PolyAPI

Internal Development Platforms

Build or Partner?

Darko Vukovic • Founder & CEO

PolyAPI

PolyAPI

Introduction

Internal Development Platforms (IDPs) are essential for modern enterprises, but no off-the-shelf solution fully meets their unique needs. Midsize companies may adopt more flexible options, but they often struggle with the high costs of customizing and integrating cloud platforms to suit their needs. While these companies may be able to experiment with various solutions, the resources required to maintain and scale them can drain their budgets and hinder innovation. Large enterprises face even more complexity—they need platforms that support both internal development and partner ecosystems while also serving the diverse needs of various lines of business, each with different stakeholders and requirements. This complexity introduces risk, as one-size-fits-all solutions often fall short of aligning with enterprise-wide objectives.

Building an IDP without the right skills, capital, or realistic timelines often leads to failure. Partnering with systems integrators (SIs) can result in expensive, custom-built platforms that miss the mark due to misaligned priorities or incomplete integration. Another option is partnering with a large cloud provider, which can be costly and create a deep vendor lock-in, limiting future flexibility. Alternatively, assembling multiple partner solutions can lead to a fragmented, hard-to-manage “Frankenstein” system, which becomes inefficient as the business scales.

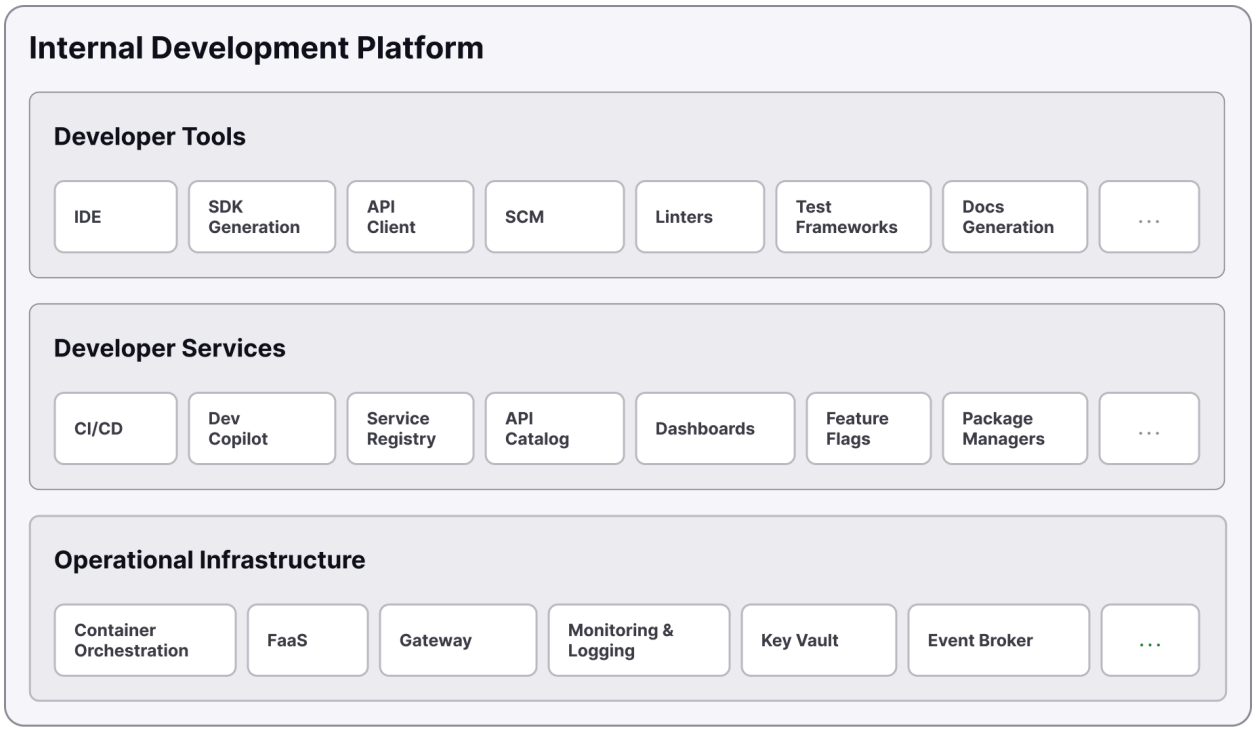
Many enterprises also fall into the sunk cost fallacy, continuing to invest in a failing platform because of the resources already spent, believing that success is just around the corner. With each additional investment, the cost of pivoting increases, creating a cycle of diminishing returns. This whitepaper will help you understand the choices available and the pros and cons of each, and it will recommend a new path forward in IDP development, allowing your organization to take complete control of its development environment and future-proof its technology stack.

