What Makes Poly So Special?

Since the inception of Poly, it's been a challenge to succinctly describe what Poly is to people interested in evaluating it for their integration, API, and microservices development needs, particularly for non-technical audiences. That's our fault. **This short paper articulates the essence of Poly.** Remember that there is much more to the platform than is described here. If, after reading this paper, the details outlined here make sense, you will have an excellent foundational understanding of what Poly is. You will also understand how a new platform is tackling 20-year-old incumbents seemingly overnight.

If after reading this paper you would like to learn more, take a look at our video series to see Poly in action:

1. Building Integrations
2. Building Microservices
3. Innovation Series

Enjoy!

AUTHOR: DARKO VUKOVIC
PUBLICATION DATE: 3/8/24

Foundational Philosophy

Before getting into how we built Poly, it's essential to understand the thinking that went into the foundational design principles of Poly. If this section is well understood, subsequent sections will be much more manageable to rationalize and understand.

Egos and Integrations

The most significant architectural breakthrough in designing Poly came from the sacrifice of our ego. Every platform always wants to be the center of its respective universe. By definition, integration is the space between different applications and services. Hence, **an integration platform that aims to be at the center of everything is an arrogant oxymoron.**

We built an integration platform by creating as little as possible. Instead, we decided to take an approach where we integrated all the best technologies to work together and only to fill the gaps between them where either they lacked a seamless integration or needed more capabilities.

If you find yourself grappling with the concept of Poly, that's completely understandable. Our approach is unconventional: Poly isn't a platform with a traditional core. Instead, it acts as the invisible yet essential bond that unites various existing services, which we will explore in the next section. We invite you to zoom out and appreciate the unique beauty of this model.

David and Michelangelo

Consider the story of Michelangelo and his creation of David. Legend has it that when asked how he crafted such a masterpiece, Michelangelo reputedly said, "I saw the angel in the marble and carved until I set him free." This resonates deeply with our philosophy in developing integrations. In the realm of software, there's the crucial task of creating integrations, microservices, and orchestration, and then there's the non-essential, unproductive work surrounding it. **Our goal with Poly is to focus intently on the essential – the 'angel' in the software – and to remove the extraneous waste of time and effort systematically.**

Gripping Poly is also challenging in part because It is a strongly opinionated development approach developed from decades of accumulated experience, industry best practices, and the necessary tools and services to implement this approach effectively.

Inheritance

In this section, we'll explore the core technologies that form the foundation of Poly, illustrating how each contributes to the platform's capabilities. **If you are only interested in the capabilities developed by Poly, please skip to the next section.** By virtue of the architectural strategy to ensure as perfect alignment with modern enterprise architecture as possible, we selected a set of platforms, applications and tools as the foundational basis of Poly. These technologies are a starting point on top of which we developed additional new services. Each of these technologies has potentially multiple alternatives which are both good and popular, so over time we aim to add support for more and more components to ensure we cover more and more combinations used by our customers.

Leveraging these established technologies has provided Poly with a significant developmental advantage, akin to beginning a marathon at mile 20. Often industry experts are amazed with how mature and robust Poly already is, and we have to give credit to the millions of people who have cumulatively contributed to the underlying technologies we leverage.

One correct definition of Poly is: **an integration of the most powerful development tools and technologies plus an additional Poly built services to specialize this composite platform for enterprise integration, microservice, and orchestration development.** We selected these technologies based on over 200 interviews we did market research in this domain. Below is the list of explicit technologies used.

Kubernetes & Tools
Poly is Kubernetes native. Whether run in AWS, Azure, GCP; operated by us, partners, or customers; Poly runs in Kubernetes with as minimal of a dependency on the cloud as we were able to engineer. Poly is hence cloud agnostic and can be easily self hosted by customers. All tools used to deploy, update, configure, and operate Kubernetes are hence implicitly usable in administering Poly. It also ensures that Poly and all integrations, microservices, orchestrations are infinitely scalable.

KNative
Within Kubernetes, KNative is our out-of-the-box option for running integrations, microservices and orchestrations built in Poly. Where it makes sense, integrations, microservices, and orchestrations should be deployed as serverless functions. With integrations, this is a really nice model as it allows for editing and updating individualized flows without risking other integrations.

Programming Languages – TS/JS, Python, Java

Using native programming languages allows developers to create complex integration and orchestration logic in superior form. While most integrations can be kept to simple code, having higher-order programming capabilities available when needed is essential. It also means that experienced programmers are proficient in developing with Poly on day one.

Poly is unique because the output is still a native application that runs in the runtime environment of each respective language. All generated clients are just native files. No particular runtime, language, compiler, or tooling is needed. Similarly, any additional libraries and SDKs the developer wants to use are also implicitly compatible with Poly. Poly applications can be deployed either as standalone applications or to KNative (within Poly) with no alteration to the code. This provides for a seamless SDLC flow going from mocking to implementation, testing, production, and depreciation. Likewise, all the tooling around intelli-sense, compilation, linting, source management, automated testing, etc... is implicitly usable with all applications leveraging Poly functions. The following language/framework we are adding support for is .NET.

VS Code

VS Code is a free IDE with an excellent extensibility framework. By poly normalizing into standard programming, we leverage all the tools of VS Code for developing, managing, debugging, running, etc., poly applications. VS Code also has a great extensibility framework, which we used to build additional tools for easier API Discovery, consumption, and development. Next in line is to create the same extensions for IntelliJ.

