



Open-Source-First Strategy for Generative AI Companies

Co-authored by Dinis Cruz and ChatGPT Deep Research

Summary:

This white paper advocates for a 100% open-source development model as a strategic foundation for generative AI (GenAI) companies. In an era where core algorithms and software can be rapidly replicated, the traditional “open-core” approach—offering a limited open-source version alongside a feature-rich proprietary version—often creates internal conflicts and stifles innovation. Instead, we argue that making *the entire technology stack open source* focuses companies on delivering unique value through services, execution, and customer experience rather than artificial feature lock-in. We discuss how fully open-source GenAI platforms can drive faster adoption and community-driven innovation, simplify architecture and deployment, and unlock new business models (such as cloud services, support, and consulting) without resorting to proprietary code. Real-world examples of successful companies that built profitable businesses on fully open-source products are provided, along with guidelines for implementing a sustainable open-source-first strategy. The goal is to present CTOs, technical leaders, and founders with a clear, authoritative framework for leveraging open source as a competitive advantage in the GenAI industry.

Table of Contents:

1. [Introduction: The Open Source Imperative in GenAI](#)
2. [The Open-Core Dilemma: Dual Personality Disorder in Startups](#)
3. [Why 100% Open Source? – Key Benefits](#)
 - 3.1 [Faster Innovation and Adoption](#)
 - 3.2 [Customer Trust and No Tech Lock-In](#)
 - 3.3 [Simplified Architecture and Deployment](#)
4. [Business Models for Fully Open-Source Companies](#)
 - 4.1 [Service and Support at Scale](#)
 - 4.2 [Managed Cloud Services \(Open SaaS\)](#)
 - 4.3 [Dual Licensing Without Closed Features](#)
5. [Case Studies and Examples](#)
 - 5.1 [Red Hat & SUSE: Pioneers of the Open-Source Model](#)
 - 5.2 [Nextcloud & Zabbix: Profitable and 100% Open Source](#)
 - 5.3 [Generative AI in the Open: Stability AI and Others](#)
6. [Best Practices for Building an Open-Source-First Company](#)
7. [Conclusion](#)

1. Introduction: The Open Source Imperative in GenAI

Open source has long been a driving force in software development, proving its value in fields from operating systems to databases. Today, in the fast-evolving domain of Generative AI, open source is increasingly seen as not just a licensing choice but a strategic imperative. GenAI startups face intense

competition and a rapid pace of innovation; models and code can be reproduced or reimplemented quickly by others. In this context, attempting to create a “moat” by keeping parts of the technology proprietary often yields diminishing returns. Instead of relying on secret code, companies are discovering that competitive advantage in AI comes from superior execution, domain expertise, data quality, and customer-centric services – elements that are not easily copied even when the source code is open ¹. As one industry analysis noted, concerns that open-sourcing a product will simply enable competitors to catch up are often overstated; successful products require far more than just code access – they demand *refined execution, deep business relationships, network effects, an established reputation*, and the ability to handle countless real-world edge cases ¹. These factors constitute a far more durable moat than a few proprietary algorithms.

The trajectory of AI technology also favors openness. Major tech players and research institutions are embracing open collaboration to drive standards and innovation. For example, when Meta released the weights of its LLaMA model to researchers, the open-source community rapidly adapted and improved it, dramatically reducing the cost and resources required to run large language models ². Such community-driven progress would be impossible in a closed environment. In summary, an open-source-first strategy aligns with the ethos of AI research – which has historically thrived on shared knowledge – and positions GenAI companies to harness the collective intelligence of a global developer community. This introduction sets the stage for why forward-looking GenAI companies should consider *everything* open source, and how this approach overcomes the limitations of hybrid models that split their personality between open and closed. We next examine the pitfalls of the prevalent “open-core” approach and contrast it with a fully open model.

2. The Open-Core Dilemma: Dual Personality Disorder in Startups

Many software startups begin with an open-source project that gains community traction. However, as they seek revenue or VC investment, a common strategy is to adopt an “open-core” model: maintain a free open-source base, but develop additional proprietary features or a separate enterprise edition for paying customers. This approach often creates a conflicted identity – a sort of Jekyll-and-Hyde split within the company. On one hand, the company touts its open-source community cred; on the other, it starts injecting closed-source hooks intended to “nudge” users into paid licenses for the full experience. The result is a tension between community goodwill and commercial interests. Companies may deliberately withhold the most useful or production-critical features from the open version, effectively using the open-source project as a lead-generation funnel. This can breed mistrust among developers and technical decision-makers, who feel the “Community Edition” is a crippled version of the real product. It also diverts internal focus: engineering discussions shift from “what’s best for the product and users?” to “what features do we paywall to drive conversions?” – a distraction that can *stifle innovation*. As Dirk-Peter van Leeuwen, CEO of SUSE (one of the oldest open-source enterprise companies), observed in a 2025 interview: “*Whenever you remove the freedom that comes with open source, you stifle innovation. The real value for vendors lies in the customer experience and service quality, not code ownership. Make sure what you sell is a tangible service and value customers are willing to pay for.*” ³. In other words, engineering effort spent creating artificial limitations or proprietary add-ons is effort not spent improving the core product for everyone.

Open-core models also introduce engineering complexity. Teams must maintain two codebases or a plugin architecture with conditional features, adding overhead to testing and release cycles. Deployments become more complex as well – users must consider which version (open vs. enterprise) to install, and migrations from one to the other might be non-trivial. In contrast, a company that is

open-source-first simplifies these dynamics by having **one unified codebase for all users**. This eliminates the internal debate over “what do we open source?” – the answer is *everything*. By removing that line of demarcation, the company can channel all its energy into making the product better for all users, while developing a separate strategy for monetization that doesn’t depend on withholding features. The next section delves into the benefits that such a 100% open approach can bring, particularly in the context of GenAI firms where agility and community engagement are paramount.