PolyAPI

What is an IDP?

An Internal Development Platform (IDP) is a combination of technology and an opinionated methodology that defines how an enterprise designs, develops, and operates custom-built applications, integrations, composite applications (orchestrations), and services (including microservices). It incorporates the architectural decisions architects and leaders make, encompassing infrastructure technologies, cloud providers, and best practices optimized for security, scalability, and quality outputs.

IDPs are designed to maximize developer productivity by offering a comprehensive suite of tools and services. This includes development tools, developer services, and operational infra to standardize the development environment. IDPs often include decisions around programming languages, libraries, and frameworks, allowing developers to work within consistent and scalable environments.



The diagram above provides a high-level overview of the components commonly found in an IDP. It demonstrates how developer tools, services, and operational infrastructure integrate to create a cohesive internal and external development platform.

More advanced IDPs go beyond internal development needs by supporting partner developers and integrating with external ecosystems. These platforms enable collaboration both within the enterprise and with external partners, making them invaluable for enterprises looking to succeed in dynamic and competitive markets.

Build, Partner, or Hybrid?

No one is doing 100% build or 100% partner for IDPs. The real question is whether you're mostly building, mostly relying on partners, or taking a balanced approach. I've met top-tier companies that have embraced each strategy.

For DIY approaches, the costs can be enormous, ranging from maintaining a 10-person team to spending $13 million a year. Development progress is often slow, and these platforms are labeled "next-generation" years into development—code for "not happening anytime soon." The sunk cost fallacy is standard, with teams stuck in salvage mode after the original initiators are gone.

In the "mostly partner" camp, there are two groups:

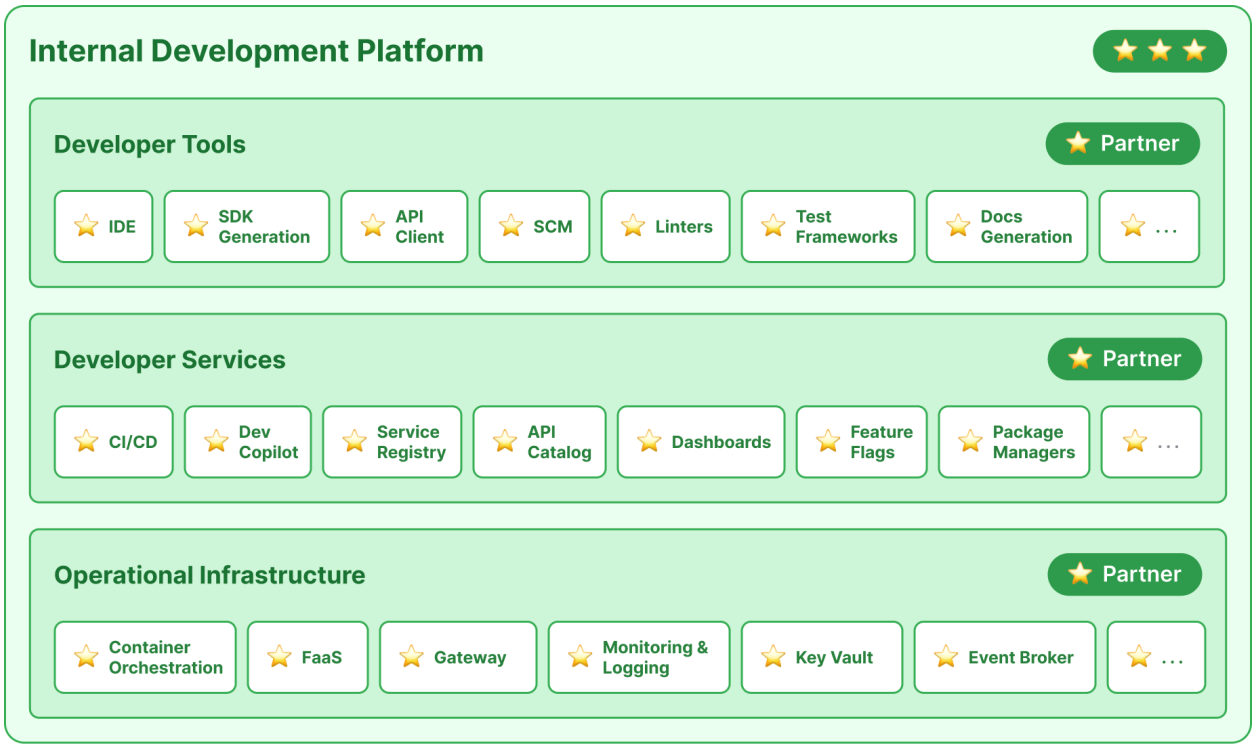
1. Companies trying to integrate over a dozen tools, often completing only one or two integrations due to the prohibitive cost and time, are hoping for a unifying solution like Backstage.

