without causing OOM errors. By implementing such a system, Predibase achieved 35% reduced training durations.

Model Training and Experimentation continued

Core Capability	Description
Best Practices and Optimizations	As datasets (and with bigger models too) grow, small optimizations make a big difference. In this blog, Predibase shares how they implemented such optimizations to significantly accelerate training: Sample Packing boosts throughput by 2-5x by eliminating padding inefficiencies. FlashAttention-2 optimizes memory and compute, achieving 72% MFU efficiency. Optimizer streamlining reduces computational overhead, while selective gradient checkpointing removal balances memory and speed. These techniques require expertise to configure effectively, as they involve trade-offs between memory, computation, and model performance.
Continuous Model Training	Periodic retraining introduces complexities, increasing the risk of errors. It may also lead to unnecessary resource consumption if updates occur without genuine need. Conversely, it can miss sudden data changes or emerging trends, potentially losing valuable insights. Continuous model training is ideal for consistent accuracy, triggered automatically by new data, performance degradation, or data drift. This is particularly beneficial in dynamic environments like recommendation systems, fraud detection, real-time personalization, and customer support chatbots, ensuring timely adaptation to evolving patterns.

Model Training GPUs

While a single GPU may suffice for fine-tuning lightweight open-source models with smaller Hugging Face datasets, fine-tuning larger foundational models or handling extensive datasets inevitably requires distributed training to overcome memory and computational limitations.

Core Capability	Description
Distributed Training	Techniques like DeepSpeed ZeRO and Fully Sharded Data Parallel (FSDP) are crucial for efficiently managing memory by sharding model parameters, gradients, and optimizer states across multiple GPUs, enabling the training of models that exceed single-GPU VRAM limits. However, these methods require careful configuration and expertise to optimize performance, as finding the appropriate settings for model sharding can be complex.
GPU Availability and Intelligent Provisioning	Distributed training requires multiple GPUs. However, high-performing GPUs are in high demand and often have limited availability from cloud providers, leading to potential shortages. The alternative offered by cloud providers to guarantee availability is 'always-on' instances, but this can be costly during idle periods. Implementing an intelligent provisioning system that dynamically selects optimal GPU configurations across multiple providers offers a solution, as Predibase details in this blog post. This approach not only overcomes availability limitations but also reduces dependency on a single cloud provider, mitigating the risk of outages. Such a strategy can significantly improve resource utilization and cost-efficiency in distributed training scenarios.

Model Training GPUs Continued

Core Capability	Description
GPU Optimization	GPU power is finite and expensive, yet training larger models on more data can significantly enhance your GenAI app. To fit within budget constraints, various optimizations are crucial. In this blog post, Predibase details several of these optimizations and their impressive results: • Loading pre-quantized weights (4x faster loading) • Sample packing (up to 5x faster training) • Implementing Flash Attention (significant speedup in attention computation) • Optimizing CUDA kernels (up to 20% training speedup) These techniques, among others, allow for training on more data within existing GPU limitations, balancing performance gains with infrastructure costs. Implementing these optimizations requires expertise but can yield substantial efficiency improvements.

Model Registry: AI Version Control

Finding the optimal base model, number of epochs, learning rate, batch size, optimizer, and data augmentation strategies are essential for successful training. A model registry provides a key role by:

- Tracking model variants and performance.
- Centralizing training configurations and hyperparameters.
- Insuring reproducible pipelines across environments.

The model registry brings order, transparency, and repeatability to the entire model lifecycle.

LLM Serving

Turning your LLM into a live, customer-facing service is anything but a “click-to-deploy.” In this section we’ll rip away the glossy endpoint wrapper and take a look below the waterline to see what's truly required to build a scalable and resilient production-grade serving stack.

Model Serving Infrastructure

Finally, your ML/MLOps engineers can focus on the serving aspect of your LLM. They might choose to leverage tools like vLLM for efficient serving, but this process also requires a robust and complete infrastructure stack.

Core Capability	Description
Cluster Management	GenAI applications require a robust container orchestration system like Kubernetes to handle GPU clusters effectively. Kubernetes enables GPU scheduling through device plugins (e.g., NVIDIA or AMD), ensuring efficient resource allocation. Features like autoscaling dynamically adjust GPU nodes based on workload demands, while node labeling and affinity rules ensure compatibility with specific GPU types. Additionally, high-speed networking solutions, such as InfiniBand or RDMA, optimize communication between GPUs. And tools like Run:AI or NVIDIA Triton further enhance workload scheduling and resource utilization, minimizing idle times and maximizing efficiency in GPU cluster operations.
Network Optimization	Serving your model also requires network optimization to maximize data transfer between GPUs and minimize latency. At the physical level, it involves configuring high-speed interconnects like InfiniBand and implementing GPUDirect RDMA for direct GPU-to-GPU communication. You may need to fine-tune the NVIDIA Collective Communications Library (NCCL) parameters for efficient multi-GPU and multi-node communication. After implementing GPU-specific load balancing strategies and ensuring proper GPU resource allocation across nodes, make sure to build continuous monitoring of GPU metrics, such as memory bandwidth and utilization for real-time performance tracking.