<aside style="background: #F0F0F0; border-left: 4px solid #CCCCCC; padding: 10px; margin: 10px 0;">*Open-Core vs. Open-Source-First – At a Glance*

Open-Core Model:		Open-Source-First Model:
- Open Source “Community” edition		- Single open source codebase for all features
- Proprietary Enterprise extensions		- No proprietary extensions or locked features
- Value proposition: “pay for extras”		- Value proposition: “pay for service & convenience”
- Risk of user lock-in via tech		- No tech lock-in; compete on service quality
- Development split between two agendas		- Development focused on user needs and innovation

Figure 1: Contrasting the open-core approach (mix of free and proprietary code) with a fully open model. The latter avoids internal conflicts and builds goodwill by treating the community as equal beneficiaries of all improvements.</aside>

3. Why 100% Open Source? – Key Benefits

Adopting an all-open-source development model is not just a philosophical stance; it brings concrete technical and business benefits. This section outlines the key advantages, particularly relevant to GenAI companies and enterprise technology providers, of keeping the entire technology stack under an open-source license.

3.1 Faster Innovation and Adoption

Open source is arguably the most successful approach to developing and distributing cutting-edge software infrastructure ⁴. By removing barriers to entry, a fully open project encourages widespread trial and adoption: any developer, researcher, or enterprise can download and experiment with the software without procurement hurdles. This broad adoption in turn fuels rapid feedback loops and community contributions, which are essential for high-velocity innovation ⁴. In the context of AI, this means more people testing models, discovering new use cases, identifying bugs, and contributing improvements. A vibrant open community can accelerate feature development and harden the software far beyond what a single company’s team might achieve in the same time frame.

Generative AI models and systems particularly benefit from openness. For example, after an initial large model is open-sourced, researchers around the world often contribute efficiency improvements, such as optimized model architectures or training techniques, that drastically reduce resource requirements ². The leaked release of Meta's LLaMA model weights spurred a wave of innovation in the open community, leading to optimized variants and cost reductions in running large language models by orders of magnitude ². This kind of iterative collective improvement simply doesn't happen when code is locked behind closed doors. The net effect is that an open-source GenAI company can evolve its product faster and stay at the cutting edge, leveraging contributions from academia, independent developers, and other companies.

Moreover, openness strengthens **ecosystem integration**. Other tools and platforms are more likely to integrate with an open-source project because they can read the source, write plugins or extensions, and ensure compatibility without needing vendor permission. This often leads to an ecosystem of extensions and add-ons around the core project, increasing its utility and stickiness in the market ⁵ ⁶. In summary, fully open sourcing your AI software "primes the pump" for exponential adoption and innovation, whereas a partially closed approach may achieve a slower trajectory due to reduced community engagement ⁷ ⁸.

3.2 Customer Trust and No Tech Lock-In

Enterprise customers and technical teams value control and transparency. When a product is 100% open source, they know they are not at risk of being locked into a proprietary technology that could change terms or stagnate. They have the *freedom to run the software anywhere* – on their own infrastructure, on any cloud, or at the edge – and to customize it to their needs. This greatly increases trust in the vendor. In contrast, if certain "enterprise" features are only available in a closed source module, savvy customers become wary: they realize their critical capabilities depend on a black box controlled solely by the vendor. By eliminating this concern, an open-source-first company can actually win more enterprise adoption, not less. Indeed, offering an open-source product with no hidden feature lock-in can be a compelling selling point for CIOs and CTOs who have been burned by proprietary vendor lock-in before.

Importantly, removing tech lock-in does *not* mean a lack of revenue opportunities. It simply means the customer is not forced to pay for the *right to use the software's features*. Instead, revenue comes from optional services that add value beyond the raw code. We will detail these models in Section 4, but they include things like hosting, support, and tailored solutions – the kinds of offerings customers are happy to pay for because they provide tangible benefits, not artificial restrictions. SUSE's CEO, for example, emphasizes that even in highly competitive arenas like enterprise Linux and AI platforms, his company's strategy is to keep the code free and open and compete by offering superior support, security, and scalability "*without restricting the freedom that drives exponential innovation*" ⁹ ³. In other words, the absence of a tech lock-in becomes an advantage: customers feel comfortable adopting the platform widely, since they know they can continue to use and even self-support it if needed, and this widespread adoption creates a larger addressable market for value-added services.

From a reputational standpoint, a fully open approach also demonstrates integrity and commitment to the community. It says "we're confident enough in our value that we don't need to withhold source code." This can attract top engineering talent (who often prefer working on open projects) and align with the values of public sector or academic partners who require open-source solutions for transparency reasons. In summary, *open source builds trust*. It assures users that they won't be handcuffed by proprietary limitations, thus encouraging deeper investment in the ecosystem. That

trust, once earned, translates into willingness to engage commercially for services around the product. As one open-source business expert succinctly put it, once users establish trust and rely on an open-source product for critical needs, they are “willing to pay top dollar for the convenience” of enterprise-grade services on top of it ¹⁰.

3.3 Simplified Architecture and Deployment

An often underappreciated benefit of an open-first strategy is the simplification of the product architecture and deployment processes. With a single open codebase, the software architecture is cleaner—there is no need to create artificial interfaces or plugin systems to separate “community” and “enterprise” functionality. Every feature is developed in the open, which means the entire system can be designed cohesively without worrying about what portion of the code will be behind closed doors. This tends to result in a more modular, transparent design (since outside contributors need to understand it), and avoids the bloat of maintaining parallel feature sets. Over time, this simplicity pays dividends in maintainability and technical agility: the engineering team can refactor or upgrade components without the complication of coordinating changes across proprietary forks.