2. Companies rely on a single provider, or SI, for the full stack. While fewer companies still follow this path, a cohort of IT leaders still clings to it.

A "Hybrid" balanced approach typically involves partnering with a top-three cloud provider for the base platform and extending it with best-of-breed solutions and in-house services. However, this often results in tight coupling with that cloud provider.
Below, I explore each of these four choices in more detail.

Partner For The Whole Platform

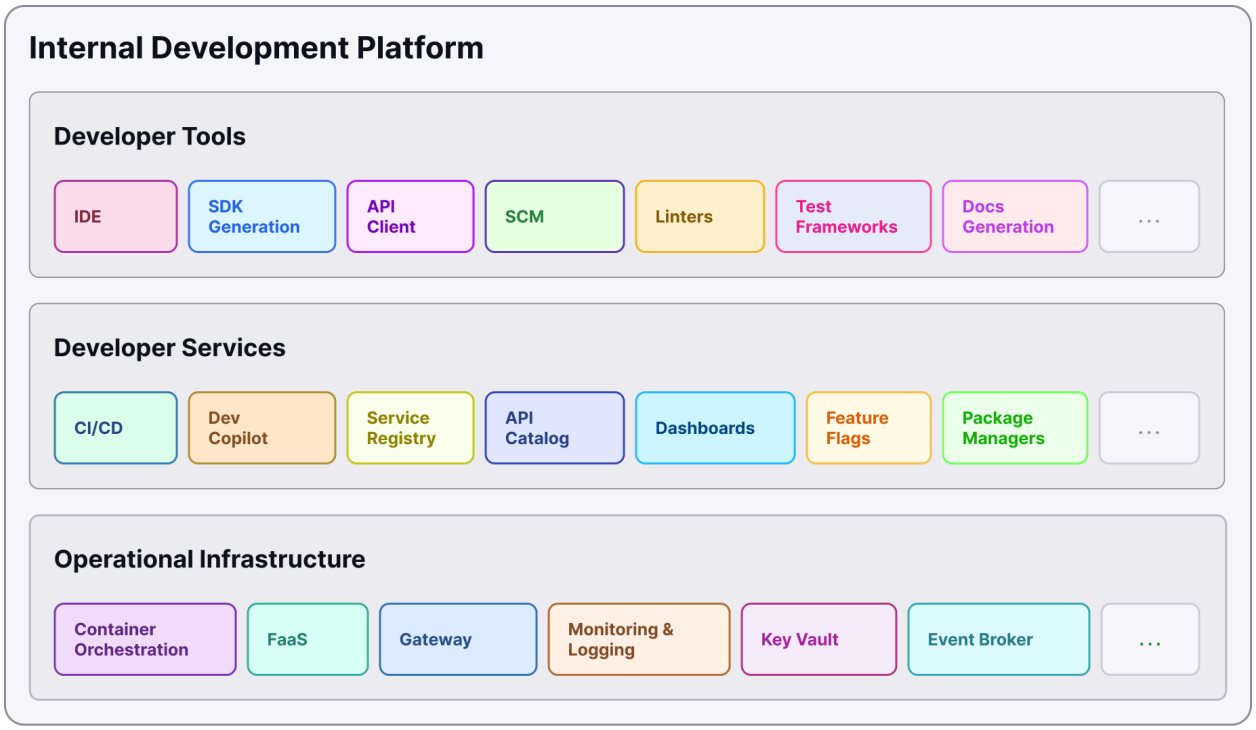
In this scenario, a single vendor or systems integrator (SI) pre-integrates everything to your specifications. The entire platform is delivered as a service, with your only responsibilities being contracting, training, and ongoing operations (which can also be outsourced). The diagram below illustrates this setup:



And this is where we wake up—it's just a dream (or an ambitious sales pitch) that has yet to be successfully realized in practice. Moving on to the second option.