Serving GPUs

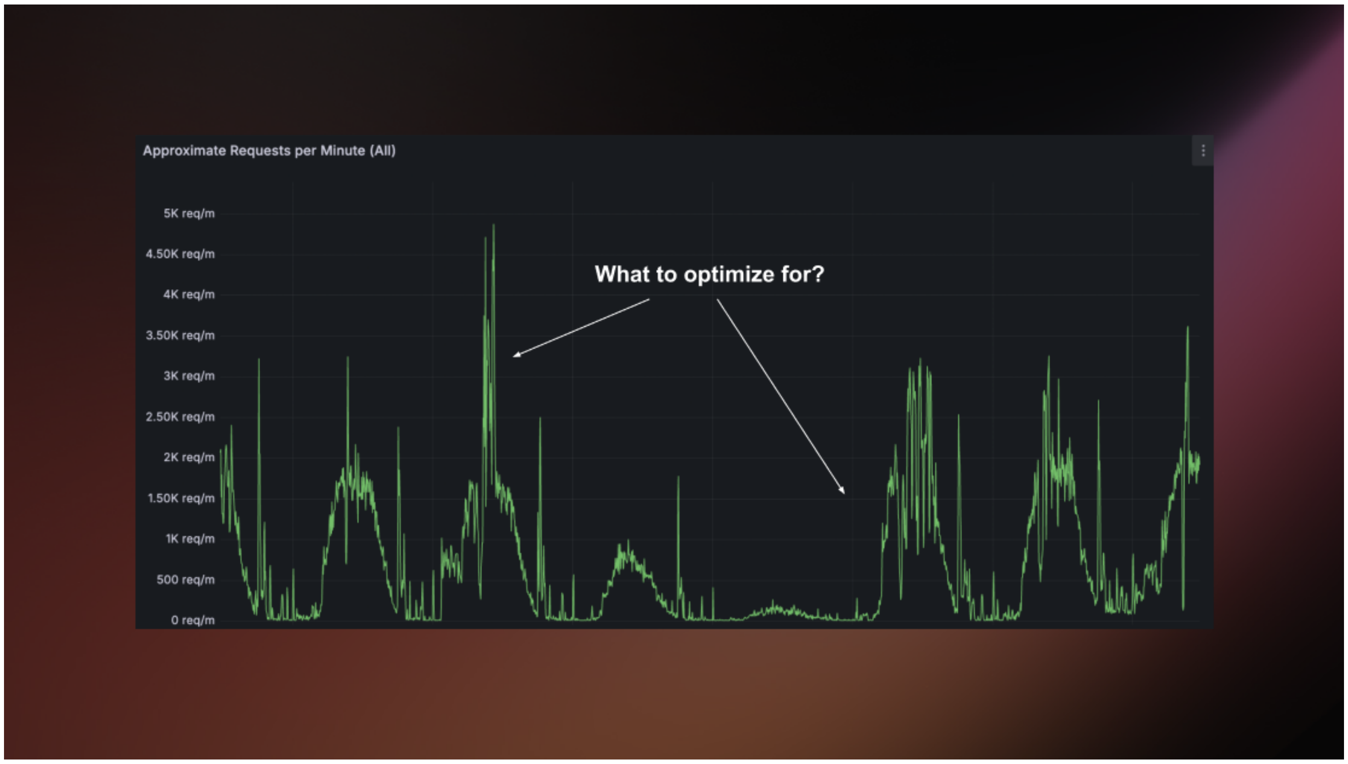
Serving GPUs for GenAI applications faces similar challenges to training, including GPU scarcity and costly always-on alternatives. While solutions from training such as intelligent provisioning and GPU optimization can be leveraged, serving also introduces unique demands:

- Low latency becomes critical for real-time user interactions.
- Handling variable traffic requires dynamic scaling.
- Cost optimization needs balancing with performance.
- High availability is essential for user-facing applications.

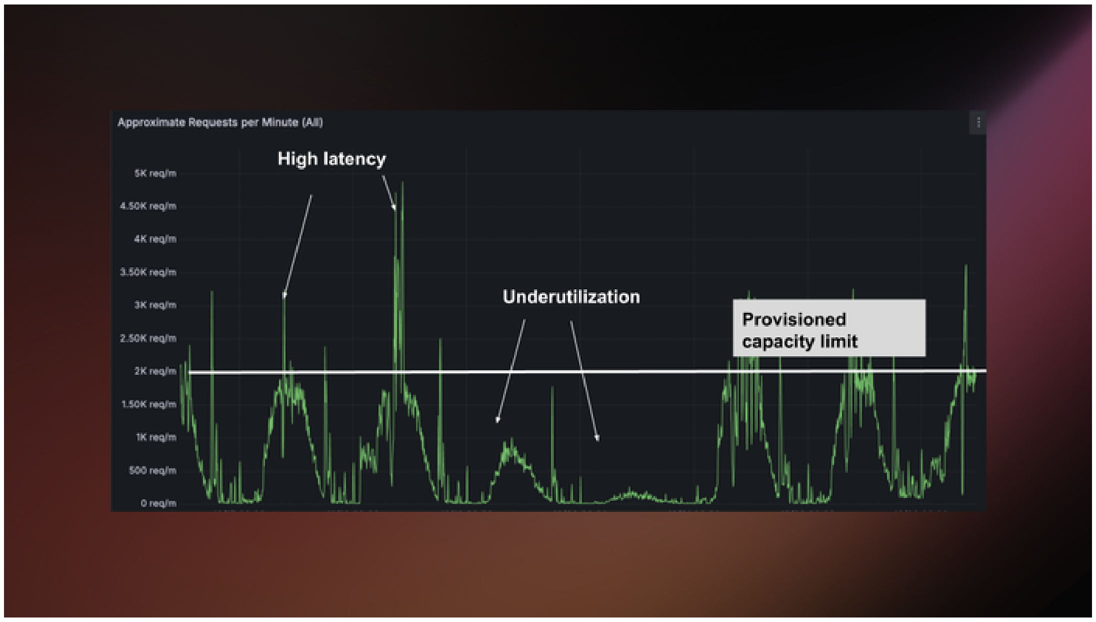
Core Capability	Description
Auto-scaling serving GPUs & Cost Management	GenAI traffic is highly unpredictable and variable. Persistent always-on deployment models create prohibitive costs, while under-provisioning GPUs lead to Denial Of Service (DOS) during peak traffic. Beyond intelligent provisioning, a strategic shift to hybrid deployments combining reserved instances with spot/preemptible nodes is crucial for optimizing costs and performance. This approach, which we detailed in **this blog,** balances cost optimization with performance, ensuring efficient resource utilization without compromising service quality. It requires intricate orchestration and advanced caching strategies to mitigate cold start latency from preempted instances. By leveraging training's lower priority aspect for LLM replicas, organizations can achieve significant cost savings while maintaining responsiveness to variable traffic demands.

Customer-facing applications typically experience fluctuating traffic patterns, with sharp spikes and dips throughout the day and week. This variability makes it difficult to consistently optimize GPU utilization.