Deployment and DevOps are also greatly simplified. For users, there’s only one edition of the product – they don’t have to evaluate a community vs. enterprise edition and deal with upgrading from one to the other when they grow. There are no hidden “switches” that only the vendor can enable in their hosted version; if something works in the vendor’s cloud, it works in the same open-source package the user deploys themselves. This consistency means fewer surprises in production. It also enables a truly *agnostic deployment model*: since the software is open, a customer can deploy it on their own cloud account, on-premises data center, or even air-gapped environments for sensitive workloads. Particularly for GenAI applications, which might need to run close to proprietary data sources or on specialized hardware, having the freedom to deploy anywhere is a huge benefit. We often see that open-source AI tools gain adoption in research and enterprise contexts precisely because they can be run locally or customized – something closed AI APIs typically don’t allow.

Additionally, open source encourages a “toolbox” mentality in the architecture. Engineers and users know they can extend or modify any part of the stack, so they are more likely to build automation scripts, plugins, or integrations that tailor the software to specific workflows. When these enhancements are shared back with the community, the product becomes easier to deploy and operate for everyone. For example, many open-source projects benefit from community-contributed Docker images, Kubernetes Helm charts, or Ansible playbooks that make installation and scaling easier. A closed-source product would have to provide all this by itself (or not at all). By being open, you essentially crowdsource not only feature innovation but also operational enablement.

Finally, an open architecture future-proofs the company against major technology shifts. If a new AI model, database, or cloud service arises, the company and community can quickly work to integrate or swap in that component because they have full access to the code. There is no concern about a third-party proprietary dependency blocking adaptation. In the rapidly changing GenAI landscape – where today’s state-of-the-art model may be obsolete in a year – this flexibility is crucial. The company can confidently state to customers that **technology is not our moat; our service is**. If a better tool emerges, an open-first company can adopt it or interoperate with it, rather than clinging to a closed legacy component. In sum, the open-source approach leads to a cleaner, more adaptable architecture and a frictionless deployment experience, which benefits both the vendor and the users.

4. Business Models for Fully Open-Source Companies

One of the most common questions about an all-open strategy is: “How do we make money if we give away the software?” It’s a valid question—after all, a company needs revenue to pay developers and to invest in the *non-functional requirements* (performance, security, usability, etc.) that open-source volunteer communities might overlook. The key is to decouple the value of the software *license* (which is zero-cost in open source) from the value of services, expertise, and convenience *around* the software. In other words, you are not selling the bits; you are selling what the bits enable. Many companies have proven that **open source and profit are not at odds** ¹¹, and indeed a healthy commercial ecosystem can drive more open-source development. Below, we outline several sustainable business models for 100% open-source companies, emphasizing that none of them require proprietary add-ons. Each model can stand alone or be combined with others.

4.1 Service and Support at Scale

The earliest open-source business model, exemplified by companies like Red Hat, SUSE, and others, was selling professional services and support for freely available software. In this model, the *software* remains free and open, but customers pay for expertise to ensure it runs smoothly in their environment. Services can include: **technical support** (e.g. 24/7 helpdesk, SLAs for issue resolution), **consulting and integration** (customizing the software, integrating it with other systems), **training and certification** programs, and **feature development contracts** (building new features that a particular customer needs, contributed back to the open project).

Some argue that pure services and support are low-margin and hard to scale, especially compared to selling software licenses. It’s true that traditional consulting can be human-capital intensive. However, modern open-source companies have found ways to *productize* their services to achieve scale. For example, support can be sold via tiered subscription plans (Gold, Silver, Bronze support levels) with predictable recurring revenue rather than one-off hours. Training has moved online to reach global audiences at low cost. Certification exams (for administrators or developers of the open-source software) can generate revenue and also increase the product’s stickiness in enterprises. By investing in knowledge bases, automated tooling, and global partner networks, even service-heavy models can reach large scale.

There are numerous instances of companies succeeding with this approach. **Red Hat** famously built a billion-dollar business on subscription support and services for enterprise Linux, all while contributing 100% of its code improvements back to the open source (Red Hat Enterprise Linux was freely reproducible from source, and indeed clones like CentOS existed) ¹² ¹³. Red Hat proved that you can generate substantial revenue by *providing timely updates, patches, integrations, and world-class support* for mission-critical open software. **Percona** is another example – a company that offers support and consulting for open-source databases like MySQL and MongoDB, competing successfully against vendors of proprietary forks. These companies show that if you become the *best in class at servicing a particular open-source technology*, customers will pay for that expertise even though they can theoretically use the software for free ¹⁴.

One crucial point is that a services-based model also fuels better software quality. When a company’s income depends on making the open product stable and easy for others to use, it has a strong incentive to continuously improve the core project. As noted in an analysis by open-source entrepreneur Sid Sijbrandij, “the goal of providing support should be to make the software itself simpler, easier, and

better to use... integrating improvements into the software" to reduce the need for support over time ¹⁵. This virtuous cycle means paying customers essentially fund a better open-source product for everyone, which in turn grows the user base and potential pool of future customers. The caveat is that pure support/services companies may not achieve the hyper-growth VCs desire ¹⁶, but they can be solid, profitable businesses – which is often perfectly suited for small to mid-sized companies focused on sustainable growth rather than "unicorn" valuations.

4.2 Managed Cloud Services (Open SaaS)

In recent years, a powerful revenue model for open-source firms has been offering the software as a hosted service. In this "open source SaaS" model, the company provides a ready-to-use cloud service running the open-source software (often with extra automation around it), and charges for usage or subscriptions. The key distinction from typical SaaS is that the *customer could run the software themselves*, but they pay the company to handle the operational burden. This leverages the fact that operating complex AI software reliably at scale is non-trivial, and many users will pay for a turnkey solution rather than managing it in-house.