Partner for Components and Assemble Yourself

This approach, often called Best-of-Breed (optimists) or Frankenstein (pessimists), is where most "IDPs" are today. While it avoids a tight dependency on a single cloud provider and doesn't require massive investment, it's still a complex model. Here's what it looks like visually:



Despite the colorful diagram, this setup isn't fun to develop or manage, especially under tight deadlines and high stakes. In large enterprises, some boxes may represent multiple products from different providers. Integrating all these components into a seamless developer experience is a near-impossible task from an economic standpoint.

Build The Whole Platform Yourself

Some CTOs attempt to build the entire platform rather than integrating multiple partner solutions. This approach often becomes an even more complex version of the Best-of-Breed model. While many quickly realize the challenges, some turn to Open Source Software (OSS) as a solution, hoping to reduce costs and gain more control. However, even with OSS, the project scope often becomes too large for the team to handle, draining budget and talent.

In the best-case scenario, the project may result in a smaller, successful initiative but at a high opportunity cost. The team remains focused on maintaining and expanding the platform, which can divert attention from delivering higher-value business capabilities.

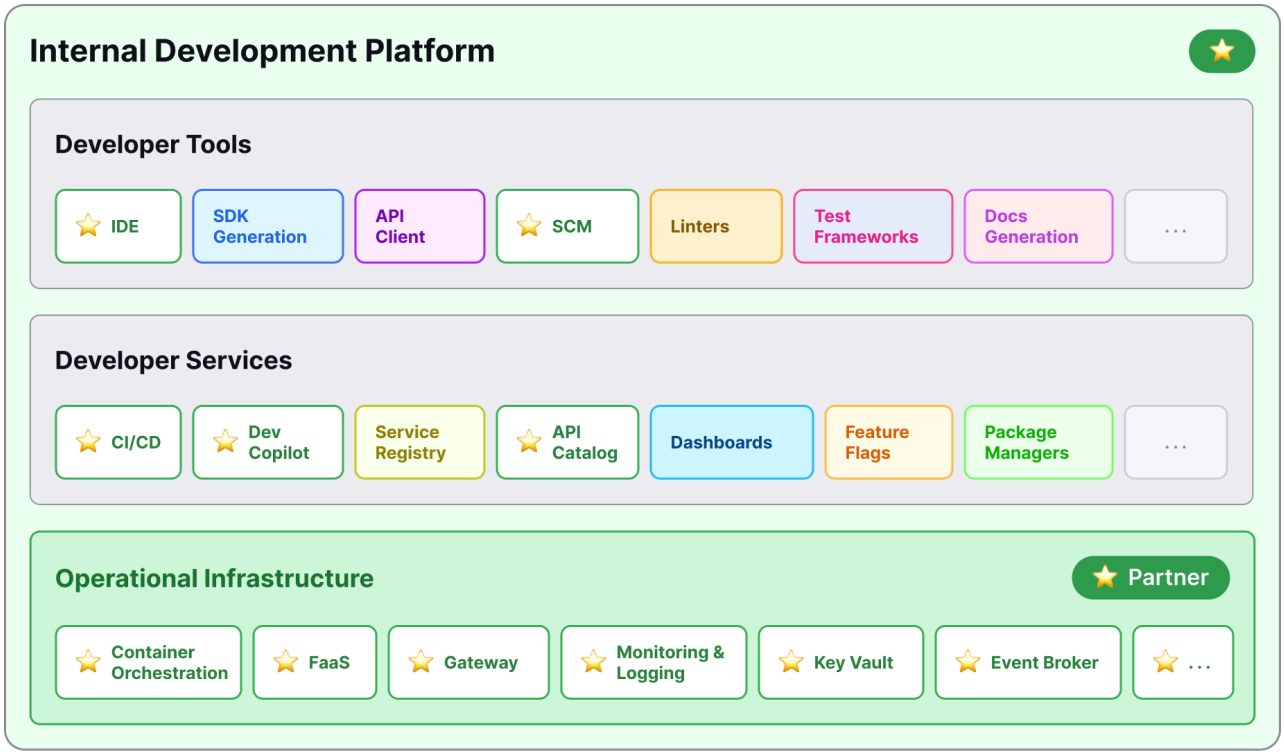
Additionally, they may miss out on established partners' advanced features and support.