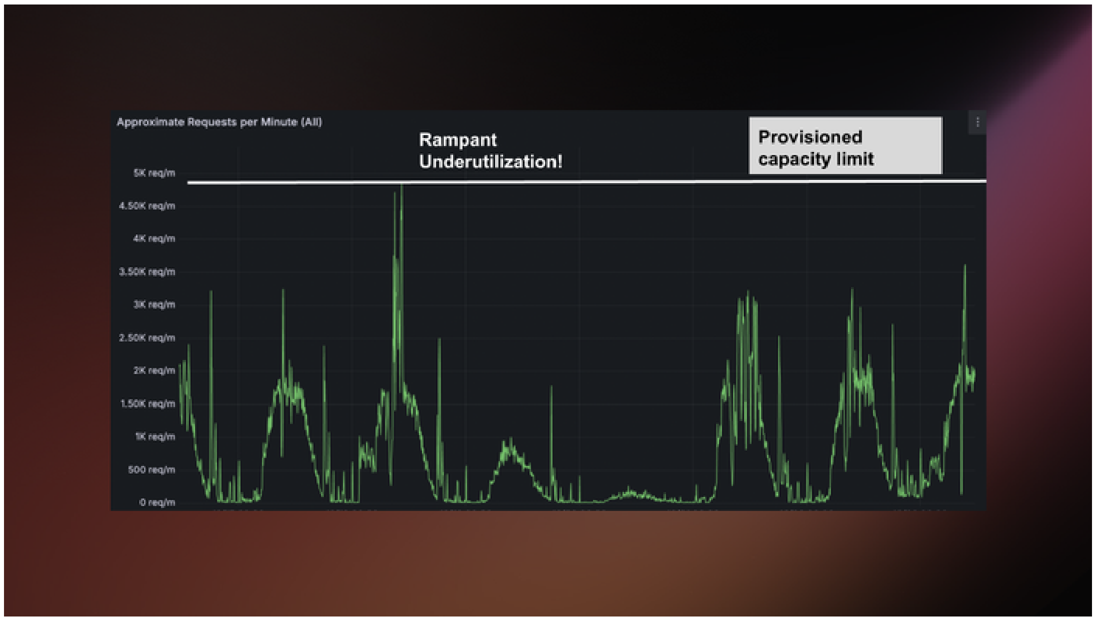
Engineering teams building their own infrastructure need to decide which traffic patterns to prioritize and what trade-offs they're willing to make.



Teams focused on cost-efficiency may provision GPUs based on average utilization. While this approach lowers expenses, it can lead to latency spikes during peak traffic, potentially degrading the customer experience.

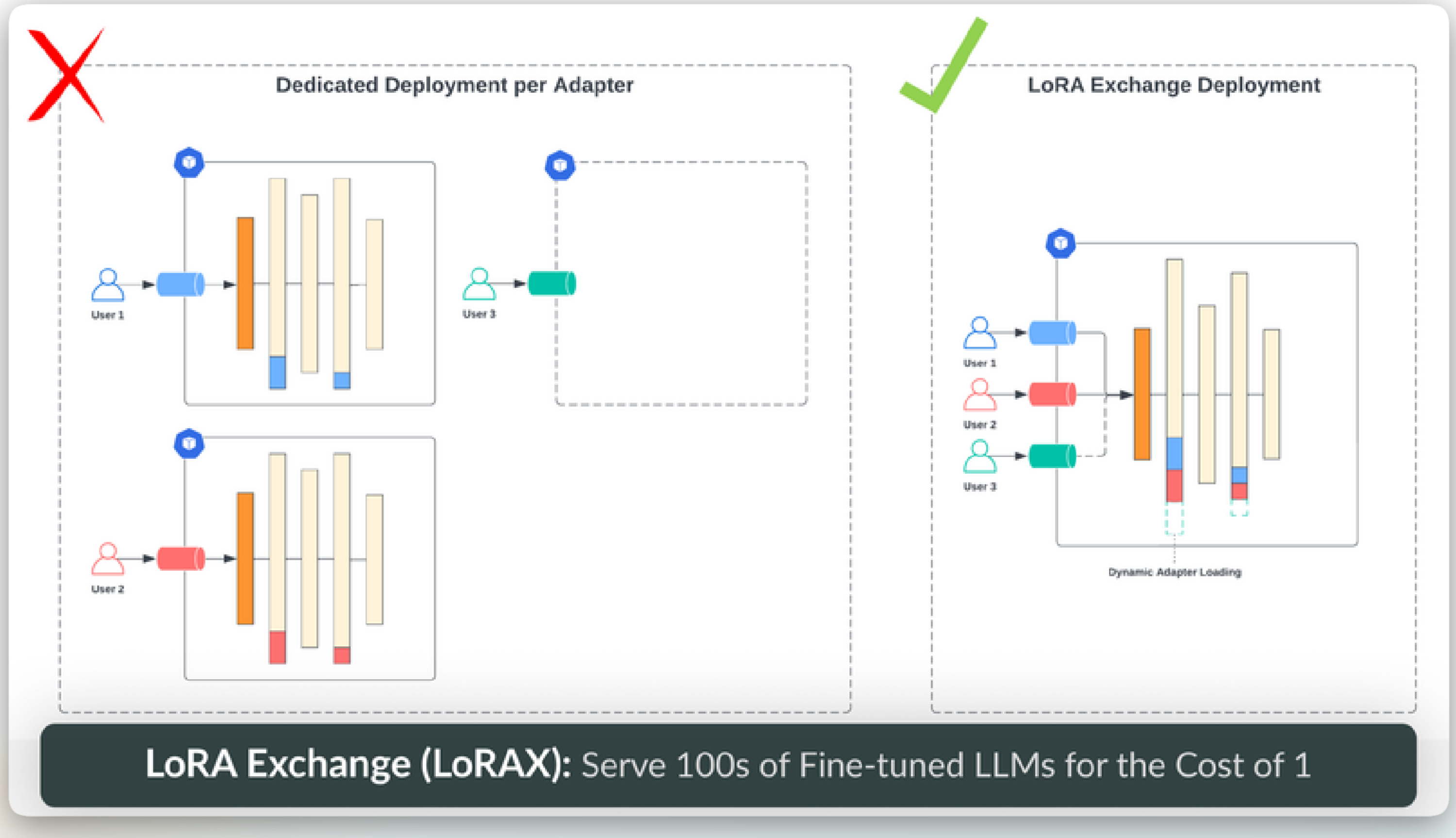


Conversely, more risk-averse teams may provision for peak capacity. This guarantees performance during high-demand periods but often leaves GPUs underutilized during off-peak hours. While it ensures availability, it also introduces significant inefficiencies and hidden costs many businesses overlook when scaling AI systems.