For GenAI companies, a managed service might mean an API endpoint for an open-source model, a web platform for AI model training/inference, or a fully managed on-prem deployment for enterprise customers. The value to the customer is convenience, time-to-market, and possibly enhanced performance/security if the company's hosted version is optimized. Importantly, *all features of the software are available in the open-source code* – the paying customer is not buying new capabilities, they are buying reliability and ease-of-use. Many modern open-source startups have taken this route, effectively becoming cloud providers of their own open tech. For instance, Elastic (creators of Elasticsearch) and MongoDB shifted focus to their hosted cloud offerings (Elastic Cloud, MongoDB Atlas) as primary revenue drivers, since users were willing to pay for a worry-free cluster management even though the core software was open. An industry survey notes that efforts to monetize open-source software via purely proprietary features tend to be only mildly successful, whereas "efforts to monetize the cloud service [can be] wildly successful," once users trust the software for critical operations ¹⁷. Users will pay a premium for the convenience of not running the infrastructure themselves, especially as their usage scales ¹⁰.

A GenAI example can be seen in the way **Stability AI** monetizes Stable Diffusion. Stability AI open-sourced its generative image model and even open-sourced its DreamStudio web interface as the project "StableStudio" to foster community contributions ¹⁸. Anyone can run Stable Diffusion on their own hardware, yet many artists and developers choose to use Stability's managed DreamStudio service for generating images because it's faster and easier than local setup. Stability AI's strategy is to ensure their hosted DreamStudio stays "the preferred, managed instance" of the open-source StableStudio, providing a seamless experience with the latest models, while the open project grows via community innovation ¹⁹. This is a classic open SaaS play – the company doesn't hold back any code (even their UI is now open), but they offer a cloud service that saves users effort.

The risk often cited in this model is the "cloud giant" problem: what if a big provider (like AWS) takes your open-source project and offers it as a service, undercutting your business? This indeed has happened (AWS offering MySQL, PostgreSQL, Elasticsearch as services, for example). Some vendors reacted by changing licenses to restrict cloud providers (Elastic, MongoDB with SSPL license). However, an open-source purist strategy accepts that competition and focuses on being the best at operating and supporting their own software. In many cases, the originating company has an edge: they have the core expertise and often can integrate latest features faster. And if AWS enters your space, it also legitimizes

it – it can actually reassure large customers that this technology is here to stay, with multiple vendors supporting it ²⁰. The company must then compete on offering an “exceptional experience” – e.g., a more enterprise-tailored service, better support, or on-prem options the generic cloud service doesn’t offer ²¹. The advantage for the customer is that, because the software is open, they could migrate off the vendor’s service if needed (no *permanent* lock-in) – but they likely won’t if the service is high quality and fairly priced. In summary, providing a managed service allows an open-source firm to capture value (cloud revenues) without needing proprietary software, and it plays to the strength of convenience at scale. This model has become so predominant that many investors evaluate open-source startups largely on their “open-to-cloud conversion” strategy (often measuring what fraction of users eventually pay for a hosted version).

4.3 Dual Licensing Without Closed Features

Another model, used historically and still viable in certain cases, is **dual licensing**. In a dual-license setup, the entire codebase is available under an open-source license (often a copyleft license like GPL), but the company also offers the same code under a different proprietary license for customers who might need it (for example, an OEM who wants to embed the software in a closed product and cannot comply with the open-source license terms). The classic example is MySQL: before being acquired by Sun/Oracle, MySQL AB made its database available under GPL for the community, but also sold commercial licenses to companies that wanted to integrate MySQL into their products without the GPL obligations. This meant MySQL’s developers could be paid via those license fees, even though community users had full access to source and features. Dual licensing was a popular early open-source business model ²², and it has the benefit of *not* creating two versions of the software – it’s one codebase, just two licensing options.

The important caveat, and the reason this fits into an “no proprietary features” philosophy, is that under dual licensing, **no user is forced to pay to get features** – they can always get them under the open license if they are willing to adhere to its terms (which often means keeping their own usage open or internal only). Paying is just a convenience to get a more permissive license. In the context of GenAI, dual licensing might be useful if, say, you release your code under a strong copyleft (to ensure improvements are shared) but offer a commercial license to, for example, an enterprise that wants to integrate your model into their closed SaaS. As long as the open-source license is a true open license (approved by OSI) and the public repo has all the code, this model doesn’t fragment the community or codebase.

One newer variant of dual licensing is the “open core but with source-available for the rest” approach – for instance, a company might open source 90% of the code under Apache/MIT and have a few add-ons under a source-available license (not open source, but code visible). However, this again crosses into having features that only paying customers can actually use in production (since source-available licenses often forbid commercial use without payment). We mention this because some companies consider it a compromise – but it does violate the principle of no proprietary features. A true dual-license model à la MySQL or Qt (the GUI framework) keeps all functionality truly open for the community. It can work if your user base includes third parties who want to embed or resell your software under different terms.

It’s worth noting that dual licensing has become less common in recent years, partly because many companies shifted to either pure open-core or pure services models. Dual licensing requires careful management of contributor agreements (you need full rights to license contributions under both licenses, often contributors must sign an agreement). If executed properly, though, it remains a model

that upholds the “everything is open” ethos while providing a revenue stream. In any case, whether via services, managed hosting, or dual licensing, the fundamental idea is that **funding for the open-source project should come from delivering genuine value — not from locking up the technology**. As Dinis Cruz (co-author of this paper) often advocates: every open-source project *should* have a commercial angle to support its development, but “*that funding should not come from locking in customers into the technology. It should come from services and making it easy to add value around the technology.*” (Voice memo, 0:29–3:00). In the next section, we’ll see how these principles have played out in practice by examining companies that have successfully navigated the open-source business journey.

5. Case Studies and Examples

No strategy would be complete without real-world examples. Here we highlight several companies – from industry giants to small startups – that have implemented a 100% open-source approach (or close to it) and discuss the outcomes. These examples serve both as proof points that “fully open” can be sustainable, and as sources of lessons for GenAI startups charting a similar course. We focus on cases where the core technology remained open source, and the business model did *not* rely on holding key features proprietary.