More commonly, these projects drag on, missing deadlines and falling short of the original vision. The goal of building a “next-generation platform” fades, and leadership often changes hands, with each successor facing the same challenges, perpetuating the cycle.

Even when the project succeeds in a limited way, the broader opportunity is missed: working with trusted partners would have delivered more capabilities, faster results, and significantly reduced risk.

Partner For Operational Infra and Assemble or Build the Rest

This “Hybrid” option involves partnering with a provider for the “operational runtime infrastructure” while building or integrating components on top. Here's what it looks like:



Some tools from the operational infrastructure partner may overlap with Dev Tools and Dev Services, but you can integrate best-of-breed options. While this hybrid model offers strong support, it often locks you into a specific cloud provider like AWS, Azure, or GCP.

Trade-offs include vendor lock-in, high migration costs, and reliance on their pricing. Despite these downsides, it provides more flexibility and control than building everything in-house but carries the risk of long-term dependence on the provider's ecosystem.

Comparison Summary

Below is a short summary of the four choices with their pros and cons:

Option	Pros	Cons
One Partner & SI Solution	Simplifies management with a **single point of contact**. **End-to-end service** delivery with minimal internal overhead. **Lower risk of integration failures** since one vendor handles everything.	**Unlikely to be realized** in practice—often more of a sales pitch. **Limited flexibility** as you're tied to a single vendor's ecosystem. Risk of **vendor lock-in** and difficulty adapting to future needs.
Partner for Components and Assemble it Yourself	Ability to choose the **best individual tools** for each component. Potentially **lower upfront costs** since you only pay for what you need. **More control** over the platform architecture.	**Integration across multiple** tools is difficult and time-consuming. High complexity in **managing multiple vendors** and technologies. Requires **significant in-house expertise** and ongoing effort to maintain.
Built It All Yourself	**Full control** over the platform—tailored exactly to your needs. **No vendor dependency**, avoiding long-term subscription costs. Can **optimize for unique internal processes** and technical preferences.	Extremely resource-intensive with **extreme costs for labor** and talent. **Very long development timelines**, with potential delays and failures. Requires **specialized skills and deep domain expertise**, which are hard to find.
Partner for Ops Infra & Partner for Rest (Hybrid)	Leverage **stable cloud infrastructure** while integrating specialized tools. **Simplified infrastructure management** with the ability to innovate on higher-level services. Balances control with **outsourcing operational complexity**.	**High costs** associated with cloud provider development services. Risk of **Cloud vendor lock-in** with the chosen cloud provider. **Slower innovation** cycles for features and services not directly controlled by you.

PolyAPI

The Ideal Choice

The "**Partner For Operational Infra and Assemble the Rest (Hybrid)**" model is the best. It delivers the most results quickly while allowing you to integrate best-of-breed tools into the core platform. However, I believe an ideal partner should go beyond the options the big three cloud providers provide. Below is my recommendation for the qualifications a perfect partner should meet.

Minimum Capabilities

1. **Container Orchestration** – Manages containers' deployment, scaling, and operation in a distributed environment. Key capabilities include automatic scaling, load balancing across containers, rolling updates and rollback, and self-healing.
2. **Functions as a Service (FaaS) Runtime** – A serverless model for deploying functions triggered by events without managing infrastructure. Key capabilities include event-driven execution, automatic scaling based on demand, stateless execution, and a pay-as-you-go pricing model.
3. **API Gateways (Inbound and Outbound)** – Manages, routes, and secures API traffic for internal and external services. Key capabilities include authentication and authorization, rate limiting and throttling, load balancing and caching, and API monitoring and logging.
4. **Monitoring & Logging** – Tracks the performance, health, and security of containers and endpoints. Key capabilities include real-time metrics, centralized logging, alerts and notifications, and integration with observability tools.
5. **Keyvault for Secrets** – Securely manages sensitive information like API keys and passwords. Key capabilities include secure storage and access control, automated secret rotation, secure secret injection, and auditing/monitoring of secret usage.
6. **Event Broker** – Facilitates event-driven architectures by allowing services to publish/subscribe to events. Key capabilities include event routing and message queuing, high availability and fault tolerance, support for multiple messaging patterns, and scalability for handling large event volumes.