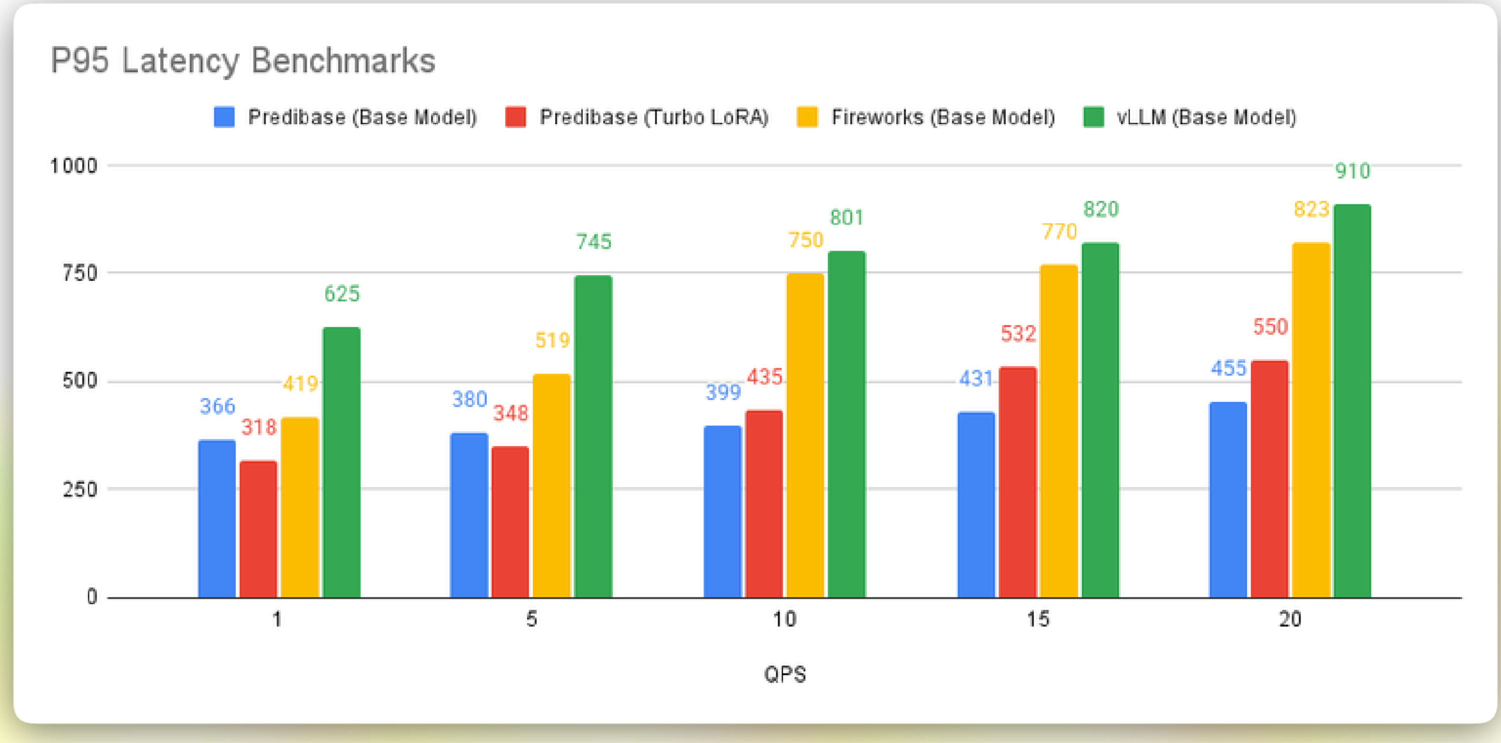
Serving GPUs continued

Core Capability	Description
Performance Maximization	Foundation models can be massive, requiring substantial compute. However, smaller fine-tuned models can outperform larger models for specific tasks. To maximize performance, tools like open-source LoRAX enable multi-LLM serving. With LoRAX, you can serve over 100 fine-tuned models on a single GPU with no degradation in throughput. This significanlty improves GPU efficiency.


Dedicated Deployment per Adapter
User 1
User 3
User 2
LoRA Exchange Deployment
User 1
User 2
User 3
Dynamic Adapter Loading
LoRA Exchange (LoRAX): Serve 100s of Fine-tuned LLMs for the Cost of 1

Serving GPUs continued

Core Capability	Description
Performance Maximization (continued)	Quantization is another powerful lever for performance optimization. It can reduce memory footprint by 50%, increasing throughput and potentially lowering costs by over 30%. However, if implemented incorrectly, it may result in quality loss, as experienced by some companies. Speculative decoding significantly boosts inference speeds. Predibase reports achieving 4x speed improvements using FP8 and speculative decoding.



Inference benchmarks showing the improvements from Predibase's out of the box optimizations

Monitoring and Maintenance

Customer-facing applications demand 24/7 monitoring and maintenance, requiring dedicated resources to ensure reliability and performance.

Core Capability	Description
Logging and Metrics	Effective monitoring begins with logging key metrics and creating dashboards to track system health. Tools like Datadog can be integrated with serving infra to provide real-time insights on performance.
System Alerts	Setting up alerts is the next step. Defining appropriate thresholds often involves game days to simulate traffic events and identify potential issues. These simulations require careful planning, resource allocation, and scheduling during low-impact periods. Once thresholds are established, tools like PagerDuty can be configured to handle incident notifications, ensuring prompt responses to critical issue
Routine Monitoring	On-call rotation is need to resolve issues in real-time. This can place a significant burden on your team. Depending on the complexity and maturity of your application. Automating routine tasks or outsourcing certain aspects of monitoring can alleviate this strain.
Infrastructure Upgrades (Blue/Green Deployments)	Regular infra and software upgrades are crucial for maintaining the performance and scalability of your GenAI platform. This includes updating open-source tools like Kubernetes and the broader infrastructure stack to address vulnerabilities, improve efficiency, and support new features. Proactive planning and thorough testing in staging environments is essential before deploying to prod.

Reliability and Resiliency

Designing for failure is the key to ensuring uptime. Through redundancy, geo-distribution, traffic mirroring, and disaster recovery plans, GenAI infrastructure can remain resilient—even when things go wrong.

Core Capability	Description
Redundancy Plans	To ensure high availability and resilience in GenAI app infrastructure, redundancy is key. Strategies include: • **Multi-Cloud and Hybrid Architectures:** Distribute workloads across multiple cloud providers or combine on-premises and cloud resources to avoid single points of failure. • **Geo-Distributed Deployments:** Deploy infrastructure across regions for continued service during local outages. • **Active-Active Kubernetes Clusters:** Operate multiple clusters simultaneously across clouds, ensuring seamless failover and load balancing. • **Geo-Sharded Model Caches:** Store frequently accessed models closer to users, reducing latency and synchronizing updates across regions. • **Traffic Mirroring:** Duplicate live traffic to validate failover mechanisms without disrupting production. • **Load Balancing and Auto-Scaling:** Dynamically distribute traffic and scale resources to handle demand spikes. • **Backup and Disaster Recovery:** Regularly back up data and establish recovery plans for quick restoration. By combining these strategies, you can build a robust, scalable, and fault-tolerant GenAI infrastructure capable of maintaining uninterrupted service.

Reliability and Resiliency continued

Core Capability	Description
Contingency Planning	Hope for the best, but prepare for the worst. Errors and unexpected events are inevitable, so reliability depends on anticipating and quickly recovering from them. This requires organizing game days to simulate failures, creating a detailed incident response playbook, and maintaining real-time dashboards to monitor key metrics. These tools enable on-call teams to swiftly diagnose and resolve issues, minimizing downtime and ensuring seamless service continuity.