5.1 Red Hat & SUSE: Pioneers of the Open-Source Model

Any discussion of open-source business models usually starts with **Red Hat, Inc.** Founded in 1993, Red Hat was the first company to prove that a pure open-source software product could be commercialized at global scale. Red Hat Enterprise Linux (RHEL) is 100% open source (GPL license); the company’s revenue came from selling subscriptions that included support, updates, and certifications for enterprise deployments. By 2012 Red Hat had surpassed \$1 billion in annual revenue, and it continued to grow, eventually being acquired by IBM in 2019 for \$34 billion – all built on open code. As an analysis by Open Core Ventures noted, Red Hat succeeded by delivering “*vetted open source updates and multi-product support, services, and training*”, creating enormous value for customers, even though none of the software was proprietary ²³ ²⁴. In fact, Red Hat’s commitment to openness was so strong that for decades it refrained from the easier path of creating a proprietary management tool, etc. The company instead expanded its open-source portfolio (JBoss middleware, OpenShift platform, etc.) and sold subscriptions for those, maintaining a principled stance that “*all its software is completely open source*” ¹³. This principled approach earned it deep trust with customers and the open-source community. The lesson for new companies is that while Red Hat’s scale is exceptional, the underlying dynamic – *focus on support/service excellence and keep the code open* – can create a durable business with loyal enterprise customers.

A companion example is **SUSE**. Founded in 1992 in Germany, SUSE was another early Linux company and remains a major player in enterprise open source. SUSE’s model has been very similar to Red Hat’s: all core products (SUSE Linux Enterprise, etc.) are open source, and revenue comes from support subscriptions and services. SUSE’s history shows long-term viability in this model; the company has over 30 years of continuous operation and has grown a global customer base. In 2023, SUSE made headlines by announcing it would open source and upstream all of its SUSE Enterprise Linux code (abandoning any oblique proprietary bits) and even pay to maintain a fork (Liberty Linux) truly open, doubling down on the strategy amid changes at Red Hat’s end. SUSE’s CEO’s view, as cited earlier, encapsulates their strategy: keep the software free and build business value on *customer choice, lack of lock-in, and*

premium services ⁹ ³. SUSE's success also underscores a point relevant to *digital sovereignty*: in regions like the EU, governments and enterprises favor open-source solutions to avoid dependence on foreign proprietary software ²⁵. By being fully open, companies like SUSE position themselves as trustworthy partners in sensitive industries (government, defense, etc.) where transparency is a must.

Both Red Hat and SUSE illustrate that even at large scale, it is possible to resist the temptation of open-core proprietary add-ons. They have had to innovate in business execution – for example, running *high-margin training and certification programs* that effectively make their Linux the standard by ensuring a workforce trained on it. Red Hat also built a rich partner ecosystem (hardware certifications, ISV software certified on RHEL, etc.), which is only possible when your code is open and broadly available. The takeaway for GenAI startups: you can learn from these pioneers by fostering a broad adoption of your open technology and then monetizing the ecosystem's need for quality, support, and assurance. While you may or may not reach billions in revenue, the model scales from niche to global—Hortonworks, for instance, followed a similar open-only Hadoop distribution model, reaching hundreds of millions in revenue before merging with its competitor. The primary caution is that pure support models have fierce competition (as Sid Sijbrandij pointed out, Red Hat's model hasn't been replicated at that billion-dollar scale by others easily ²⁶ ²⁷). Thus, blending support with the managed service model (as Red Hat did with OpenShift and SUSE with Rancher/Kubernetes services) is often a winning approach today.

5.2 Nextcloud & Zabbix: Profitable and 100% Open Source

While Red Hat and SUSE are large enterprises, it's equally important to look at small-to-mid size companies that have thrived with fully open products. Two such examples are **Nextcloud** and **Zabbix**, each leaders in their respective domains (collaboration software and monitoring software), each built around a 100% open-source core.

Nextcloud is an open-source file sync, storage, and collaboration platform (a bit like an open alternative to Dropbox/Google Drive/Microsoft 365, targeted at enterprises and privacy-conscious users). Frank Karlitschek founded Nextcloud in 2016 as a fork of ownCloud, explicitly committing to a pure open-source model. Nextcloud's software — server, clients, and features — are all licensed under the GNU AGPL or compatible open licenses. The company's website proudly states: "*Everything we do is 100% open source. We do not do open core or proprietary extensions.*" ²⁸. Instead of selling software licenses, Nextcloud's revenue comes from enterprise support subscriptions, consulting, and managed on-premise solutions. This model has been successful: Nextcloud has a wide adoption in the public sector and large companies (for example, the French government and other European institutions use Nextcloud), who pay for the assurance of support and custom development. Nextcloud's open approach means clients can inspect the code for security (a big selling point in government deals) and are not tied to Nextcloud if they ever want to switch – yet they stay because Nextcloud's service and responsiveness add value. By keeping the entire ecosystem open, Nextcloud also benefits from community apps and integrations that extend the platform, making it more attractive. Their approach demonstrates that even in a highly competitive space (collaboration software dominated by Big Tech), a nimble company can carve out a profitable niche by being *the open, no-lock-in alternative*. It's worth noting Nextcloud's CEO has actively used regulatory and community channels to challenge proprietary competitors (like filing an EU complaint against Microsoft OneDrive bundling), further leveraging their open ethos as a competitive differentiator.