Native Development

The ideal platform should natively integrate with developer tools and services an IDP should offer. Key considerations include:

1. **Native Programming Language** – The platform should support popular languages like JavaScript/TypeScript, Python, Java, C#, and Go. This ensures developers can use familiar languages, speeding up adoption and integration without compatibility issues.
2. **Local Execution and Testing** – Developers should be able to run and debug code locally before deployment, reducing feedback loops and avoiding cloud-based testing delays.
3. **Native IDE Support** – The platform should integrate seamlessly with popular IDEs like Visual Studio Code, IntelliJ IDEA, Eclipse, and PyCharm, allowing real-time debugging and direct deployment in preferred environments.
4. **Unbounded Library Support** – The platform should allow unrestricted access to third-party libraries, avoiding limitations seen in cloud provider services, enabling more complex applications without packaging constraints.

Cloud Agnostic and Kubernetes Native

The ideal platform should be **Kubernetes-native** with no cloud-specific dependencies, certified across major providers like AWS, Azure, Google Cloud, and Oracle. Key capabilities like container orchestration, API gateways, event brokering, and monitoring should operate entirely within the Kubernetes cluster.

The platform must allow seamless workload shifting between cloud providers without impacting clients, enabling flexibility to optimize cost, performance, or availability while maintaining consistent operations.

It should be available as a PaaS in critical regions, offer self-hosting options, and include managed services for operational efficiency. With multi-cloud and multi-region support, it provides flexibility, resilience, and performance while adhering to open standards and compliance.

Integrations and Standards

The ideal platform should support essential **open standards** to ensure broad compatibility, flexibility, and seamless integration with third-party systems. These include:

1. **OpenAPI Specification** – Defines RESTful APIs in a consistent, language-agnostic way, simplifying API design, documentation, and integration.

2. **JSON Schema** – Ensures structured and validated data exchange using JSON, which is essential for consistency across distributed systems.
3. **OAuth 2.0 / JWT** – Provides secure API access and token-based authentication, critical for managing identity and authorization.
4. **SCIM** – Automates user provisioning and management across services, ideal for streamlining identity across platforms.
5. **Vault Integration Protocols** – Supports integration with secret management systems like HashiCorp Vault for securely handling secrets, credentials, and environment variables through established plugins and APIs.
6. **AsyncAPI** – Facilitates asynchronous, event-driven communication for microservices and IoT systems, promoting real-time integration.
7. **CloudEvents** – A standard for event data across cloud services, ensuring interoperability in event-driven architectures.

The platform supports these standards and ensures **future-proofing**, flexibility, and seamless integrations in multi-cloud environments.

Decommissioning and Transition

The platform should be designed to continue operating in a self-hosted environment if licensing is ended, preserving the full implementation logic with minimal modifications. While some services or integrations may need rebuilding to replace proprietary components, core business logic and workflows should remain intact. The platform must ensure all data is exportable, supporting open formats to allow smooth transitions to other solutions.

Solution Engineering & Support

The ideal partner should offer the platform as a fully managed service with 24/7 support and strict SLAs, ensuring reliability, uptime, and quick issue resolution. This guarantees businesses can depend on the platform without service disruptions, with dedicated teams available around the clock.

In addition to support, the partner should provide solution engineering services directly or through certified partners with fixed-bid project implementations to ensure predictable costs and timelines. This ensures smooth platform deployment, optimized integrations, and configurations tailored to your needs.

The combination of proactive support and solution engineering allows customers to maintain operations and continuously adapt the platform to evolving business needs.

Additional Capabilities

Any capabilities within the **Developer Tools** or **Developer Services** categories are valuable bonuses if they are pre-integrated and supported directly by the partner or through other vendors within the partner's ecosystem. These additional features reduce the need for custom integrations and streamline developer workflows. Key capabilities include but are not limited to:

Developer Tools

- **Integrated Development Environment (IDE)**: A powerful, well-supported IDE is essential for efficient coding and debugging, providing a seamless development experience.
- **SDK Generation**: Automatically generate SDKs for various programming languages, making it easier for developers to interact with APIs.
- **API Client**: A pre-configured API client tool for testing and interacting with APIs during development.
- **Source Code Management (SCM)**: Tools like Git for version control to ensure collaborative and trackable code management.
- **Linters**: Automated tools that analyze code to ensure adherence to style guidelines and catch bugs early.
- **Test Frameworks**: Built-in support for testing frameworks, enabling automated unit, integration, and functional tests.
- **Docs Generation**: Tools to automatically generate documentation for APIs, SDKs, and codebases.