Chapter 3: Pros/Cons of Building vs. Buying

When deciding whether to build a custom GenAI platform or adopt a managed solution, organizations must weigh control and customization against cost, speed, and operational overhead. Below is a high-level comparison:

Eval Criteria	Build It Yourself	Buy Managed Platform
Speed to Deploy	🔴 Slow – months to build infra and deploy	🟢 Fast – deploy models in hours or less
Customization & Control	🟢 Full Control – tailor everything to your needs	🟡 Limited Customization - vendor-defined architecture
Upfront Investment	🔴 High – requires ML + infra + MLOps talent	🟢 Low – vendor manages infrastructure
Talent Requirements	🔴 High – requires specialized, expensive, hard-to-find talent	🟢 Minimal – built-in vendor expertise and professional svcs
Scalability & Reliability	🟡 Must build and maintain yourself; complex to get right	🟢 Auto-scaling, SLAs, fault tolerance built-in
Maintenance Burden	🔴 Heavy – continuous updates and fixes	🟢 Offloaded to vendor
Cost Predictability	🟡 Medium – unforseen costs of scaling	🟢 High – predictable pricing models
Security & Compliance	🟡 Must build and certify yourself	🟢 SOC 2, data privacy, enterprise controls
Long-Term Strategic Value	🟢 High – own your stack and IP	🟡 Medium – roadmap driven by vendor

Pros / Cons continue

Eval Criteria	Build It Yourself	Buy Managed Platform
Adaptability Over Time	🟡 Rigid – rework needed for new needs	🟢 Flexible – supports evolving AI use cases
Opportunity Cost	🔴 High – infra takes focus from product innovation	🟢 Low – focus stays on core AI application, not infra mgmt
Vendor Dependency	🟢 None – you're in control of your future	🔴 High – vendor sets priorities and roadmap

Choosing whether to build or buy an AI development platform isn't just a matter of technical capability—it's a strategic decision with long-term implications.

Building offers deep customization, full control, and the potential for proprietary advantages—but it demands significant investment, rare talent, and ongoing maintenance that can divert focus from core innovation

In contrast, **buying** from a managed provider accelerates deployment, simplifies maintenance, and ensures access to expert support and cutting-edge capabilities. It reduces the engineering burden, enables predictable scaling, and allows teams to focus on delivering business value rather than maintaining infrastructure. However, buying may come with tradeoffs in roadmap control and customization depth.

Ultimately, your choice depends on your organization's maturity, goals, and priorities. Another key factor to consider is the T**otal Cost of Ownership (TCO)**. In the next section we'll provide our TCO analysis for the two approaches.

Chapter 4: The TCO of Building AI Infra

Understanding total cost of ownership (TCO) is critical for making informed, long-term decisions about your GenAI strategy. The table below compares the estimated annual costs of building an in-house GenAI platform versus using a managed solution. While actual expenses will vary depending on your specific needs, this comparison highlights the major cost drivers and provides a useful framework for evaluating TCO and return on investment.

Cost Item	In-House	Managed Platform
1. Eng. Salaries (ML Eng., Data Eng.)	3-4 FTEs @ ~$300K* (~$900K – $1.2M)	Included in vendor fee
2. Hiring & Training Specialized Talent	Recruiting & Onboarding (~$20K – $50K)	Minimal
3. Infra. Setup (Compute, Storage)	Servers, GPUs, Storage (Add. setup fees)	Variable, often bundled or discounted
4. Maintenance (Upgrades/Security)	Ongoing Dev time (~$50K – $100K)	Included in subscription
5. Outages & Downtime	Internal On-Call & Productivity Loss	Vendor SLA / guaranteed uptime
6. Opportunity Cost	Eng. Time on Infra. vs model improvements	Minimal impact, focus remains on core ML
Total Annual Cost**	~$1M – $1.5M***	$50K – $200K***

* Salary data from www.levels.fyi

** Actual costs vary based on project scope, team experience, and infra scale.

*** Illustrative estimates for typical mid-size ML deployments.

How to Read This Chart

1. **Engineering Salaries**: In-house development often requires multiple engineers to manage data pipelines, DevOps, and infrastructure. A subscription model spreads these costs across the vendor's customer base.
2. **Hiring & Training**: Recruiting specialized ML infrastructure talent can be expensive and time-consuming. A vendor subscription shifts that responsibility to the managed provider.
3. **Infrastructure Setup**: Building a GenAI platform in-house may require substantial upfront investments in hardware, GPU clusters, and networking. Subscriptions often bundle or discount the infrastructure layer.
4. **Maintenance & Upgrades**: Patches, security updates, and regular feature enhancements are standard in vendor offerings, whereas in-house teams must allocate ongoing resources.
5. **Unplanned Outages & Downtime**: Managed solutions typically include guaranteed SLAs. In a DIY approach, downtime costs can escalate quickly if issues occur after hours or require specialized troubleshooting.
6. **Opportunity Cost**: Time spent on building and maintaining a platform is time not spent creating unique ML solutions or adding new features to your product—where your real competitive edge likely resides.
7. **Total Annual Cost**: These illustrative figures highlight how in-house totals can easily surpass the price of a subscription, especially when factoring in the long-term growth and complexity of ML projects.

Chapter 5: Decision Making Framework

Build vs Buy Checklist

Use this checklist to assess your organization's readiness for building a GenAI platform. Follow the steps to determine which approach is best for you.

Step 1: Initial Assessment

Number of Production Use Cases: Do you have 2 or more mission-critical ML projects that require regular updates and maintenance? (Yes / No)

- If you answered "No", a DIY approach may suffice if you're only experimenting, but reconsider if you plan to scale soon.
- If you answered "Yes", go to Step 2 to determerne the right solution for you.

Step 2: Detailed Evaluation

For questions in the table below, rate your organization using the following scoring rubric:

1 = Not at all ready

2 = Somewhat ready

3 = Moderately ready

4 = Mostly ready

5 = Fully ready

Assess each criterion honestly based on your current capabilities and infra:

Criteria	Score
1. Team Size & Specialized Expertise: Can your team handle building and maintaining a custom GenAI platform (monitoring, troubleshooting, etc.)?	
2. Strategic Alignment: Is leadership committed and resourced to develop and evolve an in-house GenAI platform long-term?	
3. Technical Maturity & Tooling: Is your DevOps/MLOps environment robust enough to support a custom-built platform?	
4. Compliance & Security: Do you operate under strict regulations (HIPAA, PCI, etc.) or have advanced security needs?	
5. Speed to Market: Do your ML models require rapid iteration and frequent updates to stay competitive?	
6. Budget & Total Cost of Ownership (TCO): Have you realistically compared the cost of in-house staffing to a subscription fee?	
7. Unique IP vs. Commodity Infrastructure: Are you creating novel platform IP, or do you simply need reliable, standard ML workflows?	
8. Future-Proofing & Scalability: Can you keep pace with evolving techniques for improving your ML model and application?	
9. Risk Tolerance & Business Continuity: How critical are high-quality proprietary models to your core operations?	
10. 24/7 On-Call & Support: Are you prepared to staff after-hours monitoring and handle downtime for critical ML systems?	