Zabbix is another remarkable story. Zabbix is a network/server monitoring software (similar to Nagios, Datadog, etc.). Founded by Alexei Vladishev, Zabbix has been *open source from day one (in 2005)* and has

remained free to use, with the company offering paid support, training, and now a Zabbix Cloud hosted option. For 20 years Zabbix grew its user base worldwide and proved the viability of an open model. In 2025, at Zabbix's user conference, the founder reaffirmed that "after 20 years of existence, the company remains committed to keeping Zabbix **license-free**" and noted: "*It has been proven that it is possible to be open source and maintain a business model... Zabbix will always remain an open source product.*"²⁹. This statement is powerful: a clear commitment that no matter how the business evolves, they will not introduce closed licensing. Zabbix's revenues come from support contracts and professional services, especially for large-scale users who need guaranteed service levels or custom features (which are often contributed back to the main project). By staying completely open, Zabbix has also achieved something important in the B2B software world – it became a **standard tool** in its category, used in thousands of companies. Competing commercial products often require convincing customers of value; Zabbix, being free, is often tried first, and if it meets the need, the company can later convert some users to paid support. The credibility that "we won't bait-and-switch you with a license change" is invaluable and likely contributes to their strong community and conference turnouts. Additionally, the openness has allowed third-party service providers to flourish (many MSPs use Zabbix for their clients, some becoming Zabbix partners), further entrenching it in the market.

Both Nextcloud and Zabbix illustrate that a *dual codebase is not necessary to be profitable*. They focus on a specific domain, deliver all the functionality openly, and build a business on services and trust. They also show that small/medium businesses can punch above their weight by harnessing community development. Nextcloud can innovate in features (like end-to-end encryption, secure video conferencing, etc.) partly because a global community of developers (and even paying customers) contribute enhancements. Zabbix similarly integrates with countless devices and software thanks to community-contributed templates and plugins. This breadth might never have been achieved by the company alone. For GenAI startups, the lesson is that even if you're not aiming to be a billion-dollar company, an open-source model can support a healthy business. Moreover, starting fully open can differentiate you in a landscape where many AI startups are closed or "open-core." There is a growing segment of customers (especially in enterprise and government) that actively prefers solutions that are open source for reasons of auditability, flexibility, and avoiding vendor lock. By positioning yourself as a 100% open provider, you tap into that demand.

5.3 Generative AI in the Open: Stability AI and Others

Finally, let's focus on generative AI companies specifically, since that is the context of this paper. The GenAI boom of 2022-2025 saw a mix of approaches: some companies went fully proprietary (e.g. OpenAI's GPT-4 model is closed-source), while others embraced open source as a way to compete with relatively limited resources. Perhaps the most prominent example of the latter is **Stability AI**, the startup behind *Stable Diffusion*. Stable Diffusion is a text-to-image model that was released openly in 2022, allowing anyone to download the model weights and run or fine-tune it. This was a watershed moment for AI: it "democratized" a capability (image generation) that until then was available mainly via closed APIs like DALL-E or MidJourney. Stability AI's open approach led to an explosion of innovation – developers built new UIs, optimized the model for various hardware, created novel art styles, and integrated it into other software. By open-sourcing their core model, Stability AI achieved massive *community adoption*: Stable Diffusion became a household name in AI circles and is embedded in dozens of products today. That wide usage created commercial opportunities: Stability AI offers a **hosted service (DreamStudio)** for end-users who want a simple web interface or API, and it also engages with enterprises to provide custom models or on-prem deployments. All the while, the core tech remains open. The company even open-sourced the DreamStudio interface as **StableStudio** in 2023, signaling a commitment that even their front-end will evolve as a community project¹⁸ ¹⁹.

Stability AI explicitly stated their reasoning: “*the best way to expand... is through open, community-driven development rather than a private iteration of a closed-source product... Our goal is to work with the broader community to create a world-class user interface for generative AI that users fully control.*”¹⁸. This underscores how in AI, being open can be a competitive advantage: while others guard their UIs or models, Stability invites collaboration to improve faster. It is a bet that the **community can outpace what any single company could do alone**¹⁸.

Another interesting case is **Meta (Facebook)** in the GenAI space. Meta AI surprised the industry by open-sourcing large language model weights (LLaMA 2) in 2023 with a permissive license (allowing commercial use with some conditions). While Meta is not monetizing LLaMA directly as a product, the strategic intent was to spur an ecosystem that could compete with closed offerings from OpenAI/Google. By releasing a powerful model openly, Meta leveraged the community to harden and find uses for it. This move validated that even AI giants see open source as beneficial for adoption. A quote from Meta’s AI division stated that for a technology to truly succeed and not get locked into one company’s ecosystem, *it needs to develop into a full ecosystem of tools and integrations*, which wouldn’t happen if only one company (themselves) used it³⁰. In essence: open sourcing was the way to ensure widespread innovation on top of LLaMA, akin to how Linux succeeded over proprietary Unix. For startups, Meta’s strategy indicates that open models can gain industry support (even Microsoft partnered with Meta on LLaMA 2) and become standards that others build on, which can indirectly benefit the originator.

We should also mention **Hugging Face**, not a model creator but a platform that became central to the open AI movement. Hugging Face hosts a repository of thousands of open-source models and datasets. Their approach is to be the hub of open AI development, offering tools (like the Transformers library, also open source) to work with models. Hugging Face the company then offers an enterprise platform, hosted inference APIs, etc., to monetize. By championing openness, Hugging Face attracted \$400M in investments and partnerships with big tech³¹. They proved that facilitating open source in AI can itself be a lucrative business model (through hosting and enterprise services), without needing any proprietary IP – in fact, their value is in being the *Switzerland* of AI models, neutral and open.

In summary, the generative AI field provides timely examples that align with our thesis: releasing models and tools openly can create a groundswell of usage and improvement. The companies that have done so (Stability AI, Hugging Face, and others) are then monetizing the secondary layers – whether it’s user-friendly managed services, or enterprise support, or custom solutions – rather than the core technology. This trend suggests a future where foundational AI models become commoditized (often open source or at least widely available), and the real business differentiation comes from how you deliver it as a product or service. GenAI startup founders should take note: your long-term defensibility might lie not in hoarding your model weights or code, but in cultivating a large user base and community around an open platform, then serving that community’s needs. An open-source strategy can be a powerful marketing tool as well – many developers will try an open AI tool out of curiosity (since it’s free), whereas a closed product might struggle to get mindshare unless it’s clearly superior. And given the pace at which AI research moves, betting on proprietary advantage in algorithms could be an ever-shrinking window; an open approach hedges that by making you the center of a shared innovation network.