OpenAI

As part of our service, we leverage OpenAI base models and embeddings. OpenAI is a leader in general-purpose AI services. By extending OpenAI's capabilities, Poly implicitly can support developers in quickly generating code and debugging issues. We don't have a set roadmap for additional providers now; this will be customer-driven as customers select their AI providers.

Hashicorp Vault

We selected the open-source Hashicorp vault as the first Vault to support it. It's packaged and delivered with Poly and securely stores and manages credentials and secrets. We leveraged standards to interface with Vault to ensure we can easily add support for customer-supplied vaults in Azure, GCP, and AWS.

PolyAPI

Postgres and Redis

We leverage two prevalent and robust technologies as our persistence and caching layers. These technologies come with capabilities such as auto encryption, cache management, etc.... Poly generally provides these services in an isolated deployment. However, it can also leverage existing instances for customers who already have them provisioned in their cloud environments.

Postman and OpenAPI Specifications

Postman has become the de facto console for developers to try out APIs. OpenAPI specifications are the de facto standard for documenting APIs in a human—and machine-readable format.

Additional Near-Future Technologies

We see opportunities shortly to add new classes of technology to improve the developers' experience further. Specifically, we are looking closely at GitHub Copilot Chat, git-based version control systems, and event subscription systems. The current technologies are just the beginning of implementing our vision.

We saw Poly in these foundational technologies just as Michelangelo saw David in the marble. Our task was skillfully integrating and enhancing them, carving out a platform optimized for enterprise integrations, APIs, microservices, and orchestrations. Our first challenge was incorporating them into a unified deployment architecture on Kubernetes. Second, we had to integrate them to become a unified development platform. The last challenge was to fill in the gaps with missing tools and services (detailed in the next section) needed to create an end-to-end seamless development platform optimized for integrations, APIs, microservices, and orchestrations.

PolyAPI

Additional Poly Services

Beyond integrating the foundational technologies, we developed the following tools and services to multiply the productivity gains with Poly. As each service is as seamlessly integrated as possible, it's hard for an untrained eye to recognize them as new developments. Compounding that challenge is that many of these tools have simplistic interfaces, such as command lines, or they run behind the scenes as services. **With these developed services, we could harness and direct the capabilities of the underlying open-source technologies to unify them into the ideal enterprise integration development platform.**

Catalog

The core data of Poly is where all APIs, custom functions, server functions, webhooks, triggers, jobs, etc... are stored. We also have additional client services and Postman extensions for interpreting OpenAPI Specs, runtime API calls, deploying functions, etc... to ingest and populate this catalog with information about APIs being produced or consumed by an enterprise, microservices, and client functions developed with Poly, event listeners and over time more. None of the other services would be possible without these catalog services and the data within. This service ensures that enterprises accumulate essential information in a centralized repository to ensure no institutional knowledge loss over time.

Proxy

A server that runs and performs critical operations such as translating SDK calls into API requests, receives webhook calls and delivers events via WebSockets and triggers, raises alerts on errors, and more. This service plays an essential runtime role and allows for many developer experience implications, such as the universal SDK, event streaming, and runtime observability.

SDK Generation

A combination of server-side operations and language-specific clients that generate universal SDKs in TS/JS, Java, Python, etc. Generated SDKs can contain API passthrough functions, custom client-side functions (referred to as utility functions), server functions (a la RPC, although they use HTTPS), webhook event handler functions, variable access/reference/update functions, and more. The contents of the SDK can include the entire enterprise catalog or be scoped at the time of generation, and customers can decide to expose the SDK generation services or publish specific versioned SDKs. With

these SDKs, developers can develop confidence in the correct models of the underlying enterprise systems.

AI Agent

Leveraging the catalog via a custom developer RAG orchestration with OpenAI Poly offers a practical “GPT4 Turbo trained on your API” service. It can help find the proper operations and generate highly accurate answers since it has insight into the exact specifics of the API operation, webhook event, or custom function. We chose an RAG model here due to the infinitely scalable architecture, the dynamic nature of the catalog, and the optionality to use other AI providers per customer. The AI agent boosts developer productivity in discovery, development, and troubleshooting.

Serverless Functions

KNative is a container management platform that automatically scales up and down containers. This has two significant benefits: extremely efficient resource utilization and inherent scalability. We extended this service to add deployment management, cataloging, container image generation, log collection, and externalized triggers. Serverless functions drastically increase reusability and allow for faster deployment to production as the runtime environment is pre-approved and implicitly designed for infinite scaling.

Credential & Config Management

Our research found that a ton of time is lost waiting for access to systems, security reviews, and ensuring that security standards are met. We designed “Vari” to be the antidote to this loss of productivity and to help CSIOs sleep better at night. Vari ensures credentials are stored in an encrypted vault and never exposed to developers or client applications (integrations) at runtime. Vari allows for credentials to be modeled and used within code as references. This allows for a fantastic developer experience, secure credentials management, and a more performant authentication flow.

Operational Services

There are too many additional medium-sized features to enumerate here, but please note that in addition to the core functionality described above, there are many tools to further streamline enterprise integration development and operations not listed here that contribute to monitoring, self-remediation, compromise detection, reusability, and efficient development and integrations.

Conclusion

We hope this paper provided you enough of an overview to understand that Poly is not just “another integration platform”. We hope you can now articulate the message that Poly is an integrated development platform, built on top of the most robust enterprise tools, extended with specialized capabilities to turn 8-month integration projects into 8-week projects. If Poly is something you would like to learn more about, please do not hesitate to reach out at info@polyapi.io.

Thank you,

Darko Vukovic
CEO & Founder of PolyAPI