Step 3: Calculate Your Score

Add the scores from questions 1-10. The maximum possible score is 50.

Step 4: Interpret Your Score

- **35-50 Points**: Strong Build Readiness: Your organization is well-equipped to build and maintain a GenAI platform in-house, especially if you answered YES to Question #1.
- **20-34 Points**: Hybrid Approach Recommended: Your organization may benefit from a hybrid approach or a managed platform to supplement your existing capabilities.
- **Below 20 Points**: Managed Solution Recommended: A managed solution is likely more cost-effective and efficient. Your organization may not be ready for the complexities of in-house development. Reassess regularly as your capabilities evolve.

Important Considerations:

- This framework is a guide, not a definitive answer. Consider your specific circumstances and priorities.
- Regularly revisit this checklist as your company matures or scales. A low score today doesn't rule out future in-house capabilities.

Risk Management & Flexibility

Vendor stability

Committing to a vendor always involves risk. Assess these factors:

- **Financial Stability**: Can the vendor provide long-term service?
- **Market Position**: What's their industry track record and reputation?
- **Long-Term Viability**: What is their potential for growth and sustainability?
- **Risk Mitigation**: What contingency plans exist in case of vendor issues?
- **Scalability**: Can the vendor grow with your organization's needs?

Thorough assessment ensures vendors align with your AI objectives.

Fallback plans

Ensure your vendor has strong disaster recovery and business continuity plans. Also, prepare your own failover strategies by:

- Ensuring data portability and export capabilities
- Planning ability to migrate to alternative solutions
- Creating contingency plans for service disruptions

Regularly evaluate vendor performance and maintain a switch-over contingency plan.

Future Proofing Infra

The table below compares the challenges of adapting to emerging AI techniques, GPU architectures, and shifting business priorities when building in-house versus leveraging a managed platform.

Feature	Building In-House	Leveraging Managed Platform
Adapting to New Techniques	Requires significant in-house expertise in ML, systems engineering, and distributed computing. High risk of incompatibility and long testing cycles.	Vendor handles monitoring, evaluation, and integration of new techniques. Offers faster adoption and reduced risk, but less control over implementation.
Adapting to New GPU Architectures	Significant upfront costs for acquiring and optimizing for new GPUs. Requires deep understanding of GPU architectures and parallel computing.	Vendor handles evaluation and integration of new GPU architectures, providing optimized performance with reduced complexity. Access may be impacted by timing and is dependent on platform support.
Adapting to Business Pivots	Requires significant rework and adaptation of existing infrastructure, leading to inflexibility, long development cycles, and technical debt.	Offers greater adaptability, enabling businesses to quickly adjust their AI strategy without major overhauls. May be limited by a lack of control and feature gaps.

Key Principles for Future-Proofing GenAI Applications

In the rapidly evolving world of GenAI, future-proofing your application is essential for long-term success. Whether you choose to build in-house or leverage a managed platform, these core principles will guide your decision-making:

1. Modularity and Abstraction

A central tenet of future-proofing any GenAI application, whether built in-house or leveraged through a managed platform, lies in modularity and abstraction. This means creating a framework that lets you swap out components without affecting the entire system. If building in-house, focus on establishing clear APIs for each component. If buying a platform, be sure to assess how simple it is to integrate new components. One must consider how easily one can upgrade their platforms.

2. Open Standards and Interoperability

Adhering to open standards and prioritizing technologies that promote interoperability is a very important step when future-proofing a GenAI application. Doing so helps with integration with other systems and helps with any solution changes over time. If you are building, be sure to use open-source tools and technologies that match the industry. For any business looking to buy, you will want to take precautions and go with platforms that give access to industry standards. Can you integrate your platform, and what measures were set to make it happen?

3. Automation and Scalability

Automation and scalability are critical pillars for future-proofing AI. This principle states the need for automation and scaling to continue improving at a steady rate. With AI becoming ever more common, it's very likely your team's usage will scale. When building, you will want to prioritize automatic infrastructure and be ready to handle anything that might come your way.

For platforms that are bought, it's great to go with one that has full development, as well as auto management. It is important to know whether your current platform will support your models if they are scaled.

4. Data Governance and Security

Having a set of data governance and security measures is key to protecting data while following compliance. These measures ensure that no one's personal data is compromised. Building the controls from the start will prove to be beneficial. Platforms can be used to guarantee that the system is stable. Make sure the new platform is going to be able to help with security.

5. Cost Optimization

Continuous cost monitoring and optimizing the amount you are spending should be done to create better output and prevent overspending. To optimize your cost, be sure to first implement monitoring tools. However, it is important that your platform is optimized to ensure that computing power isn't lacking. Can it beat the proprietary model?

Chapter 6: Real-World Build vs. Buy Case Study

convirza

Convirza's software analyzes millions of hours of customer calls, helping teams evaluate agents and gain actionable insights. Their platform dissects each conversation using 20-60 “AI indicators”, requiring it to manage highly variable workloads, high throughput, and sub-second response times.

Challenges Convirza Faced

Convirza's attempt to build an AI platform revealed several key challenges:

- **Unexplained System Crashes**: Frequent crashes disrupted operations and compromised data analysis.
- **Memory Issues**: Scaling call volumes strained computational resources.
- **Limited Expertise**: They realized they needed to expand their engineering team for more operational stability.
- **Lack of 24/7 Stability**: Ensuring round-the-clock reliability of their AI infrastructure required specialized knowledge their team didn't have.

They then explored options, including OpenAI, but they didn't meet performance or support levels. Building in-house would have tripled the timeline, which wasn't acceptable.