6. Best Practices for Building an Open-Source-First Company

Adopting a 100% open-source model requires not just a shift in licensing, but also adjustments in how a company operates. Here are some best practices and recommendations for leadership (CTOs, CEOs, and Open Source Program Officers) to successfully execute this strategy:

- **Design for Community Contribution:** Make your repository public from day one and use public issue trackers. Accept pull requests, engage with external contributors, and build a culture of collaboration. A welcoming, active community can become an extension of your R&D team. This means writing good developer documentation, using open communication channels (forums, chat), and being transparent about your roadmap. Community contributions not only improve the product but also create evangelists for your solution.
- **Adopt an Open Governance Model (if suitable):** Some projects benefit from having multiple stakeholders in governance (e.g., a foundation or a technical steering committee) to signal that it's not controlled by a single vendor. This can reassure enterprise adopters. However, governance can remain single-vendor-driven if the community trusts you; what's crucial is being responsive and not acting unilaterally against user interests. In all cases, avoid sudden shifts like moving from an open license to a closed one – that can permanently alienate your base (as seen when some companies relicensed to limit cloud providers and met community backlash ³²). Instead, commit to openness in your DNA.
- **Align Your Team Incentives with Open Success:** Sales and marketing teams in an open-source firm should be educated that the product's free availability is a feature, not a bug. Rather than hiding or downplaying the open-source project in hopes of selling an enterprise version, use the project's popularity as your primary funnel. Sales can focus on converting happy users into paying customers for extras like support or hosting. This may involve more consultative selling (identify pain points you can solve) rather than classic license sales. Structuring sales compensation around subscriptions (for services) rather than one-off license deals is important for consistency.
- **Invest in Non-Functional Requirements:** As mentioned, open communities often focus on features and innovation, but enterprise readiness involves security hardening, scalability testing, performance tuning, and polish (UI/UX). A portion of your commercial revenue should be funneled into these aspects of the open product. This could mean funding dedicated QA engineers, security audits, automated test infrastructure, etc. High quality is a huge differentiator when users compare an open product to proprietary peers. If your open product is rock-solid and well-documented, it will win trust over a flakier closed competitor. Remember, your paying customers are effectively subsidizing improvements for all users – this is good, as it grows the pie. But make sure to communicate improvements and trumpet the reliability of your open solution.
- **Offer Commercial Enhancements that Don't Fork the Code:** If you want to build premium offerings, try to do so in a way that complements the open core without bifurcating it. For instance, provide a *hosted service with extra convenience* (backup, monitoring dashboards, etc.) or proprietary *content* (not code) such as certified datasets, AI model packs, etc., if that makes sense. These add value but don't create a separate version of the software. Another example: provide an enterprise installer or management console that uses the same open APIs under the hood – technically a separate tool, but not a different version of the core product. The goal is that your customers can always fall back to the open source project itself; you're just making

their life easier. If you do develop any such tool privately initially, consider open-sourcing it once it matures and rolling it into the main project, to stay true to the “100% open” promise (like Stability AI did by open-sourcing their UI ¹⁹).

- **Build a Community Ecosystem (Plugins, Integrations, Partners):** Encourage third parties to build on your open platform. If you have an AI model, provide SDKs and libraries (again open source) for others to embed it in applications. If your software can be extended, maintain a plugin architecture and gallery. By enabling a rich ecosystem, you increase your solution’s value without doing all the work. This also creates potential partners who might co-sell services or include your technology in bigger deals. An open-source project that becomes an industry standard (or a de facto component in many stacks) gains immense leverage for monetization down the line.
- **Engage with Open Source Program Offices (OSPOs):** Many large enterprises have OSPOs that manage their use of and contribution to open source. These can be your allies in adoption. Make it easy for companies to adopt your open-source tool – clarify licensing, have a contributor license agreement process if needed, and actively reach out to OSPOs at companies that heavily use your software for partnership opportunities. They might sponsor development of features they need (i.e., fund your team to build something in the open) or become major support customers if they decide to standardize on your tool.
- **Be Prepared for Forks – and Handle Them Wisely:** When everything is open, it’s possible (indeed easy) for others to fork your project and create their own version or even a competitor. This is often cited as a fear of going fully open. The reality is, forks will only gain traction if you fail to serve the community or customers well. If you keep your quality high and your community inclusive, forks are usually short-lived or niche. In some cases, forks can even help (they might explore a different direction that you later merge back). The database company MariaDB forked from MySQL when Oracle took over MySQL – a reaction to fears of MySQL’s openness ending. If your behavior is the opposite (i.e. steadfast openness), you’re more likely to *attract* contributors from would-be forks into your fold (as Red Hat did by hiring key CentOS contributors at one point ³³). However, if a fork does gain steam, consider whether it’s addressing a legitimate need. Perhaps they wanted a lighter version, or different license, etc. Learn from that and even collaborate if possible. Fighting forks with hostility rarely works; out-innovating them in the open is the best strategy.
- **Marketing the Right Message:** Position your open-source nature as an advantage to customers: security (anyone can audit our code, and many eyes have vetted it), flexibility (you have full control, no black boxes), and community (you’re not dependent on a single vendor’s roadmap – a whole ecosystem supports this). Especially for AI, concerns about transparency and ethics are growing; being open source allows you to say “we have nothing to hide – our model weights and code can be inspected for biases or flaws.” This can be a selling point in regulated industries. Make sure your sales and marketing materials highlight stories of customers who benefited from the open aspects (e.g., a client that extended your open-source model in a new way to fit their business).