Developer Services

- **CI/CD Pipelines**: Continuous Integration and Continuous Deployment pipelines to automate testing, building, and deployment of applications.
- **Dev Copilot**: AI-assisted development tools to help developers with code suggestions, completions, and best practices.
- **Service Registry**: A centralized registry for managing and discovering services within the platform.

- **API Catalog**: A comprehensive catalog that documents all APIs available in the platform, ensuring discoverability and consistent usage.
- **Dashboards**: Monitoring and visualization tools that provide real-time insights into system performance and usage metrics.
- **Feature Flags**: Built-in support for feature flagging, allowing developers to enable or disable features in production without redeploying.
- **Package Managers**: Integration with popular package managers (e.g., npm, pip, Maven) for managing dependencies across projects.

These additional capabilities can significantly enhance developer productivity and ensure that the platform integrates smoothly with existing development processes. By offering these tools out-of-the-box, the platform helps reduce the time and effort required to set up and manage these essential services.

PolyAPI: The Partner For You?

At PolyAPI, we strive to be the ideal partner for enterprises building integrations, orchestrations, and backend services. Whether you refer to it as an Internal Development Platform, our platform empowers your organization to thrive. We believe we are the right choice for your needs if the following questions align with your goals.

We continually enhance our platform, driven by customer feedback and a commitment to delivering more excellent value. From adding new features to integrating the latest technologies, we ensure our platform evolves to meet the demands of modern enterprises.

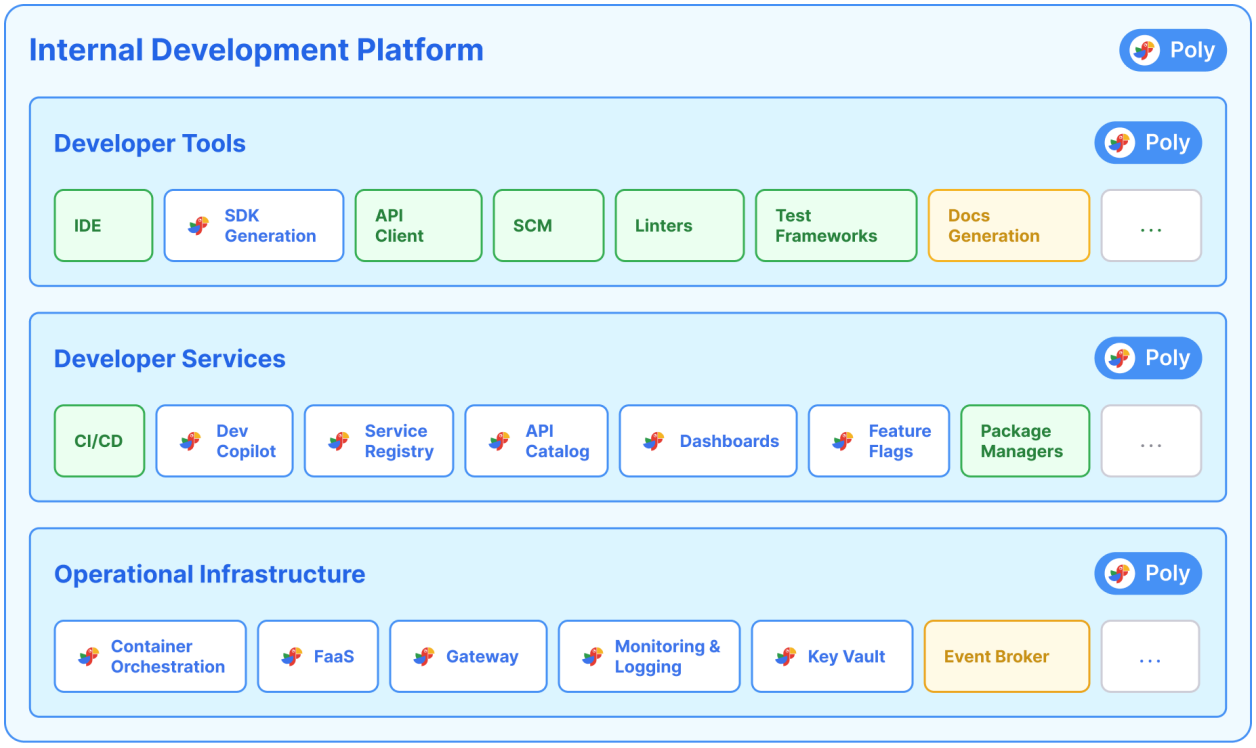
Consider the questions below, and if you answer yes to all of them (or are willing to be a bit flexible), let's start a conversation.