How a Managed Solution Addressed the Challenges

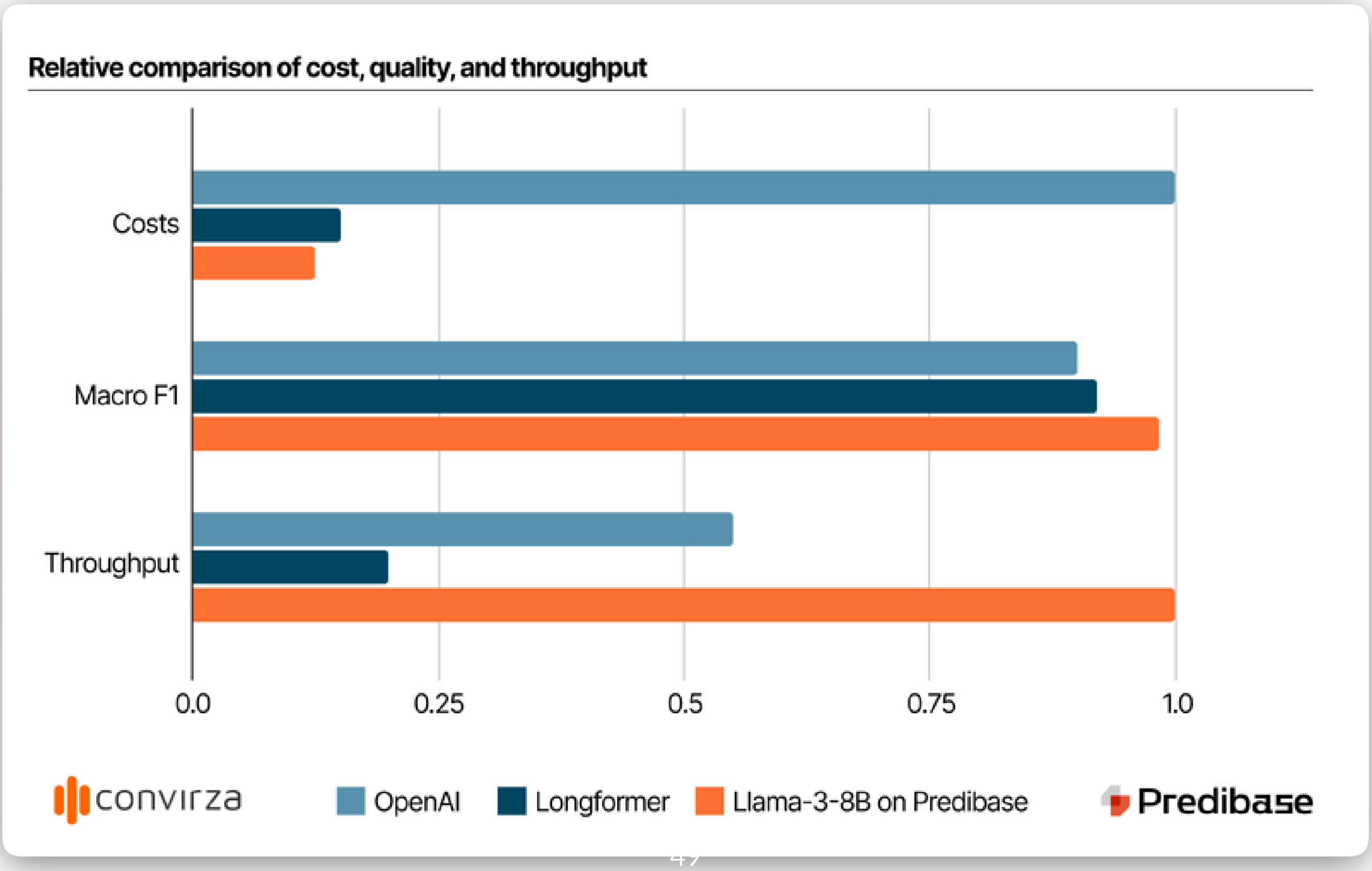
Convirza transitioned to Predibase's Inference Engine with LoRA Exchange to address these challenges. Convirza adopted a managed GenAI platform to handle their complex infrastructure needs, including serving and training. Leveraging Predibase’s LoRA eXchange (LoRAX) inference engine, they were able to increase the number of indicators from 20 to 60 without scaling up their infrastructure, and could serve all 60 specialized models at the cost of running just one base model.

Quantifiable Results

Using a managed platform helped achieve:

- 10x Reduction in Operational Costs: Compared to OpenAI.
- 8% Improvement in F1 Scores: On average.
- 80% Increase in Throughput.
- Consistent Sub-Second Mean Inference Times.

By outsourcing infrastructure management, Convirza could focus on core products such as: creating nuanced AI models that analyze calls using 40 more indicators, and help them give service.



As Giuseppe Romagnuolo, VP of AI at Convirza, noted,

“

Customers noticed a difference in the quality of their scorecards immediately. They can more effectively measure the metrics that matter to them and give agents the support they need to deliver quality service.

By combining these strategies, it allows for a managed platform that offers a strategic advantage such as faster innovation, greater reliability, and lower investment compared to an in-house solution.

This shows that in companies such as Convirza, they can compete effectively in the AI Space, and allow them to focus on value.

Chapter 7: Best Practices & Next Steps

Actionable Next Steps

Implementing a GenAI strategy is an ongoing journey. Here are concrete steps to get started:

1. Define Your Use Case

- **Identify 2-3 specific, low-risk use cases:** Focus on clear objectives, like improving customer support or automating content creation.
- **Set realistic timelines:** Aim for initial results within 1-2 months.
- **Define success metrics:** What quantifiable improvements do you expect to see? (e.g., reduced support ticket resolution time, increased content output)

2. Assess Readiness

- **Gather a cross-functional team:** Include IT, data science, and business stakeholders.
- **Use the checklist from Chapter 5:** Objectively evaluate your team's skills, resources, and infrastructure.
- **Identify key gaps:** Be honest about what you're missing.

3. Explore Solutions

- **Shortlist 3-5 vendors:** Focus on solutions aligned with your use cases and budget.
- **Request demos and free trials:** Test the platforms with your data and workflows.
- **Compare results:** Evaluate performance, scalability, and ease of use, considering their support for existing models.