By following these practices, a company can mitigate the risks and maximize the upsides of an open-first strategy. The overarching theme is to **embrace openness not just in license, but in mindset** – engage, collaborate, and focus on delivering real value beyond the code. As many of the examples showed, the companies that succeed with open source treat their users and community as extensions of the team, not outsiders to be fenced off. When done right, this creates a strong foundation for long-term success.

7. Conclusion

The landscape of generative AI and enterprise software is evolving rapidly, and the old playbook of proprietary control is no longer the sure path to success it once was. This paper has presented an alternative playbook: **build your company on a 100% open-source foundation, and center your business model on customer-centric services and execution excellence.** We've argued that in a world where technology is increasingly commoditized and reproducible, true differentiation comes from how you deliver solutions, not from hiding the implementation. Open source is not just a licensing model; it's a strategic enabler that can catalyze community-driven innovation, engender user trust, and open doors to flexible deployment and integration opportunities that closed-source companies struggle to access.

By removing the ambiguity of a dual personality (open vs. closed) and committing fully to openness, a company frees itself from internal conflicts and gains credibility. It can then focus on the hard but rewarding work of delivering value that users will pay for: making the product easier to use, more reliable, better integrated, and offering expertise and convenience that save customers time and effort. The examples of Red Hat, SUSE, Nextcloud, Zabbix, Stability AI, and others show that this model can be successful across different scales and domains. As SUSE's CEO put it, when you stop trying to make money from owning code and instead sell "a tangible service and value" around that code, you align your interests with your customers' and spur greater innovation rather than stifling it ³.

For GenAI startups, specifically, an open-source-first strategy could be the key to standing out in a crowded field and ensuring longevity. The AI community is one that deeply values open collaboration – much of the progress in AI has come from openly published research and shared code. By contributing your generative models or AI tools to that commons, you not only do a service to the community but also set yourself up as a leader who can harness collective advancements. You may give up short-term licensing revenue, but you gain adaptation and adoption, which are the lifeblood for an AI system to improve and become ubiquitous. And ubiquity can translate to substantial revenue when monetized thoughtfully (through support, cloud services, etc., as we discussed).

Of course, going 100% open source is not without challenges. It requires careful execution, a clear value proposition for revenue, and often a mindset shift for investors and team members accustomed to traditional models. However, as this paper has articulated, those challenges can be overcome with the right strategy and have been overcome by others before. The result can be a company that is both ethically aligned with the open-source ethos and economically successful – a company that drives innovation while also sustaining itself and rewarding its stakeholders. In conclusion, we assert that **an open-source-centric business model is not just a moral or ideological choice, but a pragmatic strategy** especially suited to the current era of tech. It flips the script: instead of limiting what users can do with your software, enable them freely – and then build the kinds of value on top that they'll willingly pay for. We believe this approach can lead to more robust, secure, and innovative AI solutions, and foster a healthier relationship between technology firms and the communities they serve. It's a strategy where everyone wins: developers, customers, and the company. As the generative AI revolution unfolds, those companies that embrace openness may very well be the ones that shape the future and define the new standards of our industry.

References: (sources cited inline above)

- [4] Karthik Ranganathan, "*What does it take to commit to 100% open source?*", *SD Times*, Feb 27, 2020.

- [5]** Karthik Ranganathan, "Why We Changed YugabyteDB Licensing to 100% Open Source", *Yugabyte Blog*, July 16, 2019. 7 34
- [7]** Devansh, "Understanding the Business of Open Source Software and AI", *Medium.com*, Oct 2023. 1 2
- [8]** Nextcloud Website, "Our Values - Open Source", accessed 2025. 28
- [9]** Zabbix Press Release, "Zabbix Founder... confirms profitable 100% open-source model", Oct 2025. 29 35
- [11]** Sid Sijbrandij, "The Red Hat model only worked for Red Hat", *Open Core Ventures Blog*, Apr 14, 2023. 23 13
- [13]** Evan Kirstel, *Interview with SUSE CEO Dirk-Peter van Leeuwen*, *LinkedIn Pulse*, Aug 15, 2025. 9 3
- [15]** Kyle Wiggers, "Stability AI open sources its AI-powered design studio", *TechCrunch*, May 18, 2023. 18 19
-

1 2 30 31 Understanding the Business of Open Source Software and AI | by Devansh | Medium
<https://machine-learning-made-simple.medium.com/understanding-the-business-of-open-source-software-and-ai-0aa43a480450>

3 6 9 25 How SUSE Is Shaping the Future of Open Source, AI, and Digital Sovereignty
<https://www.linkedin.com/pulse/how-suse-shaping-future-open-source-ai-digital-sovereignty-evan-kirstel-s0cpf>

4 5 8 10 11 17 20 21 What does it take to commit to 100% open source? - SD Times
<https://sdtimes.com/open-source/what-does-it-take-to-commit-to-100-open-source/>

7 32 34 Why We Changed YugabyteDB Licensing to 100% Open Source | Yugabyte
<https://www.yugabyte.com/blog/why-we-changed-yugabyte-db-licensing-to-100-open-source/>

12 13 15 16 23 24 26 27 33 The Red Hat model only worked for Red Hat | Open Core Ventures
<https://www.opencoreventures.com/blog/the-red-hat-model-only-worked-for-red-hat>

14 22 Business models for open-source software - Wikipedia
https://en.wikipedia.org/wiki/Business_models_for_open-source_software

18 19 Stability AI open sources its AI-powered design studio | TechCrunch
<https://techcrunch.com/2023/05/18/stability-ai-open-sources-its-ai-powered-design-studio/>

28 About Nextcloud
<https://nextcloud.com/about/>

29 35 "Staying free and open source": Zabbix Founder confirms at Zabbix Conference Latam 2025 that a profitable business model is possible
<https://www.zabbix.com/pr/pr682>