Ask Yourself:

1. Does your enterprise **need** to build and operate **numerous** integrations, orchestrations, microservices, or internal applications?
2. Do you **prefer partnering** with a platform provider for your **Core Operational Infrastructure** rather than building it in-house?
3. Is it essential for your IDP to be **cloud-agnostic** or **self-hosted** in your Kubernetes environment?
4. Is your organization comfortable partnering with a **growth-stage startup** for critical infrastructure like your IDP?
5. Are you an enterprise in **Retail, Hospitality, Food & Beverage, or Logistics**? If not, are you open to being PolyAPI's **first customer** in your industry?
6. Are you in the **early stages** of developing your IDP, or will you reconsider your current approach if it's **not yielding** the desired results?

If you answered **yes** to these questions, PolyAPI would be a **great partner** for your needs, helping bring tremendous value to your enterprise. And even if some of your answers are "no," that's okay—please keep us in mind as your needs evolve.

If you're ready to explore how we can help, here's a summary of **what we offer today** (blue), where we integrate seamlessly out-of-the-box (green), and the areas that are part of **our vision** but not yet realized (yellow).



Extending Poly

Poly is built to be flexible, knowing that no single platform will meet 100% of your needs out of the box. While we integrate with many popular tools, we understand that you may prefer others we don't support natively. That's why we've embraced an API-First approach, making Poly highly extensible and automatable to adapt to your requirements.

With the time saved using Poly, you have several options: extend Poly by developing custom tools and services, integrate your preferred tools from other partners, focus on solving business challenges with the platform, or redirect the savings to other strategic initiatives. Whatever path you choose, we offer the flexibility and support to ensure your success.

Conclusion

This paper aims to clarify the complexities and critical decisions involved in building or partnering on an **internal development platform**. We've discussed the trade-offs between these approaches, outlined the benefits of a hybrid model, and offered a clear framework to help you evaluate the right path forward for your organization. Additionally, we've showcased how PolyAPI's unique strengths can help you overcome the challenges of building integrations, orchestrations, and microservices—delivering the flexibility and scalability modern enterprises demand.

Whether you're just beginning to develop an IDP or reevaluating an existing approach, PolyAPI is uniquely positioned to support your success. Our platform is designed to help you achieve faster time-to-market, reduce operational complexity, and unlock new capabilities for your enterprise. If the considerations outlined here align with your vision, we would be excited to partner with you and accelerate your progress.

We value your feedback as we continuously refine our platform and approach to deliver even more excellent results. We welcome the opportunity to connect if you're ready to explore how PolyAPI can help your enterprise achieve its goals or simply wish to discuss your specific needs.

Please feel free to reach out at hello@polyapi.io to start the conversation.

Thank you for your time and consideration.
Darko Vukovic
CEO & Founder, PolyAPI

PolyAPI

About the Author

Darko Vukovic

CEO & Founder, PolyAPI

Darko Vukovic is the Founder and CEO of **PolyAPI**, a powerful platform designed for **integration**, **orchestration**, and **microservices** development in enterprise environments. Darko has a deep background in **API management**, **composite application development**, and **integration development**, having built enterprise platforms that enable organizations to streamline their operations and accelerate innovation.

Throughout his career, Darko has worked extensively with major platforms like **MuleSoft**, **Oracle**, and **Google Apigee**, driving the development of large-scale solutions for **hospitality**, **eCommerce**, and **technology** sectors. His hands-on experience in delivering robust, scalable enterprise platforms has helped businesses tackle complex integration challenges and unlock new opportunities for growth.

As a technology leader, Darko is passionate about creating solutions that empower enterprises to scale their systems with speed and efficiency. Under his leadership, PolyAPI is becoming a go-to partner for enterprises seeking to modernize their infrastructure and gain a competitive edge through seamless integration and microservices development.