4. Plan Quarterly Review

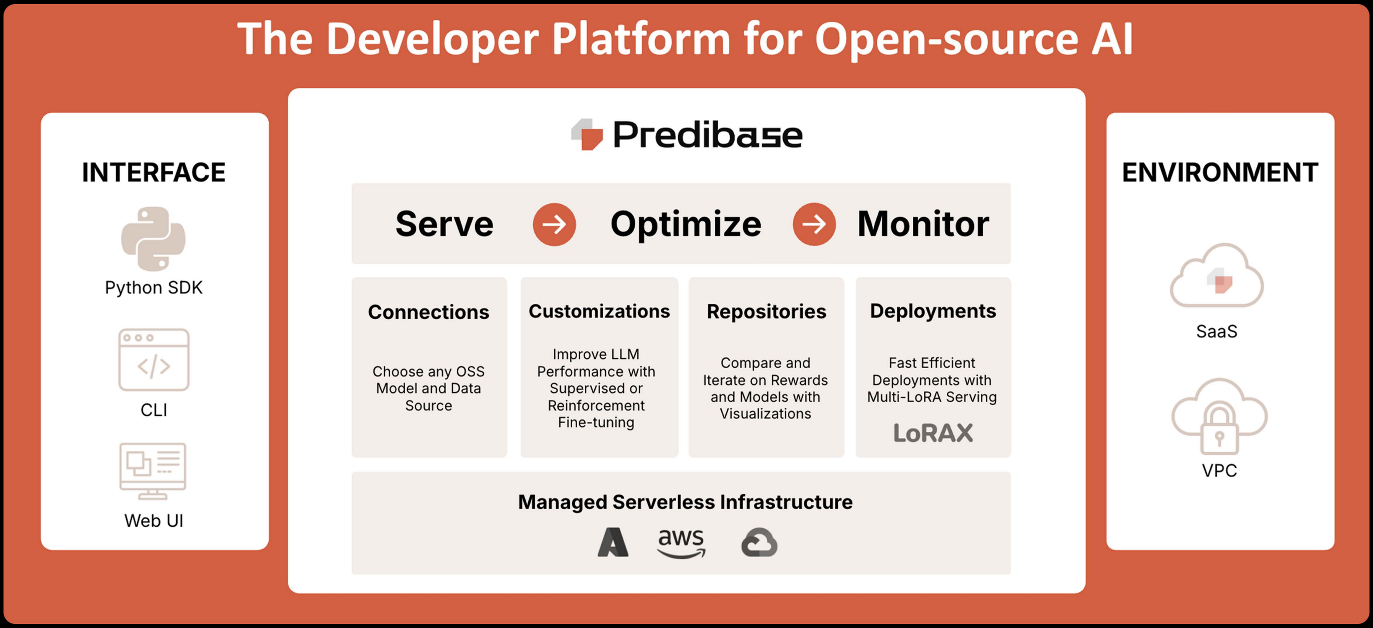
- **Schedule a recurring meeting:** Involve IT, data science, and business stakeholders.
- **Re-evaluate build vs. buy:** Consider team changes, new technologies, and evolving business needs.
- **Monitor key organizational metrics:** Track team composition, use cases, budget, and strategic priorities.

About Predibase

Predibase stands out as the premier platform for customizing and serving open-source LLMs at scale, offering a comprehensive fully managed solution for enterprises looking to harness the power of AI efficiently. Predibase is in production with AI leaders and innovators:

Qualcomm checkr MarshMcLennan Forethought nu

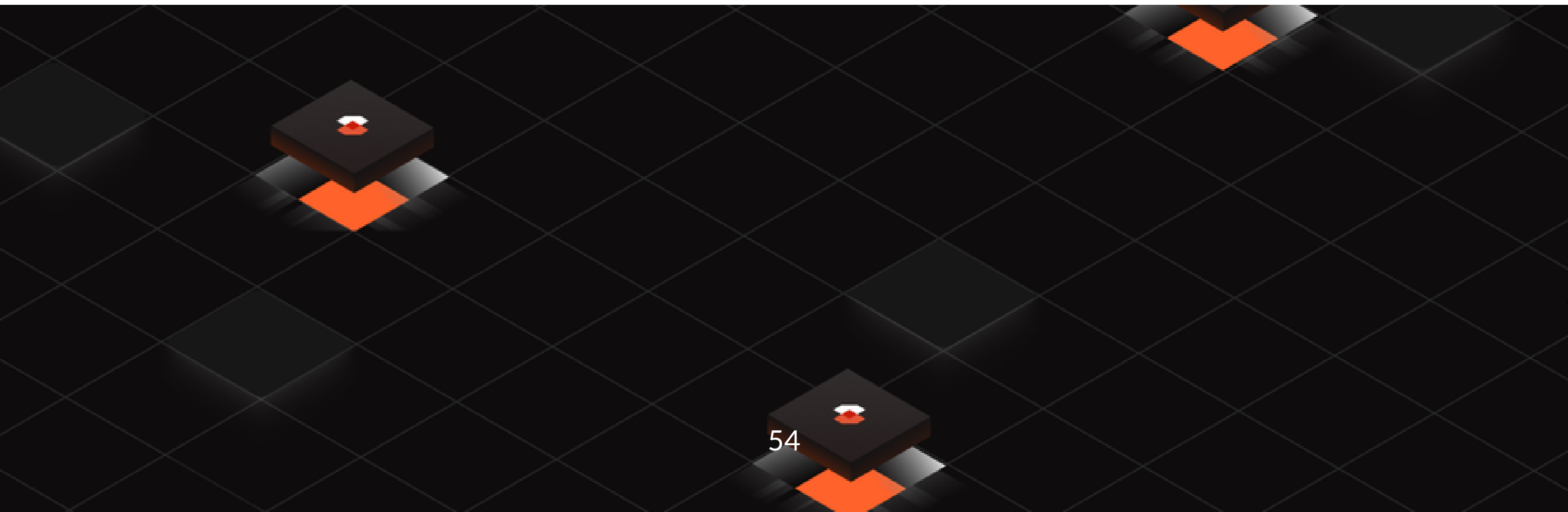
- Try for free with $25 in credits: https://pbase.ai/getstarted
- Book a custom demo with an LLM expert: https://pbase.ai/demo



The Developer Platform for Open-source AI
INTERFACE
Python SDK
CLI
Web UI
Predibase
Serve
Optimize
Monitor
Connections
Choose any OSS Model and Data Source
Customizations
Improve LLM Performance with Supervised or Reinforcement Fine-tuning
Repositories
Compare and Iterate on Rewards and Models with Visualizations
Deployments
Fast Efficient Deployments with Multi-LoRA Serving
LoRAX
Managed Serverless Infrastructure
aws
ENVIRONMENT
SaaS
VPC

The Predibase Difference

1. Hyper-Efficient Inference: Predibase simplifies inference with innovations like Turbo LoRA, which increases throughput by 2-4x compared to traditional methods, and seamless GPU autoscaling to handle enterprise workloads of hundreds of requests per second.
2. **Full Lifecycle Model Management:** The platform manages the entire model lifecycle, from fine-tuning to deployment, allowing teams to focus on innovation rather than infrastructure challenges.
3. **Enterprise-Grade Reliability:** Predibase is enterprise-ready, offering deployments in private cloud environments, guaranteed GPU capacity, and multi-region high availability to ensure mission-critical applications meet service-level agreements.
4. **Performance at Scale:** By leveraging Predibase's infrastructure, companies can significantly reduce the time and resources typically required for building and maintaining complex serving stacks while still achieving cutting-edge performance.



Predibase